RMC70/150/200 Motion Controllers
And
RMCTools Software
User Manual

Version 4.23.2 February 24, 2023

Condensed Contents

Table of Contents

1. Introducing the RMC Family

RMCTools and RMC Controllers Help

The RMCTools software is for setting up, tuning, programming and troubleshooting the RMC75, RMC150, and RMC200 motion controllers. RMCTools does not support the RMC100, which requires Delta's RMCWin software.

The RMCTools help provides the most complete and up-to-date information available for the RMC75, RMC150, RMC200 and RMCTools. To ensure you have the latest help file, download the latest version of RMCTools from Delta's website https://deltamotion.com.

The RMCTools help is intended to be a reference. For a step-by-step startup procedure, see the Startup Guide that shipped with your controller:

- RMC75 Startup Guide
- RMC150 Startup Guide
- RMC200 Startup Guide

The Startup Guides can also be accessed at deltamotion.com/dloads/.

Getting Started

Step-by-Step Startup Procedure

Video Tutorials

Delta's website offers video tutorials to help you learn the RMCTools software.

The **Education** page of Delta's website at https://deltamotion.com offers video tutorials to help you learn the RMCTools software.

RMCTools Software

RMCTools Overview

RMC Motion Controllers

RMC75
RMC150
RMC200

Features

Controller Features Overview
Programming
Communications

Troubleshooting

See the Troubleshooting topic.

Customer Support

Delta support is ready to assist you. Call if you have questions or need help:

Phone: +1-360-254-8688 (24-hour emergency support available)

Fax: +1-360-254-5435

E-mail: support@deltamotion.com
Website: https://deltamotion.com

Help in PDF Format

A manual containing the same information as the RMCTools help is available as a Portable Document Format (PDF) file from Delta's website https://deltamotion.com.

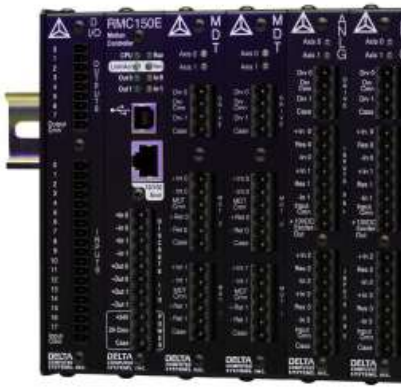
See Also

Disclaimer

1.1. RMC Family Motion Controllers

The RMC family of motion controllers consists of the RMC75, RMC150, RMC200, and the older RMC100. The RMC family brings the benefits of high-performance, easy-to-use motion control to a wide range of industrial applications. The RMCTools software supports the RMC75, RMC150, and RMC200. The legacy RMC100 requires Delta's RMCWin software.

RMC75: 1-2 Axes



RMC150: Up to 8 Axes

RMC200: Up to 50 Axes

The RMC75, RMC150, and RMC200 share nearly all the same control features, and are configured and programmed in the same way via RMCTools. If you know how to set up and configure one controller, you can configure the others just as easily!

To compare the RMC controllers, see the Hardware Overview topic.

For hardware details and specifications see the RMC75, RMC150, and RMC200 topics.

Topics

- Control Features
- RMCTools Software
- Startup
- Communications

1.2. Basics of Operation

The RMC is a high-performance motion controller. It has many advanced features, yet is easy to use and can be set up quickly due to features such as Scale and Offset wizards and Autotuning.

This topic describes the most basic concepts of using the RMC.

Motion

Motion control falls into two camps: *open loop* and *closed loop*.

Open loop means that a control signal (called Control Output in the RMC) is given to the system to move it without regard to the feedback of the system, such as position, velocity, pressure, etc. Open loop control is not very precise.

Closed loop means that the controller uses the feedback to determine how much Control Output to give to the system to move it to the desired position. Closed loop control can be very precise if the system has high-resolution feedback and a good actuator.

The RMC provides both open loop and closed loop control.

Open Loop control is fairly simple. Just issue an Open Loop Rate (10) command to the RMC to move the axis in open loop control. See the **Issuing Commands** section below for details on issuing commands.

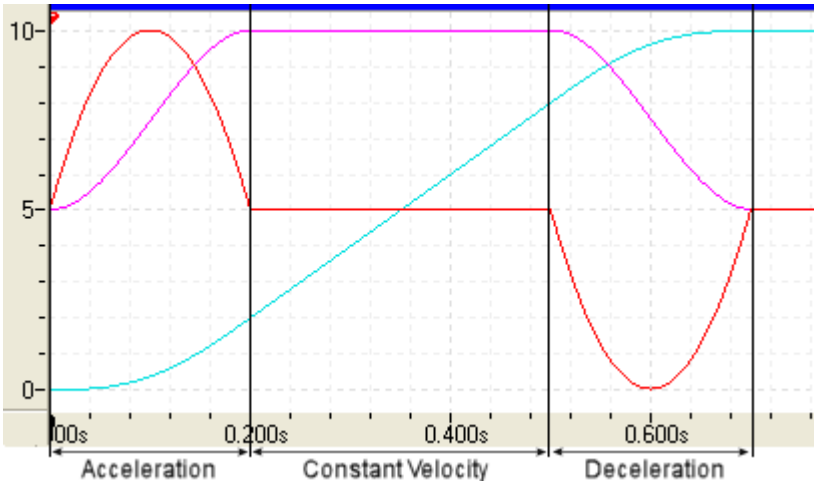
Closed Loop control is more complex. The RMC is all about closed loop motion control. Read on to learn how it works.

Target Profile

When the RMC is commanded to make a move in closed loop control, it generates a *target profile*, which specifies where the position (or velocity, pressure, or force) should be at any given point in time. This profile includes the Target Position, Target Velocity, and Target Acceleration.

For example, if a Move Absolute command is issued to the RMC, it will generate a profile according to the parameters of the command. The plot below shows a typical Move Absolute target profile. The position is moving from zero to ten. The RMC uses rounded accelerations to provide smooth motion.

Legend: — Target Position — Target Velocity — Target Acceleration



PID Algorithm

Generating a target profile determines where the axis should be, but does not make the axis move. In order to move the axis, the RMC uses a PID algorithm that calculates how much Control Output is required to make the Actual Position follow the Target Position. The RMC obtains the Actual Position from the transducer.

The PID algorithm uses several values, called **gains**, that specify how the PID calculates the Control Output. For each axis, in every motion application, these gains must be adjusted to obtain optimal control of the system. This process is called "tuning". RMCTools offers a Tuning Wizard that significantly speeds up the process of tuning. Tuning can also be done manually.

An axis *must* be tuned before the RMC can be controlled in closed loop control.

Making Moves

To make an axis move, issue a motion command. The Issuing Commands section below describes how to issue commands. The RMC has many motion commands for various types of motion. See the Commands Overview topic for a list of commands in the RMC.Motion Commands section of the Command Reference chapter for a complete description of the motion commands in the RMC.

Issuing Commands

To make the RMC do something, such as move, turn on a discrete output, or run a user program, you must issue a command. For example, to move the axis to a position, you would issue a Move Absolute command.

To issue a command from RMCTools, use the Command Tool:

1. On the **View** menu, choose **Command Tool**.
2. In the **Cmd** box of the desired axis, click the ellipsis () button.
3. Browse to the command you need and click **OK**.
4. If the command has parameters, fill them out.
5. Click **Send** to issue the command to the RMC.
6. If an error occurred, or if you wish to see if the command went through, open the Event Log by double-clicking **Event Log** in the Project pane. It shows everything that happened in the RMC.

Commands can also be issued to the RMC from a host controller, such as a PLC or HMI. You can also create user programs in the RMC that issue commands. See the Issuing Commands topic for more details.

Caution: For each command you wish to issue, write it only once so that it is issued only once. If you issue the same motion command (to the same position) multiple times, it can cause the target position to overshoot the requested position.

Setting Up the RMC

Before you can move an axis in closed loop control, you must set up the items listed below. For instructions on setting all these items, see the Startup Guide that came with the RMC. Or, you can read the Startup Procedure topic. Or, you can read the Startup Procedure in the **Starting Up the RMC** chapter.

- **Axes Definitions**
  You must define the axes. For example, will they be position or pressure or some other control, will there be reference inputs. Which inputs are connected to which output and so on.
- **Feedback Type**
  Some RMC modules require that you define the feed back type, such as voltage, current, Start/Stop, etc.
- **Scale and Offset**
  The Scale and Offset parameters convert the transducer feedback to real-life units, such as inches, millimeters, etc.
- **Tuning**
  The tuning procedure sets the PID gains for closed-loop control.

Communication

The RMC can communicate as a slave with other devices via Ethernet, Serial RS-232/485, or PROFIBUS depending on the RMC. For details on the RMC's communication protocols, see the Communications Overview topic.

The RMC does not perform motion control on feedback via any communication channels. It only controls motion on axes that are directly wired to the RMC's modules.

See Also

Help Overview

1.3. Disclaimer

Although great effort has been taken to ensure the accuracy of the information in this documentation, it is intended to be used only as a guide. Knowledge of motion control, hydraulic servos, electric servos, transducers, and safety rules is required. Delta Computer Systems, Inc. does not accept responsibility for problems resulting from errors or omissions in this documentation. The information in this documentation is subject to change without notice.

Neither Delta Computer Systems, Inc. nor anyone else involved in the creation, production, or delivery of this product shall be liable for any direct, indirect, consequential injuries and or damages arising out of the use, the results of use, or the inability to use this product.

All brand names and trademarks referenced in this manual are the property of their respective owners.

RMC Return for Repair

Should an RMC require repair, refer to the Technical Support topic for return details.

Ownership

Title to all intellectual property rights associated with Delta product (including, but not limited to firmware and software) remains with Delta. Any modifications to products, or special versions of products, will become the sole property of Delta, even if such modifications or special versions are made by request of Customer and paid for, all or in part, by Customer, unless Delta specifically agrees otherwise in writing.

2. Starting Up the RMC

Show All

2.1. RMC Startup Procedure

The best resource to start using the RMC is the Startup Guide that was shipped with the controller. The Startup Guide can also be downloaded from Delta's website at https://deltamotion.com. If you do not have a Startup Guide, the following step-by-step procedure will help you get your system up and running.

TIP: Deltas position/pressure simulator provides a simple way to test your program before connecting the controller to a real system.

1. Provide Power to the RMC

Apply power to the RMC as described in the Wiring topics.

2. Connect to Controller in RMCTools

a. **RMC75E, RMC150E, or RMC200:**
Connect a USB cable from the PC to the USB Monitor port on the RMC.

RMC75S or RMC75P:
Connect a null-modem cable from the PC to the RS-232 Monitor port on the RMC. See the Monitor Port topic for cable details.

b. Open RMCTools.

c. On the **File** menu, click **New Project**. In the **New Project Wizard**, fill in the fields and click **Finish**.

d. In the **New Controller Wizard**, type a **Controller Name** and choose **Automatically Detect the Controller Information**.

e. Click **Next**.

f. **RMC75E or RMC150E:**
Choose USB and click **OK**. Choose the desired RMC and click **Next.**

RMC75S or RMC75P:
Choose the COM port that the RMC is connected to.

g. Click **Next**.

h. You may need to wait while RMCTools connects to the controller. Once it has connected, verify that the information is correct and click **Finish**.

3. Configure Module Properties

The following hardware may require configuration before use as part of an axis:

- **RMC150 UI/O Module High-Speed Channels**
  The high-speed channels of the Universal Input/Output module need to be configured for use as a Quadrature axis input or SSI axis input before they can be used as an axis input. For details, see the Configuring UI/O High-Speed Channels topic.

- **RMC200 S8**
  The last two inputs, 6 and 7, can be configured as a single quadrature input. For details, see the S8 Module topic.
- **RMC200 D24**
  The high-speed inputs may be configured as one or two quadrature inputs. For details, see the D24 Module topic.
- **RMC200 U14**
  The high-speed channels of the Universal Input/Output module need to be configured for use as a SSI/MDT or Quadrature axis inputs before they can be used as an axis input. For details, see the Configuring U14 High-Speed Channels topic.

4. Define the Axes

RMCTools provides full flexibility for creating axes and assigning the RMC axes to the hardware. When you create or change axes, you can choose which inputs(s) and outputs are assigned to the axis.

After the RMC powers up, it has default axis definitions. To change these definitions, do the following:

a. In the **Project** tree, expand the **Axes** folder and double-click **Axis Definitions**.
b. Use the dialog to view the axis definitions and change them if you would like. For more details, click the Help button in the dialog.

5. Connect a Feedback Device

Note: This step can be done before or after step 6.

a. **Important:** Turn off power to the RMC and the feedback device before connecting any wires!
b. For each axis you wish to connect a feedback device to, wire it to the RMC according to the instructions in the Wiring topic.
c. After wiring, re-apply power to the RMC and the feedback device.
d. In RMCTools, in the **Project** pane, click the **RMC** controller.
e. On the RMCTools toolbar, click the **Controller** button and choose **Go Online**. Verify that the red circle around the RMC icon in the **Project** pane disappears, indicating that it is online with the RMC.
f. The RMC must be properly configured before it will communicate with the transducer(s). Refer to the procedure for your transducer type:

- SSI transducer
  i. On the RMCTools toolbar, click the **Axis Tools** button.
  ii. In the **Axis Parameters** pane, on the **Setup** tab, under the **Primary Control Setup**, in the **Feedback Type** register, select **SSI**.
  iii. From the information in your SSI transducer data sheet, enter the correct value for each of these registers:
    - **SSI Format** - Binary or Gray
    - **SSI Data bits** - (e.g. 24)
  iv. To apply the changes to the RMC, click the **Download** button or press Ctrl+D.

- Magnetostrictive transducer with Start/Stop or PWM
  i. On the RMCTools toolbar, click the **Axis Tools** button.
  ii. In the **Axis Parameters** pane, on the **Setup** tab, under the **Primary Control Setup**, in the **Feedback Type** register, select **MDT**.
  iii. In the **MDT Type** register, select the type of MDT transducer you have. This information should be available on the MDT datasheet. The options are:

- Start/Stop Rising Edge
- Start/Stop Falling Edge
- Pulse-Width Modulated

iv. To apply the changes to the RMC, click the **Download** button or press Ctrl+D.

- Voltage or Current transducer
  i. On the RMCTools toolbar, click the **Axis Tools** button.
  ii. In the **Axis Parameters** pane, on the **Setup** tab, under the **Primary Control Setup**, in the **Input Type** register, select **Voltage or Current**.
  iii. To apply the changes to the RMC, click the **Download** button or press Ctrl+D.

- Quadrature encoder

  There are no parameters required to set up in order to interface to a quadrature encoder. If your motion is rotary, see the rotary topic for details on setting the Rotary vs. Linear parameter.

- Resolver

  i. On the RMCTools toolbar, click the **Axis Tools** button.
  ii. In the **Axis Parameters** pane, on the **Setup** tab, under the **Primary Control Setup**, in the **Resolver Resolution** register, choose a resolution. For details, see the Resolver Resolution topic.
  iii. In the **Reference Frequency** register, choose the frequency required for your resolver. This information should be available from the resolver data sheet.
  iv. In the **Reference Amplitude** register, click the ellipsis button to open the dialog. Click the **Help** button for details on how to set the Reference Amplitude.
  v. To apply the changes to the RMC, click the **Download** button or press Ctrl+D.

g. On the RMCTools toolbar, click the **Axis Tools** button.
h. In the **Axis Tools**, in the **Axis Status Registers** pane, on the **Basic Position** tab, look at the **Actual Position** register. It may be changing slightly.
i. Move the axis and look for a corresponding change in the **Actual Position** register. If it does not change, recheck the wiring, verify that the **Primary Control Setup** registers in the **Axis Parameters** pane are correct, and check for changing **Actual Position** again. When it is working correctly, proceed to the next step.

> **Note:**
> It is important that the transducer is connected and is working properly before continuing the Start-up procedure.

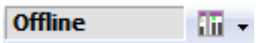
j. On the **Controller** menu, click **Update Flash**. This stores your changes in the RMC even in the event of a power outage.
k. Press Ctrl+S to save the project.

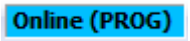
6. Connect an Actuator

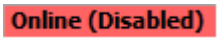
> **Note:**
> Read this section completely *before* executing any commands on the RMC.

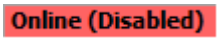
a. **Important:** Turn off power to the RMC and the actuator before connecting any wires!
b. For each axis you wish to connect an actuator to, wire it to the RMC according to the Wiring topic.
c. After wiring, re-apply power to the RMC, the feedback device, and the actuator.

> **Note:**
> To test the actuator, you will supply a Control Output voltage from the RMC to the actuator. Before doing this, make sure that the axis may safely move in either direction!

d. In RMCTools, in the **Project** pane, click the **RMC** controller.
e. On the RMCTools toolbar, click the **Controller** button 

and choose **Go Online**. The toolbar will indicate that RMCTools is online 

or 

.
f. On the RMCTools toolbar, click the **Axis Tools** button  .
g. In the **Axis Parameters** pane, on the **All** tab, expand the **Simulate** section and verify that **Simulate Mode** is unchecked. If you make changes, you must click the Download button  or press Ctrl+D to apply the changes to the RMC.
h. RMC200 Only: On the toolbar, click 

and choose Program Mode.
i. Enable the axis: In the Command Tool, in the **Cmd** box, type "Enable", then choose **Enable Controller (7)** from the list. Click **Send**.
j. In the **Axis Tools**, in the **Axis Status Registers** pane, on the **Basic** tab, verify that the **Control Output** for the axis is 0 (zero).
k. Turn on power to the motor or hydraulics for the axis being set up. Note that after starting, it is normal for the axis to drift slowly.
l. In the **Command Tool**, do the following steps to apply 0.1 V to the axis Control Output:

l. **Caution:** Use the Direct Output command with caution! It disables the safety features of the RMC!

- Double-click the **Cmd** box, type "D" and choose the **Direct Output (9)** command.
- Set the **Output** to 0.1 V.
- Set the **Ramp Rate** to 100 V/s.
- Click **Send Command**.
- Enter 0 in the **Output** box. This will allow you to quickly issue a Direct Output Command with 0 drive in the next steps.

m. Verify that the **Control Output** register for the axis is at 0.1 V.
n. Observe the **Actual Position** register (on the **Basic Position** tab) and note whether it is increasing or decreasing. One of three things could have happened at this point:

- **The actuator moved and the Actual Position increased.**
  In this case you are ready to go to the next section and set the Scale.
- **The actuator moved and the Actual Position decreased.**
  In this case, first verify that the wiring is correct. If it is not, fix it and repeat the process. If it is, invert the Output Polarity to reverse the direction:

m.

1. In the **Axis Parameters** pane, select the **Setup** tab and expand the **Primary Control Setup** section.
2. Double-click the **Invert Output Polarity** register in the axis you are using. The **Invert Output Polarity** box should now have a checkmark in it.
3. Click the **Download** button  or press Ctrl+D to apply the changes to the controller.

4. Repeat the process again: Issue the Direct Output command again, observe the Actual Position, and see which of the three things happened.

m.

- **The actuator does not move.**
  In this case, you may need to increase the Control Output. First, verify the following:
-

1. Check that the physical Control Output voltage really is 0.1 volts.
2. Check that the actuator is enabled.

If these items check out fine, repeat the process above with a larger **Output**. Some systems may require much more than 0.1 volt to move.

n. **Note:**
It is essential that the Actual Position increase when a positive output voltage is specified with Direct Output command. If this condition is not met, you will not be able to control the axis in closed loop.

o.

p. In the Command Tool, enter 0 in the **Output** box and click **Send Command**, or press Alt + S, to turn the **Control Output** voltage off.

q. In the Command tool, enter -0.1 in the **Output** box and click **Send Command**, or press Alt + S. The **Control Output** register should change to -0.1 and the axis should move in the opposite direction.

Once the actuator moves fine, you are ready to go to the next step.

7. Set the Scale and Offset

The **Scale** and **Offset** parameters convert the **Counts** from the transducer into meaningful measurement units. In order to do so, the **Scale** and **Offset** must first be configured correctly.

Set the Scale and Offset:

a. In the **Axis Tools**, in the **Axes Parameters** pane, select the **Setup** tab. Note that the first two registers under the **Primary Control Setup** section are **Position Scale** and **Position Offset**. These are the parameters you will configure.

b. To calculate what the scale and offset should be, you will need to refer to the Scaling topic.

c. When you have determined what value the scale or offset parameter should be, type the new value in the cell in the **Axis Parameters** pane. Then click the **Download** button or press Ctrl+D to apply the changes to the controller.

d. Move your axis and verify that the positions are correct.

e. On the **Controller** menu, click **Update Flash**. This stores your changes in the RMC even in the event of a power outage.

f. Press Ctrl+S to save the project.

Set the Display Units:

a. In the **Axis Tools**, in the **Axes Parameters** pane, select the **Setup** tab.

b. In the **Primary Control Setup** section, locate the **Display Units** parameter. From the drop-down list, select a unit.
If you wish to use units not listed, choose **Custom**, then type up to 4 characters in the **Custom Units** parameter.

c. For dual-loop axes, repeat this in the **Secondary Control Setup** section.

d. Click the **Download** button to apply the changes.

8. Tune Each Axis

In order to control an axis in closed-loop control, it must first be tuned. Refer to the online help for the tuning procedure:

a. In RMCTools, on the **Tools** menu, click **Tuning Tools**. This pane provides everything you need for tuning the axes.

b. Follow the instruction in the Tuning topic to tune the system.

9. Set up and Configure the Communications

If your RMC will be communicating with an external controller, such as a PLC, PC, or other device, you must set up the communication. Refer to one or more of the following topics for details on configuring the communication type of your RMC controller:

- Ethernet Overview
- PROFIBUS-DP
- Serial Overview

10. Save Your Configuration Settings

Make sure to save your settings! You must do the following:

a. **Save your RMCTools project**
To save the project, on the **File** menu, click **Save**. Before doing this, make sure the project data is the same as the controller data, by either downloading or uploading.

b. **Save the RMC data to non-volatile memory**
On the **Controller** menu, click **Update Flash**.

Tip: To do both these steps at once, on the **File** menu, choose **Save and Update Flash**.

See Also

Scaling Overview | Tuning Overview

2.2. Scaling

2.2.1. Scaling Overview

Scaling refers to converting the transducer feedback into meaningful units. The RMC uses the Scale and Offset parameters to convert the transducer Counts or Raw Counts into measurement units (position, velocity, pressure, force). For example, the Voltage returned by an analog position transducer must be converted to positions in order to be useful for control. In order to correctly convert the transducer feedback to useful units, you must calculate the Scale and Offset parameters.

Units

The RMC uses the default terms **Position Units (pu)**, **Pressure Units (Pr)**, and **Force Units (Fr)** for the scaled feedback. You can change these to feedback units of your preference by choosing from a list or entering a custom unit. See Display Units for details.

Calculating the Scale and Offset

RMCTools provides Scale/Offset wizards to help you calculate the Scale and Offset parameters. You can also calculate the parameters manually.

Scale/Offset Wizards

To access the Scale/Offset wizard, in **Axis Tools**, in the **Axis Parameters** pane, click the **Setup** tab. Expand the **Tools and Wizards** section. Click **Launch** to open the Scale/Offset Wizard.

Note:
Before using the Scale/Offset Wizard, you must define the axes and set the transducer type parameters.

Manual Calculation

The methods of calculating the scale and offset parameters depends on the axis type. See the scaling topic for your axis type:

Analog Position Scaling

Analog Velocity Scaling

Analog Acceleration Scaling

Analog Pressure/Force Scaling

Load Cell Scaling

MDT Scaling

SSI Scaling

Quadrature Scaling

Resolver Scaling

Rotary Scaling

Scale and Offset Parameters

The following parameters converting the transducer feedback Counts into meaningful units. Each axis, whether a control axis or reference axis, has these parameters:

- **Scale**
  On a linear axis, the Position, Pressure, Force, or Velocity Scale defines the number of Actual units (position, velocity, pressure or force) per count or volt returned from the transducer.
- **Offset**
  On a linear axis, the Position, Pressure, Force, or Velocity Offset moves the zero point of the Actual units to where the user wants it.
  On a rotary axis, the Position Offset defines the lower range of the position range.
- **Count Offset**
  This parameter is used for Absolute Rotary axes to determine where the zero point of the counts should be.
- **Position Unwind**
  On a rotary axis, the Position Unwind defines the span of the position range.
- **Count Unwind**
  On a rotary axis, the Count Unwind defines the range of counts over which the Position Unwind spans.
- **Current or Voltage Offset**
  Provides an offset value that is applied before the Scale and Offset parameters.
- **Millivolts/Volt Offset**
  Provides an offset value that is applied before the Scale and Offset parameters.
- **Negative Correction Factor**
  This parameter is used for bidirectional load cells, which typically have a slight gain difference between the positive and negative side.

See Also

Scale/Offset Wizard

2.2.2. Analog Position Scaling

To have any useful meaning, the Voltage or Current from an analog transducer must be scaled to measurement units. This topic describes how to manually calculate the Scaling and Offset parameters for an analog position input.

Delta Recommends using the Scale/Offset Wizard for scaling the position. If you need to do it manually, read this topic.

Manually Scaling Voltage or Current to Position Units

The RMC calculates the Actual Position every control-loop time using either of the following formulas:

Actual Position = (Voltage x Position Scale) + Position Offset

Actual Position = (Current x Position Scale) + Position Offset

Calculating the Position Scale and Offset

The Scale/Offset Wizards provide the easiest method of scaling your axis. If you want to compute your scale and offset manually, do the following:

The accuracy of this method depends on how accurately you can measure two positions of the axis.

1. Physically measure the axis' position at two points and record the value of the Volts register at each point. Call the *smaller* measured position P_0, and its corresponding voltage V_0 or current C_0. Call the *greater* measured position P_1, and its corresponding voltage V_1 or current C_1.
2. Calculate the Position Scale with the following equation:

   $$\text{Position Scale} = (P_0-P_1)/(V_0-V_1)$$
   $$\text{Position Scale} = (P_0-P_1)/(C_0-C_1)$$
3. Calculate the Position Offset with the following equation:

   $$\text{Position Offset} = P_0 - \text{Position Scale} \times V_0$$
   $$\text{Position Offset} = P_0 - \text{Position Scale} \times C_0$$

See Also

Scaling Overview | Scale/Offset Wizard | Position Scale | Position Offset

2.2.3. Analog Velocity Scaling

To have any useful meaning, the Voltage or Current from an analog transducer must be scaled to measurement units. This topic describes how to manually calculate the Scaling and Offset parameters for an analog velocity input.

Delta Recommends using the Scale/Offset Wizard for scaling the velocity. If you need to do it manually, read this topic.

Scaling Voltage or Current to Velocity Units for Tachometers

The RMC calculates the Actual Velocity from the tachometer voltage every control-loop time using either of the following formulas:

Actual Velocity = (Voltage + Velocity Offset) x Velocity Scale

Actual Velocity = (Current + Velocity Offset) x Velocity Scale

Calculating the Velocity Scale and Offset

1. From the tachometer specification sheet, determine the voltage or current at which the velocity is zero. Call it V_{ZERO} or C_{ZERO}. Or, for greater accuracy, make sure the tachometer is

stopped, then look at the voltage or current feedback value in the Status Registers pane. Call this value Call it V_{ZERO} or C_{ZERO}.

2. Negate V_{ZERO} or C_{ZERO} and enter it as the Velocity Offset:

   Velocity Offset = $-V_{ZERO}$

   Velocity Offset = $-C_{ZERO}$

3. From the tachometer specification sheet, determine what the maximum velocity is. Call it Max Vel.
4. From the tachometer specification sheet, determine what the voltage or current output should be at the Max Vel. Call it V_{MAX} or C_{MAX}. Calculate the Velocity Scale with one of the following equations:

   Velocity Scale = Max Vel / (V_{MAX} - V_{ZERO})

   Velocity Scale = Max Vel / (C_{MAX} - C_{ZERO})

1.

See Also

Scaling Overview | Scale/Offset Wizard | Velocity Scale | Velocity Offset

2.2.4. Analog Acceleration Scaling

To have any useful meaning, the Voltage or Current from an analog transducer on an acceleration input must be scaled to measurement units. This topic describes how to manually calculate the Scaling and Offset parameters for an analog acceleration transducer.

Notice that Acceleration inputs are only used in advanced applications.

Scaling Acceleration Units

The RMC calculates the Actual Acceleration every control-loop time using the following formula:

Actual Acceleration = ((Voltage or Current) + Acceleration Offset) x Acceleration Scale

Calculating the Acceleration Scale and Offset

1. From the accelerometer specification sheet, determine the voltage or current at which the acceleration is zero. Call it V_{ZERO} or C_{ZERO}. Negate that value, and enter it as the Acceleration Offset.

   Acceleration Offset = $-V_{ZERO}$

   Acceleration Offset = $-C_{ZERO}$

2. From the accelerometer specification sheet, determine what the maximum acceleration is. Call it Max Accel.
3. From the accelerometer specification sheet, determine what the voltage or current output should be at the Max Accel. Call it V_{MAX} or C_{MAX}. Calculate the Acceleration Scale with one of the following equations:

   Acceleration Scale = Max Accel / (V_{MAX} - V_{ZERO})

   Acceleration Scale = Max Accel / (C_{MAX} - C_{ZERO})

1.

See Also

Scaling Overview | Acceleration Scale | Acceleration Offset

2.2.5. Load Cell Scaling

To have any useful meaning, the mV/V signal from a load cell input must be scaled to force units. The Force Scale, Force Offset, Millivolts/Volt Offset, and Negative Correction Factor parameters are used to define force units as a function of the load cell mV/V signal. This topic describes how to manually calculate these parameters for a load cell input.

Delta Recommends using the Scale/Offset Wizard for scaling the load cell force. If you need to do it manually, read this topic.

Scaling Load Cell Force

The RMC calculates the Actual Force every control loop using the following formula:

If (Millivolts/Volt + Millivolts/Volt Offset) < 0 Then

Force = (**Millivolts/Volt** + **Millivolts/Volt Offset**) x Negative Correction Factor x Force Scale + Force Offset

Else

Force = (**Millivolts/Volt** + **Millivolts/Volt Offset**) x Force Scale + Force Offset

EndIf

To correctly scale the axis for your particular application, use one of the methods below to find the scaling parameters.

Method 1: Use Transducer Specifications

This method uses the calibration data for the load cell.

Unidirectional Load Cells (Tension-Only or Compression-Only)

1. From the transducer data sheet:
    a. Find the mV/V output of your load cell at the rated load. Call these Max mV/V and Max Force.
    b. Find the mV/V value at zero force. This may be called Zero Balance.
2. Calculate the Force Scale with the following formula:

    Force Scale = Max Force / (Max mV/V - Zero Balance)
3. Enter the negative of the Zero Balance value in the Millivolts/Volts Offset parameter.

Bidirectional Load Cells (Tension and Compression)

1. From the transducer data sheet:
    a. Find the mV/V output of your load cell at the rated positive load. Call these Max Positive mV/V and Max Positive Force.
    b. Find the mV/V output of your load cell at the negative rated load. Call these Max Negative mV/V and Max Negative Force.
    c. Find the mV/V value at zero force. This may be called Zero Balance.
2. Calculate the Force Scale with the following formula:

    Force Scale = Max Positive Force / (Max Positive mV/V - Zero Balance)
3. Calculate the Negative Correction Factor with the following formula:

    Negative Correction Factor = (Max Negative Force / (Max Negative mV/V - Zero Balance)) / (Max Positive Force / (Max Positive mV/V - Zero Balance))
4. Enter the negative of the Zero Balance value in the Millivolts/Volts Offset parameter.

Method 2: Force / Millivolts Per Volt

The accuracy of this method depends on how accurately you can measure the force of the axis with some external device such as a reference load cell or some known mass.

Unidirectional Load Cells (Tension-Only or Compression-Only)

1. For two different force values (Force0 and Force1), apply a known force, and record the mV/V values (mVV0 and mVV1) at those forces.
2. Calculate the Force Scale and Force Offset with the following formula:

   Force Scale = (Force0 - Force1) / (mVV0 - mVV1)

   Force Offset = Force0 - Force Scale x mVV0

Bidirectional Load Cells (Tension and Compression)

1. For three different force values, apply a known force. The force values should be a large positive value (ForcePos), a large negative value (ForceNeg), and one close to zero (ForceZero). For each force value, record the mV/V values at those forces (mVVPos, mVVNeg, and mVVZero).
2. Calculate the Force Scale, Force Offset, and Negative Correction Factor with the following formulas:

   Force Scale = (ForcePos - ForceZero) / (mVVPos - mVVZero)

   Force Offset = ForcePos - Force Scale x mVVPos

   Negative Correction Factor = (ForceNeg / mVVNeg) / (ForcePos / mVVPos)

   A more accurate negative correction factor would be obtained from measuring force at four points, two positive and two negative. Generate two scale values from these and use the ratio of the scales as the correction factor.

See Also

Scale/Offset Wizard | Force Offset | Millivolts/Volt Offset | Negative Correction Factor

---

2.2.6. Analog Pressure or Force Scaling

To have any useful meaning, the Voltage or Current from the analog pressure or force transducer must be scaled to pressure or force units. The Pressure/Force Scale, Pressure/Force Offset, and Invert Feedback Polarity parameters are used to define pressure or force units as a function of transducer voltage. This topic describes how to manually calculate these parameters for an analog pressure transducer.

Delta Recommends using the Scale/Offset Wizard for scaling the pressure or force. If you need to do it manually, read this topic.

Scaling Single-Input Pressure or Force

The RMC calculates the Actual Pressure/Force every control-loop using the following formula (may be either pressure or force):

Actual Pressure = (Voltage (or Current) x Pressure Scale) + Pressure Offset

To correctly scale the axis for your particular application, use one of the methods below to find the Pressure/Force Scale and Pressure/Force Offset parameters.

Method 1: Use Transducer Specifications

- From the transducer data sheet, find the maximum output voltage of your pressure transducer and the pressure at that voltage.
- Calculate the Pressure Scale with the following formula:

  Pressure Scale = Max Pressure / Max Voltage

  - Make sure the system has 0 pressure and then record the value of the Voltage register. Enter the negative of that value as the Pressure Offset.

Method 2: P0/P1 Calculation

The accuracy of this method depends on how accurately you can measure the pressure of the axis.

1. Physically measure two different pressures of the axis and record the value of the Voltage register at each point. (The difference between the two pressure should be as large as possible). Call the *smaller* measured pressure P_0, and its corresponding Voltage V_0. Call the *greater* measured pressure P_1, and its corresponding Voltage V_1.
2. Calculate the Pressure Scale with the following equation:

   Pressure Scale = $(P_0-P_1)/(V_0-V_1)$

   If the Pressure Scale is negative, ignore the minus sign. RMCTools accepts only a positive Scale.
3. Calculate the Pressure Offset with the following equation:

   Pressure Offset = P_0 - Pressure Scale x V_0

Scaling Dual-Input (Differential) Force - RMC75 and RMC150

Dual-Input Force requires two pressure transducers. Each pressure transducer must be individually scaled. The RMC75 and RMC150 calculate the Actual Differential Force every control loop using the following formulas:

Actual Force A = (Channel A Voltage (or Current) x Force A Scale) + Force A Offset

Actual Force B = (Channel B Voltage (or Current) x Force B Scale) + Force B Offset

Actual Differential Force = Actual Force A - Actual Force B

To correctly scale the axis for your particular application, use the method below to find the A and B Force Scale and Force Offset parameters.

Method: Use Transducer Specifications

This method assumes the two pressure transducers provide zero volts at zero pressure or force.

- From the transducer data sheet, find the maximum output voltage of your pressure transducer and the pressure at that voltage.
- Calculate the area of the A side (cap end) of the piston.
- Multiply the area of the A side with the maximum pressure of the A transducer to get the maximum A force.
- Calculate the Force A Scale with the following formula:

  Force A Scale = maximum A force / Max A Voltage
- Calculate the effective area of the B side (rod end) of the piston. This is the area of the cylinder minus the area of the rod.
- Multiply the area of the B side with the maximum pressure of the B transducer to get the maximum B force.
  - Calculate the Force B Scale with the following formula:

    Force B Scale = maximum B force / Max B Voltage
  - Make sure the system has 0 force and then record the value of the Channel A Actual Force (to find this register, in Axis Status Registers pane, click the All tab, and expand the Pressure/Force Feedback section). Negate the value and enter it as the Force A Offset.

Scaling Dual-Input (Differential) Force - RMC200

Dual-Input Force requires two pressure transducers. Each pressure transducer must be individually scaled. As shown by the formulas below, the RMC200 calculates the pressures for each channel, then the forces for each channel, then takes the difference and applies the Dual Channel Force Offset to arrive at the Actual Force.

Channel A Pressure = Channel A Voltage (or Current) * Channel A Pressure Scale + Channel A Pressure Offset

Channel B Pressure = Channel B Voltage (or Current) * Channel B Pressure Scale + Channel B Pressure Offset

Channel A Force = Channel A Pressure * Channel A Force Scale

Channel B Force = Channel B Pressure * Channel B Force Scale

Actual Force = Channel A Force - Channel B Force + Dual Channel Force Offset

To correctly scale the axis for your particular application, use the method below to find the A and B Force Scale and Force Offset parameters.

Method: Use Transducer Specifications

This method assumes the two pressure transducersprovide zero volts at zero pressure or force.

- From the transducer data sheet, find the maximum output voltage of your pressure transducer and the pressure at that voltage.
- Calculate the Channel A Pressure Scale with the following formula:

  Channel A Pressure Scale = Maximum A Pressure / Max A Voltage or Current
- Calculate the area of the A side (cap end) of the piston.
- Calculate the Force A Scale with the following formula:

  Channel Force A Scale = area of A side of the piston
- Repeat for Channel B
  - Make sure the system has 0 force and then record the value of the Actual Force. Negate the value and enter it as the Dual Channel Force Offset.

See Also

Scaling Overview | Scale/Offset Wizard | Pressure/Force Scale | Pressure/Force Offset

Show All

2.2.7. MDT Scaling

To have any useful meaning, the counts from the transducer must be scaled to position units. The Position Scale and Position Offset parameters define the position units as a function of transducer counts. This topic describes how to correctly calculate these parameters for an MDT transducer.

Delta Recommends using the Scale/Offset Wizard for scaling the position. If you need to do it manually, read this topic.

Scaling Counts to Position Units

The RMC calculates the Actual Position every control-loop time using the following formula:

Actual Position [pu] = (Counts [cnt] x Position Scale [pu/cnt]) + Position Offset [pu]

Note:
If the Actual Position filter is applied, the RMC filters the Actual Position after calculating it with the above formula.

Manually Calculating the Scale and Offset

The Scale/Offset Wizards provide the easiest method of scaling your axis. If you prefer to do it manually, read this section.

Determining the Scale and Offset

There are two methods of finding the correct Position Scale and Position Offset.

Method 1: P_0/P_1 Calculation

The accuracy of this method depends on how accurately you can measure two positions of the axis.

1. Physically measure the axis position at two points and record the value of the counts register at each point. Call the *smaller* measured position P_0, and its corresponding Counts C_0. Call the *greater* measured position P_1, and its corresponding Counts C_1.
2. Calculate the Position Scale with the following equation:

   $$\text{Position Scale} = (P_0-P_1)/(C_0-C_1)$$

3. Calculate the Position Offset with the following equation:

   $$\text{Position Offset} = P_0 - \text{Position Scale} \times C_0$$

Method 2: Using the MDT Calibration Number

1. Obtain the following information (usually from the transducer label or data sheet):
   - Calibration Number (calibration constant) in µs/in or µs/µm (typically about 9 µs/in)
   - Number of Recirculations (if it has no recircs, use a value of 1)
2. Calculate the Position Scale with the following formula:

   RMC75 and RMC200:

   $$\text{Position Scale (pu/counts)} = \frac{1}{\text{Calibration Number (µs/pu)} \times 240\text{Mhz} \times \text{\# of Recircs}}$$

   RMC150:

   $$\text{Position Scale (pu/counts)} = \frac{1}{\text{Calibration Number (µs/pu)} \times 120\text{Mhz} \times \text{\# of Recircs}}$$

   where

   - "pu" is the position units, e.g. in, mm, m, etc. If you want position units other than what is given by the Calibration Number (e.g. in,mm, etc.), you must convert the calibration number to the units you want.
   - The 240MHz and 120 MHz values comes from the RMC's internal counter. If you change the units from the above equation, make sure the units for the equation work out properly. The units for MHz is [10^6/sec].
3. If you wish to reverse the direction of the feedback, make the Scale negative.
4. Download the Scale to the RMC.
5. Calculate the Position Offset:
   - Make sure the Position Offset is zero (0) and downloaded to the controller before continuing.
   - In the Axis Tools window, look at the Actual Position register.
   - Move the axis to a point where you know what you want the position to be (such as 0). Called this the Desired Position. Look at the **Actual Position** register. Call this position P_0.
   - Calculate the Position Offset using the following equation: Position Offset = Desired Position - P_0.
6. Download the Offset to the RMC.

Example

1. Connected to an RMC75 is an MDT transducer with a calibration number of 9.1615 µs/in and no recirculations (this means it has just 1 recirculation). The system needs to be scaled to inches.
    - To calculate the Scale:

$$\text{Position Scale} = \frac{1}{9.1615[\mu\text{sec/in}] \times 240[\text{MHz}] \times 1} = 0.0004548018[\text{in/count}]$$

    - Download the Scale to the RMC.
    - To calculate the Offset: Move the axis to where it should be 0. The Actual Position shows 2.4528. The Position Offset should be: 0 - 2.4528 = -2.4528.
    - Download the Offset to the RMC.

2. Connected to an RMC75 is an MDT transducer with a calibration number of 9.012 µs/in and 2 recirculations. The system needs to be scaled to inches.
    - To calculate the Scale:

$$\text{Position Scale} = \frac{1}{9.012[\mu\text{sec/in}] \times 240[\text{MHz}] \times 2} = 0.00023117[\text{in/count}]$$

    - Download the Scale to the RMC.
    - To calculate the Offset: Move the axis to where it should be 10.125. The Actual Position shows 11.241. The Position Offset should be: 10.125 - 11.241 = -1.116.
    - Download the Offset to the RMC.

3. Connected to an RMC150 is an MDT transducer with a calibration number of 9.1313 µs/in and no recirculations (this means it has just 1 recirculation). The system needs to be scaled to millimeters.
    - We must first calculate the scale, but in order to do that we need to know how many position units there are in an inch: 25.4 mm/in.
    - To calculate the Scale:

$$\text{Position Scale} = \frac{1}{9.1313[\mu\text{sec/in}] \times \frac{1[\text{in}]}{25.4[\text{mm}]} \times 120[\text{MHz}] \times 1} = 0.02318[\text{mm/count}]$$

    - Download the Scale to the RMC.
    - To calculate the Offset: Move the axis to where it should be 0 mm. The Actual Position shows 24.8. The Position Offset should be: 0 - 24.8 = -24.8.
    - Download the Offset to the RMC.

See Also

Scaling Overview | Scale/Offset Wizard | Position Scale | Position Offset

Show All

2.2.8. SSI Scaling

To have any useful meaning, the counts from the transducer must be scaled to position units. The Position Scale and Position Offset parameters define the position units as a function of transducer counts. RMCTools provides Scale/Offset wizards to guide you through the scaling process. This topic describes how to manually calculate these parameters for an SSI transducer.

To scale rotary SSI axes, see the Rotary Scaling topic.

Delta Recommends using the Scale/Offset Wizard for scaling the position. If you need to do it manually, read this topic.

Scaling Counts to Position Units

The RMC calculates the Actual Position every control-loop time using the following formula:

Actual Position [pu] = (Counts [cnt] x Position Scale [pu/cnt]) + Position Offset [pu]

Note:
If the Actual Position filter is applied, the RMC filters the Actual Position after calculating it with the above formula.

To correctly scale the axis for your particular applications, use the methods below to find the Position Scale and Position Offset parameters.

Determining the Scale and Offset

There are two methods of finding the correct Position Scale and Position Offset.

Method 1: P_0/P_1 Calculation

The accuracy of this method depends on how accurately you can measure two positions of the axis.

1. Physically measure the axis position at two points and record the value of the counts register at each point. Call the *smaller* measured position P_0, and its corresponding Counts C_0. Call the *greater* measured position P_1, and its corresponding Counts C_1.
2. Calculate the Position Scale with the following equation:

   $$\text{Position Scale} = (P_0-P_1)/(C_0-C_1)$$
3. Calculate the Position Offset with the following equation:

   $$\text{Position Offset} = P_0 - \text{Position Scale} \times C_0$$

Method 2: Using the Transducer Resolution

Linear Transducers:

1. Obtain the transducer resolution (usually from the transducer data sheet), for example 0.005 mm. The resolution of the transducer indicates the smallest increment of the transducer. This increment is one count in the RMC.
2. Determine the position units you would like for your Actual Position, for example inches, meter, mm, etc.
3. The Position Scale is equal to the number of position units per count, for example, using inches and a resolution of 0.005 mm:

   $$\text{Position Scale} = \frac{1[\text{in}]}{25.4[\text{mm}]} \times 0.005\ [\text{mm/count}] = 0.0001968\overline{5}[\text{in/count}]$$
4. If you wish to reverse the direction of the feedback, make the Scale negative.
5. Download the Scale to the RMC.
6. Calculate the Position Offset:
   - Look at the **Actual Position** register and the **Counts** register.
   - Move the axis to a point where you know what you want the position to be (such as 0). Called this the Desired Position. Look at the **Actual Position** register. Call this position P_0.
   - Calculate the Position Offset using the following equation:

     $$\text{Position Offset} = \text{Desired Position} - P_0.$$
6. Download the Offset

See Also

Scaling Overview | Scale/Offset Wizard | Position Scale | Position Offset | Rotary Scaling

2.2.9. Rotary Scaling

SSI, resolver, and quadrature axes can be configured as rotary axes. To have any useful meaning, the counts from the encoder must be scaled to position units. This topic describes the scaling parameters for a rotary axis and provides instructions on how to scale the axis.

Delta Recommends using the Scale/Offset Wizard for scaling the position. If you need to do it manually, read this topic.

Calculating the Scale Parameters

To calculate the scale parameters, use the Rotational Scale/Offset wizard.

You can also enter the scale parameter manually. Refer to the **Scaling Parameter Details** section below for a description of each parameter.

Scaling Parameter Details

A rotary axis uses the Count Unwind, Position Unwind, Position Offset, and Count Offset parameters to calculate the position units from the counts:

- **Count Unwind**

  The counts for a rotary input are kept within a defined range. When the input goes beyond one end of this range, the counts wrap to the other end of the range. The Count Unwind parameter defines this range. Any control being performed using this input is not interrupted; this is not seen as a position discontinuity.

  For absolute rotary axes, this parameter must be a power of 2. If it were not, the wrap point of the counts would not coincide with the wrap point of the encoder and it would not be possible to determine absolute position.

  RMC Resolver axes will always give 65,536 counts per revolution.

  Example:
  Consider a single-turn SSI encoder with 8192 counts per revolution. If the Count Unwind is set to 8192, the counts will go from 0 to 8191 and wrap once per revolution. If the Count Unwind is set to 2048, the counts will go from 0 to 2048 and will wrap four times per revolution.

- **Position Unwind and Position Offset**

  For the range of counts described above, the positions must be kept within a corresponding range. The Position Unwind and Position Offset parameters define this range.

  The Position Unwind defines how many position units the range will span. The Position Offset adjusts the modulo position range up or down.

  For absolute rotary axes, the Counts are in the opposite direction if the Position Unwind is negative.

  The Counts formula for a positive Position Unwind is:

  Counts = (RawCounts + CountOffset) MOD MaxCounts

  The Counts formula for a positive Negative Unwind is:

  Counts = ([MaxCounts-1] - RawCounts + CountOffset) MOD MaxCounts

  Example:
  Consider an SSI rotary axis with a Count Unwind of 1024. If the Position Unwind is set to 10,

the counts range from 0 to 1024 will span 10 position units. If the Count Offset is set to 0, then the range will go from 0 up to, but not including, 10. If the Count Offset is set to 3, then the range will go from 3 up to, but not including, 13.

- **Count Offset**

  The count offset is important for absolute rotary axes. The position range—as defined by the Count Unwind, Position Unwind and Position Offset parameters—will start at the same point where the counts are 0. The Count Offset adjusts the point where the counts are zero, and thereby adjusts the point where the position range will start.

  This parameter is not used by incremental rotary axes, such as quadrature axes.

  Example:
  Consider a single-turn SSI absolute rotary encoder with 2048 counts. The axis is set up with a Count Unwind of 2048, a Position Unwind of 360 and a Position Offset of -180. Therefore, the position range will go from -180 to 180 in one revolution of the encoder. Negative 180 will occur at the point where the transducer counts are 0. Now, assume the encoder was mounted on the machine such that -180 pointed straight up, but you want -180 to be pointing down. To fix this, rotate the encoder so that it is pointing straight down. Record the counts, and enter the negative of that value in the Count Offset parameter. -180 will now be pointing down.

See Also

Scaling Overview | Scale/Offset Wizard

Show All

2.2.10. Quadrature Scaling (Linear)

To have any useful meaning, the counts from the transducer must be scaled to position units. For a quadrature axis, the Position Scale parameter defines the position units as a function of transducer counts. This topic describes how to correctly calculate this parameter for an Quadrature encoder.

Delta recommends using the Scale/Offset Wizard for scaling the position. If you need to manually calculate the scale and offset, read this topic.

If your motion is linear, read this topic. If your motion is purely rotary, see the Rotary Scaling topic.

Pulses, Lines and Counts

As described in the Quadrature Fundamentals topic, the resolution of a quadrature encoder is typically given in Pulses per Revolution (PPR), also called Lines per Revolution. The RMC feedback sees each rising or falling edge of the A or B signal as one count. Therefore, each pulse or line of the encoder will result in four counts on the RMC quadrature input. For example, a 1000-line encoder will give the RMC 4000 counts per revolution.

Scaling Counts to Position Units

The RMC calculates the quadrature axis Actual Position every control-loop time using the following formula:

Actual Position [pu] = (Change in Counts [cnt] x Position Scale [pu/cnt]) + Last Position

Note:
If the Actual Position filter is applied, the RMC filters the Actual Position after calculating it with the above formula.

Manually Calculating the Scale and Offset

The Scale/Offset Wizards provide the easiest method of scaling your axis. If you prefer to do it manually, read this section.

Determining the Scale and Offset

There are two methods of finding the correct Position Scale and Position Offset.

Method 1: P_0/P_1 Calculation

The accuracy of this method depends on how accurately you can measure two positions of the axis.

1. Physically measure the axis position at two points and record the value of the counts register at each point. Call the first measured position P_0, and its corresponding Counts C_0. Call the second measured position P_1, and its corresponding Counts C_1.
2. Calculate the Position Scale with the following equation:

$$\text{Position Scale} = (P_0-P_1)/(C_0-C_1)$$

Method 2: Using the Encoder Resolution

1. Obtain the encoder resolution (usually from the transducer data sheet), for example 4000 counts/turn.
2. Determine the position units you would like for your Actual Position, for example inches, meter, mm, etc.
3. Determine how many position units correspond to one count.
   For example, consider an encoder with 4000 counts per revolution mounted to a motor that drives a linear axis through a 4:1 gearbox and then a rack and pinion. The rack moves 5 inches for every turn of the pinion. The desired position-units of the linear rack motion are inches.
   To find the resolution:
   a. Determine the number of counts generated in one revolution of the encoder:
      From the datasheet, this is 4000 counts.
   b. Calculate the linear travel (position units) in one revolution of the encoder:
      One revolution of the encoder equals 1/4 revolution of the pinion.
      One revolution of the pinion equals a travel distance of 5 inches.
      Therefore, the linear travel in one turn of the encoder is 5/4 = 1.2 in.
   c. To calculate Scale, divide the calculated linear travel (position-units) by the number of counts:
      1.2/4000 = **0.0003** position-units/count
4. If you wish to reverse the direction of the feedback, make the Scale negative.
5. Download the Scale to the RMC.

See Also

Scaling Overview | Scale/Offset Wizard | Position Scale

2.2.11. Scaling Rates on a Position Axis

Revolutions per Minute and Feet per Minute

If your system is using a position feedback device (such as an encoder), but your system is primarily intended for velocity control, and the system must be scaled so that the velocity of the system is given in feet per minute or revolutions per minute, then this topic is for you.

If you are using a feedback device that gives only velocity feedback, such as a tachometer, see the Analog Velocity Scaling topic instead.

To scale a position system to show the velocity in fpm or rpm, you need to set the Position Scale parameter. The Position Offset need not be set if you are only concerned with velocity and not positions.

Position Scale Parameter for a Linear Axis

1. Determine how many Counts the feedback device gives in one foot or one revolution.
2. Open the Scale/Offset Wizard. If it gives a choice, choose **Position/Counts**.
3. In the **First Measured Position** section, enter zero in both boxes.
4. In the **Second Measured Position** section:
    - Enter the number of counts per foot or revolution in the **Raw Counts** box.
    - Enter 60 in the **Measured Position** box.
5. Click **Next**. The calculated Position Scale will be displayed.
6. Click **Next** and complete the wizard.
7. Run the system and make sure that the Actual Velocity is correct.
8. Set the Display Units axis parameter to the desired units. For RPM, choose rpm·s. This will result in the velocity being displayed as rpm. For FPM, choose fpm·s. This will result in the velocity being displayed as fpm.

Position Scale Parameter for a Rotary Axis

To scale a rotary axis to RPM:

1. Set the Position Unwind to 60.
2. Set the Display Units axis parameter to rpm·s. This will result in the velocity being displayed as rpm.

See Also

Scale/Offset Wizard Overview

2.3. Tuning

2.3.1. Tuning Overview

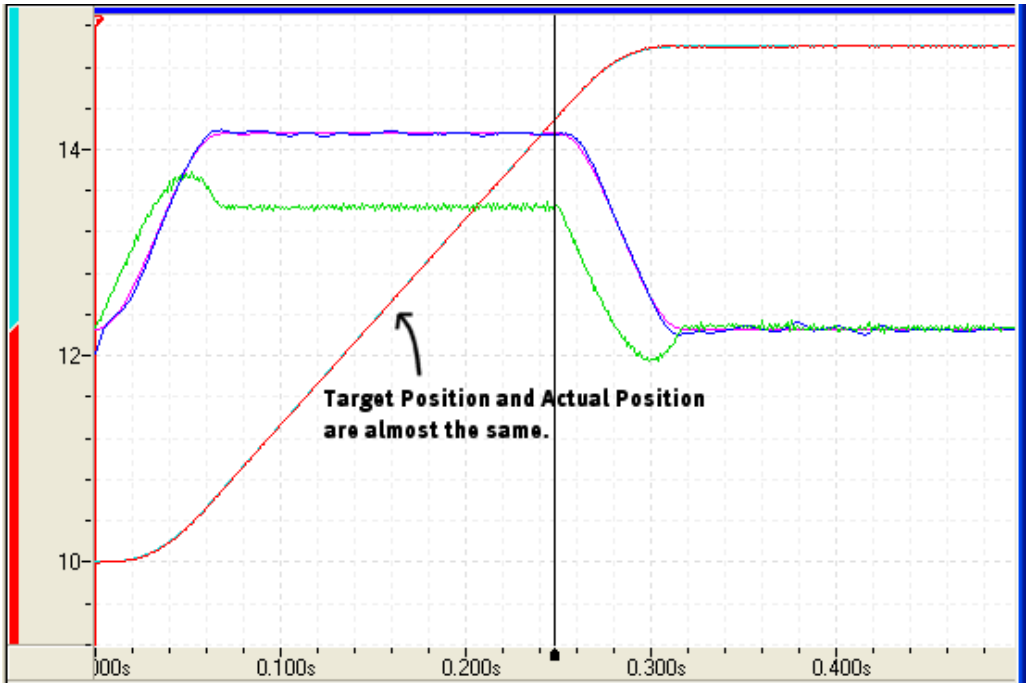
Once your system is set up and ready for use, it must be tuned in order to control it. Tuning is the process of adjusting the tuning parameters for optimum control of the system. The better tuned a system is, the closer the actual movement follows the desired path of movement.

Why is Tuning Necessary?

Tuning is required for moving an axis in closed loop control. In closed loop control, the RMC generates a target profile. The tuning parameters (gains) then dictate how much Control Output the RMC should generate each loop time in order to get the Actual Position to follow the target profile as closely as possible. The gains must be properly tuned to achieve precise motion.

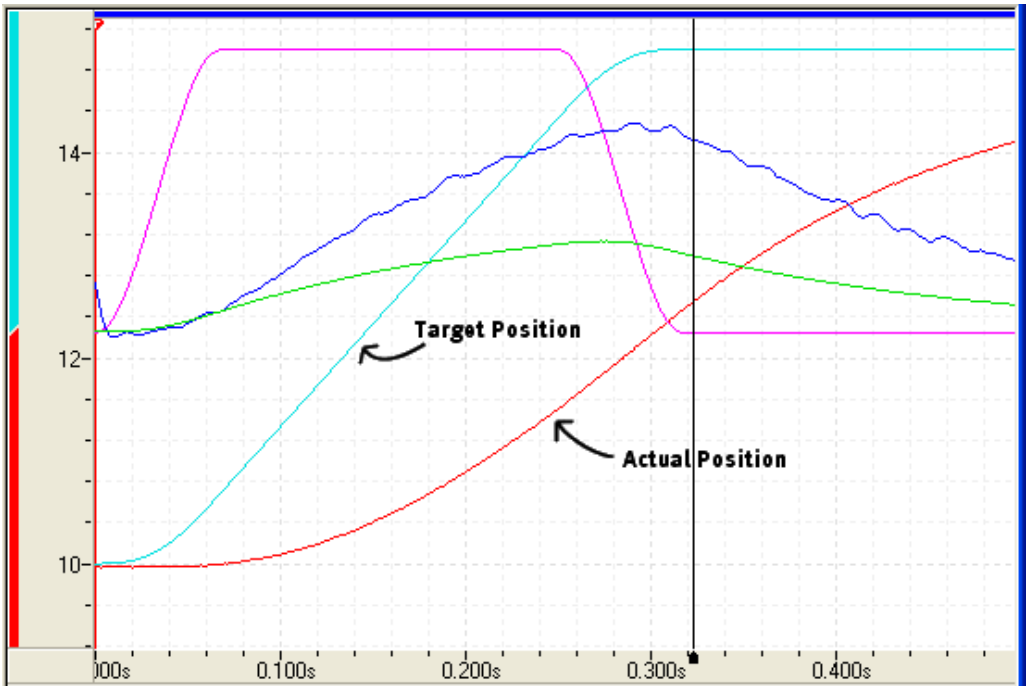
On a well-tuned system, the Actual Position will closely follow the Target Position. To see a plot, **click here**.

This is a plot of a system that is well-tuned:



On a poorly tuned system, the Actual Position will not follow the Target Position very well. To see a plot, **click here**.

This is a plot of a system that is poorly tuned:



Tuning Wizard and Autotuning

The Tuning Wizard makes the tuning process very easy. The Tuning Wizard provides *autotuning* and *tuning based on existing plot*.

Autotuning

Autotuning automatically moves the axis and then provides a range of gains from which you can choose with a simple slider bar. The RMC autotuning can be used on many systems. Even if autotuning is used, the user should posses a solid understanding of manual tuning to ensure proper tuning.

Autotuning can be used for position axes, including rotary and linear axes, and motors in velocity mode or torque mode. Autotuning does not support velocity, pressure, or force axes, although pressure or force axes can be tuned using an existing plot as described below.

See the Autotuning topic for details on how to perform autotuning.

Autotuning may not work for every situation. If it doesn't work for tuning your axis, you can use Tuning Wizard with an existing plot, as explained below.

Tuning Using an Existing Plot

If your axis cannot be autotuned, use this method to tune your axis:

1. Tune the axis using only the Proportional Gain.
2. Increase the Proportional Gain until the axis tracks well.
3. Move the axis in both directions, making sure to get a plot of each direction of motion.
4. In the Tuning Tools, click the **Tuning Wizard** button.
5. In the wizard, choose **Use Existing Plots** and continue.
6. In the wizard, choose the two recent plots you made of motion.
7. Complete the wizard, and the Gain Calculator will open. Now you can choose gains and move the axis.

See the Tuning Wizard topic for more details.

Note: Plots chosen for the **Use an Existing Plot** method should have been captured in capture mode. Do not use plots captured in trend mode, as they can cause inaccurate results.

Manual Tuning

Manual Tuning Procedures

Tuning procedures differ depending on the type of system. Please read the **Tuning Guidelines** below before continuing to any of the tuning procedures. Click the following links for suggested tuning procedures:

- Tuning a Hydraulic Position Axis or Motor in Velocity Mode **- the most common tuning procedure**
- Tuning a Motor in Torque Mode
- Tuning Position-Pressure or Position-Force
- Tuning Pressure/Force
- Tuning Active Damping and Acceleration Control
- Tuning a Pneumatic System

Manual Tuning Guidelines

Keep the following guidelines in mind throughout the tuning procedure. There is no substitute for experience when tuning an axis. The procedures offer some guidelines, tips, and suggestions for tuning your system. While the steps will work for many systems, they may not be the best for a particular system.

- **Use the Tuning Tools**
  The Tuning Tools in the Plot Manager provide a single place where you can issue repeated

commands and change and download gains. In addition, plots are automatically uploaded after a command.

- **Reiterate these Steps:**
  The tuning procedure is a reiteration of the following general steps. Use these steps throughout the tuning procedure:
  1. **Make a move**
     Typically, this is done using the Move Absolute (20) command. The Tuning Tools provide two buttons to easily issue two commands, typically one for each direction.

     Tip:
     For a typical hydraulic cylinder, the **Accel** and **Decel** parameters of the Move absolute command should be on the order of 20 -100 pu/sec^2. The speed is typically between 1 and 30 pu/sec.
  2. **View the Plot**
     When you issue a command in the Tuning Tools, the plot is automatically uploaded from the RMC. The plot will help you determine which parameters must be changed. In the Tuning Tools, you can easily disable the automatic plot upload for one of the command buttons. If you are focusing on tuning a certain direction, you can keep the plot from uploading when moving the other direction.
  3. **Change a Tuning Parameter**
     Change a gain in the Axis Parameters section of the Tuning Tools. After changing the value you must click the download button  or press Ctrl+D to download the changes to the RMC.
  4. **Repeat these steps**
     Repeat these steps until the parameter is at the desired value.

- **Use the Mean Squared Error**
  The Mean Squared Error in the RMCTools Plot Manager can help determine how the last parameter change affected the system. The Mean Squared Error provides a rough indication of how closely the Actual Position is tracking the Target Position. If this number decreases significantly, the last parameter change was good. If this number increases significantly, the last parameter change was bad.
- **Start with Long, Slow Moves, then Use Faster Moves**
  Begin the tuning procedure with fairly long, slow moves and low accelerations. This will prevent you from losing control of and potentially damaging the system. The moves should be short enough to fit in the plot.
  After gaining control of the system, you should increase the speed to values you expect to use during machine operation. Always tune your system at the speeds of intended operation.
- **Set Auto Stops**
  You may want to turn off some of the Auto Stop bits. The Auto Stops cause the axis to halt if an error occurs. In the initial stages of tuning, a Following Error or other error may occur, causing an undesired halt. Setting these bits to "Status Only" will make the RMC ignore the errors so you can tune the axis. Once you gain sufficient control of the axis, set the Auto Stop bits to halt. Turning off the Auto Stops may not be possible on some systems because of safety concerns.
- **Update Flash and Save the Project**
  When editing the parameters in the project, you must download them to apply the changes to the RMC. However, they are not updated in the RMC's Flash memory until you issue an Update Flash command. This will save the parameters even if power is disconnected. To save the RMC's parameters to the project, upload the parameters from the RMC and then save the project.

See Also

Tuning a Position Axis | Tuning a Motor in Torque Mode | Tuning Position-Pressure or Position-Force | Tuning Pressure/Force | Tuning Active Damping and Acceleration Control | Tuning a Pneumatic System

2.3.2. Tuning Wizard

The Tuning Wizard automatically calculates tuning gains for you. The wizard first guides you through the process of obtaining plots, then automatically calculates a range of valid gains, and finally presents you with a simple slider bar to choose conservative or aggressive gains.

The tuning wizard can be used with position, pressure and force axes. The tuning wizard cannot be used for velocity axes.

The Tuning Wizard works as follows:

1. Using plots of motion, the Tuning Wizard computes a mathematical model of the system. The Tuning Wizard provides two methods of obtaining plots:
    - **Autotuning Wizard - Position Axes Only**
      The Autotuning Wizard automatically moves the axis as specified by the user. With the plots generated by the motion, the Autotuning Wizard determines the system model.
    - **Using Existing Plots - Position, Pressure, and Force Axes**
      For position axes, the user chooses existing plots of motion from which the Tuning Wizard determines a system model. The existing plots must be listed in the Plot History pane.
      For pressure or force axes, the Tuning Wizard uses the currently displayed plot to determine a system model.

2. The Gain Calculator uses the system model to determines valid gains. In the Gain Calculator, the user positions a slider bar to pick from a range of appropriate gains for the system. The Gain Calculator stays open, making it easy to move the axis to try out various sets of gains. If a certain set of gains is not acceptable, simply move the slider and click **Apply** to try out a new set of gains.

Using the Tuning Wizard

Position Axes:

1. **Open the Tuning Wizard:**
   In the Plot Manager, on the **Tuning** tab, click **Tuning Wizard**.
2. Choose **Use Autotuning Wizard** if you would like RMCTools to move the axis and compute a system model. Choose **Use an existing plot** if you already have plots of motion of your system that can be used to compute a system model.
    a. **Autotuning**
       If you chose **Use Autotuning Wizard**, follow the instructions in the wizard. For more detailed information, click the **Help** button in the wizard.
    b. **Use an Existing Plot**
       If you chose **Select an existing plot**, you must already have plots of motion of your system in the Plot History pane. Make sure the Output Polarity was correct when the plot was captured, that is, that a positive Control Output results in movement in the direction of increasing position units.
       For details on generating plots suitable for the Tuning Wizard, see the Creating Plots for the Tuning Wizard topic.
3. After the wizard completes, it will open the Gain Calculator. For more details, see the Gain Calculator topic.

Pressure or Force Axes:

1. First, generate a plot that shows the pressure or force changing. To generate a good plot, do the following:

a. Set the Pressure/Force Proportional Gain to a small value. For systems scaled in pounds, 0.01 is often a good starting value.
b. Move the axis to where it encounters pressure or force.
c. Use the Hold Current Pressure/Force (19) command to enter pressure or force control.
d. Increase the Pressure/Force Proportional Gain until the Actual Pressure/Force starts moving toward the Target Pressure/Force. It should not get all the way toward to Target Pressure/Force. Give just enough gain to make sure it comes somewhat close.
e. Use the Ramp Pressure/Force (S-Curve) (41) command to ramp the pressure or force.
f. Upload the plot. Make sure the Actual Pressure or Force changed significantly in the plot. If it did not, repeat the process. For more details on generating plots suitable for the Tuning Wizard, see the Creating Plots for the Tuning Wizard topic.

2. In the Plot Manager, on the **Tuning** tab, choose the **Pressure** or **Force** tab and click **Tuning Wizard**.
3. Choose your plot from the list and click **Next**.
4. A dialog will appear with the calculated model. Click **Finish** to open the Gain Calculator. For more details, see the Gain Calculator topic.

See Also
Control Features Overview

2.3.3. Autotuning

Autotuning is part of the Tuning Wizard. Autotuning automatically moves the axis as specified by the user. With the plots generated by the motion, RMCTools determines the system model, from which it determines valid gains (if you prefer to use plots you took previously, the Tuning Wizard can generate a model from existing plots). The user then positions a slider bar on the Gain Calculator to pick from a range of appropriate gains for the system.

Autotuning can only be used to calculate control gains for position axes. It does not support calculating velocity, pressure, or force control gains.

Autotuning **can** be used for:

- Linear position axes such as with hydraulic, pneumatic, or other velocity mode motors or actuators.
- Rotary position axes with hydraulic, pneumatic, or other velocity mode motors or actuators.
- Rotary position axes with torque mode motors.

For position, pressure or force axes types that cannot be autotuned, the Tuning Wizard can still be used to calculate control gains based on plots manually generated by the user. See the Tuning Wizard topic for details.

Using Autotuning

Before autotuning, make sure you have completed all the pre-tuning steps in the Startup Procedure and all the pre-Tuning Tools steps in the tuning procedure you are using. Specifically, the axis must be properly scaled, the Output Polarity must be correct, the Positive and Negative Travel Limits must be set (for linear axes), and the axis must be able to move in open loop.

Motion control systems often do not behave identically at low speeds and high speeds. Therefore, it may be wise to use the Tuning Wizard as a first pass at tuning. Once the system is controllable, you can capture plots of motion at the intended speed and conditions of normal machine operation. You can then use these plots in the Tuning Wizard to compute a system model that more accurately represents the system during its normal operation.

To start autotuning:

1. Launch the Tuning Wizard:
    - From the Axis Tools, on the **Tune** tab of the **Axis Parameters** pane, in the **Tools and Wizards** section, on the **Position Tuning Wizard** row, click **Launch**.
    - Or from the Plot Manager, on the **Tuning** tab, click **Tuning Wizard**.
2. In the Tuning Wizard, choose **Use Autotuning Wizard**, and click **Next**.
3. Follow the instructions of the Tuning Wizard.

If Autotuning Doesn't Work

Autotuning may not work for every situation. If it doesn't work for tuning your axis, you can still take advantage of the Tuning Wizard and the Gain Calculator as follows:

1. Use the Tuning Tools to tune the axis with only the Proportional Gain.
2. Increase the Proportional Gain until the axis is tracking reasonably well.
3. Make sure to get plots of motion in both direction.
4. Open the Tuning Wizard and choose **Use an Existing Plot**.
5. Complete the tuning wizard, selecting the plots you made.
6. After the completing the wizard, the Gain Calculator will open. Use the slider bar to pick from a range of appropriate gains for the system.

See Also

Tuning Wizard | Gain Calculator

2.3.4. Creating Plots for Tuning

To successfully calculate valid gains using the **Tuning Wizard - Use Existing Plot** method, you must first generate good plots of motion. The plots should show some motion with changes. Make sure the plot does not include hitting the end, getting stuck, etc. Make sure the Output Polarity was correct when the plot was captured, that is, that a positive Control Output results in movement in the direction of increasing position units.

Plots of motion should:

- Contain significant motion. Slight changes in position, pressure, or force are usually not enough.
- Include the Control Output and the Actual Position, Actual Pressure, or Actual Force.
- Not include long sections of no motion.
- Contain motion in one direction only. If your position system behaves differently in each direction of motion, you will need to generate two plots—one for each direction of motion.
- Be captured at the smallest loop time possible and uploaded in capture mode. Do NOT use plots captured in trend mode, as they can cause inaccurate results.

Generating Plots

To generate a plot of motion:

1. Tune the axis using only the Proportional Gain.
2. Increase the Proportional Gain until the axis tracks well.
3. Move the axis in both directions, making sure to get a plot of each direction of motion.
4. In the Tuning Tools, click the **Tuning Wizard** button.

5. In the wizard, choose **Use Existing Plots** and continue.
6. In the wizard, choose the two recent plots you made of motion.
7. Complete the wizard, and the Gain Calculator will open. Now you can choose gains and move the axis.

If you are not able to move the axis with Proportional Gain, try using Open Loop commands to move the axis. With the Open Loop Rate (10) command, move the axis and stop it, then upload the plot. You may need to set the plot duration to be long enough to capture both the start and stop in the plot.

See Also

Tuning Wizard Overview | Autotuning Wizard

2.3.5. Gain Calculator

The Gain Calculator calculates gains for the axis based on the system model that was generated by the Tuning Wizard. After the Tuning Wizard completes, the Gain Calculator opens. Using a simple slider bar, the user can choose gains from conservative to aggressive. While the Gain Calculator is open, the user can use the Tuning Tools to send move commands to the axis to try various gains until satisfied.

If the axis has multiple gain sets, the Gain Calculator will apply the gains to the first gain set.

Using the Gain Calculator

Opening the Gain Calculator

In the Plot Manager, in the Tuning Tools, click **Gain Calculator**. If the axis does not have a model, you will be prompted to first run the Tuning Wizard. When the wizard completes, the Gain Calculator will open.

Choosing and Applying Gains

1. Move the slider bar to choose a set of gains. Always start with conservative gains.
2. Click **Apply Gains** to download the gains to the RMC.
3. Move the axis to test the gains. You can leave the Gain Calculator open while you make moves.
   Use the command buttons in the Tuning Tools to do this:
   1. On the **Tuning** tab, click one of the command buttons labeled **Click to set up**.
   2. Enter the position, velocity, acceleration, and deceleration for the Move Absolute command. For systems scaled in inches, typical Accel and Decel values are 10 to 100.
   3. Do the same for the other command button, using the same velocity, acceleration, and deceleration, but use a different position.
   4. You can click the buttons you just configured to move the axis back and forth to try your selected gains. The plots will automatically be uploaded so you can see how the motion is performing.
4. Repeat until you are satisfied that the axis is controlling well.

See Also

Tuning Wizard | Autotuning

2.3.6. Tuning a Position Axis

The following manual tuning procedure may be used to tune many position axes, hydraulic axes and motors in velocity mode. This procedure works for Position PID control. Please read the Tuning Overview topic before following this procedure. There is no substitute for experience when tuning an axis. This procedure offers some guidelines, tips, and suggestions for tuning your system. While these steps will work for some systems, they may not be the best for a particular system.

Before beginning the tuning procedure, set all the gains and feed forwards on the axis to zero.

Tuning Procedure

Before beginning the tuning procedure, make sure you have completed the steps in the StartUp Procedure up to the Tuning section. You must be able to move your actuator properly (a positive Control Output causes increasing position) and have valid feedback.

Pre-Tuning Steps:

These steps come before actually tuning the gains.

1. Test Wiring and Polarity

This step is for verifying that the system wiring and setup is correct before doing any closed loop control.

- In the Command Tool, issue a Direct Output (9) command to the axis. Use small **Output** value, such as 0.050-0.150 V. Use a **Ramp Rate** of 100. If the axis does not move, increase the **Output** until the axis begins to move.

  DANGER: The Direct Output (9) command disables the safety features on the RMC! Use this command carefully!

- A positive Control Output should yield increasing position units. If it does not, do the following:
  i. On the RMCTools toolbar, click the Axis Tools button.
  ii. In the Axis Parameters Pane, on the **Setup** tab, toggle the **Invert Output Polarity** parameter, then click the **Download** button.
- Issue a Direct Output (9) command again with a negative drive. This should yield decreasing position units.

2. Check the Deadband

If the axis exhibits a deadband, you may need to use the Deadband parameters.

Check Whether the System Exhibits a Deadband:

a. Give increasing amounts of **Output** to the axis with the Direct Output (9) command until the system starts to move.
b. The value of **Output** at which the system starts to move is your deadband. If this value is approximately 0.4 V or greater, or if the axis begins to move quickly at the deadband value, you should probably use the deadband parameters. If the point at which it begins to move it is less than 0.4 V, it is left to the discretion of the designer.

Set the Deadband Parameters:

If you found that your system has a deadband, set the deadband parameters in the following manner:

a. On the RMCTools toolbar, click the **Axis Tools** button.
b. In the Axis Parameters Pane, on the **All** tab, expand the **Output** section.

c. Set the Output Deadband parameter to the value of your deadband.
d. Set the Deadband Tolerance to a small value.
e. Click the **Download** button to apply the changes to the RMC.

3. Non-linear **V**alves

If you are using a non-linear hydraulic valve, set the valve linearization parameters as described in the Valve Linearization topic. Delta does not recommend using a non-linear valve if a linear valve is available.

4. Adjust the Output Bias

Some systems may drift significantly when the **Control Output** is at zero volts, which may adversely affect control. Use the **Output Bias** parameter to adjust the output such that the axis does not move when you issue an Open Loop command with zero volts **Control Output**. The RMC always adds the **Output Bias** to the **Control Output**.

Check if your system need Output Bias:

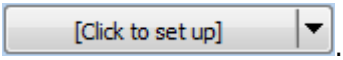
a. Issue an Open Loop Rate (10) command to the axis. Use an **Output** of zero and a **Ramp Rate** of 100.
b. If the axis moves significantly, you need to set the Output Bias.

Set the Output Bias:

a. In the **Axis Tools**, in the Axis Parameters Pane, on the **Tune** tab, enter a small number in the **Output Bias** parameter, such as 0.05 V or -0.05 V.
b. Keep increasing (or decreasing) the number until the axis stands still.
c. Click the **Download** button to apply the changes to the RMC.

5. Set Up the Tuning Tools

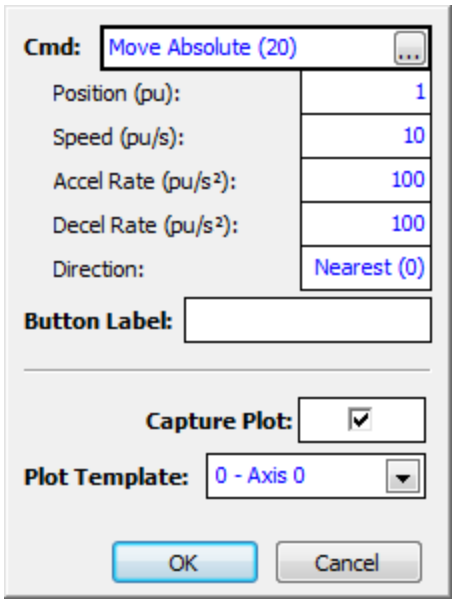
In the Tuning Tools, set up the command buttons so that one button will move the axis one direction, and the other will move it in the other direction.

a. Click the down arrow on the command button 

.
b. Enter the Position, Speed, Acceleration, Deceleration, and Direction (**Nearest** for linear axes) for the Move Absolute command, then click **OK**.

> **Tip:**
> For a typical hydraulic cylinder, the **Accel** and **Decel** parameters of the Move absolute command are typically on the order of 10 to 100pu/sec^2. The speed is typically between 1 and 30pu/sec.

Example:



c. Repeat the previous step for the other command button. Enter the same velocity, acceleration, and deceleration, but a different position.

6. Set Symmetrical/Ratioed

If your system behaves differently in each direction of motion, you will need to set the Symmetrical/Ratioed parameter is set to **Ratioed**. This will provide two Velocity Feed Forwards, one in each direction of motion. The other gains will be ratioed according to the Velocity Feed Forwards, resulting in the same control in both directions.

If you have a single-rod hydraulic cylinder, choose **Ratioed**.

The Symmetrical/Ratioed parameter can be accessed in the **Tuning Tools** on the **Tune** tab in the **Axis Parameters** section. Remember to download the changes by clicking the **Download** button.

Tuning Steps:

These steps are for adjusting the tuning gains. Use the Tuning Tools for these steps.

6. Adjust the Proportional Gain

The Proportional Gain must be adjusted to gain some control over the system for continuing the tuning procedure. You will fine-tune it later.

- Start with a small value of Proportional Gain. Remember that it is a floating point number, and you may have to start with a number smaller than 1, depending on your system. When you change the gain, click the **Download** button to apply the change to the RMC.
- From the Command Tool, issue the Move Absolute (20) command to move the axis.

> **Tip:**
> For a typical hydraulic cylinder, the **Accel** and Decel parameters of the Move absolute command should be on the order of 20 -100. The speed is typically between 1 and 30.

- Slowly increase the Proportional Gain as you make moves. Increase it until the following are true:
  - The Actual Position gets to the Command Position reasonably quickly.
  - During the constant speed portion of the move, the Actual Position parallels the Target Position.

If the system begins to oscillate, decrease the gain. In this step, do not expect the Actual Position to track the Target Position very well during the move.

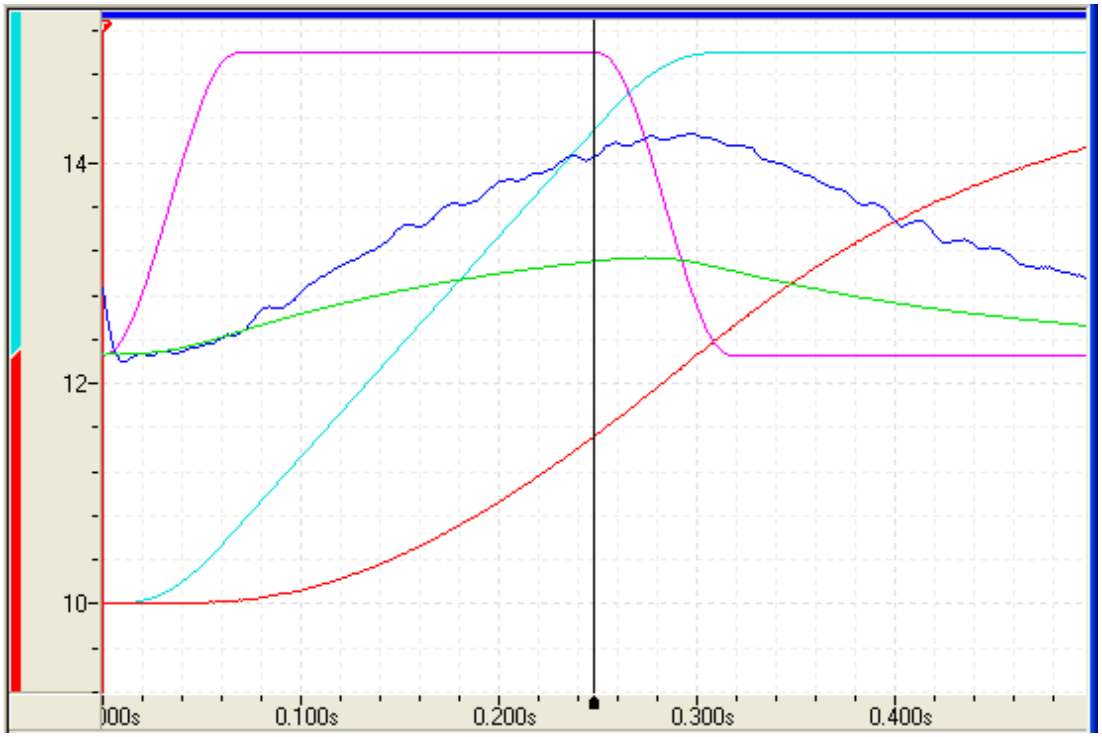
- Once you gain control over the system, increase the speeds and accelerations of your commands to values that will be used during normal machine operation.

Plots

Look at these plots for examples of good and bad plots at this step. Be aware that not all systems will be like these examples.

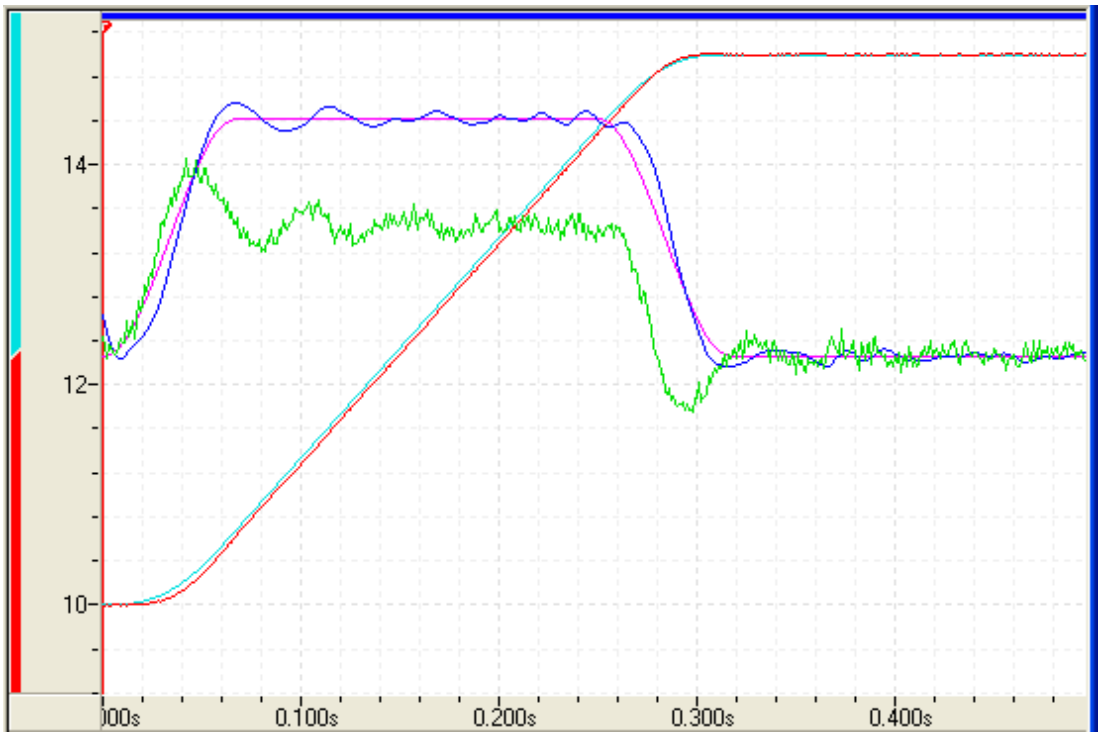
Too little Proportional Gain

The Actual Position lags the Target Position and takes a long time to get to the Command Position.



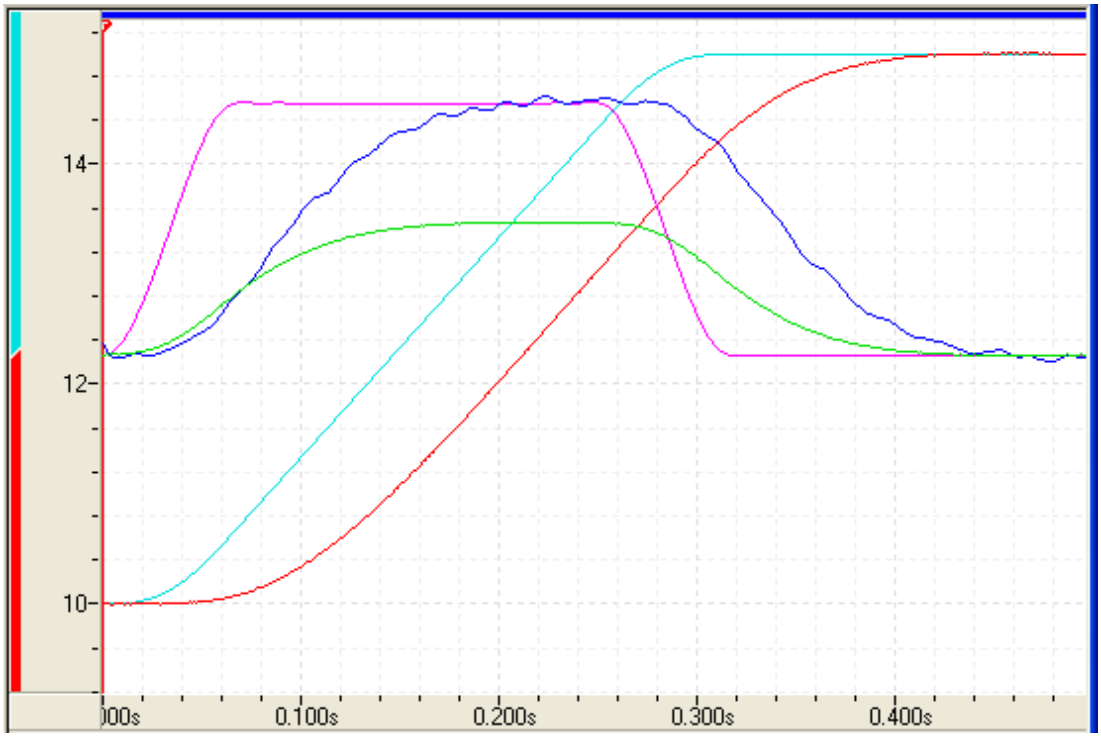
Too much Proportional Gain

The axis has started to oscillate, which is evident from the Actual Speed and Control Output.



Correct Proportional Gain

The Actual Position parallels the Target Position during the constant velocity portion of the move and does not overshoot.



7. Adjust the Velocity Feed Forward

In many systems the Velocity Feed Forward parameter is the most important parameter for position tracking during a move. To adjust the Velocity Feed Forward:

- Start with a small value of Velocity Feed Forward. Remember that it is a floating point number, and you may have to start with a number smaller than 1, depending on your system.

  Tip:
  Use the **Adjust VFF** button in the Tuning Tools to automatically determine the Velocity Feed Forward. Make a move, wait for it to complete, then click **Adjust VFF**. Repeat for the other direction.

- Make long slow moves in both directions. Adjust the Velocity Feed Forward until the axis tracks within 10% in both directions.
- If you are using Ratioed gains, make sure to adjust both Velocity Feed Forwards.
- Click the **Download** button to apply the changes to the RMC.

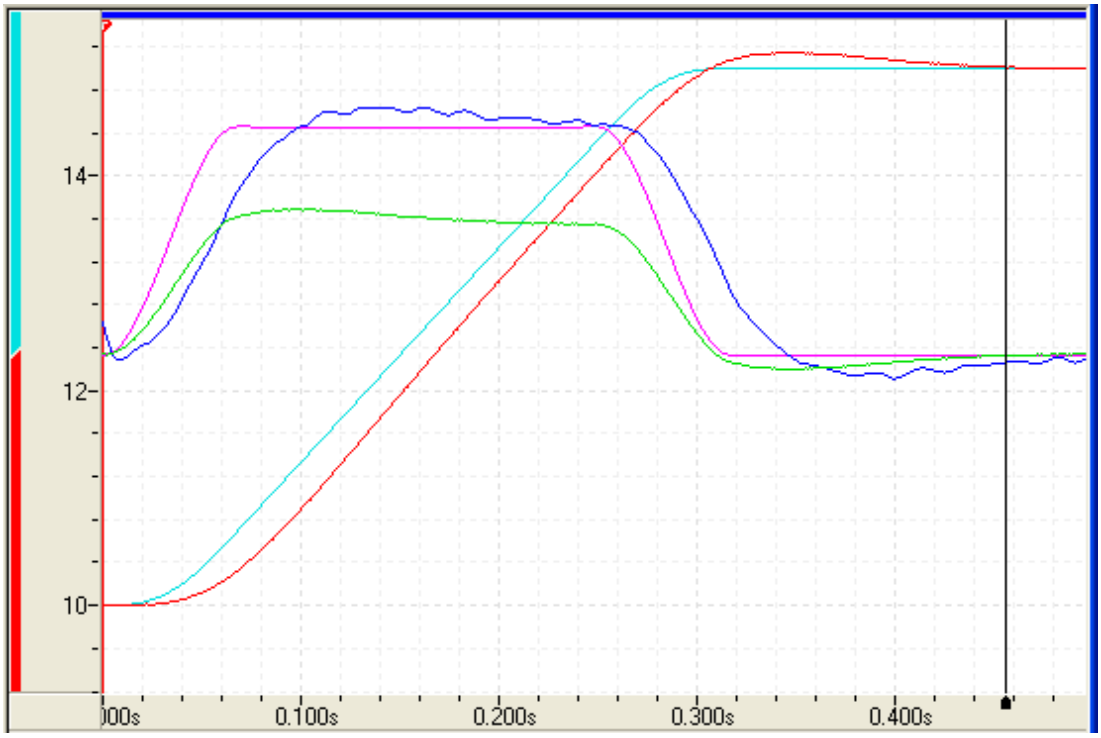
If the Symmetrical/Ratioed parameter is set to **Ratioed**, you will have two Velocity Feed Forwards, one for each direction of motion. Make sure to tune both.

Plots

Look at these plots for examples of good and bad plots at this step. Be aware that not all systems will be like these examples.

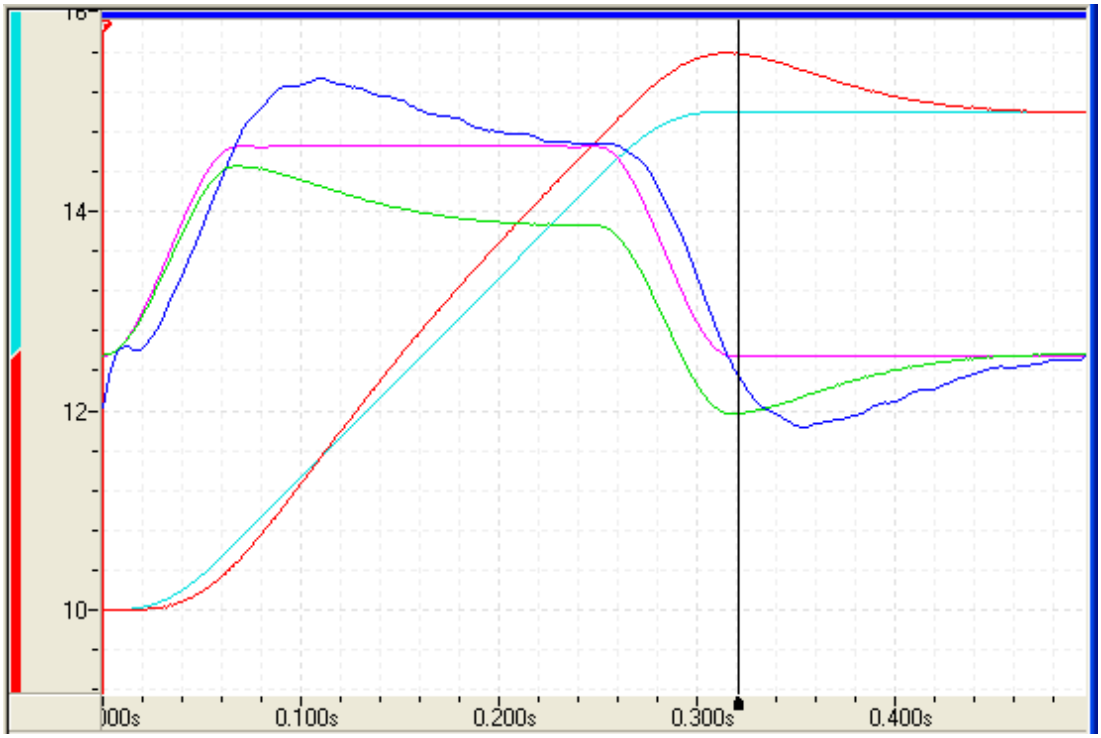
Too little Velocity Feed Forward

The Actual Position lags the Target Position during the entire move.

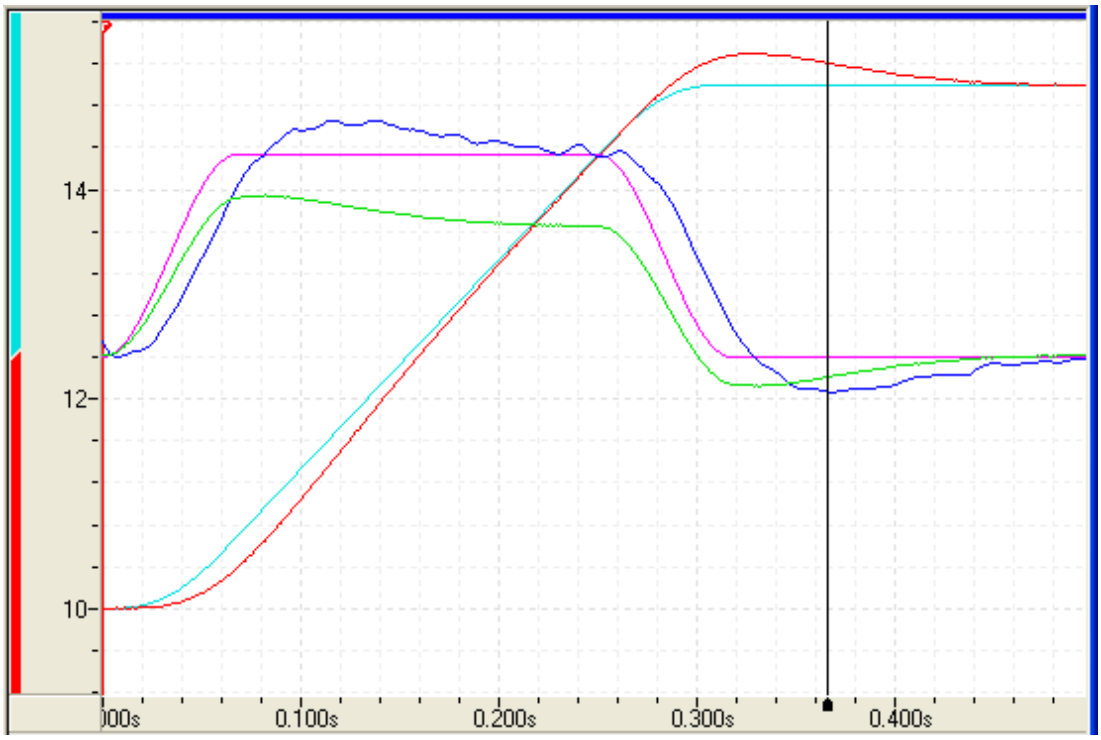


Too much Velocity Feed Forward

The Actual Position starts leading the Target Position.



Correct Velocity Feed Forward
The Actual Position tracks the Target Position perfectly during the latter half of the move.



8. Adjust the Acceleration Feed Forward

The Acceleration Feed Forward parameter is particularly useful for systems moving large masses with relatively small cylinders. Such systems often have a delay before the start of movement. The Acceleration Feed Forward terms can help compensate for this delay.

- Start with a small value of Acceleration Feed Forward. Remember that it is a floating point number, and you will very likely have to start with a number much smaller than 1, such as 0.001, depending on your system.

  Tip:
  For a 2.5 in bore hydraulic cylinder with a max velocity of 30 in/sec, the Acceleration Feed Forward is typically on the order of 0.01 to 0.3. Start with a small value.

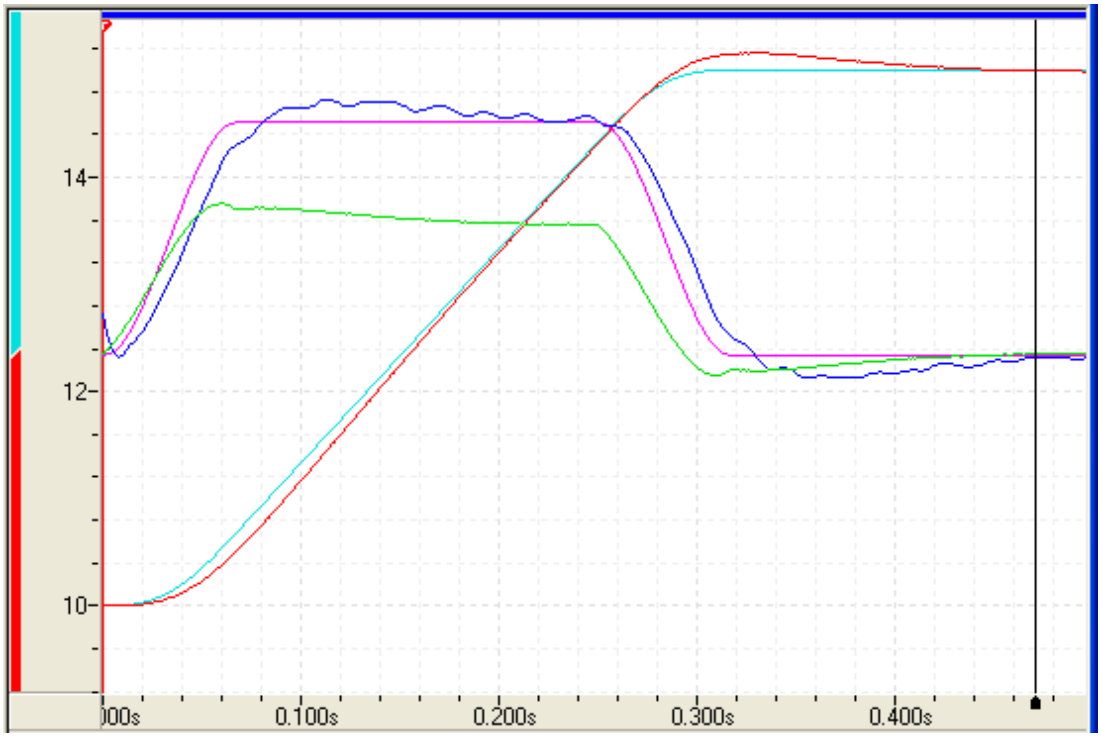
- Look for following errors during acceleration and deceleration. Increase the Acceleration Feed Forward parameter until the errors disappear.
- If you are using Ratioed gains, make sure to adjust both Acceleration Feed Forwards.
- Click the **Download** button to apply the changes to the RMC.

Plots

Look at these plots for examples of good and bad plots at this step. Be aware that not all systems will be like these examples.

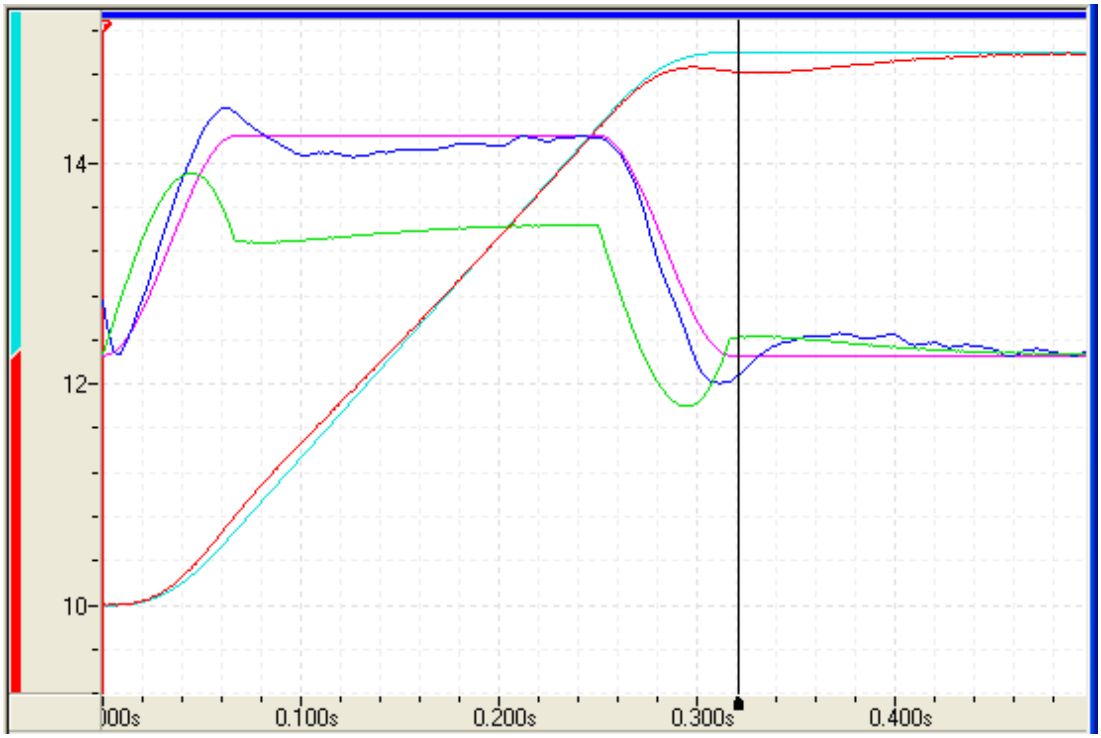
Too little Acceleration Feed Forward

The Actual Position lags the Target Position during the acceleration at the beginning of the move, and overshoots during the deceleration at the end of the move.



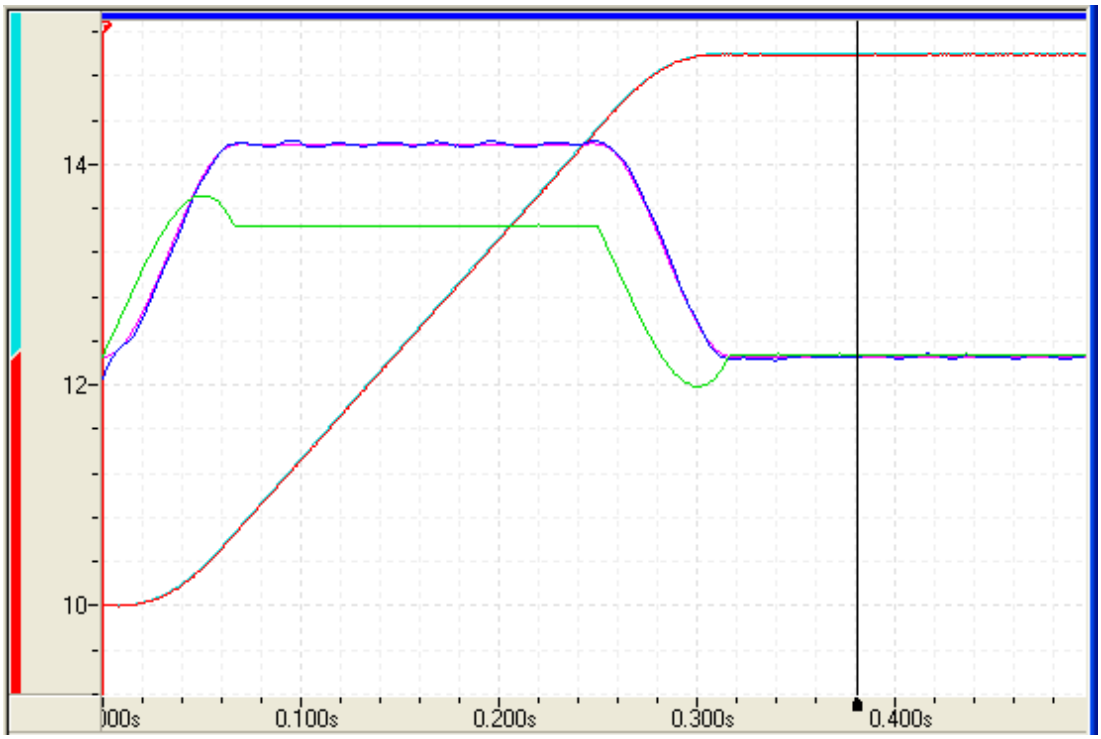
Too much Acceleration Feed Forward

The Actual Position starts leading during the acceleration and undershoots during the deceleration.



Correct Acceleration Feed Forward

The Actual Position tracks the Target Position well during the acceleration and deceleration.



9. Readjust the Proportional Gain

Proportional Gain affects the responsiveness of the system. Low gains make the system sluggish and unresponsive. Gains that are too high make the axis oscillate or vibrate.

- Slowly increase the gain. When you see a tendency to oscillate as the axis moves or stops, reduce the gain by 10 to 30 percent.

  Tip:
  For a 2.5 in bore hydraulic cylinder with a max velocity of 30 in/sec, the Proportional Gain is typically on the order of 20 to 300. Start with a small value.

- Click the Download button to apply the changes to the RMC.

10. Adjust the Integral Gain

Many hydraulic systems do not require a large Integral Gain. However, it is usually desirable to have some Integral Gain to help compensate for valve null drift or changes in system dynamics. Some systems may require larger Integral Gain, in particular if they are moving a large mass or are nonlinear. Too much Integral Gain will cause oscillations and overshoot. The Integral Gain is helpful for getting into position and for tracking during long, slow moves. It will not significantly affect tracking during short, fast moves.

Tip:
Typically, the Integral Gain should be 3 to 5 times greater than the Proportional Gain.

- Click the **Download** button to apply the changes to the RMC.

Too little Integral Gain:

The system may not get into position very quickly, and it will take a long time to recover from quick changes in the system, for example static friction, load changes, or obstacles.

Too much Integral Gain:

The system will begin oscillating.

Correct Integral Gain:

The system will get into position quickly, and any system changes will be quickly corrected.

11. Adjust the Differential Gain

Differential Gain may greatly enhance performance on many hydraulic systems. It is used mainly on systems that have a tendency to oscillate. This happens when heavy loads are moved with relatively small cylinders. Differential Gain will tend to dampen out oscillations and help the axis track during acceleration and deceleration. This will positively affect short, fast moves.

A disadvantage of Differential Gain is that it amplifies position measurement noise. If there is too much noise or the gain is too high, this can cause the system to chatter or oscillate. However, this can be compensated for by using the Output Filter. Systems that are difficult to tune can sometimes be drastically improved by using the Output Filter together with the Differential gain.

Many systems do not require any Differential Gain.

- Start with a small value of Differential Gain. Remember that it is a floating point number, and you will may have to start with a number much smaller than 1, depending on your system.

  Tip:
  For a 2.5 in bore hydraulic cylinder with a max velocity of 30 in/sec, the Differential Gain, if any, is typically on the order of 0.01 to 2. Start with a small value.

- Increase the Differential Gain. It may help the system track better. If it starts oscillating or chattering, decrease the gain.
- If the drive output during the constant velocity portion of the move is smooth, the Differential Gain is perhaps not set high enough.
- When the Differential Gain is properly adjusted, the drive output may look "fuzzy." This indicates that the drive is responding to the minute errors of the axis. Not all systems allow the differential gain to be set high enough for the drive to be "fuzzy".

Note:
If you use Differential Gain, you may be able to increase the Proportional Gain somewhat without making the system oscillate.

- If the system starts chattering or oscillating before it can achieve proper control, try using the Output Filter.
- Click the Download button to apply the changes to the RMC.

Too little Differential Gain:

The system will not keep up during acceleration and deceleration.

Too much Differential Gain:

The system will exhibit rapid oscillation.

Correct Differential Gain:

The system will track properly during acceleration and deceleration.

12. Adjust the Output Filter

If the axis is not tracking very well at this point, the Output Filter may help. It can significantly improve control of difficult systems. Without the Output Filter, the Differential Gain can cause the Control Output to oscillate, causing oscillation of the axis. By using the output filter, the Differential Gain can be increased significantly to help the Actual Velocity track the Target Velocity.

Typically, the Output Filter can be set to a value close to the natural frequency of the system. For example, if a system tends to oscillate at 10 Hz, a good starting value for the Output Filter is 10 or higher. After setting the Output Filter, you may be able to increase the Differential and Proportional Gains to improve control. If it does not help, try smaller and higher values of Output Filter, then try increasing the Differential and Proportional Gains again.

Keep in mind that the Output Filter does not always help.

13. Increase System Speed

If you have not yet done so, increase the Speed and Acceleration values of the moves. Look for following errors, overshoot, or oscillations.

- If an Output Saturated error occurs, there is not enough drive capacity to drive the axis at the requested Speed or Acceleration. Should this occur, reduce the Speed and/or Acceleration and Deceleration.
- If a Following Error occurs during acceleration and deceleration and adjusting the Gains and Acceleration Feed Forward does not help, the Acceleration and Deceleration ramps are too steep for the response of the system.
- If the actual position lags or leads the target position during the entire constant velocity section of the move, adjust the Feed Forwards.
- Should the system seem a little sloppy, try increasing the Proportional Gain.
- If the Control Output never gets very high, the gains can probably be increased for better control. If the Control Output is too high, or an overdrive error occurs, the system is not capable of performing the requested move. The Speed, and/or Accelerations may need to be decreased.
- If the system vibrates while in position, the gains may be too high, or the Dead Band value may need to be increased. However, if the oscillation is not caused by a deadband in the system, adjusting the Dead Band value will not help! A rule of thumb is to set the Dead Band Eliminator value to half of the peak-to-peak oscillation of the drive output while in position.
- The final tuning of the system should be made at the speed of intended operation.

14. Save your Settings to Flash

To retain your settings in the RMC in the event of a power loss, you must save the settings to Flash:

- On the **Controller** menu, click **Update Flash**.

See Also

Tuning Overview

Show All

2.3.7. Tuning a Pressure/Force System

The following procedure may be used to tune a pressure-only or force-only axis. For details on setting up a pressure-only or force-only axis, see Controlling Only Pressure or Force.

If you are tuning a position-pressure or position-force axis, see the Tuning a Position-Pressure or Position-Force System topic.

Please read the Tuning Overview topic before performing the tuning procedure.

There is no substitute for experience when tuning an axis. This procedure offers some guidelines, tips, and suggestions for tuning your system. While these steps will work for some systems, they may not be the best for a particular system.

Position/Pressure Tuning Procedure

Pre-Tuning Steps:

These steps come before actually tuning the gains.

1. Set the Pressure/Force Limits

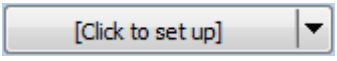
Set the Positive Pressure/Force Limit and Negative Pressure/Force Limit. These specify the pressure/force range in which the axis will be allowed to control. If the Target Pressure/Force exceeds these values, an error bit will be set and will halt the axis if the AutoStops are set to do so.

2. Set the AutoStops

Set the Pressure/Force Following Error AutoStop to Status Only to prevent following errors from halting the axis during the tuning. After the tuning is complete, set the AutoStops to values that are safe for machine operation.

3. Set Up the Tuning Tools

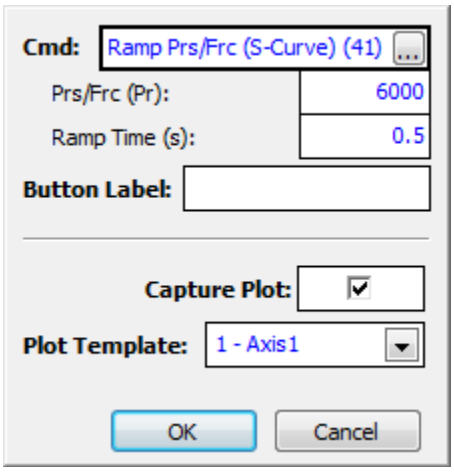
In the Tuning Tools, on the **Pressure** or **Force** tab, set up the command buttons so that one button will ramp the pressure or force in one direction, and the other button will ramp it in the other direction.

a. Click the down arrow on the command button 

.

b. Enter the **Pressure/Force**, and **Ramp Time**, then click **OK**.

Example:



c. Repeat the previous step for the other command button, and enter a different value for the **Pressure/Force**.

Tuning Steps:

1. Enter Pressure/Force Control

Set the Pressure/Force Proportional Gain to a small value, such as 0.01.

In the Tuning Tools, on the **Pressure/Force** tab, click the **Hold Prs/Frc** button to enter pressure/force control. Then click one of the buttons you set up to ramp the Command Pressure to the desired value.

You may wish to move the axis in open loop to where it encounters pressure/force before you click the **Hold Prs/Frc** button.

If the axis halts due to an Output Saturated error, reduce the Proportional Gain and repeat.

2. Adjust the Proportional Gain

Start with a small value. If the application is a hydraulic cylinder controlling force in pounds, a typical starting value may be 0.01 or smaller.

In the Tuning Tools, on the **Pressure/Force** tab, notice that the **Target Pressure/Force** and **Actual Pressure/Force** are displayed.

Keep increasing the Proportional Gain until the Actual Pressure/Force starts moving toward the Target Pressure/Force. It should not get all the way toward to Target Pressure/Force. Give just enough gain to make sure the Actual begins approaching the Target.

3. Adjust the Integral Gain

Add some Pressure/Force Integral gain, approximately twice as much as the Pressure/Force Proportional Gain. The Actual Pressure/Force should go towards the Target Pressure/Force.

Do not give a lot of Integral Gain, just enough to make sure it eventually gets to the Target Pressure/Force.

4. Ramp the Pressure/Force Up and Down

For the remainder of the tuning (steps 5, 6 and 7), repeat the following steps for each gain parameter you change:

1. Ramp the pressure/force with one of the buttons you set up in the Tuning Tools.
2. View the plot.
3. Change the gain.

Tip: You can use the Tuning Wizard to calculate a model and use the Gain Calculator to choose gains. After uploading a plot of pressure or force control, where the axis is controlling a changing pressure or force, click the **Tuning Wizard**.
If you use the Tuning Wizard, you do not need to continue manually changing the gains.

5. Adjust the Feed Forwards

Pressure/Force Rate Feed Forward

Ramp the pressure/force, and adjust the Pressure/Force Rate Feed Forward. Start with a small value. The Pressure/Force Rate Feed Forward should help the Actual Pressure/Force track the Target Pressure/Force during the ramp. Your system may not need any Pressure/Force Rate Feed Forward.

Pressure/Force Feed Forward

Most systems do not require the Pressure/Force Feed Forward. If your system exerts a pressure or force that is roughly linear to the amount of Control Output, you may need the Pressure/Force Feed Forward. To tune the Pressure/Force Feed Forward, move the Target Pressure/Force to various values, and adjust the Pressure/Force Feed Forward while the Target Pressure/Force is stationary. Do this for several value of pressure/force to ensure the best Pressure/Force Feed Forward.

6. Re-adjust the Proportional and Integral

Ramp the pressure/force, and re-adjust the proportional and integral gains. This should make it track better.

7. Adjust the Differential Gain

If you need better control, add some Differential Gain. The Differential Gain is typically a couple orders of magnitude less than the Proportional Gain.

8. Fine-tune the System

The final tuning of the system should be made at the rate and pressure/force range of intended operation. Look for following errors, overshoot, or oscillations, and consider the following:

- Should the system seem a little sloppy, try adjusting the Proportional Gain.
- If the actual pressure lags or leads the target position during the entire constant velocity section of the move, adjust the Feed Forwards.
- If an Output Saturated error occurs, there is not enough drive capacity to drive the axis at the requested rate of pressure/force change. Should this occur, increase the Ramp Time or decrease the speed of the system.
- If the Control Output is not high, the gains can probably be increased for better control, depending on system stability.
- Adding or changing the pressure/force feedback filter may help if noisy feedback is suspected.

See Also

Tuning Overview

Show All

2.3.8. Tuning a Position-Pressure or Position-Force System

The following procedure may be used to tune a system that uses pressure/force control or pressure/force limit. This procedure uses pressure/force control, but once an axis is tuned for pressure/force control, those gains can be used for pressure/force limit with few or no changes.

Please read the following topics before performing the tuning procedure:

- Tuning Overview
- Tuning a Position Axis

There is no substitute for experience when tuning an axis. This procedure offers some guidelines, tips, and suggestions for tuning your system. While these steps will work for some systems, they may not be the best for a particular system.

Position/Pressure Tuning Procedure

Pre-Tuning Steps:

These steps come before actually tuning the gains.

1. Set the Pressure/Force Limits

Set the Positive Pressure/Force Limit and Negative Pressure/Force Limit. These specify the pressure/force range in which the axis will be allowed to control. If the Target Pressure/Force exceeds these values, an error bit will be set and will halt the axis if the AutoStops are set to do so.

2. Set the AutoStops

Set the Position Following Error and Pressure/Force Following Error AutoStop to Status Only to prevent following errors from halting the axis during the tuning. After the tuning is complete, set the AutoStop to a safe setting for machine operation.

Tuning Steps:

1. Tune the Position Gains

The position gains should be tuned before attempting to tune the axis for pressure/force. Obtaining control of the axis' position greatly simplifies the tuning of the pressure/force gains. If you have not yet tuned the position gains, follow the procedure outlined in the Tuning a Position Axis topic before continuing.

2. Enter Pressure/Force Control

Set the Pressure/Force Proportional Gain to a small value, such as 0.01.

You may wish to move the axis to where it encounters pressure/force before you issue the Hold Current Pressure/Force (19) command.

Then enter pressure/force control with the Hold Current Pressure/Force (19) command. Issue the Ramp Pressure/Force (S-Curve) (41) or Ramp Pressure/Force (Linear) (42) or command to set the Command Pressure to the desired value.

If the axis halts due to an Output Saturated error, reduce the Proportional Gain and repeat.

3. Adjust the Proportional Gain

Keep increasing the Pressure/Force Proportional Gain until the Actual Pressure/Force starts moving toward the Target Pressure/Force. It should not get all the way toward to Target Pressure/Force. Give just enough gain to make sure it is beginning to.

4. Adjust the Integral Gain

Add some Pressure/Force Integral gain, approximately twice as much as the Pressure/Force Proportional Gain. The Actual Pressure/Force should go towards the Target Pressure/Force.

Do not give a lot of Integral Gain, just enough to make sure it eventually gets to the Target Pressure/Force.

5. Ramp the Pressure/Force Up and Down

For the remainder of the tuning, repeat the following steps:

1. Ramp the pressure/force.
   This can be done with the Ramp Pressure/Force (Linear) (42) or Ramp Pressure/Force (S-Curve) (41) command.
2. Change a gain.
3. View the plot.

Tip: You can use the Tuning Wizard to calculate a model and use the Gain Calculator to choose gains. After uploading a plot of pressure or force control, where the axis is controlling a changing pressure or force, click the **Tuning Wizard**.

The Tuning Tools make this very easy. In the Tuning Tools, set up the command buttons so that one button will ramp the pressure/force in one direction, and the other will ramp it in the other direction.

- In the Tuning Tools, click one of the buttons labeled **Click to set up**.
- Choose the Ramp Pressure/Force (Linear) (42) or Ramp Pressure/Force (S-Curve) (41) command and enter valid command parameters for your axis.
- Repeat the previous step on the other command button. Enter the same command, except for a different **Requested Pressure/Force**.
- Click **OK**. Now you will see that the command buttons are labeled.

To ramp the pressure/force, just click one of the buttons, and the plot will automatically be uploaded.

6. Adjust the Feed Forwards

Pressure/Force Rate Feed Forward

Ramp the pressure/force, and adjust the Pressure/Force Rate Feed Forward. Start with a small value. The Pressure/Force Rate Feed Forward should help the Actual Pressure/Force track the Target Pressure/Force during the ramp.

Pressure/Force Feed Forward

Most systems do not require the Pressure/Force Feed Forward. If your system exerts a pressure or force that is roughly linear to the amount of Control Output, you may need the Pressure/Force Feed Forward. To tune the Pressure/Force Feed Forward, move the Target Pressure/Force to various values, and adjust the Pressure/Force Feed Forward while the Target Pressure/Force is stationary. Do this for several value of pressure/force to ensure the best Pressure/Force Feed Forward.

7. Re-adjust the Proportional and Integral

Ramp the pressure/force, and re-adjust the proportional and integral gains. This should make it track better.

8. Adjust the Differential Gain

If you need better control, add some Differential Gain. The Differential Gain is typically a couple orders of magnitude less than the Proportional Gain.

9. Tune the Transition

Now that both position and pressure/force are tuned, you can tune the transition between position and pressure/force control or pressure/force limit.

Issue the commands for transitioning between position and pressure/force as you intend to do during normal machine operation. For some applications, this may involve creating a simple user program.

View the transition on a plot. If the pressure/force drops off at the transition, increase the Integrator Preload parameter in the Enter Pressure/Force command.

10. Fine-tune the System

The final tuning of the system should be made at the rate and pressure/force range of intended operation. Look for following errors, overshoot, or oscillations, and consider the following:

- Should the system seem a little sloppy, try adjusting the Proportional Gain.
- If the actual pressure lags or leads the target position during the entire constant velocity section of the move, adjust the Feed Forwards.
- If an Output Saturated error occurs, there is not enough drive capacity to drive the axis at the requested rate of pressure/force change. Should this occur, increase the Ramp Time or decrease the speed of the system.
- If the Control Output is not high, the gains can probably be increased for better control, depending on system stability.
- Adding or changing the pressure/force feedback filter may help if noisy feedback is suspected.

See Also

Tuning Overview

2.3.9. Tuning a Motor in Torque Mode

When tuning a motor in torque mode, use the basic instructions in the Tuning a Position Axis topic, but use the order of the gains below.

Tuning Order

Torque motors generally do not have much damping. Damping must be provided for the system, or it will be difficult to control. Providing some Differential Gain will effectively dampen the system. Therefore, the Differential Gain must be set early in the tuning process:

Method 1:

a. Set all the gains to zero.
b. Increase the Proportional Gain to the point where the motor moves somewhat when given a Move Absolute (20) command. You should notice that the axis has a tendency to overshoot.
c. Increase the Differential Gain so that the axis ceases overshooting. If the axis begins oscillating or chattering, decrease the Differential Gain.
d. Continue tuning the gains in the order given in the Tuning a Position Axis topic.

Method 2:

a. Set all the gains to zero.
b. Issue the Hold Current Position (5) command to the axis.
c. Increase the Differential Gain until the motor start humming (or chattering or oscillating), then decrease the Differential Gain significantly, perhaps up to 50%, to avoid oscillation later when making moves.
d. Continue tuning the gains in the order given in the Tuning a Position Axis topic.

Small Motors

For very small motors that can be rotated by hand, the Differential gain can be set before any other gains.

a. Set all the gains to zero.

b. Set the Differential Gain to a small value, then issue the Hold Current Position (5) command to put the axis in closed loop control.
c. Rotate the motor manually to get a feel for the resistance to movement (damping). Increase the Differential Gain until the damping is significant. If the motor chatters or oscillates, decrease the gain.
d. Continue tuning the gains in the order given in the Tuning a Position Axis topic.

See Also

Tuning Overview | | Torque Mode | Tuning Position

2.3.10. Tuning a Pneumatic System

When tuning a pneumatic cylinder, use the basic instructions in the Tuning a Position Axis topic, but use the order of the gains below.

Tuning Order

When using high-order gains, increasing the highest order will allow you to increase lower order gains. However, you need to first set the lower gains to some value before you can set the higher ones. Then, you can go back and increase the lower order gains.

The Output Filter can significantly improve control of pneumatics. Without the Output Filter, the Differential Gain can cause the Control Output to oscillate, causing oscillation of the axis. By using the output filter, the differential gain can be increased significantly to help the Actual Velocity track the Target Velocity.

Suggested Tuning Order:

a. Proportional Gain
b. Output Filter
c. Differential Gain
d. Double Differential Gain
e. Triple Differential Gain
f. Double Differential Gain
g. Differential Gain
h. Proportional Gain
i. Integral Gain
j. Repeat several times.

Feed Forwards are not useful on poor systems. They may be more significant on low-friction cylinders. If your application requires Feed Forwards, set them before setting the Integral Gain.

The P, D, DD and TD gains typically differ from each other by an order of magnitude. For example, the ratio of the gains to each other may be as shown below. Keep in mind that the actual value of gains required on your system may differ significantly.

P: 10
I: 10
D: 0.1 or smaller
DD: 0.01 or smaller
TD: 0.001 or smaller

Other tips:

- Use the Output Filter, setting it to a low value, comparable to the natural frequency of the system.
- Deadband may be required on systems with high static friction.
- The Velocity and Acceleration may need to be filtered, or using the model may help.

See Also

Tuning Overview

2.3.11. Tuning Active Damping and Acceleration Control

When tuning Active Damping or Acceleration Control, use the basic instructions in the Tuning a Position Axis topic, but use the order of the gains below.

If you are tuning a pneumatic cylinder, see the Tuning a Pneumatic System topic.

Tuning Order

Systems with High Damping

For system with high damping, you can follow the procedure for Tuning a Position Axis, then set the higher-order gains.

High damping means that the system does not oscillate and stops very quickly as soon as 0 volts is applied.

Systems with Low Damping

For system with low damping, you will need to set the higher-order gains earlier in the tuning procedure to gain any control at all. The higher order gains will provide the damping. When using high-order gains, increasing the highest order may allow you to increase lower order gains. However, you need to first set the lower gains to some value before you can set the higher ones. Then, you can go back and increase the lower order gains.

Suggested Tuning Order:

a. Proportional Gain
b. Differential Gain
c. Double Differential Gain or Active Damping Proportional Gain
d. Triple Differential Gain or Active Damping Differential Gain
e. Double Differential Gain or Active Damping Proportional Gain
f. Differential Gain
g. Proportional Gain
h. Feed Forwards
i. Integral Gain
j. May need to repeat several times.

See Also

Tuning Overview

3. Controller Features

3.1. RMC Controller Features

The RMC75, RMC150, and RMC200 motion controllers provide a host of features to successfully control any motion application. Browse the sections below for descriptions and links to the many features.

Motion Types

- Point-to-Point (Absolute and Relative)
- Open Loop
- Quick Move
- Gearing
- Velocity
- Sinusoidal
- Curves (Cams, Splines)

Control

Types

- Open Loop Control
- Closed Loop Control
- Position PID
- Position I-PD
- Velocity PID
- Velocity I-PD
- Cascade Control
- Unidirectional Mode
- Pressure/Force Control
- Pressure/Force Limit

Controlled Quantities

- Position
- Velocity
- Pressure
- Force
- Torque

Dual-Loop Control

- Position-Pressure
- Position-Force
- Velocity-Pressure
- Velocity-Force
- Position-Acceleration

Communications (Slave Only)

- Ethernet
- PROFIBUS
- Serial RS-232/485 (RMC75)
- Address Maps

Transducer Interfaces

- Magnetostrictive Start/Stop and PWM
- SSI
- Analog (±10V, 4-20mA)
- Load Cell
- Quadrature
- Resolver

Custom Feedback

- Custom Feedback
- Switching Feedback on the Fly
- Feedback Linearization
- Redundant Feedback

Axis Types

- Control Axes
- Reference Axes (Half-Axes)
- Rotary Axes

Motion Safety Features

- Halts
- Auto Stops
- Enable Output
- Fault Input

Tuning

- Tuning Tools
- Autotuning
- Tuning Wizard
- Gain Calculator
- Gain Scheduling

Programming

- Pre-programmed Commands
- User Program Overview
- Variables
- Retentive Variables (RMC75E and RMC150E only)
- Program Triggers

Diagnostic Tools

- Plots and XY Plots
- Event Log
- Status Registers

Discrete I/O

- Discrete I/O

Velocity-Acceleration

Control Output

RMC75/150: ±10 V Servo Output

RMC200: ±10 V, 4-20 mA, or ±20 mA Servo Output

High-Order

Active Damping

Acceleration Control

High-Speed Functions

Homing

Registration

Event Timers

3.2. General

3.2.1. Firmware

Firmware is the software that resides in the controller. Each RMC is shipped with the latest version of firmware. Delta regularly releases new versions of firmware that include bug fixes and support for new features.

Each command topic in the RMCTools help lists the firmware version required for the command. Other topics may also specify which firmware versions support the feature.

Updating Firmware

To update the firmware, the RMC must be connected to the computer running RMCTools via USB, Ethernet, or the RS-232 Monitor Port. The second serial port on the RMC75S will not work for updating firmware. To update firmware via Ethernet, the **Allow updating firmware over Ethernet** box must be checked in the Ethernet Settings Page.

To update the firmware in the RMC:

1. Go to Delta's website at https://deltamotion.com.
2. On the **Downloads** page, find the firmware for your module, and save it on your computer.
3. In RMCTools, go online with your controller.
4. In the Project pane, expand the **Modules** folder, double-click the module you wish to update and click **Firmware**. For the main firmware, the module will typically be the CPU.
5. In the **Control Program** section, click **Update Firmware**. This will open the Firmware Update Wizard. Follow the instructions to complete the firmware update.

Backing Up Firmware (RMC75/150 Only)

As part of the firmware update process, the firmware in the RMC75 and RMC150 can be backed up before the new firmware is downloaded to the RMC. Delta recommends backing up the firmware in case you need to revert to that firmware version. To back up the firmware, use the Firmware Update Wizard as described above. Before the download occurs, you will be given the option of backing up the firmware.

Firmware Base Version

Each firmware release made for the RMC is identified primarily by a two- or three-part version number such as ”3.30.0”. Each firmware release may include more than one actual firmware

image, with each image having a separate Configuration ID. When a controller is loaded with firmware, RMCTools chooses the appropriate firmware image to load into the controller.

Special Releases

In some cases, Delta will release firmware that differs in some way from the generally-released firmware. These are designated by an additional *special release code*. For example, the ”S4” special firmware has ”S4” append to the version number, as in ”2.72A-S4”. For more information on the S4 special firmware in particular, see the **Firmware Compatibility** section below.

RMC75 Firmware and Hardware Compatibility

The RMC75 firmware has the following hardware compatibility limitations:

Version	RMC75S	RMC75P	RMC75E	Notes
3.00 and above	2.1D or newer	2.1E or newer	✓	See Note 1
2.72-S4 to 2.99-S4	2.1C or older	2.1D or older	-	See Note 2
2.30 to 2.71	✓	✓	✓	See Note 3
1.00 to 2.21	✓	✓	✓	

Note 1: These firmware releases no longer support older RMC75S and RMC75P controllers, which cannot support new features.

Note 2: These firmware releases are reserved for patch releases for the older RMC75S and RMC75P controllers, which cannot support new features and are therefore not included in the 3.00 and later releases. This firmware is assigned the ”S4” special release code.

Note 3: These firmware releases support all RMC75 controllers, however, new commands added by these releases are not included in firmware for older RMC75S and RMC75P controllers.

See Also

Help Overview

3.2.2. Loop Time

The RMC is a deterministic controller. It reads the inputs, computes the control algorithms and updates the outputs at a specific interval. This interval is called the controller **loop time**, in reference to the way the controller repeatedly "loops" through its code.

The RMC will always run at its loop time setting. When it finishes all the calculations for one loop, it waits until the next loop time before doing its calculations again.

Setting the Loop Time

To set the loop time of the RMC, use the Control Loop Page.

Available Loop Times vs Axes

The loop times available on the RMC controllers are listed below, along with the expected number of axes supported by that loop time with a full load of motion commands, user programs, communications and plots. The user program time allocation is also affected by loop time, as described in Program Capacity and Time Usage.

High Control Loop Utilization

If you wish to use a loop time with more axes than listed below as being supported with a full load of motion commands, user programs, communications and plots, then you must enable **High Control Loop Utilization** in the Control Loop Page of the controller properties dialog.

The **High Control Loop Utilization** option does the following:

- Allows selecting faster loop times given the number of control axes defined in the controller.
- Allows downloading user programming with estimated worst-case processing times that exceed the allocated time. See Program Capacity and Time Usage.

If you choose **High Control Loop Utilization**, then you must ensure that your application will run within the selected control loop time. See the **Exceeding the Loop Time** and **Monitoring the Loop Time** sections below.

RMC75 Loop Times

Loop times that support all axes configurations are indicated by a check mark ✓.

	Loop Time				
	4000 µs	**2000 µs**	**1000 µs**	**500 µs**	**250 µs**
RMC75E	✓	✓	✓	✓	1 axis only
RMC75S	✓	✓	✓	1 axis only with up to one reference input	—
RMC75P (2.1F or newer)	✓	✓	✓	1 axis only with up to one reference input	—
RMC75P (2.1E or older)	✓	✓	1 axis only with up to one reference input	—	—

RM150 Loop Times

Loop times that support all axes configurations are indicated by a check mark ✓.

	Loop Time				
	4000 µs	**2000 µs**	**1000 µs**	**500 µs**	**250 µs**
RMC150E	✓[1]	✓	✓	1 or 2 axes only[2]	1 axis only[3]

[1] 4000 µs requires firmware 3.33.0 or newer or 3.36.0 if MDT modules are present. It is not supported by RMC150E/RMC151E CPU revision 1.2B or older or MDT (-M) modules revision 6.0 or older.

[2] 500 µs is not supported by SSI (-S) modules revision 5.0 or older, or Analog Input (-A, -H, and -G) modules revision 6.0 or older.

[3] 250 µs is not supported by SSI (-S) modules revision 5.0 or older, or any Analog Input (-A, -H, and -G) modules.

RMC200 Loop Times

In the tables below:

- **All** refers to the maximum number of axes supported by the controller, as listed in the Axis Types Overview.
- **Full Load** includes control, user programs, plots and communications.
- **Limited Load** indicates the range of axes possible in applications using less than a full load. Using axes in this range requires **High Control Loop Utilization** to be enabled.
- A **control axis** in this context refers to an axis with at least one PID loop, and includes cascading outer loop axes.

CPU20L

	Expected Number of Axes Supported with:	
Loop Time	**Full Load**	**Limited Load**

125 µs	0 axes	Up to 3 control axes (or 2 dual-loop)
250 µs	0 axes	Up to 7 control axes (or 5 dual-loop)
500 µs	Up to 11 control axes (or 8 dual-loop)	All
1000 µs	All	All
2000 µs	All	All
4000 µs	All	All

CPU40

	Expected Number of Axes Supported with:	
Loop Time	**Full Load**	**Limited Load**
125 µs	0 axes	Up to 6 control axes (or 4 dual-loop)
250 µs	0 axes	Up to 11 control axes (or 8 dual-loop)
500 µs	Up to 14 control axes (or 10 dual-loop)	Up to 32 control axes (or 23 dual-loop)
1000 µs	Up to 50 control axes (or 36 dual-loop)	All
2000 µs	All	All
4000 µs	All	All

When is High Control Loop Utilization Required?

For the RMC200 controller, **High Control Loop Utilization** is required if the estimated axis processing time exceeds the processing time allowed for the selected loop time. The Axis Definitions dialog automatically provides a graphic indication of the estimated axis processing time and whether **High Control Loop Utilization** is required. The following tables are used to determine whether or not High Control Loop Utilization is required:

Maximum Total Axis Processing Time Allowed at Full Load

Loop Time	CPU20L	CPU40
125 µs	0 µs*	52 µs
250 µs	0 µs*	100 µs
500 µs	152 µs	302 µs
1000 µs	302 µs	not limited
2000 µs	not limited	not limited
4000 µs	not limited	not limited

*This means **High Control Loop Utilization** is always required.

Estimated Processing Time per Axis

Axis Type	CPU20L	CPU40
Single-loop Control Axis	11.9 µs	10.2 µs
Dual-loop Control Axis	16.0 µs	14.0 µs
Virtual Axis	7.9 µs	6.3 µs
Reference Axis	3.2 µs	2.5 µs

Output Only Axis	1.5 µs	1.2 µs

Notice that the estimated per-axis processing times shown above are conservative estimates assuming maximum processing, for purposes of determining whether High Control Loop Utilization is required. Actual processing time for axes will often be significantly less than these times indicate. However, the user should always verify that the application does not exceed the loop time excessively, as described below.

Exceeding the Loop Time

How the loop time works

The loop time is the pre-selected interval at which the RMC reads its inputs, processes motion commands and user programs, computes the control algorithms and updates the outputs. When it finishes all the processing for one loop, it waits until the next loop time before doing its calculations again. The amount of calculations can vary significantly from loop to loop, based on many factors including the commands received, user program calculations, and communications load. It is common that most of the actual loop times are low, and there are some loops that require much more time, such as during the loop time in which a motion command is received by an axis, which requires more processing.

Running over the loop time

If the loop processing takes longer than the specified loop time, the RMC will finish the processing and then immediately start on processing the next loop. The start of this next loop is therefore delayed, so the updating of outputs can also be delayed. If the loop time has been exceeded by only a small amount, such as 20%, then this delay is typically negligible, and the RMC will usually catch up with its processing in the next loop time. If the loop time has been exceeded by more, such as 75%, the delay is larger, but still usually negligible relative to the responsiveness of the physical system, especially if the RMC is able to catch up with its processing in the next loop time.

Do not exceed loop time by 100%

It is very important that the loop time is not exceeded by 100% or more. If it is, that loop time will be lost, and the RMC's calculations will not take it into account. This can cause the RMC's internal time calculations to lose one loop time, and the target motion profile calculations will be slowed by one loop time. If many loop times are lost, the target motion profiles may be significantly off, which could cause machine damage.

If consecutive loop times are exceeded such that the cumulative exceeded loop time is 100% or more, the RMC will also lose a loop time. It is important that this does not occur.

Event Log Reporting

Loop time overage of less than 80% is reported in the Event Log as a warning that can be filtered out. Loop time overage of 80%, or cumulative loop time overage of 80% or more is reported in the Event Log as an error.

In summary

- Occasionally exceeding an individual loop time by a small amount should not cause any adverse effects.
- It is important to avoid exceeding a loop time by 100%. The RMC reports an error at 80%.
- It is important to avoid cumulatively exceeding a loop time by 100%. The RMC reports an error at 80%.

Monitoring the Loop Time

Monitor the loop time in the following ways:

- **Control Loop Page**
  On this setting page, you can view how much of the control loop the RMC used in the last loop, along with the maximum value since the RMC powered up or since the **Clear** button was clicked. See Control Loop Page.

- **Plotting Loop Time Registers**
  To clearly visualize the loop time usage of the RMC, you can add the loop time usage registers to a plot. See Plotting the Loop Time for details.

Reducing the Loop Time Used

If the RMC is exceeding the loop time, you can attempt to reduce the loop time used, or select a larger loop time. To reduce the loop time used, you can plot the loop time registers as described in Plotting the Loop Time to determine which part of the loop is taking the most time, then use one or more of the following tips to reduce the loop time:

User Programs

- Reduce the programming loop time used by following the tips in the Program Capacity and Time Usage topic.

Commands:

- **Stagger simultaneous commands**
  When a command is received, the RMC performs some initial calculations in that loop time, which can be extensive. Sending many commands simultaneously can therefore result in a large loop time load. If it is possible to stagger the commands, the total loop time will not be as large.
- **Do not use the Transition commands**
  The Transition Rate (56) command causes extra processing, not when it is sent, but when a transition is active, such as during the beginning of sine or curve motions. Removing transitions may possibly reduce the loop time load.
- **Do not use Dynamic Retargeting**
  Dynamic Retargeting is the continuous sending of a Move Absolute (20) or similar commands, such as every loop time. This causes a heavy processing load.

Plots:

- **Reduce the number of captured plot items**
  Having a very large number of captured plot items may impact the loop time. By default, each axis has a corresponding plot that triggers when a command is sent to the axis. Reducing the number of plotted items, or changing the settings so that not as many plots are automatically triggered may slightly reduce the loop time load.

Communications:

- **Reduce the Cyclic I/O**
  A large amount of cyclic EtherNet/IP I/O or PROFINET communications at very short intervals can increase the loop time. Reducing the Requested Packet Interval, the number of connections, or the amount of data may reduce the loop time load.

Using the Loop Time in Expressions

Certain calculations in user programs, especially for calculating time, may require accessing the current set loop time. The **_LoopTime** tag can be used to access the loop time when using expressions. _LoopTime is a REAL, and is specified in seconds.

See Also

Program Capacity and Time Usage | Plotting the Loop Time | Control Loop Page

3.2.3. Plotting the Loop Time

To clearly visualize the loop time usage of the RMC, you can add the following Controller Loop Time Usage Registers to a plot to monitor the loop time:

Register Name
Units in seconds
Loop Time Used, Last
Loop Time Used, Maximum
Loop Time Used, Minimum
Units in microseconds
Loop Time Used, Total
Loop Time Used, Axes
Loop Time Used, Programs
Loop Time Used, Plots
Loop Time Used, Comm
Loop Time Used, Overhead

To add these registers to a plot:

1. In the Plot Template Editor, create a new plot template or choose an existing plot.
2. Choose **Custom Plot**.
3. In the **Plotted Data Items** table, click **New Quantity**.
4. On the **Registers** tab, browse to **Controller**, **Loop Time**, and choose the desired **Loop Time Used** registers.

When trending a plot, you can see how the loop time usage varies. When commands are issued, the loop time will typically increase significantly for that loop time. If the total loop time used occasionally exceeds the set loop time by a small amount, there will typically be no adverse effects. Small loop time overages will be reported in the Event Log as a warning, which can be filtered using the Event Log filter settings.

It is very important that a single control loop execution or the accumulated carryover from consecutive loops does not exceed the loop time by more than 100%. The RMC will report an error in the Event Log if the loop time overage or cumulative loop time overage exceeds 80%. For more details, see the Loop Time topic.

See Also

Loop Time | Controller Loop Time Status Registers

3.2.4. RUN/PROGRAM/Disabled Mode

RUN and PROGRAM mode specify whether the User Programs and the Program Triggers can run. RUN or PROGRAM mode do not necessarily affect the motion on an axis. Motion commands can be issued to the RMC whether or not it is in RUN or PROGRAM mode.

The RMC200 also includes a Disabled Mode, in which no motion commands are accepted, and all axes are disabled.

RUN mode - RMC75/150/200

- The Program Triggers are started (if the Enable the Program Triggers task checkbox is checked in the Programming Properties).
- User Programs can run.
- LEDs
    - The RMC75 Controller LED is steady green.

  - The RMC150 RUN Mode LED is steady green.
  - The RMC200 RUN LED is steady green.
- The RMCTools toolbar will display Online (RUN).

PROGRAM mode - RMC75/150/200

- The Program Triggers are stopped.
- User Programs cannot run.
- LEDs
  - The RMC75 Controller LED slowly flashes green.
  - The RMC150 RUN Mode LED is off.
  - The RMC200 RUN LED is off.
- The RMCTools toolbar will display Online (PROG).

Disabled mode - 200 only

- All axes are disabled, and no motion commands are accepted.
- The Program Triggers are stopped.
- User Programs cannot run.
- The CPU Run and En LEDs are off.
- The RMCTools toolbar will display Online (Disabled).

Entering and RUN, PROGRAM and Disabled Modes

When the RMC enters RUN Mode:

- The controller will be enabled if it wasn't previously. This is indicated by the Enabled bit in the Controller Status register.
- **RMC75/150:** All the axes become enabled if they were not previously enabled. This means that the Enabled status bit turns on for each axis, exactly as if the Enable Controller (7) command had been issued.
- **RMC200:** If the **Auto-Enable Axes** option is checked on the RUN/Disabled page of the Programming Properties, then all the axes become enabled, and all Enable Outputs are turned on.
- All the Program Triggers are now evaluated. Notice that all the conditions are assumed to be false when the RMC is in PROGRAM mode. Therefore, if any condition is true when the RMC enters RUN Mode, it will start the specified actions.

Entering RUN mode will not affect the motion on the axes. The axes will continue doing what they were doing.

When the RMC enters PROGRAM Mode:

- The Program Triggers are stopped.
- The Tasks are stopped. All User Programs stop.
- **RMC200:** The controller will be enabled if it wasn't previously. This is indicated by the Enabled bit in the Controller Status register. If the **Auto-Enable Axes** option is checked on the RUN/Disabled page of the Programming Properties, then all the axes become enabled, and all Enable Outputs are turned on.
- If any discrete outputs have been configured to turn on or off when entering PROGRAM mode, they will do so.

Entering PROGRAM mode will not affect the motion on the axes. The axes will continue doing what they were doing.

When the RMC enters Disabled Mode:

- All axes will halt with a Direct Output Halt.
- The Controller Enabled Bit is cleared.

- The Program Triggers are stopped.
- The Tasks are stopped. All User Programs stop.
- If any discrete outputs have been configured to turn on or off when entering PROGRAM mode, they will do so.

Entering Disabled mode will stop motion by Direct Output Halt. Axes will not move while in Disabled mode.

Fault Controller

RMC75/150: If the Fault Controller (8) command is sent, the RMC will exit RUN mode and enter PROGRAM mode. All the axes will halt.

RMC200: If the Fault Controller (8) command is sent, the RMC will enter Disabled mode. All the axes will halt.

Methods of Entering RUN, PROGRAM, or Disabled Mode

From RMCTools

- On the toolbar, click the **Controller** button  ,  , or  and choose the desired mode.

From a Host Controller (PLC, HMI, etc.)

- RMC75/150: Send the RUN Mode (98) or PROGRAM Mode (99) commands to the RMC.
- RMC200: Send the RUN Mode (98), PROGRAM Mode (99), or Fault Controller (8) commands to the RMC.

Using a Discrete Input to enter RUN, PROGRAM, or Disabled mode

- In the Project Pane, in the project tree, right-click the **Programming** node, click **Properties**, and click the **RUN/PROGRAM** or **RUN/Disabled** page.
- Select **Define a RUN/PROGRAM discrete input** or **Define a RUN/Disabled discrete input**
- Choose the desired input and click **OK**.

  RMC75/150: When the input transitions from low to high, the RMC will enter RUN mode. When the input transitions from high to low, the RMC will enter PROGRAM mode. The state of the input does not necessarily reflect the RUN/PROGRAM state, because the RUN/PROGRAM mode can also be changed by other methods.

  RMC200: When the input transitions from low to high, the RMC will enter RUN mode. When the input is low, the RMC will be in the Disabled state. If the input is high, the RMC may not necessarily be in the RUN state, because RUN mode can be exited by other methods.

Starting the RMC in RUN, PROGRAM, or Fault mode

By default, the RMC75 and RMC150 start up in PROGRAM mode, and the RMC200 starts up in Disabled mode. You can set the startup mode:

- In the Project Pane, in the project tree, right-click the **Programming** node and click **Properties**.
- On the **RUN/PROGRAM** or **RUN/Disabled** page, choose the **Startup Mode** you want, then click **OK**.
- Right-click the **Programming** node and click **Download Programs**.
- Update Flash so that the settings will be saved when the RMC is powered off.

Tip:
You can start a User Program immediately when the RMC enters RUN Mode by using the FirstScan bit in the Program Triggers. If the Startup Mode is set to RUN Mode, the RMC can run a user program immediately upon powering up.

See Also

PROGRAM Mode (99) | List of Commands | Commands Overview

3.2.5. Registration

Registration is the process of recording the precise position of an axis when an external event occurs. Registration is commonly used for measurements. For example, consider a photo-eye set up on a conveyor belt, where the position of the belt is measured with a quadrature encoder. By registering the precise position at which the photo-eye is broken and the precise position at which it is reconnected, the length of a widget on the belt can be accurately determined.

The RMC provides registration on quadrature feedback axes only. This is because quadrature feedback is the only type of feedback that can record a position at any given moment, independent of the loop time of the RMC. All other feedback types can only report the position once each loop time (500µs to 4000µsec). Using the programming capabilities of the RMC, such as discrete inputs and the Program Triggers, it is possible to achieve functionality similar to registration with other feedback types, but the accuracy will not be the same as with quadrature feedback.

Registration Inputs

Each quadrature feedback axis can keep track of two independent registration events, called **Registration 0** and **Registration 1**. These registration events can come from a single registration input, or from separate inputs. The table below lists the inputs on which registration can be performed.

<table>
<tr><th>Module</th><th>Registration Sources</th></tr>
<tr><td>RMC75 QAx or
RMC150 Quad</td><td>RegX/PosLim input
RegY/NegLim input</td></tr>
<tr><td>RMC150 UI/O</td><td>DI/O inputs R0 and R1</td></tr>
<tr><td>RMC75 Q1</td><td>Reg input</td></tr>
<tr><td>RMC200 Q4</td><td>Depending on the quadrature input:
<table>
<tr><th>Input</th><th>Registration Sources Inputs</th></tr>
<tr><td>Quadrature input 0 and/or 1</td><td>Reg0, Reg1, Hm0, Hm1</td></tr>
<tr><td>Quadrature input 2 and/or 3</td><td>Reg2, Reg3, Hm2, Hm3</td></tr>
</table></td></tr>
<tr><td>RMC200 U14</td><td>Reg/Z0, Reg/Z1, D0, D1</td></tr>
<tr><td>RMC200 D24</td><td>Depending on the quadrature option:
<table>
<tr><th>Option</th><th>Registration Sources Inputs</th></tr>
<tr><td>One quadrature input (A,B,Z)</td><td>D22,D23,D18,D19</td></tr>
<tr><td>One quadrature input (A,A,B,B) with wire break detection</td><td>D22,D23,D18,D19</td></tr>
<tr><td>Two quadrature inputs, using inputs D20&D21</td><td>D18, D19</td></tr>
<tr><td>Two quadrature inputs, using inputs D22&D23</td><td>D16,D17,D18,D19</td></tr>
</table></td></tr>
</table>

Arming the Registration

Issuing the Arm Registration (52) command tells the axis to wait for a registration event. Once this command is issued, the registration is said to be **armed**. When the registration is armed, the corresponding registration armed status bit will be set (Registration 0 Armed and Registration 1 Armed). Unless the registration is armed, the registration trigger is ignored.

The Arm Registration (52) command also tells the axis which Registration Number (0 or 1) to apply the registration event to, which input the registration will be triggered from, and which edge of the input (rising or falling) will trigger the registration.

Registration Event

If the registration is armed and a registration trigger occurs, the following is done:

- The precise position at the time of the registration is **latched**. The position will be reported in the Registration 0 Position or Registration 1 Position status register.
- The corresponding latched status bit will be set (Registration 0 Latched or Registration 1 Latched).
- The corresponding armed status bit will be cleared (Registration 0 Armed or Registration 1 Armed).

Once a registration is latched, you must issue the Arm Registration (52) command again to rearm it.

Disarm Registration

To manually disarm a registration, issue the Disarm Registration (53) command. Issuing this command will clear the corresponding Registration Armed status bit (Registration 0 Armed and Registration 1 Armed).

See Also

Homing | Arm Registration (52) Command | Disarm Registration (53) Command

3.2.6. Homing

Certain encoders, such as quadrature encoders, are incremental and do not provide absolute position. To use an incremental encoder in a position application, a known reference position must be established. This position is called the **Home** position. The process of determining the home position is called **Homing**. As the encoder is rotated, it increments or decrements the position from the initial home position, and thereby the position is obtained.

When an axis is homed, the Actual Position of the axis is changed. The Target and Command positions are also adjusted by the same amount. Therefore, the motion of the axis will not be affected. It is safe to home an axis when it is moving.

The RMC offers homing of quadrature, incremental SSI, and incremental Resolver axes. All are described in this topic.

How to Home an Axis

This section describes the basic steps for homing a quadrature axis using the Home input and/or the Z (Index) pulse as the home trigger. The RMC has many options for homing; for more details, see the rest of this topic.

Note: If you are using the Z pulse, it is important to specify the Z Alignment before homing. See the **Using the Z Trigger** section below.

1. **Arm the Home**

In order to perform homing, the home must first be *armed*. This means that the RMC will be looking for the specified trigger to occur, and when it occurs, will set the Actual Position to the specified position.

To arm the home, issue the Arm Home (50) command. This parameters for this command are:

1. **Home Position:**
   This is the position to which the Actual Position will be set when the specified trigger occurs.
2. **Trigger Type:**
   This is the trigger that the RMC will wait for. This is either the Home Input (select rising or falling), or the Z Input, or a combination thereof.
3. **Repeat mode:**
   This specifies whether the homing should occur every time the trigger occurs, or only once.
4. **Home Input:**
   Specifies which input will be used as the Home input.

When the axis is armed, the Home Armed status bit will be set. This bit can be viewed in the Axis Status Registers All tab, in the Home section.

2. **Move the Axis Until it Homes**

Move the axis so that it reaches the specified trigger, such as the Home input or Z pulse. To move the axis, use any motion command you wish. Typically, a Move Velocity works well. When the specified trigger occurs, the Actual Position will be set to the specified position. The Target and Command Positions will be adjusted by the same amount that the Actual Position was adjusted, and the motion will therefore not be affected.

When the home has occurred, the Home Latched status bit will be set and the Home Armed status bit will be set. If the **Repeat Mode** parameter was set to **Repeat**, both the Home Latched and the Home Armed status bits will remain set. These status bits can be used in a user program to check that the home has occurred. The Home Input status bit also shows the current state of the home input.

When the home has occurred, issue a command to stop the axis or move it to some desired location.

Quadrature Homing Details

This section gives more detail on homing a quadrature axis.

Trigger Types

The following trigger types are available on quadrature axes.

#	Trigger Type	Description	Valid for		
			RMC75	RMC150	RMC200
0	H Rising	Trigger a Home on the rising edge of the Home Input.	QAx, Q1	Quad, UI/O	Q4, D24, U14
1	H Falling	Trigger a Home on the falling edge of the Home Input.	QAx, Q1	Quad, UI/O	Q4, D24, U14
2	Z	Trigger a Home on the Index (Z) Input.	QAx	Quad	Q4, D24[1], U14
3	Z And H	Trigger a Home on the Index (Z) Input if the Home Input is high.	QAx	Quad	Q4, D24[1], U14
4	Z And Not H	Trigger a Home on the Index (Z) Input if the Home Input is low.	QAx	Quad	Q4, D24[1], U14

[1]For the ABZ option only.

Using the Home Input (Trigger Types 0 - 1)

The Home input is defined as follows:

Feedback Type	Home Input
RMC75 QAx or RMC150 Quad	**Home** input on the axis connector.
RMC75 Q1 Module	**Reg** input on the axis connector.
Universal I/O Module quadrature input	Corresponding **Reg *n*** input on the module.
RMC200 Q4	The associated **Home** input or **Reg** input. Selection is made in the **Home Input** parameter of the Arm Home (50) command.
RMC200 D24 quadrature input	Any of input 16-23, depending on the quadrature option, and defined by the **Home Input** parameter of the Arm Home (50) command.
RMC200 U14 quadrature input	Selection is made in the **Home Input** parameter of the Arm Home (50) command: For channel 0: **Reg/Z0** or **D0**. For channel 1: **Reg/Z1** or **D1**.

When the home is armed and the Home Input trigger occurs, the exact counts at the time the trigger occurred are **latched**. This position is used as the physical home position. The Actual Position at this location is set to the requested **Home Position** as specified by the Arm Home (50) command. The Command Position and Target Position will be offset by the difference between the new and old Actual Positions.

On quadrature axes, the counts are latched within a very short time and are therefore typically very accurate (depending on the encoder resolution, of course). For details on the latching time, see the propagation delay specification for the I/O module. To obtain an accurate trigger, the direction and speed should be the same each time the axis is homed. This is because proximity switches have delays and hysteresis.

Using the Z input (Index Pulse) (Trigger Types 2 - 4)

The Z input is available on the RMC75 QA1 and QA2, RMC150 Quad, RMC200 Q4, RMC200 U14 and the RMC200 D24 module in certain modes.

Homing with the Index (Z) pulse is the most accurate method of homing. When the home is armed and the Index (Z) pulse of the encoder is encountered (with or without the Home Input, depending on the trigger type), the Home is latched. When this occurs, the Actual Position at the location of the Z pulse is set to the **Requested Position** as specified by the Arm Home (50) command. The Command Position and Target Position will be offset by the difference between the new and old Actual Positions.

In order to achieve the most accurate homing with the Index (Z) pulse, you must issue the Learn Z Alignment (54) command when setting up your axis. This command sets the Index (Z) Home Location parameter. Once this parameter is set correctly, you can home with the Index (Z) pulse in any direction, at any speed, and the home will always occur at the exact same position on the encoder. See the Learn Z Alignment (54) topic for details.

Linear Positioner Homing Tips

When using a quadrature encoder on a linear positioner, the axis must be homed to retain absolute positions. This is easiest if the proximity switch controlling the Home input is placed at one end of the travel (for example, in the retracted position); this will be the home position. This is a typical procedure for homing a linear positioner:

1. Arm the homing by issuing an Arm Home (50) command with a **Trigger Type** of **H Falling**.
2. **If the axis is at an unknown position**:

   Retract until the Home Input status bit is ON. This can be done with a Move Velocity command. Then, extend until the Home Input status bit is OFF. When the Home Input status bit turns off, the axis will home and the Home Latched status bit will be set.

   If the axis is already retracted:

If the axis is already retracted, the Home Input status will already be ON and the axis needs only extend as described above. When the axis is shut down, it should be retracted until the Home Input status bit is ON. This way the axis need only extend a small amount to be homed on startup.

Disarm Home

To disarm the home manually, use the Disarm Home (51) command.

Explicit Homing

The following command set adjusts the axis position without needing a Home or Z input:

- Offset Position (47)
  This command adds the requested **Position Change** to the Actual, Target, and Command Positions.
- Set Target Position (48)
  This command sets the Target Position to the requested **Position**, and adjusts the Command Position, and Actual Position by the same amount as the Target Position changed.
- Set Actual Position (49)
  This command sets the Actual Position to the requested **Position**, and adjusts the Command Position and Target Position by the same amount as the Actual Position changed.

Why Bother?

Explicit homing is useful when the operator physically measures something on the machine and needs to make an adjustment to the positions.

Explicit homing can de done on any axis type, not just those with incremental feedback.

Homing SSI and Resolver Axes

SSI and Resolver axes give absolute positions, and therefore, homing them is not necessary. In certain applications, these axes types can be set up as incremental, and can be homed. Homing SSI axes is not supported on the RMC200.

SSI and Resolver Homing Trigger Types

The following trigger types are available on SSI and Resolver axes. The axis *must* be set to incremental.

#	Trigger Type	Description	Valid for		
			RMC75	RMC150	RMC200
0	H Rising	Trigger a Home on the rising edge of the Home Input.	SSI*		none
1	H Falling	Trigger a Home on the falling edge of the Home Input.	SSI*		
2	Z	Trigger a Home on the Index (Z) Input.	SSI*	SSI*, Resolver*	
3	Z And H	Trigger a Home on the Index (Z) Input if the Home Input is high.	SSI*		
4	Z And Not H	Trigger a Home on the Index (Z) Input if the Home Input is low.	SSI*		
5	Absolute Adjust (H Rising)	On the rising edge of the Home Input, the zero Raw Counts point of the SSI encoder is set to the requested Home Position.	SSI*		
6	Absolute Adjust (H Falling)	On the falling edge of the Home Input, the zero Raw Counts point of the SSI encoder is set to the requested Home Position.	SSI*		

7	Absolute Adjust (Immed)	The zero count point of the SSI encoder is set to the requested Home Position immediately when this command is issued.	SSI*	SSI*, Resolver*	

Using the Home Input for SSI and Resolver Axes (Trigger Types 0 - 1)

The Home input for SSI and Resolver axes is defined as follows:

Feedback Type	Home Input
Incremental SSI	Defined by the SSI Home Source parameter (RMC75 only).
Incremental Resolver	No Home input is available on Resolver inputs.

When the home is armed and the Home Input trigger occurs, the exact counts at the time the trigger occurred are **latched**. This position is used as the physical home position. For SSI and Resolver axes, the counts are only read once per control loop, so the time to latch can be up to one control loop (500µs to 4000µs).

Using the Z input for SSI and Resolver Axes (Trigger Types 2 - 4)

The Z pulse for SSI and Resolver axes is defined as follows:

Feedback Type	Home Input
Incremental SSI	The zero Raw Counts position of the encoder.
Incremental Resolver	The zero Raw Counts position of the resolver.

For incremental SSI and resolver inputs, the Index pulse is given by the point when the Raw Counts cross zero (0). This will trigger a home in the same way that the index (Z) input will. However, there is no Learn Z Alignment parameter and no need to issue the Learn Z command.

Notice that for high-turn-count multi-turn SSI encoders, the Raw Counts may cross zero very infrequently. Therefore, homing with trigger types 2-4 on an SSI encoder is typically only useful for single-turn encoders.

Using the Absolute Home Latch (Trigger Types 5 - 7)

When the home is armed and the specified home trigger occurs, the Actual Position for the SSI axis is adjusted such that the Actual Position at zero Raw Counts of the SSI or Resolver will be set to the requested **Home Position** as specified by the Arm Home (50) command.

Notice that trigger type 7 will perform the homing immediately when the command is issued. Trigger type 7 is supported by the SSI and Resolver, but trigger types 5 and 6 are supported only by the SSI.

See Also

Registration | Arm Home (50) Command | Disarm Home (51) Command

3.2.7. System Time

The RMC motion controllers support various levels of keeping track of time. The RMC75, RMC150, and RMC200 include system time registers that keep track of time since the RMC started up. The RMC200 also contains a real-time clock and supports real-time registers. These registers are all Read Only.

The system time registers can be used for functions such as delay timers, calculating time between events, and comparing the time occurrence of motion controller events with the times of other devices.

Register Name	Data Type	Time Base	Rollover	Tag Name	RMC75	RMC150	RMC200
System Time - Time base starts when RMC powers up							
Time Gear Master	REAL	1.0 s	1 s	_Time			
Time, 16th Milliseconds	DINT	0.000 062 5 s	1.5 days (2^{31} ms/16)	_Controller.SysTime_16thmsec			
Time, Seconds	DINT	1.0 s	68 years (2^{31} s)	_Controller.SysTime_sec			
Time, Milliseconds	DINT	0.001 s	24.8 days (2^{31} ms)	_Controller.SysTime_msec Or: _SysMS			
Time, Microseconds	DINT	0.000 001 s	35.7 minutes (2^{31} µs)	_Controller.SysTime_usec			
Time, Nanoseconds	DINT	0.000 000 001 s	1 second (1 billion ns)	_Controller.SysTime_nsec			
Time, Loop Ticks	DINT	Same as the Loop Time (0.000 125 s to 0.008 s)	3.1 - 198 days (2^{31} loop times)	_Controller.SysTime_loops Or: _SysTicks			
Real Time - Time base starts Jan. 1, 1970							
Real Time UTC, Seconds	DINT	1.0 s	February 7, 2106 (2^{32} s)	_Controller.RealTimeUTC_sec			
Real Time Local, Seconds	DINT	1.0 s	February 7, 2106 (2^{32} s)	_Controller.RealTimeLocal_sec			
Real Time, Nanoseconds	DINT	0.000 000 001 s	1 second (1 billion ns)	_Controller.RealTime_nsec			

Using the Time registers

Time Gear Master

This register is intended only for use as a master register for gearing to time, such as running a curve based on time. It smoothly increments and wraps every second. It is the only time register that has a type of REAL.

Time, Seconds
Time, Milliseconds
Time, Microseconds

These DINT registers are used for general computing of time differences in units of seconds, milliseconds, or microseconds. The user can do a simple 32-bit subtraction to compare the elapsed time.

For example, to determine when 1 second has elapsed, the current time is first stored in a variable using an expression such as:

time0 := _Controller.SysTime_msec

then the following condition determines when 1 seconds has elapsed since the time was stored:

_Controller.SysTime_msec – time0 > 1000

Since each of these registers is a 32-bit DINT, each value will wrap from +2,147,483,647 to -2,147,483,648. However, because subtracting DINT values uses 32-bit modulo math, the subtraction will yield the intended result as long as no more than 2^{31}-1 (2,147,483,647) time units have elapsed between the two times.

The user must choose which time base provides adequate granularity and rollover times. For many applications in user programs, the **Time, Milliseconds** registers works well, as the default loop time is 1 ms, and times are typically in the range covered by this register (up to 2^{31} ms).

For long times and fine granularity, use **Time, Seconds** together with **Time, Nanoseconds**, as described in **Calculating Time with Seconds and Nanoseconds** below.

Time, Loop Ticks

This DINT register has a time base that matches the current loop time. For example, if the loop time is 1 ms, its time base will be 1 ms. If the loop time is 500 µs, its time base will be 500 µs.

This register is not recommended for use because its time base can be difficult to convert to a normal time, and if a user program employing this register is ported to an RMC with a different loop time, the results may not be consistent. For calculating small time increments, use **Time, 16th Milliseconds** instead.

Time, 16th Milliseconds

This DINT register with a time base of 62.5 µs can be used in some applications where small time increments must be calculated, rather than the **Time, Loop Ticks** register. It is intended to be used for taking simple differences in timestamps, similar to the **Time, Seconds**, **Time, Milliseconds**, and **Time, Microseconds** registers.

Time, Nanoseconds

This DINT register is different from the ones above, since it wraps at 1 billion. Specifically, it cannot be used like this:

_Controller.SysTime_nsec – time0 > 1000 //Do not do this!

because rolling over at 1 billion breaks that difference.

This register is intended to be used in conjunction with **Time, Seconds** for computing accurate time differences, as described in **Calculating Time with Seconds and Nanoseconds** below.

Real Time UTC, Seconds (RMC200 Only)
Real Time Local, Seconds (RMC200 Only)
Real Time, Nanoseconds (RMC200 Only)

These DINT registers capture the real time, as determined by the real-time clock. When the RMC200 powers up, these registers take on the time stored from the onboard real-time clock. The time can also be adjusted via RMCTools. The time is in Unix time, which is seconds and nanoseconds since midnight UTC/GMT January 1, 1970.

The **Real Time UTC, Seconds** register gives the Coordinated Universal Time (UTC), which corresponds to Greenwich Mean Time (GMT). The **Real Time Local, Seconds** register gives the local real time, which is adjusted for the selected time zone and daylight savings time, if applicable. The **Real Time, Nanoseconds** register applies to both.

It is not safe to compute basic mathematical differences based on these real time values because the real time can be adjusted while the controller is running by RMCTools. See **Calculating Time with Seconds and Nanoseconds** below.

Calculating Time with Seconds and Nanoseconds

The **Time, Seconds** register (_Controller.SysTime_sec) is used together with the **Time, Nanoseconds** register (_Controller.SysTime_nsec) to calculate the time between two events, with very fine granularity over long time spans. To calculate the time between two events, a time

difference calculation must be performed between the two times, of which each time is represented by a seconds DINT and a nanoseconds DINT. This calculation is somewhat complicated, so Delta has provided user functions to capture times and compute differences, as described below.

User Functions

Delta has provided several user functions to perform the calculation between two times. These user functions are available for download from Delta's forum at https://forum.deltamotion.com/. Search for **Time Functions**, download the RMCTools User Function Library (.rmcflib) file, and import it into the User Functions in RMCTools.

Time Capture User Function

This function is for capturing the current system time and storing in two DINT variables.

- **TIME_CAPTURE**(out DINT time_sec, out DINT time_nsec) : DINT
  Captures the current system time, by capturing **_Controller.SysTime_sec** and **_Controller.SysTime_nsec** into the two output parameters **time_sec** and **time_nsec**. The return value is unused and will always be zero.

Elapsed Time User Functions

These functions determine the elapsed time between the current system time and a previously captured time.

- **TIME_ELAPSED_SEC**(DINT t0_sec, DINT t0_nsec) : REAL
  Returns time elapsed since the specified timestamp in seconds with fraction. This value is imprecise because of the limitations of 32-bit floating point math.
- **TIME_ELAPSED_MSEC**(DINT t0_sec, DINT t0_nsec) : REAL
  Returns time elapsed since the specified timestamp in milliseconds with fraction. Unlike TIME_ELAPSED_SEC, this value is precise up to certain limits. For a 1 msec loop time, this value is precise up to approximately 16,000,000 msec, and for a 0.125 msec loop time, is precise up to approximately 2,000,000 msec.

General Time Difference User Functions

The user functions listed above are relatively easy to use when determining how much time has elapsed since an event. Delta also provides more advanced user functions for determining the amount of time between two events, for which the times of each has been stored:

- **TIME_DIFF**(DINT t0_sec, DINT t0_nsec, DINT t1_sec, DINT t1_nsec, out DINT dur_sec, out DINT dur_nsec) : DINT
  Calculates the difference between two times by subtracting the time t0 (seconds and nanoseconds) from t1 (seconds and nanoseconds) and returning the resulting duration in seconds and nanoseconds as output parameters. The return value is unused and will always be zero.
- **TIME_DIFF_SEC**(DINT t0_sec, DINT t0_nsec, DINT t1_sec, DINT t1_nsec) : REAL
  Returns the difference between two times in seconds by subtracting the time t0 (seconds and nanoseconds) from t1 (seconds and nanoseconds) and returning the resulting duration in seconds with fraction as a REAL. This value is imprecise because of the limitations of 32-bit floating point math.
- **TIME_DIFF_MSEC**(DINT t0_sec, DINT t0_nsec, DINT t1_sec, DINT t1_nsec) : REAL
  Returns the difference between two times in milliseconds by subtracting the time t0 (seconds and nanoseconds) from t1 (seconds and nanoseconds) and returning the resulting duration in milliseconds with fraction as a REAL. Unlike TIME_DIFF_SEC, this value is precise up to certain limits. For a 1 msec loop time, this value is precise up to approximately 16,000,000 msec, and for a 0.125 msec loop time, is precise up to approximately 2,000,000 msec.

Converting to Seconds or Milliseconds

These user functions convert a time represented by two DINTs (seconds and nanoseconds) to a single REAL, in seconds or milliseconds. These functions are typically not necessary, since the TIME_ELAPSED and TIME_DIFF functions above can output the time to a single REAL.

- **DURATION_SEC**(DINT dur_sec, DINT dur_nsec) : REAL
  Converts from a duration in seconds and nanoseconds to seconds with fraction as a REAL.
- **DURATION_MSEC**(DINT dur_sec, DINT dur_nsec) : REAL
  Converts from a duration in seconds and nanoseconds to milliseconds with fraction as a REAL.

Example: Elapsed Time

To calculate the time that has elapsed since a certain event, first, capture the time of the event, and then use a TIME_ELAPSED function to determine the elapsed time.

The functions in this example can be obtained from Delta's forum at https://forum.deltamotion.com/. Search for **Time Functions**, download the RMCTools User Function Library (.rmcflib) file, and import it into the User Functions in RMCTools.

Example: Time Difference

To calculate the time difference between two times that each have been stored as seconds and nanoseconds, use a TIME_DIFF function.

The functions in this example can be obtained from Delta's forum at https://forum.deltamotion.com/. Search for **Time Functions**, download the RMCTools User Function Library (.rmcflib) file, and import it into the User Functions in RMCTools.

First, capturing the time that the first event occurred:

Next, capture the time that the second event occurred:

Finally, calculate the difference between the two times:

See Also

System Time Registers | RMC200 Real-Time Clock | Event Timers

3.2.8. Event Timers

Event Timers are used to capture the precise time at which an external event occurs, such as a high-speed discrete input turning on or off, or a quadrature count changing. The time of an event will be latched at a much finer resolution than the Loop Time time of the controller. The captured time is stored in two 32-bit registers using the standard format for the system time (seconds & nanoseconds since the start of the first motion control loop). The RMC system time is monotonically increasing and will never be adjusted when the real time is changed, so times captures can simply be subtracted from one another.

Event Timers can be used for timing-related applications such as:

- Calculating the time between pulses on a single input.

- Calculating the time between two different inputs turning on.
- Calculate a pulse width.

See the **Applications** section below for instructions on how to perform specific Event Timer applications.

Supported Modules

Module	# of Event Timers	Time Resolution
D24	4	Measurement resolution: 25 ns See the D24 topic for propogation delay specifications.

Capturing the Time of an Event

1. **Determine slot number**
   Determine the slot number of the module with the high-speed input.

Base	Slot Numbers
B5	2–4
B7	2–6
B11	2–10
B15	2–14

2.
3. **Choose an Event Timer and Event Source**
   The **Event Timer** is the internal software resource for the timer, not the actual hardware input. The Event Timers serve to keep track of the physical timing events. The number of simultaneous events that can be timed on a module is limited by the number of Event Timers available for the module.

Module	Event Timers
D24	0–3

4.
   The **Event Source** is the physical event that will be timed. This can be a quadrature count change, or a discrete input change. Certain Event Timers are supported for certain Event Sources.
5. **D24 Hardware Inputs:**

Event Timers	Event Source Names	Hardware Input
0 or 1	Quad Count: Any change in the quadrature count, useful for calculating velocity from last two counts.	Quadrature Channel 0
	In0 Rising Edge, In0 Falling Edge	Discrete Input D20
	In1 Rising Edge, In1 Falling Edge	Discrete Input D21
2 or 3	Quad Count: Any change in the quadrature count, useful for calculating velocity from last two counts.	Quadrature Channel 1
	In0 Rising Edge, In0 Falling Edge	Discrete Input D22
	In1 Rising Edge, In1 Falling Edge	Discrete Input D23

6.
7. **Choose Timer Mode**
   The following modes are available. See the **Applications** section below for more details.

- **One-Shot**
  The Event Timer will capture the time of the first occurrence of the event after the timer is armed. This is useful when multiple pulses may occur, but only the time of the first one is desired.
- **Continuous**
  The Event Timer will continue to capture the time of each occurrence of the event after the timer is armed. The time of the most recent event will be stored in the Event Timer registers. This is useful when multiple events may occur within a single loop time, and the time of the last event is desired. To stop capturing, use the Disarm Event Timer (106) command.
- **Alternating**
  The Event Timer will capture the time of every other occurrence of the event after the timer is armed. To use the Alternating mode, two Arm Event Timer commands must be used on two separate Event Timers, with both Event Timers using the same module slot, event source, and timer mode (Alternating), and the two Event Timers must be in the same pair. On the D24, 0 and 1 are a pair, and 2 and 3 are a pair.
  Once Alternating mode has been armed, the first event timer will latch on the first occurrence of the event, then the second event timer will latch on the next occurrence, then the first event timer will latch on the next occurrence, and the event timers will continue to alternate in this fashion.
  To stop capturing, use the Disarm Event Timer (106) command.
  Alternating mode is useful for measuring the time between the last two counts from the quadrature encoder, which can then be used to calculate speed.

8. **Send Command**
   Send the Arm Event Timer (105) command. The commanded axis is irrelevant. This will arm the specified Event Timer, and the Event Timer Armed status bit will be set. See the **Event Timer Status Bits** section below for more details.
9. **Wait for Event**
   Once the event has occurred, the Event Timer Latched bit will be set. See the **Event Timer Status Bits** section below for more details. For the Continuous and Alternating timer modes, use the Disarm Event Timer (106) command to disarm the timer when done.
10. **View the Latched Time**
    The latched time will be stored in the following registers, as described in the **Event Timer Registers** section below.
    - Event Timer *n* Seconds
    - Event Timer *n* Nanoseconds
11. **Calculate Time Differences**
    Typically, any timing application will involve calculating the time between events. See the **Event Timer Calculations** section below for details on how to calculate the time differences.

Event Timer Status Bits

Each Event Timer has an **Event Timer Armed** status bit and an **Event Timer Latched** status bit:

- **Event Timer Armed status bit**
  The Event Timer Armed status bit will be set when the Event Timer is armed. For the **One-Shot** timer mode, once the Event Timer latches, the Armed bit will be cleared. For the **Continuous** and **Alternating** timer modes, the Armed bit will remain set until the Disarm Event Timer (106) command is sent to disable the Event Timer.
- **Event Timer Latched status bit**
  If the Event Timer is armed, the Event Timer Latched status bit will be set once the event occurs. In One-Shot mode, the Latched bit will remain set until the Event Timer is re-armed or disabled. In Continuous or Alternating mode, the Latched bit will be set only for the loop when the event time is latched, and then cleared on the next loop.

Name	Tag Name	Data Type	Address
Event Timer Armed	none	Boolean	%MD*s*.*b*.0 *s* = 129 + slot number *b* = 101 + 4 x Event Timer Number
Event Timer Latched	none	Boolean	%MD*s*.*b*.1 *s* = 129 + slot number *b* = 101 + 4 x Event Timer Number

Event Timer Registers

The time of an event is recorded in the **Event Timer Seconds** and **Event Timer Nanoseconds** registers. The Event Timer Seconds register holds the number of whole seconds, and the Event Timer Nanoseconds register holds the remaining number of nanoseconds.

Each Event Timer has one dedicated Event Timer Seconds register and one dedicated Event Timer Nanoseconds register. The time uses the standard epoch for the RMC system time (seconds & nanoseconds since the start of the first motion loop). Notice that the **Event Timer Seconds** and **Nanoseconds** registers share the same time base as the **Time, Seconds** and **Time, Nanoseconds** registers described in the System Time topic.

Name	Tag Name	Data Type	Address
Event Timer Seconds	none	DINT	%MD*s*.*b* *s* = 129 + slot number *b* = 102 + 4 x Event Timer Number
Event Timer Nanoseconds	none	DINT	%MD*s*.*b* *s* = 129 + slot number *b* = 103 + 4 x Event Timer Number

Event Timer Calculations

Event times are stored as Seconds and Nanoseconds registers that together represent a time. These two registers share the same time base. For example, a time of 13.568 seconds would be represented by a value of 13 in the Seconds register, and 568,000,000 in the Nanoseconds register.

To calculate the time between two events:

1. Determine the register addresses of the Event Timer Seconds and Event Timer Nanoseconds registers for each of the two events. This is described in the **Event Timer Registers** section above.
2. Check the Event Timer Latched status bits to ensure the timer values have latched. This is described in the **Event Timer Status Bits** section above.
3. Calculate the time difference of the two events.
   Use one of the TIME_DIFF functions provided by Delta, as described in the **Time User Functions** section below. The function will take the Event Timer Seconds and Event Timer Nanoseconds registers for each of the two events.
4. If necessary, convert the time difference to a REAL data type in the desired units of seconds, milliseconds, or nanoseconds.
   The function used to compute the time difference may perform this conversion. Otherwise, use one of the provided functions to perform the conversion.

Time User Functions

Delta has provided several user functions as described in the System Time topic to calculate the difference between two times, and to convert to a REAL value in seconds or milliseconds. The functions can be obtained from Delta's forum at https://forum.deltamotion.com/. Search for

Time Functions, download the User Function Library (.rmcflib) file, and import it into the User Functions in RMCTools.

Applications

Following are instructions for measuring times for specific applications.

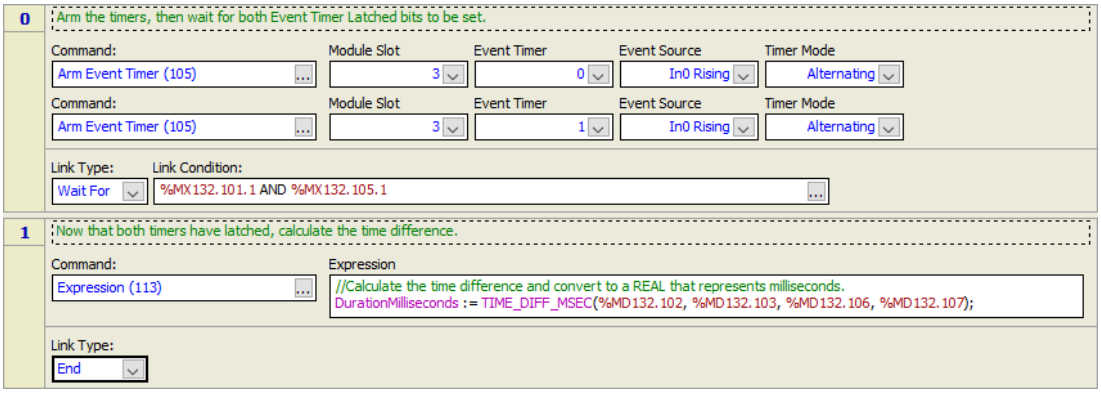
The functions in this example can be obtained from Delta's forum at https://forum.deltamotion.com/. Search for **Time Functions**, download the User Function Library (.rmcflib) file, and import it into the User Functions in RMCTools.

Time Between Pulses on a Single Input

Follow these steps to measure the time between two pulses on a single input. This will capture the two last events prior to measurement. For example, if the inputs alternated 10 times in a loop time after arming, step 1 of the user program would perform calculations on the last two input events.

1. Arm two different timers in alternating mode. The Event Source will be identical for both timers, for example In0 Rising Edge.
2. Wait for the timers to latch.
3. Calculate the time difference.

Example user program:

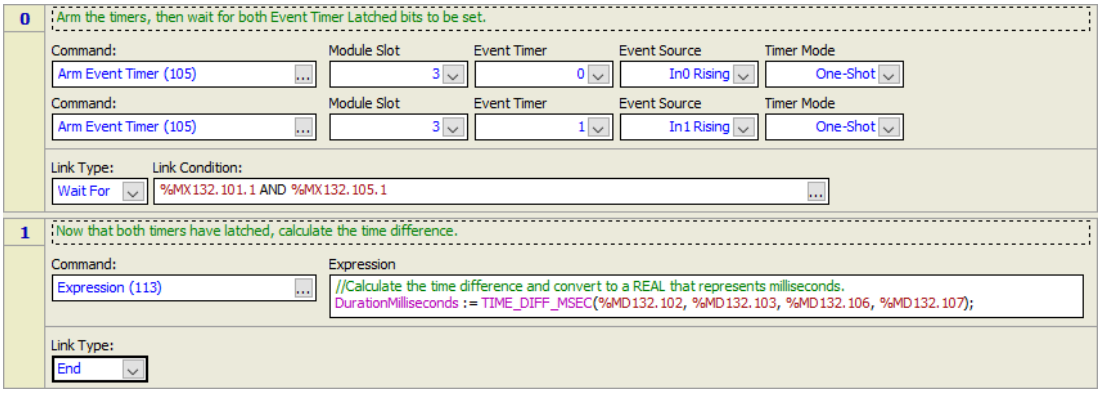


Time Between Pulses on Two Different Inputs

Follow these steps to measure the time between a pulse on one input, and a pulse on another input. This will capture the first pulse on each input after the Event Timer is armed.

1. Arm two different timers in one-shot mode. Choose the two inputs you wish to use for the Event Sources. Typically, both timers will be the rising edge, but your specific application may differ depending on the edge you wish to measure.
2. Wait for the timers to latch.
3. Calculate the time difference.

Example user program:

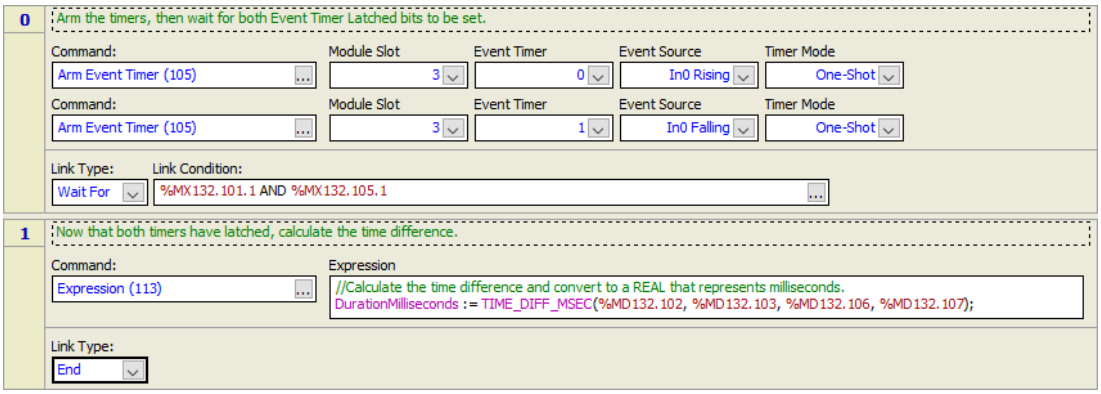


Measure Pulse Duration (Pulse Width)

Follow these steps to measure the width of a pulse. This will capture the first pulse on each input after the Event Timer is armed.

1. Arm two different timers in one-shot mode. The Event Sources will be the rising edge and falling edge of the same input.
2. Wait for the timers to latch.
3. Calculate the time difference.

Example user program:



See Also

Arm Event Timer (105) | Disarm Event Timer (106) | System Time Registers | System Time

3.2.9. Physical Limit Inputs

Each RMC axis allows for optional physical limit inputs to specify the boundaries in which the axis is allowed to operate. Each axis can have two physical limit inputs: Positive Limit Input and Negative Limit Input. Typically, these inputs are wired to limit switches.

The physical limit inputs differ from the Travel Limits, which limit the range of travel of an axis based on the feedback (position, pressure, etc.). The Travel Limits are required to be set on an axis; physical limit inputs are optional.

Setup

Use the Positive Limit Input and Negative Limit Input parameters to specify the inputs to be used for the physical limit inputs. The following options are possible:

Option	Description
none	This is the default setting.
Fault Input	**RMC75**: The Fault Input of the axis. **RMC150**: The Fault Input of the axis. Only available on the Quadrature Module. **RMC200:** n/a
Dedicated	Available on RMC75 QA and RMC150 Quadrature modules only. For the Positive Limit Input, this is the **PosLim** input. For the Negative Limit Input, this is the **NegLim** input.

general input	**RMC75**: any input from a D8 module, but only from the first 12 I/O points as listed in the Discrete I/O Monitor. **RMC150**: any general discrete input from any module. **RMC200**: any discrete input from any module.

Operation

The Positive Limit Input and Negative Limit Input *status* bits indicate the state of the Limit Inputs. When one of these inputs becomes active (as indicated by the status bits), the corresponding Positive Limit Input or Negative Limit Input *error* bit will be latched. The error bit will cause a Halt to occur if the Positive Limit Input and Negative Limit Input Auto Stops are configured to do so and the Direct Output Status bit is off.

Issuing Commands while a Limit Input is Active

While a Limit Input is active, closed loop motion commands that move the axis *away* from the valid range will set the corresponding Limit Input error bit, causing a halt if the Auto Stops are configured to do so. Motion commands that move *toward* the valid range will not set the a Limit Input error bit. This allows you to move an axis back within the limits without causing further errors.

Therefore, if the Negative Limit Input is active, a motion command that moves the axis in the *negative* direction will cause an error. A motion command that moves the axis in the *positive* direction will not cause an error. Likewise, if the Positive Limit Input is active, a motion command that moves the axis in the *positive* direction will cause an error. A motion command that moves the axis in the *negative* direction will not cause an error.

Moving Toward the Valid Travel Range

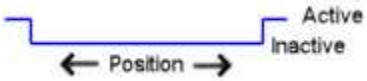
If a Limit Input is active, and the axis is in open loop, and is drifting slightly, issuing a closed loop motion command may again trigger an overtravel error. This is because the target position starts at the actual position, velocity and acceleration at the time the command is issued. Therefore, the target may continue moving away from the valid range while accelerating to turn around. This will trigger an overtravel error. To avoid this, first issue a Hold Current Position (5) command, then the motion command. Or, simply use an open loop move to move into the valid range.

Limit Input Polarity

The Limit Input Polarity parameter determines whether Positive Limit Input and Negative Limit Input are active high or active low. The Positive Limit Input status bit and Negative Limit Input status bit indicate whether or not the corresponding limit input is active. For details on voltage levels, see the specifications for the module the input is located on.

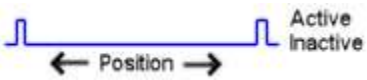
Physical Placement of Sensors for Limit Inputs

Typically, the sensors for Limit Inputs should be installed at both ends of travel on the axis. For safe operation, they should be designed such that they become active close to the end of travel, and remain active to the physical end of travel. This increases safety, because motion commands (other than Direct Output) will only be allowed in the direction toward the valid range of travel. A diagram is shown below:



Good Practice

Do not install limit sensors such that they become active close to the end of travel, but then become inactive at the end of travel, as shown below. In this case, commands in the wrong direction will be allowed.



Poor Practice

See Also

Positive Limit Input | Negative Limit Input | Travel Limits

3.2.10. Feedback Resolution

Feedback Resolution specifies the smallest increment that can be measured by the feedback device (such as a position transducer or encoder). With analog feedback, the resolution can also be a function of the input circuit (Analog-to-Digital converter) on the controller. Sometimes resolution is referred to as granularity.

Why is Feedback Resolution Important?

There are two main reasons why feedback resolution is important:

a. **Positioning Accuracy**
The controller cannot hold a position if it can't accurately determine how close it is to the desired position. It is generally necessary to have resolution that is several times better than the desired accuracy. Notice, however, that high resolution is a requirement for, but is not equivalent to high accuracy.

b. **Quantization Noise**
A less obvious, but equally important reason is quantization noise. Since velocity is the change in position per unit of time, the velocity resolution is dependent on the position resolution and the controller loop time. If the controller has a 1 millisecond loop time, the velocity resolution will be 1000 times worse than the position resolution. The acceleration measurement will be 1000 times worse than the velocity measurement. The differential gain and double differential gain use the actual velocity and actual acceleration respectively, to increase the system stability. Excessive quantizing noise severely limits the effectiveness of these gains.

Maximum Feedback Resolution for the RMC

The maximum resolution available on the RMC for various feedback types is listed below:

- **MDT**
For MDT Start/Stop or PWM feedback with a typical transducer having a gradient (or calibration constant) of 9 µs/in, the resolution of obtained by the RMC is:
    - RMC75 MA Module: 0.0005 inch with 1 recirculation
    - RMC150 MDT Module: 0.001 inch with 1 recirculation
    - RMC200 S8 and U14: 0.0005 inch with 1 recirculation
- **SSI**
The SSI transducer sends the position information digitally, so the only limit is in the transducer or encoder. One micron resolution is common for linear SSI transducers, and 8192 counts per turn is common for rotary encoders. Other resolutions are readily available.
The number of supported SSI bits is:
    - RMC75 MA module: 8 to 32 SSI bits
    - RMC150 Universal I/O module: 8 to 32 SSI bits
    - RMC150 SSI module: 8 to 31 SSI bits
    - RMC200 S8 and U14: 8 to 32 SSI bits

  Delta recommends that the SSI Counts value should not exceed 24 bits (16,777,216). See the **Exceeding 24 Bits** section below.
- **Load Cell**

  The analog-to-digital converter on the LC8 load cell input is 24 bits. The effective resolution is lower, at approximately 17 bits, depending on the LC8 module filter settings and external

noise. Given that the input range is ±34.5 mV, and an approximate resolution is 17 bits, the effective resolution is 540 nanovolts.

Note:
Typically, noise on the load cell wires will exceed these small resolution values, and will have a larger impact on system performance.

- **Analog**
  The RMC Analog-to-Digital converter support the following number of bits:
  - RMC75 AA, A2, and AP2: 16 bits
  - RMC150 H, G, and Universal I/O: 16 bits
  - RMC150 A: 12 bits
  - RMC200 A8 and U14: 18 bits

  The effective resolution of the Analog-to-Digital converted signal is increased by the following items:
  1. **Oversampling**
     The analog-to-digital converters are read multiple times per sample, increasing the effective resolution of the Analog-to-Digital converted signal:
     1. AA, A2, AP2, H, and G: eight times oversampling per sample, increasing the effective resolution of the Analog-to-Digital converted signal. For example, the eight times oversampling causes a 16-bit input's effective resolution to be 19 bits (one part in 524,288) over the full ±10V range.
     2. RMC150 Universal I/O Module: sampled at 60kHz, which is a minimum of 15 times oversampling.
     3. RMC200 A8 and U14 module: sampled internally at 200 kHz. The A8 is effectively 20 bits, and the U14 is effectively 21 bits.
  2. **Input Range Gain**
     Choosing the ±5V or 4-20mA ranges on the RMC150 H or G modules or the 4-20 mA range on the U14 module changes the gain of the analog input, increasing the resolution over the requested range.

  Due to the factors listed above, the effective resolution of the RMC analog inputs is as shown in the table below.

Module	±10 V	±5 V	4-20 mA
AA AA, A2, AP2	38.1 µV		0.15 µA
Analog (H)	38.1 µV	19.1 µV	0.076 µA
Analog (G)	38.1 µV		
Analog (A)	610 µV	305 µV	1.221 µA
Universal I/O	38.1 µV		0.15 µA
A8	20.1 µV		0.0805 µA
U14	9.77 µV		0.0316 µA

Note:
Typically, noise on the analog signals will exceed these small resolution values, and will have a larger impact on system performance. This is especially true on 18-bit inputs such as the A8 and U14 modules.

- **Quadrature**
  Quadrature encoder resolution is specified in pulses (also called lines) per revolution on rotary encoders or pulses per inch (or meter) on linear encoders. The RMC quadrature input will

decode the pulses to generate four counts per pulse. So the internal resolution on the RMC will be four times the resolution of the encoder given in pulses. For example, encoders with 2000 pulses will have 8000 counts.

The maximum quadrature encoder frequency is limited:

Module	Max quadrature frequency (counts/sec):
RMC75 QA, Q1	8,000,000
RMC150 Quad	4,000,000
RMC150 UI/O	8,000,000
RMC200 Q4	12,000,000, depending on input type
RMC200 S8	8,000,000
RMC200 D24	1,000,000, depending on input type
RMC200 U14	8,000,000, depending on input type

Delta recommends that the quadrature Counts value should not exceed 24 bits (16,777,216). See the **Exceeding 24 Bits** section below.

- **Resolver**
  The RMC150 Resolver module can be configured for 14 or 16 bits per revolution. One revolution of the resolver will always consist of 65,536 counts on the RMC, regardless of the resolver resolution parameter. With 16-bit resolution, the counts will increment by one; with 14-bit resolution the counts will increment by four.

Exceeding 24 Bits

The quadrature and SSI inputs can handle Counts values up to 32 bits. However, Delta recommends that you design your system and programming such that the quadrature and SSI Counts value do not exceed 24 bits (16,777,216). The RMC can still interface with SSI devices that have more than 24 bits, but you should make sure the counts will not exceed 16,777,216. These limitations do not apply to voltage, current, or MDT feedback types, since their values never exceed 24 bits.

Delta recommends that the Counts value not be allowed to exceed 24 bits (16,777,216) because this causes the Actual Position to lose resolution. This occurs because the Actual Position is stored as a 32-bit floating point number, which is limited to 24 bits of precision. The resolution of a floating-point number depends on how large the number is. For example, a floating point number can precisely represent any integer that fits in 24 bits (-16,777,216 to +16,777,216) or it can represent numbers at a 0.001 resolution in the range of -16,777.216 to +16,777.216.

To determine when an axis' Actual Position will lose resolution, look at the Counts register. As long as the Counts register stays within 24 bits (-16,777,216 to +16,777,216), then the Actual Position register will approximately match the resolution of the transducer. However, as the Counts move outside that range, the Counts register and the Actual Position register will lose resolution.

For example, if the Counts are 16,777,220 (slightly larger than 24 bits), and the transducer counts (the counts directly from the transducer are represented exactly in the Raw Counts register, which may be different from the Counts register, but it does indicate the change in counts exactly) change by one to 16,777,221, the Counts value will still read 16,777,220. It will not change until the counts have changed by 2, to 16,777,222. This will cause the Actual Position to become "jerky" and will affect the control. This problem is doubled for each power-of-two increase in the Counts value. Notice that the Counts and Actual Position registers are still accurately keeping track of the position change, but they are losing resolution.

Notice that in order to preserve accuracy on incremental feedback types, the RMC internally maintains a 32-bit integer accumulator. This ensures that the position does not drift due to loss of resolution.

Rotary axes use the Count Unwind value, which will keep the Counts within a defined range. Therefore, as long as the Count Unwind is kept below 16,777,216, this additional loss in resolution will not occur.

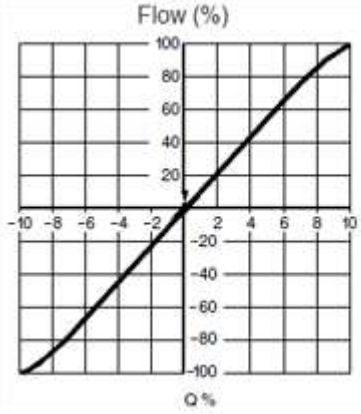
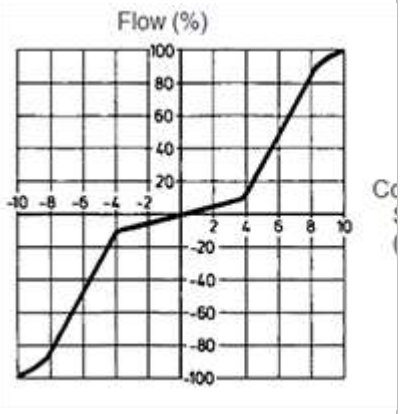
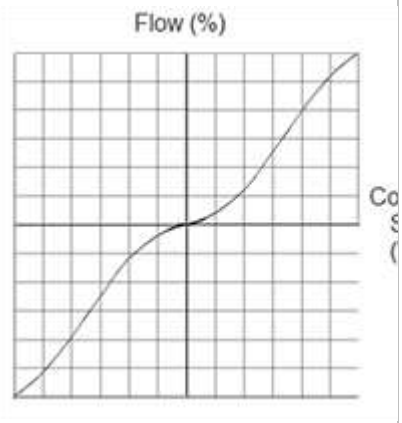
For linear axes, you can keep the Counts register from exceeding 24 bits by homing the axis, or using the Set Actual Position (49) or Offset Position (47) commands. With quadrature encoder inputs, the usable range with full resolution can be doubled by setting the zero value of the Actual Position to the middle of travel. For absolute linear SSI feedback, you can also use the Count Offset parameter to move the usable Counts range closer to zero.

3.2.11. Valve Linearization

Valve linearization refers to compensating for non-linear hydraulic valves.

Valve Types

Linear and **non-linear** refer to the flow versus command signal profile of a valve. Typically, Delta recommends using linear valves. Linear valves are nearly always of very high quality, provide excellent response and are very easy to set up. Very high-quality, non-linear valves may be useful in certain situations that require high flow together with very fine positioning or pressure accuracy.

Linear	Non-linear	
	Single-knee	**Curvilinear**
The flow is proportional to the command signal input and does not require linearization.	The flow has a single sharp 'kink' and can be linearized via the Single-Point method.	These flow profiles can be linearized via curves.
		
Notice that it is common for the flow profile to roll off as it approaches 100%. This is common, and the valve is still considered linear.		

Linearization Types

Valve linearization is applied only when the axis is in closed loop control. The RMC supports valve linearization via two methods:

- **Single-Point**
  Used only for valves with a single "knee" or "kink" in the flow versus command signal diagram
- **Curves**
  Used for valves with any flow diagram, especially curvilinear valves.

Single-Point Valve linearization

The RMC provides single-point valve linearization to compensate for valves with one sharp "knee" or "kink" in the flow versus command signal diagram, as shown above. The flow diagram is assumed to include the points (0,0), (-100,-100), and (100,100).

To Apply Single-Knee Valve Linearization:

1.

1. In the Axis Tools, in the Axis Parameters Pane, on the **All** tab, expand the **Output** section.
2. In the **Valve Linearization Type** cell, choose **Single-Point**
3. Set the axis parameters as follows:
   RMC75/150:

   Knee Command Voltage: The input voltage value of the knee. This voltage can be obtained from the valve data sheet. For example, in the Single-Knee Valve diagram above, on the horizontal axis, the voltage is 4 volts.

   Knee Flow Percentage: The flow percentage of the valve at the knee. This value can be obtained from the valve data sheet. For example, in the Single-Knee Valve diagram above, on the vertical axis, the flow percentage is 10%.

   RMC200:

   Knee Command Input: The input value of the knee. This percentage value can be obtained from the valve data sheet. For example, in the **Single-Knee Valve** diagram above, on the horizontal axis, the voltage of 4 volts would equal an input value of 40%.

   Knee Flow Output: The flow percentage of the valve at the knee. This value can be obtained from the valve data sheet. For example, in the **Single-Knee Valve** diagram above, on the vertical axis, the flow percentage is 10%.
5. Download the changes and update Flash.

If the valve has an overlapped spool, the Output Deadband and Deadband Tolerance axis parameters can be used as normal.

Curve Valve linearization

Valve linearization via curves may be used to compensate for valves with any flow versus command signal diagram.

To Apply Valve Linearization via Curves:

1. Using the Curve Tool, create a curve for valve linearization. See **Creating a Curve for Valve Linearization** below.
   You can also import existing valve linearization curves into the Curve Tool.
2. In the Axis Tools, in the Axis Parameters Pane, on the **All** tab, expand the **Output** section.
3. In the **Valve Linearization Type** cell, choose **Curve**.
4. In the **Valve Linearization Curve ID** cell, enter the number of the desired curve to use for linearization.
5. Download the changes and update Flash.

Creating a Curve for Valve Linearization

To create a curve for valve linearization, make a curve that matches the flow profile of the valve, with the x-axis being the input signal in percent, and the y-axis being the flow output in percent. For overlapped-spool valves, see **Curve Valve Linearization and Deadband** below.

Most valve flow profiles are given as positive flows for both positive and negative x-values. The curve must have negative flows for negative x-values.

The curve must meet the following requirements:

- Curve Properties:
  - **Curve Type** must be **Valve Linearization**.

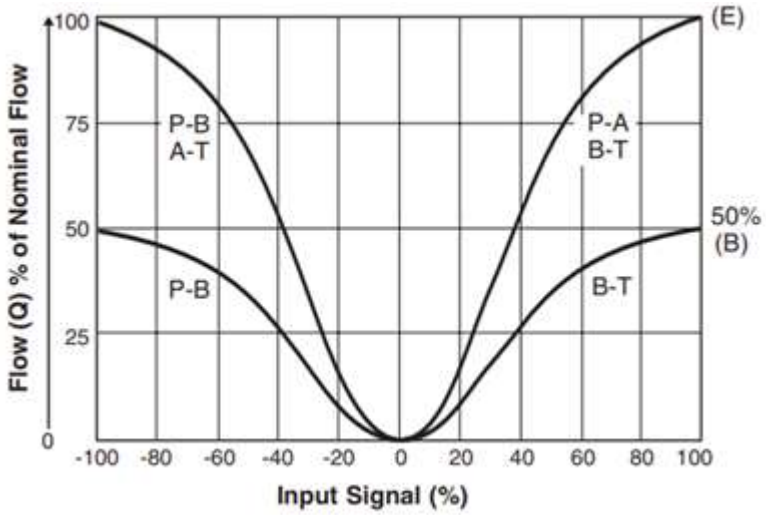
- **Endpoint Behavior** must be **Natural Velocity**.
- **Interpolation** must be **Linear** or **Cubic**.

- The first point must be (-100, -100).
- The last point must be (100, 100).
- The curve must contain the point (0,0).
- The x-values must be within the range -100 and 100.
- The y-values must be within the range -100 and 100, including the interpolated portions of the curve between points.
- The curve must be positive monotonic. This means the curve is always increasing or flat. Any flat section, if present, should only be around the point (0,0). See Curve Valve Linearization and Deadband below.

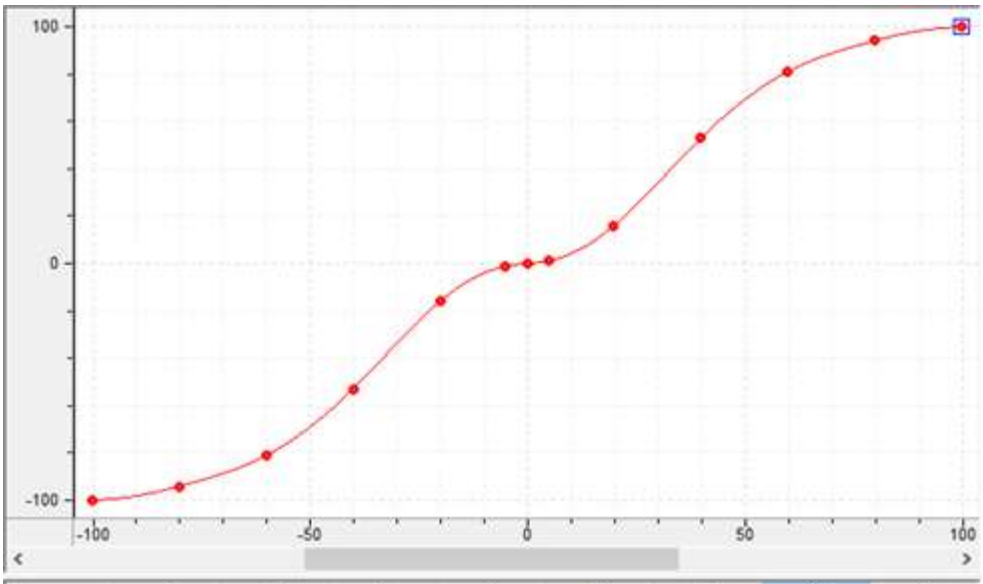
Tip: If the y-values of the flow profile are in gpm or lpm, you will need to convert them to percent when creating the curve.

Example:

Consider profile E in this valve flow profile diagram:

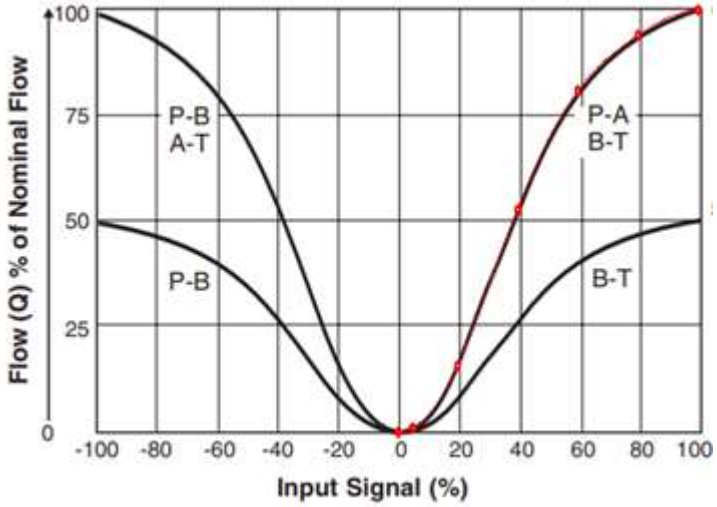


The following curve was created for this flow profile. Points were chosen at 20, 40, 60, and 80, plus an additional point close to zero (5.1, 1.1) so that the curve follows the profile accurately in that region. The negative values mirror the positive values.



	6	7	8	9	10	11	12	✳
X Value	0.0	5.1	20.0	40.0	60.0	80.0	100.0	
Y Value	0.0	1.1	16.0	53.0	81.0	94.0	100.0	
Velocity	0.136	0.375	1.551	1.809	0.965	0.482	0.125	0.0
Type	Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standard

One way to verify that the curve matches is to take a screen shot and overlay the curve on the flow profile using an image editing program:



Curve Valve Linearization and Deadband

Deadband refers to the range for an overlapped spool where the flow is zero until the command signal reaches a certain value.

If the valve has deadband, follow these steps for valve linearization via curves:

1. Create a curve that matches the valve flow profile and select that curve for the axis' valve linearization. See **Creating a Valve Linearization Curve with Deadband** below.
2. If the axis hunts when it is in position, you can mitigate it as follows:
    1. Set the Output Deadband axis parameter to the deadband of the valve.

2. Set the Deadband Tolerance axis parameter to a small value. This will ratio the Output Deadband when the Actual Position is close to the Command Position to prevent hunting. The axis typically holds position within a tolerance of about half of the Deadband Tolerance.

Creating a Valve Linearization Curve with Deadband

A linearization curve for a valve with deadband must follow the requirements listed in **Creating a Curve for Valve Linearization** above. It can be especially challenging to create the flat spot around zero, since the curve tends to become non-monotonic.

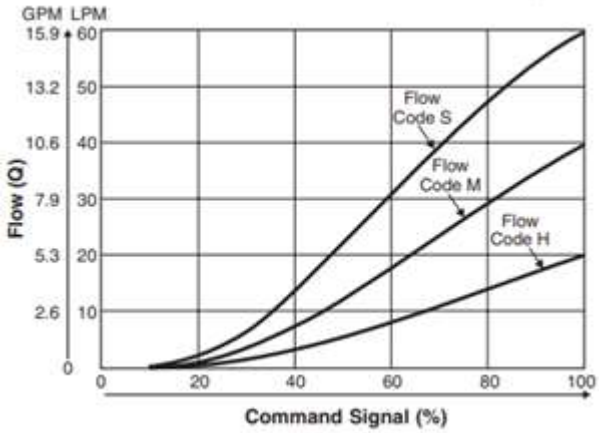
To make a curve flat around zero:

1. For the two points next to zero on the x-axis (the points to the left and right of zero), set the y-values to zero.
2. If using a cubic interpolation:
   For the point to the *left of zero*, and for the point *at zero*, in the spreadsheet editor, set the point type to **Const Vel**.
   Or, in **Curve Properties**, you can set the **Auto Constant Velocity Behavior** to **Enabled**.

Tip: The actual deadband of the valve may vary from the profile given in the datasheet. You may need to manually determine the deadband of the valve and adjust the curve accordingly.

Example

Consider profile S in this valve flow profile diagram:

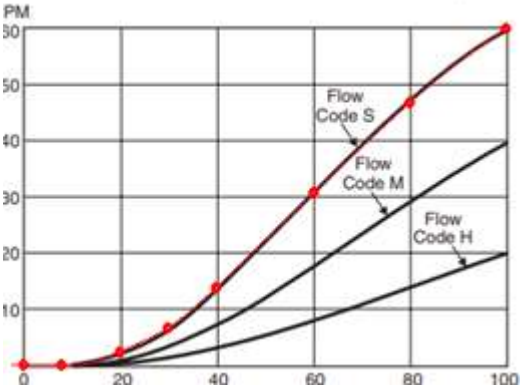


The curve below was created for this flow profile. Points were chosen at 8, 20, 30, 40, 60, and 80 so that the curve follows the profile accurately. The y-points were converted to percent. The negative values mirror the positive values.

Notice that the y-values for points 5, 6 and 7 are zero, and the Type for points 5 and 6 is Constant Velocity. The Constant Velocity type is necessary to maintain the monotonicity of the flat section of a cubic-interpolated curve.

	0	1	2	3	4	5	6	7	8	9	10	11	12
X Value	-100.0	-80.0	-60.0	-40.0	-20.0	-8.0	0.0	8.0	20.0	40.0	60.0	80.0	100.0
Y Value	-100.0	-78.0	-51.0	-23.0	-4.0	0.0	0.0	0.0	4.0	23.0	51.0	78.0	100.0
Velocity	0.975	1.257	1.442	1.225	0.708	0.000	0.000	0.000	0.708	1.225	1.442	1.257	0.975
Type	Standard	Standard	Standard	Standard	Standard	Const Vel	Const Vel	Standard	Standard	Standard	Standard	Standard	Standard

Overlaying the curve on the flow profile using an image editing program shows that the curve closely follows the profile:



See Also

Knee Flow Percentage | Knee Command Voltage | Valve Linearization Type | Valve Linearization Curve ID

3.3. Axes

3.3.1. Axis Types

The axis types listed in this topic are available on the RMC.

What is an axis?

An axis is a combination of inputs and/or outputs in one single entity, and includes all the necessary motion features for those inputs and/or outputs, which can include feedback scale and offset, filtering, control algorithms, output scaling, etc.

The major types of axes available in the RMC are:

- **Control Axis:**
  Has one Control Output and control either zero, one or two feedback quantities, such as Position, Velocity, Acceleration, Pressure, or Force.
  A control axis with no feedback quantities is called an **Output Only** axis.
- **Reference Axis:**
  Has an input for transducer feedback and does not have a Control Output. A reference axis is not capable of controlling a system. Reference axes are commonly used in gearing, synchronization, or monitoring applications.
- **Virtual axis:**
  Has only a virtual target position (with velocity and acceleration) with no feedback or control loop. Typically used as a gear or cam master.
- **Cascading Outer Loop**:
  A control axis that provides a virtual Control Output, and does not use a physical Control Output. For more details, see the Cascade Control topic.

Counting Axes

Any axis in the RMC counts as one single axis, regardless of the number of inputs or outputs. For example:

- A position-force application may require one analog position input, two analog pressure inputs, and 1 output, and is considered one axis.
- A reference axis with only one input is considered one axis.
- A virtual axis has no feedback or output and is considered one axis.

Supported Number of Axes per Controller

	RMC75	RMC150	RMC200 CPU20L	RMC200 CPU40
Max Control Axes[1]	2	8	18	50
Max Total Axes, including Virtual, Reference, and Outer Loop	4	16	48	128

[1] For the RMC75 and RMC150, Control Axes includes Output Only axes. For the RMC200, Control Axes does not include Output Only axes.

Note: It is possible to add more analog inputs on the RMC75 than can be assigned to axes. However, it is still possible to view the voltage of the extra analog inputs using the Analog Input Registers. Inputs that are unassigned to axes can have no status bits, error bits, scaling, filtering, etc.

Axes and Loop Time

The number of axes for each RMC is limited depending on the specified Loop Time. See the Loop Time topic for details.

Axes Physical Feedback Types

The RMC supports the physical feedback types listed below. Notice that any type of feedback (e.g. torque, temperature, etc.) can be controlled if the transducer is compatible with the RMC.

Any of these feedback types can be used as part of a Control Axis, Reference Axis, or Cascading Outer Loop axis.

Feedback Type	Transducer Type	Supported Transducer Output Types
Position Notice that a position feedback control axis can also control velocity.	position	MDT
		SSI
		Quadrature
		Resolver
		Analog Voltage or Current
Velocity	tachometer	Analog Voltage or Current
Acceleration (single- or dual-input)	1 or 2 accelerometers	Analog Voltage or Current
Pressure	pressure sensor	Analog Voltage or Current
Force (single-input)	load cell (±10V, 0-10V, ±5V, 4-20mA)	Analog Voltage or Current
Force (single-input)	load cell (mV/V)	Millivolts/Volt
Force (dual-input, differential)	2 pressure sensors	Analog Voltage or Current

Custom Feedback

An axis' feedback can be defined as custom, which requires creating a user program to continuously calculate the feedback value. This allows for using the sum, difference, or average of other transducers as feedback. It also provides for switching feedback on-the-fly, redundant feedback, and feedback linearization. For details, see Custom Feedback.

Dual Loop Control

The RMC supports dual-loop control. This is typically used for controlling two quantities, such as position and force, with a single actuator. See the Dual-Loop Axes topic for details.

Feedback Type	Description
Position-Pressure	Used for controlling both position and pressure with one actuator. Typically used with hydraulic cylinders.
Position-Force	Used for controlling both position and force with one actuator. Typically used with hydraulic cylinders.
Position-Acceleration	Used for advanced control such as active damping, which provides precision control of difficult systems, such as pneumatic systems.
Velocity-Pressure	Used for controlling both velocity and pressure with one actuator.
Velocity-Force	Used for controlling both velocity and force with one actuator.
Velocity-Acceleration	Used for advanced control applications.

Defining Axes

The user has full control over how the inputs and outputs on the Axis Module and the inputs on the Expansion Modules are assigned to axes.
See the Defining Axes topic for details on assigning axes to the hardware.

Other Axis Types

In addition to the axis types listed above, further distinction can be made between axes:

- Rotary vs. Linear
- Absolute vs. Incremental

3.3.2. Defining Axes

When creating axes, the user must define the axis type and define which physical hardware is assigned to each axis. For details on the axis types supported by the RMC, see the Axis Types Overview topic.

Changing the Axis Definitions

When the RMC powers up for the first time, it will create default axes. You may need to change the axis definitions to fit your application. To change the axis definitions, use the Axis Definitions Dialog:

- In the Project pane, expand the **Axes** folder and double-click **Axis Definitions**.
- To remove an axis, click the **Remove**  button. To add a new axis, click **New**. To change an existing axis, click **Change**.
- To change the axis order, use the **Move Up** and **Move Down** buttons. The axes will adhere to this order throughout the entire RMC. The Axis Tools and Command Tool will display the axes in this order, and the register addresses and tag names will be defined by this order.

Note: If the controller contains a Universal I/O module, you must first set up it's channels prior to defining axes.

You will need to define the following items for each axis:

Item	Options
Axis Type	**Control:** Has a physical Control Output and zero to two inputs **Reference:** Has only an input **Virtual:** Has no feedback or output. Has only a target. **Cascading Outer Loop:** Used for Cascade Control
Control Loops (Control Axes and Cascading outer Loop Axes Only)	**None:** For axes with no feedback, only a Control Output **Single-Loop:** For controlling a single quantity, such as position, velocity, acceleration (single- or dual-input), pressure, or force (single- or dual-input). **Dual loop:** For controlling two quantities with a single Control Output, such as position-pressure, position-force, position-acceleration, velocity-pressure, velocity-force, velocity-acceleration.
Output	**Type:** Analog output.

(Control Axes Only)	**Using:** Choose which physical Control Output on the RMC to use.
Feedback Types (for axes with feedback)	**Type:** • Position • Velocity • Acceleration (single- or dual-input) • Pressure • Force (single- or dual-input) **Using:** Choose which physical inputs will be assigned to the axis, or select **Custom**.

Axis Labels

In addition to the axis number assigned by RMCTools, each axis can be assigned a name by the user. To change the name of an axis, in the Axis Definitions Dialog, select an axis and click **Rename**.

Notice that the connectors on the front of the RMC75 labeled "Axis 0" or "Axis 1" do not necessarily have to belong to the internal Axis 0 or Axis 1 in the RMC75. The connectors on the RMC150 also do not necessarily indicate which internal axis they belong to.

See Also

Axis Types: Overview

3.3.3. Control and Reference Axes

3.3.3.1. Axis Type: Control

A control axis has a Control Output and controls either zero, one or two quantities, such as Position, Pressure, or Force. The controlled quantity is provided by a feedback input. A control axis is capable of controlling a system because it has a Control Output.

There are three types of control axes:

1. One-Input Control Axis

A control axis that controls a single quantity, such as position or pressure, is called a *one-input control axis*.

One-Input Control Axis Types	Required Inputs	Required Control Outputs	Transducer Types
Position	1	1	MDT, SSI, Analog, Quadrature, Resolver
Velocity	1	1	Analog Voltage or Current
Pressure	1	1	Analog Voltage or Current
Force	1 or 2	1	Analog Voltage, Current, Load Cell

Note:
A force input requires 1 analog or mV/V input if a load cell is used, or 2 analog inputs for differential force on a hydraulic cylinder. In the differential case, it is still considered one input, since it is only one quantity.

2. Two-Input Control Axis

A control axis that controls two quantities, such as position *and* pressure, is called a *two-input control axis*. Notice that the secondary inputs must always be analog or mV/V inputs.

Two-Input Control Axis Types	Required Primary Inputs	Required Secondary Analog or mV/V Inputs	Required Control Outputs
Position-Pressure	1	1	1
Position-Force	1	1 or 2	1
Position-Acceleration	1	1 or 2	1
Velocity-Pressure	1	1	1
Velocity-Force	1	1 or 2	1
Velocity-Acceleration	1	1 or 2	1

Note:
A force input requires 1 analog or mV/V input if a load cell is used, or 2 analog inputs for differential force on a hydraulic cylinder. In the differential case, it is still considered one input since it is only one quantity, although it requires two physical analog inputs.

3. Output Only Axis (No Input)

An Output Only axis has a Control Output and does not have any inputs. Control Outputs are available only on an axis module.

Creating a Control Axis

For details on defining axes, see the following topics:

Defining Axes

Axis Definitions: Dialog

Axis Definitions: Edit

See Also

Axis Types: Overview | Axis Types: Reference

3.3.3.2. Axis Type: Reference

A **reference axis** is an axis that has only an input, such as position, pressure, or force. A reference axis does not have a Control Output and cannot control anything. A reference axis is commonly called a "half-axis".

Reference axes are typically used for monitoring some quantity, such as:

- Position
- Velocity
- Acceleration
- Pressure
- Force

Creating a Reference Axis

To create a Reference axis, you must add a new reference axis or change an existing axis. For details on defining axes, see the following topics:

Defining Axes

Axis Definitions Dialog

Axis Definitions: Edit

Using Reference Axes

Reference axes are typically used for monitoring some quantity. The following are examples of specific uses with the RMC:

- Gear some axis to the reference input.
- Use the value of the reference to affect the flow of the User Program.
- Use the value of the reference in a Expression command in the User Program to perform mathematical calculations.

See Also

Axis Type: Control | Axis Types: Overview

3.3.3.3. Axis Type: Cascading Outer Loop

A Cascading Outer Loop axis is used as the outer loop of cascaded axes. A cascading outer loop axis is identical to a control axis, but it provides a virtual Control Output, and does not use a physical Control Output. This is only for advanced applications. For details, see the Cascade Control and control axis topics.

For descriptions of all available RMC axis types, see the Axis Types Overview topic.

There are two types of cascading outer loop axes:

1. One-Input Control Axis

A control axis that controls a single quantity, such as position or pressure, is called a *one-input control axis*.

One-Input Control Axis Types	Required Inputs	Transducer Types
Position	1	MDT, SSI, Analog, Quadrature, Resolver
Velocity	1	Analog Voltage or Current
Pressure	1	Analog Voltage or Current
Force	1 or 2	Analog Voltage or Current, or mV/V

Note:
A force input requires 1 analog or mV/V input if a load cell is used, and 2 analog inputs for differential force on a hydraulic cylinder. In the differential case, it is still considered one input, since it is only one final quantity.

2. Two-Input Control Axis

A control axis that controls two quantities, such as position *and* pressure, is called a *two-input control axis*. Notice that the secondary inputs must always be analog or mV/V inputs.

Two-Input Control Axis Types	Required Primary Inputs	Required Secondary Analog or mV/V Inputs
Position-Pressure	1	1
Position-Force	1	1 or 2
Position-Acceleration	1	1 or 2
Velocity-Pressure	1	1
Velocity-Force	1	1 or 2
Velocity-Acceleration	1	1 or 2

Note:
A force input requires 1 analog or mV/V input if a load cell is used, and 2 analog inputs for differential force on a hydraulic cylinder. In the differential case, it is still considered one input since it is only one final quantity, although it requires two physical analog inputs.

Creating a Cascading outer Loop Axis

For details on defining axes, see the following topics:

Defining Axes

Axis Definitions: Dialog

Axis Definitions: Edit

See Also

Axis Types: Overview | Axis Type: Control | Cascade Control

3.3.4. Other Axis Types

3.3.4.1. Axis Type: Rotary and Linear

This topic describes rotary and linear axes. All axes are by default linear. To define an axis as rotary, use the Linear/Rotary axis parameter.

A **rotary** axis is typically used for rotary feedback devices such as encoders. The RMC supports rotary feedback for both control axes and reference axes. For rotary axes, the counts per revolution must be

a power of two, such as 1024, 8192, etc. This typically means the encoder counter per turn must be a power of two.

For details and examples on using rotary motion, see the Using Rotary Motion topic.

A **linear** axis is the standard axis type with a transducer that has definite endpoints, for example a magnetostrictive rod.

Rotary vs Linear

SSI, Quadrature, and Resolver position inputs can be configured to be rotary. All other input types are linear only. The following items point out the differences between these rotary and linear orientations for position inputs:

- The counts for a rotary input are kept within a defined range. When the input goes beyond one end of this range, the counts wrap to the other end of the range. Similarly, the target is also kept within this range. Any control being performed using this input is not interrupted; this is not seen as a position discontinuity.
- The position units for a linear input never wrap.
- Rotary inputs use the Position Unwind parameters to determine how many counts can be accumulated before wrapping. This is used in conjunction with the Position Offset parameter, to define the position unit range. Linear inputs do not use the Position Unwind parameter at all.
- Linear incremental inputs do not use the Position Offset parameter because the actual position is really an accumulator, so offsetting the position is done instead by resetting the accumulator through homing or commands. Linear absolute inputs use this parameter to offset the position when converting from counts to position units. Rotary inputs (incremental or absolute) use this parameter to adjust the modulo position range up or down. For example, the position units on a rotary input with modulo 2000 could be shifted to range from -1000 to +1000 or from 0 to 2000.
- Linear position inputs use the Positive and Negative Travel Limits to restrict the target position. Rotary position inputs do not use these parameters.

See Also

Axis Types: Overview

3.3.4.2. Axis Type: Incremental and Absolute

This topic describes incremental and absolute axes. The feedback type of most axes determines whether it is incremental or absolute. Only position axes with SSI or Resolver feedback can be either.

Absolute Axes

An absolute axis has a feedback type that provides absolute information. That is, it always knows its exact position. When the RMC powers up, the position is known. For example, the voltage or current from an analog input will always tell exactly where the axis is located.

Incremental Axes

An incremental axis has a feedback type that provides only information about how much it has incremented. When the RMC powers up, it has no information about where the axis is, but it can keep track of how much the position changes. Incremental axes typically need to be homed so the position is known.

Quadrature inputs are always incremental.

By Axes Type

The following feedback types are **always absolute**:

- Position with MDT or analog feedback
- Velocity
- Acceleration
- Pressure
- Force

The following feedback types can be **either** incremental or absolute:

- Position with SSI or Resolver feedback
  To define these axis types as incremental or absolute, use the Absolute/Incremental axis parameter.

The following feedback types are **always incremental**:

- Position with Quadrature feedback

See Also

Axis Types: Overview

3.3.4.3. Virtual Axes

Virtual axes have no feedback or output. The RMC can have any number of virtual axes as long as the total number of axes for that controller type is not exceeded, as listed in Axis Types Overview.

Virtual axes have a Target and Command Position, but no Actual Position. Motion commands issued to the axis will move the Target Position just like on a normal position control axis.

Using a Virtual Axis as a Gearing Master or Curve Master

A virtual axis is typically used as a master axis that other axes can gear to or use a curve master. It is sometimes desirable to gear to a virtual axis rather than executing the motion as a function of time. All the axes geared to the virtual axis can be sped up or slowed down by speeding up or slowing down the virtual axis. The virtual axis can even be moved backwards causing the geared axes to back up too. This cannot be done using time-based commands.

When using a virtual axis as a master as described above, it is often useful to set it up as a rotary axis because it will never need to be reset. When used as a master, the virtual axis is typically commanded to move with a Move Velocity (37) command. Moving it at 1 unit/sec as the standard velocity makes gearing ratio calculations very easy. The acceleration and deceleration provide a smooth start and stop for the geared axis.

Setting Up a Virtual Axis

Add the Virtual Axis

To add a virtual axis to the RMC, use the Axis Definitions dialog:

1. Open the Axis Definitions dialog.
2. Whether you choose to change an existing axis or add a new axis, set the axis type to **Virtual.**
3. Click **OK** on the Axis Definitions dialog. If you are online, these changes will be applied to the controller. Make sure to update Flash and save your project.

Set up the Virtual Axis Parameters

A virtual axis has very few parameters. You need to set the Linear/Rotary parameter.

If you choose Linear, you will need to set the Positive and Negative Travel Limits.

If you choose Rotary, you will need to set the Position Unwind and Position Offset.

Once the virtual axis has been set up, you can issue closed-loop motion commands to it as to any position axis.

See Also

Axis Types: Overview | Gearing Overview | Curves Overview

3.3.4.4. Input Type: Pressure

Pressure input force refers to measuring pressure using a single pressure transducer. A pressure input requires only one analog input on the RMC.

In applications using hydraulic cylinders, notice that it is not possible to calculate resultant force on the rod with only one pressure transducer because the pressure on the other side of the cylinder is unknown. Applications with one pressure transducer cannot use force control, but can use pressure control. For details on measuring force on a hydraulic cylinder, see the Input Type: Dual-Input Force topic.

See Also

Input Type: Dual-Input Force

3.3.4.5. Input Type: Single-Input Force

Single-input force refers to measuring force using a single transducer, typically a load cell. Single-input force requires either an analog input or a load cell input on the RMC.

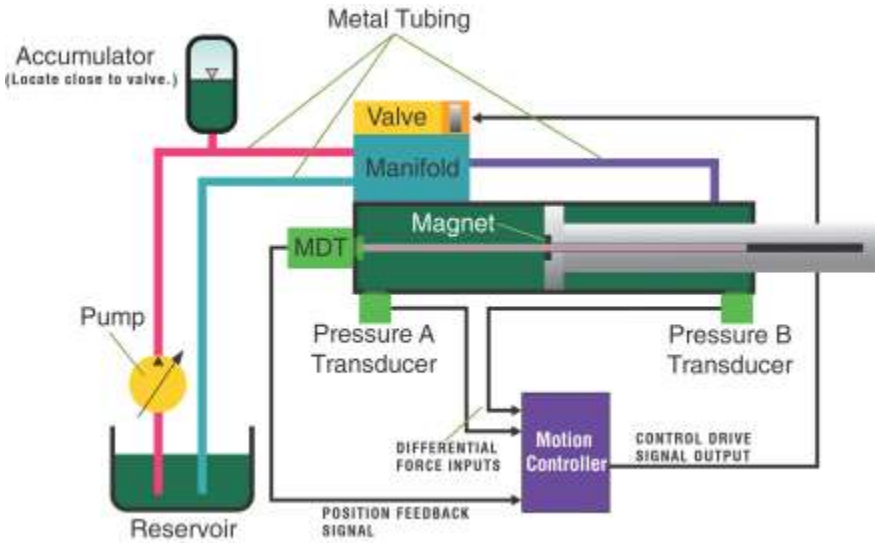
In applications using hydraulic cylinders, force can also be measured using dual-input force, requiring two pressure transducers mounted on either side of the cylinder. For details, see the Dual-Input Force topic.

See Also

Input Type: Dual-Input Force | Dual-Input Force

3.3.4.6. Input Type: Dual-Input (Differential) Force

Dual-input force, also known as differential force, is a method of determining the force applied by a hydraulic cylinder. As shown in the illustration below, this method requires two pressure transducers mounted in either end of a hydraulic cylinder; one on the A port (cap end) and another on the B port (rod end). Multiplying the measured pressure on each side of the piston by the respective area of the piston gives the force on each side of the piston. The difference of the absolute value of these two forces is the differential force applied to the cylinder rod.



Notice that it is *not* possible to calculate resultant force on the rod with only one pressure transducer because the pressure on the other side of the cylinder is unknown.

Differential force can only measure the force applied to the piston due to the hydraulic pressure. The resulting force applied to the external load is also affected by the friction in the cylinder. For this reason, a load cell may be more accurate than differential force.

Required Equipment

Differential force requires the following:

- **Two Identical Pressure Transducers**
  The measurement range and output ranges must match on order to use the scale and offset wizard provided for differential force. For example, if one has a range of 0-300 bar, and the other has a range of 0-100 bar, the Scale and offset Wizard cannot be used.
- **Two Analog Inputs On the RMC**
  Differential Force requires a pair of analog inputs on the RMC. The inputs must be next to each other on the same module.

Scaling a Differential Force Input

RMCTools offers an easy-to-use Scale and Offset wizard for differential force feedback. All that is needed is the transducer data and the cylinder dimensions.

For more details, see Scale/Offset Wizard: Calculate Diff Force.

Differential Actual Force Calculation

The method of calculating the differential actual force varies by controller. The RMC200 calculates the intermediate Channel A and B pressures, whereas the RMC75/150 controllers do not. The RMC200 allows for scaling pressure sensors that have different calibrations.

RMC75/150
The RMC75/150 calculate the forces for each channel, then takes the difference to arrive at the Actual Force. To apply a force offset, use either the Channel A Offset or Channel B Offset.

Channel A Force = Channel A Voltage (or Current) * Channel A Scale + Channel A Offset

Channel B Force = Channel B Voltage (or Current) * Channel B Scale + Channel B Offset

Actual Force = Channel A Force - Channel B Force

RMC200

The RMC200 calculates the intermediate pressures for each channel, then the forces for each channel, then takes the difference and applies the Dual Channel Force Offset to arrive at the Actual Force. To apply a force offset, use the Dual Channel Force Offset.

Channel A Pressure = Channel A Voltage (or Current) * Channel A Pressure Scale + Channel A Pressure Offset

Channel B Pressure = Channel B Voltage (or Current) * Channel B Pressure Scale + Channel B Pressure Offset

Channel A Force = Channel A Pressure * Channel A Force Scale

Channel B Force = Channel B Pressure * Channel B Force Scale

Actual Force = Channel A Force - Channel B Force + Dual Channel Force Offset

See Also

Input Type: Single-Input Force | Input Type: Single-Input Force

3.3.4.7. Input Type: Custom

Custom feedback refers to feedback that is continuously calculated by the user, such as with a user program. Custom feedback is not assigned to any hardware input. The axis will use this calculated feedback for control.

The following applications can be done using custom feedback:

- Switching Feedback
- Feedback Linearization using Curves
- Feedback Linearization Using Mathematical Formula
- Redundant Feedback using Custom Feedback

For more details, see Custom Feedback.

See Also

Custom Feedback

3.4. Halts

3.4.1. Halts Overview

Halts stop motion on an axis. Halts will also stop any Tasks that are running, if the Programming Properties are set to do so. The RMC has four types, or levels, of halts to safely stop the axis in various circumstances. These halts are well-suited for use when error conditions occur. The RMC, by default, triggers a halt when any Error Bits turns on.

Tip:
The Halts do more than just stop the axis. If you simply wish to stop the motion in closed loop control, use the Stop (Closed Loop) (6) command instead. If you wish to stop the motion in open loop control, use the Stop (Open Loop) (22) or Open Loop Rate (10) commands.

Halt Types

The RMC has four types of halts:

- **External Halt**
  This halt only sets External Halt status bit. This halt type is intended to signal the PLC to initiate its fault handling.
- **Closed Loop Halt**
  The axis stays in closed loop and ramps down the Target Velocity to zero at the rate specified by the Closed Loop Halt Deceleration parameter. If an axis is in Open Loop, this halt will automatically be promoted to an Open Loop Halt.
- **Open Loop Halt**
  The axis will be put in open loop and the Control Output will be ramped down to the value of the Output Bias parameter using the Open Loop Halt Ramp parameter.
- **Direct Output Halt**
  The axis will be put in Direct Output and the Control Output will be ramped down to the value of the Output Bias parameter using the Open Loop Halt Ramp parameter. When the Control Output reaches the Output Bias, the Enable Output will be turned off.
- Disable Axis Halt
  The axis will be disabled so that it will not accept any motion commands, will be put in Direct Output, and the Control Output will be ramped down to the value of the Output Bias parameter using the Open Loop Halt Ramp parameter. When the Control Output reaches the Output Bias, this halt turns off the Enable Output.

Halt Actions

When a halt occurs on an axis, the RMC takes the following actions:

- The axis begins the halt, as specified above.
- When a halt other than an External Halt occurs on an axis, the Halted status bit is set and the axis is said to be halted. If an External halt occurs, the External Halt status bit is set instead.
- If the axis is part of a Halt Group, it starts the same level of halt on all axes in the group. See the Halt Group Number topic for more details.
- The RMC immediately stops all Tasks by default. This setting can be changed on the Programming Properties dialog.

  Caution:
  If you disable this feature, User Programs may still be running after a halt occurs and may cause motion on the axis. Make sure you handle the halt condition safely. One method is to create a User Program to handle the halt condition and use the Program Triggers to start the User Program when the Halt bit turns on.

Pressure/Force Control

The Closed Loop Halt and Open Loop Halt do remove the axis from pressure or force control. The Direct Output Halt (3) and Fault Controller (8) commands will.

Triggering a Halt

Halts can then be triggered in two ways:

- **Issue a Command**
  By issuing commands (except External Halt). See each halt command topic for details.

1.

Halt	Command

Closed Loop Halt	Closed Loop Halt (1)
Open Loop Halt	Open Loop Halt (2)
Direct Output Halt	Direct Output Halt (3)

2.

- **Auto Stop**
  Auto Stops automatically trigger a halt when an Error Bit is set. The Auto Stops can be configured to trigger any level of halt, or Status Only, for each individual Error bit. See the Auto Stops topic for details.
- **Fault Controller (8) Command**
  The Fault Controller (8) command will halt all the axes.

See Also

Error Bits | Programming Properties

3.4.2. External Halt

The External Halt is not intended to actually halt motion; rather, it is intended only to signal a master controller, such as a PLC, that an axis should be halted, and the PLC should then take care of the halting. The External Halt does not affect motion in any way. An External Halt will not stop any user programs. The External Halt is one of the four types of RMC Halts.

When an External Halt occurs, only the following happens:

- The External Halt status bit is set.
- If the axis is part of a Halt Group, it sets the External Halt bit on all axes in the group.

If the External Halt was triggered because of the occurrence of a No Transducer or Transducer Overflow error, which in themselves may severely impact motion, the feedback value will be held at its last valid value.

Triggering an External Halt

An External Halt can only be triggered via Auto Stops.

After a Halt has Occurred

If the halt was caused by an Auto Stop, you should first make sure the error condition that caused it has been resolved before continuing. Once it has been fixed, you can clear the External Halt status bit by issuing the Clear Faults (4) command. Issuing a valid motion command will also clear the External Halt status bit, if the underlying error condition has gone away.

See Also

Halts Overview

3.4.3. Closed Loop Halt

The Closed Loop Halt is one of the four types of RMC Halts. Use the Closed Loop Halt to halt the axis while remaining in closed loop control.

Tip:
The Halts do more than just stop the axis. If you simply wish to stop the motion in closed loop control, use the Stop (Closed Loop) (6) command instead. If you wish to stop the motion in open loop control, use the Stop (Open Loop) (22) or Open Loop Rate (10) commands.

When a Closed Loop Halt occurs, it takes the following steps:

- The axis stays in closed loop and stops the axis at the rate specified by the Closed Loop Halt Deceleration axis parameter. If the axis is in Open Loop when this halt occurs, this halt will start an Open Loop Halt instead.
  See the **Stopping Details** section below for details on how the stop behaves for various control modes.

  Note:
  It is the *Target* Velocity that is ramped down. If the Actual Position is lagging behind the Target Position, it will not stop until it reaches the Target Position. If you need the Actual Position to stop immediately, use an Open Loop Halt or Stop (Open Loop) (22) command instead.
  Note:
  The deceleration specified by the Closed Loop Halt Deceleration is the *average* deceleration. The instantaneous deceleration may exceed this value.
  Note:
  The Command Position is not affected by a Closed Loop Halt.

- The Halted status bit is set and the axis is said to be in the halted state.
- If the axis is part of a Halt Group, it starts a Closed Loop Halt on all axes in the group.

Pressure/Force Control

The Closed Loop Halt will keep the axis in pressure or force control, and will stop the target immediately.

Pressure/Force Limit

The Closed Loop Halt will keep the axis in pressure or force limit, and will not stop the target.

Stopping Details

This command ramps the current velocity or rate to zero in closed loop. The behavior of this command depends on the type of axis control:

- **Position PID**
  The velocity will ramp down from the current velocity to zero at the rate specified by the Closed Loop Halt Deceleration parameter, while remaining in position control.
- **Position I-PD**
  If the last motion command was Move Absolute (I-PD) (28) or Move Relative (I-PD) (29), the Target Position will stop immediately. Otherwise, the velocity will ramp down from the current velocity to zero at the rate specified by the Closed Loop Halt Deceleration parameter, while remaining in position control.
- **Velocity PID**
  The velocity will ramp down from the current velocity to zero at the rate specified by the Closed Loop Halt Deceleration parameter, while remaining in velocity control.
- **Velocity I-PD**
  If the last motion command was Move Velocity (I-PD) (38), the Target Velocity will stop immediately. Otherwise, the velocity will ramp down from the current velocity to zero at the rate specified by the Closed Loop Halt Deceleration parameter, while remaining in velocity control.
- **Open Loop**
  The axis transitions to closed-loop control and ramps the velocity to zero at the rate specified by the Closed Loop Halt Deceleration parameter in V/sec.

- **Pressure/Force Control**
  The Target Pressure Rate or Target Force Rate will be ramped to zero at the rate specified by the Closed Loop Halt Deceleration parameter in Pr/sec or Fr/sec.

Triggering a Closed Loop Halt

A Closed Loop Halt can be triggered in two ways:

- By issuing the Closed Loop Halt (1) command.
- Via Auto Stops.

After a Halt has Occurred

If the halt was caused by an Auto Stop, you should first make sure the error condition that caused it has been resolved before continuing. Once it has been fixed, you can clear the Halted status bit by issuing the Clear Faults (4) command. Issuing a valid motion command will also clear the Halted status bit.

Why Bother?

This halt is useful when you want to stop the axis but remain in closed loop control. If you have to stop the axis because it is vibrating, use the Open Loop Halt instead, since the axis will probably keep vibrating or oscillating as long as it is in closed loop control.

The Auto Stops can be set up to cause this halt when an error bit turns on.

See Also

Halts Overview | Closed Loop Halt (1) Command

3.4.4. Open Loop Halt

The Open Loop Halt is one of the four types of RMC Halts. Use the Open Loop Halt to halt the axis in open loop control.

When an Open Loop Halt occurs, it takes the following steps:

- It puts the axis in open loop and ramps down the Control Output to the Output Bias using the Open Loop Halt Ramp parameter.
- The Halted status bit is set and the axis is said to be in the halted state.
- If the axis is part of a Halt Group, it starts an Open Loop Halt on all axes in the group.

Pressure/Force Limit

The Open Loop Halt will ramp the Control output to zero, but will not remove the axis from pressure/force limit. The Direct Output Halt (3) and Fault Controller (8) commands will remove it from pressure/force limit.

Tip:
The Halts do more than just stop the axis. If you simply wish to stop the motion in open loop control, use the Stop (Open Loop) (22) or Open Loop Rate (10) commands. If you wish to stop the motion in closed loop control, use the Stop (Closed Loop) (6) command instead.

Triggering an Open Loop Halt

An Open Loop Halt can be triggered in two ways:

- By issuing the Open Loop Halt (2) command.
- Via Auto Stops.

After a Halt has Occurred

If the halt was caused by an Auto Stop, you should first make sure the error condition that caused it has been resolved before continuing. Once it has been fixed, you can clear the Halted status bit by issuing the Clear Faults (4) command. Issuing a valid motion command will also clear the Halted status bit.

Why Bother?

This halt is useful when you want to set the Control Output to zero because of a potentially dangerous error. The Open Loop Halt Ramp parameter is used to avoid an abrupt (and potentially damaging) stop. If you just want to stop the axis but still have closed loop control over it, use the Closed Loop Halt.

The Auto Stops can be set up to cause this halt when an error bit turns on.

See Also

Halts Overview | Open Loop Halt (2) Command

3.4.5. Direct Output Halt

The Direct Output Halt is one of the four types of RMC Halts. Use the Direct Output Halt to halt the axis, put it in Direct Output, and turn off the Enable Output.

When a Direct Out Halt occurs, it takes the following steps:

- It puts the axis in Direct Output, turns on the Direct Output status bit, and ramps down the Control Output to the Output Bias using the Open Loop Halt Ramp parameter.
- When the Control Output reaches the Output Bias, this halt turns off the Enable Output.
- The Halted status bit is set and the axis is said to be in the halted state.
- If the axis is part of a Halt Group, it starts a Direct Output Halt on all axes in the group.

Tip:
The Halts do more than just stop the axis. For example, the Direct Output Halt will turn off the Enable Output as well. If you simply wish to stop the motion in open loop control, use the Stop (Open Loop) (22) or Open Loop Rate (10) command. If you wish to stop the motion in closed loop control, use the Stop (Closed Loop) (6) command instead.

Triggering a Direct Output Halt

A Direct Output Halt can be triggered in two ways:

- By issuing the Direct Output Halt (3) command.
- Via Auto Stops.

After a Halt has Occurred

If the halt was caused by an Auto Stop, you should first make sure the error condition that caused it has been resolved before continuing. Once it has been fixed, you can clear the Halted status bit by issuing the Clear Faults (4) command. Issuing a valid motion command will also clear the Halted status bit.

Why Bother?

This halt is useful when you want to set the Control Output to zero and turn off the Enable Output because of a potentially dangerous error. The Open Loop Halt Ramp parameter is used to avoid an abrupt (and potentially damaging) stop. If you wish to halt the axis but still have closed loop control over it, use the Closed Loop Halt.

The Auto Stops can be set up to cause this halt when an error bit turns on.

See Also

Halts Overview | Direct Output Halt (3) Command

3.5. Control Modes

3.5.1. Control Modes Overview

This topic describes various modes of control. For details on motion types, see the Controller Features Overview.

Open Loop Control

Open Loop Control is the simplest form of control. A control output is given to a system, for example 2 volts, causing the motion system to move at some approximate speed. Open loop control does not use the feedback to determine how much Control Output should be given. Therefore, in open loop control, there is no way of commanding the system to go exactly at a specific speed or go to an exact position. See the Open Loop Control topic for more details.

Closed Loop Control

Closed Loop control uses feedback from the system being controlled. For example, a command is issued to go to 20 inches. The RMC computes a target (a motion path) to get to 20 inches. For each control loop, the controller (the RMC) uses the feedback and the gains to compute the amount of Control Output that should be given to the system so that it follows the profile. The system will automatically go to 20 inches. See the Closed Loop Control topic for more details.

The RMC offers several different algorithms for closed-loop control, depending on the feedback type:

Feedback Type	Closed-Loop Control Algorithms
Position	Position PID **Advanced:** Position I-PD Velocity PID Velocity I-PD
Velocity	Velocity PID **Advanced:** Velocity I-PD
Pressure	Pressure/Force Control, Pressure/Force Limit
Force	Pressure/Force Control, Pressure/Force Limit

High-Order Control

The RMC also supports Acceleration Control and Active Damping, for difficult-to-control systems such as pneumatic cylinders.

Other

Cascaded Loops

Gain Scheduling

Unidirectional Mode

Unidirectional Mode, also known as Absolute Mode, is specifically designed for systems that require a unipolar control signal. Some common cases are:

- **Two Hydraulic Valves**
  Some hydraulic systems have two valves, one for direction control, the other for flow control. Unidirectional mode is for controlling the flow valve. The directional valve must be controlled by other means, but Unidirectional Mode does provide for easy switching of the control direction based on the directional valve setting.
- **Unidirectional Belt**
  A belt that must always move in the same direction.

For more details, see the Unidirectional Mode axis parameter.

Velocity and Torque Drives

Most actuators, together with their power source and/or drive electronics, can be classified as velocity mode or torque mode. Which type it is affects the tuning procedure and how the actuator handles certain RMC commands.

A velocity mode actuator produces a speed proportional to the Control Output. A torque mode actuator produces a torque or force proportional to the Control Output.

For details, see the Velocity and Torque Drives topic.

See Also

Closed Loop Control | Open Loop Control | Velocity and Torque Drives

3.5.2. Closed Loop Control

Closed Loop control uses feedback from the system being controlled. First, the RMC generates a target (position, velocity, pressure, or force) which specifies where the axis should be at each moment in order to move to the requested position. Then, the RMC compares the feedback to the target and calculates how much Control Output should be given to make the feedback match the target. The RMC repeats this for every control loop time. All closed-loop commands specify the target to be generated. The tuning parameters specify how to calculate the Control Output, which makes the axis follow the target.

Closed Loop Control Algorithms

The RMC offers several different algorithms for closed-loop control, depending on the feedback type:

Feedback Type	Closed-Loop Control Algorithms
Position	Position PID **Advanced:** Position I-PD Velocity PID Velocity I-PD
Velocity	Velocity PID **Advanced:** Velocity I-PD
Pressure	Pressure/Force Control and Pressure/Force Limit
Force	Pressure/Force Control and Pressure/Force Limit

Dual-Loop Control

The RMC supports dual-loop control, for example position-pressure or position-force. This allows controlling two quantities with a single actuator. For example, consider an injection molding application. The system first moves in position control to inject the material, then needs to

maintain a certain force. Using dual-loop control, this can be done with a single actuator (typically a hydraulic valve).

On the RMC, dual-loop control requires one Control Output, one position (or velocity) feedback, and a pressure or force input (this is called the secondary feedback).

Dual-Loop Control Requirements

To define a dual-loop control axis, the RMC requires:

RMC75: Dual-loop control requires that the secondary feedback is from an AP2 module.

RMC150: Dual-loop control requires a pressure-control enabled RMC150, designated as an RMC151.

RMC200: Each dual-loop control axis requires two control loops on the Feature Key.

See the Position-Pressure and Position-Force Control topic for more details.

Closed Loop Commands

See the List of Commands topic for a list of closed-loop commands.

See Also

Control Modes Overview | Open Loop Control

3.5.3. Open Loop Control

Open Loop Control is the simplest form of control. A signal is given to a system, for example, 2 volts of Control Output to a valve, causing a cylinder to extend at a speed determined by the system characteristics (valve, cylinder and load, or drive, motor and load). In open loop control, there is no way of commanding the system to go exactly at a specific speed or go to an exact position.

Sometimes, improperly tuned closed loop control may cause a system to oscillate or exhibit other erratic behavior. Usually, issuing an open loop command with 0 volts of Control Output will stop the system safely. See Auto Stops for details on halting a system.

Some of the RMC open loop commands have a ramp rate parameter that specifies the rate at which the Control Output ramps up to the Requested Output. The ramp avoids jerking the system.

Open Loop Commands

The RMC has the following open-loop commands:

- Open Loop Rate (10)
  This is the basic open-loop command. It ramps the Control Output from the current value to the Requested Output at the rate specified by the Ramp Rate parameter. Use this command for normal open-loop moves.
- Direct Output (9)
  This command is similar to the Open Loop Rate command, except it disables all the safety features of the RMC. It is intended only for testing the Control Output. Use the Open Loop Rate command for all other purposes.
- Open Loop Halt (2)
  This halt ramps the Control Output from its current value to zero. It also performs other halt actions.
- Open Loop Absolute (11) and Open Loop Relative (12)
  These are specialty open loop commands. They specify the Control Output as a function of distance. As such, they do require position feedback on the axis.

Partial Open Loop Commands

The following commands use open-loop control for part of the motion. As the axis reaches the requested position, the axis switches to closed loop control, decelerates, and holds position. These commands are useful for fast motion.

- Quick Move Absolute (15)
- Quick Move Relative (16)

See Also

Control Modes Overview | Closed Loop Control

3.5.4. Position PID

Position PID is the algorithm typically used to perform closed-loop motion control on a position feedback axis. PID stands for the central gains used in this mode: Proportional, Integral, and Differential. The Position PID provides very good control and is suitable for nearly all motion control systems with position feedback.

The Position PID works on a position feedback only and controls both position and velocity. In certain advanced applications, other control modes may be preferred, such as Position I-PD, Velocity PID, or Velocity I-PD.

Position PID Advantages

- Tracks position very well.
- Is very well understood by most people in the motion control industry.

Position PID Disadvantages

- May chatter or oscillate when following an irregular target, such as a step jump or a noisy reference signal.
- Tendency to overshoot final position on some systems.

Motion Commands in Position PID Mode

The default control mode of a position axis is Position PID. If the axis is not in Position PID, use the Set Pos/Vel Ctrl Mode (68) command to set the Next Pos/Vel Control Mode to Pos PID. The next closed-loop motion command will use the control mode specified in the Next Pos/Vel Control Mode status register. The Current Control Mode register indicates the mode currently in use.

See the Closed Loop Control topic for details on which commands are supported in Position PID control.

Algorithm

Each closed loop motion command issued to the RMC specifies a target profile, which defines where the axis should be at any given moment. For each loop time when the axis is in closed loop control, the RMC uses the specified target profile to calculate the desired position of the axis at that moment (called the Target Position) and subtracts the Actual Position to determine the Position Error. The Position PID algorithm then uses this information, together with the gains and feed forwards, to calculate how much Control Output should be generated to move the axis to the Target Position. The values of the gains and feed forwards must be set to achieve proper control. The process of setting the gains is called tuning and is done as part of the setup procedure.

The Position PID uses the gains and feed forwards listed below. Each gain or feed forward is multiplied by some quantity related to the Target Position and Actual Position to come up with a percentage. The resulting percentages are all summed and then multiplied by the maximum output (typically 10V), to come up with the Control Output voltage for that loop time.

- Proportional Gain
  The Proportional Gain is multiplied by the Position Error. This is the most important gain.

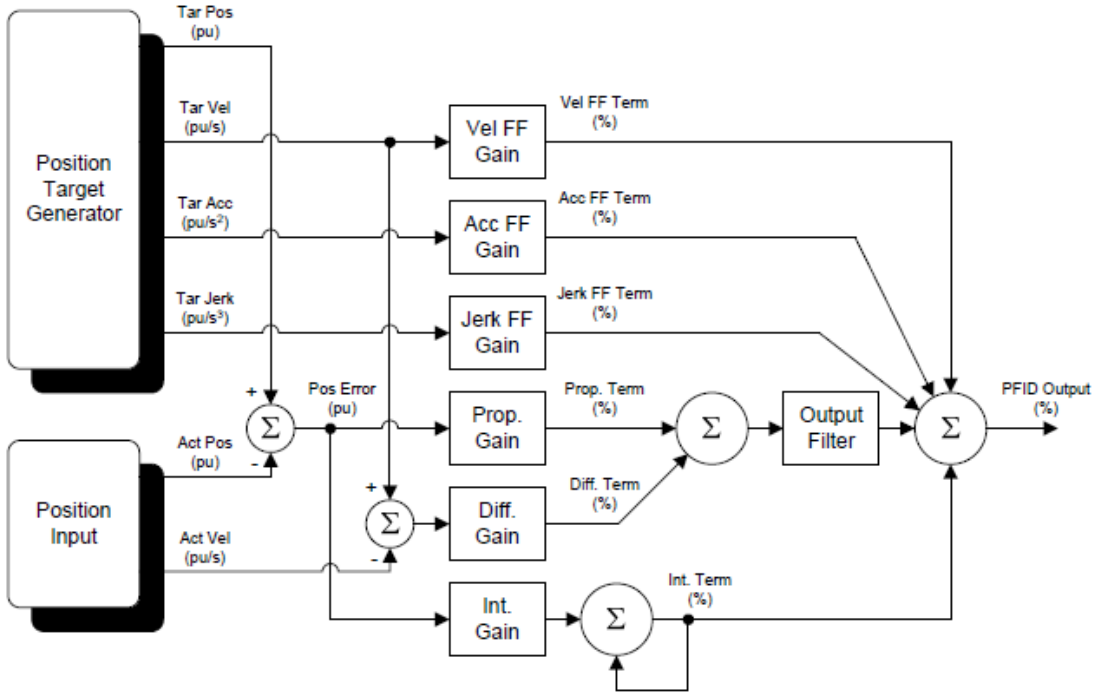
- Integral Gain
  The Integral Gain is multiplied by the accumulated Position Error. This helps the axis get into position over time.
- Differential Gain
  The Differential Gain is multiplied by the difference between the Target and Actual Velocities. This helps the axis keep up with quick changes in velocity.
- Velocity Feed Forward
  The Velocity Feed Forward is multiplied by the Target Velocity.
- Acceleration Feed Forward
  The Acceleration Feed Forward is multiplied by the Target Acceleration.
- Jerk Feed Forward
  The Jerk Feed Forward is multiplied by the Target Jerk. The Jerk Feed Forward is not necessary for most applications.

In addition, higher-order gains may be used if Acceleration Control or Active Damping are selected.

Tuning Position PID

See the Tuning Overview topic for details. The Tuning Wizard can be used to tune position PID control.

Diagram



See Also

Control Modes Overview

3.5.5. Velocity PID

Velocity PID is the algorithm typically used to perform closed-loop velocity control on a position or velocity axis. PID stands for the central gains used in this mode: Proportional, Integral, and Differential. The Velocity PID provides very good control and is suitable for nearly all motion control systems with velocity feedback. In certain cases, Velocity I-PD control may be preferred.

Velocity PID Advantages

- Excellent for controlling an axis that follows a smooth target, such as one generated by the RMC motion commands.

Velocity PID Disadvantages

- May not control very well with an irregular target, such as step jumps or a joystick.

Motion Commands in Velocity PID Mode

To use Velocity PID, use the Set Pos/Vel Ctrl Mode (68) command to set the Next Pos/Vel Control Mode to Vel PID. The next closed-loop motion command will use the control mode specified in the Next Pos/Vel Control Mode status register. The Current Control Mode register indicates the mode currently in use.

See the Closed Loop Control topic for details on which commands are supported in Velocity PID control.

Algorithm

Each closed loop motion command issued to the RMC specifies a target profile, which defines where the axis should be at any given moment. For each loop time when the axis is in closed loop control, the RMC uses the specified target profile to calculate the desired velocity of the axis at that moment (called the Target Velocity) and subtracts the Actual Velocity to determine the Velocity Error. The Velocity PID algorithm then uses this information, together with the gains and feed forwards, to calculate how much Control Output should be generated to move the axis to the Target Velocity. The values of the gains and feed forwards must be set to achieve proper control. The process of setting the gains is called tuning and is done as part of the setup procedure.

The Velocity PID uses the gains and feed forwards listed below. Each gain or feed forward is multiplied by some quantity related to the Target Velocity and Actual Velocity to come up with a percentage. The resulting percentages are all summed and then multiplied by the maximum output (typically 10V), to come up with the Control Output voltage for that loop time.

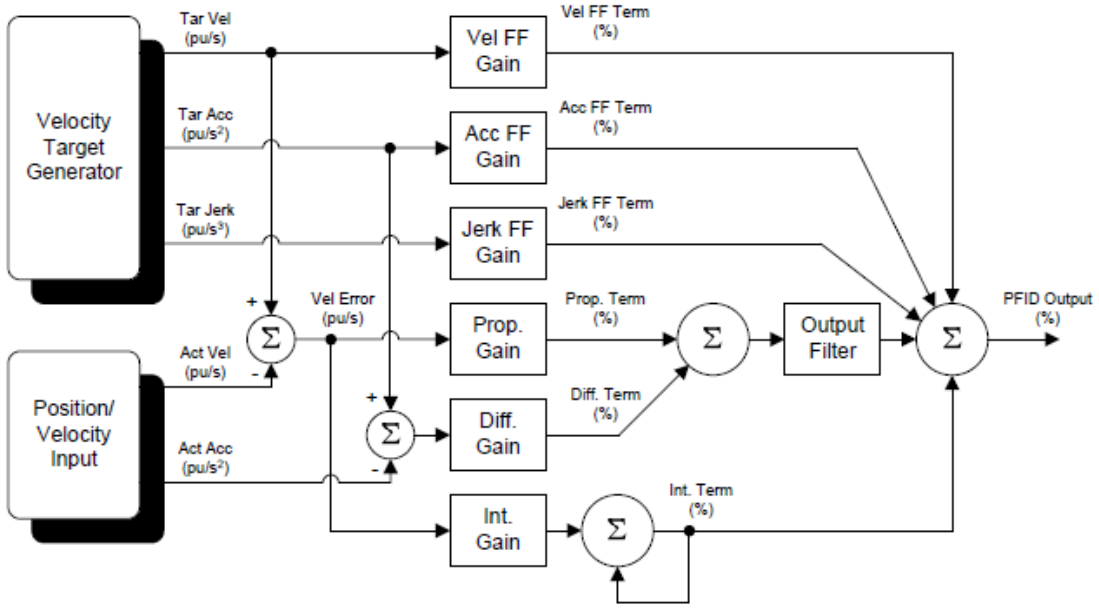
- Proportional Gain
  The Proportional Gain is multiplied by the Velocity Error. This is the most important gain.
- Integral Gain
  The Integral Gain is multiplied by the integrated (sum of value x time) Velocity Error. This helps the axis get to velocity over time.
- Differential Gain
  The Differential Gain is multiplied by the difference between the Target and Actual Accelerations. This helps the axis keep up with quick changes in velocity.
- Velocity Feed Forward
  The Velocity Feed Forward is multiplied by the Target Velocity.
- Acceleration Feed Forward
  The Acceleration Feed Forward is multiplied by the Target Acceleration.
- Jerk Feed Forward
  The Jerk Feed Forward is multiplied by the Target Jerk. The Jerk Feed Forward is not necessary for most applications.

In addition, higher-order gains may be used if Acceleration Control or Active Damping are selected.

Tuning Velocity PID

The velocity PID gains must be tuned manually. The Tuning Wizard cannot be used to tune velocity PID control.

Diagram



See Also

Velocity Control | Velocity I-PD | Control Modes Overview

3.5.6. Gain Sets Overview

Most control applications require only one set of gains that never change. However, some applications may require different gains at different times. For example, a press may switch from a large valve to a small valve, or an application may require different gains for Position I-PD and Velocity I-PD control modes. The Gain Sets provides a method of storing two separate sets of gains and applying those gains as needed.

Each position or velocity axis can have two **gain sets**. The RMC applies the gains from one of the sets according to the selected option of the Gain Sets axis parameter, listed in the table below. The Current Gain Set axis status register displays the currently applied gain set.

Gain Set Option	Description
Single (default)	Only one gain set is available. This option is the best choice for most applications.
Dual (RMC200 only)	Two gain sets are available. Use the Select Gain Set (75) to switch between gain sets. The axis will use gain set #0 when the RMC powers up.
Pos, Vel	Automatically chooses a gain set based on position or velocity control. Gain Set#0 applies when the Current Control Mode is Position PID or Position I-PD.

	Gain Set#1 applies when the Current Control Mode is Velocity PID or Velocity I-PD.
PID, I-PD	Automatically chooses a gain set based on PID or I-PD control. Gain Set#0 applies when the Current Control Mode is Position PID or Velocity PID. Gain Set#1 applies when the Current Control Mode is Position I-PD or Velocity I-PD.

Notice that using velocity or I-PD control does not necessarily require using dual gain sets. If you have a single gain set, that set will always be used for the position or velocity control, whether it be PID or I-PD. If you will be using both position and velocity or both PID and I-PD on the same axis, the gain sets will be useful to you.

The Symmetrical/Ratioed parameter always applies to both gain sets.

The Gains Sets feature is related to, but distinct from, ratioed gains in the forward and reverse direction of motion on asymmetrical systems, and gain scheduling, where the gain values are continuously modified based on the state of the system.

Choosing a Gain Set Option

To set the Gain Set option, use the Gain Sets parameter. This parameter is located in the Axis Tools, Axis Parameters pane, on the **All** tab, in the **Position/Velocity Control** section.

After choosing an option other than Single, two gain sets will appear. The tuning tools will always apply to Gain Set 0. You can tune up the axis using the Tuning Tools, then copy the gains to gain set 1.

See Also

Symmetrical/Ratioed | Current Gain Set | Current Control Mode

3.5.7. Ratioed Gains

Position or velocity control axes that behave differently in either direction of motion are called asymmetric systems. Such systems require different gains in each direction. The RMC can ratio the gains such that the system controls identically in both directions. A single-rod hydraulic cylinder is an asymmetrical system. A motor with the same load in both directions is typically symmetrical.

A related concept is using different gains for different control types. See the Gain Sets Overview topic for details.

To apply different directional gains to an asymmetrical system, the RMC *ratios* the gains. To ratio the gains, set the Symmetrical/Ratioed parameter to Ratioed. This creates two Velocity Feed Forwards, one for each direction. Once the Velocity Feed Forwards have been tuned, the gains in the direction of the highest Velocity feed Forward will be applied as they are. The gains in the direction of the lowest Velocity feed Forward will be decreased by the ratio of the Velocity Feed Forwards. This matches the PID output to the system dynamics in each direction, resulting in very good control of the cylinder.

The RMC uses the Target Velocity and Position Error to determine whether to use the forward or reverse gains. If there is a non-zero Target Velocity, then its sign determines which directional gains to use. If the Target Velocity is zero, then the Position Error (or Velocity Error for Velocity Control) determines the intended direction of movement.

Note:
Ratioed gains are automatically selected by the Tuning Wizard if moves in both directions are used.

Example

Consider a single-rod hydraulic cylinder that moves at 3 in/sec when a Control Output of 1V is applied and -2.3 in/sec when a Control Output of -1V is applied. Since it moves faster in the positive direction, those gains need to be smaller.

The correct Positive Velocity Feed Forward for this system is:

1 [V] x 10 [%/V] / 3 [in/s] = **3.33 [%/in/s]**

The correct Negative Velocity Feed Forward is:

-1 [V] x 10 [%/V] / -2.3 [in/s] = **4.35 [%/in/s]**

By choosing Ratioed gains, the gains in the negative direction will be applied as they are. The gains in the positive direction will be multiplied by 3.33/4.35, which is 0.766.

(*NOTE:* The RMC200 Control Output is in percentage in RMCTools vs. Voltage for the RMC75/150.)

Choosing Ratioed Gains

Use the Symmetrical/Ratioed parameter to choose ratioed gains. This parameter is located in the Axis Tools, Axis Parameters pane, on the **Tune** tab.

See Also

Gain Sets Overview

3.5.8. Gain Scheduling

Gain scheduling is the process of dynamically changing the gains of an axis based on some *scheduling variable*. The scheduling variable may be any measurable quantity in any RMC register, such as axis position, pressure, temperature, etc.

For example, an axis may require different gains based on its position. In this case, the axis' position is the scheduling variable. By defining the gains as a function of the axis's position, the gain scheduling results in optimum control.

Gain Scheduling Using Curves

If the axis only needs two different sets of gains, dual gain sets may be used for gain scheduling. This is typically used for systems that change drastically at some well-define point, such as switching from a large valve to a small valve at some point in the travel.

Gain Scheduling Using Curves

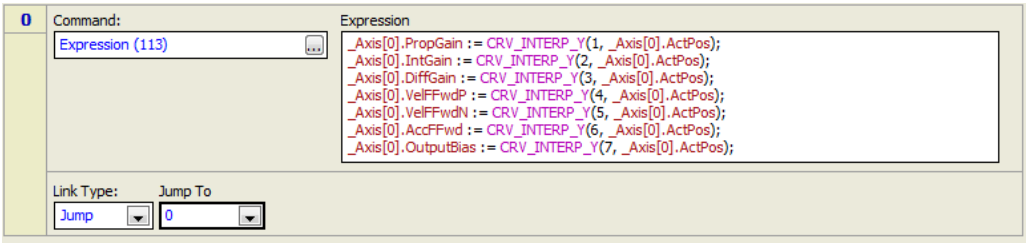
Gain scheduling can be implemented in the RMC by using curves and the curve interpolation functions. Gain scheduling involves these steps:

1. **Determine the gain values at several points**
   Tune the axis for several different values of the scheduling variable. For example, if the scheduling variable is the axis position, you should tune the axis at the ends, and perhaps one or more positions in the middle. Record the gains for each position.

2. **Define the gain curve for each gain**
   Using the Curve Tool and the gain values you recorded, create a curve for each gain. The scheduling variable will be the x values of the curve, and the gain values will be the y values. The curve interpolates between the points you entered, providing a smooth gain scheduling function. Cubic curves will provide the smoothest transition between gains, although linear curves will work as well. If you choose a cubic curve, choose the Natural Velocity Endpoint Behavior. Constant curves are not recommended, as they will cause undesirable discontinuities in the gain terms, especially the Feed Forwards.

3. **Continually apply the gain to the axis**
   Create a user program that uses the CRV_INTERP_Y function to get the curve values for the current scheduling variable value, and apply the values to the gains. The user program must run continuously.

   For example, the user program below will apply the gains and the output bias based on the axis position. The expression can easily be expanded to handle multiple axes.



   To successfully use this program, make sure the RMC is in RUN mode. This can be done by setting the RMC to start up in RUN mode in the Programming Properties, on the **RUN/PROGRAM** page. Also, in the Programming Properties, on the **Halts** tab, make sure the task this program is running on is set to halt. To start this user program when the RMC starts up in RUN mode, use the _FirstScan tag on the Program Triggers.

See Also

CRV_INTERP Functions

3.5.9. Unidirectional Mode

Unidirectional Mode, also known as Absolute Mode, is specifically designed for systems that require a unipolar control signal. Some common cases are:

- **Two Hydraulic Valves**
  Some hydraulic systems have two valves, one for direction control, the other for flow control. Unidirectional mode is for controlling the flow valve. The directional valve must be controlled by other means, but Unidirectional Mode does provide for easy switching of the control direction based on the directional valve setting.
- **Unidirectional Belt**
  A belt that must always move in the same direction.

Unidirectional Mode will prevent the Control Output from going negative even if the Actual overshoots the Target. When this occurs, the Control Output will be truncated at the Output Bias, and the Integral Term will not wind up.

Unidirectional Mode applies to closed-loop position, velocity, pressure, and force control. It does not affect the Open Loop Rate (10) and Direct Output (9) commands, but it does affect the Quick Move Absolute (15), Quick Move Relative (16), Open Loop Absolute (11), and Open Loop Relative (12) commands. The open loop portion of these moves is adjusted in the same way as a closed loop move.

For more details, see the Unidirectional Mode axis parameter.

See Also

Unidirectional Mode Parameter | Set Control Direction (96)

3.5.10. Velocity and Torque Mode

Most actuators, together with their power source and/or drive electronics, can be classified in two types: *velocity mode* or *torque mode*. Which type it is affects the tuning procedure and how the actuator handles certain RMC commands.

Definition

The actuator type is defined by its response to the Control Output voltage applied by the RMC:

- A **velocity mode** actuator produces a *speed* proportional to the Control Output.
- A **torque mode** actuator produces a *torque* or *force* proportional to the Control Output.

In practice, this means that if you issue an Open Loop Rate command with a certain amount of Control Output to a velocity mode actuator, the actuator will move at a speed roughly proportional to the Control Output voltage. If you issue the Open Loop Rate command to a torque mode actuator, the actuator will provide a torque proportional to the voltage. The actuator speed will keep increasing until the torque is equal to the friction in the system. The final drive speed for a torque mode actuator is not necessarily proportional to the voltage.

Note:
It is not recommended that you issue Open Loop commands to a torque mode system during normal operation, because you cannot predict the final speed. Use closed loop commands instead.

Examples

Velocity mode actuators:

- Hydraulic cylinder
- Motors with a velocity drive

Torque mode actuators:

- Motors with an amplifier (without an inner velocity loop)

Effect on Tuning

Whether an actuator is velocity or torque mode affects the tuning of a system in the following ways:

- **Damping**
  Velocity mode systems typically have high damping because they use a velocity-mode drive, or for hydraulic cylinders, they have high damping during motion, and when stopped, don't move because of the low compressibility of oil. In these cases, the system has, or appears to have, significant damping. Torque mode systems, however, typically have little damping. That means that if given some Control Output, they will coast after the Control Output goes back to zero. When tuning a torque drive, some damping must be provided initially with the Differential gain. This is the primary difference between the tuning methods of velocity drives and torque drives.
- **Feed Forwards**
  On a velocity mode system, the Velocity Feed Forwards often provide most of the drive required to move the axis and may therefore be large. On a torque mode system, the Velocity Feed Forwards are basically only for overcoming friction and are often small. On torque mode systems, the Accel Feed Forwards do a lot of the work.

Effect on Commands

The Open Loop Absolute (11) and Open Loop Relative (12) commands are intended only for velocity drives. The Open Loop Rate (10) command can be used on either, but the Control Output voltage should be kept very low for torque mode systems to keep them from running away.

See Also

Control Modes Overview

3.5.11. Advanced

3.5.11.1. Position I-PD

Position I-PD is an algorithm that can be used to perform closed-loop motion control on a position axis. I-PD stands for the central gains used in this mode: Integral, Proportional, and Differential. The "-" indicates how the algorithm uses these gains.

For most motion applications, the Position I-PD is not as well-suited as the Position PID. However, the Position I-PD differs from the Position PID in that it can easily be tuned so that it does not overshoot. The Position I-PD is therefore very suitable for applications where the Target Position jumps. An example is an axis geared to a reference input that makes discrete jumps. The Position I-PD typically does not respond as quickly and does not use Feed Forwards. Therefore, it will always lag behind the Target Position when moving.

Position I-PD Advantages

- Easy to tune for minimal overshoot, making it excellent for controlling an axis that follows an irregular target, such as step jumps or a joystick.
- Is not disrupted by a saturated Control Output.

Position I-PD Disadvantages

- Not well suited for following a specific target profile.
- Does not track position changes very quickly. May take longer to get into position.

Motion Commands in Position I-PD Mode

The following commands are designed for use only with Position I-PD and automatically put the axis into the Position I-PD control mode during the commanded motion:

- Move Absolute (I-PD) (28)
  This command immediately sets the target to the requested position. It does step-jump; it does not ramp it.
- Move Relative (I-PD) (29)
  This command immediately sets the target to the requested distance from the specified position (Target, Actual, or Command). It does step-jump; it does not ramp it.

To use the Position I-PD control mode with other motion commands, first use the Set Pos/Vel Ctrl Mode (68) command to set the Next Pos/Vel Control Mode to Pos I-PD. The next closed-loop motion command will use the control mode specified in the Next Pos/Vel Control Mode status register. The Current Control Mode register indicates the mode currently in use.

See the Closed Loop Control topic for details on which commands are supported in Position I-PD control.

Special Notes

Decreasing Jerk at Start of Motion

When using the Move Absolute (I-PD)(28) and Move Relative (I-PD)(29) commands, the system will start moving with a sudden jerk. This is because the Target Position is set to the Command Position immediately. In many systems this is acceptable. If it is not acceptable for your system, you can instead use the Move Absolute (20) or Move Relative (21) commands in I-PD mode. With these commands, the Target Position is ramped toward the Command Position at the speed you specify (set the Accel and Decel to a high value, such as 1000). This will essentially eliminate the sudden jerk. This may increase the time it takes to get into position at the end of the move.

Fast Moves (Saturating the Output)

If the Control Output saturates, the I-PD is not disrupted like the PID is (with the PID, the Integrator doesn't handle it very well). Therefore, the I-PD can be used to move the system at it's maximum speed (typically the speed at 10V Control Output). To achieve this, with the Move Absolute (I-PD)(28) and Move Relative (I-PD)(29) commands, set the Maximum Speed command parameter to a value greater than you system's maximum speed. The moves will then saturate the Control Output during the move, indicating that it is moving at it's maximum speed. You will, of course, need to set the Output Saturated Auto Stop to Status Only.

Saturating the output can be very useful with non-linear valves where the gain "rolls off" at the upper end. With a PID, it is especially difficult to get close to maximum speed, because a small increase in speed can suddenly saturate the output. The I-PD makes it easy to get to maximum speed.

Position I-PD Algorithm

Each closed loop motion command issued to the RMC specifies a target profile, which defines where the axis should be at any given moment. For each loop time when the axis is in closed loop control, the Position I-PD algorithm calculates the values from each gain, as described below. Then, the terms from the Proportional and Differential gains are subtracted from the Integral Gain term. The resulting value (in percent) is multiplied by the maximum output (typically 10V), to come up with the Control Output voltage for that loop time.

Gains and Feed Forwards

The Position I-PD uses the gains listed below. It does not use any Feed Forwards.

- Integral Gain
  The Integral Gain is multiplied by the accumulated Position Error.
- Proportional Gain
  In this control mode, the Proportional Gain multiplied by the change in the Actual Position is subtracted from the Control Output each control loop.
- Differential Gain
  In this control mode, the Differential Gain multiplied by the change in the Actual Velocity is subtracted from the Control Output each control loop.

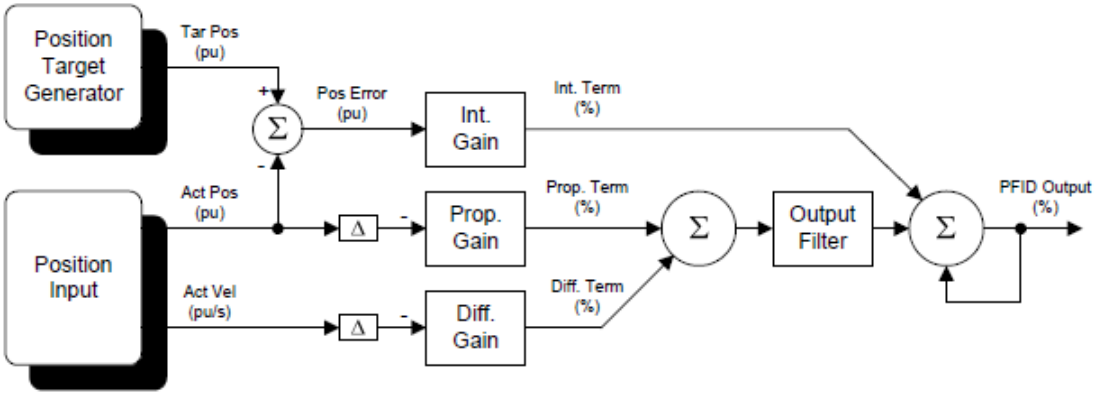
In addition, higher-order gains may be used if Acceleration Control or Active Damping are selected.

Tuning Position I-PD

The tuned position I-PD gains are typically the same values as the tuned position PID gains. Therefore, you can use the same tuning procedures for as for position PID. See the Tuning Overview topic for details. Keep in mind that the I-PD algorithm does not use the Velocity or Acceleration Feed Forwards.

You can also use the Tuning Wizard to tune I-PD control. However, if the gains are set to ratioed, the Velocity Feed Forwards are used in I-PD only to determine the ratio of the gains.

Diagram



See Also

Control Modes Overview

3.5.11.2. Velocity I-PD

Velocity I-PD is an algorithm that can be used to perform closed-loop velocity control on a velocity axis. I-PD stands for the central gains used in this mode: Proportional, Integral, and Differential. The "-" indicates how the algorithm uses these gains.

For most velocity applications, the Velocity PID is more suitable than the Velocity I-PD. The Velocity I-PD does not respond as quickly and does not use Feed Forwards. Therefore, it will always lag behind the Target Velocity when it changes. It is easier to tune for no overshooting and can handle step jumps better than the velocity PID.

Velocity I-PD Advantages

- Excellent for controlling an axis that follows an irregular signal, such as a reference signals that jumps, or a joystick.

Velocity I-PD Disadvantages

- Lags behind the Target Velocity.

Motion Commands in Velocity I-PD Mode

The following commands are designed for use only with Velocity I-PD and automatically put the axis into the Velocity I-PD control mode during the commanded motion:

- Move Velocity (I-PD) (38)
  This command immediately sets the Target Velocity to the requested velocity. It does step-jump; it does not ramp it.

To use the Velocity I-PD control mode with other motion commands, use the Set Pos/Vel Ctrl Mode (68) command to set the Next Pos/Vel Control Mode to Vel I-PD. The next closed-loop motion command will use the control mode specified in the Next Pos/Vel Control Mode status register. The Current Control Mode register indicates the mode currently in use.

See the Closed Loop Control topic for details on which commands are supported in Velocity I-PD control.

Algorithm

Each closed loop motion command issued to the RMC specifies a target profile, which defines where the axis should be at any given moment. For each loop time when the axis is in closed loop control, the Velocity I-PD algorithm calculates the values from each gain, as described below. Then, the terms from the Proportional and Differential gains are subtracted from the Integral Gain

term. The resulting value (in percent) is multiplied by the maximum output (typically 10V), to come up with the Control Output voltage for that loop time.

Gains and Feed Forwards

The Velocity I-PD uses the gains listed below. It does not use any Feed Forwards.

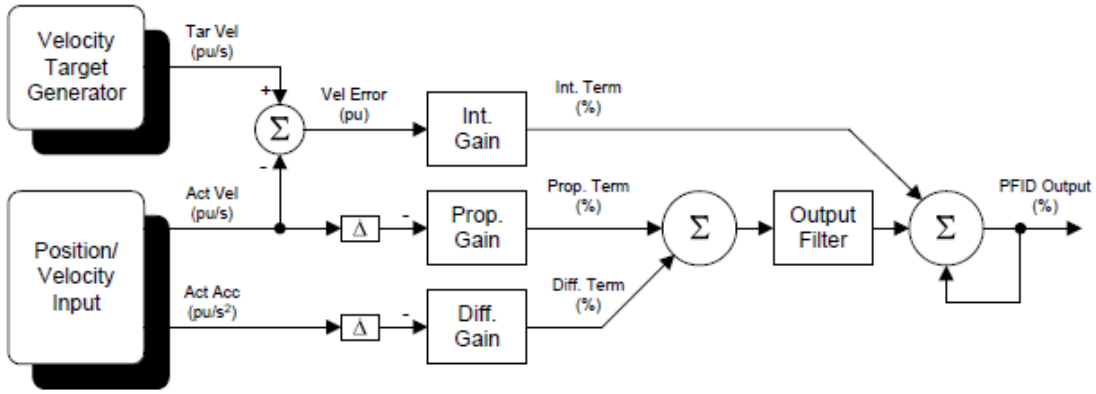
- Integral Gain
  The Integral Gain is multiplied by the accumulated Velocity Error.
- Proportional Gain
  In this control mode, the Proportional Gain multiplied by the change in the Actual Velocity is subtracted from the Control Output each control loop.
- Differential Gain
  In this control mode, the Differential Gain multiplied by the change in the Actual Acceleration is subtracted from the Control Output each control loop.

Higher-order gains may be used if Acceleration Control or Active Damping are selected. In addition, in Active Damping control, the Velocity I-PD does not use the Differential gain.

Tuning Velocity I-PD

The velocity I-PD gains must be tuned manually. The Tuning Wizard cannot be used to tune velocity I-PD control.

Diagram



See Also

Velocity Control | Velocity PID | Control Modes Overview

3.5.11.3. Active Damping

Active Damping is an algorithm that reduces oscillation in a motion control system. It can help control systems prone to oscillation or with low damping, such as pneumatic cylinders. Active damping requires information on the accelerations or forces acting on the system. The active damping algorithm uses this information to affect the Control Output so that the accelerations or forces do not cause oscillations.

Active damping gives best results when a secondary input is used. although in some cases, active damping can be achieved using only the primary position or velocity feedback.

Active Damping is especially useful for controlling fluid power systems with significant compression in the fluid. This includes:

- **Pneumatics**
  Due to the compressibility of air, pneumatic systems are notorious for oscillating. The active damping limits the oscillation, resulting in much better control.
- **Small Hydraulic Cylinders with Large Loads**
  In hydraulic cylinders with a large mass and a relatively small bore, the effects of the compressibility of the oil are the most pronounced, and can result in oscillation. Active damping limits the oscillation, resulting in much better control. In certain cases, the use of active damping can decrease the bore of the cylinder required, which can also reduce the size of the hydraulic power unit.

Deciding whether a System Needs Active Damping

To determine whether a system needs active damping, give it an open loop Control Output. If the system oscillates, especially after the initial start of motion, it could benefit from active damping.

Active Damping Options

Active damping requires information on the accelerations or forces acting on the system. This information can be obtained in a variety of ways. The table below lists the options available. Typically, using accelerometers or force feedback provides the best control. Position-based acceleration does not provide as good control.

Active Damping Type	Possible Sources	Required Axis Type
Force	Secondary Force Input (single- or dual-input)	Position-Force* or Velocity-Force*
Acceleration Input	Secondary Acceleration Input (single- or dual-input)	Position-Acceleration or Velocity-Acceleration
Position-based or or Velocity-based Acceleration	Position or Velocity Input (uses the Actual Acceleration of the feedback) • Unfiltered Acceleration (not recommended) • Filtered Acceleration (not recommended) • Modeled Acceleration	Position or Velocity

*In rare cases, Active Damping with pressure feedback is possible on a position-pressure or velocity-pressure axis, requiring a secondary pressure input, but is generally not recommended.

Setting Up Active Damping

Pressure or Force Input

a. Define a position-pressure or position-force axis. On the RMC75, this secondary pressure or force input requires an AP2 module. On the RMC150, the CPU must have the pressure control option, designated as RMC151.
b. Set the High-Order Control parameter to **Active Damping**.

Acceleration Input

a. Define an axis with a secondary acceleration input. If you are using one accelerometer, choose **Accel (single-input)**. If you are using two accelerometers, choose **Accel (dual-input, diff.)**. On the RMC75, the secondary acceleration input(s) require an AP2 module. On the RMC150, the CPU must have the pressure control option, designated as RMC151.
b. Set the High-Order Control parameter to **Active Damping**.

Position-based

a. Set the High-Order Control parameter to **Active Damping**. This is only valid if the axis is position only. If the axis has a secondary feedback, the active damping will use the secondary feedback.
b. Set the Acceleration Filter Type parameter to **Model** or **Low Pass**. This is necessary to obtain usable acceleration readings. See the modeling and filtering topics for details.

Velocity-based

a. Set the High-Order Control parameter to **Active Damping**. This is only valid if the axis is position only. If the axis has a secondary feedback, the active damping will use the secondary feedback.
b. Set the Acceleration Filter Type parameter to **Model** or **Low Pass**. This is necessary to obtain usable acceleration readings. See the modeling and filtering topics for details.

Tuning Active Damping

Active Damping adds the Active Damping Proportional Gain and Active Damping Differential Gain to the PID or I-PD control algorithm. See the Tuning Active Damping and Acceleration Control topic for details on tuning these gains.

See Also

High-Order Control | Acceleration Control

3.5.11.4. Acceleration Control

Acceleration Control is a High-Order Control option that adds high-order gains to the position control algorithm. These high-order gains operate on the higher-order derivatives of the controlled value. For example, on a position control axis, the Double Differential Gain operates on the second derivative of position.

Using acceleration control improves motion control in certain cases, such as for pneumatic systems.

The higher-order gains made available by Acceleration Control depends on the axis type:

Axis Type	Available High-Order Gains	Operates on:
Position Control	Double Differential Gain	Actual Acceleration based on the position input (2nd derivative of Actual Position)
Position-Acceleration Control	Double Differential Gain	Actual Acceleration from the secondary acceleration input
	Triple Differential Gain	Actual Jerk based on the secondary acceleration input (1st derivative of Actual Acceleration)
Velocity Control	Double Differential Gain	1st derivative of Actual Velocity
Velocity-Acceleration Control	Double Differential Gain	1st derivative of Actual Velocity

Effect on Control

Acceleration control can improve the motion control in certain cases because of the added terms.

For position only or velocity only control axes, the noise in the position or velocity measurement (mostly due to the conversion to digital values) requires the derived acceleration measurement to be filtered or smoothed quite a bit before it can be used. This introduces delays or other errors which diminish its effectiveness.

For position-acceleration or velocity-acceleration axes, the secondary input from an accelerometer provides very good acceleration measurements. However, this requires extra

components and wiring out to the moving load where the accelerometer is mounted. It is also generally necessary to use two accelerometers, one on the stationary frame and one on the moving load. This is because the stationary frame is not usually truly stationary—its vibrations will impact the motion of the load.

Setting up Acceleration Control

Acceleration control uses a dual-loop axis. On the RMC75, this requires an AP2 module. On the RMC150, the CPU must have the pressure control option, designated as RMC151.

To enable Acceleration Control on a position or velocity control axis:

- Set the High-Order Control parameter to **Acceleration Control**.

To enable Acceleration Control using accelerometers:

- Define an axis with a secondary acceleration input. If you are using one accelerometer, choose **Accel (single-input)**. If you are using two accelerometers, choose **Accel (dual-input, diff.)**.
- Set the High-Order Control parameter to **Acceleration Control**.

Note:
If you use the Tuning Wizard and choose a second-order model, the wizard will automatically set the High-Order Control parameter to Acceleration Control.

Tuning Acceleration Control

Manual Tuning

Acceleration Control adds the Double Differential Gain and Triple Differential Gain to the PID or I-PD control algorithm. See the Tuning Active Damping and Acceleration Control topic for details on tuning these gains.

Tuning Wizard

In the Tuning Wizard, on the Confirm Model page, uncheck **Limit models to first order**. If the wizard then selects a second-order model, it will automatically set the Double Differential Gain. It will not set the Triple Differential Gain.

See Also

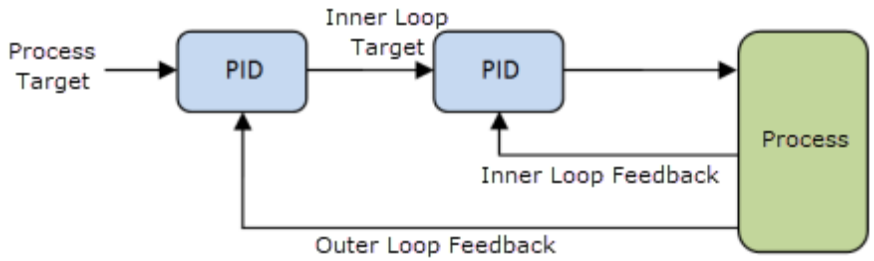
High-Order Control | Active Damping

3.5.11.5. Cascade Control

Cascade control is supported with any number of loops up to the number of axes supported by the RMC.

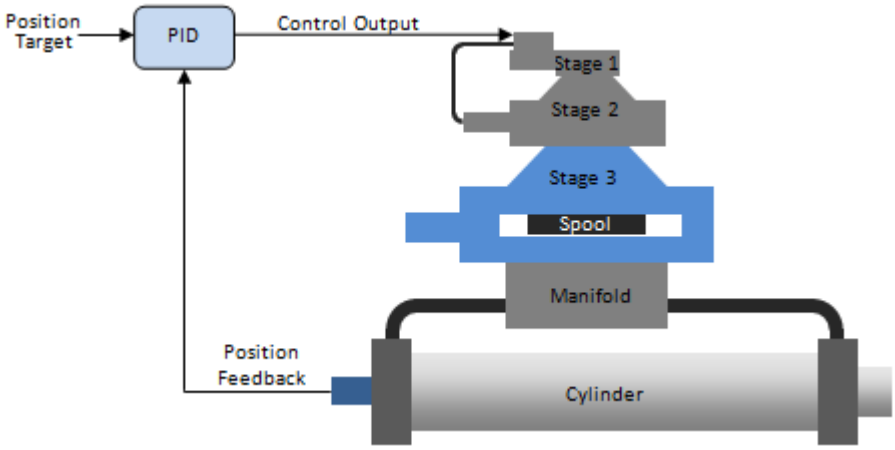
What is Cascade Control?

Cascade control is a control algorithm in which the output of one control loop provides the target for another loop, as shown in the diagram below.



The ultimate goal of the cascaded loops is to control the end process. Cascade control can provide precise control for certain difficult systems, for example systems in which some lag time exists.

To better understand cascade control, consider the following example in hydraulic motion control:

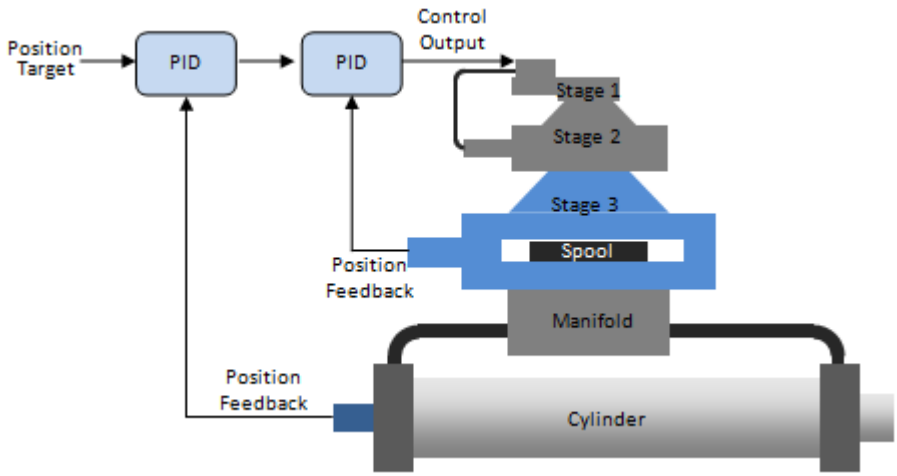


The figure above shows a very large hydraulic cylinder with a 3-stage proportional valve. The goal is to control the cylinder position.

The first two valve stages have internal electronic controls, and all we need to know about them is that the first two stages will produce a velocity on the third stage that is roughly proportional to the Control Output signal received by the valve. The third stage has no electronic controls.

A standard PID control loop can be applied to this system, as shown in the figure. However, this results in poor control. The valve spool responds very quickly compared to the cylinder. Therefore, when the Position Target of the system (the desired position of the cylinder) changes, the valve spool will immediately move over its full length of travel, while the cylinder has barely begun moving, and when the actual position eventually catches up to the target, the spool will immediately move back the full length, and in that way, the system can begin to oscillate and will result in poor control.

A better method is to apply an additional inner PID loop to the valve in a cascaded configuration, as shown in the figure below. This requires spool position feedback from the third stage of the valve. The valve will then provide a much more linear flow in relation to the Control Output, which will make the outer position PID much easier to tune and will provide better control.



Cascade Control Advantages

- Allows inner loop to handle non-linear valve and other final control element problems.
- Allows operator to directly control inner loop during certain modes of operation (such as startup).
- Allows controller to respond quickly to the faster inner loop.

Cascade Control Disadvantages

- Cost of measurement of the secondary variable (assuming it is not measured for other reasons).
- Additional complexity.

Requirements for Cascade Control:

- Inner loop system dynamics must be significantly faster (such as four times faster) than the outer loop system dynamics. If the inner loop is not faster than the outer loop, then the cascade will not offer any significant improvement in the system control.
- Inner loop must have influence over the outer loop.
- Inner loop must be measured and controllable.

Using Cascaded Loops in the RMC

Setting up Cascaded Loops

Setting up cascade control in the RMC involves the following steps:

1. **Define Each Loop As an Axis**
   Each loop of the cascade control must be defined as an axis.
   a. For the inner loop, define a standard Control axis.
   b. For the outer loop, define a Cascading Outer Loop axis. This axis type provides a virtual Control Output, but does not use a physical Control Output. If you have more than two loops, all loops except the inner loop should be Cascading Outer Loop axes.

2. **Set up and Tune Inner Loop Axis**
   Set up the inner loop axis as normal, including setting the Scale/Offset and tuning.

3. **Tie Output of Outer Loop to Input of Inner Loop**
   Use gearing to tie the output of the outer loop to the input of the inner loop. If the inner loop is position, use the Gear Absolute (25) command. If the inner loop is pressure or force, use the Gear Absolute (Prs/Frc) (59) command. Set the Gear Absolute command parameters as listed below. You will also need to use the Transition Rate (56) or Transition Rate (Prs/Frc) (64) command together with the Gear Absolute commands.
   a. The **Master Register** command parameter should be set to the PFID Output of the outer loop axis.

b. The **Master Point A** and **Master Point B** command parameters should be set to -100 and 100, respectively. This is because the PFID output ranges from -100% to +100%.
c. The **Slave Point A** and **Slave Point B** command parameters should be set to the minimum and maximum values that you want the inner loop axis to control to. For example, if the inner loop is a valve spool that you want to allow to travel from 0 to 0.5 inches, set the **Slave Point A** and **Slave Point B** command parameters to 0 and 0.5, respectively.
d. Set the **Endpoint Behavior** command parameter to **Truncate**.

If the inner loop is velocity, use the Gear Vel (Clutch by Time) (31) command, with the **Master Register** as described in item **a** above. The **Denominator** should be set to 200, and the **Numerator** should be set to the velocity range of the inner axis.

4. **Set up Outer Loop Axis**
Set up the outer loop axis, including scale/offset and tuning. During closed loop control of the outer loop (including during tuning), make sure that the *inner* loop axis remains in closed-loop control and geared to the PDIF output of the outer loop axis. If the inner loop halts, you will need to put in closed loop control again and restart the Gear Absolute. You should make sure the AutoStops and tolerances for the inner loop are set to avoid unnecessary halts.

Using Cascade Control

Once both loops of the cascade control have been set up and tuned, the system can be controlled by sending motion commands to the outer loop axis. There are a few items to consider:

- **Control Order**
The inner loop must always be in closed-loop control when the outer loop is controlling the system. When manually controlling the inner loop, the outer loop should be in open loop control.
- **Starting Up the System**
Every time the system starts up, send the Transition Rate (56) and Gear Absolute (25) commands to the inner loop axis. Then you can begin controlling the outer loop axis.
- **Error Handling**
You need to consider error handling between the axes. For example, if the inner loop axis halts, the outer loop axis must halt. If the outer loop axis halt, the inner loop does not necessarily need to halt.

See Also

Control Modes Overview

3.6. Motion

3.6.1. Synchronizing Axes

In motion control, the term *synchronization* is a very general term. In the most general sense, the RMC does everything synchronously, that is, it does things at specific times and responds to events within a certain amount of time. This topic describes specific types of synchronization of axes and explains how to achieve them in the RMC.

Identical Motion (Synchronized Identical Positions)

In this type of synchronization, the motion of the axes are identical. For example, two cylinders on a press always move to the same positions at the same time, and must always be at the same position during the moves.

To achieve this type of synchronization, issue *identical* commands (such as the Move Absolute (20) command) to each axis simultaneously. To issue the commands simultaneously, create a user program and put the commands in the same step. Or, from a PLC issue identical commands to all axes in the same write.

Keeping the axes in sync:

To keep the axes from getting out of sync, do the following:

- **Put all Axes in the Same Halt Group**
  Put all the axes in the same Halt Group. If one axis in the halt group halts, then all axes in the group will halt.
- **Set the Position Error Tolerance**
  Set the Position Error Tolerance for each axis. This will cause the Following Error bit to turn on when the Actual Position comes too far away from the Target Position. When the error bit turns on, its Auto Stop setting determines what type of halt occurs.
  Make sure the Following Error Auto Stop is set identically for each axis. Also make sure the Open Loop Halt Ramp (or Closed Loop Halt Deceleration, depending on your halt type) parameter is identical for each axis.

With these parameters set correctly, when you simultaneously issue identical motion commands to each axis, the Target Position will be identical for each axis at all times. If the Position Error on one axis becomes too large, the following error bit will turn on, and all the axes will halt at the same time. Notice that the maximum allowable skew is twice the Position Error Tolerance.

Ratioed Motion (Synchronized Unequal Travel Distances)

In this type of synchronization, the motion of the axes is synchronized such that all the axes start and stop moving simultaneously, and at any point during the move, each axis has completed the same percentage distance (or ratio) of its move. For example, if three axes are at 0 inches, and are to move to 2, 4, and 6 inches, respectively, then at any point in the move, the axes' positions will be at a 1:2:3 ratio.

To achieve ratioed motion, use the Sync move commands:

- Sync Move Absolute (13)
- Sync Move Relative (14)

Gearing

In this case, the motion of one axis must move at a rate proportional to the position or speed of another axis or reference input. See the Gearing topic for more details.

Synchronizing RMCs

An RMC can synchronize all its axes. To synchronize more axes, multiple RMCs are needed. There are several ways to synchronize axes across RMCs.

- **Synchronize all axes to a single axis**
  This applies to applications such as gearing multiple axes to a single feed chain. A transducer provides the master positions, to which all the other axes are synchronized to. A quadrature encoder signal can be daisy-chained to multiple RMC quadrature inputs. Or, the Universal I/O Module can be used to daisy-chain SSI position data to multiple RMCs. An analog position signal can also be connected to multiple RMCs.
- **Start identical motion simultaneously**
  If the axes need only do identical motion, then the motion can be started simultaneously using a discrete output to trigger each RMC.
- **Via communications**
  If the axes need only do identical motion, and do not require very tight synchronization, then the motion can be started simultaneously via the communication channel. The communication channels do exhibit a certain amount of latency and jitter.

- **UI/O Inter-Module Communciation**
  The Universal I/O Module can exchange data between RMC150s each loop time. Therefore, this is suitable for tight synchronization of motion between controllers, such as gearing. See the Universal I/O Module for details.

See Also

Gearing

3.6.2. Using Rotary Motion

This topic describes and provides examples of how to use rotary motion with the RMC series motion controller. For a description of rotary axes, see the Axis Type: Rotary/Linear topic.

In general, to move rotary axes, use the same motion commands as you would for linear axes. When positions wrap on the axis, the RMC automatically accounts for it when calculating target profiles, Position Error, etc.

When Should Rotary be Used?

If your machine has a rotary encoder, it does not necessarily mean the axis should be defined as rotary. The *application* determines whether the axis should be defined as rotary.

Typically, rotary applications are applications where the machine (motor, belt, etc.) rotates through a position cycle and does not have endpoints of travel. That is, it is possible for it continuously travel in one direction without needing to move the opposite direction (it doesn't have to continuously travel in one direction, but it should be possible). Typically, single-turn or multi-turn absolute encoders or incremental encoders are used in rotary applications.

The Positive Travel Limit and Negative Travel Limit do not apply for rotary axes. If you need positive and negative travel limits and you are using a rotary encoder, you should set up the axis as linear.

Rotary Configuration Basics

For rotary axes, the counts per revolution must be a power of two, such as 1024, 8192, etc. This typically means the encoder counter per turn must be a power of two.

The Position Unwind parameter defines the position range the axis will work in. The Position Unwind parameter can be any positive non-zero value. It works together with the Count Unwind and Position Offset parameters to scale the counts to position units.

For more details, see the Rotary Scaling topic.

Example 1:
Assume you have a 10 ft (120 in) long belt and an encoder. You wish to scale the axis for inches. You would set the Position Unwind to 120. As the belt travels, the positions will go from 0 up to, but not including, 120 in. The Count Unwind should be set to the number of counts generated in 120 inches of travel.

Example 2:
Assume you have a motor with an encoder with 8000 counts per turn, and the encoder turns 4 times per machine revolution. You wish to scale the axis for degrees. You would set the Position Unwind to 360 and the Count Unwind to 32000 (8000 x 4). As the motor turns, the positions will go from 0 up to, but not including, 360 degrees.

Rotary Motion Command Behavior

The sections below describe how commands behave on rotary axes:

- Rotary Motion with Absolute Position Moves
- Rotary Motion with Relative Position Moves

- Rotary Motion with Velocity and Gear Moves
- Rotary Motion with Open Loop Commands

Rotary Motion with *Absolute* Position Moves

This section applies to the following commands:

- Move Absolute (20)
- Quick Move Absolute (15)
- Time Move Absolute (23)
- Sync Move Absolute (13)
- Advanced Time Move Absolute (26)
- Move Absolute (I-PD) (28)
- Transition Rate (56)

Each of the commands listed above has a **Direction** parameter with the following options for rotary axes:

- **Positive:**

  The axis will move to the Requested Position in the direction of increasing position units. If the Command Position is less than the current Target Position, the axis will wrap around to the Requested Position. If the Requested Position command parameter is outside the valid range of the axis, the Command Position will be set within the valid range using modular arithmetic such that the position will be the same location within the range. For example, if the valid range is 0-360 (not including 360), and the Requested Position is 800, the Command Position will be set to 80 (800 mod 360). Likewise, with a Requested Position of -100, the Command Position will be set to 260 (360 + (-100 mod 360)). The Command Position will never be more than 1 revolution from the current Target Position.

  Example:

  Consider a rotary axis with a Position Unwind of 360 and a current Target Position of 180. A move to 90 with the Positive Direction parameter will cause the axis to pass 0 (wrap), and end at 90 position units.

- **Negative:**

  The axis will move to the Requested Position in the direction of decreasing position units. If the Command Position is greater than the current Target Position, the axis will wrap around to the Requested Position. If the Requested Position command parameter is outside the valid range of the axis, the Command Position will be set within the valid range using modulo arithmetic such that the position will be the same location within the range. For example, if the valid range is 0-360 (not including 360), and the Requested Position is 800, the Command Position will be set to 80 (800 mod 360). Likewise, with a Requested Position of -100, the Command Position will be set to 260 (360 + (-100 mod 360)). The Command Position will never be more than 1 revolution from the current Target Position.

  Example:

  Consider a rotary axis with a Position Unwind of 360 and a current Target Position of 180. A move to 270 with the Negative Direction parameter will cause the axis to pass 0 and end at 90 position units.

- **Nearest:**

  The axis will move to the Requested Position in the direction that results in the shortest distance of travel. If the Requested Position command parameter is outside the valid range of the axis, the Command Position will be set within the valid range using modulo arithmetic such that the position will be the same location within the range. For example, if the valid range is 0-360 (not including 360), and the Requested Position is 800, the Command Position will be set to 80 (800 mod 360). Likewise, with a Requested Position of -100, the Command Position will be set to 260 (360 + (-100 mod 360)). The Command Position will never be more than 1 revolution from the current Target Position.

  Example:

Consider a rotary axis with a Position Unwind of 360 and a current Target Position of 180. A move to 270 with the Nearest Direction parameter will cause the axis to go directly to 270. Likewise, with a current Target Position of 45, a move to 270 with the Nearest Direction parameter will cause the axis to pass 0 and go to 270.

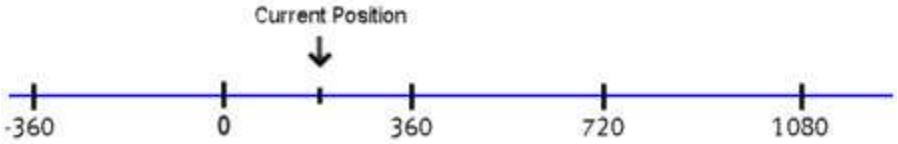
- **Absolute:**

  This option is very useful for moving the axis through multiple revolutions. With the Absolute option, the Move command mimics a linear axis. The Requested Position command parameter can be any value; it does not need to within the valid range of the axis. When the command is issued, the value in the Requested Position command parameter is treated as a position on a linear axis; the axis begins moving toward the position as if on a linear scale. If the position is outside of the valid position range, the axis rotates through the number of revolutions required to reach the position. Each time the Target Position wraps during the move, the Position Unwind value is subtracted from the Command Position until the Command Position is within the valid position range.

  Relative point-to-point moves can also be used to move the axis through multiple rotations. See the Rotary Motion with *Relative* Point-to-Point Moves section below.

  Example:

  For example, consider a rotary axis with a Position Unwind of 360 where the Actual Position is at 180. When a Move command is issued with the Absolute Direction parameter, the Move command considers the axis as if it were linear, as shown below. Each labeled mark on the axis represents one revolution. Therefore, moving to 720 would move the axis 1.5 revolutions. Notice that when it reaches the final position, the Command Position will be 0, not 720, because it subtracted 360 each time it wrapped.



- **Current:**

  This option will move to the Requested Position in the current direction. The current direction is determined by current velocity of the axis. If the axis is already in closed-loop control, the Target Velocity is used. If the axis was in open-loop control when the command was issued, then the Actual Velocity will be used. If the axis's current velocity is zero when the command is issued, it will behave as the Nearest direction parameter option.

  If the Requested Position command parameter is outside the valid range of the axis, the Command Position will be set within the valid range using modulo arithmetic such that the position will be the same location within the range. For example, if the valid range is 0-360 (not including 360), and the Requested Position is 800, the Command Position will be set to 80 (800 mod 360). Likewise, with a Requested Position of -100, the Command Position will be set to 260 (360 + (-100 mod 360)). The Command Position will never be more than 1 revolution from the current Target Position.

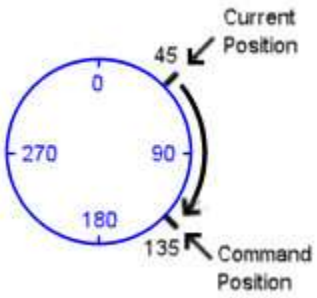
Rotary Absolute Motion Examples

Example 1: Basic Rotary Move

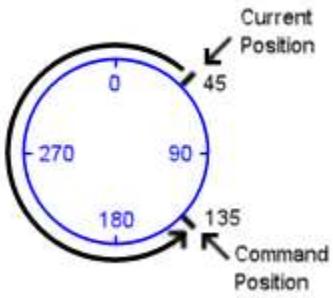
Consider a rotary axis with a single-turn encoder with a **Position Unwind** value of 360 and a Position Offset of 0. Therefore, the position range extends from 0 up to, but not including, 360. Assume the current position of the axis is 45. You issue a **Move Absolute** command with a **Position** command parameter of **135**. The following illustrations explain how the axis will behave for each possible **Direction** parameter:

Positive	**Negative**

The axis will move in the positive direction to the Command Position, as shown below:

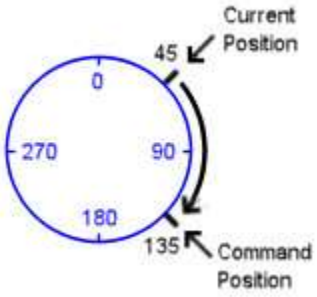


The axis will move in the negative direction to the Command Position, as shown below:



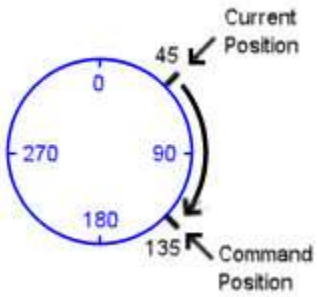
Nearest

The axis will move in the direction that gives the shortest path to the Command Position, as shown below:



Absolute

The axis will move as if it were a linear axis. Therefore, the axis will move to the Command Position in the direction of increasing position units, as shown below:

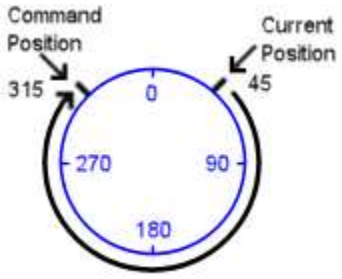


Example 2: Basic Rotary Move

Consider a rotary axis with a single-turn encoder with a **Position Unwind** value of 360 and a Position Offset of 0. Therefore, the position range extends from 0 up to, but not including, 360. The current position of the axis is 45. You issue a **Move Absolute** command with a **Position** command parameter of **315**. The following illustrations explain how the axis will behave for each possible **Direction** parameter:

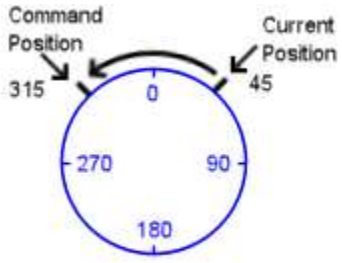
Positive

The axis will move in the positive direction to the Command Position, as shown below:

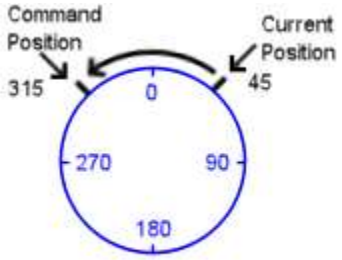


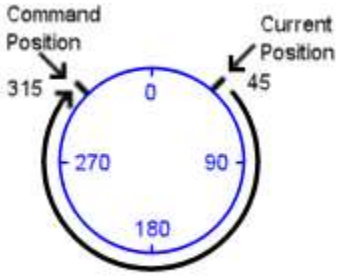
Negative

The axis will move in the negative direction to the Command Position, as shown below:



Nearest	Absolute
The axis will move in the direction that gives the shortest path to the Command Position, as shown below:	The axis will move as if it were a linear axis. Therefore, the axis will move to the Command Position in the direction of increasing position units, as shown below:

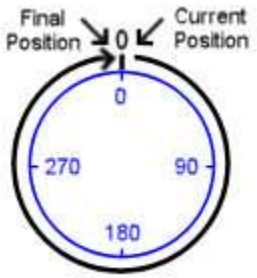


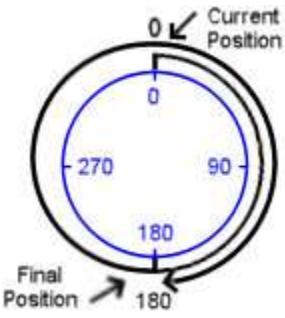


Example 3: Multiple Rotations

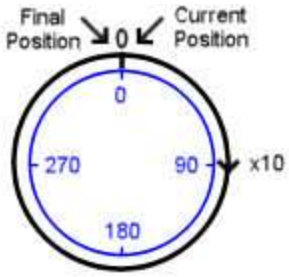
This example illustrates how to use the Absolute option in the Direction command parameter to move a rotary axis through multiple revolutions.
Consider a rotary axis with a single-turn encoder with a **Position Unwind** value of 360 and a Position Offset of 0. Therefore, the position range extends from 0 up to, but not including, 360. The current position of the axis is 0. You issue a **Move Absolute** command with the **Absolute** option in the **Direction** command parameter. The following illustrations explain how the axis will behave for various values of the **Position** command parameter:

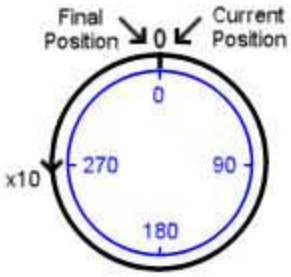
Position command parameter = 360	Position command parameter = 540
The axis will move 1 revolution in the positive direction. The final position will be 0.	The axis will move 1.5 revolutions in the positive direction. The final position will be 180.





Position command parameter = 3600	Position command parameter = -3600
The axis will move 10 revolutions in the positive direction. The final position will be 0.	The axis will move 10 revolutions in the negative direction. The final position will be 0.





Rotary Motion with *Relative* Position Moves

This section applies to the following commands:

- Move Relative (21)
- Quick Move Relative (16)
- Time Move Relative (24)
- Sync Move Relative (14)
- Advanced Time Move Relative (27)
- Move Relative (I-PD) (29)

These commands are useful for moving the axis through multiple rotations.

Each of the above commands has a command parameter called **Displacement**. Each command works by moving the axis by the requested **Displacement** relative to the selected value of Command Position, Target Position, or Actual Position. If the Command or Target Position is selected, and the axis is currently in a mode that does not use that Position (such as open loop) it will be relative to the current Actual Position. The sign of the Displacement specifies the direction of movement. The Displacement can be greater then the Position Unwind, which makes these commands useful for moving the axis through multiple rotations. The Command Position is initially set to the current selected relative-to position plus the requested Displacement. Each time the position wraps, the Position Unwind is subtracted from the Command Position until it reaches the final position.

For example, consider an axis with a Position Unwind of 1.0. Therefore, the position range extends from 0 up to, but not including, 1.0. If the current Command Position is at 0.0 and a Move Relative command is issued with a Displacement of 100 relative to the Command Position, the axis will rotate 100 revolutions and end up at 0.0.

Rotary Motion with Velocity and Gear Moves

This section applies to the following commands:

- Move Velocity (37)
- Move Velocity (I-PD) (38)
- Gear Pos (Clutch by Time) (30)
- Gear Vel (Clutch by Time) (31)

These commands work like on a linear axis, except that the positions will wrap as usual for a rotary axis.

Rotary Motion with Open Loop Commands

This section applies to the following commands:

- Direct Output (9)
- Open Loop Rate (10)
- Open Loop Absolute (11)
- Open Loop Relative (12)

The Direct Output (9) and Open Loop Rate (10) commands work like on a linear axis, except that the positions will wrap as usual for a rotary axis.

The Open Loop Absolute (11) and Open Loop Relative (12) commands ramp the Control Output from it's current value to the requested Output. This ramping is linear based on distance (not time) from the current Actual Position to the requested Position or requested Displacement command parameters.

For both these commands, the ramping range is calculated in the same manner as the Command Position is calculated for the point-to-point moves described above; for the Open Loop Absolute command, the range of the ramping is determined by the Direction parameter, and for the Open Loop Relative command, the range of the ramping is determined by the sign of the requested Displacement.

3.6.3. Velocity Control

The RMC fully supports velocity control. Velocity control can be achieved on either a position axis or a velocity axis.

On a Position Axis:

On a position axis (an axis that has a position feedback, such as an encoder), the RMC can control velocity in several ways:

- **With Position Moves**
  The most basic velocity control on a position axis is with standard position moves in Position PID mode (the standard control mode). For example, when issuing a Move Absolute (20) command, the speed must be specified in the command. The axis will move at the specified speed until it reaches the requested position.
- **Continuous Position Control**
  Velocity can also be controlled on a position axis with the Move Velocity (37) command in Position PID mode. In this mode, the Move Velocity command does not actually generate a Target Velocity. Instead, it generates a moving Target Position. Therefore, if the axis falls behind, it will not try to catch up to the Target Velocity; it will actually attempt to catch up to the Target Position. This may cause the axis to move considerably faster than the Target Velocity while it catches up.
  The Move Velocity command works well on axes where position is important.
- **True Velocity Control**
  True velocity control can be achieved on a position axis with the Move Velocity (37) command in Velocity PID mode. In this mode, the Move Velocity command generates a Target Velocity. A Target Position is not calculated; it is set to the same as the Actual Position. Velocity I-PD mode can also be used for true velocity control.

Velocity control on a position axis is often done with a rotary encoder. See the Using Rotary Motion topic for details on using rotary motion.

On a Velocity Axis:

On a velocity axis (an axis that has a velocity feedback, such as a tachometer), the RMC can control true velocity. Notice that in velocity control on a velocity axis, it is impossible to achieve exactly zero speed. This is because velocity transducers report the speed in voltage, which can never be exactly zero due to its analog nature and noise.

The standard command for velocity control is the Move Velocity (37) command. The Velocity PID mode is typically used for velocity control on a velocity axis. See the Velocity PID topic for more details, such as commands, etc.

For advanced applications, the Velocity I-PD control mode may be used.

See Also

Using Rotary Motion | Defining Axes | Control Features Overview

3.6.4. Gearing

Gearing is used when one axis (the slave axis) must move incrementally and proportionately to a register (the gear master), which is typically the position or velocity of another axis. The RMC has several commands to cover a wide range of simple and advanced gearing applications. This topic describes the basics of gearing and gives an overview of the gearing commands.

If you need to gear using a non-linear profile, see the Curves Overview topic.

In general, to gear an axis, issue a gearing command to the axis to be geared (the slave axis). The axis will remain geared until another command is issued to the axis. A register does not need to do anything to become the master of a gearing relationship. The slave axis will select its gear master.

The RMC provides both relative gearing, which defines only the rate at which the slave moves based on the rate of the master, and absolute gearing, which defines exactly the position of the slave based on the value of the master.

Gearing Commands

The RMC offers the following gearing commands:

Absolute Gearing Commands

Absolute gearing defines exactly the position of the slave based on the value of the master. The absolute gearing commands work very well for making an axis follow a reference input (half axis).

Gear Absolute (25)	Sets up an absolute linear gearing relationship and will make the axis follow that relationship.
Gear Absolute (Prs/Frc) (59)	Sets up an absolute linear gearing relationship for pressure/force and will make the axis follow that relationship.

Relative Gearing Commands

Relative gearing defines the rate at which the slave moves based on the rate of the master.

Gear Pos (Clutch by Rate) (39)	Gears the position of an axis to a master. The clutching is done such that the slave axis ramps its target velocity using the acceleration and jerk parameters until it reaches the synchronized gear ratio.
Gear Pos (Clutch by Time) (30)	Gears the position of an axis to a master. The clutching is done such that the gear ratio ramps from the current ratio to the specified ratio in the specified time.
Gear Pos (Clutch by Distance) (32)	Gears the position of an axis to a master. The clutching is done such that the master and slave synchronize exactly at the requested positions. Typically used in flying-cutoff type applications.
Gear Vel (Clutch by Time) (31)	Gears the velocity of an axis to a register. The clutching is done such that the gear ratio ramps from the current ratio to the specified ratio in the specified time.

Advanced Gear Move (33)	For very advanced gearing applications. It is intended to be used in user programs along with mathematical calculations.

Gearing with Motion Limits Commands

The Track commands provide gearing, with limits on the position, velocity, acceleration and jerk. These commands are useful for smoothly tracking a signal containing noise or step-jumps, or for gearing to another position while not exceeding specified motion limits.

Track Position (57)	Continuously tracks the specified master register. The axis position is limited by the positive and negative travel limits, and the specified velocity, acceleration, and jerk limits.
Track Position (I-PD) (58)	Continuously tracks the specified master register. The axis will be controlled using the I-PD algorithm. The axis position is limited by the positive and negative travel limits and the specified velocity.

Gear Modifying Commands

These commands modify an existing gearing relationship.

Geared Slave Offset (35)	Superimposes a move of the requested distance onto the currently-geared axis while the master moves the specified distance.
Advanced Gear Move (33)	For very advanced gearing applications. It is intended to be used in user programs along with mathematical calculations.

Gear Ratio

The gear **ratio** specifies the gearing of the axis to its master. To make gearing infinitely accurate in more applications, most of the RMC gear commands use a **Numerator** and **Denominator** to specify the ratio. For example, if a user has a rotary application and needs a gear ratio of 1:3, it cannot be represented with a single decimal number. However, the Numerator and Denominator will represent this accurately and the system can gear for any number of revolutions without losing the position.

The Numerator is the distance the slave axis travels as the master travels the distance specified by the Denominator. This relationship is shown below in equivalent equations:

$$\text{Gear Ratio} = \frac{\text{Numerator}}{\text{Denominator}} = \frac{\text{Change in slave axis position}}{\text{Change in master position}} = \frac{\text{\#Master gear teeth}}{\text{\#Slave gear teeth}} = \frac{\text{Slave speed}}{\text{Master speed}}$$

Gear Ratio of Zero (0)

A gear ratio of zero will cause the slave axis to stop. This can be very useful if the axis is already geared. By specifying a gear ratio of zero and clutching by distance, the slave can be instructed to stop when the master reaches a certain position.

High Gear Ratios

High gear ratios can cause unstable control. A high gear ratio will cause the slave axis to move a large distance while the master moves a small distance. Any noise in the master will be amplified in the motion of the slave axis. In addition, the feedback increments will be amplified. Try to avoid high gear ratios.

If a high gear ratio cannot be avoided, try to use the lowest noise and highest resolution feedback possible on the master axis. This will minimize the amplification of the feedback increments and noise. If the master axis is a control axis, make sure the motion is smooth and minimize quick changes in velocity. If the master axis is a reference axis, consider adding filtering to it to reduce noise.

Gear Absolute

The Gear Absolute commands do not use a numerator and denominator. Instead, the gearing relationship is defined with Master Point A, Master Point B, Slave Point A, Slave Point B. The ratio can be calculated as follows. See the Gear Absolute (25) command for more details.

Gear Ratio = (Slave Point B - Slave Point A) / (Master Point B - Master Point A)

Clutching

Clutching is used for relative gearing commands. Clutching is the transition of the motion of the slave axis at the time the gearing command is issued to the final specified ratio. Most often, this cannot be done instantaneously.

Example 1

Consider a basic gearing application where Axis 1 (the slave axis) is to gear to Axis 0 (the gear master) with a 1:1 ratio. Before gearing, Axis 0 is moving at 10 in/sec and Axis 1 is stopped. If the gearing were to start instantaneously, then Axis 1 must go from 0 in/sec to 10 in/sec instantaneously, which is impossible.

The RMC supports several methods of clutching, making it very flexible for use in many gearing applications.

Clutching Methods

- **By Rate**

  Clutching by rate specifies the rate at which the slave axis' velocity changes in order to reach the requested gear ratio. The rate is specified with an Acceleration and a Jerk parameter. In Example 1 above, if the clutching Acceleration is 20 and the Jerk is 1000, then Axis 1 will change from 0 in/sec to 10 in/sec at that rate. Once it reaches 10 in/sec, it will lock in at the 1:1 ratio.

  For more details, see the Gear Pos (Clutch by Rate) (39) command, which starts gearing with clutching by rate.

- **By Time**

  Clutching by time specifies the amount of time it should take to reach the requested gear ratio. In Example 1 above, if the clutching time is 0.5 seconds, then Axis 1 will go from 0 in/sec to 10 in/sec in 0.5 seconds. It will then be geared at 1:1.

  For more details, see the Gear Pos (Clutch by Time) (30) command, which starts gearing with clutching by time.

- **By Distance**

  Clutching by distance specifies the distance in which the requested gear ratio should be reached. This allows the user to specify the exact positions at which the axes should be locked in at the specified gear ratio. This is especially useful for flying-cutoff type applications.

  Consider a gearing application where Axis 1 (the slave axis) is to gear to Axis 0 (the gear master) with a 1:1 ratio. Before gearing, Axis 0 is moving at 10 in/sec and Axis 1 is stopped. With clutching by distance, the user can specify that the axes should start clutching when the master reaches 5 in. and reach the final gear ratio when the master reaches 10 inches. The user also wants the slave to be at 2 inches at this point.

  Therefore, when the master reaches 5 inches, the slave will start moving. When the master reaches 10 inches, the slave will be at 2 inches, and will be geared 1:1 with the master.

  For more details, see the Gear Pos (Clutch by Distance) (32) command, which starts gearing with clutching by distance. This command is used in flying-cutoff type applications.

Transitions

Transitions are used for absolute gearing commands. When an absolute gearing command is issued to an axis, the axis must already be on the gearing relationship, or a Transition command must previously have been issued to the axis to define how the axis should move from it's current position onto the gearing relationship.

See the Transition Rate (56) command or more details.

Possible Gear Masters

The gear master can be any register in the RMC. A register does not need to do anything to be a gear master. Most registers in the RMC are not useful as a gear master. Some practical gear masters are described below:

- **Target Position or Actual Position of a control axis:**
  In this case, the axis is geared to the Target or Actual Position (or pressure or force) of another control axis. For example, one motor may need to gear to another motor, or one cylinder may have to move at the same rate as another cylinder. Typically, if you are gearing to a control axis, gearing to the Target Position provides smoother motion than gearing to an Actual Position because Actual Positions, as an actual feedback signal, tend to be less smooth due to transducer noise and quantization.
- **Actual Position of a reference input (half axis):**
  In this case, the axis is geared to the position (or pressure or force) of a reference input, such as from a joystick, an analog output from a PLC, or a belt position sensor. The Gear Absolute (25) and Gear Absolute (Prs/Frc) (59) commands work very well for this type of application.
- **A Variable:**
  Using the Expressions command in User Programs, you can create a Variable that varies in time. By gearing to the variable, you can move the axis in that profile. The Gear Absolute (25) and Gear Absolute (Prs/Frc) (59) commands work very well for this type of application.

The RMC will process the slave after the master, which reduces latency in the slave axis—as long as there are no g"gear chains". For example, if axis A is geared to axis B, which is, in turn, geared to axis C, there is no guarantee that C will be processed before B which will be processed before A. However, if A and B are both geared to directly C, then C will be processed before both A and B.

Noisy Masters

Gear masters that are not very smooth, such as Actual Positions or reference inputs, may cause chatter in the slave axis. The velocity, acceleration and jerk of the slave are calculated from the master. Any noise on the master will be exaggerated in these calculations, especially the acceleration and jerk. To reduce chatter, you may need to set the Acceleration Feed Forward and Jerk Feed Forward to 0 on the slave axis.

If the gear master is a reference input, it can be filtered to make it smoother.

Using a Virtual Axis as a Gearing Master

A virtual axis can be used as a gearing master axis. It is sometimes desirable to gear to a virtual axis rather than executing the motion as a function of time. All the axes geared to the virtual axis can be sped up or slowed down by speeding up or slowing down the virtual axis. The virtual axis can even be moved backwards causing the geared axes to back up too. This cannot be done using time-based commands.

When using a virtual axis as a master as described above, it is often useful to set it up as a rotary axis because it will never need to be reset. When used as a master, the virtual axis is typically commanded to move with a Move Velocity (37) command. Moving it at 1 unit/sec as the standard velocity makes gearing ratio calculations very easy. The acceleration and deceleration provide a smooth start and stop for the geared axis.

Halt Groups

For safety, both the master and slave axis should be included in the same Halt Group. If one axis in a Halt Group halts due to an error, all the other axes in that halt group will also halt.

Target Generator Components

The behavior of the components of the gearing target, including the Target Velocity, Target Acceleration, and Target Jerk, depend on the type of register used as the master:

- **Target Position**
  When gearing to a Target Position, the slave Target Velocity will be the ratioed master Target

Velocity, the slave Target Acceleration will be the ratioed master Target Acceleration, and the slave Target Jerk will be the ratioed master Target Jerk.

- **Actual Position**
  When gearing to an Actual Position, the slave Target Velocity will be the ratioed master Actual Velocity (filtering applies), the slave Target acceleration will be the ratioed master Actual Acceleration (filtering applies), and the slave Target Jerk will be zero (0).
- **Actual Pressure/Force**
  When gearing to the Actual Pressure/Force, the slave Target Velocity will be the ratioed master Actual Pressure/Force Rate, the slave Target Acceleration will be the ratioed simple derivative of the velocity, and the slave Target Jerk will be zero (0).
- **Other Registers**
  When gearing to any other register, the slave Target Velocity will be the ratioed simple derivative, the slave Target Acceleration will be the ratioed simple derivative of the velocity, and the slave Target Jerk will be zero (0).

See Also

Synchronizing Axes

3.6.5. Simulating Motion

The RMC can simulate motion on certain axis types. The axis simulates a physical system, including the feedback, and may be used to:

- Learn how to move and tune an axis.
- Develop and run user programs on your RMC before the machine is ready for motion.
- Approximate a given real hydraulic system. This require advanced knowledge of system models. Set the Simulate parameters to values that correspond to your real system and then tune the simulated axis. This can provide an indication of the tuning values required for the real system.

Supported Axis Types	
Position Control	✓
Position-Pressure Control	✓
Position-Force Control	✓
Velocity Control	✗
Pressure-Only Control	✗
Force-Only Control	✗
Reference Axes	✗
Virtual Axes	*

*Simulate mode does not apply to virtual axes. They have only a virtual target profile that can be commanded and have no physical inputs or outputs to simulate.

Notes:

- During simulate mode, the axis' physical output is 0 V.
- Simulate mode requires a physical RMC, and is not possible only within RMCTools.

- If any axis is in simulate mode, the status of the Simulate Mode parameter is displayed in the Axis Status Registers Pane to clearly indicate that to the operator.
- The simulated system will be limited to the limits of the simulated transducer. For example, if the Counts for a simulated MDT-feedback axis tries to go under 0 counts, or above 1048575 (16#FFFFF), a Transducer Overflow error will occur. If a voltage feedback axis tries to go below -10 Volts or above +10 V, a Transducer Overflow error will occur. To recover from a transducer overflow error, use the Direct Output (9) command to move away from the limit.

Setting up Simulate Mode

Use the Simulator Wizard to quickly set up an axis in simulate mode. It will set the simulator parameters, travel limits, and tuning gains so you can immediately move the axis in simulate mode. Axis parameters such as Linear/Rotary, SSI Data Bits, and Analog Input Type (±5V, ±10V, 4-20mA) should be set before using the wizard. Simulate mode requires a physical RMC, and is not possible only within RMCTools.

1. **Enter Desired Position Range**
   Enter the desired range of position travel. For rotary axes, choose the number of position units per rotation.
2. **Enter Maximum Velocity**
   Enter the desired maximum velocity. This is the velocity at which the axis will move with 100% of Control Output.
3. **Enter Maximum Acceleration**
   Enter the desired maximum acceleration. This is not necessarily a true limit of the acceleration, but helps determine the response of the simulator and the tuning gains. For best results, set this value significantly higher than the acceleration rates you intend to use on the axis. This value is typically at least an order of magnitude (10x) greater than the maximum velocity.
4. **Enter Pressure or Force Information**
   If the axis is position-pressure or position-force, enter the maximum force of the simulator, and the desired range of the force area at the ends of the travel.

   The tuning of the pressure or force is affected by the position settings, especially the **Maximum Acceleration**. If the pressure or force tuning is poor, or pressure or force following errors occur, run the Simulator Wizard again and increase the **Maximum Acceleration**.
5. **Review Parameters**
   Click **Next** and review the axis parameter settings in the **Proposed** column. Any parameters with blue text can be changed. When you are satisfied with the values, click **Finish**.
6. **Download Axis Parameters**

   In the Axis Tools window, click the download button  to apply the parameter changes to the controller.

After setting these parameters, make sure to enable the axis. This can be done by sending the Enable Controller (7) to the RMC. To do a basic move on the axis, send the Move Absolute (20) command.

Note: The simulator wizard will set the simulator to 2nd order.

Simulating a Real System

The simulator can be configured to approximate your actual system. The simulator parameters are in the **Axis Tools**, in the **Axis Parameters**, on the **All** tab, in the **Simulator** section. You will need to set the parameters listed in the table below. Use the guidelines below to calculate the Simulate parameters to approximate your real system:

Parameter	To calculate this parameter:

System Gain	The System Gain units are pu/s/V or pu/s/%, which is the speed the system moves for 1 V c of control output. The RMC200 has individual Positive System Gain and Negative System Gain parameters to simulate different behavior in each direction of motion.
Simulator Order	The simulator can be set to 1st or 2nd order. A 1st order system is defined by the System Gain and Time Constant. A 2nd order system is defined by the System Gain, Natural Frequency, and Damping Factor.
Time Constant	For a 1st order system. To calculate the time constant, determine the time it should take the velocity to reach 95% of the steady state, and divide by 3.
Natural Frequency	For a 2nd order system. The Natural Frequency for a hydraulic system is normally between 1 30. Use the following formula to calculate the natural frequency for hydraulic system: ω = sqrt[(4*200000 *A^2) /(mass * volume)] where ω = Natural Frequency (Hz) A = area of the piston (in^2) mass = the mass moved by the system (lb) volume = the volume of trapped oil in the cylinder (in^3)
Damping Factor	For a 2nd order system. The damping factor is a unitless number. Hydraulic systems typically from 0.3 to 0.8. If the load has a lot of friction, this value will become larger. A lower value m the system more difficult to control.
Positive Physical Limit Negative Physical Limit	Specifies the maximum and minimum positions the simulator can move to. Notice that the Maximum Compression distance goes beyond these limits. Setting the limits both to zero me there is no limit. However, the RMC will simulate the feedback limit, whether it be counts, vo current.
Output Deadband	Simulates overlapped spools on hydraulic valves.
Output Null	Simulates a null offset on hydraulic valves.
Weight	Used to calculate the forces during motion.
Maximum Force	The max force of the system. Used when hitting the ends and in calculating the max speed.
Maximum Compression	The maximum compression at the positive and negative physical limits. This simulates a spri provide pressure or force feedback.

Once you have calculated the tuning parameters, you must set the positive and negative travel limits and tune the axis according to the Tuning Position topic in order to move it.

Notice that on axes with analog feedback, if you move the simulated axis too far, it will turn on the Transducer Overflow bit.

Troubleshooting

Applying Simulator Parameters

For best results, always change the Simulator axis parameters only from Axis Tools.

RMC75 and RMC150: If the axis is in Simulate mode, and a simulator parameter is changed, the simulate model will be suspended until the last simulate mode parameter register (Max Compression [Fx:127]) has been updated. At this point, the new simulator model will be calculated and the simulator model will be activated. When the simulator model is suspended, the Actual Position will not move.

RMC200: The simulator parameters can be updated on the fly.

Simulator Unresponsive

In certain cases, the Actual Position may become unresponsive. One such case is if the Actual Position is moved such that the Counts become zero on a linear MDT, SSI, or Resolver axis.

If this occurs, you can reset the simulator by exiting and entering simulate mode:

1. In the Axis Tools, uncheck the **Simulate Mode** check box and download the axis parameters.
2. Check the **Simulate Mode** check box and download the axis parameters. The Actual Position will be set to the midpoint of the Positive and Negative Physical Limits.

Invalid Model

When the simulator parameters are changed, the RMC calculates the new simulator model for the axis. This can take several control loops to complete. The axis uses the model to simulate the motion. Due to the digital nature of the RMC, not all models are valid. It is possible for the model generation to fail due to more subtle interactions between the model parameters. Specifically, it is possible to have 2nd order models rejected if the Natural Frequency is near its maximum, especially when the Damping Factor is high.

If the simulator does not appear to be working, you may have an invalid model. Make sure the simulator parameters are within their ranges. The valid ranges are given in the help topics for each simulator parameter.

See Also

Simulator Wizard | Simulate Mode | System Gain | Natural Frequency | Damping Factor | Positive Physical Limit | Negative Physical Limit | Output Deadband | Output Null | Weight | Maximum Force | Maximum Compression

3.6.6. Step Jumps

For closed-loop motion control, the RMC target generator typically generates a motion profile that ramps the position to the requested position in a controlled manner. This provides smooth and precise motion. However, some users may prefer that the Target Position jumps immediately to the requested position. This is called a Step Jump.

The RMC can generate a step-jump command with the Time Move Absolute (23) and Time Move Relative (24) commands. The Move Time parameter must be set to 0. See the respective commands for details.

The Move Absolute (I-PD) (28), Move Relative (I-PD) (29), and Move Velocity (I-PD) (38) commands also generate step jumps.

See Also

Time Move Absolute (23) | Time Move Relative (24)

by Delta Computer Systems, Inc.s

3.6.7. Curves, Cams, and Splines

3.6.7.1. Curves Overview

Cams, splines, and custom profiles for position, pressure, or force are handled by **curves**. The RMC creates curves by interpolating data points provided by the user or host controller. The RMC provides many options for creating and following curves to satisfy nearly any application, including curve sawing, veneer lathes, blow-molding, animation, electronic camming, cyclic testing profiles, valve linearization and more.

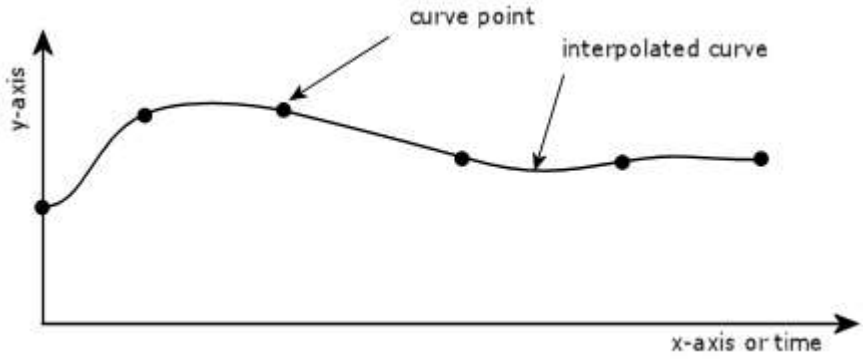
The RMC curves can be used for both time-based motion, where the curve defines the position of the axis at certain times, and master-based motion, where the curve defines the position of the axis based on a master, such as the position of another axis.

For time-based sinusoidal motion, use the Sine Start (72) command instead of curves.

Curves, together with the curve interpolations functions, can be used for other purposes, such as Gain Scheduling.

Curve Basics

A typical curve profile has many points which the RMC interpolates to create a smooth curve that goes through all the points. Many options are available, such as constant, linear or cubic interpolation, zero-velocity endpoints, cyclic curves, overshoot protection, and more. Curve x-values and y-values can be scaled and offset, and master or relative alignments can be selected.



Creating Curves

Curve Tool
Creating and viewing curves is easy in the graphical Curve Tool. This method is excellent for applications that require only pre-defined curves. For details, see the Curve Tool.

Curve Add Command
This method is suitable for creating curves via a host controller, such as a PLC or PC, or even via user programs. For details, see Creating Curves Using the Curve Add Command, and Example: Create Curve Using the Curve Add Command.

Managing Curves

Use the Curve Tool to manage curves whether they were created in the Curve Tool or using the Curve Add command. See the Managing Curves in the Curve Tool and Curve Tool Overview topics for more details.

How to Run a Curve Based on Time

Note: These methods refer to position. Curves apply to pressure and force as well, with the Curve Start (Prs/Frc) (87) and Curve Start Advanced (Prs/Frc) (89)commands.

Use this method to make your axis follow a curve based on time. That is, the x-values of the curve are time.

Start with Axis at First Position of Curve

This method requires that the axis be at the first position of the curve before starting the curve.

1. Move the axis to the first point of the curve (Y_0) and make sure it is in position in closed-loop control.
2. Send the Curve Start (86) command. The axis will move as defined by the curve profile.
3. For advanced features such as scaling and offsetting the curve, or choosing relative or absolute alignments use the Curve Start Advanced (88) command.

Start with Axis NOT at Starting Position

This method will start the curve even if the axis is not at the starting point.

1. Send the Transition Rate (56) command to specify how the axis should move to get to the curve. This command will not make the axis move immediately, it simply defines how the axis should get to the curve if the axis is not at the starting point when the Curve Start command is sent.
2. Send the Curve Start (86) command. The curve will start running, and the axis will move toward the curve as defined by the curve profile. Once it locks onto the curve, it will follow the curve profile exactly.
3. For advanced features such as scaling and offsetting the curve, or choosing absolute or relative alignments use the Curve Start Advanced (88) command.

Run a Curve Based on a Master Position

Use this method to make your axis follow a curve based on the position of another axis. That is, the x-values of the curve are positions of the master axis.

1. Send the Transition command to specify how the axis should move to get to the curve. This command will not make the axis move immediately, it simply defines how the axis should get to the curve if the axis is not on the curve when the Curve Start command is sent.
2. Move your master axis and the curve axis to the desired starting positions. Neither the master axis nor the curve axis need to be at the start of the curve.
3. Choose the Curve Start Advanced (88) command. Set the **Options** command parameter to **6**, which indicates the following:
    - Absolute Curve Alignment (+0)
    - Absolute Master Alignment (+2)
    - Endpoint Behavior Truncate (+4)
    - **Note:** You can choose other **Options** settings, but these are typical.
4. Send the Curve Start Advanced (88) command. As specified by the Transition command, the curve axis will move to the correct position on the curve, which is determined by the location of the master axis. As the master axis moves in either direction, the curve axis will follow the curve profile. If the master reaches the endpoints (ax defined by the first and last x-value of the curve), it will behave as specified by the **Endpoint Behavior** in the **Options** command parameter.

Curve Capacity

In RAM

The RMC can hold up to 128 curves in it's random-access memory, which is lost when power is removed from the RMC. This curve storage capacity varies by RMC. The size required for any curve also depends on the download method, data format, and interpolation type. The following table gives approximate storage capacities for the commonly used simple cubic-interpolated curve. See the Curve Storage Capacity topic for more details.

	Max Length for a Single Curve	Max Length for 16 Equal-length Curves	Max Length for 128 Equal-length Curves
RMC75E	209,708 points	24,665 points	3,244 points
RMC75P **RMC75S**	8,186 points	783 points	95 points
RMC150E	209,708 points	24,665 points	3,244 points

RMC200 Lite	419,424 points	49,336 points	6,494 points
RMC200 Standard	1,677,715 points	197,370 points	26,002 points

In Non-Volatile Memory

The RMC can store curves in non-volatile Flash memory, which is retained even when power is removed from the RMC. Only curves downloaded from the Curve Tool or added using the Curve Add command with the Permanent life cycle option are saved to flash.

The size consumed by curves in Flash is 8 bytes per point for Constant and Linear types, and 16 bytes per point for Cubic types, plus 1 byte per character in the curve name and description, plus approximately 40 bytes of overhead.

The curves share the same Flash storage as the entire RMC project, including axis parameters, plot templates, variable table defaults, user programs, and curves. The total Flash size is as follows:

Controller	Flash Size
RMC75E (1.1G and newer)	1024 KB
RMC75E (1.1F and older)	256 KB
RMC75P	96 KB
RMC75S	96 KB
RMC150E	1024 KB
RMC200E	6016 KB

Firmware and Hardware Limitations

RMC75S and RMC75P require hardware versions 2.1D and 2.1E and newer for curves. The RMC75/150 require firmware 2.40.0 or newer, and RMCTools 3.38.0 and newer support the Curve Tool.

See Also

Curve Start (86) | Curve Start Advanced (88) | Curve Start (Prs/Frc) (87) | Curve Start Advanced (Prs/Frc) (89) | Transition Rate (56) | Transition Rate (Prs/Frc) (64) | Creating Curves Using the Curve Add Command | Curve Data Formats | Curve Interpolation Methods and Options | Example: Create Curve using the Curve Add Command | Curve Storage Capacity

3.6.7.2. Managing Curves in the Curve Tool

Use the Curve Tool for managing curves, including creating, editing, viewing, downloading, uploading, importing and exporting.

Managing Permanent Curves

A **permanent** curve is a curve either created in the Curve Tool or using the Curve Add (82) command with the *Permanent* life cycle. Permanent curves are synchronized between the controller and project file using the Curve Tool. Permanent curves are also saved to the flash memory in the controller when a Flash Update is initiated. See Updating Flash for details.

The permanent type curves in the Curve Tool are uploaded and downloaded as an entire set.

- Click the **Download Curves to Controller** button to download the entire set of curves in the project into the controller, replacing all permanent curves in the controller. It will leave temporary curves unaffected, except that temporary curves whose curve IDs matched a downloaded curve will be overwritten.

- Click the **Upload Curves from Controller** button to upload the entire set of permanent type curves from the controller into the project, replacing all curves in the project. Temporary curves are not uploaded into the project.

The Sync column in the Curves In Project and Curves in Controller windows is used by permanent curves to indicate the differences between the curves in the project and in the controller as follows:

	The curve exists both in the project and the controller, but the two curves are different.
	The curve exists only in the project. Download to copy the curve into the controller. Upload to delete the curve from the project.
	The curve exists only in the controller. Upload to copy the curve into the project. Download to delete the curve from the controller.

Managing Temporary Curves

A **temporary curve** is a curve that is created using the Curve Add command with the *Standard*, *Start-Once*, or *Complete-Once* life cycle. Temporary curves are not saved to flash memory or in the project file, and can be can be viewed but not edited in the Curves in Controller window in the Curve Tool.

A temporary curve can be copied into the project, making the copy a permanent type curve. To copy a temporary curve into the project, select a temporary curve and click the **Copy Curve to Project** button on the toolbar.

Creating and Editing Curves

These are basics of creating and editing curves in the Curve Tool. For more details, see the Curve Tool Overview topic.

Create a New Curve

1. In the toolbar, click the **Create New Curve** button. The new curve will appear in the **Curves in Project** window.
2. In the **Properties** pane, on the **Curve** tab, in the **Name** cell, enter a name for the new curve. You may also enter a **Description**.

Add Curve Points

1. In the spreadsheet located below the curve graph view, in the right-most column, enter the X value and Y value for the new point, then press Enter. Curve data can be copied and pasted from spreadsheets programs as described in the **Copying and Pasting** section in the Curve Tool topic.
2. Continue adding points by entering the X values and Y values in the right-most column. Even if the point is not going to be the last point in the curve, you must enter it in the right-most column. After entering the X and y values, the new point column will automatically be placed in the correct location in the spreadsheet.

Edit Curve Points or Properties

- To edit the points, simply edit the desired X values and Y values in the spreadsheet. Clicking a point in the graph view will highlight the point in the spreadsheet view, making it easy to find.
- Set the properties of the curve on the **Curves** tab in the Properties pane. See Curve Properties for details.

Download Curves to the Controller

- In the toolbar, click the **Download** button. This will download the curves in the project to the controller, overwriting the existing curves.

For more details, see the Curve Tool Overview topic.

See Also

Curves Overview | Curve Tool Overview | Curve Start (86)

3.6.7.3. Creating Curves Using the Curve Add Command

This topic describes how to create a curve using the Curve Add command and the variable table. This method is suitable for creating curves via a host controller, such as a PLC or PC, or from user programs.

For details on creating curves graphically in the Curve Tool, see the Curve Tool Overview. For details on creating curves that are too large to fit in the variable table, see Creating Large Curves Using the Curve Add Command.

Step-by-Step Example

Because of their many options, creating curves using the Curve Add command may seem complex. To better understand the process, refer to the step-by-step example in the Example: Create Curve Using the Curve Add Command topic.

How to Create a Curve using the Curve Add Command

1. **Choose a Data Format**
   The data you write to the variable table must be in the correct format. For adding a single curve, use one of the formats listed below and described in detail in the Curve Data Formats topic:

Single Curve Formats	Format Number	Description
Evenly-Spaced Points	0	The points are evenly spaced along the X axis. The starting x-value (X_0) and x-interval are specified, and then the Y values of each point.
Variable-Spaced Points	1	The points are each given independent X and Y values, allowing variable spacing. The X values must be increasing ($X_{i+1} > X_i$).
Advanced Points	2	This format expands on the Variable-Spaced Points format by allowing the velocity to be set at any point, and constant-velocity segments to be specified. The X values must be increasing ($X_{i+1} > X_i$).
Multiple Curve Formats		
Evenly-Spaced Points	10	Similar to the formats above, but are used to create multiple curves simultaneously, saving time. The multiple-curve formats are suitable for time-critical applications, such as curve sawing.
Variable-Spaced Points	11	
Advanced Points	12	

2.
3. **Write the Curve Data to the Variable Table**
   The curve data must be written to the Variable Table. You can write to any area of the variable table. Make sure to follow the format you chose, as listed in the Curve Data Formats topic.

4. **Send the Curve Add (82) Command**
   Use the Curve Add (82) command to tell the RMC that the data at a certain location in the Variable Table is a curve. The RMC grabs the data, and creates a curve with the specified Curve ID. If you chose multiple curves, the IDs are given sequentially, beginning with the specified ID.
   If a curve with the requested ID already exists, the existing curve will be deleted and the new curve with that ID added. The existing curve will not be deleted if the new curve is not added because of an error.

5. **Verify the Curve was Added Successfully**
   If the curve was successfully created, the Curve Status (the first item of the curve data in the Variable Table) will be set to a 3. If an error occurred, it will be set to a value indicating the error, as described in the Curve Add (82) topic. In addition, the Event Log will log an entry every time a curve is added or deleted.

Additional Tasks

Viewing the Curve

To view the curve, open the Curve Tool in RMCTools. In the Curves In Controller window, the curve you added will have the ID you assigned it. Click the curve to view it.

Editing the Curve in the Curve Tool

If you wish to edit the curve in the Curve Tool, first right-click the curve in the Curves in Controller window and choose **Copy Curve to Project**. Select the curve in the Curves in Project window. Now you can edit the points of the curve. When you are finished editing, click the download button  to download the curves in the project to the controller.

Saving Curves

Curves created in the Curve Tool or using the Curve Add command with the Permanent life cycle can be saved to Flash, will be included in the curves upload or download. After uploading the curves, the permanent type curve can also be saved in the project.

Deleting Curves

To delete curves, use the Curves in Controller window in the Curve Tool, or use the following commands:

- Curve Delete (83)
- Curve Delete All (85)
- Curve Delete Except (84)

Curves can also be deleted automatically, by specifying the **Start-Once** or **Complete-Once** life cycle in the Curve Add (82) command.

Curves with **Start-Once** life cycles can be deleted manually but will always be automatically deleted after being started once. The curve will exist internally for as long as it is being followed, but the curve ID will be freed up.

Curves with **Complete-Once** life cycles can be deleted manually but will always be automatically deleted after being completed once. The curve will exist internally for as long as it is being followed, but the curve ID will be freed up.

When a curve is deleted, an entry will be logged in the Event Log. A curve can be deleted while it is in use. The curve will exist internally for as long as it is being followed, but the curve ID will be freed up.

When power is removed from the RMC, all temporary curves (curves created using the Curve Add command with the Standard, Start-once, or Complete-once life cycles) and regular curves not saved to flash will disappear.

See Also

Curves Overview | Curve Add (82) | Curve Data Formats | Creating Large Curves Using the Curve Add Command

3.6.7.4. Creating Large Curves using the Curve Add Command

Large curves are curves that are too large to be written at once in their entirety to the variable table. Creating large curves via the Variable Table involves writing the curve data in parts to the variable table and issuing the Curve Add (82) command after each part until all the curve data has been written. This topic describes how to add curves in parts. Only one partial curve download can be in progress at a time.

See the Curve Storage Capacity topic for limits on how many curves can be added to the RMC.

Partial Curve Formats

Downloading a curve in parts requires using one of the Partial Curve formats:

- **Partial Curve - Evenly-Spaced Points (20)**
- **Partial Curve - Variable-Spaced Points (21)**
- **Partial Curve - Advanced Points (22)**

These formats are based on the single curve formats (see the Curve Data Formats topic), but with modifications to support downloading in parts. The following chart summarizes the partial curve formats:

Offset	Register	Description
0	Status	The value of this register when passed to the Curve Add (82) command is ignored, but after the Curve Add (82) command is issued, the value of this register will be controlled by the RMC and indicates the status of processing the curve. The user should monitor this to determine when the command is finished processing the curve data. The following values are defined: • (0) Processing Not Started By convention the user may want to set this register to this value before issuing the command. This is not strictly necessary, but otherwise the user must make sure not to check this register until after the command has been received. • (1) Processing Once the command has been received, the Status will immediately be set to Processing. While in this state the command is currently using the Curve Data structure, and the curve is not ready for interpolation. • (2) Part Complete For all but the last part in a partial curve download, once the command has completed processing the part, this status value will be used. The user can now write down the next part of the curve. • (3) Curve Ready Once the curve data has been processed, and the entire curve is ready for interpolation, the status will be set to this value. Notice that this value will be used for only the final part for curves submitted in parts. Therefore, for the last part of a curve, the **Part**

		Complete (2) state won't be used—the status will change directly from **Processing (1)** to **Curve Ready (3)**. • (10+) Error Values of 10 and above will indicate various errors. When an error code is set in this register, it indicates that the command is done trying to process this Curve Data, but the curve was not submitted to the curve store. See Curve Status Error Codes for a list of possible error values. It is good practice to set this value to zero when writing the data.
1	Format	One of the following: 20 - Evenly-Spaced Points 21 - Variable-Spaced Points 22 - Advanced Points This value must be the same for each part.
2	PartOffset	Offset of this part of the whole curve (in registers). This value will change for each part.
3	PartLength	Number of curve data registers in this part, not including the 5 header registers with offsets 0-4. Typically, this value is the same for each part except the last.
4	TotalLength	Total length of curve data (sum of all parts). This is the length of the corresponding single curve format, but with the Status and Format registers stripped off. This value must be the same for each part.
5..5+(L-1)	PartData	Segment of the curve data in the respective single curve format. The curve data here consists of the corresponding single curve format, but with the Status and Format registers stripped off.

Downloading Partial Curves to the RMC

To download a large curve to the RMC, repeat the following procedure until all the data has been downloaded:

- Write the data part to the RMC.

  Registers 0 - 4 of the Partial Curve Format listed above must always be included in each write. The actual data of the curve is included in registers 5 and on. The data in the registers 5 and on is the data of the corresponding single curve format, but with the Status and Format registers stripped off.
- Issue the Curve Add (82) command.
- Wait for the Curve Status register to be 2 or 3.

  The value 2 indicates that the part was successfully added.
  The value 3 indicates that the entire curve has now been successfully added.

Example

Suppose that the user wants to download a 2500-point curve through a block of 512 consecutive variables starting at %MD58.0 (Variable #512). The user decides to use the evenly-spaced format, and interpolation option of zero (0) for Zero-Velocity Endpoints, initial X_0 of 0, and an X interval of 0.1.

Since the user chose the **Evenly-Spaced Points** format, the curve data itself will require a total of 2504 registers, consisting of the items below. Notice that this is the **Single Curve Evenly-Spaced Points** format, but with the Status and Format registers stripped off.

Curve Data	
Offset	**Data Item**
0	Point Count
1	Interpolation Options
2	X_0
3	X interval
4	Y_0
5	Y_1
:	:
2503	Y_{2499}

The Variable Table area the user decided to use for the curve parts is 512 registers in length, which means the user can write up to 512 registers each time. The header for the Partial Curve format has 5 registers, which must be included in each part. Therefore, the user can write up to 507 Curve Data registers per part.

Here is the sequence of parts that would add this curve to the curve store, with each part being written and then submitted with a Curve Add (82) command:

Part	Address	Number Regs written to in Variable Table	PartOffset	PartLength	TotalLength
#1	%MD58.0	512	0	507	2504
#2	%MD58.0	512	507	507	2504
#3	%MD58.0	512	1014	507	2504
#4	%MD58.0	512	1521	507	2504
#5	%MD58.0	481	2028	476	2504

The user would send this curve to the RMC by completing the following sequences of writes to the Variable Table:

Note: This is very tedious to do directly from RMCTools, but you may need to when you are first trying it.

Part #1:

Variable	Data	Description	Curve Data Offset
512	0	Status = 0	-
513	20	Format = 20	-
514	0	Part Offset = 0	-
515	507	Part Length = 507*	-
516	2504	Total Length = 2504	-
517	2500	Point Count = 2500	0
518	0	Interpolation Options = 0	1

519	0	$X_0 = 0$	2
520	0.1	X interval = 0.1	3
521	Y_0	Y_0	4
522	Y_1	Y_1	5
:	:	:	:
1023	Y_{502}	Y_{502}	506

*This number includes the four parameters *and* the number of points in this curve.

After writing part #1 and issuing the Curve Add command, the user should wait for the Curve Status at Variable 512 to become 2 (if you are looking in RMCTools, make sure to look in the Monitor tab of the Variable Table Editor), indicating that the RMC has completed processing the part.

Part #2

Variable	Data	Description	Curve Data Offset
512	0	Status = 0	-
513	20	Format = 20	-
514	507	Part Offset = 507	-
515	507	Part Length = 507	-
516	2504	Total Length = 2504	-
517	Y_{503}	Y_{503}	507
:	:	:	:
1023	Y_{1009}	Y_{1009}	1013

After writing part #2 and issuing the Curve Add command, the user should wait for the Curve Status at Variable 512 to become 2 (if you are looking in RMCTools, make sure to look in the Monitor tab of the Variable Table Editor), indicating that the RMC has completed processing the part.

Part #3

Variable	Data	Description	Curve Data Offset
512	0	Status = 0	-
513	20	Format = 20	-
514	1014	Part Offset = 1014	-
515	507	Part Length = 507	-
516	2504	Total Length = 2504	-
517	Y_{1010}	Y_{1010}	1014
:	:	:	:
1023	Y_{1516}	Y_{1516}	1520

After writing part #3 and issuing the Curve Add command, the user should wait for the Curve Status at Variable 512 to become 2 (if you are looking in RMCTools, make sure to look in the Monitor tab of the Variable Table Editor), indicating that the RMC has completed processing the part.

Part #4

Variable	Data	Description	Curve Data Offset

512	0	Status = 0	-
513	20	Format = 20	-
514	1521	Part Offset = 1521	-
515	507	Part Length = 507	-
516	2504	Total Length = 2504	-
517	Y_{1517}	Y_{1517}	1521
:	:	:	:
1023	Y_{2023}	Y_{2023}	2027

After writing part #4 and issuing the Curve Add command, the user should wait for the Curve Status at Variable 512 to become 2 (if you are looking in RMCTools, make sure to look in the Monitor tab of the Variable Table Editor), indicating that the RMC has completed processing the part.

Part #5

Variable	Data	Description	Curve Data Offset
512	0	Status = 0	-
513	20	Format = 20	-
514	2028	Part Offset = 2028	-
515	476	Part Length = 476	-
516	2504	Total Length = 2504	-
517	Y_{2024}	Y_{2024}	2028
:	:	:	:
992	Y_{2499}	Y_{2499}	2503

After writing part #5 and issuing the Curve Add command, the user should wait for the Curve Status at Variable 512 to become 3 (if you are looking in RMCTools, make sure to look in the Monitor tab of the Variable Table Editor), indicating that the RMC has completed processing the curve. Notice that the curve data point numbering begins with 0, and therefore 2499 is the last point of the 2500-point curve.

See Also

Curves Overview | Curve Data Formats | Curve Add (82)

3.6.7.5. Curve Interpolation Methods and Options

The RMC supports several interpolation methods and options to satisfy a wide range of curve applications.

Interpolation Methods

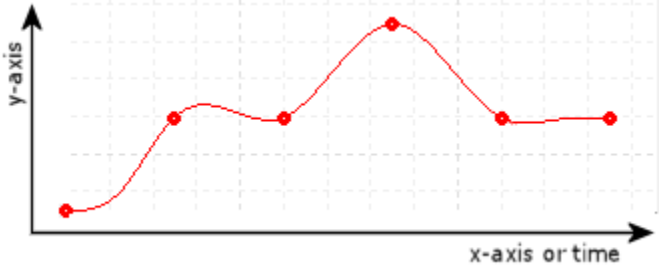
The interpolation method is specified in the Properties pane in the Curve tool, or in the Curve Add (82) command. Choose from one of the methods below. The **Cubic (2)** method is the most common method and creates the smoothest motion.

- **Cubic (2)**

The curve will smoothly go through all points. This is the most common interpolation method. This method will create smooth motion.

On position axes, the Velocity Feed Forward and Acceleration Feed Forward will apply to cubic interpolated curves, but higher order gains should not be used, such as the Jerk Feed Forward, Double Differential Gain, and Triple Differential Gain.

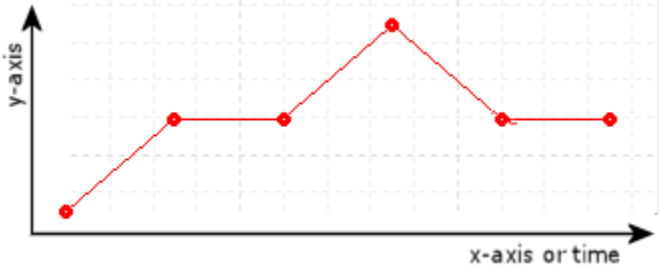
On pressure or force axes, the Pressure/Force Rate Feed Forward will apply.



- **Linear (1)**

  The curve will consist of straight-line segments between each point. Because the velocity is not continuous, a position axis will tend to overshoot at each point. This type of curve is typically more suitable for pressure or force axes.

  On position axes, the Target Acceleration will always be zero. Therefore, the Acceleration Feed Forward will have no effect for linear interpolated curves.

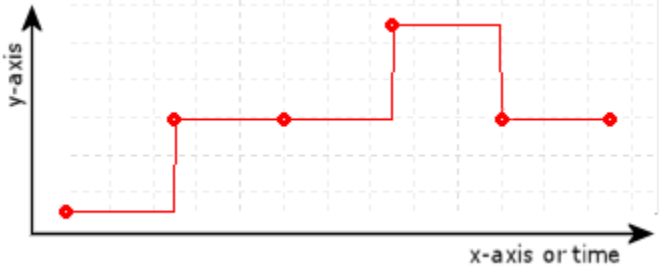


- **Constant (0)**

  The curve will consist of step jumps to each point. The curve will not be continuous. This method is seldom used, but may be useful in applications where step jumps are desired, such as some blow-molding systems. This method requires that the axis not be tuned very tightly, or the axis may oscillate and Output Saturated errors may occur. The Position I-PD control algorithm is recommended for following constant interpolated curves.

  On position axes, the Target Velocity and Target Acceleration will always be zero. Therefore, the Velocity Feed Forward and Acceleration Feed Forward will have no effect for constant interpolated curves.

  On pressure or force axes, the Target Rate will always be zero. Therefore, the Pressure/Force Rate Feed Forward will have no effect for constant interpolated curves.



Interpolation Options

The interpolation options are specified in the curve data. The available options depend on the interpolation method, as shown in the table below. Add the numbers for each desired option. For example, to choose Cyclic Curve and Overshoot Protection, the Interpolation Option value would be 2+8 = 10.

Interpolation Type	Interpolation Options
Constant	None
Linear	None
Cubic	**Endpoint Behavior** Choose only one. See the **Endpoint Behavior** section below for details. **+0 Zero-Velocity Endpoints** **+1 Natural-Velocity Endpoints** **+2 Cyclic Curve** **Overshoot Protection** Choose only one. See the **Overshoot Protection** section below for details. **+0 Disabled** **+8 Enabled** **Auto Constant Velocity** Choose only one. See the **Auto Constant Velocity** section below for details. **+0 Disabled** **+16 Enabled**

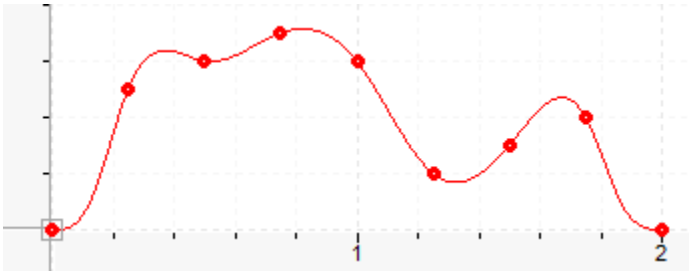
Endpoint Behavior

This option defines the behavior at the endpoints. Each option below shows a plot for the following cubic curve data with 9 points at 0.25 second intervals:

Y-axis	1.0	1.5	1.6	1.7	1.6	1.2	1.3	1.4	1.0
time	0.00	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2.00

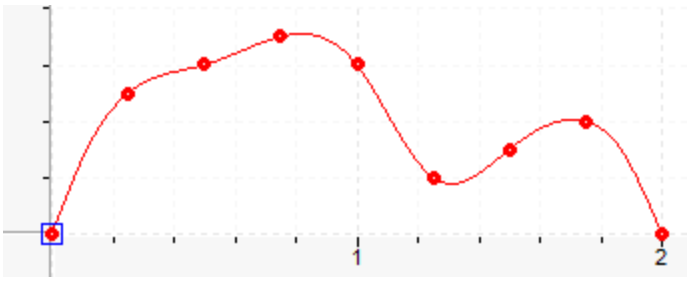
- **+0 Zero-Velocity Endpoints**

  The endpoints will have zero velocity and acceleration. This is the most commonly-used option. Curves with zero-velocity endpoints can be repeated cyclically.



- **+1 Natural-Velocity Endpoints**

  The endpoints will have their velocity automatically selected to match the natural slope of the curve at the endpoints. The acceleration at the endpoints will be zero. Curves with natural-velocity endpoints cannot be repeated cyclically because the endpoint velocities are typically not equal.

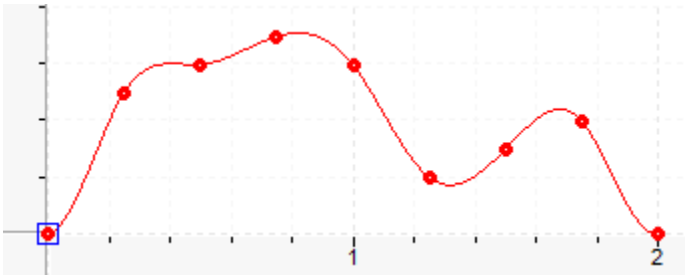


- **+2 Cyclic Curve**

  The endpoints are assumed to wrap. Therefore the acceleration, velocity, and position will be continuous between cycles of this curve, although the velocity and acceleration at each endpoint will not necessarily be zero.

  If the y-values of the first and last points are not equal, and the curve is run for more than 1 consecutive cycle, the curve will be automatically offset so that the first point of the next cycle matches the last point of the previous cycle.

  For Advanced format curves, if an endpoint is defined as a Fixed Velocity type, it will use that velocity even if the curve is cyclic. Setting each endpoint to a different fixed velocity will cause a discontinuity in the velocity between cycles of the curve, but the curve can still be used cyclically.



Overshoot Protection

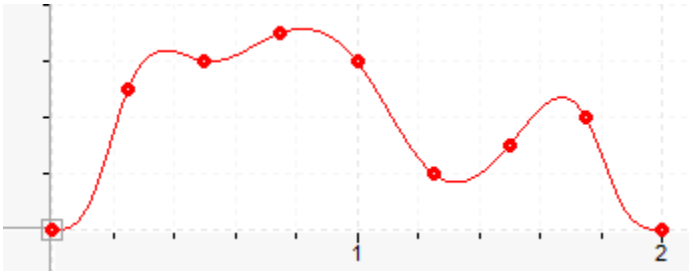
The Overshoot Protection option prevents the curve from exceeding a local maximum or local minimum point. A local maximum occurs where a point is greater than or equal to both the preceding point and the following point. A local minimum occurs where a point is less than or equal to both the preceding point and the following point. Overshoot protection will not prevent overshooting in other locations.

When overshoot protection is enabled, the velocity is set to zero at each local minimum/maximum point, which eliminates the chance of the curve overshooting that point for the curve segments on either side of the point.

For Advanced format curves, Overshoot Protection will not apply to Fixed-Velocity points, or points at the beginning or end of a Constant-Velocity segment.

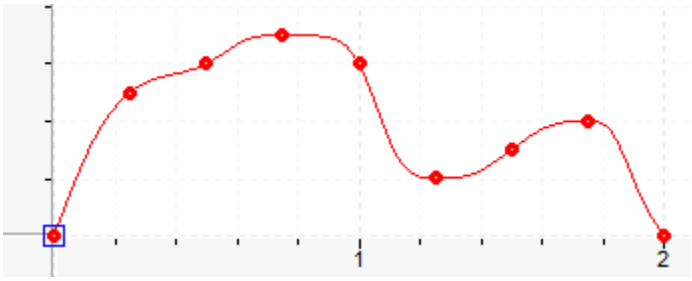
Example 1

Consider the cubic curve data in the **Endpoint Behavior** section above. Without Overshoot Protection enabled, the curve looks like this:



Notice that the curve overshoots the points after times 0.75, 1.25, and before 1.75.

With overshoot protection enabled, the curve will look like this:



Notice that the curve no longer overshoots the points after times 0.75, 1.25, and before 1.75. Notice, however, that the curve still overshoots between points 0.25 and 0.5 because neither point is a local minimum or maximum.

Auto-Constant Velocity

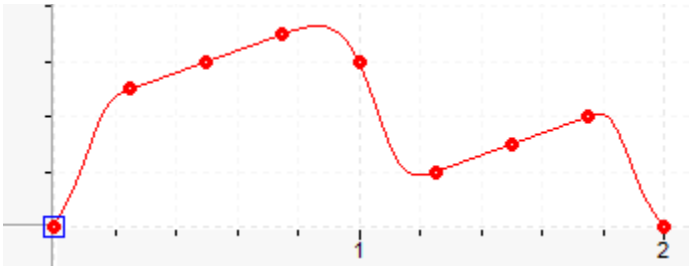
The Auto-Constant Velocity option will automatically insert a linear segment in the curve if three or more data points are in a straight line.

If you have also selected Overshoot Protection, be aware that the points identified as local minimums or maximums do not count as consecutive points for the Auto-Constant Velocity (see the example below).

For Advanced format curves, be aware that the Fixed Velocity type points do not count as consecutive points for the Auto-Constant Velocity (see the example below).

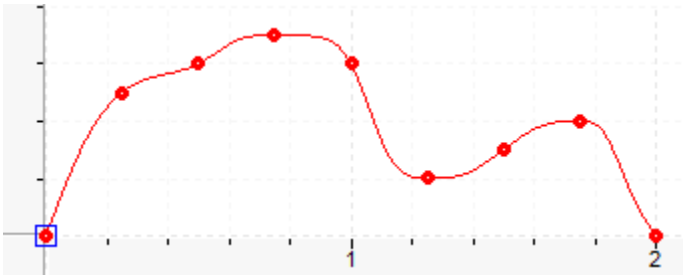
Example 2

Consider the cubic curve data in the **Endpoint Behavior** section above. The points at times 0.25, 0.50, and 0.75 are in a straight line, as are the points at times 1.25, 1.50, and 1.75. With the Auto-Constant Velocity option, the curve will look like this:



Example 3

Consider this same curve with both Overshoot Protection and Auto-Constant Velocity enabled. This particular curve ends up looking the same as it does with only Overshoot Protection, because both constant-velocity segments are lost because at least one of each set of 3 consecutive points was identified by Overshoot Protection as a local minimum or maximum.



See Also

Curves Overview | Creating Curves Using the Curve Add Command | Curve Data Formats

3.6.7.6. Curve Data Formats

Several Curve Data Formats are available, which specify several things, including (1) the format of each point in the curve, (2) whether a single curve or multiple curves are being submitted, and (3) whether a complete curve or a part of a curve is being submitted.

The Curve Data always starts with the following two registers:

Note: Data types are all REAL.

Offset	Register	Description
0	Status	The value of this register when passed to the Curve Add (82) command is ignored, but after the Curve Add (82) command is issued, the value of this register will be controlled by the RMC and indicates the status of processing the curve. The user should monitor this to determine when the command is finished processing the curve data. The following values are defined: • (0) Processing Not Started By convention the user may want to set this register to this value before issuing the command. This is not strictly necessary, but otherwise the user must make sure not to check this register until after the command has been received. • (1) Processing Once the command has been received, the Status will immediately be set to Processing. While in this state the command is currently using the Curve Data structure, and the curve is not ready for interpolation. • (2) Part Complete As described below, very long curves can be submitted in parts. Once the command has completed processing this Curve Data structure (i.e. this part of the total curve), this status value will be used. The user can now write down the next part of the curve. • (3) Curve Ready Once the curve data has been processed, and the entire curve is ready for interpolation, the status will be set to this value. Notice that this value will also be used for the final part for long curves submitted in parts. Therefore, for the last part of a curve, the Part Complete (2) state wont be usedthe status will change from Processing (1) directly to Curve Ready (3). • (10+) Error Values of 10 and above will indicate various errors. When an error code is set in this register, it indicates that the command is done trying to process this Curve Data, but the curve was not submitted to the curve store. See the Curve Status Error Codes topic for details.
1	Format	This register is set by the user and indicates the format of the curve data structure that follows. The following formats are defined:

		(0) Single Curve - Evenly-Spaced Points In this format, the points are assumed to be evenly spaced along the X axis. Therefore, the X_0 and X Interval are specified, but otherwise, just an array of Y values is included. (1) Single Curve - Variable-Spaced Points In this format, the points are each given independent X and Y values, allowing variable spacing. (2) Single Curve - Advanced Points This format expands on the Variable-Spaced Points format by allowing the velocity to be set at any point, and constant-velocity segments to be specified. The Multiple Curve Formats allow multiple curves will be added at once. The user need only issue one Curve Add command and check a single Status register and know that all the curves have been successfully submitted. (10) Multiple Curves - Evenly-Spaced Points In this format, the points are assumed to be evenly spaced along the X axis. Therefore, the X_0 and X Interval are specified, but otherwise, just an array of Y values is included. (11) Multiple Curves - Variable-Spaced Points In this multiple curve format, the points are each given independent X and Y values, allowing variable spacing. (12) Multiple Curves - Advanced Points This multiple curve format expands on the Variable-Spaced Points format by allowing the velocity to be set at any point, and constant-velocity segments to be specified. The partial curve formats are for adding a curve that is too large to submit all at once, and is described in detail in the Creating Large Curves using the Curve Add Command topic. (20) Partial Curve - Evenly-Spaced Points In this format, the points are assumed to be evenly spaced along the X axis. Therefore, the X_0 and X Interval are specified, but otherwise, just an array of Y values is included. (21) Partial Curve - Variable-Spaced Points In this partial curve format, the points are each given independent X and Y values, allowing variable spacing. (22) Partial Curve - Advanced Points This partial curve format expands on the Variable-Spaced Points format by allowing the velocity to be set at any point, and constant-velocity segments to be specified.

Single Curve - Evenly-Spaced Points

This is the simplest method. The spacing of the X values is constant. The X value for each point is defined as $X_i = X_0 + \Delta X \cdot i$. This method does not support fixing velocities at any point or constant-

velocity segments on cubic interpolated curves, except as defined by the selected Interpolation Options.

Note: Data types are all REAL.

Offset	Register	Description
0	Status	See Above
1	Format	(0) Single Curve - Evenly-Spaced Points
2	PointCount	Number of points in the curve (N). The minimum number of points is 2.
3	InterpOpt	Interpolation Options. See the Curve Interpolation Methods and Options topic for details. +0: Zero-Velocity Endpoints +1: Natural-Velocity Endpoints +2: Cyclic Curve +8: Overshoot Protection +16: Auto Constant Velocity
4	X_0	Point 0 X value This value is *only* used when following a curve with absolute master alignment (only available with the Curve Start Advanced (88) command). This value is *not* used *w*hen following a curve using the Curve Start (86) or Curve Start (Prs/Frc) (87) comamnds; the X_0 point is the current value of the master at the time the command is issued. The remaining X-axis points are computed relative the master position when the Curve Start command is issued.
5	X Interval (ΔX)	Interval in X between Points
6	Y_0	Point 0 Y value
7	Y_1	Point 1 Y value
:	:	:
6+(N-1)	Y_{N-1}	Point N Y Value

Single Curve - Variable-Spaced Points

In this format, points are each given independent X and Y values—allowing variable spacing—but does not support fixing velocities or specifying constant-velocity segments on cubic interpolation, except as defined by the selected Interpolation Options. The X values must be increasing ($X_{i+1} > X_i$).

Note: Data types are all REAL.

Offset	Register	Description
0	Status	See Above
1	Format	(1) Single Curve - Variable-Spaced Points
2	PointCount	Number of points in the curve (N). The minimum number of points is 2.
3	InterpOpt	Interpolation Options. See the Curve Interpolation Methods and Options topic for details. +0: Zero-Velocity Endpoints +1: Natural-Velocity Endpoints

		+2: Cyclic Curve +8: Overshoot Protection +16: Auto Constant Velocity
4	X_0	Point 0 X value This value is *only* used when following a curve with absolute master alignment (only available with the Curve Start Advanced (88) command). This value is *not* used *w*hen following a curve using the Curve Start (86) or Curve Start (Prs/Frc) (87) comamnds; the X_0 point is the current value of the master at the time the command is issued. The remaining X-axis points are computed relative the master position when the Curve Start command is issued.
5	Y_0	Point 0 Y value
6	X_1	Point 1 X value
7	Y_1	Point 1 Y value
:	:	
4+(N-1)·2	X_{N-1}	Point N-1 X value
5+(N-1)·2	Y_{N-1}	Point N-1 Y value

Single Curve - Advanced Points (2)

In this format, points are each given independent X and Y values—allowing variable spacing—and may be given fixed velocities. Also, segments can be designated as having a constant velocity. The X values must be increasing ($X_{i+1} > X_i$).

Note: Data types are all REAL.

Offset	Register	Description
0	Status	See Above
1	Format	(2) Single Curve - Advanced Points
2	PointCount	Number of points in the curve (N). The minimum number of points is 2.
3	InterpOpt	Interpolation Options. See the Curve Interpolation Methods and Options topic for details. +0: Zero-Velocity Endpoints +1: Natural-Velocity Endpoints +2: Cyclic Curve +8: Overshoot Protection +16: Auto Constant Velocity
4-7	Pt_0	Point 0 structure. Each field is described below:
4	.Type	Point 0 Type. It can have these values: **(0) Standard** Only the X and Y values are used. **(1) Fixed Velocity** The V value later in this structure specifies the velocity to be used at this point. This point type is not supported by Constant or Linear interpolated curves.

		If a point is defined as Fixed Velocity, but the preceding point is defined as Constant-Velocity, then the fixed velocity will be ignored and will be the same as the velocity calculated for the preceding Constant-Velocity segment. **(2) Constant–Velocity Segment** The segment between this point and the next is assumed to have fixed velocity. The velocity is implied from the X_i, Y_i, X_{i+1}, and Y_{i+1} values, and therefore the V value is unused. This point type is not supported by Constant interpolated curves. Setting the type of the last point in the curve to Constant-Velocity Segment will have no effect; it will behave as a Standard type.
5	.X	Point 0 X value This value is *only* used when following a curve with absolute master alignment (only available with the Curve Start Advanced (88) command). This value is *not* used *w*hen following a curve using the Curve Start (86) or Curve Start (Prs/Frc) (87) comamnds; the X_0 point is the current value of the master at the time the command is issued. The remaining X-axis points are computed relative the master position when the Curve Start command is issued.
6	.Y	Point 0 Y value
7	.V	Point 0 V value. This is used only for type 1 (fixed velocity), and ignored by all other types.
8-11	Pt_1	Point 1 structure
12-15	Pt_2	Point 2 structure
:	:	
4+(N-1)·4.. **7+(N-1)·4**	Pt_{N-1}	Point N-1 structure

If the Zero-Velocity Endpoints interpolation option is chosen, the velocity at the endpoints will zero, regardless of the point type.

Multiple Curve Formats

The multiple curve formats allow submitting multiple curves at once. They share the same curve data as the single curve formats with only slight modifications. When adding curves using the Multiple Curves formats, the curves will receive sequential ID numbers, starting with the ID specified by the Curve Add (82) command.

- Multiple Curves - Evenly-Spaced Points (10)
- Multiple Curves - Variable-Spaced Points (11)
- Multiple Curves - Advanced Points (12)

The following chart summarizes the multiple curves formats:

Note: Data types are all REAL.

Offset	Register	Description

0	Status	See Above
1	Format	Multiple Curves format (10-12)
2	CurveCount	Number of curves that follow (C)
3	RegsPerCurve	Number of registers to reserve for each curve (R) This must be at least the number of registers required for the single curve format, minus two. Notice that each curve in the Multiple Curve Format need not fill the area specified by the **RegsPerCurve**. Extra registers are ignored.
4..4+(R-1)	$Curve_0$	Data for Curve (PointCount) This is the Point Count, since this the Data for Curve is the same data as the single curve formats, but without the first two registers (Status and Format).
4+R.. **4+R+(R-1)**	$Curve_1$	Data for Curve 1
:	:	:
4+C·R-R.. **4+C·R-1**	$Curve_{C-1}$	Data for Curve C-1

The format of the data within the $Curve_i$ is the same as the single curve formats, but without the first two registers (Status and Format).

Each curve in the Multiple Curve Format need not fill the area specified by the **RegsPerCurve**. Extra registers are ignored.

Partial Curve Formats

The partial curve formats allow submitting a very long curve in pieces. They are based on the single curve formats, but with modifications to support segmenting the entire download. For details on how to download large curves using the partial format, see the Creating Large Curves Using the Curve Add Command topic.

- Partial Curve - Evenly-Spaced Points (20)
- Partial Curve - Variable-Spaced Points (21)
- Partial Curve - Advanced Points (22)

The following chart summarizes the partial curve formats:

Note: Data types are all REAL.

Offset	Register	Description
0	Status	See Above
1	Format	Partial Curve format (20-22)
2	PartOffset	Offset of this part of the whole curve (in registers)
3	PartLength	Number of registers in this part (L), excluding the 5-register part header. This is the number of PartData registers.
4	TotalLength	Total length of curve data (sum of all parts)
5..5+(L-1)	PartData	Segment of the curve data in the respective single curve format.

See Also

Curves Overview | Curve Status Error Codes | Curve Interpolation Methods and Options | Curve Add (82) | Curve Start (86) | Curve Start (Prs/Frc) (87)

3.6.7.7. Curve Status Error Codes

When downloading a curve to the RMC via the variable table, the Curve Status register in the curve data will contain an error code if the Curve Add (82) command could not successfully add the curve data to the curve store. For more details on the Curve Status, see the Curve Data Formats topic.

The following error code values are defined:

10: Invalid curve data address.

Indicates that the Curve Data command parameter was not a valid curve data address. Only addresses in the Variable Tables current or initial values are acceptable.

11: Curve data is not in the correct format.

The Curve Data did not follow one of the supported Curve Data Formats. Examples of problems that will give this error include:

- Invalid Format value.
- Curve Data runs off the end of the variable table.
- In a partial curve, the Part Offset is beyond the Total Curve Data Length.
- In a partial curve, the Part Offset is negative.
- In a partial curve, the Total Curve Data Length is negative or zero.
- In a partial curve, the Part Length is negative or zero.
- In a multiple curve format, the number of registers per curve is less than six (6).
- In a multiple curve format, the number of curves is negative or zero.
- In a multiple curve format, an individual Curve Data does not fit in the registers reserved for it.
- The Interpolation Options field is not a valid value.

12: Too many pending Curve Adds.

This error indicates that a Curve Add command could not be accepted because the curve processing queue was full. That is, there are already sixteen Curve Add or Delete requests in the queue. The Curve Add command will also signal a command error for this error.

13: No room for the curve in the curve storage area.

Unable to allocate memory from the Curve Store for the curve. There are three times that a curve may require a block of memory from the Curve Store. First, if a partial curve format is used, a temporary buffer must be allocated to hold the entire download. Second, if the curve interpolation method requires significant pre-processing then a temporary block of memory must be allocated. Third, the processed curve itself will be allocated from the Curve Store. This error code will be used if the Curve Store does not have enough space free for any of these allocations. See Curve Storage Capacity for details on memory requirements.

14: Non-increasing X value found in curve data.

The X values for each successive point must be increasing. This error will be generated for the Variable-Spaced or Advanced curve data formats if the X values are not increasing, or in the Evenly-Spaced format if the X Interval is less than or equal to zero.

15: Invalid floating point value found in curve data.

This indicates that one or more point X, Y, or V values are invalid floating point values. For the Evenly-Spaced format, this also includes the X_0 and X Interval values.

16: Point type found in curve data that is not valid.

In the Advanced curve data formats, the Point Type field must be 0, 1, or 2.

17: The curve has fewer than the required number of data points.

The Curve has fewer than the required number of points. All curves must have at least 2 points.

18: Interpolation Options are selected that are not supported by the selected interpolation method.

The Interpolation Options register had an invalid value for the selection interpolation method. For example, the Constant and Linear interpolation methods do not support any Interpolation Options, so this field must be zero. Refer to the Curve Interpolation Methods and Options topic for acceptable values for each interpolation method.

19: Point type found that is not supported by the selected interpolation method.

This error can occur if the Advanced curve data format is used with the Constant or Linear interpolation methods if the point type is inconsistent with the interpolation method. For example, the Constant interpolation method can only have point types of **Standard**, and the Linear interpolation method can only have point types of **Standard** or **Constant-Velocity Segment**.

20: Cyclic curve with not equal first and last points.

This error will occur if the curve with cubic interpolation uses the Cyclic interpolation option, but the first and last points in the curve do not have the same Y value. Firmware versions 3.39.0 and newer allow unequal first and last y values, so this error will not occur.

21: Curve calculations overflowed.

This error can occur for curves with cubic interpolation if processing the curve caused a numeric overflow. This will occur only for curves with extreme point definitions, such as extremely small changes in X, extremely large changes in Y, or extremely large velocities.

22: Curve Part had fields that mismatched previous part(s).

This error can occur when downloading a curve in multiple parts if certain values do not match between subsequent parts of the same curve. Specifically the following items must match across all parts in a curve:

- Format field
- Total Curve Data Length field
- Curve ID command parameter
- Interpolation Method command parameter
- Life Cycle command parameter

23: Curve Part was received out of order.

This error can occur when downloading a curve in multiple parts if the Part Offset field does not match the next expected offset. The next part to be downloaded must have its Part Offset set to the previous Part Offset plus the previous Part Length. A part offset of zero (0) is always accepted as a new curve download, and will cause any in-progress partial download to be discarded.

24: Maximum number of curves reached.

This error will occur if there are already 128 curves in the curve store. You must delete one or more curves before adding more curves.

25: Interpolation method requires equal intervals.

Reserved for future use.

See Also

Curves Overview | Curve Add (82) | Curve Data Formats

3.6.7.8. Curve Storage Capacity

Curves are stored in the Curve Store. The Curve Store can hold a maximum of 128 curves. The non-volatile storage capacity for curves is less. The curve storage capacity varies by controller, as shown below. When a curve is added, the Event Log will display the size of the curve in bytes.

Controller	Curve Store Capacity
RMC200 Lite	16 MB
RMC200 Standard	64 MB
RMC150E	8 MB
RMC75E	8 MB
RMC75S	256 KB
RMC75P	256 KB

Non-Volatile Memory

The curves in the Curve Store will be lost when power is removed from the RMC. However, curves can be saved to non-volatile Flash memory if they were downloaded from the Curve Tool or added using the Curve Add command with the Permanent life cycle option.

The size consumed by curves in Flash is as follows, where N is the number of points in the curve:

Curve Type	Size
Constant, Linear	8xN + name length + description length + 40 bytes
Cubic	16xN + name length + description length + 40 bytes

The curves share the same Flash storage as the entire RMC project, including axis parameters, plot templates, variable table defaults, user programs, and curves. The total Flash size is as follows:

Controller	Flash Size
RMC200	6016 KB
RMC150E	1024 KB
RMC75E (1.1G and newer)	1024 KB
RMC75E (1.1F and older)	256 KB
RMC75P	96 KB
RMC75S	96 KB

Curve Store Capacity Quick Charts

The following charts show the maximum allowable sizes for equal-length simple cubic curves in the various RMC controllers with various numbers of curves. This assumes curves are added via the Curve Add command with the Partial Curve - Equally Spaced format. This will fill the curve store, but the curves will not fit in non-volatile memory.

Simple Cubic Curves, Max Length for:	RMC200 Standard	RMC200 Lite	RMC75E RMC150E	RMC75P RMC75S
Single Curve	1,677,715 points	419,424 points	209,708 points	8,186 points
8 Equal-length Curves	372,819 points	93,199 points	46,596 points	1,518 points
16 Equal-length Curves	197,370 points	49,336 points	24,665 points	783 points

50 Equal-length Curves	65,784 points	16,440 points	8,216 points	253 points
80 Equal-length Curves	41,416 points	10,348 points	5,170 points	156 points
128 Equal-length Curves	26,002 points	6,494 points	3,244 points	95 points

Curve Store Memory Uses

The Curve Store contains a pool of memory as described above. There are four operations that will consume Curve Store memory:

- **Download Buffer**
  This buffer is allocated for each curve downloaded from the Curve Tool only for the duration of that curve download. Therefore no more than one download buffer is allocated at a time.
- **Re-assembly Buffer**
  This buffer is allocated when a partial curve download is started. This buffer is used to re-assemble the parts submitted individually into a single curve data structure. This buffer is freed up after the final part is added and the curve is ready, or when a new partial curve download is started—cancelling the existing download—or when the Curve Delete All (85) command is issued. Notice that *no more than one* re-assembly buffer is allocated at a time.
- **Computation Buffer**
  This buffer is allocated when a curve is being added if the interpolation method requires extra memory to prepare the curve for interpolation. This currently only applies to curves using the cubic interpolation method. This buffer is freed up when the curve adding is completed. Like the re-assembly buffer, *no more than one* computation buffer is allocated at a time.
- **Curve Object**
  Each curve that has been successfully added to the Curve Store will occupy a block of memory in the Curve Store until that curve is deleted. See the Curves Overview topic for details on deleting curves. There can be up to 128 curve objects in the curve store at once, each potentially a different size.

Download Buffer Size

This buffer is allocated for each curve downloaded from the Curve Tool only for the duration of that curve download. No more than one download buffer is allocated at a time. The download buffer size depends on the interpolation type as shown below, where N is the number of points in the curve:

Curve Type	Size
Constant, Linear	8xN + name length + description length + 40 bytes
Cubic	16xN + name length + description length + 40 bytes

Re-assembly Buffer Size

As described in the Creating Large Curves using the Curve Add Command topic, the partial curve data format has a field called Total Curve Data Length (TCDL). This field indicates the number of registers in the re-assembled curve data. Using the TCDL value, the size of the re-assembly buffer size can be found using the following formulas:

	RMC200	RMC75E RMC150E	RMC75P RMC75S
Re-assembly Buffer Size	TCDL x 4 + 16 bytes	TCDL x 4 + 32 bytes	TCDL x 4 + 8 bytes

Computation Buffer and Curve Object Sizes

The amount of space required for the computation buffer and curve object depends on several factors. The most important factor is the interpolation method to be used. Formulas for calculating the memory requirements are given below.

Constant and Linear Interpolated Curves

These two types of curves do not require a computation buffer. The curve object sizes are given in the chart below:

	RMC200	RMC75E RMC150E	RMC75P RMC75S
Curve Size	8xN + 112 bytes	8xN + 112 bytes	8xN + 88 bytes

Cubic Interpolation Curves

The memory requirements for cubic curves depend on the options and characteristics of the curve. The term "simple curve" below means that the curve has no fixed-velocity points or constant-velocity segments, which can be added by the Advanced curve data format or the Overshoot Protection and Auto-Constant Velocity interpolation options. The term "advanced curve" below describes a curve that *does* have at least one fixed-velocity point or constant-velocity segment.

The following charts show the memory requirements for simple curves:

Simple Curves with Zero- or Natural-Velocity Endpoints

	RMC200	RMC75E RMC150E	RMC75P RMC75S
Curve Size	20xN + 160 bytes	20xN + 144 bytes	20xN + 120 bytes
Computation Buffer Size	16xN + 48 bytes	16xN + 64 bytes	8xN + 24 bytes

Simple Cyclic Curves

	RMC200	RMC75E RMC150E	RMC75P RMC75S
Curve Size	20xN + 120 bytes	20xN + 120 bytes	20xN + 96 bytes
Computation Buffer Size	48xN - 32 bytes	48xN + 32 bytes	24xN + 8 bytes

The sizes for advanced curves are difficult to compute, but will generally be at least as large as the values shown in the **Simple Curves with Zero- or Natural-Velocity Endpoints** chart above. The best way to determine the size required by an advanced curve is to add the curve and look at the size reported in the Event Log.

Memory Requirement Examples

Example 1

How much Curve Store memory is used to add a 10,000-point Simple Cubic Curve with Zero-Velocity Endpoints to the RMC75E using the Curve Add command?

Downloading this curve using the Curve Add command will require the Partial Curve Data Format, and therefore require a re-assembly buffer. If we use the **Partial Curve – Evenly-Spaced** format (see Curve Data Formats), then the Total Curve Data Length will be 10,004 registers. Therefore, our re-assembly buffer will require 4xTCDL + 32 or 40,048 bytes.

Next, because this curve uses the cubic interpolation method, the curve will require a Computation Buffer. Using the charts above, we find that the Computation Buffer for a Simple Cubic Curve with Zero-Velocity Endpoints will require 16xN+64 bytes or 160,064 bytes.

Finally, the curve object itself will require 20xN+144 bytes or 200,144 bytes. Therefore, the Curve Store must have 40,048 + 160,064 + 200,144 or 400,256 bytes available to successfully download this curve. Notice however, that after the curve has been added, the re-assembly and computation buffers will be freed up, leaving only 200,144 bytes used for this curve from the Curve Store.

Example 2

How much Curve Store memory is used to add fifty (50) Cubic Curves with Zero-Velocity Endpoints—each with 10,000 points—to the RMC75E using the Curve Tool?

When downloading a curve using the Curve Tool, each download will require a Download Buffer of 16 bytes per point plus curve name and description length plus 40 bytes. If we assume that each curve's name and description use 20 characters total, then each 10,000-point Cubic curve will require a download 160,060 bytes. The Computation Buffer and Curve Object will have the same sizes as calculated in Example 1, 160,064 bytes and 200,144 bytes respectively.

When each curve is downloaded, the Download Buffer will be allocated, then the Curve Object, and finally the Computation Buffer. Once the curve is successfully added, the Download and Computation Buffers will be freed up, leaving only the Curve Object in the Curve Store. Each of the 50 curves is downloaded in this manner. Therefore, the approximate total Curve Store memory requirement for 50 of these curves is 50 Curve Objects, 1 Download Buffer, and 1 Computation Buffer, which is (200,144 x 50) + 160,060 + 160,064 = 10,327,324 bytes or 9.8 MB. Notice that this exceeds the 8MB Curve Store capacity available on the RMC75E. Therefore, these curves cannot be downloaded to this controller.

Fragmentation of the Curve Store Memory

After adding and deleting curves, the Curve Store memory may become fragmented, limiting the memory available for adding curves. To clear the memory completely, use the Curve Delete All (85) command.

See Also

Curves Overview | Curve Data Formats

3.6.7.9. Example: Creating a Curve using the Curve Add Command

This topic gives a simple example of how to create a curve using the Curve Add (82) command and the variable table. For more details on curves, see the Curves Overview and Curve Tool topics.

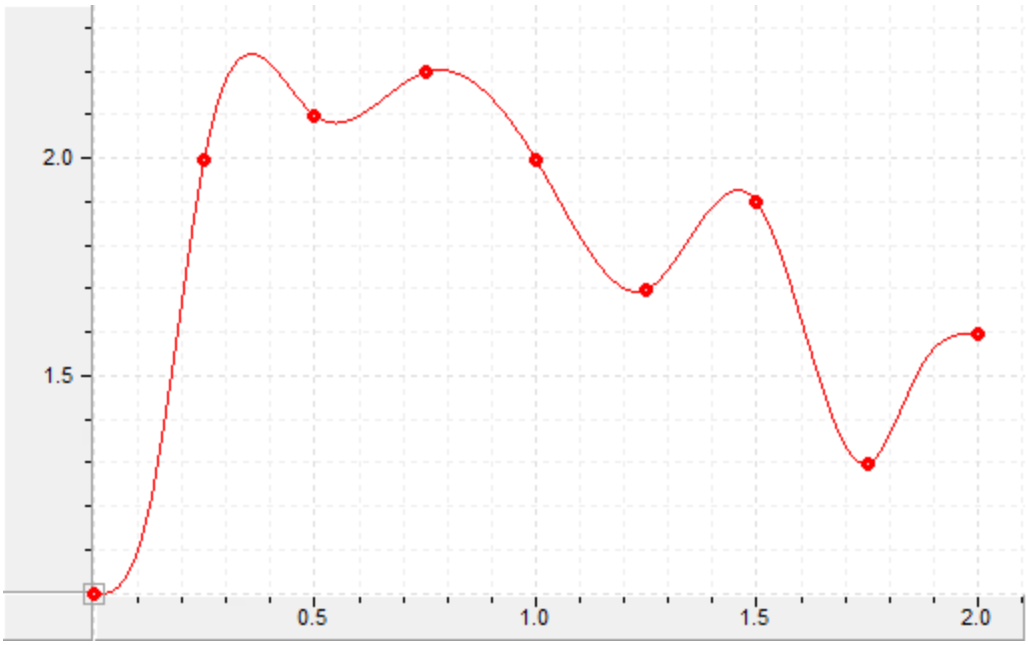
This procedure is for curves small enough to fit in the Variable Table, and the points are spaced evenly on the X-axis (or time axis). It can be used for position, pressure, or force, and can be used for a time-based curve or a master-based curve.

This example will create an example curve with nine points. The X-axis (or time axis) points are evenly spaced, and the end points have zero velocity (the axis will be stopped at the ends).

1. **Determine the Curve Points**

   Determine the distance or time between points on the X-axis (or time axis), and determine each point on the Y-axis. This example will use a time between points of 0.25, and nine Y-axis points as shown below:

X-axis (time)	0.00	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2.00
Y-axis	1.0	2.0	2.1	2.2	2.0	1.7	1.9	1.3	1.6



When determining the curve points, keep in mind that the axis that follows the curve must be at the starting Y-axis position before the Curve Start command is issued. That is, the first point is where the axis must be at the start.

2. Create the Curve Data in the Variable Table

Several different curve data formats are available, as described in Curve Data Formats. This procedure uses the simplest format: **Single Curve - Evenly-Spaced Points:**

Offset	Description
0	Curve Status (set to 0, the RMC will change this value)
1	Curve Format (0 = Single Curve - Evenly-Spaced)
2	Number Points
3	Interpolation Options (set to 0 for none)
4	Point 0 x value (set to 0 for time, or the starting master x position for master-based)
5	x interval (distance between x points)
6	y0 (first y position)
7	y1 (second y position)
:	:
5+*n*	y(*n*-1), *n* = number of points (*n*th y position)

Place the curve data anywhere in the Variable Table. The curve data for this example starts at F56:10. Adding descriptions helps keep track of what the data is for. By entering this data into the initial values of the Variable Table, the curve data can be saved to flash. Notice that the curve in the Curve Store cannot be saved to Flash, but the curve data in the initial values in the Variable Table can be. Optionally, you can set all the curve data variables to Retained, and they will be remembered between power cycles without requiring a Flash update.

Variable Table:

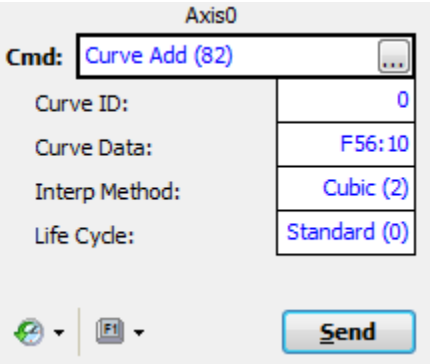
	Register	Tag Name	Units	Type	Retain	Initial	Description
10	F56:10			REAL	☐	0.0	Curve Status
11	F56:11			REAL	☐	0.0	Format 0 = single curve, evenly spaced
12	F56:12			REAL	☐	9.0	Number of points
13	F56:13			REAL	☐	0.0	Interpolation options
14	F56:14			REAL	☐	0.0	Point 0 x value
15	F56:15			REAL	☐	0.25	x interval
16	F56:16			REAL	☐	1.0	y0
17	F56:17			REAL	☐	2.0	y1
18	F56:18			REAL	☐	2.1	y2
19	F56:19			REAL	☐	2.2	y3
20	F56:20			REAL	☐	2.0	y4
21	F56:21			REAL	☐	1.7	y5
22	F56:22			REAL	☐	1.9	y6
23	F56:23			REAL	☐	1.3	y7
24	F56:24			REAL	☐	1.6	y8

You can enter the curve data in the Variable Table using RMCTools, or you can write the data from a PLC or HMI. If you use a PLC or HMI, it is a good idea to first enter the Descriptions in the Variable Table using RMCTools so that you can easily troubleshoot it.

3. Send the Curve Add (82) Command

Send the Curve Add (82) command to instruct the RMC to create a curve using the data in the Variable Table and the Curve ID specified by the command.

For the example curve, send the following command:



This command tells the RMC that the Curve ID will be zero (0), and the curve data starts at F56:10. The interpolation method is cubic (the most common method), and the life cycle **Standard** means you can run the command as many times as you would like. When you send this command, the curve will be placed in the Curve Store with an ID of 0.

Notice if you wish to be able to save the curve to Flash, you should choose the **Permanent** Life Cycle option. Only curves created with the Permanent Life Cycle option or downloaded from the Curve tool can be saved to Flash.

After the Curve Add (82) command has been added the curve, the data in the Variable Table is no longer needed. You can, for example, use it to add additional curves.

4. Wait for the Curve Status to be 3

Wait for the Curve Status to be 3. This indicates that the RMC has completed processing the curve data and has created the curve. The Curve Status is the first item you entered in the curve data in the Variable Table. Notice that you will need to look in the Monitor tab of the Variable Table Editor to see the real-time value.

If the Curve Status changes to a value of 10 or greater, it indicates an error. See the Curve Status Error Codes topic for a list of possible error values.

If you are using the Curve Add (82) command in a user program, you should make a Link Type that waits for the Curve Status to be 3 before continuing. The Curve Status will be the Current Value of the first variable in the curve data you entered in the variable table. In this example, it would be variable 10. You can assign a tag name to variable 10 to make it easier to find.

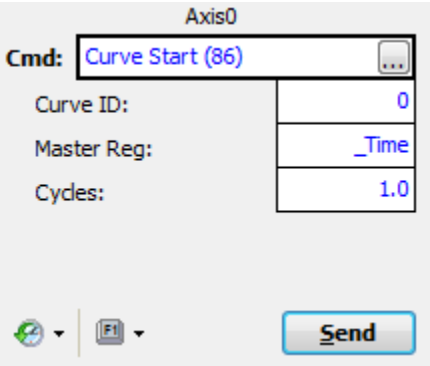
5. Move the axis to the first point.

Before you can follow the curve, the Target Position (Target Pressure or Force) of the axis must be at the first Y-axis position of the curve, which is 1.0 in this example. This can be done by using a Move Absolute command to move the axis to 1.0. For following a pressure or force curve, you will need to move the Target Pressure/Force to the first point's Y value.

Notice that you can move the axis to the first point before the Curve Status is 3.

6. Send the Curve Start Command

To start following a curve, send the Curve Start (86) command for a position curve, or the Curve Start (Prs/Frc) (87) command for a pressure or force curve. If the axis is not at the first point of the curve, an error will result and the curve will not start.



This command will start curve 0, will follow it based on time, and will follow it once.

For more details, see the Creating Curves Using the Curve Add Command and Curves Overview topics.

Tips for Using Curves from a PLC or HMI

For more efficient communication, you can make a user program that issues the Curve Add and Curve Start commands. Then the host controller need only start a user program, instead of issuing several commands. After writing the data to the Variable Table, start the user program. After the user program issues the Curve Add command, it must wait for the Curve Status to be 3 before issuing the Curve Start command.

See Also

Curves Overview | Curve Tool Overview | Curve Start (86) | Curve Start (Prs/Frc) (87) | Curve Interpolation Methods and Options | Creating Curves Using the Curve Add Command | Creating Large Curves Using the Curve Add Command | Curve Data Formats

3.7. Pressure and Force Control

3.7.1. Pressure/Force Control Overview

The RMC can handle a wide range of pressure, force, or torque applications with ease. Torque control is identical to pressure or single-input force control.

The major features of the RMC pressure/force control are:

- **Pressure/Force Control**
  Controlling pressure or force only using a pressure/force PID
- **Pressure/Force Limit**
  Limit pressure or force during the position or velocity motion of an axis.
- **Position to Pressure or Force Transition**
  Smoothly transition from position or velocity control to pressure or force control. See Position-Pressure and Position-Force Control for more details.
- **Input Types**
  - **Pressure** (±10V or 4-20mA)
  - **Single-input Force** (±10V or 4-20mA)
    For example, a load cell via a signal conditioner.
  - **Single-input Force** (mV/V)
    A directly connected load cell.
  - **Dual-input (Differential) Force** (±10V or 4-20mA)
    For example, two pressure transducers mounted on either end of a hydraulic cylinder.
- **Available Pressure/Force Target Profiles**
  - **Linear Ramps**
    Ramp Pressure/Force (Linear) (42), Ramp Pressure/Force (Rate) (18)
  - **S-Curve Ramps**
    Ramp Pressure/Force (S-Curve) (41)
  - **Sine wave**
    Sine Start (Prs/Frc) (76)
  - **Splines and Cams**
    Curve Start (Prs/Frc) (87)
    Curve Start Advanced (Prs/Frc) (89)
  - **Gearing**
    Gear Absolute (Prs/Frc) (59)

For details on setting up, tuning, and controlling pressure or force, see the following topics:

- Controlling Only Pressure or Force
- Position-Pressure and Position-Force Control
- Pressure Limit or Force Limit

3.7.2. Controlling Only Pressure or Force

This topic describes how to perform pressure/force control on a pressure-only or force-only axis. Torque control is identical to pressure or single-input force control. For details on position-pressure or position-force axes, see the Position-Pressure and Position-Force Control topic.

Required Hardware

A pressure-only or force-only axis requires the following RMC hardware:

- **One Control Output**
- **Pressure or Force Input, which may be:**
  - **One Analog Input (±10 V, 0-10 V, 4-20 mA, 0-20 mA)**
    A pressure sensor input or a load cell signal conditioner.
  - **Two Analog Inputs (±10 V, 0-10 V, 4-20 mA, 0-20 mA)**
    Two pressure sensor inputs for measuring the pressure on each end of a cylinder for resultant force.
  - **Load Cell Input (mV/V)**
    A directly connected mV/V load cell.

Note: For pressure-only or force-only control, any analog input will suffice. The RMC75 *does not* require the AP2 module for pressure-only or force-only axes. The RMC150E *does not* require the pressure option (indicated as RMC151) for pressure-only or force-only axes. These are only required for dual-loop axes, such as position-pressure or position-force.

Pressure or Force feedback requires analog feedback transducers (voltage or current) or load cells with mV/V signals. The input range from the transducer must fall within the RMC supported ranges of ±10V or 4-20mA, or 5mV/V. For the best resolution, the feedback should use a large portion of the input range.

Define the Axis

The first step for pressure/force control is to define the pressure/force axis. This means assigning the physical inputs and analog Control Output to an axis in RMCTools.

1. Open the Axis Definitions dialog to define the axes.
2. In order to define the axes as you want them, you may need to first remove the existing axes and add new axes. Or, you may be able to change an existing axis.
3. Whether you choose to change an existing axis or add a new axis, set the axis to the following:
   - **Control Axis**
   - **1 Input**
   - **Input Type: Pressure**, **Force (single-input)**, or **Force (dual-input))**
     If you are using a single pressure transducer or load cell for the feedback, choose **Pressure** or **Force (single-input)**. If you are using two pressure transducers for differential force feedback, choose **Force (dual-input, diff.)**.
4. The Axis Definitions dialog will show you exactly which Control Output and analog input(s) on the RMC will be used for the axis. You can change them if you wish.
5. Click **OK** on the Axis Definitions dialog. If you are online, these changes will be applied to the controller. Make sure to update Flash and save your project.

Set Up the Pressure/Force Transducer Parameters

Make sure to set the following pressure/force parameters:

Analog Inputs (±10 V, 0-10 V, 4-20 mA, 0-20 mA)

- Input Type: Choose Voltage or Current.
- Positive Pressure/Force Limit and Negative Pressure/Force Limit: These specify the pressure/force in which the axis will be allowed to control in. If the Target Pressure/Force exceeds these values, an error bit will be set and will halt the axis if the AutoStops are set to do so.

Load Cell Inputs (mV/V)

- **Wire Sense**
  The LC8 offers two methods of compensating for the voltage drop in in the Exciter+ and Exciter- wires:
  - **Adaptive:**
    The LC8 periodically measures the voltage at the Sense pin and determines the voltage drop. It assumes the voltage drop of the Exc+ and Exc- wires is the same. Therefore, the wire length and gauge of the Exc+ and Exc- wires must be identical.

    Available only for 6-wire load cells.

    To choose this option, set the Exciter Mode axis parameter to **Adaptive**.
  - **Fixed Value:**
    The user can use a voltmeter to measure the voltage at the load cell and enter the value into the Fixed Exciter Voltage axis parameter.

    This option is available for 4-wire and 6-wire load cells.

    To choose this option, set the **Exciter Mode** axis parameter to **Fixed Value**.

- Positive Pressure/Force Limit and Negative Pressure/Force Limit: These specify the pressure/force in which the axis will be allowed to control in. If the Target Pressure/Force exceeds these values, an error bit will be set and will halt the axis if the AutoStops are set to do so.
- The following axis parameters may also need to be set, especially if external excitation is used:
  - Exciter Mode
  - Load Cell Overflow Limit
  - Load Cell Underflow Limit

Scale the Pressure/Force

Scaling the feedback converts it from volts or current or mV/V to useful units such as pounds, newtons, etc. To scale the feedback, use the Pressure/Force Scale/Offset Wizard.

1. In the Axis Tools, in the Axis Parameters Pane, on the **Setup** tab, in the **Tools and Wizards** section, in the axis column you are using, click **Launch** to open the Pressure/Force Scale/Offset Wizard.
2. Complete the wizard, then download the parameters. Make sure to update Flash and save your project.
3. After scaling, move the axis and apply pressure/force and verify that the correct Actual Pressure or Actual Force is displayed in the Axis Status Registers in Axis Tools.

Tune the Pressure/Force

See Tuning a Pressure/Force Axis.

Enter Pressure or Force Control

To enter pressure or force control, issue one of the following commands:

- Hold Current Pressure/Force (19)
  This command will enter pressure/force control and hold the current pressure/force . This command is a good choice if your axis is stopped or moving slowly.
- Enter Pressure/Force Control (Time) (45)
  This command will enter pressure/force control and then ramp to the requested pressure/force within the specified time.
- Enter Pressure/Force Control (Auto) (44)
  This command will enter pressure/force control and ramp to the requested pressure/force. The ramp is automatically calculated based on the current rate of change of the Actual Pressure or Force. This command is designed to be used when the system is moving and the Actual Pressure or Force is rising quickly, which indicates the axis is encountering the resisting pressure/force. This command can provide a very smooth transition from position to pressure control while the system is moving.
- Enter Pressure/Force Control (Rate) (46)
  This command will enter pressure/force control and then ramp to the requested pressure/force at the requested rate and acceleration.

When the axis is in pressure/force control, the Pressure/Force Control status bit will be set.

Controlling Pressure or Force

Once the axis is in pressure/force control, you can issue any of the pressure/force commands, including the commands that enter pressure/force control. For example, you can ramp the pressure up or down, perform a sinusoidal profile, or follow a spline.

For details on each command, see the respective help topics.

- Ramp Pressure/Force (Rate) (18)
- Hold Current Pressure/Force (19)
- Ramp Pressure/Force (S-Curve) (41)
- Ramp Pressure/Force (Linear) (42)
- Stop Pressure/Force (43)
- Enter Pressure/Force Control (Auto) (44)
- Enter Pressure/Force Control (Time) (45)
- Enter Pressure/Force Control (Rate) (46)
- Sine Start (Prs/Frc) (76)
- Sine Stop (Prs/Frc) (77)
- Change Target Parameter (Prs/Frc) (81)
- Curve Start (Prs/Frc) (87)
- Curve Start Advanced (Prs/Frc) (89)

Exiting Pressure or Force Control

To exit pressure/force control, issue the Open Loop Rate (10) or Direct Output (9) command. An Open Loop Halt or Direct Output Halt will also cause the axis to exit pressure/force control.

Pressure/Force Limit

Pressure/Force Limit can be used on a pressure-only or force-only axis, although it is typically not very useful. Pressure/force limit is most useful on axes that also have position or velocity feedback. On pressure/force axes, pressure/force limit can only limit the pressure/force while moving the axis with the Open Loop Rate (10) command.

For example, consider an axis on which force limit is enabled. If the is moving with an open loop Control Output of 2 volts, and the Target Force is 300 lbs, the axis will move with 2 volts Control Output until the Actual Force reaches 300 lbs. Then, the Control Output will be limited so that the force will not exceed 300 lbs.

Notice that with a large open loop voltage, a pressure/force limited axis will behave similarly to an axis in pressure/force control. That is, the pressure/force can be controlled very accurately, even with the specialty pressure/force commands, such as a sine wave (using the Sine Start (Prs/Frc) (76) command) or a cam (using the Curve Start (Prs/Frc) (87) command).

Pressure/force limit cannot be used simultaneously with pressure/force control. If the axis is in pressure/force limit, and a command is issued that puts the axis in pressure/force control, the pressure/force limit will no longer be active and the axis will start controlling the pressure/force. However, the Pressure/Force Limit Enabled status bit will still be set, and if the axis exits pressure/force control, it will resume pressure/force limit.

Pressure/Force Control Status Bits

The following status bits give the status of the pressure/force control. These tell you what the control is doing, and can also be very useful in user programs.

At Pressure/Force

When the axis is in Pressure/Force control or Pressure/Force Limit and the Target Pressure/Force reaches the **Requested Pressure/Force** and the Actual Pressure/Force is within the At Pressure/Force Tolerance window from the Target Pressure/Force, the At Pressure/Force Status bit will be set. This bit indicates that the ramp is complete and the axis is at the pressure/force.

Pressure/Force Control

The Pressure/Force Control Status bit indicates that the axis is in closed-loop pressure/force control.

Pressure/Force Limit Enabled

This bit indicates that pressure/force limit is enabled. If an axis is in pressure/force control, pressure/force limit will not be active. If the axis exits pressure/force control to open loop or closed-loop position or velocity control, it will enter pressure/force limit if the Pressure/Force Limit Enabled bit is on.

Pressure/Force Limited

This bit indicates that pressure/force limit is enabled and the axis is limiting the pressure/force. If an axis is in pressure/force control, pressure/force limit will not be active and this bit will not be set.

Pressure/Force Target Generator Bits

The Pressure/Force Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Pressure/Force Target Generator Done bit

This bit indicates that the Target Pressure or Target Force has reached the **Requested Pressure/Force**. If the ramp is interrupted, e.g. due to a halt, the done bit will not be set because the commanded motion was not completed. Notice that this bit does not indicate whether the Actual Pressure or Actual Force has reached the **Requested Pressure/Force**.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Pressure/Force is accelerating
1	0	Constant
1	1	Pressure/Force is decelerating

See Also

Pressure/Force Control Overview

3.7.3. Position-Pressure and Position-Force Control

This topic describes position-pressure and position-force control. For information on pressure control only or force control only, see the Pressure/Force Limit topic.

Pressure/Force Control or Pressure/Force Limit can be used on position-pressure or position-force control axes. Position-torque is identical to or position-force with a single-input force, or position-pressure.

Required Hardware

A position-pressure or position-force axis requires the following RMC hardware:

- **Pressure-Enabled Controller**
  This is the term for an RMC that supports dual-loop control, such as position-pressure or position-force.
  - **RMC75:** Requires the AP2 module.
  - **RMC150E:** Requires the pressure option (denoted as RMC151). This must be specified when ordering the RMC.
  - **RMC200:** Requires two Feature Key control loops per dual-loop control axis.
- **One Control Output**
- **One Position Input**
  This can be any type of position feedback supported by the RMC.
- **Pressure or Force Input, which may be:**
  - **One Analog Input (±10 V, 0-10 V, 4-20 mA, 0-20 mA)**
    A pressure sensor input or a load cell signal conditioner.
  - **Two Analog Inputs (±10 V, 0-10 V, 4-20 mA, 0-20 mA)**
    Two pressure sensor inputs for measuring the pressure on each end of a cylinder for resultant force.
  - **Load Cell Input (mV/V)**
    A directly connected mV/V load cell.

Pressure or Force feedback requires analog feedback transducers (voltage or current) or load cells with mV/V signals. The input range from the transducer must fall within the RMC supported ranges of ±10V or 4-20mA, or 5mV/V. For the best resolution, the feedback should use a large portion of the input range.

Define the Axis

The first step for pressure/force control is to define the axis. This means assigning the physical inputs and analog Control Output to an axis in RMCTools.

1. Open the Axis Definitions dialog to define the axes.
2. In order to define the axes as you want them, you may need to first remove the existing axes and add new axes. Or, you may be able to change an existing axis.
3. Whether you choose to change an existing axis or add a new axis, set the axis to the following:
   - **Control Axis**
   - **2 Inputs**
   - **First Input:** Select your position input type.
   - **Input Type: Pressure**, **Force (single-input)**, or **Force (dual-input)**
     If you are using a single pressure transducer or load cell for the feedback, choose **Pressure** or **Force (single-input)**. If you are using two pressure transducers for differential force feedback, choose **Force (dual-input, diff.)**.

4. The Axis Definitions dialog will show you exactly which Control Output and analog input(s) on the RMC will be used for the axis. You can change them if you wish.
5. Click **OK** on the Axis Definitions dialog. If you are online, these changes will be applied to the controller. Make sure to update Flash and save your project.

Set Up and Tune the Position

Set up and tune the position control first. Refer to the Startup Procedure for details.

While tuning the position, you may need to set some of the pressure/force AutoStops to Status Only to prevent any pressure/force errors from halting the axis.

Set Up the Pressure/Force Transducer Parameters

Make sure to set the following pressure/force parameters:

Analog Inputs (±10 V, 0-10 V, 4-20 mA, 0-20 mA)

- Input Type: Choose Voltage or Current.
- Positive Pressure/Force Limit and Negative Pressure/Force Limit: These specify the pressure/force range within which the axis will be allowed to control. If the Target Pressure/Force exceeds these values, an error bit will be set and will halt the axis if the AutoStops are set to do so.

Load Cell Inputs (mV/V)

- **Wire Sense**
  The LC8 offers two methods of compensating for the voltage drop in in the Exciter+ and Exciter- wires:
  - **Adaptive:**
    The LC8 periodically measures the voltage at the Sense pin and determines the voltage drop. It assumes the voltage drop of the Exc+ and Exc- wires is the same. Therefore, the wire length and gauge of the Exc+ and Exc- wires must be identical.

    Available only for 6-wire load cells.

    To choose this option, set the Exciter Mode axis parameter to **Adaptive**.
  - **Fixed Value:**
    The user can use a voltmeter to measure the voltage at the load cell and enter the value into the Fixed Exciter Voltage axis parameter.

    This option is available for 4-wire and 6-wire load cells.

    To choose this option, set the Exciter Mode axis parameter to **Fixed Value**.

- Positive Pressure/Force Limit and Negative Pressure/Force Limit: These specify the pressure/force range within which the axis will be allowed to control. If the Target Pressure/Force exceeds these values, an error bit will be set and will halt the axis if the AutoStops are set to do so.
- The following axis parameters may also need to be set, especially if external excitation is used:
  - Exciter Mode
  - Load Cell Overflow Limit
  - Load Cell Underflow Limit

Scale the Pressure/Force

Scaling the pressure or force feedback converts it from volts or current or mV/V to useful units such as pounds, newtons, etc. To scale the feedback, use the Pressure/Force Scale/Offset Wizard.

1. In the Axis Tools, in the Axis Parameters Pane, on the **Setup** tab, in the **Tools and Wizards** section, in the axis column you are using, click **Launch** to open the Pressure/Force Scale/Offset Wizard.

2. Complete the wizard, then download the parameters. Make sure to update Flash and save your project.
3. After scaling, move the axis and apply pressure or force and verify that the correct Actual Pressure or Actual Force is displayed in the Axis Status Registers in Axis Tools.

Tune the Pressure/Force

See Tuning a Position-Pressure System.

Before tuning the pressure or force, you should read the rest of this topic to become familiar with the pressure/force commands that will be used during tuning.

If the pressure/force is scaled in small units, such as psi, pounds, Pascal, or Newtons, the gains will probably be very small. For example, you may want to start with a value of 0.01 for the Proportional and Integral Gain, and even smaller for the Differential Gain.

You can tune the pressure or force in either pressure/force control or pressure/force limit, and it will control in either mode. For many applications, it is easiest to first tune the axis in pressure control, even if you will be using pressure/force limit in your application.

Choose Pressure/Force Control or Pressure/Force Limit

At this point you should decide whether you will be using pressure/force control or Pressure/Force Limit. With pressure/force control, the axis transitions from position control to pressure/force control, and the pressure/force control is then independent of the position control.

Pressure/force limit is a special type of pressure or force control. With pressure or force limit, the pressure or force is limited during the position or velocity motion of an axis.

Pressure/force limit cannot be used simultaneously with pressure/force control. If the axis is in pressure/force limit, and a command is issued that puts the axis in pressure/force control, the pressure/force limit will no longer be active and the axis will start controlling the pressure or force. However, the Pressure/Force Limit Enabled status bit will still be set, and if the axis exits pressure/force control, it will resume pressure/force limit.

Determining Whether Pressure/Force Limit is Required

- If you want to transition between position and pressure or force control, and each control mode does not depend on the other, then you probably do not need pressure or force control.
- If you want to transition between position and pressure or force control, but you want to make sure the axis will not exceed a certain position while controlling pressure/force, you probably need pressure/force limit. However, you can use pressure/force control and monitor the other values from a user program to handle error conditions.
- If you want to perform position or velocity motion, but limit the pressure or force during the motion, then you probably need pressure or force limit.

If you will be using pressure/force limit, see the Pressure/Force Limit topic. If you will be using pressure/force control, continue reading this topic.

Enter Pressure or Force Control

To enter pressure or force control, issue one of the following commands. Notice that these commands will enter pressure/force control, not pressure/force limit.

- Hold Current Pressure/Force (19)
  This command will enter pressure or force control and hold the current pressure or force. This command is a good choice if your axis is stopped or moving slowly.
- Enter Pressure/Force Control (Time) (45)
  This command will enter pressure or force control and then ramp to the requested pressure or force within the specified time.
- Enter Pressure/Force Control (Auto) (44)
  This command will enter pressure or force control and ramp to the requested pressure or force. The ramp is automatically calculated based on the current rate of change of the Actual Pressure or Force. This command is designed to be used when the system is moving and the Actual Pressure or Force is rising quickly, which indicates the axis is encountering the resisting

pressure or force. This command can provide a very smooth transition (bumpless transfer) from position to pressure control while the system is moving.

- Enter Pressure/Force Control (Rate) (46)
  This command will enter pressure/force control and then ramp to the requested pressure/force at the requested rate and acceleration.

When the axis is in pressure or force control, the Pressure/Force Control status bit will be set.

Controlling Pressure or Force

Once the axis is in pressure or force control, you can issue any of the pressure/force commands, including the commands that enter pressure/force control. For example, you can ramp the pressure up or down, or perform a sinusoidal profile, or follow a spline.

For details on each command, see the respective help topics.

- Ramp Pressure/Force (Rate) (18)
- Hold Current Pressure/Force (19)
- Ramp Pressure/Force (S-Curve) (41)
- Ramp Pressure/Force (Linear) (42)
- Stop Pressure/Force (43)
- Enter Pressure/Force Control (Auto) (44)
- Enter Pressure/Force Control (Time) (45)
- Sine Start (Prs/Frc) (76)
- Sine Stop (Prs/Frc) (77)
- Change Target Parameter (Prs/Frc) (81)
- Curve Start (Prs/Frc) (87)
- Curve Start Advanced (Prs/Frc) (89)

Exiting Pressure or Force Control

To exit pressure or force control, issue an open-loop or closed-loop motion command. An Open Loop Halt or Direct Output Halt will also cause the axis to exit pressure or force control. A Closed Loop Halt will not exit pressure or force control.

Releasing a Pressure or Force

Exiting pressure or force control when the actuator is exerting a force against a dead stop may require careful consideration, especially if it is critical that the force does not overshoot.

One method is to issue the Hold Current Position (5) command. However, the axis will now be holding position, and since the Actual Position typically jumps around a bit, it may cause an undesired increase in the force. Therefore, this may not be a good choice.

Another option is to issue a Move Absolute (20) to move back from the current position. However, the Actual Velocity typical jumps around a bit, and if it moving in the direction of increasing force at the instant the Move Absolute command is issued, then the Target Position must first turn around, meaning that the force may actually increase before decreasing.

Sometimes, the Open Loop Rate (10) command may work best because it alleviates both of the aforementioned problems. The Quick Move Absolute (15) will also work well, with the added benefit that it moves to a known position.

Pressure/Force Control Status Bits

The following status bits give the status of the pressure or force control. These tell you what the control is doing, and can also be very useful in user programs.

At Pressure/Force

When the axis is in Pressure/Force control or Pressure/Force Limit and the Target Pressure/Force reaches the **Requested Pressure/Force** and the Actual Pressure/Force is within the At

Pressure/Force Tolerance window from the Target Pressure/Force, the At Pressure/Force Status bit will be set. This bit indicates that the ramp is complete and the axis is at the pressure or force.

Pressure/Force Control

The Pressure/Force Control Status bit indicates that the axis is in closed-loop pressure or force control.

Pressure/Force Limit Enabled

This bit indicates that pressure/force limit is enabled. If an axis is in pressure or force control, pressure/force limit will not be active. If the axis exits pressure/force control to open loop or closed-loop position or velocity control, it will enter pressure/force limit if the Pressure/Force Limit Enabled bit is on.

Pressure/Force Limited

This bit indicates that pressure/force limit is enabled and the axis is limiting the pressure/force. If an axis is in pressure or force control, pressure/force limit will not be active and this bit will not be set.

Pressure/Force Target Generator Bits

The Pressure/Force Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Pressure/Force Target Generator Done bit

This bit indicates that the Target Pressure or Target Force has reached the **Requested Pressure/Force**. If the ramp is interrupted, e.g. due to a halt, the done bit will not be set because the commanded motion was not completed. Notice that this bit does not indicate whether the Actual Pressure or Actual Force has reached the **Requested Pressure/Force**.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Pressure/Force is accelerating
1	0	Constant
1	1	Pressure/Force is decelerating

See Also

Pressure/Force Control Overview

3.7.4. Velocity-Pressure and Velocity-Force Control

The RMC supports velocity-pressure and velocity-force control. This means controlling both velocity and pressure or force with a single actuator. For example, an injection molding press may need to control the velocity of the injection, but also limit the force during the motion.

Setting up, tuning and performing velocity-pressure or velocity-force is virtually identical to position-pressure or position-force. See the Position-Pressure and Position-Force Control for details and procedures.

See Also

Pressure/Force Control Overview

3.7.5. Pressure/Force Limit

Pressure limit or force limit is a special type of pressure or force control, typically used on position-pressure or position-force axes. Pressure/Force Limit limits the Actual Pressure or Force during position or velocity motion. Pressure/Force Limit is useful in applications that require moving to a position while not exceeding a certain pressure or force during the move.

For example, consider a vehicle manufacturing application that presses a bearing into a steering knuckle. The bearing must be pressed to a final position, but the force during the move cannot exceed a certain value. Force limit is an excellent way to ensure that the move occurs as quickly as possible without damaging the bearing due to excessive force.

You can choose to apply Pressure/Force Limit in the positive or negative directions, or both. The Target Pressure/Force (or the negated Target Pressure/Force for the negative direction) is the value at which the Actual Pressure or Force will be limited. The axis position or velocity can be controlled normally as long as the Actual Pressure or Force does not approach the limit. As the Actual Pressure or Force approaches the threshold, the RMC will limit the motion so that the pressure/force limit is not exceeded. For more advanced details, see the Pressure/Force Limit Details topic.

Pressure/Force Limit cannot be used in Direct Output. To issue the Set Pressure/Force Limit Mode (40) command, the Direct Output Status bit must be off. To turn off the Direct Output Status bit, put the axis in Open Loop or Closed Loop control.

Note:
Pressure Limit mode may affect motion even when the Actual Pressure is well below the pressure limit. In order to achieve precise motion when pressure is not important, do not enable Pressure Limit mode. This may require the user to enable Pressure Limit mode only after the pressure has increased close to the point where the pressure is to be limited.

Pressure/Force Limit versus Control

Pressure/Force Limit differs from Pressure/Force Control. In Pressure/Force Control, the axis is controlling pressure or force independent of the position or velocity motion on the axis. In Pressure/Force Limit, the axis is performing position or velocity control while limiting the pressure or force.

Determining Whether Pressure/Force Limit is Required

Pressure/force Limit is usually more complicated than just pressure/force control. Therefore, you should not use pressure/force limit unless you need it.

- If your application is only controlling pressure or force, then you probably do not need to use pressure or force limit.
- If you want to transition between position and pressure or force control, and each control mode does not depend on the other, then you probably do not need pressure or force limit.
- If you want to transition between position and pressure or force control, but you want to make sure the axis will not exceed a certain position while controlling pressure or force, you probably need pressure/force limit. However, you can use pressure or force control and monitor the other values from a user program to handle error conditions.
- If you want to perform position or velocity motion, but limit the pressure or force during the motion, then you probably need pressure or force limit.

Using Pressure/Force Limit

Setting Up the Axis

Pressure/Force Limit is most useful on dual-loop axes, such as position-force. For details on setting up a position-pressure or position-force axis, see Position-Pressure and Position-Force Control. For details on setting up a pressure-only or force-only axes, see Controlling Only Pressure or Force.

Entering Pressure/Force Limit

Pressure/Force Limit requires first moving the axis in open loop or closed loop in order for the RMC to limit the pressure. The RMC will limit the Control Output to prevent the pressure/force from exceeding the Target Pressure/Force. Once the pressure/force is being limited the Target Pressure/Force can be changed using any of the pressure/force commands, except the commands that enter pressure/force control, because this will exit pressure/force limit

After setting up the axis for pressure or force control, follow the steps below to limit pressure/force. In order to limit the pressure or force, you must first tune it. See Tuning Overview topic for details.

1. Issue a pressure/force command (such as Ramp Pressure/Force (Linear) (42)) to set the Command Pressure to the desired value.

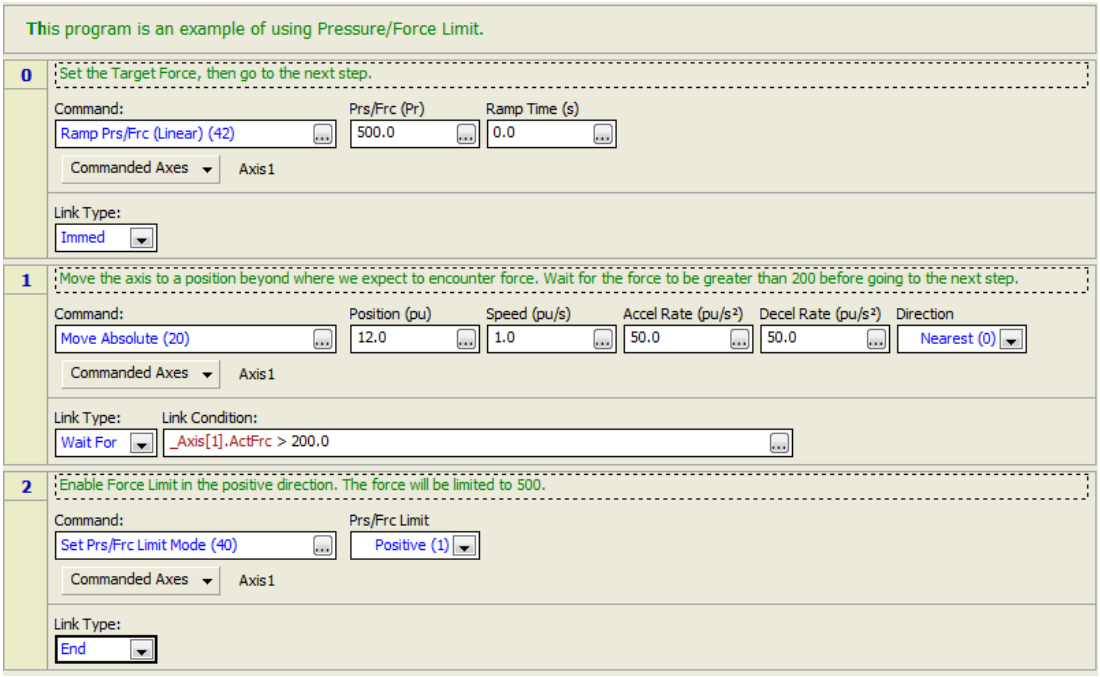
> **Note:**
> After starting up the RMC, the Target Pressure/Force must be set up with a pressure/force command before issuing the Set Pressure/Force Limit Mode (40) command for the first time, or a Command Error will result.

2. You can move the axis in position or velocity control normally until you want to enter pressure/force limit.
3. When you wish to start limiting the pressure/force, issue the Set Pressure/Force Limit Mode (40) command to enable pressure/force limit. Pressure/Force Limit mode may affect normal closed-loop motion even when the pressure is very low. Therefore, if possible, do not enter Pressure/Force Limit until you need to.
4. Move the axis. When the motion begins to affect the pressure, the RMC will limit the motion such that the Actual Pressure/Force does not exceed the Command Pressure/Force. To make an axis go to a certain pressure/force, the axis must be commanded to move to a point at or beyond the point where the pressure/force limit is reached.

Use a User Program

It is often advantageous to make a User Program to handle the steps above. For example, one step waits until the pressure reaches a certain level, and then goes to the next step that enables pressure limit.

Example:



Changing the Target Pressure/Force

Once the axis is in pressure or force control, you can issue any of the following pressure/force commands to change the pressure/force. For example, you can ramp the pressure up or down, or perform a sinusoidal profile, or follow a spline. For details on each command, see the respective help topics.

- Ramp Pressure/Force (Rate) (18)
- Ramp Pressure/Force (S-Curve) (41)
- Ramp Pressure/Force (Linear) (42)
- Stop Pressure/Force (43)
- Sine Start (Prs/Frc) (76)
- Sine Stop (Prs/Frc) (77)
- Change Target Parameter (Prs/Frc) (81)
- Curve Start (Prs/Frc) (87)
- Curve Start Advanced (Prs/Frc) (89)

DO NOT issue the pressure/force commands that enter pressure/force control, because this will exit pressure/force limit:

- Hold Current Pressure/Force (19)
- Enter Pressure/Force Control (Auto) (44)
- Enter Pressure/Force Control (Time) (45)

For details on control issues to be aware of, see the Pressure/Force Limit Details topic.

Exiting Pressure/Force Limit

An axis will exit pressure/force limit mode if any of the following occurs:

- The **Set Pressure/Force Limit Mode** command is sent with the **Pressure/Force Limit** command parameter set to 0 (Disabled).
- A Direct Output (9) command is sent to the axis.
- An Open Loop Halt or Direct Output Halt occurs on the axis.
- The Fault Controller (8) command is issued to the RMC.

Switching Between Pressure/Force Limit and Control

Pressure/force limit cannot be used simultaneously with pressure/force control. If the axis is in pressure/force limit, and a command is issued that puts the axis in pressure/force control, the pressure/force limit will no longer be active and the axis will start controlling the pressure or force. However, the Pressure/Force Limit Enabled status bit will still be set, and if the axis exits pressure/force control, it will resume pressure/force limit.

See Also

Pressure/Force Control Overview | Pressure/Force Limit Details

3.7.6. Pressure/Force Limit Details

This topic provides more in-depth information on the pressure/force limit. For general information, and a step-by-step procedure, see the Pressure/Force Limit topic.

Pressure/Force Limit with Closed Loop Moves

When a closed-loop position move is made on the axis with pressure/force limit enabled, the axis will move in position control until the Actual Pressure or Force comes close to the Target Pressure/Force, at which point the position move will be limited such that the pressure or force will be limited to the Target Pressure/Force.

The Limit algorithm works as follows:

1. Each loop time, for both the position and the pressure/force, the RMC calculates the total of the Proportional, Differential, and Feed Forward terms and finds the smaller total.
2. The appropriate integral term is added to the smaller total to calculate the Control Output.

Therefore, the axis is performing both PIDs simultaneously and using the smaller one such that the pressure or force will be limited to the Target Pressure/Force.

Position-Pressure Interplay

If the position of the axis varies a lot as the pressure or force changes, the closed loop motion PID and the pressure PIDs may affect each other. That is, tuning the pressure alone will not make the axis control better. Both the position and pressure PID must be tuned to properly control the combined motion.

Pressure Limit may affect position moves even though Actual Pressure is far from the Target Pressure. If you wish to make position moves regardless of the pressure, verify that the Pressure Limit mode is off, or the motion may be affected.

Pressure/Force Limit with Open Loop Moves

This type of control can be done on any control axis with pressure feedback, whether or not the axis also has position feedback. The algorithm in the open-loop case works the same as in the Closed Loop Move case, except that the Output of the pressure PID is compared to the open-loop Control Output. The smaller of these values is then used for the Control Output.

Pressure/force limit with open loop is typically not very useful, as standard pressure/force control is easier and generally provides better control.

Using Pressure Limit to Achieve Set Pressure Control

The Pressure/Force Limit algorithm can be used to achieve Set Pressure Control. That is, controlling the pressure/force of the axis, regardless of position. Use one of the following methods:

With Closed-Loop Motion

1. Make a move that is far beyond the position where the pressure will be limited.
2. Use the Ramp Pressure/Force (S-Curve) (41) or Ramp Pressure/Force (Linear) (42) command to change the pressure.

With Open-Loop Motion

This method is simpler than the above method because it does not require tuning the position, nor will the pressure PID affect the position PID.

1. Issue an Open Loop command with a Control Output that is higher than you will need for the pressure control.
2. Use the Ramp Pressure/Force (S-Curve) (41) or Ramp Pressure/Force (Linear) (42) command to change the pressure.

See Also

Pressure/Force Control Overview | Pressure/Force Limit

3.8. Filtering/Modeling

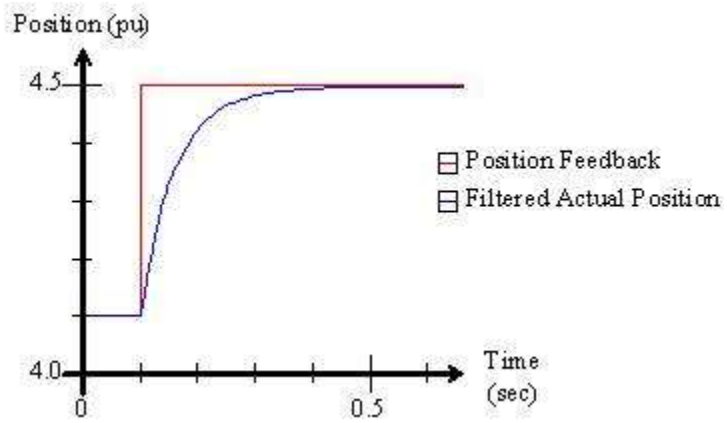
3.8.1. Filtering

Filtering can be used to smooth values. A noisy position feedback or Control Output signal will be smoother if filtering is applied.

Filtering increases the phase delay in the filtered value, meaning the signal becomes delayed. Phase delay on the feedback can be detrimental to control, especially if the system must respond quickly. Therefore, feedback filtering on control axes should be used only when necessary.

Filtering can be especially useful on reference axes. For example, consider a control axis that is geared to a reference axis. The reference axis is controlled by a potentiometer. When the operator turns the potentiometer, it produces a fairly noisy signal. The axis that is geared to this noisy signal will, in turn, produce jerky motion. To smooth the motion, a filter can be applied to the reference Actual Position.

For example, suppose the position feedback makes a step jump from 4.1 to 4.5 position units. With the position filter disabled (set to zero), the Actual Position would also make a step jump. By applying a filter, the Actual Position is filtered, and does not make a step jump.



The RMC offers filtering in the following places:

- **Feedback Filter (Input Filter)**
  The feedback inputs such as position, velocity, acceleration, pressure, force, etc. can be filtered. Filtering the feedback can improve control if the feedback is noisy or has quantization errors. However, feedback filtering *must be used with caution* to avoid phase delay.

  Feedback filtering may be applied to the values used by the control algorithm, or only to the values displayed to the user, or to both. By default, the velocity and acceleration display values are filtered since they normally have a lot of quantization error.

  The feedback filtering options differ between the RMC75/150 and the RMC200. See the **RMC75/150 Feedback Filtering** and **RMC200 Feedback Filtering** sections below.

- **Output Filter**
  The Output Filter filters the Control Output components contributed by the Proportional, Differential, and Double Differential gains. Filtering the output can overcome the negative effects of quantization error on the values derived from the feedback, for example, velocity and acceleration, or the pressure rate or force rate.

  The Output Filter can provide great improvement in control in conjunction with the Differential and Double Differential gains, especially for systems that tend to oscillate. For details on how to use the Output Filter, see the Tuning procedures, such as Tuning Position.

Filter Algorithm Types

The RMC's offer various filter algorithms, such as first order, Butterworth, ABG, etc. For a description of each filter types, and where each type may be applied in each RMC, see the Filter Algorithm Types topic.

RMC75/150 Feedback Filtering

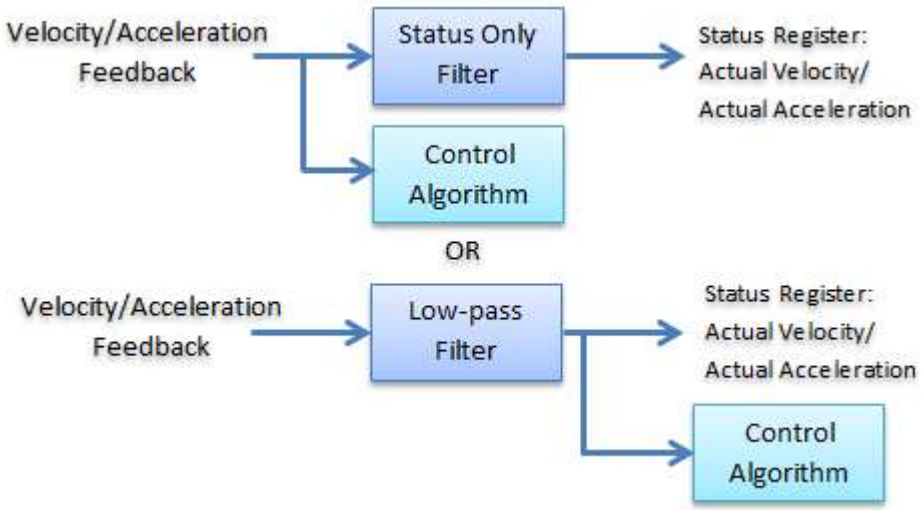
The RMC75 and RMC150 provide a 4-pole Butterworth filter for the feedback.

Position Filtering

The position filtering will apply to the value the control algorithm uses *and* to the value that is displayed.

Velocity and Acceleration Filtering

The velocity and acceleration filters can be selected to apply only to the displayed value, or to the value that the control algorithm uses, which will then be the displayed value as well.



To Apply Display Filtering:

Setting the display filtering means you are filtering only so that the values are easier to see on a plot, or so that a displayed number doesn't jump around so much.

1. Make sure the Velocity Filter Type and Acceleration Filter Type are set to **Status Only**.
2. Set the **Actual Velocity Filter** and/or Actual Acceleration Filter values to the desired cut-off frequencies:
    a. Start with a large value, such as 100 or 200 Hz.
    b. Decrease the number to apply heavier filtering. A lower number provides heavier filtering. A value of zero means the filter is off.
3. Do not use the Actual Position filter, since it always applies to the value used by the control algorithm.

To Apply Filtering on the Controlled Signal:

If the feedback is so noisy that it is causes problems for the control algorithm, you may need to filter it. The displayed value will be the same as that filtered value. The RMC75/150 use the 4-pole Butterworth that has a large phase delay, which can cause large oscillations if not carefully applied.

1. In the Axis Tools, on the **All** tab, in the **Feedback** or **Secondary Feedback** section, expand the **Filtering/Modeling** section.
2. Make sure the Velocity Filter Type and Acceleration Filter Type are set to **Low Pass**.
3. Set the Actual Position Filter, **Actual Velocity Filter** and/or Actual Acceleration Filter values to the desired cut-off frequencies:
    a. Start with a large value, such as 100 or 200 Hz.

b. Decrease the number to apply heavier filtering. A lower number provides heavier filtering. A value of zero means the filter is off.

Velocity Feedback

The velocity feedback filtering is identical to the position feedback filtering, except that the position filter is of course not available, and the Model is not available.

Pressure/Force Feedback

The RMC75 and RMC150 provide a 4-pole low-pass Butterworth filter for the pressure or force feedback as follows:

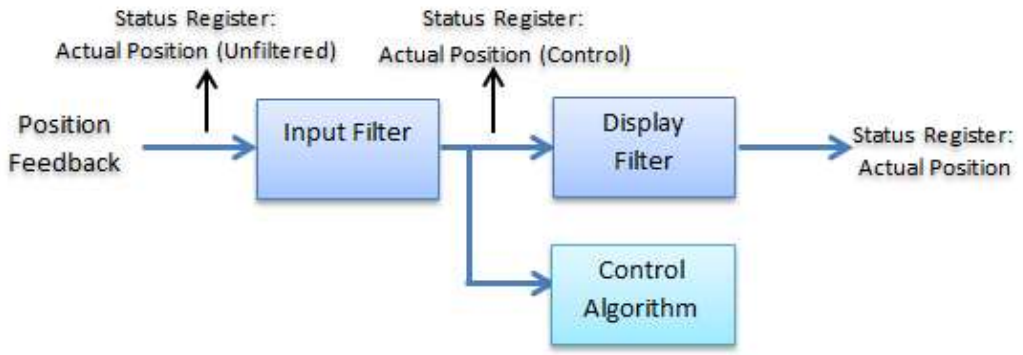
- **Pressure/Force Filter:** Applies to the pressure or force value used by the control algorithm.
- **Pressure/Force Rate Filter:** Applies to the displayed value only.

RMC200 Feedback Filtering

The RMC200 supports the following feedback filters:

Input Filter: Applies to the control algorithm and offers several filter algorithm types. The default setting for this filter is off.

Display Filter: Filters the value that has already been filtered by the input filter. This filter is a fourth-order Butterworth filter. The default setting is that the display filter is applied to velocity and acceleration.



Status Registers

The RMC200 Axis status registers for feedback include:

- The unfiltered feedback, such as Actual Position (Unfiltered).
- The filtered feedback values used by the control algorithm, such as Actual Position (Control).
- The final feedback value after the input filter and display filter have been applied, such as Actual Position.

These status registers are useful for troubleshooting and for determining the effectiveness of the filtering.

To Apply Display Filtering Only:

Setting the display filtering means you are filtering only so that the values are easier to see on a plot, or so that a displayed number doesn't jump around so much, and not affecting the feedback values used by the control algorithm.

1. In the Axis Tools, on the **All** tab, in the **Feedback** or **Secondary Feedback** section, expand the **Input Filter** section.
2. Set the Input Filter Type to **None**.
3. In the **Display Filter** section, set the Position Display Filter, Velocity Display Filter and/or Acceleration Display Filter values to the desired cut-off frequencies:
    a. Start with a large value, such as 100 or 200 Hz.

b. Decrease the number to apply heavier filtering. A lower number provides heavier filtering. A value of zero means the filter is off.

To Apply Filtering on the Controlled Signal:

If the feedback is so noisy that it is causes problems for the control algorithm, you may need to filter it. The RMC200 has many options for filtering the signal used for control.

1. In the Axis Tools, on the **All** tab, in the **Feedback** or **Secondary Feedback** section, expand the **Input Filter** section.
2. Set the Input Filter Type parameter to the desired filter.
3. Set the Position Input Filter, Velocity Input Filter, and/or Acceleration Input Filter values to the desired cut-off frequencies:
    1. Start with a large value, such as 100 or 200 Hz. For the ABG and ABGD filters, start with the maximum allowed value as described in the Filter Algorithm Types topic.
    2. Decrease the number to apply heavier filtering. A lower number provides heavier filtering. A value of zero means the filter is off.
4. In the **Display Filter** section, you may choose whether or not to also apply a display filter. To turn off the display filtering, set the Position Display Filter, Velocity Display Filter and/or Acceleration Display Filter values to zero.

Velocity Feedback

The velocity filtering is identical to the position filtering, except that the position filter is of course not available, and the Model is not available.

Pressure/Force Feedback

For pressure and force axes, the RMC200 provides the same options as for position axes, except that there is only an Actual Pressure/Force and an Actual Pressure/Force Rate, and the Model is not available.

See Also

Filter Algorithm Types | Modeling | Control Features Overview

3.8.2. Filter Algorithm Types

This topic describes the different types of filter algorithms in the RMC and lists where they can be applied. For more general information on filtering in the RMC, see the Filtering topic.

The RMC offers the following input filter algorithms in the following areas:

Filter Type	RMC75/150			RMC200		
	Feedback (Control)	Feedback (Display)	Control Output	Feedback (Control)	Feedback (Display)	Control Output
Low Pass, Single Pole			✓	✓		✓
Low Pass, 2-pole Butterworth				✓		
Low Pass, 4-pole Butterworth	✓	✓		✓	✓	
Low Pass ABG				✓		
Low Pass ABGD				✓		
Model	✓			✓		

Phase Delay and Attenuation Comparison

The filters exhibit the following frequency response and phase delay at the cut-off frequency:

At Cut-off Frequency:	Magnitude	Phase Delay
Low Pass, Single Pole	-3 dB (0.707)	45º
Low Pass, 2-pole Butterworth	-3 dB (0.707)	90º
Low Pass, 4-pole Butterworth	-3 dB (0.707)	180º
Low Pass ABG	4 dB (1.56)	- 20º (leading)
Low Pass ABGD	4.5 dB (1.67)	0º

Cut-off Frequency Limits

The filter cut-off frequencies are limited to the ranges listed below, based on the frequency of the selected controller Loop Time.

Filter Type	Min Cut-off Frequency	Max Cut-off Frequency
Low Pass, Single Pole (1P)	0.01	50% x loop frequency
Low Pass, 2-pole Butterworth (BW2)	0.01	25% x loop frequency
Low Pass, 4-pole Butterworth (BW4)	0.01	25% x loop frequency
Low Pass ABG	0.01	10% x loop frequency
Low Pass ABGD	0.01	7.5% x loop frequency

For example, the frequency of a 1 msec loop time is 1 kHz, so the ABG filter would be limited to 100 Hz. The frequency of a 0.5 msec loop time is 2 kHz, so the ABG filter would be limited to 200 Hz.

Filter Descriptions

Low Pass

A low-pass filter passes filters out high frequencies, and passes through lower frequencies. The cut-off frequency defines which frequencies are passed through.

Low Pass, Single Pole (1P)

This is the most simple of all filters. A single-pole filter has a small phase delay, but the amplitude starts dropping off with frequency immediately.

Low Pass, 2-pole Butterworth (BW2)

The main advantage of Butterworth filters is that they have unity amplitude out to the cut-off frequency. However, they have a large phase delay. Therefore, a Butterworth filter works very well for viewing signals, since the human eye doesn't mind some delay. Butterworth filters are not so good for high-speed control applications where phase delay cannot be tolerated.

The 2-pole Butterworth has a more gradual cut-off than the 4-pole, and less phase delay.

Low Pass, 4-pole Butterworth (BW4)

Same as the 2-pole Butterworth, but a sharper cut-off and more phase delay.

Low Pass, ABG

The Alpha-Beta-Gamma filter is a simplified form of observer and is closely related to Kalman filters and linear state observers. The ABG filter has very little phase delay, actually leads for

some frequency ranges, and the amplitude is above unity for some frequency ranges. This means the filtered signal can overshoot the input signal.

Low Pass, ABGD

The Alpha-Beta-Gamma-Delta filter extends the ABG filter (above) one more level, using a "delta" term for a fourth jerk state. This helps provide more filtering without significantly increasing the phase delay, but also makes this filter a bit more finicky, with a lower maximum filter cut-off frequency.

Model

The Model-base filter uses a model of the system to calculate the Actual Velocity and Actual Acceleration, factoring the Control Output into the model. The Tuning Wizard generates the system model. See Modeling for details.

See Also

Filtering

3.8.3. Modeling

Each position, pressure, or force control axis on the RMC can have a system model. A model is an approximate mathematical representation of a real motion system. Directly manipulating the model is typically an advanced feature. Most RMC users will never need to be concerned with the system model.

The model parameters are in the Axis Tools, All tab, Feedback section, in the Filtering/Modeling section.

The system model is used by the **Gain Calculator** or for **Model-based Filtering**:

1. Gain Calculator

When using the Tuning Wizard for tuning position, pressure, or force axes, a model is generated for that axis. Once this model is created, the Gain Calculator uses it to provide a range of gains appropriate for that model. The user chooses a set of gains with a slider bar.

2. Model-based Filtering (Position Axes Only)

Velocity and Acceleration Quantization Error

On standard position control axes, the RMC derives the velocity and acceleration from the position feedback by differentiating the position. It then uses these values in the position control algorithms. This method of deriving the velocity and acceleration causes quantization errors, which are typically extremely large for acceleration.

Modeling for More Accurate Velocity and Acceleration

More accurate values of velocity and acceleration can be obtained by observing the system as it moves, constantly updating the model according to the system's behavior, and deriving the velocity and acceleration from the model. This can result in improved control, since the Differential and Double Differential gains are able to operate on noise-free signals.

For most systems, using model-based filtering is not necessary. It may improve the control on certain difficult-to-control systems.

How to Use Model-based Filtering

To use model-based velocity and acceleration filtering:

1. Use the Tuning Wizard to determine the system model. The Wizard calculates the model and stores the model in the model axis parameters.
   If your system tends to oscillate, make sure in the Tuning Wizard, on the **Model Complete** page, to choose **Advanced** and uncheck the **Limit models to first order box**.

2. After running the Tuning Wizard, you can view the model settings in the axis parameters:
   RMC75/150: In the Axis Tools, on the **All** tab, expand the **Feedback** section and the **Filtering/Modeling** section.
   RMC200: In the Axis Tools, on the **All** tab, expand the **Feedback** section and the **Modeling** section.
3. Set the feedback filter to use **Model**:
   RMC75/150: Set the Velocity Filter Type and/or the Acceleration Filter Type parameters to **Model**.
   RMC200: Set the Input Filter Type to **Model**.
4. Tune the axis using the Gain Calculator to choose gains that are suitable for your system. On the RMC200, you can add the Actual Velocity (Control) and Actual Acceleration (Control) to the plot. You will see that the velocity and acceleration values are very smooth, compared to what they would be without the model-based feedback.
5. You may need to adjust the Model Response parameter, which determines how quickly the model responds to the motion.

Invalid Model

It is possible for the model generation to fail due to more subtle interactions between the model parameters. Specifically, it is possible to have 2nd order models rejected if the Natural Frequency is near its maximum, especially when the Damping Factor is high.

If the feedback model does not appear to be working, you may have an invalid model. Make sure the model parameters are within their ranges. The valid ranges are given in the help topics for each model parameter.

The feedback model will not be active if the model is invalid. The velocity and /or accelerations will be obtained from the feedback instead of the model.

An invalid model will always be logged in the Event Log. If you wish to see more model calculation activities, enable the **Advanced Feedback and Simulator Events** axis item in the Event log. The Event Log will then report the model events.

See Also

Control Features Overview | Model Order | Model Gain Positive | Model Gain Negative | Model Time Constant | Model Natural Frequency | Model Damping Factor

3.9. Plots

3.9.1. Using Plots

The RMC provides very flexible plotting capabilities. Virtually any register in the RMC can be plotted, and multiple registers may be plotted simultaneously. The plot trigger allows events to easily be captured. Plots can be controlled and viewed as time-based or XY plots with the Plot Manager in RMCTools. Plots can be also be started and read via a PLC or other host controller.

Use the Plot Manager to view plots in RMCTools.

How to Plot a Move in RMCTools

See the Plot Manager for more details.

Viewing a Captured Plot

When an axis receives a motion command, the RMC stores a plot of the motion internally, which can be uploaded and viewed.

1. Send a motion command to an axis, such as Move Absolute (20).

2. In the Project Pane, double-click **Plots** to open the Plot Manager.
3. On the **Plotting** tab, click the **Capture** button to upload the plot.
   If you sent a command to Axis 0, upload Plot 0. If you sent a command to Axis 1, double-click Plot 1, and so on.
4. The Plot Manager will open and the plot will automatically upload.
5. The plot will be automatically sized to fit to the plot window. Use the toolbar buttons to zoom in and out, or Ctrl + mousewheel.

You can specify the capture duration in the Plot Template Editor.

Trending a Plot

RMCTools can trend a plot in real time.

1. In the Project Pane, double-click **Plots** to open the Plot Manager.
2. On the **Plotting** tab, for the desired plot template, click the **Trend** button.
3. Send any motion commands to move the axis.
4. To stop the trend, click the **Stop Trend** button.

You can specify the trend duration in the Plot Template Editor. When the trend reaches the duration, the trend will continue and old data will be lost.

Viewing Plots in RMCTools

The Plot Manager can be used to view, trigger, save, and export plots. See the Plot Manager topic for details.

Setting Up Plots

Use the Plot Template Editor to change the plotted registers, number of plots, plot duration, pen colors, plot trigger, etc.

Saving and Exporting Plots

Plots in the Plot Manager can be saved for use later in the Plot Manager, or to send to Delta for technical support. Individual plots can be exported to a file that can be used by other programs, such as Excel, Word, etc. For more details, see the Saving and Exporting Plots topic.

RMCTools automatically saves a backup copy of every uploaded plot to an internal folder on the PC in order to prevent data loss if the user forgets to save plots before closing RMCTools. For details, see Auto-Saved Plots.

Plot Commands

You can use the following commands to start, stop and trigger plots. Issue these commands like any other command. Some of these commands can also be issued directly from the Plot Manager. See each command topic for more details.

Plot Command	Function
Start Plot (100)	Starts a plot immediately.
Stop Plot (101)	Stops a plot immediately.
Trigger Plot (102)	Triggers a plot. See the Triggering Plots topic for more details.
Rearm Plot (103)	Rearms a plot so that it can be triggered again.
Enable/Disable Plot Trigger (104)	Enables or disables the plot trigger.

Reading Plots with a PLC, HMI, or other Host Controller

For details on how to read RMC plots with a PLC, HMI, or other host controller, see the Reading Plots with a Host Controller topic.

Plot Limits

CPU	Max Plot Templates	Max Data Items per Plot	Max Total Samples* per Loop Time	
RMC75E	8	16	128	
RMC75S	8	16	128	
RMC75P	8	16	128	
RMC150E	8	16	128	
CPU20L	48	48	512	
CPU40	64	128	1024	(firmware 1.14.0 and later)
	32	32	1024	(firmware 1.00.0 - 1.13.1)

* Total Samples is the sum of the number of data items in all plot templates.

Storage Size

The RMCs have a fixed storage capacity for captured plots as follows:

CPU	Total Plot Samples	
RMC75E	12,582,912	(loader 1.03 and later)
	4,194,304	(loader versions 1.00-1.02)
RMC75S	32,768	
RMC75P	32,768	
RMC150E	12,582,912	
CPU20L	12,582,912	
CPU40	50,331,648	(firmware 1.12.0 and later)
	12,582,912	(firmware 1.00.0 - 1.11.3)

For the RMC75 and RMC150, the storage is equally divided between the plots. For the RMC200, each plot uses only the storage it needs.

See Also

Plot Manager Overview | Using Custom Plot Templates | Using XY Plots

3.9.2. Saving and Exporting Plots

Plots that have been uploaded in the Plot Manager can be saved for use later in the Plot Manager, or to send to Delta for technical support. Plots can also be exported to various file formats for use by other programs, such as Microsoft Excel, Word, etc. as described in the **Exporting Plots** section below.

Plots are not saved in an RMCTools project. To save plots, you must explicitly save them. If you forgot to save the plots before closing RMCTools, you can retrieve a backup copy that RMCTools automatically saves to an internal location on the PC as described in the **Auto-Saved Plots** section below.

To Save Plots

In the Plot Manager, the **History** tab shows the uploaded plots, and any other open plot files. To save one or more plots in this list:

1. On the Plot Manager toolbar, click **Save Plot(s)**.
   Or, on the Plot History pane, right-click a plot in the list and choose **Save Plot(s)**.
   Or, in the plot window, right-click the currently displayed plot and choose **Save Plot(s)**.
2. In the **Select Plot to Save** dialog, choose the plots to include, then click **Save**.
3. Browse to the desired folder and click **Save**.

To Open a Saved Plot File

In the Plot Manager toolbar, click the **Open Plot File** button. In the **Open** dialog, browse to the desired plot file, select it, and click **Open**. In the **Open** dialog, you can use the Ctrl or Shift key to select and open multiple files. You can open .rmcplotx and .rmcplots file types.

Auto-Saved Plots

RMCTools automatically saves a backup copy of every uploaded plot to an internal folder on the PC. This is intended for preventing data loss if the user forgets to save plots before closing RMCTools, and is not intended to replace saving plots manually to the location and filename of the user's choosing.

Note: This feature does not automatically upload plots from the RMC. It only automatically saves a backup of each plot that you manually upload from the RMC in the Plot Manager.

Once the size of the auto-saved plots exceeds the defined storage size limit, the oldest plots are automatically deleted. The size limit can be adjusted as described below in **Auto-Save Settings**.

Retrieving Auto-Saved Plots

1. In the Plot Manager, on the **History** tab, expand **Auto-Saved Plots**, then click **View Auto-Saved Plots**.
2. Choose the plot files you wish to retrieve. You can select multiple files by holding the Shift or Ctrl keys while clicking the files.

   Each file contains one plot and the filename describes the plot as:

   ProjectName_ControllerName_PlotTemplateName_PlotTemplateNumber_CaptureDate_CaptureTime.rmcplotx,

   where

   CaptureDate is in *yyyymmdd* format (year, month, day)

   CaptureTime is in *hhmmssxxx* format (hours, minutes, seconds, milliseconds)

3. Click **Open**.
4. Save the retrieved auto-save plot to a location and filename of your choice. Without saving the file, the auto-saved plot will remain in the internal auto-save folder and will eventually be replaced by newer plots.

Auto-Save Settings

To change the size limit of the auto-stored plots storage area or to enable or disable the auto-saving:

1. In the Plot Manager, on the **History** tab, expand **Auto-Saved Plots**, then click **Auto-Save Settings**.
2. In the **Plot Auto-Save** section, check or uncheck the **Automatically Save Uploaded Plot file to your PC** box to enable or disable the feature.
3. In the **Plot Auto-Save** section, enter the **Plot Auto-Save Size Limit**, then click **OK**. The size limit can be adjusted between 1 and 2048 MB.

Storage Location

The auto-saved plots are saved within the user's AppData folder, which is a hidden folder. The location is typically C:\Users\<user>\AppData\Local\Delta Motion\RMCTools\AutoSavedPlots.

Since this is a hidden folder, it is easiest to access it as described in **Retrieving Auto-Saved Plots** above.

Exporting Plots

The data from an RMC plot can be exported to a file for use in other programs, such as Excel. The Plot Manager can export the data to three different file types, all of which can be opened by a text editor:

- Comma Delimited (.csv)
- Tab Delimited (.txt)
- XML Spreadsheet (.xml)
- XML Data (.xml)

Tip:
If you will be opening the exported file in Microsoft Excel, choose **XML Spreadsheet**. It will provide a very clean spreadsheet.

To Export a Plot:

1. In the Plot Manager, on the **History** tab, right-click a plot and choose **Export**.
2. In the **Filename** box, enter a filename.
3. In the **Save as Type** box, choose the file type, and click **Save**.
4. The plots will be saved in the file with the extension you chose. Notice that only the plot you selected will be exported to the file.

Plot File Types

RMCTools plots can be saved and opened in two formats:

- **RMCTools Plots - Compressed (*.rmcplotx)**
  Compressed file containing one or more plots. Typically, this file size is 10% of the .rmcplots file size. The compressed (.rmcplotx) format is supported in RMCTools 4.09.0 and newer. The uncompressed (.rmcplots) format must be used if plots need to be opened in earlier versions of RMCTools.
- **RMCTools Plots (*.rmcplots)**
  Uncompressed file containing one or more plots. This uncompressed XML file is easily readable in a text editor.

Default File Type When Saving

When saving plots, the default file type is **RMCTools Plots - Compressed (*.rmcplotx)**. To change the default:

1. On the **Tools** menu, choose **Options**.
2. On the **Environment** page, in the **Plot File Format** section, choose the desired default format.

Compression Type

The .rmcplotx file is an .rmcplots file compressed with the gzip algorithm, with the addition of some header information. An .rmcplotx file can be uncompressed into an .rmcplots file by changing the file extension from .rmcplotx to .gzip, then unzipping with file compression software such as 7-Zip.

A .rmcplots file can also be made into an .rmcplotx file by compressing it with the gzip algorithm via a file compression software such as 7-Zip. Rename the compressed file to .rmcplotx, and RMCTools should be able to open it.

See Also

Plot Overview

3.9.3. Triggering Plots

Triggering a plot means capturing plot data. Triggering a plot allows you to see data from before you triggered the plot and after you triggered the plot. This is a valuable aid in troubleshooting. For example, if you set up the trigger to contain 25% of the data from before the time the trigger occurred, then the plot will contain data from before the command was issued until some time after it was issued. To specify how much of the plot data is from before the plot was triggered and how much is after the plot trigger, you must use the Trigger Position value.

The default setting is a trigger position of 0%. This means that a triggered plot will not contain data from before the time the plot was triggered. If you set the trigger position to a value other than 0, you will need to manually rearm a plot before it can trigger. Read below for more details.

Note:
Compare *triggering* a plot to *starting* a plot with the Start Plot (100) command. The Start Plot command starts plotting immediately and will never contain any past information.
Note:
You can entirely enable or disable the plot trigger by issuing the Enable/Disable Plot Trigger (104) command. A plot cannot be triggered unless the plot trigger is enabled.

Using the Trigger Percentage

The Trigger Percentage specifies how much of the plot data is from before the plot was triggered and how much is after the plot Trigger. The Trigger Percentage ranges from 0% to 100%. If the Trigger Percentage is set to 0%, the triggered plot will not contain any past data. If the Trigger Percentage is set to 100%, the triggered plot will contain only past data. To set the Trigger Percentage, see the **Changing the Trigger Settings** section below.

If the plot was not re-armed early enough so that the plot has time to capture the data before the trigger occurs, the plot will not include the expected amount of pre-trigger data.

Example:

You have set the Plot Duration to 4 seconds and the Trigger Percentage to 25%. When you trigger a plot, the first 1 second of the plot will contain data from immediately before the trigger occurred, and the last 3 seconds will contain data from immediately after the trigger occurred.

How to Trigger a Plot

Note:
To trigger a plot, the trigger must first be armed. See the **Rearming the Trigger** section below.
Note:
If a plot is triggered immediately after rearming it, and the trigger percentage is greater than 0%, the plot will not contain any data before the trigger. After rearming, there must be a delay long enough to record the data to fill the plot before the trigger time.

There are two ways of triggering a plot: manually and automatically:

- **Automatic Trigger**

  Plots can be automatically triggered by certain conditions. Currently, a plot can be automatically triggered only by motion commands. If automatic triggering is enabled, a plot will trigger every time a motion command is issued to the specified axis.

  To set up automatic triggering, see the **Changing the Trigger Settings** section below.

  Note:
  The RMC default setting is to automatically trigger plots on motion commands.

- **Manual Trigger**

To manually trigger a plot, issue the Trigger Plot (102) command. You can also issue the Trigger Plot command by choosing **Trigger Plot** on the **Plots** menu.

Note:
Even if you manually trigger a plot, the RMC may also automatically trigger plots because it is set to do so by default. To disable automatic triggering, see the **Changing the Trigger Settings** section below.

Rearming the Trigger

Before triggering a plot, the trigger must first be *armed*. This tells the RMC to start recording the data in the plot so that when the plot is triggered, it can provide the data. The plot will not trigger if it is not armed. When the RMC starts up, the trigger is armed and therefore prepared to trigger. After triggering a plot, the trigger is no longer armed and must be rearmed before you can trigger a plot again.

The default setting is Automatic rearm. Therefore, you do not need to rearm the plot if you have the default settings.

There two methods of rearming the trigger:

- **Automatically Rearm**

  The plot trigger can be set to automatically rearm after it triggers. This is the default RMC plot setting. After a plot triggers, it is ready to be immediately triggered again.

  Note:
  If you are issuing motion commands in rapid succession, the automatic rearm feature may cause the plots to be cut short when the next plot is triggered. To get long plots, you should change the plot settings to *manual* rearm. See the **Changing the Trigger Settings** section below.

- **Manually Rearm**

  To manually rearm the trigger, issue the Rearm Plot (103) command. You can also issue the Rearm Plot command from the Plot Manager by choosing **Rearm Plot** on the **Plots** menu. To enable manual rearming, see the **Changing the Trigger Settings** section below.

  Rearming a plot will clear all previously captured data for that plot.

Changing the Trigger Settings

The trigger settings can be configured for each individual plot. To change the trigger settings for a plot, follow these steps:

1. In the Plot Template Editor, on a plot tab, choose **Custom**.
2. In the **Trigger Settings** area, click **Edit Trigger Settings**.
3. Make the desired changes and click **OK**.

See Also

Plot Overview | Plot Template Editor

3.9.4. Using Plots with a Host Controller

RMC plots can be controlled from a PLC or other host controller. To start, stop, or trigger a plot, use the plot commands. For additional information on using RMC plots, see the Plot Overview topic.

Plot Commands

The RMC has several commands specifically for plots.

Start Plot (100)

Stop Plot (101)

Rearm Plot (103)

Trigger Plot (102)

Enable/Disable Plot Trigger (104)

Reading RMC Plots with a PLC

The RMC plots can be read with a PLC, HMI, or other host controller. For details, see the Reading Plots with a Host Controller topic.

See Also

Plot Overview

3.9.5. Reading RMC Plots with a Host Controller

This topic describes how to read the RMC plots with a PLC, HMI, PC (via RMCLink) or other host controller. If you need help on viewing RMC plots in RMCTools, see the Plot Overview topic. For help on configuring the plots, such as changing the sample period or which registers are plotted, see the Plot Template Editor topic.

Determining which Data Items to Read

The RMC plot data items you can read via a host controller are not necessarily the same as the items displayed in the Plot Manager because the Plot Manager calculates some items. For example, some velocities are not actually uploaded from the controller, but are calculated from the position data.

To see which data items are actually captured and stored in the RMC:

1. In the Plot Template Editor, on a plot tab, choose **Custom**.
2. In the Plotted Data table header, click **Show Data Items**. These are the plot data items that you can read via a host controller.
3. The order of the captured data items can be changed by clicking the Up and Down arrows.

Addressing Methods

You can read plot data via any addressing method that your RMC supports for the protocol being used:

RMC75 and RMC150:
You can directly address the plot registers.

RMC200:
For protocols using Modbus, DF1 (e.g. EtherNet/IP), and FINS addressing, or for PROFINET Data Records, the plot data registers are not included in the fixed data map addresses. You will need to assign the plot data registers to address ranges for your protocol in the Address Maps. For details, see DF1 Address Map, FINS Address Map, Modbus Address Map, or PROFINET Data Records Address Map.

For other communication types, such as those using DMCP, you can address the plot data registers directly, since they use the IEC Address Map, which includes all RMC200 registers.

Methods

The RMC provides several methods of reading plots with a host controller:

- **Method 1: Read a Captured Plot - Basic**
  This is the simplest method and is suitable for use by HMIs with limited sequencing capabilities and by low-bandwidth media, such as Basic or Enhanced PROFIBUS modes. This method is limited to:
  - **RMC75:** Maximum 2 plots, maximum 4 data sets per plot
  - **RMC150:** Maximum 8 plots, maximum 4 data sets per plot
  - **RMC200:** Maximum 4 plots, maximum 16 data sets per plot
- **Method 2: Read a Captured Plot - Advanced**
  With this method, you can read any plot in the RMC. However, it requires more sequencing capabilities in the host controller than Method 1.
- **Method 3: Read a Continuous Plot - Continuous Data**
  With this method, you can read a plot continuously. As long as the host controller reads keep up with the data capture rate, there will be no gaps or overlap in the data. This method requires sequencing capabilities in the host controller.
- **Method 4: Read a Continuous Plot - Newest Data**
  With this method, you can read a plot continuously. The newest data is always captured whether or not the data causes an overlap or gap with the last data that was read. This method requires sequencing capabilities in the host controller.

The methods refer to the registers listed in the **Registers** section below.

Method 1: Read a Captured Plot - Basic

This method makes it very easy to read plot data, but limits the amount of data that can be read. This method allows reading the following:

RMC75: Maximum 2 plots, maximum 4 data sets per plot

RMC150: Maximum 8 plots, maximum 4 data sets per plot

RMC200: Maximum 4 plots, maximum 16 data sets per plot

The numbers of samples each data set that can be read using this method depends on the communication method:

Communication Method	Max Samples
Modbus RTU, Modbus/TCP, FINS/UDP, Mitsubishi Procedure Exist, Mitsubishi Bidirectional Protocol	255
PROFINET (RMC75 and RMC150)	2048
PROFINET (RMC200)	4096
DF1, CSP, EtherNet/IP, RMCLink	4096

These sample limits do not necessarily limit the length of the plot in seconds, because the sample time of the plot can be changed. You can also configure which registers are plotted. See the Plot Template Editor topic for details.

Due to addressing limitations of the communication protocols themselves, the RMC cannot provide directly mapped registers for all the plot data. The protocols simply do not have enough address space for all the data in the RMC. The Basic method strikes a compromise by providing some of the plot data as directly mapped registers. If you need to read all the plot data, use one of the other methods.

To read a plot using this method, follow these steps:

1. Wait for the Plot to Complete

Before reading the plot data using this method, the plot must have completed capturing data. Reading the plot before it has completed may result in unusable information.

Use the **Plot State** register to tell whether the plot has completed capturing. The plot has completed when the **Plot State** register is 2:
0 = not triggered, 1 = capturing, 2 = complete.

2. Read the Plot Data

Read the data from the **Static Plot Upload Area**. To find the correct addresses to read from, use the Address Maps in RMCTools, or see the appropriate register map topic: RMC75 Register Map - Static Plot Upload, RMC150 Register Map - Static Plot Upload, RMC200 Register Map - File 640-703 Static Plot Upload

For the RMC200, you may need to configure addresses in the Address Map so your external device can access the Static Plot Upload Area.

If there is a chance that the plot ended early, read the **Plot Captured Samples** register to find out how many plot samples were actually captured. Plot data beyond the specified number of captured samples is invalid.

Note:
Many host controllers do not support reading files with 4096 elements. However, smaller blocks of data can be read just as well.

Tip:
In order to use the plot data, you will probably need to know the sample period. The **Plot Sample Period** register contains that information.

Verifying that the Plot wasn't Overwritten

It is possible that while you are reading up a plot, the plot re-triggers and is overwritten. In that case, your data may be unusable. To check whether the plot was overwritten, compare the **Plot ID** register, before and after reading the plot data.

Reading a Plot While it's Still Capturing

It is possible to begin reading up a plot before it has finished capturing. However, reading beyond the end of the data set's currently-captured limit will result in unusable information.

To begin reading up a plot before it has finished capturing, follow these steps:

1. Make sure the plot is currently capturing

Read the **Plot State** register and make sure it is 1 or 2, which means it is in the process capturing data or has completed.

2. Check how many samples have been captured

Read the **Plot Captured Samples** register to find out how many plot samples have already been captured.

3. Read the Plot Data

Read any of the plot data from sample 0 up to, but not including, the sample number you obtained from the **Plot Captured Samples** register in step 2. Reading beyond that will result in unusable data.

Note:
Method 2 can also be used to read a plot while capturing.

Method 2: Read a Captured Plot - Advanced

Use this method to read any data from any plot. This method uses the Dynamic Plot Upload Area registers.

To read a plot using this method, follow these steps:

1. Write the Requested Read Samples per Data Set

Write the number of samples per data set that you would like returned per read to the **Requested Read Samples** register. The minimum length of the read itself will be affected by this value, as described later.

2. Write 1 to the Upload Mode/Status Register

Write a value of one (1) to the **Upload Mode/Status** register. This sets the upload mode to "triggered" and resets the **Current Index** register to zero.

The **Current Index** register indicates which sample number of the plot the next read will start at. When it resets to 0, the next read will start at sample 0.

3. Read the Plot Data Starting from Dynamic Plot Upload register #0

You can read the plot data in one read or several reads. Each time, start the read at **Dynamic Plot Upload** register #0 (**Upload Mode/Status Register)**. The **Current Index** register will automatically be set to the index of the first sample of the data returned in the read. Each read will return the **Dynamic Plot Upload** registers 0-4 *and* the plot samples beginning at the sample number in the **Current Index** register. Since each read includes registers 0-4, you will have access to this information after each read.

If there is a chance that the plot ended early, read the **Plot Captured Samples** register to find out how many plot samples were actually captured. Plot data beyond the specified number of captured samples is invalid.

To read the plot data, repeat these steps until you have read the entire plot:

a. Calculate the Length of the Read.

The length of the read must be long enough to receive the number of samples you wrote to the **Requested Read Samples** register. In addition to returning **Dynamic Plot Upload** registers 0-4, each read will return an equal number of samples from each data set in the plot. Use the following equation to calculate the length of the read based on how many samples you wish to read:

length = (**Requested Read Samples**) x (# of data sets) + 5

The number of samples should not be more the number of samples in the plot. Typically, you will make this calculation once and then always make reads of the same length.

Tip: For best performance over TCP/IP, the read length should be less than 350. This keeps the data within one Ethernet frame, which helps prevent communication delays. The **Requested Read Samples** may need to be reduced to meet this requirement.

b. Read from Dynamic Plot Upload register 0

Read from register 0 with the length calculated in step **a**.

c. Do any Error Checking

If you wish, you can use **Dynamic Plot Upload** registers 0-4 to do error checking, as described below:

- Verify that the **Samples Uploaded** is the value you expect. It should be the number of samples read per data set. If it is zero, you have read past the end of the currently captured data and the data will be unusable.

  You can make use of this to read a plot while it's capturing. You can continuously make reads and only use the data when the **Samples Uploaded** is not zero.
- Verify that the **Current Index** is the value you expect. It should be the index of the first sample of the data returned in the read.
- Verify that the **Plot ID** is unchanged. If it has changed, it indicates that you are no longer reading from the same plot. The plot has probably re-triggered and is overwritten. Your data may be unusable.

Note:
If you attempt to read a plot with the **Upload Mode** set to 1, and the plot is untriggered, the **Upload Status** will be set to 1 to indicate the error, and the **Current Index** and **Samples Uploaded** registers will be 0.

d. Copy Plot Data to Buffer

Before the next read, you should, of course, copy the data you just read to a plot buffer. The **Samples Uploaded** register reflects how many valid samples you read. If you read past the end of the plot, the **Samples Uploaded** will be zero.

The data is returned in this format:

R_0	R_1	R_2	R_3	R_4	A_0,A_1,A_2 ... A_{n-1}	B_0,B_1,B_2 ... B_{n-1}	C_0,C_1,C_2 ... C_{n-1}	D_0,D_1,D_2 ... D_{n-1}	...

where

R_0= **Upload Mode/Status** register

R_1= **Requested Read Samples** register

R_2= **Current Index** register

R_3= **Plot ID** register

R_4= **Samples Uploaded** register

A = sample from data set 0, B = sample from data set 1, etc.

n = number of samples read per data set.

Tip:
In order to use the plot data, you will probably need to know the sample period. The **Plot Sample Period** register contains that information.

Method 3: Read A Continuous Plot (Trending) - Continuous Data

Use this method to read plot data continuously. This method guarantees no gaps or overlap of data, but it requires that the upload keeps up with the capture of data.

To use this method, the plot should be continuously capturing. To set the plot to continuously capture, issue the Rearm Plot (103) command and do not trigger or start the plot. **Important:** You must disable the plot's automatic trigger in the Plot Template Editor.

This method uses the Dynamic Plot Upload Area registers.

To read a plot using this method, follow these steps:

1. Verify that the Plot is Continuously Capturing.

Use the **Plot State** register to tell whether the plot is continuously capturing. The plot is continuously capturing when the **Plot State** register is 0:
0 = not triggered, 1 = capturing, 2 = complete.

2. Write the Requested Read Samples per Data Set

Write the number of samples per data set that you would like returned per read to the **Requested Read Samples** register. The minimum length of the read itself will be affected by this value, as described later.

3. Write 2 to the Upload Mode/Status Register

Write a value of two (2) to the **Upload Mode/Status** Register. This sets the upload mode to "continuous" and resets the **Current Index** register to zero. The **Current Index** register indicates which sample number of the plot the next read will start at. When it resets to 0, the next read will start at sample 0. In continuous mode, the **Current Index** may become very large. It is a 32-bit number and will wrap when it reaches its limit.

If you are reading multiple plots simultaneously and need them synchronized with each other, make sure to write a 2 to the Upload Mode/Status Register to all the desired plots before beginning to read the data from any of them. At the first read of data, the RMC will synchronize the data of all the armed (currently continuously capturing) plots in Upload Mode 2 that have not yet been read.

4. Read the Plot Data Starting from Dynamic Plot Upload register #0

Upload Mode 2 will return valid data only if there is enough captured data to fill the request. If there is not enough captured data, the **Samples Uploaded** register will return 0. Therefore,

the typical method of reading a plot is to keep reading the plot data and only using the data when the **Samples Uploaded** register is non-zero.

If the read falls too far behind to keep up with the data, the **Upload Status** bits will be set to two (2) to indicate an overrun error. To recover, restart the plot by writing zero (0) to the **Current Index** register.

Each time you read the plot data, start the read at **Dynamic Plot Upload** register #0. The **Current Index** register will automatically be set to the index of the first sample of the data returned in the read. Each read will return **Dynamic Plot Upload** registers 0-4 *and* the plot samples beginning at the sample number in the **Current Index** register. Since each read includes registers 0-4, you will have access to this information after each read.

To read the plot data repeat these steps:

a. Calculate the Length of the Read.

The length of the read must be long enough to receive the number of samples per data set you wrote to the **Requested Read Samples** register. In addition to returning **Dynamic Plot Upload** registers 0-4, each read will return an equal number of samples from each data set in the plot. Use the following equation to calculate the length of the read based on how many samples you wish to read:

length = (**Requested Read Samples**) x (# of data sets) + 5

The number of samples should not be more the number of samples in the plot. Typically, you will make this calculation once and then always make reads of the same length.

Tip: For best performance over TCP/IP, the read length should be less than 350. This keeps the data within one Ethernet frame, which helps prevent communication delays. The **Requested Read Samples** may need to be reduced to meet this requirement.

b. Read from Dynamic Plot Upload register #0

Read from the **Dynamic Plot Upload** register #0 with the length calculated in step **a**.

c. Do any Error Checking

If you wish, you can use **Dynamic Plot Upload** registers 0-4 to do error checking, as described below:

- Verify that the **Upload Status** is 0, meaning no errors have occurred.

  If the **Upload Status** is 1, the plot has been triggered, completed capturing, and has been completely uploaded. To recover, issue the Rearm Plot (103) command to set it to continuous capture.

  If the **Upload Status** is 2, the capturing data overran the data you were going to read. To recover from this error, write zero (0) to the **Current Index** register, which will restart the plot. You will have to restart reading the plot data.

- Verify that the **Plot ID** register is unchanged. If it has changed, it indicates that you are no longer reading from the same plot. The plot may have been re-triggered or reset.

d. Check the Samples Uploaded Register

Verify that the **Samples Uploaded** register is not zero. If it is, there is not enough captured data to fulfill the request. Repeat steps **b**, **c**, and **d** until the **Samples Uploaded** register is non-zero.

e. Copy Plot Data to Buffer

Before the next read, you should, of course, copy the data you just read to a plot buffer.

The data is returned in this format:

R_0	R_1	R_2	R_3	R_4	A_0,A_1,A_2 ... A_{n-1}	B_0,B_1,B_2 ... B_{n-1}	C_0,C_1,C_2 ... C_{n-1}	D_0,D_1,D_2 ... D_{n-1}	...

where

R_0= **Upload Mode/Status** register

R_1= **Requested Read Samples** register

R_2= **Current Index** register

R_3= **Plot ID** register

R_4= **Samples Uploaded** register

A = sample from data set 0, B = sample from data set 1, etc.

n = number of samples read per data set.

Tip:
In order to use the plot data, you will probably need to know the sample period. The **Plot Sample Period** register contains that information.

Method 4: Read A Continuous Plot (Trending) - Newest Data

Use this method to read plot data continuously. This method captures data whether or not the data causes an overlap or gap with the last data that was read.

To use this method, the plot should be continuously capturing. To set the plot to continuously capture, issue the Rearm Plot (103) command and do not trigger or start the plot. **Important:** You must disable the plot's automatic trigger in the Plot Template Editor.

This method uses the Dynamic Plot Upload Area registers.

To read a plot using this method, follow these steps:

1. Verify that the Plot is Continuously Capturing.

Use the **Plot State** register to tell whether the plot is continuously capturing. The plot is continuously capturing when the **Plot State** register is 0:
0 = not triggered, 1 = capturing, 2 = complete.

2. Write the Requested Read Samples per Data Set

Write the number of samples per data set that you would like returned per read to the **Requested Read Samples** register. The minimum length of the read itself will be affected by this value, as described later.

3. Write 3 to the Upload Mode/Status Register

Write a value of three (3) to the **Upload Mode/Status** Register. This sets the upload mode to "continuous-always newest" and resets the **Current Index** register to zero. The **Current Index** register indicates which sample number of the plot the last read started at. In continuous mode, the **Current Index** may become very large. It is a 32-bit number and will wrap when it reaches its limit.

If you are reading multiple plots simultaneously and need them synchronized with each other, make sure to write a 3 to the Upload Mode/Status Register to all the desired plots before beginning to read the data from any of them. At the first read of data, the RMC will synchronize the data of all the armed (currently continuously capturing) plots in Upload Mode 3 that have not yet been read.

4. Read the Plot Data Starting from Dynamic Plot Upload register #0

Upload Mode 3 will always return the newest data whether or not the data causes an overlap or gap with the last data that was read. Use the **Current Index** register to keep track of the data. The **Current Index** register indicates which sample number of the plot the last read started at.

Each time you read the plot data, start the read at **Dynamic Plot Upload** register #0. Each read will return **Dynamic Plot Upload** registers 0-4 *and* the plot samples beginning at the sample number in the **Current Index** register. Since each read includes registers 0-4, you will have access to the **Upload Mode/Status**, **Current Index**, **Plot ID**, and **Samples Uploaded** after each read.

To read the plot data repeat these steps:

a. Calculate the Length of the Read.

The length of the read must be long enough to receive the number of samples per data set that you wrote to the **Requested Read Samples** register. In addition to returning **Dynamic Plot Upload** registers 0-4, each read will return an equal number of samples from each data

set in the plot. Use the following equation to calculate the length of the read based on how many samples you wish to read:

length = (**Requested Read Samples**) x (# of data sets) + 5

The number of samples should not be more the number of samples in the plot.

Typically, you will make this calculation once and then always make reads of the same length.

Tip: For best performance over TCP/IP, the read length should be less than 350. This keeps the data within one Ethernet frame, which helps prevent communication delays. The **Requested Read Samples** may need to be reduced to meet this requirement.

b. Read from Dynamic Plot Upload register #0

Read from the **Dynamic Plot Upload** register #0 with the length calculated in step **a**.

c. Do any Error Checking

If you wish, you can use **Dynamic Plot Upload** registers 0-4 to do error checking, as described below:

- Verify that the **Upload Status** is 0, meaning no errors have occurred. If the **Upload Status** is 1, the plot has been triggered, completed capturing, and has been completely uploaded. To recover, issue the Rearm Plot (103) command to set it to continuous capture.
- Verify that the **Plot ID** is unchanged. If it has changed, it indicates that you are no longer reading from the same plot. The plot may have been re-triggered or reset.

d. Copy Plot Data to Buffer

Before the next read, you should, of course, copy the data you just read to a plot buffer. Use the **Current Index** to know where to place the data.

The data is returned in this format:

R_0	R_1	R_2	R_3	R_4	A_0,A_1,A_2 ... A_{n-1}	B_0,B_1,B_2 ... B_{n-1}	C_0,C_1,C_2 ... C_{n-1}	D_0,D_1,D_2 ... D_{n-1}	...

where

R_0= **Upload Mode/Status** register

R_1= **Requested Read Samples** register

R_2= **Current Index** register

R_3= **Plot ID** register

R_4= **Samples Uploaded** register

A = sample from data set 0,

B = sample from data set 1, etc.

:

n = number of samples read per data set.

Tip:
In order to use the plot data, you will probably need to know the sample period. The **Plot Sample Period** register contains that information.

Registers

The following registers are used in the methods above. The addresses are given here.

Note: For the RMC200, for some protocols, you cannot directly address the plot registers, but will need to use the Address Maps to assign them to addresses for your protocol.

#	Register Name	Details	RMC75 Address (*n= plot #)*	RMC150 Address (*n= plot #)*	RMC200 Address (*n= plot #)*	Access	Data Type
Plot Status/Configuration Registers							
2	**Plot Sample Period**	The time between consecutive samples.	%MD(32+*n*).2	%MD(96+*n*).2	%MD(512+*n*).2	Read/Write	REAL
8	**Plot ID**	If this value changes, it indicates that you are no longer reading from the same plot. The plot has probably re-triggered.	%MD(32+*n*).8	%MD(96+*n*).8	%MD(512+*n*).8	Read Only	REAL
9	**Plot State**	Use to determine if the plot is ready to read. 0 = not triggered 1 = capturing 2 = complete	%MD(32+*n*).9	%MD(96+*n*).9	%MD(512+*n*).9	Read Only	REAL
10	**Plot Captured Samples**	The number of plot sample periods that are actually captured. For captured plots, this value is useful for determining how many samples were captured	%MD(32+*n*).10	%MD(96+*n*).10	%MD(512+*n*).10	Read Only	REAL

		if the plot ended early.					
Dynamic Plot Upload Registers							
0	**Upload Mode/Status**	Bits 0-3: **Upload Mode** 1 = triggered 2 = continuous 3 = continuous, always newest Bits 4-7: **Upload Status** 0 = normal 1 = plot state mismatches upload mode 2 = data overran (mode 2 only)	%MD(40+ *n*).0	%MD(104+ *n*).0	%MD(576+ *n*).0	Read/Write	UDINT
1	**Requested Read Samples**	This register controls the number of samples provided per data set in the **Plot Data** registers.	%MD(40+ *n*).1	%MD(104+ *n*).1	%MD(576+ *n*).1	Read/Write	UDINT
2	**Current Index**	Indicates the plot sample number currently in the **Plot Data**	%MD(40+ *n*).2	%MD(104+ *n*).2	%MD(576+ *n*).2	Read/Write	UDINT

		registers. Use this to keep track of where you are in reading the data.					
3	**Plot ID**	If this value changes, it indicates that you are no longer reading from the same plot. The plot has probably re-triggered.	%MD(40+*n*).3	%MD(104+*n*).3	%MD(576+*n*).3	Read Only	UDINT
4	**Samples Uploaded**	The number of samples available in each data set in the **Plot Data**. This will either match the **Requested Read Samples** or be zero if there is not enough data available to read.	%MD(40+*n*).4	%MD(104+*n*).4	%MD(576+*n*).4	Read Only	UDINT
5+	**Plot Data**	The actual data.	%MD(40+*n*).5-4095	%MD(104+*n*).5-4095	%MD(576+*n*).5-4095	Read Only	*

See Also

Plot Overview

3.9.6. Mean Squared Error

The Mean Squared Error (MSE) is a quantity that can be included in a plot. The MSE is a single number that indicates how closely two other plot quantities are to each other during the entire plot. The closer the quantities are, the smaller the MSE will be.

The Mean Squared Error is typically used during manual tuning as a measure of how close the Actual Position, Velocity, Pressure or Force is tracking the Target. As the tuning progresses, the MSE should become smaller, indicating that the actual value is tracking the target more closely. The actual value of the MSE is not necessarily important. The importance is that the MSE decreases.

Notice that the Mean Squared Error should not be used as the main guide for tuning. Tuning a system solely to achieve the lowest MSE possible may result in a system that is tuned very close to instability. Any change in the system, such as temperature or load, may cause the system to begin oscillating.

Adding the MSE to a Plot

The Mean Squared Error is included in default plots for control axes. To add the MSE to a custom plot, do the following:

1. In the Plot Template Editor, click **New Quantity**.
2. On the **Advanced** tab, in the **Formula** box, choose **Mean Squared Error**.
3. To choose the MSE for the position, velocity, pressure, or force, choose **Standard Error Quantity** and select the axis and quantity.
   To choose the MSE for any other values, choose **Custom Error Quantity** and select the two registers.
4. Click **Finish**, then download the Plot Template to the RMC. The next plot captured for that plot template will include the MSE.

Mathematical Definition

The Mean Squared Error is the average of the square of the difference. The difference between each sample is squared, and the average of all the differences is the MSE.

See Also

Plot Template Editor

3.10. Custom Feedback

3.10.1. Custom Feedback

Custom feedback gives the flexibility to use feedback from a source other than directly from a hardware input. Custom feedback requires that the user create a user program that continuously calculates some value to be used by the axis for feedback. Custom feedback requires firmware 3.54.0 or newer. Notice that simulate mode is not available for axes with custom feedback.

Custom feedback has many uses, including the following applications:

- **Controlling to the sum, difference, or average of feedback devices**
  Custom feedback makes it possible to control to the sum of the forces of multiple hydraulic cylinders, control the difference between two actuators, or control the average of actuator positions.
- **Switching feedback**

Switching feedback for an axis is useful in certain testing applications, or for using several transducers to provide the desired resolution over the range of the feedback.

For details, see Switching Feedback using Custom Feedback.

- **Feedback linearization**

  Feedback linearization refers to either "straightening" the output of a transducer to make up for its nonlinearity, or calculating the measurement of some point that is geometrically related to the transducer measurement, such as calculating the end of a swing arm based on the position of the cylinder that moves the swing arm.

  Feedback linearization can de done using curves or a mathematical formula. For details, see Feedback Linearization using Curves and Feedback Linearization using Mathematical Formula.

- **Redundant feedback**

  If one of multiple feedback devices fails, the system can continue operating without interruption.

  For details, see Redundant Feedback using Custom Feedback.

How It Works

Axes defined with custom feedback do not have an assigned physical feedback. Rather, they have an _Axis[].CustomCounts register (_Axis[].SecCustomCounts for secondary inputs) that can be written to. You must create a user program that continuously calculates the feedback value and assigns it to this register.

The _Axis[].CustomCounts or _Axis[].SecCustomCounts value will have the Scale and Offset applied to it, and the result then appears in the Actual feedback register for that axis, whether it be Actual Position, Actual Velocity, Actual Pressure, Actual Force or Actual Acceleration. Typically, the Scale and Offset are left at their default values of 1 and 0, which means that the Custom Counts value remains unchanged and the value you want as the feedback can be directly written to the _Axis[].CustomCounts or _Axis[].SecCustomCounts register.

Limitations

Custom feedback has the following implications:

- The RMC must always be in RUN mode
- One task will be continuously running the user program that is performing the custom feedback calculations. You must make sure this program always runs, as described below.
- Auto-tuning cannot be used, since auto-tuning requires that the RMC be in PROGRAM mode. Auto-tuning may be used for other axes in the RMC, but the custom feedback axes can not be operating while auto-tuning other axes.
- Rotary axes are not supported by custom feedback.

Setting Up and Using Custom Feedback

Planning Number of Axes

Applications making use of custom feedback usually require the use of extra reference axes. This may result in exceeding the total number of axes available, particularly in the RMC75, which is limited to four axes.

For example, consider a single-axis position feedback linearization application utilizing custom feedback. Without custom feedback, a single-axis position application uses one axis in the RMC. However, the custom feedback will require two axes: the control axis, which will use custom feedback, and a reference axis, which will use the physical input from the position transducer.

Make sure to carefully determine the axis usage so that it does not exceed the number of axes available, which is listed in Axis Types Overview. A reference axis is counted as one axis.

Define the Axes

Custom feedback is supported by all axis types with feedback, except differential force or differential acceleration.

To define an axis with custom feedback:

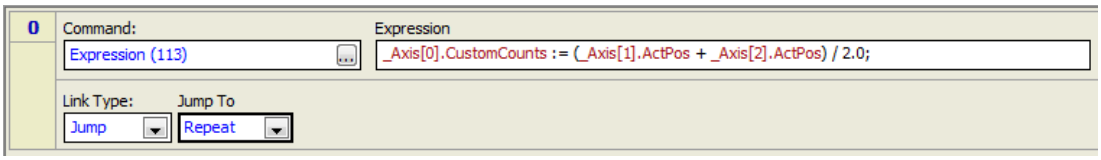
1. Begin defining an axis as usual in the Axis Definitions dialog.
2. For the feedback of the axis, choose the type you need, such as position, velocity, pressure, etc.
3. Instead of choosing a physical input, choose **Custom**.

You will also need to create reference axes for each physical input that will be used together with the Custom Feedback axis.

Make a User Program to Continuously Assigns a Value to Custom Feedback

Create a user program that continuously writes to the _Axis[].CustomCounts or _Axis[].SecCustomCounts register. The user program should consist of one step that jumps back to itself. The step should have an Expression (113) command to do the calculation.

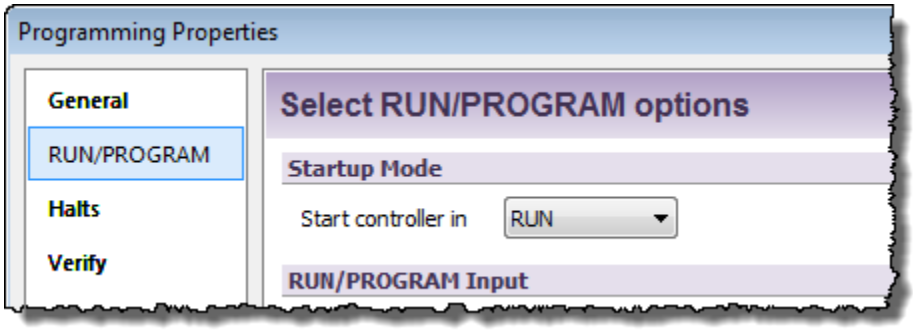
For example, this user program calculates the average of two positions and assigns it to the Axis 0 Custom Counts register. Notice that this program does not yet properly handle errors, as described in the **Error Handling** section below.



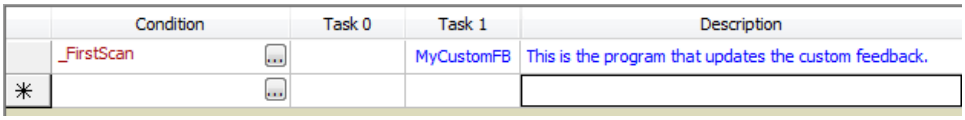
Make Sure the User Program Always Runs

This involves several steps:

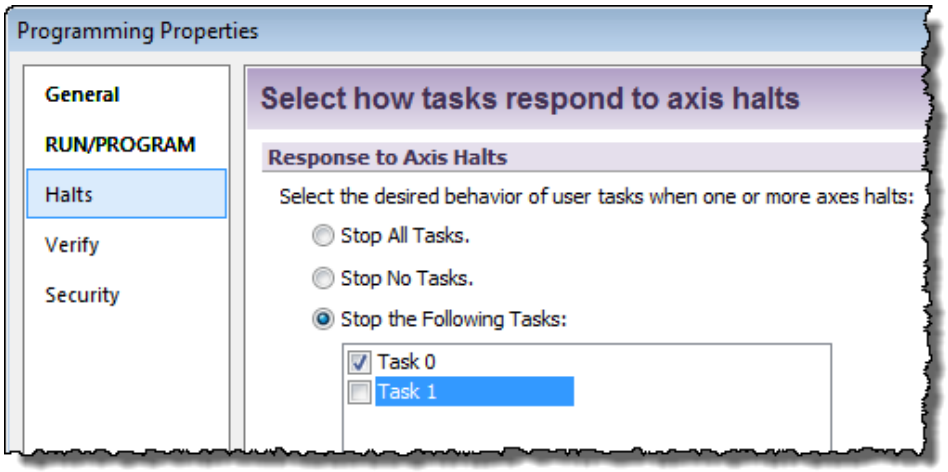
1. **Set the RMC to start in RUN mode.**
   a. In the Project Pane, right-click **Programming** and choose **Properties**.
   b. On the **RUN/PROGRAM** page, set the **Startup Mode** to **RUN**.



   c. Click **OK**.
2. **Start the user program when the RMC enters RUN mode.**
   a. In the Program Triggers, create a _FirstScan condition.
   b. In one of the task columns, choose the user program you created. Make sure no other user programs will ever run on that task.



3. **Make sure the Task does not stop when an axis halts.**
   a. In the Project Pane, right-click **Programming** and choose **Properties**.
   b. On the **Halts** page, choose **Stop the Following Tasks**.
   c. Uncheck the box for the task that will be running the user program.



Verify the Custom Feedback is Correct

Before verifying that the custom feedback value updates correctly, you will need to make sure the other feedback values used in the calculation are correct. For example, it your custom feedback calculates the average of two physical pressure inputs, make sure those pressure inputs are set up properly and operating correctly before verifying the custom feedback.

To verify the custom feedback:

a. Make sure the axis is in RUN mode, and the user program you made is running.
b. In the Axis Status Registers, on the **All** tab, expand the **Feedback** or **Pressure/Force Feedback** section.
c. The **Custom Counts** register displays the value as written to it by the user program. Making sure this value updates properly as the axis moves.

Tune the Axis

After completing all the steps above, the axis is ready to be tuned. You will need to manually tune the axis, because auto-tuning cannot be used in RUN mode.

Error Handling

An axis with custom feedback will receive a No Transducer error if any of the items below occur. The No Transducer error will then cause the axis to halt according to the No Transducer Auto Stop setting.

- The Custom No Transducer error bit is set.
- The Custom Feedback Auto-Fault Mode is violated (see **Custom Feedback Auto-Fault Mode** below).
- The Custom Counts value is NaN, Inf, or -Inf. These values can result from undefined operations, such as divide by zero.

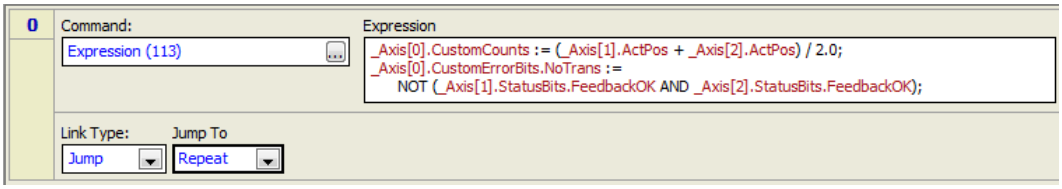
Custom No Transducer Error Bit

The **Custom No Transducer** bit in the Custom Error Bits register can be written to from the user program to provide an error indication. Setting the Custom No Transducer Error bit will result in the **No Transducer** axis error bit being set, which will then halt the axis according to the No Transducer Auto Stop setting.

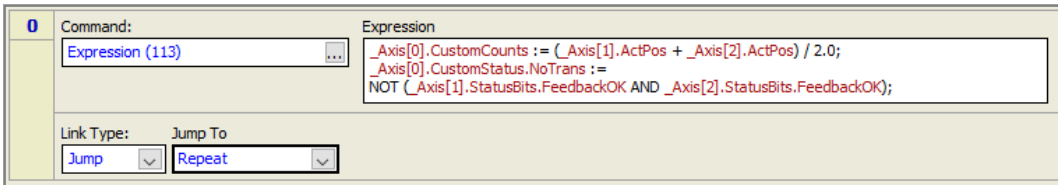
For example, in an application that controls to the average position of multiple transducers, if any of the transducers receive a transducer-related error, the control axis should halt. The user program that calculates the average position can set the No Transducer bit if any transducer errors in the reference axes have occurred. The Transducer OK axis status bit is a simple method of determining whether any transducer-related errors have occurred.

Below is a method using the Custom No Transducer Error bit. In this example, Axis 0 is the control axis with custom feedback. The feedback is calculated as the average position of the Axis 1 and Axis 2 reference axes. If the transducer of either reference axis receives an error, its Feedback OK status bit will turn off and the control axis' Custom No Transducer Error bit will be set. As soon as the errors in the reference axes clear, the Custom No Transducer Error bit will also clear, allowing the axis to perform closed-loop motion.

RMC75 and RMC150:



RMC200:



Custom Feedback Auto-Fault Mode

The Custom Feedback Auto-Fault Mode axis parameter defines certain cases in which the No Transducer or Prs/Frc No Transducer error bit will automatically be set. The following options are available:

- **Missed Update** (default setting)

  If the Custom Counts register is not written to in a given loop time, the No Transducer error will be set. This is the preferred mode for most applications. In this mode, you should make sure that the Custom Counts register is written to each loop time.

- **PROGRAM Mode**

  If the controller is in PROGRAM mode, the No Transducer error will be set. This mode may be useful in situations where you do not need to update the Custom Counts very often, but you do wish to halt the axis if the program that updates the Custom Counts is stopped due to the controller entering PROGRAM mode.

- **Disabled**

  The No Transducer error will not be set by any of the above options. In this mode, the updating of the Custom Counts register is not monitored in any way. This could be used in cases where the Custom Counts register is being modified from an external device, such as a PLC.

Additional Error Handling Considerations

In certain Custom Feedback applications cases, such as switching feedback on the fly, it may be normal for one of the reference axes to receive a transducer error. If you want the user programs to continue operating, make sure the tasks do not stop when the reference axis halts due to the error. This can be set in the Programming Properties, on the **Halts** page.

See Also

Switching Feedback using Custom Feedback | Feedback Linearization using Curves | Feedback Linearization using Mathematical Formula | Redundant Feedback using Custom Feedback

3.10.2. Switching Feedback using Custom Feedback

Switching feedback on the fly can be implemented in the RMC via Custom Feedback. This topics describes how.

Tip: The Examples section of Delta's online forum includes a Switching Feedback example. You can use that example to help you get started.

Setting Up Switching Feedback

Read the Custom Feedback topic before completing this procedure.

1. Define a Control Axis with Custom Input

a. Define a control axis with the feedback type required (position, velocity, pressure, force, or acceleration). For the feedback source, choose **Custom**.

2. Define the Reference Axes

a. Create reference axes for each physical feedback input to be used.

3. Configure the Reference Axes

a. Configure the feedback parameters for the reference axes and verify that the transducer gives valid readings.

b. Set the Scale and Offset so that the reference axes provide correct values throughout the entire range of feedback.

4. Create a Variable

Create a variable in the Variable Table to specify which feedback will be used, for example **ActiveFeedback**. The type can be either REAL or DINT.

5. Create a User Program

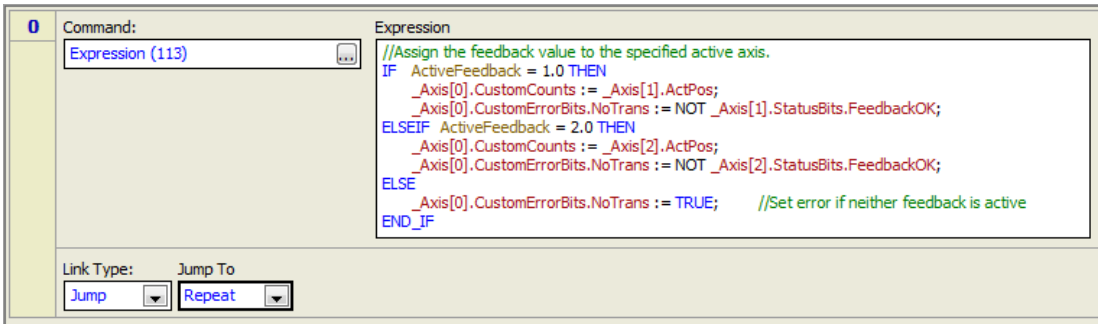
a. Create a single-step user program with a **Jump** Link Type and a **Repeat** Jump To location.

b. Add an Expression (113) command.

c. Create an expression that assigns the feedback value from the reference axes to the CustomCounts of the control axis, based on the value of the **ActiveFeedback** variable.

d. Set the CustomErrorBits.NoTrans bit according to the state of the active feedback. This can be done by using the Feedback OK status bits of the reference axes.

For example:



6. Make Sure the User Program Always Runs

As described in more detail in the Custom Feedback topic, do the following:

a. Set the RMC to start in RUN mode.

b. Use a _FirstScan condition in the Program Triggers to start the user program when the RMC enters RUN mode.

c. Make sure the task does not stop when an axis halts.

7. Tune the Axis

a. Tune the axis manually (auto-tuning does not work in RUN mode).

8. Add Other Code

- If other values need to be changed when the feedback is switched, add the necessary code to the expression. For example, some applications may require different tuning gains for each feedback.
- Depending on the application, more programming may be needed to make a smooth transition when switching feedback.

See Also

Custom Feedback

3.10.3. Feedback Linearization using Curves

This topic describes how to implement single-axis feedback linearization by using a curve to define the relationship between the transducer and the real measurement you need. For example, a curve can be used to linearize a transducer with as many points as you need. Or, consider a transducer mounted in a cylinder that drives a swing arm, but you actually need the position of the end of the arm. Given the machine geometry, a curve can be used to convert the transducer measurement to the position of the swing arm.

Another method of feedback linearization is via a mathematical formula, as described in Feedback Linearization Using a Mathematical Formula.

Tip: The Examples section of Delta's online forum includes a Feedback Linearization Using Curves example. You can use that example to help you get started.

Setting Up Feedback Linearization

Read the Custom Feedback topic before completing this procedure.

1. Determine Actual Measurement Versus Transducer Measurement

a. Determine the relationship of the desired true measurement to the transducer's measurement. This may be in the form of an equation, or measurements taken at a number of points throughout the transducer's measurement range.

b. Assemble the data as a table of data consisting of the desired true measurements and the transducer's measurements. If you have an equation, you can use a spreadsheet program such as Microsoft Excel to calculate points from the equation.

2. Create a Curve

a. Create a new curve in the Curve Tool.

b. In the curve data, enter the transducer's values in the x-axis. Enter the desired true measurements in the y-axis. If you have the data in a spreadsheet program such as Microsoft Excel, you can copy the data and paste it into the Curve Tool.

c. Make sure the x-values will cover the entire travel range of the physical feedback. If it does not, you may encounter a runtime error when you run the system.

d. In the curve properties, set the **Endpoint Behavior** to **Natural-Velocity**.

e. In the curve properties, set the **Interpolation** to either **Cubic** or **Linear**, as required by your application.

Note: It is important that the x-axis contains the transducer's values, and the y-axis contains the desired true measurements.

3. Define Custom Axis and Reference Axis

a. Define a control axis with the feedback type required (position, velocity, pressure, force, or acceleration). For the feedback source, choose Custom.

b. Create a reference axis for the feedback input to be used.

c. Configure the feedback parameters for the reference axis and verify that the transducer gives valid readings.

d. Set the Scale and Offset so that the reference axis provides correct values throughout the entire range.

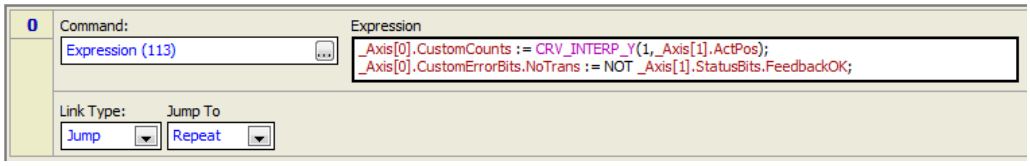
4. Use the Curve in the Custom Feedback User Program

a. Create a single-step user program with a **Jump** Link Type and a **Repeat** Jump To location.

b. Add an Expression (113) command.

c. In the expression, use the CRV_INTERP_Y function to get the curve's interpolated desired true value from the transducer's measurement.

d. Set the CustomErrorBits.NoTrans bit according to the state of the active feedback. This can be done by using the Feedback OK status bits of the reference axes.



5. Make Sure the User Program Always Runs

As described in more detail in the Custom Feedback topic, do the following:

a. Set the RMC to start in RUN mode.

b. Use a _FirstScan condition in the Program Triggers to start the user program when the RMC enters RUN mode.

c. Make sure the Task does not stop when an axis halts.

6. Tune the Axis

Tune the axis manually (auto-tuning does not work in RUN mode).

See Also

Custom Feedback | Feedback Linearization Using Mathematical Formula

3.10.4. Feedback Linearization Using a Mathematical Formula

This topic describes how to implement single-axis feedback linearization if you have a mathematical formula that describes the relationship between the transducer and the real measurement you need. For example, a transducer may be mounted in a cylinder that drives a swing arm, but you actually need the position of the end of the arm. Given the machine geometry, a mathematical formula can be used to convert the transducer measurement to the position of the swing arm.

Another method of feedback linearization is via curves, as described in Feedback Linearization using Curves. In fact, even if you have a mathematical formula, you can create a curve from the formula and use that method.

Tip: The Examples section of Delta's online forum includes a Feedback Linearization Using Mathematical Formula example. You can use that example to help you get started.

Setting Up Feedback Linearization Using a Mathematical Formula

Read the Custom Feedback topic before completing this procedure.

1. Determine Actual Measurement Versus Transducer Measurement

Determine the equation that defines the relationship of the desired measurement to the transducer's measurement. The machine designer should be able to provide a formula.

2. Define Custom Axis and Reference Axis

a. Define a control axis with the feedback type required (position, velocity, pressure, force, or acceleration). For the feedback source, choose Custom.

b. Create a reference axis for the physical feedback input to be used.

c. Configure the feedback parameters for the reference axis and verify that the transducer gives valid readings.

d. Set the Scale and Offset so that the reference axis provides correct values throughout the entire range.

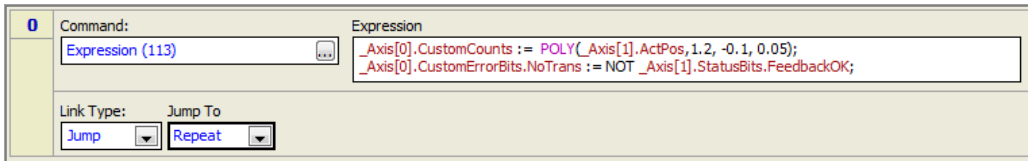
3. Create a User Program

a. Create a user program with an Expression (113) command, and **Jump** Link Type set to **Repeat**.

b. In the expression, enter the formula and assign the value to the control axis' Custom Counts register.

c. Set the CustomErrorBits.NoTrans bit according to the state of the active feedback. This can be done by using the Feedback OK status bits of the reference axes.

For example:



4. Make Sure the User Program Always Runs

As described in more detail in the Custom Feedback topic, do the following:

a. Set the RMC to start in RUN mode.
b. Use a _FirstScan condition in the Program Triggers to start the user program when the RMC enters RUN mode.
c. Make sure the Task does not stop when an axis halts.

5. Tune the Axis

Tune the axis manually (auto-tuning does not work in RUN mode).

See Also

Custom Feedback | Feedback Linearization using Curves

3.10.5. Redundant Feedback using Custom Feedback

Redundant feedback can be implemented in the RMC by using Custom Feedback. When the controller detects that one transducer has problems, it can immediately switch to another feedback on the fly, even in the middle of motion. The user has the flexibility to program the logic that determines which feedback should be used, making redundancy using any number of transducers possible.

Tip: The Examples section of Delta's online forum includes a Redundant Feedback example. You can use that example to help you get started.

Setting Up Redundant Feedback

Read the Custom Feedback topic before completing this procedure.

1. Define a Control Axis with Custom Input

a. Define a control axis with the feedback type required (position, velocity, pressure, force, or acceleration). For the feedback source, choose Custom.

2. Define the Reference Axes

a. Create reference axes for each physical feedback input to be used.

3. Configure the Reference Axes

a. Configure the feedback parameters for the reference axes and verify that the transducers give valid readings.
b. Set the Scale and Offset so that the reference axes provide correct values throughout the entire range of feedback.

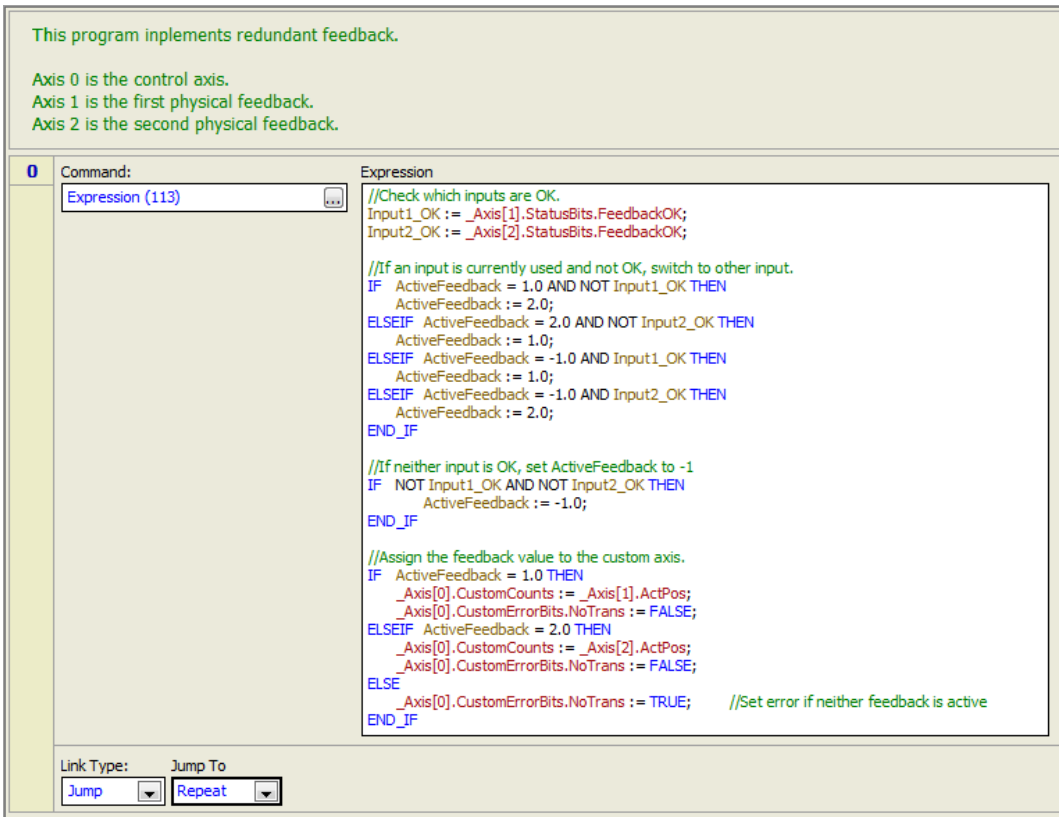
4. Create a Variable

Create a variable in the Variable Table for reporting which feedback is being used. For example, **ActiveFeedback**.

5. Create a User Program

a. Create a single-step user program with a **Jump** Link Type and a **Repeat** Jump To location.
b. Add an Expression (113) command.
c. Create an expression that implements the logic for selecting which feedback to use. For example, for a redundant system using two transducers, you may wish to:
   i. Check which feedback inputs are valid. The TransducerOK status bit is useful for this.
   ii. If a feedback input is not valid, select the other input.
d. After selecting the feedback input to be used, assign the value of the feedback to the Custom Counts (or Secondary Custom Counts for secondary inputs) register of the control axis.
e. If none of the feedback inputs are valid, make sure to set the CustomErrorBits.NoTrans bit.

For example:



6. Make Sure the User Program Always Runs

As described in more detail in the Custom Feedback topic, do the following:

a. Set the RMC to start in RUN mode.

b. Use a _FirstScan condition in the Program Triggers to start the user program when the RMC enters RUN mode.

c. Make sure the task does not stop when an axis halts. This can be set in the Programming Properties, on the **Halts** page.

7. Tune the Axis

a. Tune the axis manually (auto-tuning does not work in RUN mode).

8. Program your application

Program the rest of your application. Keep in mind that if you want other user programs to run even if one feedback transducer fails, you will need to set the Programming Properties Halts page such that the tasks running those user programs do not halt.

See Also

Custom Feedback

3.11. Applications

3.11.1. Electric Servo Motor Control

The RMC family of motion controllers excels at electric servo motion control. The RMC is a general-purpose controller. To interface with an electric motor, you will need an amplifier or drive that takes a ± 10V command signal from the RMC. Feedback can be quadrature encoder, SSI, analog voltage or current, or a resolver.

Amplifier or Drive Details

To interface with an electric motor, you will need an amplifier or drive that takes a ±10V command signal from the RMC. This includes Variable Frequency Drives.

If the drive is a smart drive, you should set it to it's simplest setting so that the drive does not do the position control. The RMC should do the position control.

Feedback

Most motors are available with encoder feedback. This is sufficient for use with the RMC. Make sure the encoder sends out a 5V differential (RS-422) quadrature signal.

If you wish to use absolute feedback, SSI is a good option.

Motor Control Features

The RMC has many features for motor control. A few of them are listed below:

Rotary Motion

Rotary-specific Command Options

Setup parameters (Position Unwind, Count Unwind)

Command Direction (Nearest, Positive, Negative, Multi-turn)

Incremental or Absolute Feedback

Linear Motion

Positive and Negative Travel Limits

Physical Limit Inputs

Incremental or Absolute Feedback

Velocity Control

Control velocity with either position or velocity feedback.

Can be used on linear as well as rotary systems.

Advanced PID

High-Order gains

Feed-forward gains (Velocity Feed Forward, Acceleration Feed Forward, Jerk Feed Forward)

See Also

Hydraulic Control | Pneumatic Control

3.11.2. Hydraulic Control

The RMC family of motion controllers excels at hydraulic motion control. The RMC includes many features for controlling hydraulic cylinders:

Directional Gain Ratio

Single-rod hydraulic cylinders require different gains when going moving in each direction. The RMC has Velocity Feed-Forward gains for each direction. The rest of the tuning gains are scaled according to the ratio of the feed-forward gains to compensate for the asymmetrical cylinder.

Deadband Compensation

Some hydraulic valves have overlapped spools which create a dead zone or deadband where no oil flows even though a control signal is applied to the valve. The RMC includes parameters to compensate for the Deadband. Though this compensation can improve performance significantly, Delta does not recommend using valves with overlapped spools – in particular for high performance applications or for pressure control.

Position-Pressure/Force Control

Hydraulic cylinders are ideal for applications where large forces must be applied to a load, and the RMC includes control modes and parameters specifically for these types of applications.

Active Damping and Acceleration Control

Active Damping uses Acceleration or force feedback to increase the stability of systems with heavy loads relative to the cylinder diameter. This can increase the performance of a hydraulic system without significant increases in cost from increasing the size of the cylinder, valve and power unit.

Specialized Commands

The RMC has several commands that allow you to obtain the highest performance possible from your hydraulic system or control difficult applications by combining Open Loop and Closed Loop control in a single move. See the Quick Move command.

See Also

Electric Servo Control | Pneumatic Control

3.11.3. Pneumatic Control

The RMC family of motion controllers excels at pneumatic motion control applications that demand precise, smooth motion.

Pneumatic Control has been serving industry for a long time and provides several advantages, not the least of which is less costly devices, clean operation and lower maintenance costs. However, precision and smooth motion has been elusive because of inherent drawbacks to pneumatics. The major drawback is the compressibility of air. This drawback has been minimized with new motion control techniques.

The RMC gives the user an option to replace start/stop or ”bang-bang” control that cause shock and wear allowing machines and equipment to last longer. This is done with motion controllers and by using feedback from position and pressure sensors. The RMC then controls the servo proportional valve to eliminate banging and bottoming out at the ends of cylinders. Use of proportional servo valves also allows a better option for accurate control in place of using a pressure relief valve. Delta recommends pressure relief valves be used for relieving pressure and not for controlling motion or pressure.

Precise midpoint positioning is also easily done with an RMC motion controller using servo valves by moving the spool to meter air in and out of a cylinder. Pneumatic motion control that is fast and smooth can make a system stiffer and act much like an elecro-mechanical system at a much less expensive cost.

High-performance motion control requires a servo valve and a high-resolution position transducer. For optimal control, a low-friction cylinder should be used. In addition, for the best control, either pressure transducers or accelerometers should be used.

Once the pneumatic system has been set up and tuned, it can be controlled much like any typical hydraulic or electric servo system.

Pneumatic Control Options

The RMC has several options for pneumatic control:

PID or I-PD with Active Damping with force feedback (load cell or pressure transducers) or accelerometers

Easy to tune without overshooting; smooth.

Force Feedback can be a load cell or differential force, which requires two pressure transducers; one on each end of the cylinder.

PID with Acceleration Control with accelerometers

This gives the tightest control. However, it really works the valve and will always overshoot slightly. Tuning hot tends to cause small high-speed vibrations and pushes stability limits.

This will generally require the use of a filter on the acceleration. Either the Low Pass filter or the Model based filter can be used, but Delta recommends trying the Low Pass filter first because it is easier to set up and does not require that an accurate model be generated by the Tuning Wizard. See the Acceleration Filter Type and Actual Acceleration Filter topics for more details.

PID with Acceleration Control without accelerometers

Much lower performance, may be good for some applications, especially for low-friction cylinders with small load. Does not require extra feedback devices or inputs on the controller. Least expensive option.

Required Modules

Active Damping and Acceleration Control require an AP2 module for the secondary acceleration or force feedback. Using Acceleration Control without accelerometers does not require any additional modules other than the axis module.

See the Active Damping or Acceleration Control topics for details on required hardware and how to define the axes.

Suggested Components

Pressure Transducers

Choose pressure transducers with a power supply range that includes 24V, and a voltage output range of 0 to 5 volts or more. The pressure transducers should be mounted on the ends of the cylinder so that they measure the pressure inside the cylinder, *not* the pressure in the A or B port lines.

Accelerometers

For pneumatic control, choose 1-axis accelerometers that cover approximately -2g to +2g, depending on the accelerations in your application. To simplify the wiring, choose one with an internal voltage regulator, a power supply range that includes 24V, and a voltage output range of 0 to 5 volts or more.

Delta recommends using two accelerometers, one on the stationary frame and one on the moving load. This is because the stationary frame is not usually truly stationary – its vibrations will impact the motion of the load.

Delta has successfully used accelerometers from Crossbow Technology Inc. (www.xbow.com). The model used was the LP series - General Purpose, part number CXL04LP1-R, which is a ± 4g, 1-axis accelerometer with an internal voltage regulator. It can be powered with a 24V power supply.

Pneumatic Servo Valves

Choose a zero-lap servo valve.

Pneumatic Cylinder

For best results, choose a low-friction cylinder.

See Also

Hydraulic Control | Electric Servo Control | Tuning a Pneumatic System

3.12. Transducers Basics

3.12.1. MDT Transducer Fundamentals

Magnetostrictive Displacement Transducers (MDT) are absolute position transducers designed for use in rugged industrial environments. Early MDT sensors employed the Start/Stop or Pulse Width Modulated (PWM) signals. Delta generally refers to these signal types as MDT signals. Magnetostrictive Displacement Transducers can also have other types of outputs, such as analog or SSI.

MDTs are non-contact, wear-free, highly reliable, and offer accurate and repeatable linear position measurement. In the motion control industry, magnetostrictive displacement transducers are typically inserted into hydraulic cylinders for measurement of the cylinders position.

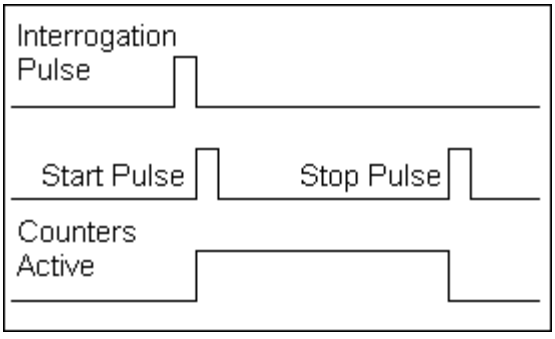
MDT feedback (Start/Stop or PWM) is supported by the following modules:

- **RMC75:** MA axis module
  Each axis on the RMC75 MA axis module can be individually configured for MDT or SSI inputs.
- **RMC150:** MDT module
- **RMC200:** S8 Module, U14 Module

Each MDT axis can be configured for a Start/Stop transducer or a Pulse Width Modulated (PWM) transducer.

Start/Stop

To make a measurement with a Start/Stop transducer, the RMC sends an interrogation pulse to the transducer. The transducer responds by returning 2 pulses—a Start pulse and a Stop pulse. The RMCs internal counters begin to count when the first pulse, Start, is received and stop counting when the second pulse, Stop, is received. The time between the start pulse and the stop pulse is proportional to the transducer position.



Start/Stop Pulse Transducer

Blanking Period (For Neuter Outputs)

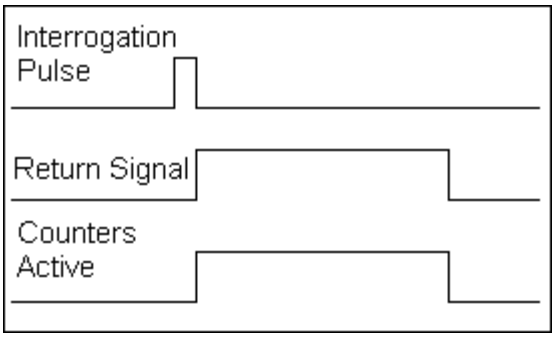
After the RMC receives the start pulse, it waits a brief amount of time before looking for the stop pulse. This time, called the MDT blanking period, gives time for noise to settle out.

The RMC150 and RMC200 have an MDT Blanking Period parameter, with options of 5µsec or 21µsec. The default value is 5µsec. Some older transducers, such as Temposonics I and II with neuter outputs, require the longer blanking period of 21µsec, due to noisy signals.

The RMC75 does not have an MDT Blanking Period parameter and always uses a 5µsec blanking period. Therefore, Temposonics I and II transducers with neuter outputs should not be used with the RMC75, but can be used with the RMC150 or RMC200.

PWM

To make a measurement with a Pulse Width Modulated transducer, the RMC sends an interrogation pulse to the transducer. The transducer responds with a return signal. The return signal is high while the transducer is determining its position. The RMC counts during the time that the return signal is high. The time that the return signal is high is proportional to the transducer position.



Pulse Width Modulated Transducer

The value obtained from the PWM or Start/Stop counter is put in the Raw Counts register for that axis. The Raw Counts are converted to Counts and then into an Actual Position in user-defined units.

Resolution

The resolution of MDT sensors depends on the RMC's clock speed for timing the pulses. For MDT Start/Stop or PWM feedback with a typical calibration value of approx. 9 µs/in, the maximum resolution supported by the RMCs is as follows:

Module	Maximum Resolution with one recirculation
RMC75 MA	0.0005
RMC150 MDT	0.001
RMC200 S8	0.0005
RMC200 U14	0.0005

Recirculations

Delta does not recommend using recirculations on MDT transducers. Recirculations have historically been used to gain more accuracy in the measurement. However, with faster counters on the controller, recirculations have become less important. In addition, the highest accuracies are only available for MDT transudcers with SSI ouput, which don't have any recirculation options.

Transducer Lengths

An MDT transducer takes a certain amount of time to determine its position. The time it takes increases with position. Therefore, the length of the transducer dictates the minimum Loop Time you can use.

The approximate formula for determining the maximum transducer length for a given loop time is as follows:

Length(in.) = (LoopTime(ms) - 0.22)/0.0095

Using the formula, the following are approximate maximum lengths for a given loop time. The actual maximum length will vary slightly by RMC model. This assumes no recirculations.

Loop Time	Maximum MDT Transducer Length
125µs[1]	5 inches[2]
250µs	11 inches[2]
500µs	29.4 inches
1000µs	82.1 inches
2000µs	187.3 inches
4000µs	397.9 inches

[1]125µs Loop Time only available on RMC75 and RMC200 series.

[2]Estimated max transducer length

Start/Stop and PWM Cable Length

For magnetostrictive sensors with Start/Stop or PWM signals, there is no set maximum cable length. Modern sensors use RS-422 for these signals, and can typically work with cables lengths of up to 4000 ft, using proper twisted-pair, shielded cable. Older, single-ended signals cannot transmit as far.

See Also

Feedback Resolution | Wiring Guidelines

3.12.2. SSI Fundamentals

Synchronous Serial Interface (SSI) is a widely accepted controller interface. Position data from the sensor is encoded in a binary or Gray Code format and transmitted over a high-speed serial interface. Many types of transducers are available with SSI, including magnetostrictive displacement transducers (MDTs), absolute encoders, and laser measuring devices.

SSI has a number of advantages over other transducer interfaces:

- High noise immunity
- Absolute position
- Supports a wide variety of transducers.
- Many SSI devices offer higher precision; for example, magnetostrictive transducers with SSI output are commonly available with resolutions to 1 µm, and some offer 0.1 µm.

Important!
The RMC SSI inputs require RS-422 (5V differential) SSI inputs. They do not support single-ended SSI inputs or higher voltage inputs.

SSI feedback is supported by the following modules:

- **RMC75:** MA axis module
- **RMC150:** SSI Module, Universal I/O module
- **RMC200:** S8 Module, U14 Module

The RMCs provide the following SSI options:

SSI Options	RMC75 MA Module	RMC150 SSI Module	RMC150 UI/O Module	RMC200 S8 Module	RMC200 U14 Module
Data Bits	8 to 32	8 to 31	8 to 32	8 to 32	8 to 32
SSI Format	Binary or Gray Code	Binary or Gray Code	Binary or Gray Code	Binary or Gray Code	Binary or Gray Code
SSI Errors	None, all zeros, all ones, or bit 21	None, all zeros, all ones, or bit 21	None, all zeros, all ones, or bit 21	None, all zeros, all ones, or bit 21	None, all zeros, all ones, or bit 21
Clock Rates	150, 250, 375 kHz	230, 921kHz	250, 500, 971kHz	100, 150, 250, 400, 625, 1000, 1500, 2500 kHz	100, 150, 250, 400, 625, 1000, 1500, 2500 kHz
Wire Delay	n/a	n/a	available	not yet available	not yet available

SSI Modes on the RMC

A standard SSI interface on an RMC consists of a Clock Output and Data Input. The RMC sends a clock signal to the SSI device, and the SSI device returns a data signal while the RMC is clocking.

Some RMC modules support additional SSI modes. These modes are:

- **SSI Monitor**
  In SSI Monitor mode, the RMC SSI interface consists of a Clock Input and a Data Input. This allows the RMC to listen to the SSI traffic between a separate controller and a device.
- **SSI Device**
  In SSI Device mode, the RMC SSI interface consists of a Clock Input and a Data Output. This allows the RMC to act like an SSI device. That is, when another controller sends a clock signal to the RMC, the RMC will send the data.
- **SSI Output**
  In SSI Output mode, the RMC SSI interface consists of a Clock Output and a Data Output. This allows the RMC to send data to an SSI Monitor. This can be used to send data from an RMC150 UI/O module to another controller's SSI Monitor input.

These modes are supported by configuring the modules listed below as follows:

Mode	RMC75 MA Module	RMC150 SSI Module	RMC150 UI/O Module	RMC200 S8 Module	RMC200 U14 Module
SSI Input	✓	✓	✓	✓	✓
SSI Monitor			✓	✓	✓
SSI Device			✓		✓ *
SSI Output			✓		

*Via the SSI Echo mode, which echoes the channel 0 data onto channel 1, with channel 1 as an SSI device.

Configuring the SSI Modes

Follow the steps below to set up each SSI mode:

RMC150 UI/O Module

Follow the instructions in Configuring UI/O High-Speed Channels to set up a channel, with the follow settings for the desired SSI mode:

- **SSI Monitor:** Choose **SSI Axis Input** or **SSI Register Input**. Then, in Axis Parameters, on the **All** tab, expand **Feedback**, and set the **SSI Clock Mode** to **Monitor**.
- **SSI Device:** Choose **SSI Output** and set the **SSI Output Mode** to **Slave**.
- **SSI Output**: Choose **SSI Output** and set the **SSI Output Mode** to **Master**.

RMC200 S8 Module

The only mode other than the standard SSI mode is the **SSI Monitor** mode that uses inputs 6 and 7.

1. In the Project Pane, expand the **Modules** folder.
2. Double-click the desired S8 module and choose **Configuration**.
3. Select **One SSI Monitor Input**.
4. Click **OK**.

RMC200 U14 Module

Follow the instructions in Configuring U14 High-Speed Channels to set up a channel, with the follow settings for the desired SSI mode:

- **SSI Monitor:** Choose **SSI Monitor input**.
- **SSI Echo:** Set Channel 0 to **SSI/MDT input**, and set Channel 1 to **SSI Echo output**.

For more details on configuring, see the Configuring UI/O High-Speed Channels, Configuring U14 High-Speed Channels, and Configuring S8 Channels.

Synchronized SSI for Linear Magnetostrictive Transducers

When using magnetostrictive SSI transducers, it is highly recommended that a **synchronized** SSI transducer be selected. This ensures that the time between position samples matches the control loop time of the RMC controller. If the transducer is not synchronized, the sample time may not match and will adversely affect control. When the sample time does not match the loop time it introduces jitter to the position measurement which gets worse as the axis speed increases. Make sure to specify that the transducer be of the synchronized type.

Synchronized SSI is not an issue for rotary encoders.

SSI Advantages

SSI transducers and absolute encoders offer the following advantages:

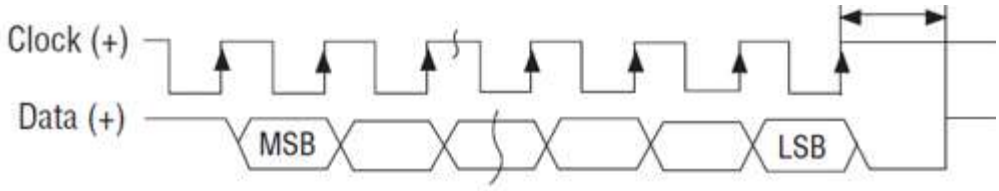
- High resolution. Down to 0.1 µm (approx. 0.000004") for linear SSI transducers.
- Noise immunity
- Cost effective data transfer (only one 5-wire cable with shield is needed)
- Transmission rate independent of data length and resolution
- Transmission over long distances
- Direct connection to the RMCs SSI axis module

Data Format

To read an SSI position, the RMC sends clock pulses to the transducer and the SSI device returns the data as follows:

1. The SSI channel sends the first clock pulse by setting the Clock signal low, then high.
2. On the first rising edge of the Clock signal, the SSI transducer returns the most-significant bit of the data on the Data line.

3. The SSI channel sends the second clock pulse by setting the Clock signal low, then high. When the Clock signal goes high, the SSI channel samples the bit on the Data line. When the SSI device sees the clock signal go high, it places the second bit of data on the Data line.
4. This continues until all bits have been clocked and sampled.
5. The value obtained from the SSI data is put in the Raw Counts register for that axis. The Raw Counts are converted to Counts and then into an Actual Position in user-defined units.



SSI Cable Length

The maximum allowable SSI cable length depends on the SSI Clock Rate**.** For SSI inputs on the UI/O module, wire delay compensation is available to allow longer lengths, as described in the **Wire Delay Compensation** section below.

Clock Rate	Maximum Cable Length*
100 kHz	2100 ft (640 m)
150 kHz	1360 ft (415 m)
230 kHz	850 ft (255 m)
250 kHz	770 ft (235 m)
375 kHz	475 ft (145 m)
400 kHz	450 ft (135 m)
500 kHz	325 ft (99 m)
625 kHz	225 ft (70 m)
921 kHz	120 ft (37 m)
971 kHz	110 ft (34 m)
1000 kHz	100 ft (30 m)
1500 kHz	25 ft (7.5 m)
2500 kHz	3 ft (1 m)

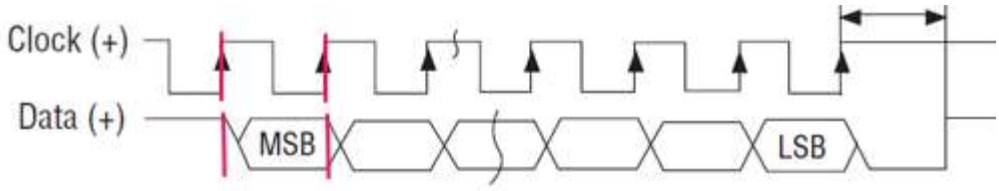
* The cable lengths are approximate, and may be affected by the type of wire and transducer.

Wire Delay Compensation (RMC150 UI/O Module Only)

Wire delay compensation is available on the RMC150 Universal I/O Module and is required for SSI wire runs that exceed the lengths given in the **SSI Cable Length** section above. If the wire to the SSI device is very long, there will be a significant delay between the clock signal and the returned data signal. As shown in the diagrams below, if this delay exceeds one clock period, the RMC will not receive the correct data, unless the SSI Wire Delay parameter is used.

Minimal Delay

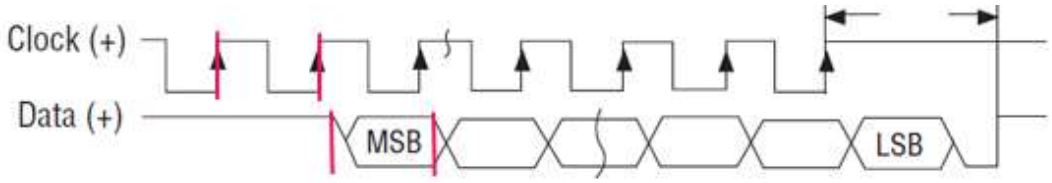
The timing diagram below shows an SSI system with very little delay. On the first rising edge of the Clock, the SSI device puts the first bit of data on the Data line. By the next rising edge of the Clock, when the RMC samples the data, the data is valid, and the read is successful.



Excessive Delay

The timing diagram below shows an SSI system with a time delay of more then one clock period. On the first rising edge of the Clock, the SSI device puts the first bit of data on the Data line. By the next rising edge of the Clock, when the RMC samples the data, the data from the SSI device has not yet arrived, and the SSI input will not return the correct value.

To compensate for the delay, set SSI Wire Delay parameter. You can enter the wire length or enter the time delay directly. The SSI input will then use the delay value to correctly read the SSI input data.



Transducer Lengths

Magnetostrictive linear transducers with SSI output may require a minimum time between interrogation based on the length of the transducer. Check your transducer data sheet for details. Increasing the Loop Time of the RMC will increase the maximum allowed length of the transducer. See the MDT Fundamentals topic for details.

Setting up Axes with SSI Feedback

To set up axes with SSI feedback, read the following topics:

- Wiring: RMC150 SSI Wiring, RMC150 UI/O Wiring, RMC75 MA Wiring, RMC200 S8 Wiring, RMC200 U14 Wiring
- SSI Scaling

Note:
The RMC150 UI/O and RMC200 U14 high-speed channels must be configured as SSI before being used as SSI inputs.

The following parameters must be also set for axes with SSI feedback:

	RMC75 MA	RMC150 SSI	RMC150 UI/O	RMC200 S8	RMC200 U14
Required					
SSI/MDT Feedback Type	✓			✓	✓
SSI Data Bits	✓	✓	✓	✓	✓
SSI Format	✓	✓	✓	✓	✓
Application-specific					
Linear/Rotary	✓	✓	✓	✓	✓
SSI Clock Rate	✓	✓	✓	✓	✓
Wire Break Detection		✓		✓	✓
SSI Overflow Mode	✓	✓	✓	✓	✓
SSI Home Source	✓				
SSI Clock Mode			✓		

SSI Termination			✓	✓	✓
SSI Wire Delay			✓		
SSI High Bits to Ignore				✓	✓
SSI Low Bits to Ignore				✓	✓

See Also

Feedback Resolution | Wiring Guidelines

3.12.3. Analog Fundamentals

"Analog" in the RMC refers to analog voltage or current.

Analog Inputs

Analog voltage or current feedback coming to the RMC is sampled multiple times per loop time. At each sampling, the voltage or current is converted to a number by an ADC (Analog to Digital Converter). The samples are then averaged to produce one number per loop time. See the data sheets of each analog feedback module for details on the resolution of the analog-to-digital conversion.

The following modules support analog feedback from a transducer or potentiometer:

Module	±10 V	±5 V	4-20 mA	±20 mA
RMC75				
AA1	✓		✓	
AA2	✓		✓	
A2	✓		✓	
AP2	✓		✓	
RMC150				
Analog (H)	✓	✓	✓	
Analog (G)	✓			
Analog Inputs (A)	✓	✓	✓	
Universal I/O	✓		✓	
RMC200				
A8	✓		✓	✓
U14	✓		✓	✓

Broken Wire Detection

The RMC analog inputs incorporate internal biasing so that when an input is not connected, the input voltage will be pulled down to its full negative value.

On the RMC75 and RMC150, a broken wire will cause the Transducer Overflow error bit for the associated axis to turn on.

On the RMC200, a broken wire will cause the No Transducer error bit for the associated axis to turn on.

Analog Outputs

The following modules have analog outputs, normally intended for use as a Control Output:

Module	±10 V	4-20 mA	±20 mA
RMC75			
AA1, AA2	✓		
MA1, MA2	✓		
QA1, QA2	✓		
RMC150			
Analog (H)	✓		
Analog (G)	✓		
MDT (M)	✓		
Quad (Q)	✓		
Resolver (R)	✓		
SSI (S)	✓		
RMC200			
CA4	✓	✓	✓
CV8	✓		
U14	✓	✓	✓

See Also

Feedback Resolution | Wiring Guidelines

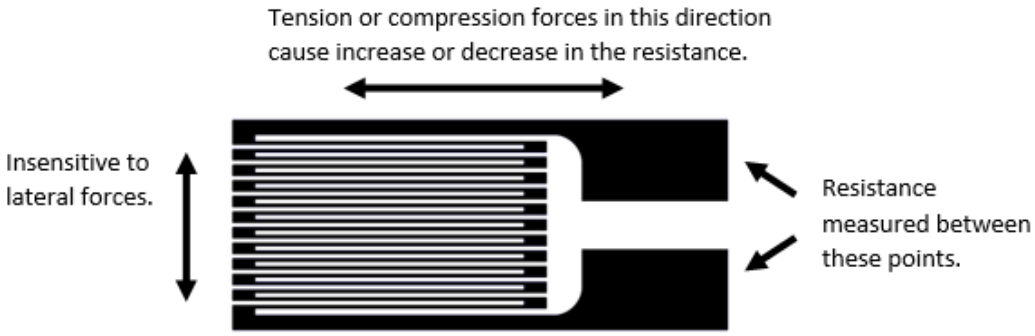
3.12.4. Load Cell and Wheatstone Bridge Fundamentals

The RMC200 LC8 module supports load cells made of strain gauges in a Wheatstone bridge configuration. This topic briefly describes these concepts. More complete information is readily found from other sources.

Strain Gauge

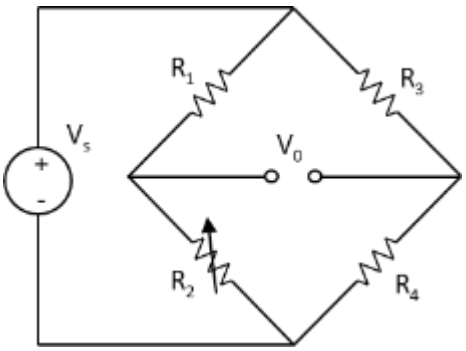
A strain gauge is an electrical sensor that measures strain (deformation of a material) caused by stress (force on a material). It consists of a resistive element, such as a wire, bonded to or printed on the surface of a material. As the material stretches or compresses, the length and cross-sectional area of the resistive element change, resulting in a change in resistance. This change in resistance can be used to determine the change in force.

The change in resistance is typically very small and can be difficult to measure. A long wire improves the measurability. To fit a long wire into a small area, the wire is bent in a zigzag pattern with long parallel lines in the direction of the strain to be measured.



Wheatstone Bridge

A typical method of accurately measuring a small change in resistance is to use a Wheatstone bridge. Wheatstone bridges are used in many applications, including measuring strain. The bridge circuit consists of two voltage dividers wired in parallel, with a common voltage source:

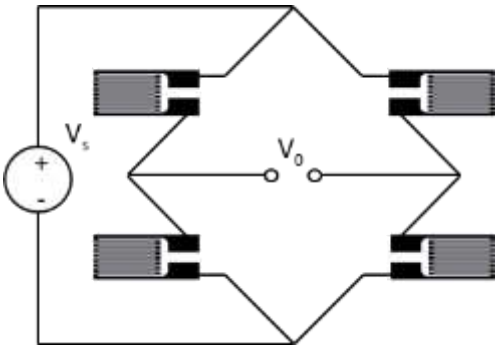


The supply voltage V_s is a constant dc voltage. The output voltage is measured at V_0. If all the resistances are equal, V_0 is zero, and the circuit is said to be balanced. If one resistance, such as R_2, changes, V_0 will change.

Load Cell

A load cell consists of strain gauges in a Wheatstone configuration, bonded to elements that are designed to bend as a load is applied. There are many possible permutations of the number of strain gauges, their orientations, the geometry of the load cell, and the strain gauge locations in the load cell.

A load cell with four strain gauges is the most common and is called a full bridge:



Load Cell Output Signal

The change in V_0 is typically very small, on the order of millivolts. The voltage change is also proportional to the supply voltage, V_S. Therefore, the output V_0 of a load cell made of a Wheatstone bridge is specified as the number of millivolts per volt of supply voltage at the rated load of the load cell.

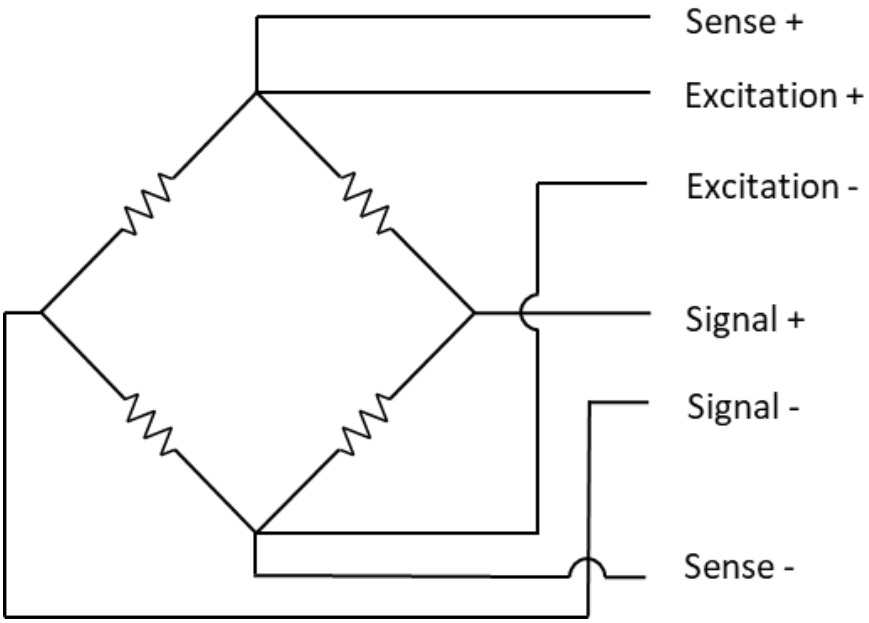
For example, a load cell output may be specified as 2 mV/V. That means for a supply voltage of 6.75 V, the output of the load cell at its maximum rated load will be 2 mV x 6.75 V = 13.5 mV. If the load cell is compression only or tension only, as the load changes, the output of the load cell will vary between 0 and 13.5 mV, proportional to the load. If the load cell works in both compression and tension, as the load changes, the output of the load cell will vary between -13.5 mV and 13.5 mV, proportional to the load.

Typically, any load cell that has an output specified in mV/V is compatible with the RMC's load cell inputs, as long as the maximum output voltage V_0 is less than the RMC's maximum input rating.

The LC8 module excitation voltage is 6.75 V. However, an external excitation may be applied to the load cell instead. Therefore, if the 6.75 V excitation will cause the load cell signal to exceed the LC8 input specification, a lower external excitation may be used to reduce the load cell signal to within the limit of the RMC's voltage input. Conversely, a higher excitation voltage may be used (up to 10V) to increase the output from the load cell as long as the maximum voltage output of the load cell does not exceed the input limit. This can improve signal to noise ratio in particularly sensitive applications.

Wire Sense for Load Cells

Achieving accurate measurements from a load cell requires knowing the precise voltage that is being applied to the load cell. The controller supplies a fixed DC voltage. The resistance in the supply and return wires will reduce that voltage by an amount dependent on the wire length and gauge. This effect can be significant for long wire runs. Therefore, many load cells add two remote sensing wires so the controller can measure the voltage that is actually being applied at the load cell and use that voltage in the calculation of the mV/V signal. This provides a more accurate measurement.

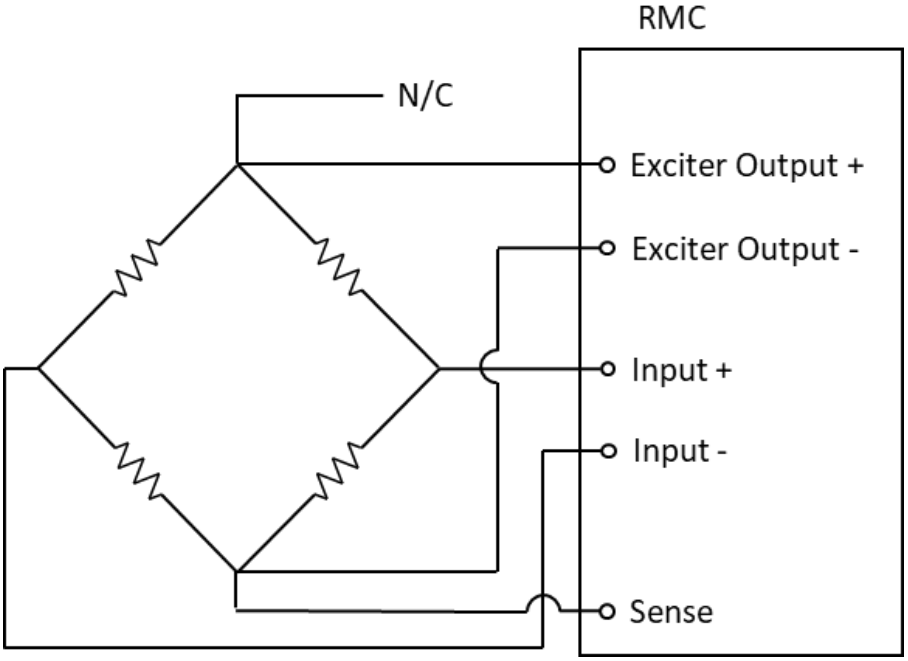


Wire Sense in the LC8 Module

To measure the supply voltage at the load cell, the LC8 module offers a single Wire Sense input. This measures the voltage of the Sense- wire relative to the zero reference of the excitation

voltage (Exc-). The RMC assumes that the voltage drop from the supply voltage output (Exc+) to the load cell is identical to the voltage drop from the load cell to Exc-. Therefore, the Exc+ and Exc- wires must be of the same length and gauge to use the Wire Sense feature.

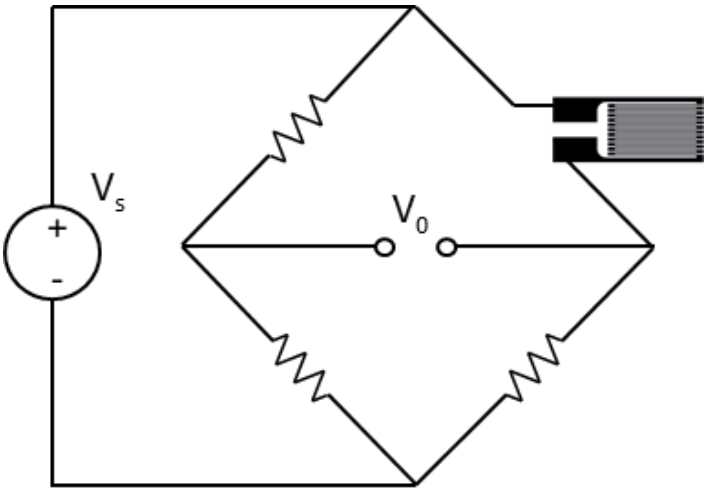
With the LC8 module, one of the sense wires in 6-wire load cells will be unused:



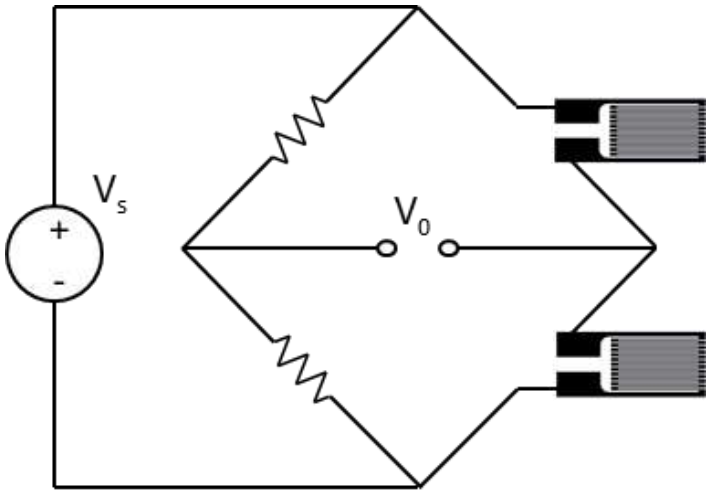
Using Individual Strain Gauges

Instead of using a complete load cell with a full Wheatstone bridge, some applications may use a single strain gauge or two strain gauges. In these cases, a bridge completion circuit must be used to provide a complete Wheatstone bridge for the RMC's load cell input. The individual resistances of the Wheatstone bridge must be matched to the strain gauge resistance. Common resistance values are 120 or 350 Ohms. Commercially available bridge completion modules typically offer an adjustment to zero the bridge output. See the **Third Party Recommendations** section of Delta's forum for commercially available bridge completion modules.

A single load strain gauge requires three resistors to complete the bridge. This is called a quarter bridge:



Two strain gauges require two resistors to complete the bridge. This is called a half bridge:



See Also

Feedback Resolution | Wiring Guidelines | LC8 Wiring | LC8 Module (RMC200) | Load Cell Scaling

3.12.5. Quadrature Encoder Fundamentals

Quadrature encoders are one of the most common types of encoder. They are often referred to as A quad B, or incremental encoder. Quadrature encoders are incremental, meaning that they can accurately measure changes in position, but cannot provide an absolute position.

Quadrature encoders are often used on motors. They are also commonly used to provide a reference input, such as the position of a belt, or the position of a board on a flying cut-off application.

Quadrature feedback is supported by the following modules:

- **RMC75:** QA axis module, Q1 expansion module
- **RMC150:** Quadrature (Q) module, Universal I/O module
- **RMC200:** Q4 Module, S8 Module, D24 Module, U14 Module

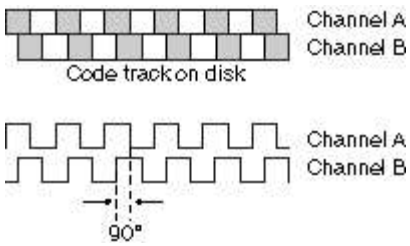
Delta recommends an RS-422 line driver output for quadrature encoders, as it provides the highest speed and very good noise immunity. The TTL and HTL input types supported by the D24, Q4 and U14 are intended for retrofit applications where an existing encoder cannot easily be changed to RS-422. RS-422 is also commonly referred to as a 5V differential signal (although that is not the strict definition).

Operation

A and B Signals

Quadrature encoders use two output channels (A and B) to indicate position. Using two code tracks with sectors positioned 90 degrees out of phase, the two output channels of the quadrature encoder indicate both position and direction of rotation. If A leads B, for example, the disk is rotating in a clockwise direction. If B leads A, then the disk is rotating in a counter-clockwise direction.

By monitoring both the number of pulses and the relative phase of signals A and B, both the position and direction of rotation can be tracked.



The resolution of quadrature encoders is typically given in Pulses per Revolution (PPR), also called Lines per Revolution. The RMC feedback sees each rising or falling edge of the A or B signal as one count. Therefore, each pulse or line of the encoder will result in four counts on the RMC feedback. For example, a 1000 PPR encoder will give the RMC 4000 counts per revolution.

The RMC increments the counts register when A leads B. It decrements the counts register when B leads A.

Index Pulse

Some quadrature encoders also include a third output channel, called a zero or index or reference signal, which supplies a single pulse per revolution. This single pulse is used for precise determination of a reference position. The RMC75 QA, RMC150 Quad, and RMC200 Q4, U14, and D24 modules support an Index (Z) pulse input; the RMC75 Q1 and RMC200 S8 modules do not.

When using the Index (Z) pulse for homing, the Index (Z) Home Location must be set to specify the edge of the quadrature A signal on which a Z home is triggered. See the Homing topic for details.

Homing Quadrature Axes

Quadrature encoders do not provide absolute position. To use a quadrature encoder in a position application, a known reference position must be established. This position is referred to as the **Home** position. The process of moving the axis to find the home position is called **Homing**. Once the Home position is established, the encoder increments or decrements the position from the initial home position as it rotates.

There are several methods of homing. Typically, it involves moving the axis to a proximity switch that then sends a signal to the RMC. The RMC sets the position at that precise point to whatever value the user requested. The index pulse can also be used for homing.

For details on homing, see the Homing topic.

Registration

Registration is the process of recording the precise position of an axis when an external event occurs. Registration is commonly used for measurements. For example, consider a photo-eye set up on a conveyor belt, where the position of the belt is measured with a quadrature encoder. By registering the precise position at which the photo-eye is broken and the precise position at which it is reconnected, the length of a widget on the belt can be accurately determined.

The RMC provides registration on quadrature feedback axes only. This is because quadrature feedback is the only type of feedback that can record a position at any given moment, independent of the loop time of the RMC. All other feedback types can only report the position once each loop time (500µs to 4000µsec).

For more details, see Registration.

See Also

Feedback Resolution | Wiring Guidelines

3.12.6. Resolver Fundamentals

Resolvers are absolute rotary position transducers. They are very simple and robust, can be very accurate, and, in some cases, inexpensive. A wide variety of resolvers are available in many configurations and for various applications.

Resolver feedback is supported by the RMC150 Resolver (R) and Resolver (RW) modules.

The RMC150 Resolver module supports the following resolver options:

Resolver Options	Resolver (R) - Standard	Resolver (RW)
Resolution	14 or 16 bits	14 or 16 bits
Reference Frequency	800Hz to 5kHz (generated by RMC)	400Hz (externally generated)
Reference Amplitude	1.42 to 4.80 V RMS (generated by RMC)	26 V RMS (externally generated)

Contact Delta for other options.

How Resolvers Work

Resolvers are rotary transformers with one primary winding and two secondary windings. The primary winding is generally on the rotor and the two secondary windings are on the stator. The secondary windings are arranged 90 degrees from each other such that when one is lined up with the rotor winding (full coupling) the other is at a right angle (no coupling).

The primary winding is driven with an alternating current signal at a specified voltage and frequency. This signal is called the *Reference Signal* and is generated by the RMC. The position measurement of the resolver is determined by the ratio of the amplitudes of the signals on the secondary windings and their phase with respect the signal on the primary winding (the reference signal).

Important Resolver Specifications

Resolvers are commercially available in many varieties with different specifications. The primary specifications of interest as applied to the RMC are:

- **Frequency**
  The frequency of the Reference Signal driving the primary winding.
- **Voltage**
  The specified voltage of the Reference Signal driving the primary winding.
- **Output Voltage**
  The output voltage from the secondary windings or the transformation ratio between primary and secondary windings.

The RMC Resolver module is designed to work with only a subset of all resolver types available. The Resolver Module topic lists the range of values compatible with the standard RMC Resolver module. In addition, a wider range of resolvers can be used by modifying components on the interface card module or by using external reference generators to drive the primary winding. Contact Delta for more details.

Setting Up the Resolver Module

1. Wire the resolver according to the RMC150 Resolver Wiring topic.
2. Set the following parameters:
   - Resolver Resolution
   - Reference Amplitude
   - Reference Frequency
3. Scale the axis. Typically, resolvers will use Rotary Scaling. Notice that one revolution of the resolver will always consist of 65,536 counts on the RMC, regardless of the resolver resolution parameter.

See Also

Feedback Resolution | Wiring Guidelines

3.13. Other

3.13.1. Controller Image Upload/Download

The **controller image** refers to the entire RMC project as it is stored in RAM memory in the RMC. Only RMCTools can create the controller project (image) and download it to the RMC. However, other devices such as a PLC or PC, or the RMC itself, can read the RMC image via Ethernet or USB, store it, and later download it to the RMC. For the RMC200, the image can also be stored on the SD card.

The Image Upload/Download feature is supported only on the RMC75E, RMC150E, and RMC200. The controller image upload/download is an advanced feature and is not necessary in most applications.

The controller image can only be downloaded to an RMC that matches the hardware configuration of the image. Downloading and applying a controller image typically takes between 3 and 20 seconds, depending on the options selected.

Uses for storing and downloading the controller image include:

1. **Know the exact contents of the RMC**
   Allowing the PLC to control the contents of the RMC removes any problems due to tampering with an RMC. For example, when a machine starts up, the PLC may write the image to the RMC so that it knows the RMC has the correct project.

2. **Replace an RMC without using RMCTools**
   If an RMC needs to be replaced, RMCTools is not needed if the PLC or RMCLink will download the controller image, or if the image is stored on the RMC200 SD card. This may simplify machine repair when an experienced RMC user may not be available.
3. **Make more user programs available to the user**
   If an application requires more user programs than will fit in the RMC, and only a few user programs are used at any given time, this feature allows splitting up the user programs between several images. When a certain user program is required, the respective image is downloaded to the RMC, then the user program can be run.

Image Size

The size of the image varies based on the specific configuration, especially the amount of user programming and curves included. The Image Upload/Download supports the sizes listed below. Typically, the image size is much smaller, unless the controller includes large curves, in which case the image may exceed the maximum image size, which is not supported by the Image Upload/Download.

RMC	Max Image Size (32-bit registers)	Typical Image Size (varies greatly, and can be much larger with curves)
RMC75E	65,000	4,000
RMC150E	65,000	4,000
RMC200	1,575,949	11,000

Uploading an Image into a PLC

Refer to the **Image Area** section below for the registers referred to by this procedure.

1. **Preparation**

- Make sure the RMC contains the desired project and that it is saved to Flash.
- Verify that the project's IP address and subnet mask are configured correctly, otherwise communication may be lost after the image is later applied.
- Connect the PLC to the RMC via Ethernet.

2. **Request the RMC to Build the Configuration Image**
   The PLC must first request the RMC to construct an image that can be uploaded into the PLC. This is done by writing **Build Upload Image (1)** to the **Image Area Command (%MDx.0)** register.
3. **Wait for the Image to be Built**
   The RMC may take several seconds to build the requested image. During this time, the PLC should repeatedly read the **Image Area State (%MDx.1)** register and wait for the state to be **Upload Image Built (2)**.
4. **Determine the Length of the Image**
   The length of the image, in 32-bit registers, can now be read from the **Image Size (%MDx.2)** register. The PLC can use the contents of this register either to control the length of the image upload, or to simply verify that the image will fit in the space provided on the PLC.
5. **Read the Image from the RMC Controller**
   Read the image from the RMC Controller by repeatedly reading from the **Image Data (%MDx.4-…)** registers until the entire image has been transferred.

   For example, to read a 8000-register image, the PLC could read 4000 registers twice, starting each read at %MDx.4. Or, if the PLC is limited to 1000 registers per read, then the PLC could read 1000 registers eight times, again starting each read at %MDx.4. Both will result in loading the same 8000-register image. The PLC is responsible for assembling the results from each read into a single image in its storage area.

The PLC may choose to include the Current Index (%MDx.3) register in each read. This value can be used to verify that the PLC and RMC agree on the index for the current block of data.

Downloading an Image from a PLC

Refer to the **Image Area** section below for the registers referred to by this procedure.

1. **Preparation**
   - This process will restart the RMC. Therefore, make sure the RMC is in a state where restarting it will not cause problems. For example, you may wish to put the axes in open loop and the RMC in PROGRAM mode.
   - Connect the PLC to the RMC via Ethernet.
2. **Reset the Image Area.**
   Write **Reset Image Area (3)** to the **Image Area Command (%MDx.0)** register.
3. **Write the Image to the RMC Controller**
   Write the image to the RMC Controller by repeatedly writing to the **Image Data (%MDx.4-…)** registers until the entire image has been transferred. Each subsequent write must begin at the start of the **Image Data** array. The RMC will reassemble the data from each write into a single image.

   The PLC may choose to include the **Current Index (%MDx.3)** register in each write. The RMC will verify that the index value written matches the next expected image index. If the index value is incorrect, then the **Image Area State (%MDx.1)** will change to **Download Sequence Error (22)**.

   The PLC can choose to write a fixed-length image to the RMC—as long as the image is known to fit within that size—or it can adjust the length of the image it writes based on the actual length of the image, which is found in the first register of the stored image.
4. **Trigger the RMC to Apply the Image**
   The PLC must then instruct the RMC to apply the newly-downloaded image. To do so, write **Apply Download Image (2)** to the **Image Area Command (%MDx.0)** register.

5. **Wait for the Apply to Complete**
   Applying the image takes between 10-20 seconds (to shorten this time, see **Do Not Restart Controller Option** below). The controller will verify the integrity of the downloaded image, save a copy of the image in the controller's flash memory, and then automatically restart to apply the new image.

   The PLC should monitor the progress of the apply process using the **Image Area State (%MDx.1)** register. If the image is successfully verified and saved to flash, then the state will transition from **Applying Image (4)** to **Controller Restart Pending (5)**. The controller holds this state for two (2) seconds before restarting. When the PLC sees that the controller is in the **Controller Restart Pending (5)** state, it should stop communicating with the RMC for approximately 10 seconds to allow the controller to restart without triggering unnecessary communication errors. After restarting, the **Image Area State (%MDx.1)** register will read **Idle (0)**.

Do Not Restart Controller Option

This option is not supported by the RMC200.

The **Do Not Restart Controller (+16)** option can be used if the downloaded image does not change the axis definitions or loop time. This option will speed up the process, as the RMC does not need to restart. The image will be saved to Flash and all the settings will be available for use immediately. Downloading using this option typically takes less than 5 seconds.

If the **Do Not Restart Controller (+16)** option is used, then you should instead wait for the **Image Area State (%MDx.1)** register to become **Image Applied (10)**. There is no need to stop communicating with the RMC when this option is used. The RMC must be in PROGRAM mode when the image is applied.

Save/Restore Image Using the RMC200 SD Card

The RMC200 controller image can be saved to the SD card and restored from the SD card. For details, see the Using the SD Card topic.

Using RMCLink to Store and Download Images

The Controller Image Upload/Download feature can also be accessed from a Windows-based PC using the RMCLink component. This component contains methods that make it easy to upload and download configurations in custom applications using a variety of programming languages. RMCLink can be downloaded from Delta's website. Review the RMCLink documentation for details, particularly the topics relating to the **ReadImageToFile** and **WriteImageFromFile** methods.

Image Area

The Image Area is located in file 30 on the RMC75, and in file 94 on the RMC150. This file has the following structure.

RMC75 Address	RMC150 Address	RMC200 Address	Type	Access	Description
%MD30.0	%MD94.0	%MD23.0	DINT	Write Only	**Image Area Command** One of the following: **0=No Command** **1=Build Upload Image** **2=Apply Download Image** **3=Reset Image Area** Options for Apply Download Image (2):

					+16=Do not restart controller
%MD30.1	%MD94.1	%MD23.1	DINT	Read Only	**Image Area State** One of the following: **0=Idle** **1=Building Upload Image** **2=Upload Image Built** **3=Download in Progress** **4=Applying Image** **5=Controller Restart Pending** **10=Image Applied** **20=Image Build Error** **21=Upload Sequence Error** **22=Download Sequence Error** **23=Invalid Download Image** **24=Cannot Apply without Restart** **25=Failure writing to Flash** **26=Must be in PROGRAM mode to apply without restart** **27=Image supports a different hardware configuration** **28=Controller unable to build upload image because copy protection is enabled**
%MD30.2	%MD94.2	%MD23.2	DINT	Read Only	**Image Size** Provides the size of the Upload Image, in 32-bit registers.
%MD30.3	%MD94.3	%MD23.3	DINT	Read/Write	**Current Index** Provides the current index, in registers, in

					a multi-part image upload or download.
%MD30.4 : %MD30.4095	%MD94.4 : %MD94.4095	%MD23.4 : %MD23.4095	DINT[4092]	Read/Write	**Image Data** Used to upload or download the next block of the controller image.

See Also

RMC75 Register Map - File 30 Image Area | RMC150 Register Map - File 94 Image Area | RMC200 Register Map - File 23 Image Area | Save Controller Image (120) | Restore Controller Image (121)

---

4. Using RMCTools

4.1. Using RMCTools

RMCTools Overview

RMCTools is a Windows-based software package for the RMC75, RMC150 and RMC200 motion controllers.

RMCTools allows the user to set up, display, troubleshoot, program and control all features of the RMC motion controller. Fully detailed plots of motion can be captured at any time. RMCTools offers high speed communications to the RMC via a serial USB, Ethernet, or RS-232, allowing the user to tune even the most time-critical applications.

RMCTools is an Integrated Development Environment. All the editors and tools are part of the environment and the data is saved as part of the project. Some items, such as plots and Event Log captures, must be saved separately.

Operating Systems

This version of RMCTools requires Windows 11, 10, 8.1, or 7. For Windows XP (with SP3) or Vista, use RMCTools version 4.12.1 (September 2019) or older. For Windows 2000, use RMCTools version 3.37.1 (June 2010) or older. Both of these older versions of RMCTools are available for download on Delta's website.

RMCTools has been run on Linux under Wine, but is not officially supported.

Getting Started

Using the RMCTools Interface
Project and Controller Data
Startup Procedure

RMCTools Components

Project
Creating a New Project
Project Pane

Controller
Adding a New Controller
Connection Path
Go Online
Run/Program Mode
Communication Statistics
Communication Log

Axes
Axis Tools
- Axis Status Registers Pane
- Axis Parameters Pane

Axis Definitions

Tuning
Tuning Tools
Tuning Wizard
Autotuning
Gain Calculator

Programming
Programming Overview
Variable Table Editor
Discrete I/O Configuration
Discrete I/O Monitor
Program Triggers
Step Editor (User Programs)
Task Monitor

Address Maps
Address Maps

General Tools
Menu and Toolbars
Upload and Download
Error Bubble
Address Selection Tool
Output Window
Verify Results Window
Actuator View
RMCTools Options
Communication Log
Keyboard Shortcuts
Copy and Paste
Printing

Wizards
New Project Wizard

Command Tool
Command Tool

Plots
Plot Manager
Plot Template Editor
XY Plots

Indirect Data Map Editor

Shortcut Commands
Shortcut Commands

Event Log
Event Log Monitor

New Controller Wizard
Scale/Offset Wizards
Simulator Wizard
Firmware Update

See Also

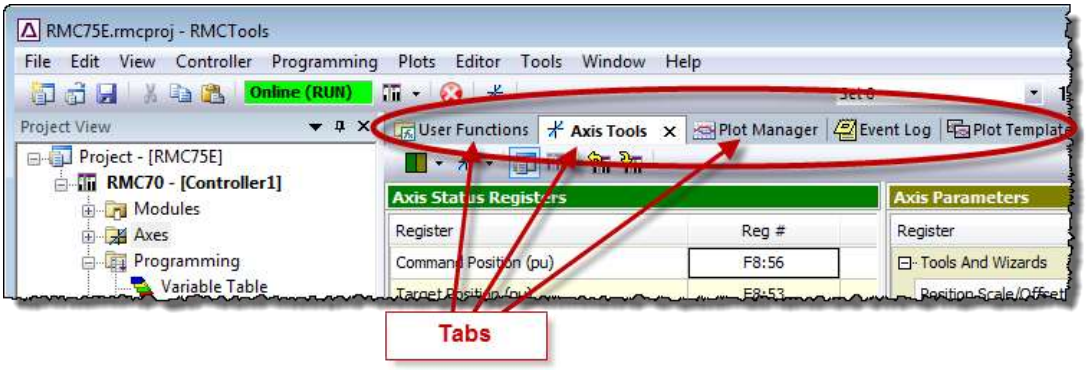
Help Overview

4.2. Using the RMCTools Interface

RMCTools is an Integrated Development Environment, which means that the project is contained within one window. Many dockable and non-dockable windows may be open in RMCTools.

Tabbed Windows

The main windows, such as Axis Tools, Plot Manager, etc., open in a tabbed format:



Closing a Tabbed Window

To close a tabbed window, right-click the tab and choose **Close**.

Splitting Tabs

To split a window into a new tab group, right-click the tab and choose **Split Side-by-Side** or **Split Above-Below**.

Moving Tabs Between Tab Groups

To move windows between tab groups, right-click the tab and choose **Move to Previous Tab Group** or **Move to Next Tab Group**.

Moving Tabs Between Tab Groups

To move windows between tab groups, right-click the tab and choose **Move to Previous Tab Group** or **Move to Next Tab Group**.

Dockable Panes

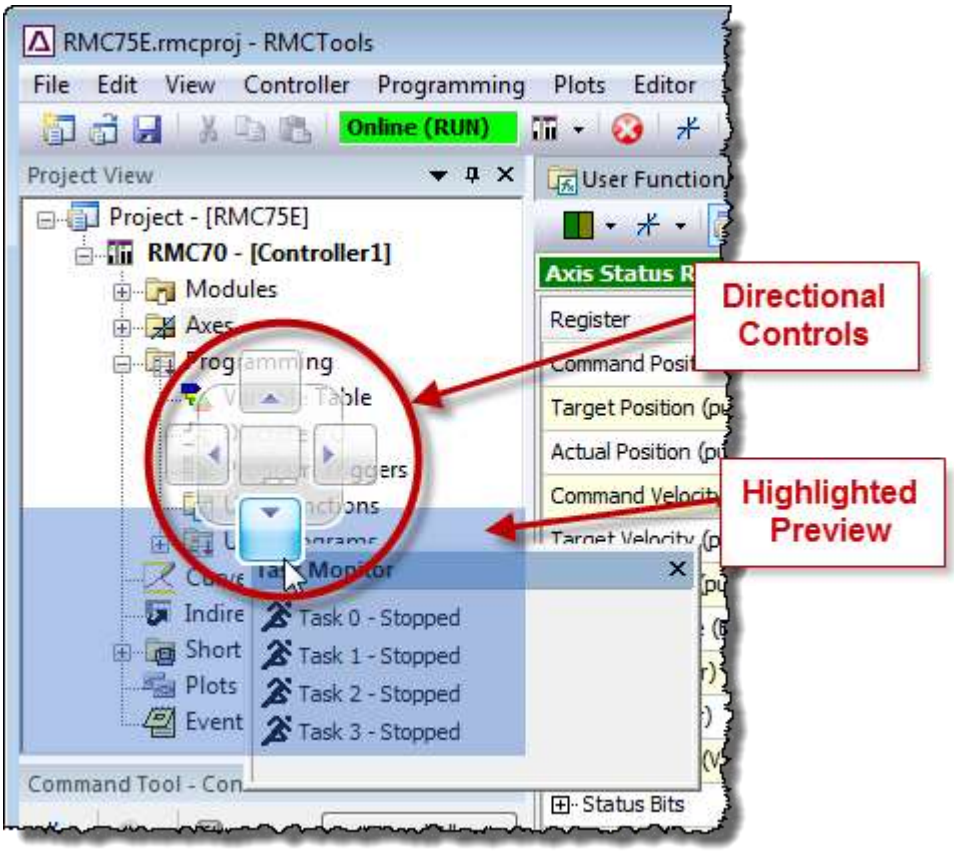
The following windows, or *panes*, are normally *docked* to the sides of the RMCTools window:

- Project Pane
- Command Tool
- Discrete I/O Monitor
- Task Monitor
- Output Window
- Verify Results Window

Moving Dockable Panes

To move a dockable pane, click and drag the title bar of the pane. As you drag, directional controls will appear. Dragging to a directional control will give a highlighted preview of the new pane location. Dropping on the directional control will place the window at that location.

Example of docking the Task Monitor below the Project pane:



Placing Dockable Panes in Tab Groups

To dock panes together into tab groups, move the pane to the center square in the directional controls.

Auto-Hiding Dockable Panes

To auto-hide a pane, in the title bar, click the **Auto Hide** button. When pane will collapse to the edge of the RMCTools window.

To access a collapsed pane, click the collapsed location. The pane will expand. When you click outside of the expanded pane, the pane will collapse again.

Floating Dockable Panes

To float a dockable pane, in the title bar, click the arrow and choose **Float**.

Resetting the Window Layout

If the windows become jumbled, on the **Window** menu, choose **Reset to Basic Layout**.

If you are using a small monitor, the **Reset to Small-screen Layout** will auto-hide the Project pane and Command Tool, freeing up space for other windows, such as the Plot Manager.

See Also

RMCTools Overview | Help Overview

4.3. Project and Controller Data

Project data is the data in the RMCTools project file that you can saved to your computer.

Controller data is the data in the RMC. You can save the controller values in the RMC by Updating Flash.

Values not Equal 

The  symbol indicates that the data in the project is different from the data in the controller. The data that is not equal will be highlighted. To apply the controller data to the project, upload the parameters. To apply the project data to the controller, download the parameters.

In the Project Pane, if an item displays a difference, you can right-click the item and choose the download to the controller or upload form the controller.

Data in Editors

In the Axes Parameters Pane, Indirect Data Map Editor, and Variable Table Editor, you can view the project controller values or the controller values.

When editing data in these windows, you need to click the **Download** button  to apply the values from the project to the RMC.

View the project values

To view the project values, click the **Show Project Values** button  . If there are no differences between the project values and the controller values, the editor will always be in this mode. You can edit the values in this mode.

View the controller values

If RMCTools has detected a difference between the data in the controller and the data in the project (indicated by the  symbol), you can click the **Show Controller Values** button  to show the values in the RMC. If there are no differences, this button is inactive. You cannot edit the controller values in this mode. To change the values in the controller, you must first click the **Show Project Values**  button.

Values not Equal 

The  symbol indicates that the project data and the data in the controller are not equal. The data that is not equal will be highlighted. To apply the controller data to the project, click the **Upload** button  to apply the values from the RMC to the project. To apply the project data to the controller, click the **Download** button  .

See Also
RMCTools Overview

4.4. Project

4.4.1. RMCTools Project

The RMCTools project contains all the information involved in setting up and programming the RMC. To save the project, on the **File** menu, click **Save**. The project will saved to a file with the ".rmcproj" extension.

The following items are not saved in the project file and can be saved separately:

- Plot Captures
- Event Log

Some items stored in the project can also be exported to files, such as:

- User Programs
- User Functions
- Curves
- Shortcut Command Sets

Starting a New Project

1. On then **File** menu, click **New**, then click **Project**.
2. Review the following data and modify as desired:
    a. **Project Name**
    b. **Project Path**
    c. **Author**
    d. **Checkbox: Create a Subfolder for this project**
    Checking this box will create a subfolder in the project path and save the project in that folder. The subfolder will have the same name as the project.
    e. **Checkbox: Start the New Controller Wizard after this wizard is finished**
    Checking this box will cause the **New Controller Wizard** to open after clicking OK in this dialog. Typically, you will want this box checked so you can immediately add a new controller after creating a project.
3. Click **Finish**. The Project Pane will show the project. If the checkbox was checked, the New Controller wizard will open to add a new controller to the project.

Project Properties

To view the project properties, in the Project Pane, right-click **Project** and click **Properties**.

The following properties can be set:

General Page

- **Project Name**
  The name of the project.
- **Project File**
  The location to save this project to. To save project to a different location, on the **File** menu, choose **Save As**. If you change the Project File and then save the project, RMCTools will attempt to save the project as the path and file you have specified.

- **Author**
  Your name.

Description Page

Enter a description for the project. This is for your own reference.

See Also

Project Pane

4.4.2. RMCTools Project Pane

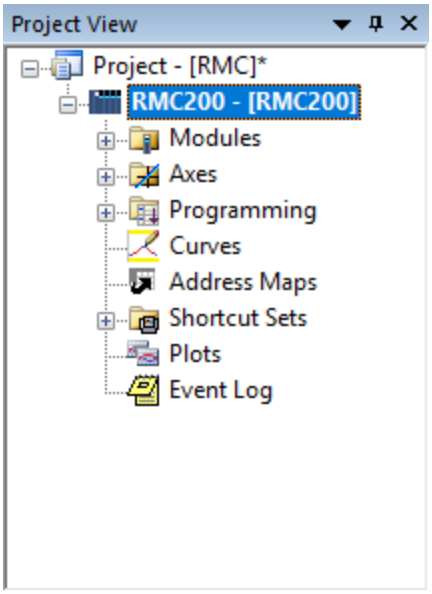
To access this pane:
The Project pane should already be open in RMCTools. If it is not, on the View menu, click **Project**, or press Alt+0.

The Project pane provides an hierarchical overview of the components of the RMCTools Project. All components can be accessed from the Project pane by double-clicking or right-clicking the item. For ease of use, leave this pane open when using RMCTools.

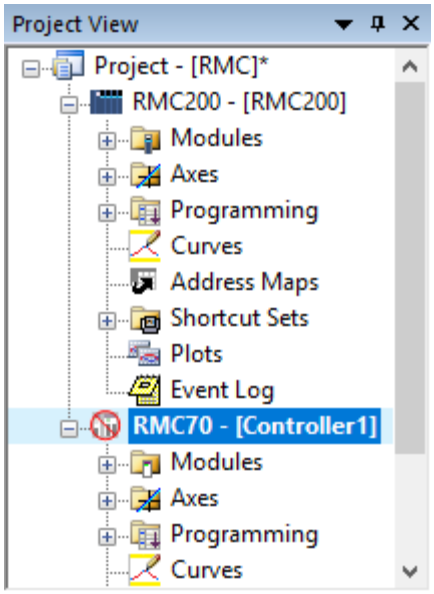
When saving the project, the information available from the Project pane is saved in the Project File. To create a new project, see the New Project wizard topic.

The Project pane shows all of the controllers in the project and their components. A project can have any number of controllers. Use the New Controller wizard to add a controller to a project. If the project has more than one controller, the active controller will be highlighted, as shown below:

Single Controller



Two Controllers



Differences Between Controller and Project

When RMCTools is online with the motion controller, any areas in the project that are different from the controller will be marked with a ⚠ icon and yellow highlighting. To resolve the difference, either upload from the controller or download to the controller.

Components

Each RMC controller has the following components, visible in the Project Pane:

Modules

Shows the hardware modules. Double-click a module to view the module properties. To view or change modules, expand the Modules folder and double-click **View/Change Modules**.

Axes

Contains axis-related tools, such as the Axis Definitions and Axis Tools. An error icon indicates some of the axis definitions are invalid.

Programming

Contains items for programming the controller:

Double-click **Variable Table** to open the Variable Table Editor.

Double-click **Discrete I/O** to open the Discrete I/O Configuration. To display the Discrete I/O Monitor, right-click **Discrete I/O** and choose **Discrete I/O Monitor**.

Double-click **Program Triggers** to open the Program Triggers Editor.

Double-click **User Functions** to open the User Function Editor.

Multiple User Programs may be created for each controller. Right-click **User Programs** and choose **New Program** to create a new program. You can sort the user programs by name or by number. To do so, right-click **User Programs** and choose **Sort by Name** or **Sort by Number**.

The Task Monitor is for monitoring user programs. On the **View** menu, click **Task Monitor**.

Curves

Double-click to open the Curve Tool.

Address Maps

Double-click Address Maps to access the maps, such as the Indirect Data Map.

Shortcut Sets

Expand the Shortcut Sets folder and double-click a set to edit it. To add a new Shortcut Set, right-click the folder and choose **New Shortcut Set**. See the Shortcut Commands topic for details.

Plots

Double-click to open the Plot Manager for uploading plots and editing plot templates.

Event Log

Double-click to open the Event Log. This log is useful for troubleshooting. It logs issued commands, changed parameters, and errors.

See Also

RMCTools Project

4.5. Controller

4.5.1. New Controller Wizard: Welcome

Next Wizard Page (Auto Detect)

Next Wizard Page (Manually Configure)

To access this wizard:
In the Project pane, right-click **Project** and click **New Controller**.

Or:
On the file menu, point to **New** and choose **Controller**.

Use this wizard to add a motion controller to the project.

Welcome Page

1. Enter a name for the new controller.
2. Choose one of the following:
   - **Automatically Detect the Controller Information**
     Choose this option to let RMCTools connect to the controller to get the information.
   - **Manually Configure the Controller Information**
     Choose this option to manually enter the controller information.
3. Click **Next** to proceed to the next page.

If you chose **Automatically Detect the Controller Information**, the next page is Connection Path.
If you chose **Manually Configure the Controller Information**, the next page is Select a Controller Family.

Show All

4.5.2. Connection Path

To access this dialog:
Right-click the desired controller in the **Project** pane and choose **Connection Path**.
Or:
On the **Controller** menu, click **Connection Path**.

The Connection Path specifies how RMCTools connects to the RMC.

Connection Methods

The following connection methods are possible for the RMC:

Controller	Connection Methods
RMC75E	USB, Ethernet
RMC75S	Serial Port
RMC75P	Serial Port
RMC150E	USB, Ethernet
RMC200	USB, Ethernet

To Select a Connection Path

1. In the Project Pane, right-click the controller you wish to connect to and click **Connection Path**.
2. Choose the connection type:
   - **Ethernet**
     a. Click **Browse** to choose an RMC.
     b. Choose your RMC from the list, or enter an IP address in the **IP Address** box.

c. Click **OK.**
If your RMC appears in the list, but has an IP Address of 0.0.0.0, click **Configure Device** to configure it's IP settings.
If your RMC does not appear in the list, see the Troubleshooting Ethernet topic for help.

- **USB**
  a. Click **Browse** to choose an RMC.
  b. Choose your RMC from the list.
  c. Click **OK.**
  If your RMC does not appear in the list, verify that the USB cable is connected.
- **Serial**
  a. Choose the COM port.
  b. Click **OK**.

1.

3. Click **OK** to apply the changes, or click **Go Online** to apply the changes and go online.

See Also

Monitor Port | Go Online

4.5.3. Go Online, Go Offline

The **Go Online** command in RMCTools connects RMCTools to an RMC. The **Go Offline** command stops communication between RMCTools and the RMC.

Going Online or Offline

To go online or offline, select the desired controller in the Project pane, then, on the toolbar, click the **Controller** button  and choose **Go Online** or **Go Offline**. The toolbar controller indicator shows whether RMCTools is online or offline:

Offline	Offline.
Online (PROG)	Online and the controller is in PROGRAM Mode.
Online (RUN)	Online the controller is in RUN Mode.
Online (Disabled)	Online the controller is in the Disabled state (RMC200 Only).

Connecting to a Controller

Use these steps to connect from RMCTools to the RMC using the monitor port:

1. **RMC75E, RMC150E, or RMC200:**
   Connect a USB cable from the PC to the USB Monitor port on the RMC.

   RMC75S or RMC75P:
   Connect a null-modem cable from the PC to the RS-232 Monitor port on the RMC. See the **RS-232 Monitor Port Wiring** section above for cable details.
2. In RMCTools, in the Project Pane, right-click the desired controller you wish to connect to and choose **Connection Path**.
3. **RMC75E, RMC150E, or RMC200:**
   Choose USB and click **Browse.** Choose the desired RMC and click **OK.**

RMC75S or RMC75P:
Choose the COM port that the RMC is connected to.

4. Click **Go Online**.

See Also

Monitor Port | Connection Path

4.5.4. Using RUN/PROGRAM/Disabled Mode in RMCTools

Tip:
For more details on RUN, PROGRAM, and Disabled Mode, see the Run/Program/Disabled Mode topic.

To change the RUN/PROGRAM/Disabled mode in RMCTools, use the Run/Program/Disabled icons in the Standard Toolbar:

- Select the desired controller in the Project pane, then, on the toolbar, click the **Controller** button  and choose the desired mode. The toolbar controller indicator shows the current mode of the controller:

Online (PROG)	Online and the controller is in PROGRAM Mode.
Online (RUN)	Online the controller is in RUN Mode.
Online (Disabled)	Online the controller is in the Disabled state (RMC200 Only)

See Also

Standard Toolbar | RUN/PROGRAM Mode

4.5.5. Communication Statistics Window

To access the Communication Statistics Window:
In the **Project** pane, select the desired controller. On the **Controller** menu, click **View Communication Statistics**.

The Communication Statistics window displays a number of diagnostic counters that are useful for monitoring and troubleshooting the communication via the standard port and the Monitor Port. For more details, see the Communication Statistics topic.

Some of the communication statistics can be cleared (rest to zero). To clear a statistic, select a folder in the navigation tree, then click **Clear Statistics** on the toolbar.

The Event Log is also very useful for troubleshooting the communications.

See Also

Communications Overview | Event Log | Communication Statistics

4.5.6. Updating Flash

Updating Flash is the process by which the data in the RMC is stored to non-volatile Flash memory. In order to save data even when power is removed, the Flash must be updated. After changing the configuration of the RMC, such as parameters, axis definitions, variable initial values, curves (temporary curves cannot be saved to Flash), and more, you must update Flash in order to save the settings through a power cycle.

Updating Flash from RMCTools

To update flash from RMCTools:

- Go online with the controller.
- On the **Controller** menu, click **Update Flash**.
  or,
  on the **File** menu, click **Save and Update Flash**, which will update Flash in the controller and save the RMCTools project.

The Flash updates typically take 1 or 2 seconds, but may take a few seconds longer on larger projects. The Event Log will log the start and finish time of the Flash update, and the amount of Flash memory used.

The RMC will continue to control motion as usual during a Flash update. On the RMC75/150 (not the RMC200), the green CPU LED will flash while a Flash update is in progress.

You may also be prompted to save to Flash when the controller is warm restarted after changing certain controller data.

Updating Flash via a Command

You can use the Update Flash (110) command to update Flash. While a Flash update is in progress, the green CPU LED will flash.

Flash Size

The entire RMC project is saved to Flash, including axis parameters, plot templates, variable table defaults, user programs, and curves. After updating Flash, the Event Log will log the amount of Flash memory used. The total Flash size is as follows:

Controller	Flash Size
RMC75E (1.1G and newer)	1024 KB
RMC75E (1.1F and older)	256 KB
RMC75P	96 KB
RMC75S	96 KB
RMC150E	1024 KB
RMC200	6016 KB

Updating Flash versus Retentive Variables

Variable values can be remembered between power cycles by marking them as retentive (RMC75E, RMC150E, and RMC200 only), or by updating Flash. The table below compares each method. Usually, marking variables as retentive is a better choice than using Flash.

	Updating Flash	Retentive Variables (RMC75E, RMC150E, and RMC200 only)
Value Stored	Initial Value	Current Value

Update Frequency	When the Update Flash is commanded by user or program	Automatically every 100msec
Update Time	RMC75S/P: 1 second RMC75/RMC150: 2 seconds RMC200: Multiple seconds	< 10 milliseconds
Write Cycle Limits	100,000 or 1 million, see below	None

Flash Write Cycle Limits

The Flash memory chip on the RMC75S, RMC75P, and RMC200 is limited 100,000 update cycles. The RMC75E and RMC150E are limited to 1 million update cycles. Due to these limits, you should take care to program the RMC so that this limit will not be exceeded during the life of the controller.

See Also

Update Flash (110) Command

4.5.7. Resetting the RMC to Defaults

Resetting the RMC to its default settings will erase the settings in Flash memory and replace them with the default settings. If you want to save the settings in the RMC before resetting it to default settings, do so by uploading all and then saving the project in RMCTools.

To reset the RMC to its default settings:

- On the **Controller** menu, click **Reset Controller to Defaults**.
- Click **Yes**. The RMC Flash memory will be erased and replaced with the default settings.

4.6. Modules

4.6.1. Modules List

The **Modules** folder in the Project Pane lists all the RMC's hardware modules.

View/Change Modules

To view or change a module in the controller, double-click **View/Change Module**.

Modules can only be changed in the project when RMCTools is offline with the controller.

View/Change Module Properties

To view or change the properties of each module listed, double-click the module.

View Module Specifications

For hardware specifications for each module, see the topic for the individual module, accessible from the RMC75, RMC150, and RMC200 overview topics.

See Also

Help Overview

4.6.2. RMC Hardware Configuration Dialog

To access this dialog:
In the Project pane, expand **Modules** and double-click **View/Change Modules**.

Use this dialog to view and change the RMC hardware modules. The image in the dialog displays the complete RMC along with the part number. The modules in the project can only be changed when RMCTools is offline with the RMC.

RMC75

For each section, choose **Change, Add** or **Delete**  to change, add or remove a module. Use the **Move Up** ↑ and **Move Down** ↓ buttons to change the module order for moveable modules.

Making changes to the RMC75 hardware configuration will cause the Axis Definitions to be reset to the default settings.

RMC150

Click on the image to choose a slot. For each slot, choose the desired module from the list. Click **Delete** to remove a module from a slot.

Making changes to the RMC75 hardware configuration will cause the Axis Definitions to be reset to the default settings.

RMC200

In the **Base** box, chose the desired base. For each row in the **Modules** column, choose the desired module.

Click **Delete**  to remove a module from a slot. Use the **Move Up** ↑ and **Move Down** ↓ buttons to change the module order for moveable modules.

Making changes to the RMC200 hardware configuration will not reset the axis definitions, but may result in invalid Axis Definitions. Make sure to review the Axis Definitions after making any module changes.

See Also

Axis Definitions

4.7. Axes

4.7.1. Axis Tools

4.7.1.1. Axis Tools Window

To access this window:
On the RMCTools toolbar, click the **Axis Tools** button.

The Axis Tools window displays several types of information for the RMC. It consists of two panes:

- **Axis Status Registers**
  The Status Registers provide information on the status of each axis in the RMC. For information on each register, click on the register cell in any **Axis** column, and press **F1**. Status registers are read-only.
  For details on the Axis Status Registers pane, see the Axes Status Registers Pane topic. For a list of all the Status Registers, see the Register Map topic.
- **Axis Parameters**
  The Axis Parameters provide configuration information for the RMC. For information on each parameter, click on the parameter cell in any **Axis** column, and press **F1**. Parameter registers can be read or written.
  For details on the Axis Parameters pane, see the Axes Parameters Pane topic. For a list of all the Axis Parameter registers, see the Register Map topic.

Using the Axis Tools Window

Choosing Views

Use the **Select Layout** box to choose the pane layout in the Axis Tools. The various layouts may help you place the Axis Tools pane such that other panes in RMCTools are also visible.

F6 switches between the Axis Status Registers and the Axis Status Registers.

Hiding and Showing Columns

To choose which axes to display, use the **Select Axes** button, or right-click any column header.

Showing Project or Controller Values

To view the values in the RMCTools project, click the **Show Project Values** button. To view the values in the RMC, click the **Show Controller Values** button. If the values in the RMC and in the RMCTools project are identical, the **Show Controller Values** button will be disabled.

Uploading and Downloading Values

After making changes to any editable parameters in the Axis Tools, you must download the changes to RMC. To download the project values from the Axis Tools to the RMC, click the Download button. To upload the controller values from the RMC to the project, click the Upload button.

Choosing the Address Format

The addresses of the registers in the Axis Tools are displayed in the **Reg #** column. To change the address format, right-click any cell in the **Reg #** column, choose **Address Formats**, and choose the desired format.

See Also

Axis Status Registers Pane | Axis Parameters Pane

4.7.1.2. Axis Status Registers Pane

To access this pane:
On the RMCTools toolbar, click the **Axis Tools** button. If the **Axes Status Registers** pane does not appear, use the **Select Layout** box to choose the **Axes Status Registers** pane, or press F6.

The **Axis Status Registers** pane is part of the Axis Tools pane. It displays the Axis Status registers. This data is very useful in all phases of using the controller.

Tabs

The Axis Parameters pane has 2 tabs. Within each tab, the registers are arranged in sections. To expand a section, clicking the '+'.

- **Basic Tab**
  Use this tab to view the basic status registers for an axis.
- **All Tab**
  Use this tab to view all the status registers. This tab is useful for advanced troubleshooting.

Using the Axis Status Registers Pane

Columns

Column Name	Description
Registers	This is the descriptive name of the parameter register, along with the units.
Reg #	The Reg# column contains the address number for each register. The addresses apply to the selected axis. To change the address format, right-click any cell in the Reg # column, choose **Address Formats**, and choose the desired format.
Axis	This is the current value for this axis of the register in the **Reg #** column.

Hiding and Showing Columns

To hide a column, right-click the column heading and click **Hide Column**. To see it again, right-click any column heading, click **Add Column**, and click the desired column name.

See Also

Status Registers

4.7.1.3. Axis Parameters Pane

To access this pane:
On the RMCTools toolbar, click the **Axis Tools** button. If the **Axes Parameters** pane does not appear, use the **Select Layout** box to choose the **Axes Parameters** pane, or press F6.

To access from the Plot Manager:
Click the **Tuning** tab (next to the **History** tab).

The **Axis Parameters** pane is part of the Axis Tools. It is also part of the Tuning Tools. Use the **Axis Parameters** pane for the following:

- Monitor and edit the Axis Parameters.
- Set up and tune the RMC.

For general information on using editors, see the Using Editors topic.

Tabs

The Axis Parameters pane has 3 tabs. Within each tab, the registers are arranged in sections. To expand a section, clicking the '+'.

- **Setup Tab**
  Use this tab for setting up the controller. This tab contains the basic parameters required for setup.
- **Tune Tab**
  Use this tab for tuning the controller. This tab contains the basic parameters required for tuning.
- **All Tab**
  Use this tab to edit advanced parameters. This tab contains all the Axis parameters.

Using the Axes Parameter Pane

Editing Parameter Values

- Find the parameter you wish to edit.
- Click the cell in the desired **Axis** column.
- Type or choose a value and press Enter.
- Click the **Download** button  to apply the changes to the RMC.
- To save the values to Flash memory, on the **Controller** menu, click **Update Flash**.

Columns

Column Name	Description
Registers	This is the descriptive name of the parameter register, along with the units.
Reg #	The Reg# column contains the address number for each register. The addresses apply to the selected axis. To change the address format, right-click any cell in the Reg # column, choose **Address Formats**, and choose the desired format.
Axis	This is the current value for this axis of the register in the **Reg #** column. You can edit these values.

Hiding and Showing Columns

To hide a column, right-click the column heading and click **Hide Column**. To see it again, right-click any column heading, click **Add Column**, and click the desired column name.

See Also

Axis Parameters

4.7.2. Axis Definitions Dialog

To access this dialog:
In the **Project** pane, expand the **Axes** folder and double-click **Axis Definitions**.

Or:
From the Axis Definitions page of the New Controller Wizard, click **View/Change Axis Definitions**.

Use this dialog to view or change existing axis definitions, or to add or remove axes.

View an Axis Definition

To view an axis definition, click the axis in the Axis Definition table. The hardware assigned to the selected axis is highlighted in the RMC image, and is color-coded for output, primary input, and secondary input.

To view detailed information on the selected axis definition, click **Change**.

The **Axis** column provides the following information:

	A control axis.
	A reference axes — commonly called a "half-axis".
	An invalid axis definition.
#	Indicates the number of the axis.
[]	The user-defined name of the axis.

Invalid axis definitions will be marked with a  icon and red text. Invalid axis definitions can be downloaded to the RMC, but the axis may not be functional.

Change an Axis Definition

To change an axis definition, click the axis in the Axis Definition box, then click **Change**. When making major changes to the axis definition, it may be easier to first delete the axes, and then create new ones.

See the Axis Definitions: Change topic for details on changing an axis definition.

Rename an Axis

To rename an axis, select the axis and click **Rename**. Type a name and press Enter.

Add or Remove an Axis

To add an axis, click the axis in the Axis Definition box, then click **New**. See the New Axis Wizard topic for details on adding a new axis. To remove an axis, click the axis in the Axis Definition box, then click **Remove**  .

The axis limits for the controller are listed at the top of the dialog.

Note: It is possible to add more analog inputs on the RMC75 than can be assigned to axes. However, it is still possible to view the voltage of the extra analog inputs using the Analog Input Registers. Inputs that are unassigned to axes can have no status bits, error bits, scaling, filtering, etc.

Change the Order of Axes

Note: Delta recommends defining the order of the axes at the beginning of the project. Once the project is underway, changing the axis order can break any programming you have already done.

To change the axis order, use the **Move Up** ↑ and **Move Down** ↓ buttons.

The axes will adhere to this order throughout the entire RMC. The Axis Tools and Command Tool will display the axes in this order, and the register addresses and tag names will be defined by this order.

Axis and Control Loop Limits (RMC200 Only)

As axes are added, the dialog lists the number of physical control axes, Feature Key control loops, total axes, and the maximum allowed values. For details on the limits, see the RMC200 Overview.

The Control Loop Utilization Advisory is an approximate graphical indication of how much of the loop time will be used for processing the axes. As described in the Loop Time topic, it is important that the actual loop processing time does not frequently or excessively exceed the selected loop time.

Buttons

Button	Description
New	Add a new axis.
Change	Change or view the definition of the selected axis.
Remove 	Remove the selected axis.
Move Up ↑	Move the selected axis up one position in the list.
Move Down ↓	Move the selected axis down one position in the list.
Rename	Rename the selected axis.
OK	Close the dialog and apply the changes to the project. If you are online with the controller, the changes will also be applied to the controller. If you accessed this dialog from the New Controller Wizard, the changes will be applied to the wizard.
Cancel	Close the dialog without applying any changes.

See Also

Axis Types Overview

4.8. Command Tool

4.8.1. Command Tool

To access this tool:
If the Command Tool is not already open, on the View menu, click **Command Tool**, or press Alt+8. The Command Tool is dockable and can be resized and moved as desired.

Use the Command Tool to send commands to the RMC.

Tip:
To send commands to the RMC quickly, or to ease the process of issuing the same command multiple times, use Shortcut Commands.

Sending Commands

To send a command to one axis:

a. Determine which axis you wish to send a command to.
b. In the **Cmd** box for the desired axis, select a command. There are several ways to do this:
    - In the **Cmd** box, begin typing the *name* of the command and then select it from the list that opens upon typing.

- In the **Cmd** box, begin typing the *number* of the command and then select it from the list that opens upon typing.
- In the **Cmd** box, click the ellipsis button and choose the command from the command tree.

c. If the command has parameters, you must enter values in the parameters.
d. Click **Send Command** to send the command to the axis.

Tip:
Use the Event Log to see if the command was actually sent to the RMC. The Event Log will also display any errors that may have occurred.

To re-send a previous command:

a. In the **Cmd** box for the desired axis, click the **Command History** button .
b. In the drop-down list box, click the command you want.
c. Click **Send Command** to send the command to the axis.

Note:
The command history stores the last 10 commands.

To send commands simultaneously to multiple axes:

a. In the Command Tool, right click anywhere and choose **Show Send All options**. The **Send All** button will appear.
b. For each axis that you will send a command to, select the command and enter values in the parameters if there are any.
c. For the axes that should not be receiving any command, choose the No-op (0) command.
d. Click the **Send All** button. This will send all the commands from each axis in the Command Tool.

Tip:
Use the Event Log to see if the commands were actually sent to the RMC. The Event Log will also display any errors that may have occurred.

Selecting Axis to Display

In projects with many axes, you may wish to display only a few of the axis at one time in the Command Tool. The axes displayed in the Command Tools are determined by the axes displayed in the Axis Tools window. To specify which axes to display, use the Axis Selection button in the Axis Tools window.

Shortcut Commands

Shortcut commands enable you to send commands quickly with a keypress or the click of a button. See the Shortcut Commands topic for details.

See Also

Issuing Commands from a PLC or HMI

4.9. Plots

4.9.1. Plot Manager Overview

To access the Plot Manager:
On the **Plots** menu, click **Open Plot Manager**.

Use the Plot Manager to:

- **View Plots**
  From the Plot Manager, you can start, trigger, save and export plots.
- **Tune the Axes**
  Use the Tuning Tools to efficiently and easily issue commands, change and download gains, and view plots of moves.
- **Edit Plot Templates**
  Access the Plot Template Editor from the **Plotting** tab.

For general details on using plots in the RMC, see the Plots Overview topic. For help on the various parts of the Plot Manager, see the Plot Manager Elements topic.

The following instructions apply to plots in general and to time-based plots. For instructions specific to XY plots, see Using XY Plots.

How To...

- **Upload a Captured Plot**
  On the **Plotting** tab, click **Capture** for the desired plot template to upload the latest captured plot.

  The RMC starts capturing plot data when a motion command is sent, the Start Plot (100) command is sent, or the plot is triggered. The RMC captures this data for as long as specified by the plot capture duration in the Plot Template Editor, and stores this data internally. To view the captured plot data, you need to upload the captured plot. By default, each plot template contains data for the corresponding axis number, although you can edit each template to include any data in the RMC.

  If you attempt to upload a captured plot that has not been triggered, or has been re-armed, the Plot Manager will indicate it is waiting for a plot trigger before it can upload plot data.

  It is usually not necessary to stop the upload of the captured plot, as it will stop at the defined capture duration. If you do wish to stop the capture upload, click **Capture**, or press Esc.

- **Trend a Plot**
  On the **Plotting** tab, click **Trend**. To stop the trend, click **Trend**, or press Esc.

  Trending continuously uploads the plot data while it is being captured in the RMC. Set the trend duration in the Plot Template Editor. Trend data beyond that time will be lost. Trending will upload data as fast the communications allows, and some plot samples may be missed. If you need every plot sample without any gaps, use Capture mode.

  While trending, the cursor will be at the end of the plot (with the newest data) and the **Plot Details** window will show the values of the newest data. If you click in the plot while trending, the cursor will be placed at that location. To snap the cursor to the newest data again, click or press End, or move the horizontal scroll bar all the way to the right.

- **Zoom and Scroll**

	Mouse	Keyboard	Toolbar
Horizontal Zoom	Ctrl+Mouse Wheel	Ctrl+Plus Sign Ctrl+Minus Sign	1:1
Horizontal Scroll	Shift+Mouse Wheel	-	-

Horizontal Reset[1]	-	-	
Vertical Zoom[2]	Ctr+Shift+Mouse Wheel	Ctrl+Shift+Plus Sign Ctrl+Shift+Minus Sign	
Vertical Scroll[2]	Mouse Wheel	-	-
Vertical Reset[1]	-	-	
Reset All[1]	-	-	

[1]Resets auto-scaled items to fit the data. For manually scaled items, resets the scale to the manual setting.

[2]Vertical zoom and scroll applies to the currently selected item and all items in the same scale group.

Vertical manual scaling:

The vertical scale of items in a plot can be manually set. The scale applies to the item currently selected in the Plot Details list, and all items in the same scale group, which are highlighted in light blue. In the **Plot Details** window, right-click the item and choose **Set Scale**. Or, in the plot window, right-click and choose **Set Scale**.

- **Move the Cursor in the Plot Window**
  Click and drag in the plot window to move the cursor. The **Plot Details** window in the lower left displays all the plot data at the location of the hairline cursor. Move the hairline cursor to view the data at any sample.
  To display the absolute time of the cursor, which is the same time as shown in the Event Log, on the **Plots** menu, point to **View**, then click **Show Absolute Time**.

- **Understand the Vertical Scale**
  Clicking an item in the Plot Details list will change the vertical scale in the plot to apply to that item, if the item has a plotted line. All items that the currently displayed vertical scale applies to will be highlighted in light blue. The vertical color bar next to the vertical scale will also display the colors of all the items that share the same vertical scale group.

- **View Tuning Parameters at the Time the Plot was Captured**
  Click the **Parameters** tab below the **Plot Details** tab to view the axis parameters at the time the plot was captured. These parameters are uploaded immediately after the plot upload is completed.

- **Trigger a Plot**
  On the **Plots** menu (in the menu bar at the top of RMCTools), click **Trigger Plot**. Before triggering a plot, it may need to be rearmed. To rearm a plot, on the **Online** menu, click **Rearm Plot**. You can also use the Trigger Plot (102) and Rearm Plot (103) commands.

  Triggering a plot will cause the RMC to start capturing plot data and store the data internally. To view it, you will also need to upload the captured plot.

  You can change the trigger settings. See Triggering Plots for details.

- **Trim a Plot**
  First, show the second cursor:
  - Press and hold Shift and click in the plot window.
  - Or, in the plot window, right-click and choose **Show 2nd Cursor**.
  Second:
  - Use the mouse to adjust the cursors for the desired trim area

- Right-click and choose **Create Trimmed Plot**.

A new trimmed plot will be added to the Plot History. The original plot will be unchanged. Trimming a plot is useful for picking out only the information needed. It can also be used for picking out one move to be used in the Tuning Wizard with an existing plot. Using the Tuning Wizard with existing plots requires plots of motion that include motion in only one direction.

- **Use an XY Plot**
  See Using XY Plots.

- **Save Plots**
  Plots you have uploaded are listed on the **History** tab. They are not saved in the RMCTools project. To save plots, you must save them as follows:

  a. On the Plot Manager toolbar, click **Save Plot(s)**.
     Or, on the Plot History pane, right-click a plot in the list and choose **Save Plot(s)**.
     Or, in the plot window, right-click the currently displayed plot and choose **Save Plot(s)**.
  b. In the **Select Plot to Save** dialog, choose the plots to include, then click **Save**.
  c. Browse to the desired folder.
  d. Click **Save**.

  See Saving and Exporting Plots for details.

If you forgot to save plots before closing RMCTools, you can retrieve backup copies of uploaded plots:

a. In the Plot Manager, on the **History** tab, expand **Auto-Saved Plots**, then click **View Auto-Saved Plots**.
b. Choose the plots you wish to retrieve. You can select multiple files by holding the Shift or Ctrl keys while clicking the files.
c. Click **Open**.

For more details, see Auto-SavedPlots.

- **Open a Saved Plot File**
  In the Plot Manager toolbar, click the **Open Plot File** button. In the **Open** dialog, browse to the desired plot file, select it, and click **Open**. In the **Open** dialog, you can use the Ctrl or Shift key to select multiple files.

  See Saving and Exporting Plots for details.

- **Export a Plot**
  On the history tab, right-click a plot and choose **Export Plot**. The plot can be exported use in other programs such as Microsoft Excel. See Saving and Exporting Plots for details.

- **Print a Plot**
  On the toolbar, click the **Print** button to print the currently viewed plot.
  If the number of vertical scales causes the available plot area to be less than 50% of the printable area, the remaining vertical scales will not be included in the plot. To prevent this from occurring, group scales together in the Plot Template settings.

Using the Tuning Tools

To open the tuning tools, click the **Tuning** tab in the Plot History pane. For details, see the Tuning Tools topic.

See Also

Plot Overview | Using XY Plots | Using Custom Plot Templates

4.9.2. Plot Manager Elements

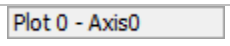
The Plot Manager is divided into the following elements. For more details on using these, see the Plot Manager topic.

Plot Window

The Plot window is the large right pane which graphically displays the plot.

- **Zooming**
  Use Ctrl+Mouse Wheel, Ctrl+Plus Sign, Ctrl+Minus Sign, or the zooming buttons on the toolbar.
- **Cursor**
  Move the hairline cursor to view the data at any sample. The data will appear in the Plot Details pane. To move the cursor, click and drag with the mouse or use the left and right arrow keys, Page Up, and Page Down. To add a second cursor, right-click the plot and choose **Show 2nd Cursor**.
- **Captured Data Bar**
  The Captured Data bar is located at the top of the Plot pane. This bar has a blue tick mark at every sample in the Plot pane that contains data captured from the RMC. This bar may appear as a continuous blue bar if the uploaded samples are dense.

Toolbar

Toolbar Item	Description
	**Open Plot File**
	**Save Plot(s)**
	**Print Plot**
	**Current Plot Template** indicates which plot template the **Capture** and **Trend** toolbar buttons apply to.
	**Upload Captured Plot**
	**Start Trend**
	**Stop Trend** or **Stop Plot Upload**
	**Zoom to Window** Horizontally and vertically fits the currently displayed plot to the plot window as follows: • Resets vertical scale for auto-scaled groups. • Resets vertical scale to the manual scale setting for manually scaled groups. • Zooms horizontally to fit the plot in the window for time plots. • Zooms horizontally to fit the plot to the defined x-axis scale for XY plots (auto or manual).
	Zoom to X Scale For time-based plots, zooms horizontally to fit the plot in the

	window. For XY plots, resets the horizontal scale to the manually set scale.
1:1	**Zoom to Sample Period** Zooms the plot so that one horizontal pixel is one sample of the plot.
	Zoom In
	Zoom Out
	Scroll to End Snaps the cursor to the end of a trending plot (the end with newest data). This may be necessary after clicking elsewhere in the plot while trending, in order to see the newest data in the **Plot Details** window.
	Vertical Zoom In
	Vertical Zoom Out
	Zoom to Y Scale For scale groups set to Auto Scale, resets the vertical scale to autofit the contents of the scale group. For scale groups set to Manual Scale, resets the vertical scale to the manually set scale.

Plotting Tab

On the Plotting tab, you can upload a captured plot, start a trend, or change plot template settings.

- **Edit a Plot Template**
  Click **Edit**.
- **Upload a Captured Plot**
  Click **Capture**.
- **Start a Trend**
  Click **Trend**.

History Tab

The History tab, located in the upper left pane, displays a list of all the plots you have uploaded. Every plot that is uploaded from the module is automatically saved under the **Recently Uploaded Plots** folder in this list. The following actions can be done in the History view:

- **View an Uploaded Plot**
  On the **History** tab, click the uploaded plot you wish to view.
- **Delete an Uploaded Plot**
  Select the plot, then click **Delete**.
- **Rename an Uploaded Plot**
  On the **History** tab, click the plot you wish to rename and press F2. Type the new name and press Enter.
- **Save Uploaded Plots**
  To save plots, on the Plot Manager toolbar, click **Save Plot(s)**.
- **Export an Uploaded Plot**
  To export plot data for use in other programs such as Microsoft Excel, right-click the plot and choose **Export Plot**. See Saving and Exporting Plots for details.
- **View and Manage Auto-Saved Plots**
  RMCTools automatically saves a backup copy of every uploaded plot to an internal folder on the PC in order to prevent data loss if the user forgets to save plots before closing RMCTools.

To view backup copies of uploaded plots, expand **Auto-Saved Plots** and click **View Auto-Saved Plots**.

To change the settings of auto-saved plots, expand **Auto-Saved Plots** and click **Auto-Save Settings**.

For details, see Auto-Saved Plots.

Tuning Tab

The Tuning Tools, located on the Tuning tab in the upper left pane, is for tuning the axes. The Tuning Tools streamlines the tuning process by providing a single place where you can issue commands, change and download gains, and view plots of moves. In addition, plots are automatic uploaded after sending a command. All uploaded plots are listed on the History tab, and include the tuning parameters used for each plot, which can be viewed in the Parameters pane.

See the Tuning Tools topic for details.

Plot Details

The Plot Details tab, located in the bottom left pane, displays the plot data at the sample at which the hairline cursor in the plot is located. The color of each data item in the Plot Details corresponds to the color of each line in the plot. Values of DWORD data type are not displayed with a line, but the individual bits can be viewed in the Plot Details by clicking the '+' preceding the tag name.

Clicking an item in the Plot Details list will highlight the item in blue, and the vertical scale in the plot will change to apply to that item, if the item has a plotted line. All items that the currently displayed vertical scale applies to will be highlighted in light blue.

To change the data to be plotted, see the Plot Template Editor topic.

From the Plot Details, the following items can be changed for each data item:

- **Show/Hide Line**
  To show or hide a plot line do any of the following:
  - Click the color box of any item in the Plot Details to toggle the plot line on or off.
  - If the item in the Plot Details is selected, press the Spacebar to toggle the plot line on or off.
  - Double-click the item in the Plot Detail to toggle the plot line on or off.
  - Right-click an item in the Plot Details and choose **Show Line** or **Hide Line**, or **Hide All But This** or **Show All Lines**.
- **Set Pen Color**
  Right-click an item in the Plot Details and choose **Set Pen Color**. Choose a color and click **OK**.
- **Set Scale**
  Right-click an item in the Plot Details and choose **Set Scale**. Choose a scale option and click **OK**.
- **Format Number**
  This sets the number of decimal places to display. Right-click an item in the Plot Details and choose **Format Number**. Choose a format option and click **OK**.
- **Add Pen**
  If an item does not have a pen associated with, you can right-click the item in the Plot Details, choose **Add Pen**, then select a color to show the line in the plot.
- **Set X Axis**
  You can view the plot as an XY plot with the selected data item as the horizontal axis (x-axis) value, with all the other items plotted on the vertical axis (y-axis).
- **Clear X Axis**
  If the selected item is currently an X Axis, this will change the axis to a time-based plot, where the x axis is time.

- **Set X Axis Scale**
  If the selected item is currently an X Axis, this will set the scale of the horizontal axis, with options of Auto Scale and a Manual Scale.

Absolute Time

In the Plot Detail window, you can choose to display the absolute time of the cursor. This absolute time corresponds to the time displayed in the Event Log, and is very useful for correlating the Event Log with the plot when troubleshooting.

To show the absolute time:

- On the **View** menu, click **Show Absolute Time**.

Parameters Pane

The Parameters tab, located in the bottom left pane, contains the axis gains and related parameters for each axis that is included in the currently displayed plot. This information is uploaded immediately after the plot has completed uploading. This information is also saved when a plot is saved.

See Also

Plot Overview | Plot Manager Overview | Using XY Plots

4.9.3. Plot Template Editor

To access the Plot Template Editor:
On the **Plots** menu, choose **Plot Manager**. On the **Plotting** tab, click **Edit Plot Templates**.

Use this editor to set up the RMC plots. For example, you can change the number of plots, choose which registers to plot, set the pen color, etc.

After making changes, click **OK**. If RMCToos is online with the controller, the changes will be downloaded to the controller.

Managing Plot Templates

Plot templates can be added, deleted, or reordered.

- **Add a Plot Template**
  Click **Add**.
- **Delete a Plot Template**
  Click **Delete**.
- **Move Template Up or Down.**
  Click **Up** or **Down**.

Editing Individual Plot Templates

For any plot, the following items can be set:

- **Name**
  The name of the plot. This can be used to provide a descriptive name for user reference.
- **Sample Interval**
  Specifies the time between each sample. A smaller value will result in finer detail on the plot. Because the memory for a plot is limited, a larger Sample Interval will allow for a longer Plot Duration. The **Sample Interval** cannot be smaller than the Loop Time of the controller.
- **Capture Duration**
  Specifies the length of a captured plot. A captured plot is one that is triggered by a motion

command, the Start Plot (100) command, or Trigger Plot (102) command, and is stored internally. If the captured plot duration value is too long for the available plot memory in the RMC, an error bubble will appear. The memory usage depends on the number of data items, the sample interval, and the capture duration.

- **Trend Duration**
  Specifies the length of a plot trend. Trending continuously uploads the plot data while it is being captured in the RMC. When the trend reaches the duration, the trend will continue and old data will be lost. If the trend duration value will result in excessive use of PC RAM, an error bubble will appear. The RAM usage depends on the number of data items, the sample interval, and the trend duration.
- **Default or Custom**
  Each individual plot can be set as either a **Default** or **Custom** plot, as described below. To add or remove plot quantities, the plot template must be set to **Custom**. See Using Custom Plot Templates.

Default and Custom Plots

Default Plots

Default plots are managed by RMCTools. This option makes managing plots easy and is recommended for most users. The user can choose the axis to plot, and can change the **Plot Duration** and **Sample Interval**. Changing other settings requires choosing Custom Plot.

RMCTools will automatically determine which registers should be plotted and how they are displayed, based on the axis type. For example, if Axis 1 is a position-force axis, and a plot is set to be a default plot for Axis 1, the plot will automatically be set to include the Target and Actual Position, Velocity, and Force, Control Output, Status bits, and Mean Squared Error for Position and Force.

Custom Plots

Custom plots can be fully configured by the user. For more details, see Using Custom Plot Templates.

Plot Template Memory Usage

To determine how much memory the Plot Templates will use, below the Plot Template list, click the **Template Memory Used** link. The dialog shows the amount of RMC memory that will be used for the defined templates, and the memory available.

The memory limits are listed in the Plot Overview topic.

See Also

Plot Overview | Using Custom Plot Templates

4.9.4. Using Custom Plot Templates

A plot template can be set as either a Default or Custom plot template. Default plot templates are managed by RMCTools and cannot be edited. RMCTools will automatically determine which registers should be plotted and how they are displayed, based on the axis type. In order to change items in an individual plot template, such as plotted registers and pen color, the plot must be set to **Custom Plot**.

To set a plot to Custom:

1. Open the Plot Template Editor.
2. Choose a plot from the list on the left.
3. On the right side, choose **Custom Plot**.

Changing a Custom Plot

To edit the plot templates, make the desired changes in the Plot Template Editor, then click the **Download** button in the Plot Template Editor toolbar. Update Flash to retain this plot set in the controller even after power is removed from the controller.

Any changes made to a plot in the Plot Template Editor will apply to all subsequent plots uploaded in the Plot Manager for the selected plot template.

Adding and Removing Plot Quantities

Adding a Plot Quantity

1. Click **New Quantity**.
2. In the Plot Quantity Selection Tool, browse to the register you need, then click **Add** or press the space bar. Repeat to quickly add consecutive registers. Use the **Groups** tab to quickly add the basic status registers or tuning terms of any axis.
3. Click **OK** when done selecting new quantities.

Adding Quantities for Multiple Axes At Once

You can choose to add quantities for many axes at once. This can be done by duplicating the existing quantities in the plot template, or by adding new quantities to the plot template and duplicating them.

1. If the plot template does not already contain the quantities you wish to duplicate, then, In the Plot Quantity Selection Tool, add the desired registers from a single axis, so that they appear in the **Quantities to Add to Plot Template** box.
2. Click **Duplicate for axes**. The Plot Quantity Duplication Tool will open.
3. In the **Quantities to Duplicate** box, choose the quantities you wish to duplicate.
4. In the **Axes for Quantity Duplication** box, choose all the axes for which you want to add the selected quantities. Use the **Select All** and **Clear All** buttons to speed up the process if there are many axes.
5. Click **OK** to close the Plot Quantity Duplication Tool. The new quantities will now appear in the **Quantities to Add to Plot Template** list.
6. Click **OK** when done selecting new quantities.

Removing a Plot Quantity

1. Select the quantity to delete, and click **Delete Quantity** or press Delete.

Changing a Plot Quantity

1. Select the **Formula** cell of the desired quantity, then click the ellipsis button.
2. Select the desired register and click **OK.**

Changing the Order of the Plot Quantities

1. Select the quantity to move, and click **Move Up** or **Move Down** or press Ctrl+Up or Ctrl+Down.

Customizing Plot Quantities

Filter

You can choose to filter a plot quantity. This is most useful for the Actual Velocity and Actual Acceleration, which tend to be noisy. To choose a filter, select **Filter On** in the Filter cell of the desired Plot Quantity.

Number Format

You can set the number of decimal places for plotted quantities of REAL type:

1. Click the **Format** cell for the desired plot quantity and click the ellipsis button.
2. Choose from the following:
    a. **Fixed Number of Decimal Places**
    The number will be represented with a fixed number of decimal places. For more details, click the **Help** button.

b. **Fixed Number of Digits of Precision**
The number will be represented with a fixed number of digits of precision. For more details, click the **Help**button.

3. Click **OK**.

Show/Hide Plot Line

The Graph column indicates whether this quantity should be graphed with a line in the plot. All quantities will always be displayed in the Detail window, even if the line is not graphed in the plot.

Pen Color

To set the color of the graphed line:

1. Click the **Pen Color** cell for the desired plot quantity and click the ellipsis button.
2. Choose a color from the color picker and click **OK**.

Scale

To set the scale of a plot quantity:

1. Click the **Scale** cell for the desired plot quantity and click the ellipsis button.
2. Choose from the two options:
   a. **Auto Scale**
   RMCTools will automatically determine the scale based on the values of all the plot quantities in the same scale group in the uploaded plot.
   b. **Manual Scale**
   The plot scale will be set to the specified range.
3. Click **OK**. Notice that the new scale will be applied to all plot quantities in the same scale group.

Scale Groups

All plot quantities with the same Scale Group will share the same scale in an uploaded plot. Changing the scale for any quantity will apply the same scale to all quantities in the same scale group.

To choose a Scale Group:

1. Click the **Group** cell for the desired plot quantity and click the drop-down button.
2. Choose from the following options:
   a. **Auto**
   RMCTools will automatically assign the group for this item. This is indicated by "Auto-" followed by a letter that indicates the group chosen by RMCTools. For example, all position registers are assigned to group A, all velocities to group B, and so on.
   b. **None**
   The plot quantity will not be part of a scale group.
   c. **Number**
   Choose a number to assign the plot quantity to your own user-defined scale group.

Trigger Settings

To edit the trigger settings, click **Edit Trigger Settings**. For details on using the plot trigger, see the Triggering Plots topic.

Setting	Description
Enable Automatic Trigger	If this checkbox is set, a plot will automatically trigger when the action in the Trigger Type box occurs. Notice that the plot must be rearmed before it will trigger.
Trigger Type	Only valid if the **Enable Automatic Trigger** is selected. Specifies the action that will cause a plot to automatically trigger.

	Motion Commands: The plot will trigger when a motion command is issued to the specified axis.
Rearm Options	Specify how the plot is to be rearmed. **Manual:** The plot must be rearmed manually by issuing the Rearm Plot (103) command. This option lets you set the Trigger Position to any value. **Automatic:** The plot is rearmed automatically after a triggered. The plot is then immediately ready to be triggered again. The trigger position must be set to 0%.
Trigger Percentage	Specifies how much of the plot contains data from before the trigger. 0% means there is no data from before the trigger. 100% means there is data only from before the trigger.

XY Plot

To define a plot as an XY plot:

1. Check the **XY Plot** check box.
2. In the **X Axis** box, select the plot item to be the x axis (horizontal) value in the plot.
3. The x-axis scale of the plot will default to be automatic. If you wish to set a scale value, check the **Manual Scale** box and enter the desired range.

For more details, see Using XY Plots.

See Also

Plot Template Editor

4.9.5. Using XY Plots

An XY plot displays all the data items in a plot relative to any other data item in the plot, with that other data item being the horizontal (x) axis, and the remaining data items being plotted on the vertical (y) axis. An XY plot requires that a data item in the plot be set as the X axis. If a plot does not have an X Axis item, then the plot data items will be displayed versus time, with the time as the horizontal (x) axis.

XY plots only apply to how the plot is viewed in the Plot Manager. The underlying data is still stored in the same form as a typical time-based plot; that is, the data is stored as samples captured at each plot sample interval. An uploaded XY plot can be easily be viewed as a time-based plot, and a time-based plot can easily be viewed as an XY plot. Any data item in any plot can be selected to be the X-axis value.

Defining an XY Plot Template

To define a plot template as an XY plot:

1. In a plot template, choose **Custom**.
2. In the **XY plot** section, check the **XY Plot** check box.
3. In the **X Axis** box, select the plot item to be the x axis (horizontal) value in the plot.
4. The x-axis scale of the plot will default to be automatic, meaning that the plot will scale horizontally to fit the contents. If you wish to set a fixed scale value, check the **Manual Scale** box and enter the desired range.

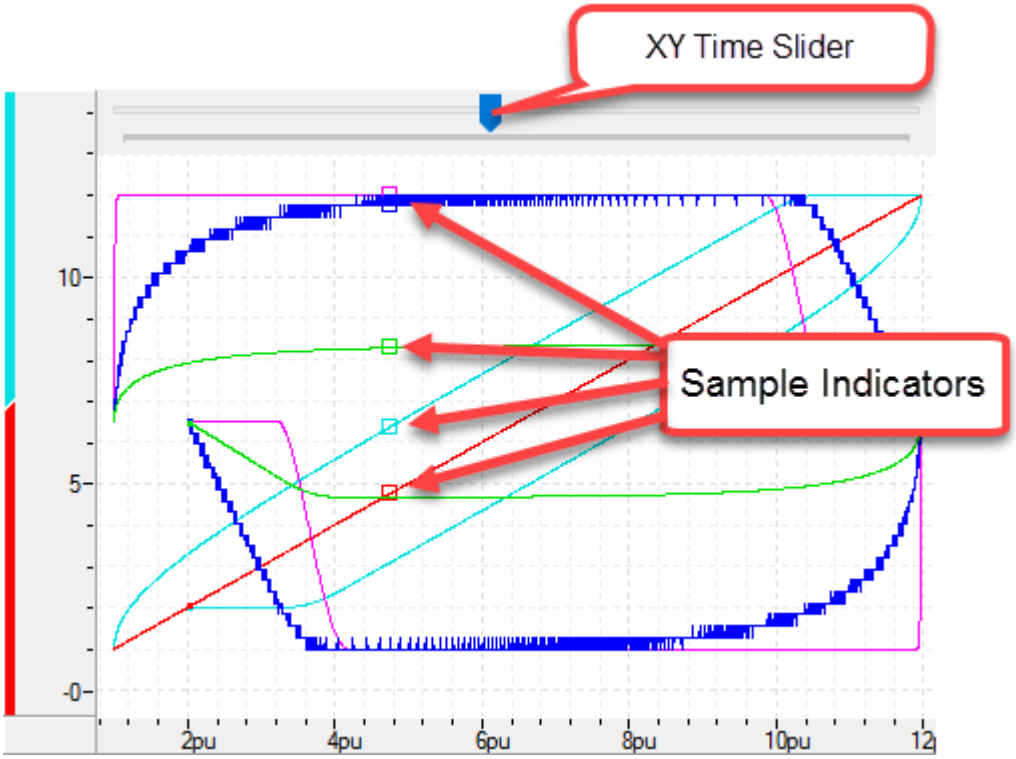
After applying the changes, every plot uploaded by this template will be displayed as an XY plot.

Notice that the X-axis item may still be plotted (and will appear as a diagonal line on the plot). This is controlled by checking or unchecking the **Graph** box for the item in the **Plotted Data Items** table.

Using the XY Time Slider

The XY Time Slider in the Plot Manager is very useful for tracking the progression of an uploaded plot in time, or determining the exact values of the data at a specific point in time.

The XY Time Slider is located at the top of an uploaded plot. It covers the entire time duration of the plot. When the slider is clicked or dragged with the mouse, the Sample Indicator boxes will show the data points for that given point of time in the plot. In the Plot Details window, the **Data Values** column will show the values of the data points, as well as the time.



Using the Cursors

The cursors are useful for approximating the values of the data items at some location in the plot and making measurement in the plots. The Plot Details window will display the value of a cursor relative to the scales for each data item. It will not display the exact values of the data items, since exact data values do not exist for every location in the plot (to view exact data values, use the XY Time Slider). In the Plot Details window, the x-axis values of the cursors will be given by the X Axis item, and the other items will list their respective y values.

How to:

- **Place a cursor:**
  Click anywhere in the plot. In the Plot Details window, the **Cursor Value** column will display the cursor values.
- **Place a second cursor:**
  Right-click anywhere in the plot and choose **Show 2nd XY Cursor**. In the Plot Details window, the **Cursor 1** and **Cursor 2** columns will display the cursor values, and the **Diff** column will display the difference between the values.
- **Move a cursor:**
  Click a cursor and drag it. The data values in the Plot Details window will update as the cursor is moved.

- **Hide the cursors:**
  Right-click anywhere in the plot and choose **Hide XY Cursors**.
- **Hide only the second cursor:**
  Right-click anywhere in the plot and choose **Hide 2nd XY Cursor**.

Switching Between XY and Time

An uploaded XY plot can be viewed as a time-based plot, and an uploaded time-based plot can be viewed as an XY plot.

How to:

- **Change an XY plot to a time-based plot:**
  Right-click anywhere in the plot and choose **Convert to Time Plot.**
  Or:
  In the Plot Details window, right-click the item labeled **X Axis** and choose **Remove as X Axis**.
- **Change a time-based plot to an XY plot:**
  Right-click anywhere in the plot and choose **Convert to XY Plot.** In the **X-Axis** box, choose the item that you wish to be the X axis and click **OK**.
  Or:
  In the Plot Details window, right-click the item that you wish to be the X axis and choose **Set as X Axis**.

Changing the X-Axis in an XY Plot

To change the x-axis item in an uploaded XY plot:

- In the Plot Details window, right-click the item that you wish to be the X axis and choose **Set as X Axis**.

Changing the Scale in an XY Plot

The X-axis item of the plot is listed in the Plot Details window, and has two scales: x-axis and y-axis. Setting the y-axis scale of the X-axis item (if the x-axis item is being plotted vertically as well) is the same procedure as for the other data items in the plot: right-click the item and choose **Set Scale**.

To change the x-axis scale in an XY plot:

1. In the Plot Details window, right-click the item labeled **X Axis** and choose **Set X Axis Scale.**
2. Choose **Auto Scale** or **Manual Scale**.
   The Auto Scale option will scale the plot horizontally to fit the contents of the plot. The Manual Scale option will set a fixed scale. You can zoom in from the fixed scale.

Zooming an XY Plot

	Mouse	Keyboard	Toolbar
Horizontal Zoom	Ctrl+Mouse Wheel	Ctrl+Plus Sign Ctrl+Minus Sign	
Horizontal Scroll	Shift+Mouse Wheel	-	-
Horizontal Reset[1,2]	-	-	
Vertical Zoom	Ctrl+Shift+Mouse Wheel	Ctrl+Shift+Plus Sign Ctrl+Shift+Minus Sign	
Vertical Scroll	Mouse Wheel	-	-
Vertical Reset[1]	-	-	
Zoom to Window[2]	-	-	

[1]Resets auto-scaled items to fit the data. For manually scaled items, resets the scale to the manual setting.

[2]For manually scaled items, resets the scale to the manual setting. Manually setting the horizontal scale will set the maximum horizontal zoom for the plot. If the manual scale does not show all of the plot, you must change the scale in order to see the entire plot.

See Also

Plot Manager Overview | Using Custom Plots

4.9.6. Plot Quantity Selection Tool

To access this dialog:
In the Plot Template Editor, on any plot tab, choose **Custom Plot**, and click **New Quantity**.

Use this dialog to add one or more plot quantities to the plot template. After selecting the desired quantities, click **OK** to close the dialog.

Adding Individual Registers

1. On the **Registers** tab, browse to the desired register.
2. Click **Add** or press the space bar to choose the register. Repeat to add consecutive registers quickly. The registers will appear in the **Quantities to Add to Plot Template** list.

Adding Groups of Registers

Use groups to quickly add the basic status registers or tuning terms of any axis.

1. On the **Groups** tab, double-click an axis. The registers will appear in the **Quantities to Add to Plot Template** list.

Removing and Moving Registers

Use the **Remove**  , **Move Up**  , or **Move Down**  buttons to remove or re-order registers in the **Quantities to Add to Plot Template** box.

Adding Quantities for Multiple Axes At Once

You can choose to add quantities for many axes at once. This can be done by duplicating the existing quantities in the plot template, or by adding new ones to the plot template and duplicating them.

1. If the plot template does not already contain the quantities you wish to duplicate, then, from the **Registers** tab, add the desired registers from a single axis, so that they appear in the **Quantities to Add to Plot Template** box.
2. Click **Duplicate for Axes**. The Plot Quantity Duplication Tool will open.
3. In the **Quantities to Duplicate** box, choose the quantities you wish to duplicate.
4. In the **Axes for Quantity Duplication** box, choose all the axes for which you want to add the selected quantities. Use the **Select All** and **Clear All** buttons to speed up the process if there are many axes.
5. Click **OK** to close the Plot Quantity Duplication Tool. The new quantities will now appear in the **Quantities to Add to Plot Template** list.

Advanced Quantities

On the **Advanced** tab, you can add the following:

- **Mean Squared Error**

  The Mean Squared Error calculates a single number that records indicates how closely two registers tracked each other during the plot. The Mean Squared Error is typically using during manual tuning to compare how close two plot quantities are, such as Target and Actual Position. The smaller the number, the closer the items tracked.

  Choose **Standard Error Quantity** to select the Mean Squared Error between the Target and Actual Values of an axis control quantity. Choose **Custom Error Quantity** to select the Mean Squared Error between any two registers that you choose.

- **Rate of Change of a Register**

  This option will plot the calculated rate of change of a register. If the original register is already included in the plot, plotting the rate of change does not require the plot to capture another register. This allows the plot template to include more quantities than the number of data sets that the plot is limited to. Also, because using the rate of change may reduce the number of plot items to upload, the plot can upload faster, which will be noticeable when using the slower monitor ports on the RMC75S and RMC75P.

See Also

Plot Template Editor | Plot Overview

4.10. Tuning

4.10.1. System Identification

To access the System Identification Dialog:
On the **Plots** menu, point to **View**, then click **System Identification**.

The System Identification tool calculates system models for the currently displayed plot. The System Identification tool assigns a score to each model, with a lower number indicating a better fit. This tool does not apply the calculated models to the axis. However, you can manually enter these values in the axis' model parameters.

Use the Tuning Wizard to calculate a system model and apply the values to the axis parameters. See the Modeling topic for details on how the system model is used.

ID a Portion of the Plot

The System Identification tool allows you to calculate a system model for only a portion of the plot:

1. On the **Plots** menu, choose **View**, and click **Show 2nd Cursor**.
2. Use the mouse to drag the cursors to select the desired portion of the plot.
3. On the **Plots** menu, choose **View**, then click **System Identification Tool**.

To remove the second cursor, on the **Plots** menu, choose **View**, and click **Hide 2nd Cursor**.

Advanced

Click **View Log** to view the calculation results. The log also shows the parameters calculated for type 0 or type 1 systems, and zero, first, and second order models.

In a Type 0 system, the controlled variable is roughly proportional to the Control Output.

In a Type 1 system, the rate of the controlled variable is roughly proportional to the Control Output.

See Also

Modeling

4.10.2. Tuning Tools

To access the Plot Manager:
On the **Tools** menu, click **Tuning Tools**.

Use the Tuning Tools to tune the axes. The Tuning Tools pane streamlines the tuning process by providing the following:

- **Access to Tuning Wizard and Autotuning**
- **One-Stop Tuning**
  The Tuning Tools pane provides a single place where you can issue commands, change and download gains, and view plots of moves.
- **Automatic Plot Upload**
  Plots are automatic uploaded after you issue a command.

Overview

The Tuning Tools pane contains all the tools you need to tune an axis:

Axis

Choose the axis to tune, and choose position or pressure or force.

Click **Tuning Wizard** to start Autotuning or to tune from an existing plot.

Click **Gain Calculator** to choose gains using the slider bar.

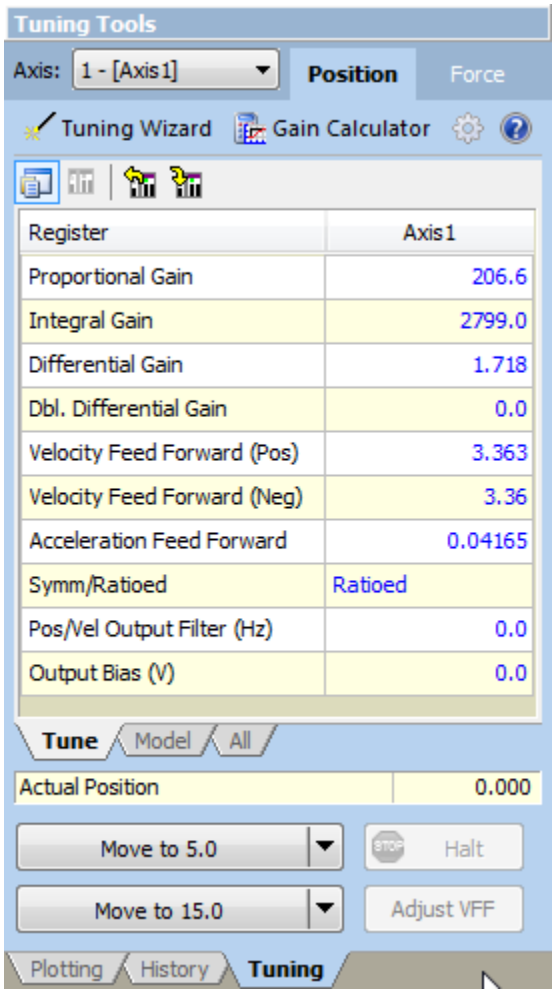
Tuning Parameters

You can change the tuning parameters and then download them to the controller by clicking the Download button. For challenging tuning systems, you also have access to all the axis parameters, just as in the Axes Parameters pane in the Axis Tools.

Status Registers

This section provides you with basic axis information. For example, the Actual Position is helpful so you know which direction to move the axis.

You can choose which registers appear by clicking the **Customize Tuning Tools** button.



Command Buttons

Tuning involves repeatedly issuing the same commands to move the axis back and forth. The command buttons (**[Click to Set up]**) simplify this process. Click the button to customize it.

Using the Tuning Tools

After you have set up your actuator and feedback, and completed the scale and offset, you are ready to tune the axis. The following procedure describes how to use the Tuning Tools to tune your axis.

Autotuning

Position Axes

To autotune a position axis, click **Tuning Wizard** and follow the wizard. After completing autotuning, you must test the tuning by moving the axis. Read the **Manual Tuning** section below to learn how to set up the command buttons to move the axis and view plots of the motion.

Pressure or Force Axes

The Tuning Wizard can calculate gains for pressure or force axes given an existing plot that shows the pressure or force changing. On the **Pressure** tab, click **Tuning Wizard**.

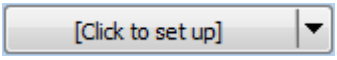
Manual Tuning

This section describes how to use the Tuning Tools during the tuning process. It does not describe the order of changing gains. For details on which gains to change when during the tuning procedure, see the Tuning Overview topic.

1. Set Up the Command Buttons

For position axes, click the **Position** tab. For pressure or force axes, click the **Pressure** or

Force tab. Then, set up the command buttons so that one button will move the axis one direction, and the other will move it in the other direction.

a. Click the down arrow on the command button 

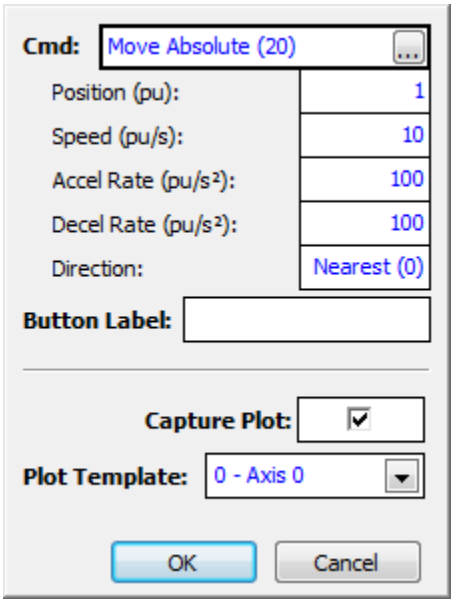
.

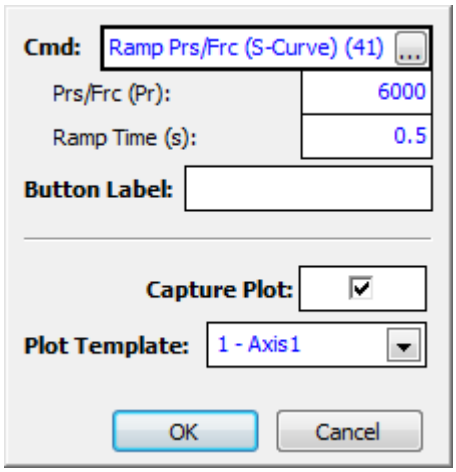
b. Enter the parameters for the command, then click **OK**.

c. Make sure the correct plot template is selected.

Tip:
For a typical hydraulic cylinder position axis, the **Accel** and **Decel** parameters of the Move absolute command are typically on the order of 10 to 100pu/sec^2. The speed is typically between 1 and 30pu/sec.

Examples:





d. Repeat the previous step for the other command button. For position axes, make sure to enter the same velocity, acceleration, and deceleration, but a different position. For pressure/force axes, use the same ramp time, but a different pressure/force value.

e. Click **OK**. Now you will see that the command buttons are labeled.

2. Send the Command

a. Click the command button for the direction you wish to tune. This will send the command and automatically upload the plot from the RMC.

b. Click the other command button to move the axis back to where it was. This will send the command and automatically upload the plot from the RMC.

3. Change a Gain

Based on the plot, change a gain, and download it.

a. Look at the plot that was automatically uploaded from the RMC and determine which gain you wish to change. For details on which gains to change when during the tuning procedure, see the Tuning Overview topic.

b. Click the Download button to download the change to the RMC.

4. Repeat

Repeat steps 2 and 3 until you are satisfied with the tuning, but keep these items in mind:

a. For position axes, you can use the **Adjust VFF** button to have the RMC automatically determine the Velocity Feed Forwards in both directions. Make a move, then click the **Adjust VFF** button. Repeat for the other direction.

b. For pressure/force axes, you can use the **Hold Prs/Frc** button to enter pressure/force control. For details, see the Tuning Pressure/Force topic.
c. Soon after you have gained decent control over the system, you should increase the speed or rate of your moves to the speed or rate you expect to use during machine operation.
d. To see how your tuning has progressed, or to see which gains you used several moves ago, click the **History** tab and choose a plot. The **Parameters** tab below the **Plot Details** window displays the gains that were used for each plot.

See Also
Plot Overview

4.11. Programming

4.11.1. Programming Folder Overview

The Programming folder in the Project Pane contains the components for programming the RMC. These include:

- Programming Properties
- Variable Table Editor
- Discrete I/O Configuration
- Program Triggers
- Step Editor (User Programs)

For details on programming these items, see the Programming Overview topic.

See Also
Programming Overview

4.11.2. Step Editor

To access this editor:
Expand the desired controller in the Project pane, expand **Programming**, and expand **User Programs**. Double-click an existing User Program, or right-click **User Programs** and click **New Program** to create a new one.

Use the Step Editor to create a User Program. The Step Editor provides an easy and intuitive programming interface.

User Programs allow you to run a sequence of commands on the RMC. A User Program consists of a series of *steps* that are linked together in sequences. The steps consist of the following:

- Command - This is the command that will be issued to the specified axis or axes on the RMC.
- Link - This specifies which step to jump to next and when to jump to that step.

Example User Programs

For examples of user programs, see the Programming Examples Overview topic.

Using the Step Editor

Steps

- **Adding a Step:**
  On the Step Editor toolbar, click the **Insert New Step** button.
  Or, right-click in the space below the step number and choose **Add Before** or **Add After**.
  Or, select a step and press Insert to add a command before, or Alt+Insert to add a step after.
- **Delete a Step:**
  On the Step Editor toolbar, click the **Delete Step** button.
  Or, right-click in the white space below the step number and choose **Delete**.
- **Moving a Step:**
  Click in the white space below the step number and drag the step to the desired location.
  Or, select the step and use the **Move Step Up** and **Move Step Down** buttons on the toolbar.

When adding and deleting steps, the Step Editor automatically updates the Link Jump To numbers. For example, if step 3 has a Jump To number of 4, and you insert a step before step 3, then the old step 3 becomes the new step 4. The Step Editor automatically increments its Jump To number from 4 to 5.

If a deleted step was linked to with a number from some other step, a warning will appear in the Output window.

Commands

There are two ways to choose a command:

- **Choosing a command:**
  In the **Command** box, click the ellipsis button to choose a command from the command tree.
  Or, in the **Command** box, begin typing the name or number of the command, then choose the command from the pop-up list.
- **Adding a new Command:**
  On the Step Editor toolbar, click the **Insert Command** button .
  Or, right-click close to a command and choose **Add Before** or **Add After**.
  Or, select a command and press Ctrl+Insert to add a command before, or Ctrl+Alt+Insert to add a command after.
- **Delete a Command:**
  On the Step Editor toolbar, click the **Delete Command** button .
  Or, right-click close to the command such that the entire command is selected, then choose **Delete**.
- **Moving a Command:**
  Click close to the command such that the entire command is selected, then drag the step to the desired location.
  Or, click close to the command such that the entire command is selected, and use the **Move Command Up** and **Move Command Down** buttons on the toolbar.

Entering Command Parameters

If the command has any parameters, enter their values. In most parameters, you may enter a number, tag, or an expression. If you use a tag name, register, or expression, it must evaluate to the data type required by the parameter, typically a REAL.

Tip:
For help on the command's parameters, click the command box and press F1.

- Click a parameter box.
- To enter a **number**, type the number.

- To enter an expressions, type the expression. The Expression Browser is not available for command parameters, but the auto-complete will assist you.
- To enter a tag, click the ellipsis button and choose the tag from the list. Or, start typing the tag name and then use the arrow keys to select the address.

> **Note:**
> A tag may be a variable or any register in the RMC.

Selecting Commanded Axes

The Commanded Axes specifies which axes to issue the command to.

- Click the **Command Axes** button and choose the axes you wish to issue the command to. For example, selecting Axis 0 and Axis 1 will make the command be issued to both axes.
- The **Default Axis** and **Use Expression** options are typically used only in advanced applications. If you choose Default Axis, the command will be issued to the same axis as the Start Task command was issued to. The **Use Expression** option can be used to programmatically select the commanded axes.
- Click **OK** to close the commanded axes list.

Important: A maximum of one non-immediate command per axis can be issued in a single step. If you try to issue more than one command to an axis per step, the verify will report an error and you will not be able to download the programs.

Choosing a Link Type

1. Click the down-arrow in the **Link Type** box and choose the desired link type.
2. If the link type has a condition, enter it according to the table below:

Link Type	To enter a Link condition:
Immed	N/A
Jump	N/A
Delay	Type a number, or click the ellipsis button to choose a tag.
Wait For	Click the ellipsis button to open the Link Condition Wizard.
Cnd Jump	Click the ellipsis button to open the Link Condition Wizard.
End	N/A

3. If you chose **Jump**, **Delay**, or **CndJump**, enter the step to jump to in the **Jump To** box(es). You can enter a step number, "Next", or a step label.

Adding A Comment

- On the Step Editor toolbar, click the **Edit Step Comment** button. Or, right-click in the space below the step number and choose **Edit Step Comment**.
- Type a comment and click **Save**.

Adding A Label

- On the Step Editor toolbar, click the **Edit Label** button. Or, right-click in the space below the step number and choose **Edit Label.**
- Type a label and click **Save**.

Adding Local Variable Declarations

If you are using the Expression (113) command, you can also define local variables for use in the step. See Local Variables in Expressions for details.

- On the toolbar, click the **Declarations** button.

Copy/Paste

The Step Editor fully supports copy and paste. You can copy and paste any section of a step, entire steps, multiple steps, etc. The copy and paste also works between instances of RMCTools.

Undo

Undo is available for all editing actions in the Step Editor. Undo is also available within an individual expression, although once an expression edit is accepted, the undo history internal to the expression box is cleared.

- To undo a change, on the Edit menu, choose **Undo**, or press Ctrl+Z.
- To reverse an undo action, choose **Redo**, or press Ctrl+Y.

Printing

To print the currently open user program, on the Step Editor toolbar, click the **Print** button . Or, on the **File** menu, choose **Print**.

See Also

User Program Overview

4.11.3. Program Triggers

To access the Program Triggers: In the Project pane, expand **Programming**, and double-click **Program Triggers**.

The Program Triggers start user programs when user-specified events occur. For example, you can set up the Program Triggers to start a user program when an input turns on, or to start a user program when a variable becomes a certain value. This is a good way to start RMC user programs from a PLC or host controller.

For a real programming example of using the Program Triggers, see the Example: Jogging an Axis topic.

Triggers

The Program Triggers can contain the following number of **Triggers:**

	Number of Triggers
RMC75S	64
RMC75P	64
RMC75E	64
RMC150E	64
RMC200	128

A trigger is one complete row in the table, consisting of a **Condition** and **Task Actions**. The RMC checks all the conditions every loop time. When a condition becomes true, the user program(s) specified by the user are started on the specified tasks. The user programs are started only on the rising edge of the entire condition becoming true (one-shot). The user program will be started on that task even if the task is already running a user program. The task will stop running the current user program an run the new one.

The Task Actions are performed only on the rising edge of the entire condition. Therefore, when the entire condition becomes true, the specified Task Actions will be performed. While the condition remains true, the trigger will not start or stop anything.

Important!
The Task Actions are performed ***only*** on the rising edge of the ***entire condition***. Therefore, when the entire condition ***becomes*** true, the specified Task Actions will be performed ***once***. While the condition ***remains*** true, the trigger will ***not*** start or stop anything.

Conditions

The Program Triggers can handle simple or complex conditions. For example, a condition can check for certain inputs to be on *and* the Actual position to be greater than a certain value. Conditions are created using expressions and are therefore very flexible. For details on expressions, see the Expressions topic. To find out how to create a condition, see **Creating a Trigger** below.

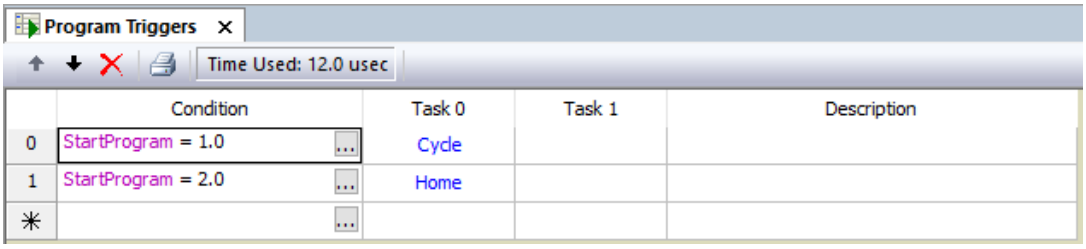
Task Actions

Task Actions can start or stop Tasks. Tasks run User Programs. For each trigger, you can define which Tasks to start, stop, or do nothing to. The Program Triggers will allow a Task to be started only at a label in a User Program. To find out how to set up the Task Action, see **Creating a Trigger** below.

The Program Triggers has one column for each Task. To increase the number of tasks, use the **General** page of the Programming Properties dialog.

Example

This is an example of two triggers in the Program Triggers:



The first trigger will cause the user Program "Cycle" to be started on Task 0 when the variable StartProgram becomes 1, and the second trigger will cause the user Program "MoveHome" to be started on Task 0 when the variable StartProgram becomes 2. As illustrated here, writing to a variable that triggers a user program is an easy way for a PLC to start a user program in the RMC.
Notice that the user program will be started when the condition *becomes* true. It will not restart while the condition remains true.

Order of Execution

If there are multiple triggers (rows), they will be executed in order from top to bottom. Therefore, if two triggers become true in the same loop time, and they start different user programs on the same task, the user program specified by second of these two triggers will end up running on the task. This is because the RMC will always do the last thing it is instructed to do.

Typically, in well-structured programs, this type of conflict does not occur.

How to Create and Manage Triggers

Create a Trigger

1. In the last row of the Program Triggers, click the **Condition** cell, then click the ellipsis button ... . The New Condition Wizard will open.
2. Select the type of condition you want to create, and continue through the wizard to complete the condition.

Note:
If you want to create a complex condition, double-click the **Condition** cell. It will let you create a custom condition. See the Expressions topic for details on creating expressions.

3. In each **Task** cell for the condition you just entered, click the cell and select one of the following options from the drop-down box:

    1.

    - **User Program**
      Choose a user program from the drop-down box.
    - **No Entry**
      If you do not want the trigger to affect the Task, do not enter anything in the cell. To remove an entry, click the cell and press Delete.
    - **<StopTask>**
      The Task will be stopped immediately when the condition becomes true.

Note:
The Program Triggers Editor has one column for each Task. To increase the number of tasks, use the **General** page of the Programming Properties dialog.

Move Triggers

To move rows up or down, select one or more rows, then use the **Move Up** and **Move Down** buttons. To delete rows, select one or more rows, then click the **Delete Row** button. The order of the triggers can be important. If several conditions become true at the same time and multiple actions are triggered on the same Task, the last action for that Task will be the one that runs.

Download the Program Triggers

In the Project Pane, right-click **Programming** and click **Download Programs**.

After the Program Triggers are downloaded, the RMC must be put into RUN mode before the Task Actions will be performed whenever when the associated conditions becomes true.

Print the Program Triggers

To print the Program Triggers, on the Program Triggers Editor toolbar, click the **Print** button. Or, on the **File** menu, choose **Print**.

Special Case: When the RMC Enters RUN Mode

With older firmware, the Program Triggers will trigger when a condition was true when the RMC enters Run mode. Newer firmware has a selection for this behavior, and defaults to triggering only when a rising edge of the condition occurs, and therefore, will not trigger simply because a condition happens to be true when the RMC enters Run mode. RMC75/150 firmware versions 3.64.0 and and later and RMC200 firmware 1.01.0 and later support the selection.

The Program Trigger behavior can be selected in the Programming Properties dialog. The following two options are available:

- **Trigger on rising edge of condition only** will start the specified action only when the condition experiences a rising edge, which means the condition changes from false to true. This is the recommended option.
- **Trigger on rising edge of condition OR when the condition is true upon entering Run mode** will start the specified action when the condition changes from false to true. It will also start the specified action if the condition happens to be true when the RMC enters Run mode, even though there was not a specific rising edge of the condition at that time.
  This option is not recommended, but exists to be compatible with the functionality of older firmware.

Legacy Behavior

With older firmware, when the RMC transitions from PROGRAM mode to RUN mode, it assumes all the Program Trigger conditions were previously false. Therefore, when the RMC enters RUN mode, all Program Trigger conditions that evaluate to true will run the corresponding Task actions, because the RMC thinks their states just transitioned from true to false.

Example

Consider a trigger with the condition **MyInput = False**. If MyInput is off when the RMC enters RUN mode, the Task action for that loop time will run.

If several conditions become true at the same time and multiple actions are triggered on the same Task, the last action for that Task will be the one that runs.

If your Program Triggers has any conditions that may be true when the RMC enters RUN mode, but you do not wish those Task actions to run when you enter RUN mode, you can do the following:

1. Use the **FirstScan** bit (see below) in the condition in the last line of the Program Triggers.
2. For each task that you do not wish to start when entering RUN mode, choose **<StopTask>**. Alternatively, if you have a user program that you want to run when the RMC enters RUN Mode, you can enter that user program for the desired Task.

The method above works because the FirstScan condition will become true at the same time as the other items when you enter RUN mode, and since it is the last item in the Program Triggers, it will override the other items.

First Scan Bit

The _FirstScan bit is true during the first loop time after the RMC enters RUN Mode. The _FirstScan bit is false otherwise. This bit can be used in a trigger condition to start programs when the RMC starts up. In order to do so, you must configure the RMC to start in RUN mode.

The _FirstScan bit is a Controller Status Bit.

To use the FirstScan bit:

1. In the last row of the Program Triggers, double-click in the **Condition** cell. The Expression Editor will open below the Condition cell.
2. In the **Tags** list, expand **Controller** and double-click **FirstScan** so that it appears in the box at the top.
3. Click **OK**.
4. In the **Task** columns, choose the user program you wish to run. You must have created a user program first.
5. To apply the changes to the RMC, right-click **Programming** and click **Download Programs**.

Starting a User Program when the RMC Powers Up

To automatically start a user program when the RMC powers up:

1. Create a user program that contains the commands you would like to run when the RMC powers up.
2. In the Program Triggers, create a condition using the _FirstScan bit as described above. The condition should be "_FirstScan".
3. For the condition you entered in the Program Triggers, choose the user program on a task.
4. On the **RUN/PROGRAM** page of the Program Properties dialog, set the RMC to start up in RUN Mode.
5. Update Flash.
6. Cycle power to the controller. You can view the Event Log Monitor to see if the user program did start immediately after the RMC powered up.

See the Example: Closed Loop Motion on Startup for a detailed walk-through example.

Running a User Program on Startup Only

If you wish to have the Program Triggers start a user program only when the RMC starts up, and not each time it enters RUN mode thereafter (such as after you download the Programming node), do the following:

1. Create a variable called **Running**, and set the Default Value to 0.
2. Create the following Program Triggers condition:
   _FirstScan and Running = 0
3. In the first step of the user program, add an Expression (113) command with the following expression:
   Running := 1

When the RMC starts up in RUN mode, it will run the user program. However, it if exits and enters RUN mode after that, the user program will not start because the variable **Running** is not zero.

Disabling the Program Triggers Task

Sometimes when troubleshooting or setting up the RMC, you may wish to be in RUN Mode but have the Program Triggers disabled.

To disable the Program Triggers:

1. In the **Project** pane, expand the desired controller.
2. Right-click **Programming** and click **Properties**.
3. On the **General** page, in the **Program Triggers** section, set the Program Triggers to **Disabled**. Click **OK**.
4. In the **Project** pane, right-click **Programming** and click **Download Programs** to apply the changes to the RMC.

> **Note:**
> The Program Triggers will never run when the RMC is in PROGRAM Mode.

See Also

Programming Overview | User Program Overview

4.11.4. Task Monitor

> **To access this monitor:**
> On the **View** menu, click **Task Monitor**.

Use the Task Monitor to view which user programs are running, and to start and stop User Programs. The Task Monitor shows task information only when RMCTools is online with the controller.

Tasks

The Task Monitor lists the following information:

- Each Task in the RMC
- The user program and step each task is running
- Whether the Program Triggers Task is enabled
- The Default Axis for each task

Starting and Stopping User Programs

To start a User Program:

- Right-click a Task listed in the Task Monitor, choose **Start Task**, and click the desired User Program.

To stop a User Program:

- Right-click a Task that is running and click **Stop Task**.

Customizing the Task Monitor

To hide the Program Triggers Task or Default Axis:

- Right-click anywhere, and choose **Hide Program Triggers Status** or **Hide Default Axis**.

To show the Program Triggers Task or Default Axis:

- Right-click anywhere, and choose **Show Program Triggers Status** or **Show Default Axis**.

Task Tags (Advanced)

The data shown in the Task Monitor can be accessed in the user programs via the following tags:

Task[].CurAxis

Task[].CurProg

Task[].CurStep

Task[].Status

Task[].CurProgStep

Program Monitor (Old)

In RMCTools versions 3.41.0 and earlier, the Task Monitor was located in the Program Monitor, which no longer exists. The Program Monitor also contained the Current Values of the Variables, which are now located on the Monitor tab in the Variable Table Editor.

See Also

User Program Overview | Variables

4.11.5. Variable Table Editor

To access this editor:
In the Project pane, expand **Programming**, then double-click **Variable Table**.

Use the Variable Table Editor to create variables and monitor and edit the current values in real time. For more details on variables, see the Variables topic. For general information on using editors, see the Using Editors topic.

Creating Variables

Use the **Edit** tab to create variables. After editing any variable definitions, you must download the Programming node to apply the changes to the RMC. On the **Programming** menu click **Download Programs to Controller**. You should also Update Flash.

Enter Tag Name

- Click an unused **Tag Name** cell.
- Type a name for the variable and press Enter. The variable can now be referenced from anywhere in RMCTools using this tag. Tag names are limited to 64 characters.

Set the Initial Value for a Variable

- Click a cell in the **Initial** column.
- Type a value and press Enter.
- Download the programming to apply the changes to the RMC.
- The variable's Initial Value field will be changed, and its Current Value will immediately be set to the Initial Value. Variables are also set to the Initial Value when the RMC is restarted.

  When the Variable Table is downloaded to the controller, the Current Value will be set to the Initial Value for all variables for which the Initial Value changed.

Select Retained Option

Retained means that the Current Value of the variable is automatically saved, or retained, in non-volatile memory and will be preserved even when power is cycled to the RMC. This is useful for retaining data such as setpoint positions, machine cycle counters, and recipe data.

Retained variables are saved to non-volatile memory every 100 msec. See the Variables topic for more details on retained variables, such as the maximum number of retained variables.

Retentive variables are available only on the RMC75E, RMC150E, and RMC200. Retentive variables on the RMC75E and RMC150E require firmware 3.30.0 or newer.

To choose the retained option, check the **Retained** box.

Select a Data Type

- Click a cell in the **Type** column.
- Click the ellipsis button  . Select the desired data type. To create an array, increase the **Size** to the desired size of the array. The following data type options exist:
  - REAL
  - DINT
  - DWORD
- To create a BOOL variable, first define the variable as a DWORD. Then, expand the variable to see the bits. Each bit can be assigned a name. For details, see the Boolean Variables topic.

  If the DWORD variable itself has no name, then the bit names are said to be global and are referenced directly like any other tag name. If the DWORD variable has a name, then the individual bits are local to that DWORD variable, and references to that bit must be qualified by the DWORD variable name followed by a period, as in **MyDWORD.myBit**.

Enter the Units

- Click a cell in the **Units** column.
- Type the desired units and press Enter. The units are for user reference only and may be anything.

Add a description

- Click a cell in the **Description** column.
- Type a description and then press Enter.

Monitoring and Editing Current Values

Use the **Monitor** tab view and edit the current value of the variable in real-time.

Changing the Current Value of a Variable

- Make sure you are online with the controller.
- On the Monitor tab, in the **Current** column, enter a number for the desired variable.
- On the toolbar, click the **Download Current Values** button  .

Note:
The Current Value may also change when the Variable Table itself (not the Current Value) is downloaded to the controller. The Current Value will be set to the Initial Value for all variables for which the Initial Value changed.

Choosing the Address Format

The addresses of the variables are displayed in the Register column. To change the address format, right-click any cell in the Register column, choose **Address Formats**, and choose the desired format.

Expanding and Collapsing Arrays

To expand all the arrays: On the toolbar, click the **Expand All**  button.

To collapse all the arrays: On the toolbar, click the **Collapse All**  button.

The expand/collapse all applies to arrays and not to individual DWORD variables.

Columns

Tip:
To hide a column, right-click the column heading and click **Hide Column**. To see it again, right-click any column heading, click **Add Column**, and click the desired column name.

Column Name	Description
First Column	The number of the variable.
Register	Gives the address of the *Current* value of the variable. To change the address format, right-click any cell in the Register column, choose **Address Formats**, and choose the desired format.
Tag Name	This is the descriptive name of the variable. **Note:** When communicating from a PLC, you cannot address a variable using the Tag Name. You must use the register address.
Units	You can enter the units of the variable in this column. The units are only for your reference and have no further importance.
Type	You must specify the correct format for successful operations. For example, a variable that is to be compared with an Actual Position must be of the same format as the Actual Position, which is a REAL.
Retain	Specifies whether a variable will be retained.
Initial	This is the value of the variable before it is modified by a user or a user program. You can edit this value.
Current	The current value of the variable. Available only on the Monitor tab and when RMCTools is online with the controller.
Description	You can enter a description of the variable in this column.

Choosing the Address Format

The addresses of the variables are displayed in the Register column. To change the address format, right-click any cell in the Register column, choose **Address Formats**, and choose the desired format.

Printing

To print the variables, on the Variable Table toolbar, click the **Print** button. Or, on the **File** menu, choose **Print**.

See Also

Variables

4.11.6. User Functions

To access the User Function Editor: In the Project pane, expand **Programming**, and double-click **User Functions**.

User functions are custom functions created or imported by the user. User Functions provide flexibility and efficiency for advanced applications. Most applications do not need user functions. User functions

can be used anywhere expressions are used in the RMC, including the Expression (113) command, link conditions in user programs, and the Program Triggers.

User functions can have any number of parameters, and return a single value as the result. The parameters can be input, output, or input/output types. Therefore, a function can return many values via the output or input/output type parameters.

Example User Functions

See the Example User Functions for examples that you can copy and paste into your project.

Creating a User Function

1. In the Project Pane, expand **Programming** and double-click **User Functions**.
2. In the User Function Editor toolbar, click the **New User Function**  button.
3. In the New User Function dialog, type a **Name** for the function.
4. Choose the desired **Return Data Type**. This is the data type of the resulting value returned by the function.
5. **Add Parameters (optional)**
   Do the following for each parameter you wish to add. Parameters appear as Input, Output, and Input/Output variables in the user function. Parameters can also be added and modified later when editing the user function, as described in Declaring Variables in User Functions.
   1. Click the **Add** button.
   2. Type a **Name** for the function.
   3. Choose the desired **Data Type** of the parameter. To create an array parameter, set the **Size** parameter to a value greater than 1.
   4. In the **Input/Output** box, choose the parameter type:
      - Input
      - Output
      - Input/Output
   5. Click **OK**.

   To adjust the order of the parameters, use the **Up** and **Down**  arrows. To delete parameters, use the **Delete**  button.
6. Click **OK**.
7. The new function will be added to the User Functions list, and will appear in the editor.

Editing a User Function

User Function Description

The initial comment in the user function is the function description, enclosed by green parentheses with an asterisk: (* *). This description will appear when browsing the functions in the Expression Editor.

Enter a description that describes what the function does. It is good practice to clearly describe the parameters and return value. You do not need to describe the parameter data types, as this is automatically displayed in the Expression Editor.

Function Name

The function name and return data type is defined by the keyword **FUNCTION**. To change the function name or data type, change the text following the FUNCTION keyword. The entire function must also end with the **END_FUNCTION** keyword.

For example, **mm2inch** is this function's name, which returns a value of type REAL:

```
FUNCTION mm2inch : REAL
```

Variable Declarations

Functions can have input variables, output variables, input/output variables, and local variables. For details, see Declaring Variables in User Functions.

Editing the Function Body

The function body contains the code that the function executes when it is called. The code in the function body must follow the same syntax as the code used in the Expression (113) command. For details, see the assignment expressions topic. The function body can use all the same operators, functions, user functions, and keywords as used in assignment expressions.

Values can be passed into and out of a function via parameters. All variables of type Input, Output, or Input/Output that are declared in the function will be the parameters of the function. See Declaring Variables in User Functions for more details.

The following limitations apply to the user function body:

- Output variables can only be assigned to. That is, Output variables can only be used on the left-hand side of assignment expressions.
- Input variables cannot be assigned to.

Accessing Controller Tags

The code in a user function can directly reference controller tags, such as variables in the Variable Table, axis tags, etc. However, in order to keep a user function portable between RMCTools projects, it is good practice to limit direct referencing of controller tags and instead pass values in and out via the Input, Output, and Input/Output function parameters.

Function Return Value

Each function must have a return value. This is done by assigning a value to the name of the function. For example, consider a function named **Average3.** The return value could be expressed as:

```
Average3 := (Var1 +  Var2 + Var3 ) / 3;
```

Managing User Functions

All user functions in the project are displayed in the User Function list. To add a user function, click the **New User Function** button. To delete a user function, click the **Delete User Function** button.

User Functions Folders

User functions can be organized into folders in the user function list.

1. To create a new folder, click the **New Folder** button on the toolbar. Type the folder name and press Enter.
2. To add new functions to the folder, select the folder, then click the **New User Function** button. You can also drag existing functions into the folder with the mouse.

Click the **Delete Folder** button to delete a folder and it contents.

Importing User Functions

1. Right-click in the function list pane and choose **Import User Functions**.
2. Browse for the RMCTools User Function Library (.rmcflib) file and click **Open**.
3. The user functions available in the RMCTools User Function Library file will be listed. Check the desired functions in the file to import. The **Select All** and **Clear All** buttons may assist you. Imported functions cannot have the same name as a user function already in the project. These functions are labeled **(duplicate)**. Use the **Rename** button to rename a function in the import list so that it can be imported.
4. After selecting the desired user functions to import, click **OK**.

Exporting User Functions

1. Right-click in the function list pane and choose **Export User Functions**.
2. Choose the functions to export, then click **OK**.

3. Browse to the desired location, enter a filename, then click **Save**.

Calling User Functions

To insert a user function into an expression:

1. Begin typing the name of the function and choose the function from the pop-up list.
   Or, browse for the function on the **Function** tab of the Expression Editor and double-click it.
2. Type an opening parenthesis. A pop-up window will open, listing the required parameters for your convenience.
3. Make sure to enter all required parameters, separated with a comma.
4. Type a closing parenthesis.

Using the Function's Return Value

Typically, when calling a user function, the function's return value is assigned to some register. For example, the return value of the user function inch2mm is assigned to the variable MyPos here:

```
MyPos := inch2mm(_Axis[0].ActPos);
```

A user function can also be called without assigning the function's result to a register. This is useful when the function has output parameters. For example, consider the following user function that converts polar coordinates (radius and angle) to cartesian coordinates (x and y). The function has two input parameters and two output parameters:

```
BOOL Polar2Cart( [in] REAL radius, [in] REAL angle, [out] REAL x, [out] REAL y)
```

This user function can be used in an expression as follows:

```
Polar2Cart(MyRadius, MyAngle, MyX, MyY);
```

Limitations

The following limitations apply to calling user functions:

- Any Output and Input/Output type parameters can only be step-local variables or variables in the Variable Table. They cannot be system tags. If the result of the function must go to a system tag, you can use a variable for the Output parameter, then use a separate expression to assign the variable value to the system tag.

See Also

Functions Overview | Standard Functions | Example User Functions | Declaring Variables in User Functions | Expressions Overview | Assignment Expressions

4.11.7. Discrete I/O Configuration Window

To access this editor:
In the Project pane, expand **Programming**. Double-click **Discrete I/O**.

Use this window to configure Discrete I/O on the RMC. This topic describes the columns in the Discrete I/O Configuration window. For instructions on how to configure the Discrete I/O, see the Configuring Discrete IO topic.

Columns

Column Name	Description

First Column	Displays the address of the I/O point. This address can be used to reference the I/O point in User Programs or the Program Triggers, but the preferred method is to use the tag name.
Module	List the module of the DI/O point. RMC75 expansion modules are numbered 1 to 4, starting closest to the RMC75 base unit. RMC150 slots are numbered from the left starting at 0. RMC200 slots are numbered from the left starting at 0 for the power supply, 1 for the CPU, 2 for the first I/O module, etc.
I/O Point	Lists which DI/O point on the module
Tag Name	The tag name assigned to the I/O point by the user. This cell is editable. Use the tag name to reference the I/O point in User Programs or the Program Triggers.
Type	This cell specifies whether the I/O point is an input or output. If the I/O point is configurable as either an input or output, you can change by double-clicking the cell and choosing or **Input** or **Output**.
Program State	Valid only for outputs. Defines the state of the output when the RMC enters PROGRAM mode. See the Configuring Discrete IO topic for details.
Fault State	Valid only for outputs. Defines the state of the output in the following conditions: - The RMC receives a Fault Controller (8) command - The RMC200 is in the Disabled state (RMC200 Only). - The RMC200 powers up, regardless of the startup mode (RMC200 Only). See the Configuring Discrete IO topic for details.
Assigned To	Lists the functionality to which a discrete I/O point has been assigned. This includes: • Axis Fault Input • Axis Enable Output • Axis Positive Limit Input or Negative Limit Input • RUN/PROGRAM or RUN/Disabled Input
Description	Enter a description of the I/O point in this column.

Applying Changes to the RMC

To apply the changes made in the Discrete I/O Configuration window, you must download the Programming to the RMC. To do this, on the **Programming** menu, click **Download Programs**.

See Also

Discrete I/O | Configuring Discrete IO

4.11.8. Discrete I/O Monitor

To access this dialog:
On the View menu, click **Discrete I/O Monitor.**
Press Alt+3 to show or hide the Discrete I/O Monitor.

Use this monitor to view the current state of the discrete I/O, toggle outputs, and to force inputs or outputs on or off. To configure the discrete I/O, use the Discrete I/O Settings Editor. For details on using the discrete I/O, see the Discrete I/O topic.

Viewing the Discrete I/O

The I/O Monitor displays the states of all the general discrete inputs and outputs of the RMC.

Icon	Description
	Output - Off
	Output - On
	Output - Forced Off
	Output - Forced On
	Input - Off
	Input - On
	Input - Forced Off
	Input - Forced On

Discrete I/O Tags and Addresses

In the Discrete I/O Monitor, you can choose to view the address, tag name, or neither, of each input and output.

- In the Discrete I/O Monitor, right-click anywhere, point to **Labels**, and click **None**, **Address**, or **Tag**. If you choose **Tag**, and a tag name has not been assigned to the I/O point, the address will be displayed instead.

Toggling Outputs

To toggle outputs:

- In the Discrete I/O Monitor, right-click the output icon you want to toggle, and click **Toggle Output**. The output state will toggle.

Forcing the Discrete I/O On or Off

Inputs and Outputs can be forced on or off. The inputs and outputs will always be in the forced state until the force is removed.

To force an input or output:

- In the Discrete I/O Monitor, right-click the input or output icon you want to force, and click **Force On** or **Force Off**. The input or output icon will be highlighted with a yellow background to indicate is forced.

To remove a force:

- In the Discrete I/O Monitor, right-click the input or output you want to remove the force from, and click **Remove** F**orce**.

Or,

- In the Discrete I/O Monitor, right-click anywhere and click **Remove All Forces**.

See Also

Discrete I/O | Configuring Discrete IO | Using Discrete I/O

4.12. Curve Tool

4.12.1. Curve Tool

To access the Curve Tool:
In the Project Pane, double-click **Curves**.

Use the Curve Tool for creating, editing, and monitoring curves. Curves, also called cams or splines, allow the user to create custom motion and camming profiles.

How to Create and Edit a Curve

The **Curves in Project** window displays the curves in the project. The **Curves In Controller** window displays the curves in the controller. Curves are created and edited in the project. Curves in the controller can only be viewed.

Creating a New Curve

1. In the toolbar, click the **Create New Curve** button. The new curve will appear in the **Curves in Project** window. Later, after setting the curve properties and adding points as described below, you will download the curve to the controller, and it will appear in the **Curves in Controller** window.
2. In the **Properties** pane, on the **Curve** tab, in the **Name** cell, enter a name for the new curve. You may also enter a **Description**.

Adding Curve Points

1. In the spreadsheet located below the curve graph view, in the right-most column, enter the X value and Y value for the new point, then press Enter. Curve data can be copied and pasted from spreadsheets programs as described in **Copying and Pasting Points** below.
2. Continue adding points by entering the X values and Y values in the right-most column. Even if the point is not going to be the last point in the curve, you must enter it in the right-most column. After entering the X and y values, the new point column will automatically be placed in the correct location in the spreadsheet.

Editing Curve Points or Properties

1. To edit existing points, simply edit the desired X values and Y values. Clicking a point in the graph view will highlight the point in the spreadsheet view, making it easy to find.
2. Set the properties of the curve on the Curves tab in the properties pane. See the Curve Properties for details.

Deleting Points

- In the spreadsheet, click the column header and press Delete.
- In the graph view, click a point and press Delete.

Copying and Pasting Points

Points can be copied and pasted, including to and from spreadsheet programs such as Microsoft Excel®. The pasted data will be automatically transposed if necessary. The pasted values must not duplicate any X values.

For best results, paste values into a new curve or delete all the existing points before pasting new points. All cells can be copied and pasted, but typically, for curves with standard point types, you only need to copy and paste the X and Y values.

How to View a Curve

Viewing a Curve

Click a curve in the project or controller to view it. Curves in the controller are visible only when RMCTools is online with the controller. Curves in the controller cannot be changed. However, as described in **How to Manage Curves** below, they can be uploaded or copied to the project to be edited.

Viewing Multiple Curves

Click the pin next to the curve in the list to view that curve. All pinned curves will be displayed, together with the currently selected curve. Any curve that is pinned will be displayed in black in the graph view. The currently selected curve is always displayed in red.

Using the Hairline Cursor

In the graph, click and drag the hairline cursor. Or, right-click in the graph and choose **Move Cursor Here**. The X, Y, velocity, and acceleration values at the cursor location will be displayed on the **Cursor** tab in the **Properties** pane.

Viewing Curve Velocity and Acceleration with Limits

1. On the toolbar, click the **Show/Hide Velocity** and **Show/Hide Acceleration** buttons. This will display the curve's velocity and acceleration in the graph view.
2. In the **Properties** pane, on the **Curve** tab, in the **Velocity Limits** and **Accel Limits** cells, enter the values for the limits. Dashed lines will appear on the graph view to indicate the limits.
3. If the velocity or acceleration exceeds the limit, the toolbar will display the error.

Zooming and Scrolling

To zoom, use the following methods:

- The buttons on the Curve Tool toolbar:
- Ctrl+Mouse Wheel. The zooming will be centered on the mouse.
- Ctrl+Plus Sign and Ctrl+Minus Sign

To scroll, use the following methods:

- Use the horizontal scroll bar.
- Shift+Mouse Wheel
- Mouse Wheel Tilt
- Horizontal Mouse Wheel

Show/Hide Points

To hide the points of a curve, on the toolbar, click the **Show/Hide Points** button. The points of a curve will also automatically be hidden if the number of visible points exceeds 5000.

How to Manage Curves

Understanding the Sync Column

The Sync column in the **Curves In Project** and **Curves in Controller** windows is used by permanent curves to indicate the differences between the curves in the project and in the controller as follows:

The curve exists both in the project and the controller, but the two curves are different.

The curve exists only in the project. Download to copy the curve into the controller. Upload to delete the curve from the project.

The curve exists only in the controller. Upload to copy the curve into the project. Download to delete the curve from the controller.

Temporary curves do not use the Sync column. For details on temporary curves, see the Managing Curves in the Curve Tool topic.

Downloading Curves to the Controller

Click the **Download Curves to Controller** button to download the entire set of curves in the project into the controller, replacing all permanent type curves in the controller. It will leave temporary curves unaffected, except that temporary curves whose curve IDs matched a downloaded curve will be overwritten.

Uploading Curves from the Controller

Click the **Upload Curves from Controller** button to upload the entire set of permanent type curves from the controller into the project, replacing all curves in the project. Temporary curves are not uploaded into the project.

Saving Curves to Flash

In the Project Pane, right-click the controller and choose **Update Flash**. Curves downloaded from the Curve Tool or created using the Curve Add (82) command with the Permanent option will be saved to Flash memory.

Copying and Pasting Curves

Entire curves in the project can be copied and pasted. This is typically used to copy and paste between controllers or projects. Curves can also be exported and imported as described below.

Exporting and Importing Curves

Curves are saved within the project file, but can also be exported to a separate RMCCurves file that can be imported into another RMCTools project:

1. In the Curves in Project window, right-click and choose **Export Curves**.
2. Check the curves you wish to export, then click **OK**.
3. Browse to a folder, enter the file name, and click **Save**. The curves will be saved in that file with extension **.rmccrvs**.

To import the curve file:

1. In the Curves in Project window, right-click and choose **Import Curves**.
2. Browse to the file and click **Open**.
3. Check the curves you wish to import, then click **OK**.

Note: If you wish to export curves to a different format, such as CSV, simply copy the curve data from the spreadsheet in the Curve Tool and paste into a program such as Microsoft Excel. Excel has options to save in various formats.

Deleting a Curve in the Project

In the Curves in Project window, select a curve and press Delete.

Deleting a Curve in the Controller

In the Curves in Controller window, select a curve and press Delete.

Note: Curves in the controller can also be deleted programmatically via commands. See Curve Delete (83) for details.

Copying a Temporary Curve into the project

A temporary curve can be copied into the project, making the copy a permanent type curve.

To copy a temporary curve into the project, select a temporary curve and click the **Copy Curve to Project** button on the toolbar.

See Also

Curves Overview | Managing Curves in the Curve Tool | Curve Properties | Curve Start (86)

4.12.2. Curve Properties

The **Properties** pane of the Curve Tool lists the properties of the selected curve. Each curve created in the Curve Tool may have all or some of the following properties:

Property	Description
Curve ID	This number uniquely defines the curve. The ID is used in the Curve Start commands to specify the curve to start.
Name	A name for the command. This is mainly for the user's reference.
Description	A description of the command, for the user's reference.
Number of Points	Specifies the number of points in the curve. This number cannot be edited directly, but reflects how many points the curve contains.
Curve Type	• **Standard** - A curve that can be used for any purpose other than Valve Linearization. • **Valve Linearization** - A curve that may be used for Valve Linearization. This type of curve has special requirements.
Interpolation	Specifies how the lines should be drawn through the points. As described in detail Curve Interpolation Methods and Options, the options are: • **Cubic** - A smooth line is drawn through the points. • **Linear** - Straight lines are drawn from point to point. • **Constant** - A horizontal line is drawn from each point.
End Point Behavior	Defines the conditions of the first and last points of cubic-interpolated curves. As described in detail in Curve Interpolation Methods and Options, the options are: • **Zero-Velocity** - The endpoints have zero velocity. • **Natural-Velocity** - The endpoints velocity is defined by the natural slope of the interpolated curve. • **Cyclic** - Matches the velocity of the first and last point so that the curve can be run cyclically.
Overshoot Protection	For cubic-interpolated curves, eliminates overshoot of any local maximums or local minimums in the curve points. For details, see Curve Interpolation Methods and Options.
Auto Constant Velocity Behavior	For cubic-interpolated curves, this option will automatically insert a linear segment in the curve if three or more data points are in a straight line. For details, see Curve Interpolation Methods and Options.
Life Cycle	The Life Cycle defines whether the curve will automatically be deleted, and whether it can be saved to Flash.

	The possible life cycles are: • **Permanent** All curves created in the Curve Tool are Permanent. These can be saved to Flash. These curves will not be automatically deleted. • **Standard** These curves cannot be saved to Flash, must be deleted manually, or will disappear when power is removed from the controller. • **Start-Once** This curve will automatically be deleted after it has been started once, or will disappear when power is removed from the controller. • **Complete-Once** This curve will automatically be deleted after it has been completed once, or will disappear when power is removed from the controller. The Standard, Start-Once, and Complete-Once life cycles can be specified only when creating curves using the Curve Add command. For more details, see the Curve Add (82) command.
Velocity Limit	Use this to specify a velocity limit for the curve. If the curve velocity exceeds this limit, the Curve Tool will give an error and will not allow the curve to be downloaded. To view the velocity limit on the graph, click the **Show Velocity Limit** button on the toolbar.
Acceleration Limit	Use this to specify an acceleration limit for the curve. If the curve acceleration exceeds this limit, the Curve Tool will give an error and will not allow the curve to be downloaded. To view the acceleration limit on the graph, click the **Show Acceleration Limit** button on the toolbar.

See Also

Curve Tool Overview

4.13. Address Maps

4.13.1. Address Maps Editor

To access this editor:
Expand the desired controller in the Project pane and double-click **Address Maps**.

Use this editor to view and edit the Address Maps. The Address Maps includes the following maps:

- Indirect Data Map
- Modbus Address Map
- FINS Address Map
- DF1 Address Map

- PROFINET Data Records Address Map
- IEC Address Map

For more details, see the Address Maps topic.

See Also

Address Maps | Indirect Data Map

4.13.2. Indirect Data Map Editor

To access this editor:
Expand the desired controller in the Project pane, double-click **Address Maps**, then click **Indirect Data Map Editor**.

Use this editor to view and edit the Indirect Data Map. The Indirect Data Map is intended primarily for certain communication types. See the Indirect Data Map topic for instructions on using the Indirect Data Map.

For general information on using editors, see the Using Editors topic.

Editing the Data Map

Adding registers to the Data Map

- In the desired row, in the **Map To** column, click the ellipsis button  . The Address Selection Tool will open.
- In the Address Selection Tool, in the **Available Registers** box, browse to the register you wish to add.
- Double-click the register (or press Spacebar) to add it to the **Registers to Add** box.
- Continue browsing and adding registers to the **Available Registers** box.
- Click **OK** when finished.

  Or:
- In the desired row, in the **Map To** column, start typing the tag name or address of the item, such as _Axis[0].TarPos, or %MD12.0. As you type, you can choose the desired item from the drop-down list.

Duplicating Registers for Multiple Axes

To add identical registers for multiple axes:

- In the Address Selection Tool, add the desired registers for a single axis only, such that the registers appear in the **Registers to Add** box.
- Click the **Duplicate for Axes** button.
- Select the axes for which to repeat the same registers. You can use the **Select All** button to select all the axes.
- Click **OK**. The **Registers to Add** box will list all the registers you have selected to add.
- Click **OK** again to close the Address Selection Tool and add the registers to the Indirect Data Map.

Write to a mapped register

For mapped registers that are writable, you can write a value to the register via the Indirect Data Map:

- Enter the desired value in the **Current** column.

- In the Indirect Data Map editor toolbar, click the **Download** button  to apply the changes to the RMC.

Columns

Tip:
To hide a column, right-click the column heading and click **Hide Column**. To view a hidden column, right-click any column heading, click **Add Column**, and click the desired column name.

Column Name	Description
First Column	The first column is the address of the indirect data.
Reg #	This is the address of the Indirect Data register. This is the address you should use when reading from or writing to the Indirect Data. The format of the addresses in this column can be selected from several common formats. See **Choosing the Address Format** below.
Map To	This is the address that this Indirect Data item references.
Description	The description of the register in the **Map To** column, including the units.
Current	This is the current value of the register in the **Map To** column. If it is not a read-only register, you can edit this value.

Choosing the Address Format

The addresses of the registers in the Indirect Data Map are displayed in the Reg # column. To change the address format, right-click any cell in the Reg # column, choose **Address Format**, and choose the desired format.

See Also

Indirect Data Map

4.14. Shortcut Sets

4.14.1. Shortcut Commands

Tip:
If you are looking for keyboard shortcuts, click here: Keyboard Shortcuts

Shortcut Commands are used to quickly send a command to an axis by pressing the 0-9 shortcut buttons on the RMCTools toolbar, or by pressing Ctrl+0 through Ctrl+9 on the keyboard.

You can assign one or more commands to each shortcut key. Shortcut commands are useful when setting up and tuning an axis, which usually requires moving repeatedly between two or more positions.

Assigning Commands to a Key Combination

There are two ways to assign commands to a shortcut command key combination:

From the Command Tool

a. In the **Cmd** box, choose a command and enter the command parameters if there are any.

b. To store the command, click the Store Shortcut Command button  , and click the key combination you wish to assign the command to.

From the Shortcut Command Set Editor

a. In the Project pane, expand **Shortcut Sets**, and double-click the shortcut command set you wish to change.

b. In the Shortcut Command Set Editor, add or remove commands. The number to the left of each item indicates which key combination it corresponds to.
See the **Editing Shortcut Command** section below for details.

Sending Shortcut Commands

There are two methods of sending shortcut commands.

- Press CTRL+*num*, where *num* is a digit from 0 to 9.
- Click one of the shortcut command buttons on the toolbar. These buttons correspond to pressing CTRL-*num*:

A black number indicates that the shortcut command has one or more commands assigned to it. A gray number indicates that the shortcut command does not have any commands assigned to it.

Sending a shortcut command will send the command(s) assigned to that key combination of the *active* shortcut command set. See **Shortcut Command Sets** below.

Shortcut commands apply to the currently selected controller in the Project pane.

Editing Shortcut Commands

To edit Shortcut Sets, in the Project pane, expand **Shortcut Sets**, and double-click the shortcut command set you wish to change.

In the Shortcut Command Editor, the numbers to the left of each item correspond to the shortcut key combinations Ctrl+0 through Ctrl+9. For each key combination, you can define which commands will be issued. Each key combination can have only one command for each axis.

To add a command to a key combination:

1. Click **Add Command**.
2. In the **Axis** box, choose which axis the command should be sent to.
3. In the **Cmd** box, choose the command.
4. If the command has any parameters, fill them out.

To remove a command from a key combination:

1. Highlight the entire command by clicking the word **Command**, or the whitespace above the **Command** box.
2. Press Delete.

Uploading and Downloading

Any edits in the Shortcut Sets will immediately be reflected on shortcut commands sent to the RMC, without requiring changes to be downloaded. Edits will also be saved in the RMCTools project file when a save is initiated. Downloading Shortcuts Sets will allow them to be saved to Flash, so that they can be uploaded later.

To save the Shortcut Sets in the RMC, in the Shortcut Commands editor, click the Download button  , then update Flash.

To upload the controller values from the RMC to the project, in the project tree, right-click **Shortcut Sets** and choose **Upload Shortcut Sets from the Controller**.

Shortcut Command Sets

Any number of shortcut command *sets* be created. For each set, any commands can be assigned to each key combination. Only one shortcut command set can be active at a time.

Activating a Shortcut Command Set

- On the main toolbar, in the Active Shortcut Command Set box, choose a shortcut command set. The shortcut command set will become active immediately:

Creating a New Shortcut Command Set

- In the Project pane, right-click **Shortcut Sets**, and click **New Shortcut Set**. A new shortcut command set editor will open.

Renaming a Shortcut Command Set

- In the Project pane, expand **Shortcut Sets**, right-click the shortcut command set and click **Rename**, or select the shortcut command set and press F2.

Exporting and Importing Shortcut Command Sets

Shortcut command sets can be exported and imported in RMCTools. If you have a favorite command set, you can export it to a file, and then import it to each project you create. Shortcuts command sets can also be copied and pasted.

To export:

1. In the Project pane, expand **Shortcut Sets**.
2. Right-click a shortcut command set and click **Export Shortcut Cmd Set**.
3. If you wish to export all the command sets, right-click **Shortcut Sets** and click **Export Shortcut Cmd Set**.

To import:

1. In the Project pane, right-click **Shortcut Sets** and click **Import Shortcut Cmd Set**.
2. Browse to a shortcut command set file that you previously exported and click **Open**.

See Also

Command Tool | Keyboard Shortcuts

4.15. Event Log

4.15.1. Event Log Monitor

To access this dialog:
In the Project Pane, double-click **Event Log**.

The Event Log Monitor is one of your most important troubleshooting tools. The Event Log Monitor displays all events that have occurred in the controller, such as issued commands, changed parameters and errors. The Event Log Monitor is an important aid in troubleshooting. It can be used to:

- Determine whether a command was successfully issued. The entire command, with parameters, is displayed.
- Find out which, if any, error occurred.

- Determine where a command was issued from, for example, from a PLC, from a User Program, from the Command Tool.

Each entry in the Event Log Monitor has the following components:

Component	Description
#	The number is solely for indicating the order of the events.
Time	The time the event occurred is given in the following format: *day*d *hour*:*minute*:*second*:*millisecond* The time is measured from when the RMC was last powered up, and is not a precise time. For troubleshooting, this same time can be displayed in plots and is useful in matching events with the plot data. In the Plot Manager, on the **View** menu, choose **Show Absolute Time**. The Plot Detail window will include the absolute time of the cursor in the currently displayed plot.
Event	Describes the type of event.
Details	Provides additional details on the event, such as the previous and changed values, command source, etc

Number of Entries

The maximum number of Event Log entries is listed below. The Event Log itself is in the RMC. The Event Log Monitor simply displays the RMC's Event Log. On the RMC75E, RMC150E and RMC200, the Event Log will hold up to 8192 entries. When the maximum number of entries is reached, the oldest entries will disappear.

On the RMC75P and RMC75S, the Event Log is limited to the lesser of 512 entries or 2048 total words. When the maximum number of entries is reached, the oldest entries will disappear. Notice that if the Event Log Monitor is open and online with the RMC, it will store up to 8192 entries, even though the RMC75P and RMC75S store less.

Clearing the Event Log

To clear the Event Log, click the **Clear Event Log** button on the toolbar in the Event Log window. This will remove all the entries from the Event Log.

Saving the Event Log

To save the Event Log, click the **Save Event Log** button on the toolbar in the Event Log window. A Save As dialog will open. Drowse to a folder, enter a filename and click **Save**. The sEvent Log will be saved with the file extension "rmcelog".

Pausing the Event Log Monitor

To Pause the Event Log Monitor, click the **Pause Event Log** button on the toolbar in the Event Log window. The Event Log Monitor will stop uploading new Event Log items, although the Event Log in the RMC is still logging. When the Event Log Monitor is paused, it will also try to backfill any missing items as much as possible. Because the Event Log is finite, the Event Log Monitor may not be able to backfill all its missing data.

Pausing the Event Log

To pause the Event Log itself in the RMC, and not just the Event Log *Monitor*, use the Pause/Resume Log (95) command. This is useful for advanced troubleshooting.

Resuming the Event Log and Event Log Monitor

To Resume the Event Log in the RMC, and the Event Log Monitor in RMCTools, click the **Resume Event Log** button on the toolbar in the Event Log window. The Event Log will start running, and the Event Log Monitor will resume uploading new Event Log items.

Opening a Saved Event Log File

A saved Event Log file can be opened in the following ways:

- In the Project Pane, right-click **Event Log** and choose **Open Event Log File**. Browse to the file and click **Open**.
- On the **File** menu, choose **Open**. In the Open dialog, in the **Files of Type** box, choose **RMCTools Event Log Files (*.rmcelog)**. Browse to the file and click **Open**.

The file you select will be opened in a new Event Log window. The new Event Log window will not record new data from any RMC.

Filtering the Event Log

Many of the events in the Event Log can be filtered out. This is useful when looking for a particular event while many other events are occurring, making it difficult to find that particular event. Or, certain events can be added, such as communication transactions.

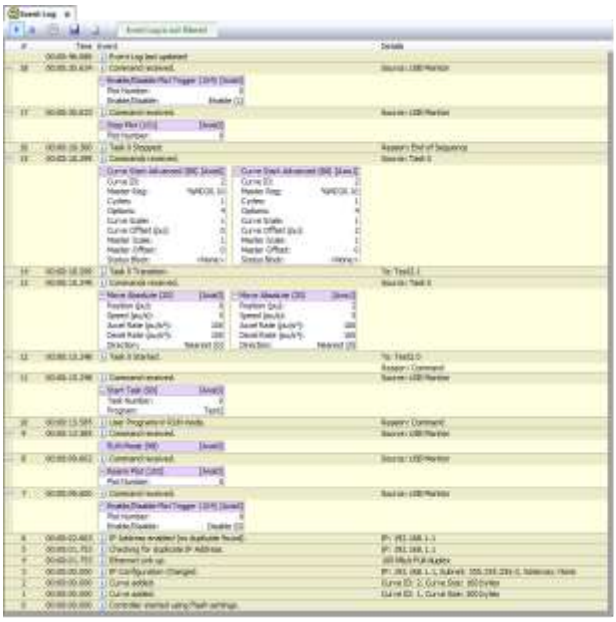
To change the filter configuration, on the Event Log toolbar, click the **Event Log Properties** button. The Event Log Properties dialog will open. In the tree on the left, select an object. To exclude an event for that object, clear the check box. When finished, click **OK**.

User-Defined Event Log Entries

For troubleshooting of advanced user programs, you can create your own custom Event Log entries. Use the LOG_EVENT() function in an Expression (113) command in a user program. This function will cause values of its three function parameters to be reported in the Event Log.

See the Functions topic for details.

Example Event Log



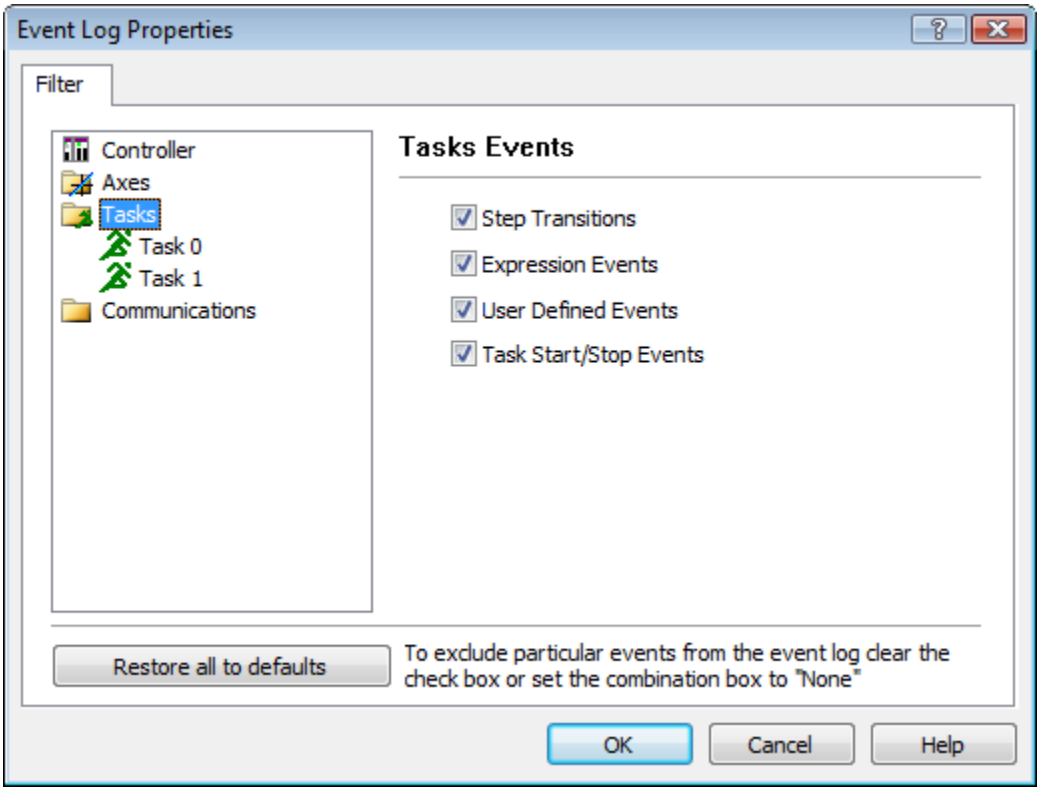
See Also

Troubleshooting Overview | Event Log Properties

4.15.2. Event Log Filtering

The Event Log is a powerful troubleshooting tool. It records most events that occur in the RMC. The Event Log filter specifies which events should appear in the Event Log. The default filter configuration includes many events. However, some useful events, such as communication transactions are not included because they could easily flood the Event Log. To view these events, you must set the filter to include them. Other events that appear by default may sometimes flood the log - such as user program step transitions - and make it difficult to see other events. You can set the filter to exclude these events.

To change the filter configuration, on the Event Log toolbar, click the **Event Log Properties** button. The Event Log Properties dialog will open:



For each item on the left, a list of events will be displayed on the right. All events that are checked will be included in the Event Log. To exclude an event, choose an item on the left, and clear the check box of the event you wish to exclude. When you are finished, click **OK** and the changes will be automatically applied to the controller if you are online.

In general, errors cannot be filtered out.

Filtered Items

Item	Description
Controller	
Register Write Errors	Logs all errors that occur when registers are written to .
Flash Update Events	Logs when the controller Flash update is started and when it successfully completes. This category does not include logging of errors during Flash updates.
RUN/PROGRAM Mode Changes	Logs all changes to the RUN/PROGRAM mode.
Axes	

Commands Received	Logs all commands received by the controller, and logs transitions.
Parameters Changed	Logs all changes to the Axis Parameters.
Pressure/Force Mode Changes	Logs all changes to the pressure/force mode.
Home/Latch Events	Logs Home/Latch input triggers.
Advanced Feedback and Simulator Events	Logs Feedback Model and Simulator Model events.
Debug Target Info	Logs target generator solutions.
Tasks	
Step Transitions	Logs all transitions between steps.
Expression Events	Logs an event for every assignment within an expression.
User Defined Events	Logs all user defined events generated using the LOG_EVENT() function.
Task Start/Stop Events	Logs all Task start and stop events. These can be initiated by a command or the Program Triggers.
Communications - USB Monitor / RS-232 Monitor	
Serial Protocol Errors	Logs all serial errors (framing, parity, etc.) and protocol errors.
All Transactions	Logs all transactions received on this port.
Communications - 10/100 Ethernet	
Ethernet Link Up/Down	Logs when the Ethernet Link goes up or down.
IP Address Events	Logs all events regarding the IP Address configuration, including BOOTP and DHCP events.
TCP/IP Events	Logs the opening and closing of all TCP/IP connections, and errors that occur while opening a connection or forcing a connection to close.
Application Protocol Errors	Logs all application layer protocol errors.
Application Connection Events	Logs events related to the opening and closing of application layer connections.
All Application Transactions	Logs all application layer transactions.
Include RMCTools Transactions	If **All Application Transactions** is checked, this option will include the RMCTools Ethernet traffic. Unchecking this option will exclude RMCTools traffic if RMCTools and firmware versions are 3.32.0 or newer.
Ethernet I/O Logging	Choose **All** to see every change in incoming I/O data. Choose **Requests** to see the incoming I/O data when it is applied. Choose **None** to see no incoming I/O data.
Communications - RS-232/485	
Serial Protocol Errors	Logs all serial errors (framing, parity, etc.) and protocol errors.
All Transactions	Logs all transactions received on this port.
Communications - PROFIBUS/DP	
Command Channel Logging	For the Basic/Enhanced modes, logs the specified actions from the command area in the RMC PROFIBUS command block. **None:** Do not log any actions from the command area. **Requests:** Log only command requests.

	All: Log every change in the command area data.
Data Channel 0 Logging	For the Basic/Enhanced modes, logs the specified actions from Data Channel 0. **None:** Do not log any actions from Data Channel 0. **Requests:** Log only read or write requests from Data Channel 0. **All**: Log every change in the Data Channel 0 data.
Data Channel 1 Logging	For the Basic/Enhanced modes, logs the specified actions from Data Channel 1 (available only in Enhanced RMC PROFIBUS modes). **None:** Do not log any actions from Data Channel 1. **Requests:** Log only read or write requests from Data Channel 1. **All**: Log every change in the Data Channel 1 data.
I/O Mode Data Logging	For the I/O modes, logs the PROFIBUS Output Data (coming from the PLC to the RMC) each time there is a change in the data.
Configuration Information	Logs PROFIBUS initialization state changes and configuration and parameterization information received from the PROFIBUS master.

See Also

Event Log Monitor | Event Log Properties

4.16. General Tools

4.16.1. Uploading and Downloading

Downloading transfers data from RMCTools to the RMC controller.

Uploading transfers data from the RMC controller to RMCTools.

Note:
RMCTools must be online in order to upload or download.

Note:
When downloading, the new data will write over the existing RMC data. However, it will not be saved in Flash until you update Flash.

Instructions

In general, items in the Project pane can be uploaded or downloaded by right-clicking and choosing **Upload from Controller** or **Download to Controller**. In many editor windows, the window contents can be uploaded or downloaded by clicking the download button or the upload button. Uploading and downloading can also be accessed from the Controller, Programming and Editor menus.

Item	Location of Download/Upload
All project contents	In the project pane, right-click the controller.
Axis Parameters	In Axis Tools window toolbar.
Programming (includes Variables (not Current Values), Discrete I/O Configuration, Program Triggers, User Functions, User Programs)	In the project pane, right-click the **Programming** folder.
Current Value of Variables	In the Variable Table Editor, on the **Monitor** tab, on the toolbar.
Curve Tool	In the Curve Tool toolbar.
Indirect Data Map	In the Address Map Editor toolbar.
Plot Templates	In the Plot Template Editor toolbar (download only). To upload, right-click **Plots** in project pane.
Event Log filter Settings	In the project pane, right-click the **Event Log** folder. If online, the settings are downloaded automatically when clicking OK in the Event Log Properties dialog.

4.16.2. RMCTools File Types

An RMCTools project is contained in a single project file. In addition, RMCTools can save and open other types of files as listed below.

Description	File Extension
Main RMCTools Project Contains all the RMC settings and programming.	.rmcproj
Plots - compressed Compressed file containing one or more plots. Typically, this file size is 10% of the .rmcplots file size. The compressed (.rmcplotx) format is supported in RMCTools 4.09.0 and newer. The uncompressed (.rmcplots) format must be used if plots need to be opened in earlier versions of RMCTools. See Saving and Exporting Plots.	.rmcplotx
Plots Uncompressed file containing one or more plots. This uncompressed XML file is easily readable in a text editor.	.rmcplots
Event Log Contains the currently captured Event Log information. This file is very useful for troubleshooting.	.rmcelog
User Programs Import/Export	.rmcprog

Used for transferring a user program between projects. This file includes the variables used in the user program.	
User Function Library Used for transferring user functions between projects. This file contains one or more user functions.	.rmcflib
Shortcut command Import/Export Used for transferring shortcut command sets between projects. This file contains one shortcut command set.	.rmcsset
Curves Import/Export Used for transferring curves between projects. This file contains one or more curves.	.rmccrvs

See Also

Help Overview

4.16.3. Error Icon and Error Bubble

The error icon is located at the bottom of the RMCTools window, in the Status Bar. The error icon indicates whether an error has occurred, and will automatically pop up a bubble when an error occurs. The pop-up can be disabled.

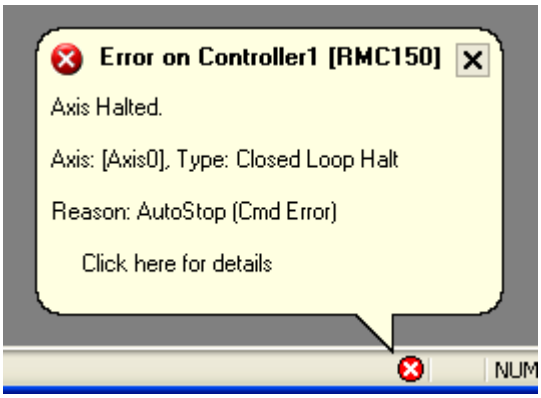
The error icon takes on the following forms:

	Description
	This indicates that either no error has occurred, or the Error History has been cleared.
	This indicates that an error has occurred and is listed in the Error History.
	This indicates that the error pop-up is disabled.

Using the Error Icon and Error Bubble

Error Bubble

When RMCTools is online with an RMC and an error occurs in the RMC, an error notification bubble will temporarily pop up, similar to the image below. The bubble will disappear in 5 seconds if no more errors occur.



To view more details on the error, click the bubble, and the Event Log will open to that error.

Disabling the Error Bubble Pop-Up

The error bubble can be disabled entirely, or can be set to pop up only for errors resulting from actions in the RMCTools software. This is useful when you don't want to see errors or warnings caused by a program running in the RMC, but you still want to see error resulting from actions in RMCTools, such as issuing a command with invalid parameters.

To disable the Error Bubble Pop-up:

- Right-click the Error Icon and choose **Disable Error Popup**. The Error Bubble will no longer pop up.
  Or, right-click the Error Icon and choose **Pop up for Local Errors Only**. The Error Bubble will only pop up for errors caused by action in RMCTools.

Viewing the Error History

The Error History displays up to the last ten errors that were displayed in the error bubble. To view the Error History, click the Error Icon.

Clearing the Error History

To clear the Error History, right-click the Error Icon and choose **Clear Error History**. The icon will no longer contain an x:  .

Note:
If the Event Log is paused due to the Pause/Resume Log (95) command, the Error Icon and Error Bubble will not operate normally. Using the Pause button in the Event Log Monitor will not affect the Error Icon and Error Bubble.

See Also

Event Log Monitor | Status Bar

4.16.4. Address Selection Tool (Single Register)

To access this dialog:
This dialog can be accessed from Command Parameters that require a register, or the Delay Link Type in User Programs. To open this dialog, click the ellipsis button  in the box that requires an address entry.

Use this dialog to select a register to enter in a Command Parameter in a User Program or the Command Tool, or in the Delay Link Type in User Programs.

To choose a register:

a. Browse to the register you need and select it.
b. Click **OK**.

You can also select a register by typing the address in the Address box.

See Also

Register Map

4.16.5. Output Window

To access this window:
On the View menu, click **Output**, or press Alt+1.

The Output window is a dockable window in RMCTools. The output window has the following functions:

- Displays the results of downloading all settings to the RMC.
- Displays the results of uploading all from the controller.

If the upload or download is successful, the Output window will automatically close after 0.5 seconds. In the RMCTools Options dialog, in the **Environment: General** section, this option can be disabled.

See Also

Upload and Download

4.16.6. Verify Results Window

To access this window:
On the View menu, click **Verify Results**, or press Alt+2.

The **Verify Results** window shows the results of the **Verify Programs** operation. When RMCTools verifies the User Programs, it checks the User Programs and Program Triggers for errors. The results are displayed in the Verify Results window.

Use the Verify Results window for the following:

- Double-click the reported error to go to it in the User Programs or Program Triggers. Or, press F4 to open the location of the next error and press Shift + F4 to open the location of the previous error.
- Find out how much of the User Programs capacity is used. If the allotted memory space is exceeded, an error will be reported. For help on this error, see the Program Capacity and Time Usage topic.
- View the execution time information of the user programs. If the time exceeds the allotted time, an error will be reported. For help on this error, see the Program Capacity and Time Usage topic. The amount of timing information displayed can be controlled in the RMCTools Options dialog, in the Programming: General section.

If the verify is successful, the Verify Results window will automatically close after 0.5 seconds. In the RMCTools Options dialog, in the **Programming: General** section, this option can be disabled.

See Also

Verifying User Programs | Program Capacity and Time Usage

4.16.7. Actuator View

To access this pane:
On the **View** menu, click **Actuator View**.

The Actuator View window provides a graphical indication of the position of all position control axes in the RMC. Each position axis is shown as a hydraulic cylinder. The endpoints are set to the Positive and Negative Travel Limits axis parameters. The Actuator View window is a dockable pane and can be docked and moved as described in the **Dockable Panes** section of Using the RMCTools Interface.

See Also

Help Overview

4.16.8. RMCTools Options Dialog

To access this dialog:
On the **Tools** menu, choose **Options**.

The Options dialog contains various RMCTools settings. These settings apply to RMCTools, and are not saved to the project file or the RMC controller. The Options dialog contains the following sections:

Environment Page

Startup

Choose what happens when RMCTools starts up.

Address Format

This applies to windows with a **Reg #** column that lists addresses of the displayed registers such as in the Axis Tools, Indirect Data Map Editor, and Variable Table Editor. This option only affects how the register addresses are displayed.

If you are using a PLC or HMI to communicate with the RMC, setting this to the addressing method you are using makes it easy to find addresses for registers in the RMC. You can also set the address format by right-clicking any **Register** cell in RMCTools and choosing **Address Format**.

Output Window

Check this box to automatically close the Output window after a successful Upload All or Download All, unless it was previously open.

Plot File Format

Choose the default plot file type when saving plots. See Saving and Exporting Plots for details on the plot file types.

Programming Page

Program Timing

When RMCTools verifies the Programming, it displays the results in the Verify Results window. This option specifies how much detail is shown in the Verify Results window. For more details, see the Program Capacity and Time Usage topic.

Verify Results

Check this box to automatically close the Verify Results window after a successful verify, unless it was previously open.

Step Editor

Select whether to show the tag browser when editing expression in the expression editor.

Communications Page

Communication Log

Choose how much information is saved in the Communication Log. This option should only be changed when directed by Delta technical support.

Firmware Update

Specify the communication rate at which the firmware download to the RMC75S and RMC75P occurs. Higher baud rates will mean quicker firmware updates. The default baud rate is 115200, and should be supported by most PCs. If you have trouble with this baud rate, you should try selecting 38400. Baud rates above 115200 are generally only supported on specialized serial ports.

USB/Serial Timeouts

This section specifies the timeouts used for DF1 communciation via the RMC75P monitor port, the RMC75S monitor port and comm port, and the RMC75E USB monitor port. The timeouts can be adjusted to accommodate a poor serial connection.

Ethernet Timeouts

If you are connected to the RMC via Ethernet and are experiencing lost connections, you can adjust the Ethernet timeouts to accommodate the unreliable connection. Several preset profiles are available, or you can choose custom to set your own timeouts:

- **Default:** This setting works for most LANs (Local Area Networks) and Internet connections.
- **LAN:** This setting is optimized for LAN connections.
- **Remote:** This setting is intended for slow or unreliable Internet connections.

Plots Page

Plot File Format

Defines the default file type when manually saving plots. RMCTools plots can be saved and opened in two formats:

- **RMCTools Plots - Compressed (*.rmcplotx)**
  Compressed file containing one or more plots. Typically, this file size is 10% of the .rmcplots file size. The compressed (.rmcplotx) format is supported in RMCTools 4.09.0 and newer. The uncompressed (.rmcplots) format must be used if plots need to be opened in earlier versions of RMCTools.
- **RMCTools Plots (*.rmcplots)**
  Uncompressed file containing one or more plots. This uncompressed XML file is easily readable in a text editor.

Auto-saved plots will always save in compressed format (.rmcplotx). See Saving and Exporting Plots for more details.

Plot Auto-Save

Enable or disable the Auto-Saved Plots feature that automatically saves uploaded plots to an internal folder on the PC.

You may adjust the amount of storage reserved for the auto-saved plots. The Plot Auto-Save Size Limit range is 1-2048 MB.

See Auto-Saved Plots for details.

See Also

Help Overview

4.16.9. Communication Log

The Communication Log is a file that records the communication between RMCTools and the RMC. This log is only useful when requested by Delta technical support.

To save the Communication Log

- On the **Controller** menu, choose **Save Communication Log**.

The amount of detail to display in the Communication Log can be selected in the RMCTools Options dialog, in the Communications: General section.

See Also

Monitor Port

4.16.10. Keyboard and Mouse Shortcuts

Tip:
If you are looking for shortcuts for sending commands to the RMC, see the Shortcut Commands topic.

Use the keyboard and mouse shortcuts listed in this topic to use RMCTools.

General

Key	Action
F1	Help
Tab	Next Item
Shift Tab	Previous Item
F4	Go to next line in Find Results or Verify Results window.
Ctrl+F4	Close Window
Wheel button	Click a window tab with the wheel button to close.
Alt+F4	Close Application
Arrow Key	Move selection in tables
Ctrl+W	Opens the dialog or wizard in any cell or box that has one, for example, the Command Selection Dialog in the Command box in the Command Tool.
Mouse Wheel	Vertical scrolling (supported in many areas)
Shift+Mouse Wheel or Mouse Wheel Tilt or Horizontal Mouse Wheel	Horizontal scrolling (supported in many areas)

File Menu Commands

Key	Action
Ctrl+N	Create a New Project
Ctrl+O	Open a Project
Ctrl+S	Save the Project

Edit Menu Commands

Key	Action
Ctrl+C	Copy
Ctrl+V	Paste
Ctrl+X	Cut
Ctrl+A	Select All
F2	Rename
Ctrl+F	Find
Ctrl+H	Replace
Ctrl+Shift+F	Find All
F3	Find Next

View Menu Commands

Key	Action
Alt+0	Open Project pane
Alt+1	Open Output window
Alt+2	Open Verify Results window
Alt+3	Open Discrete I/O Monitor
Alt+4	Open Find Results window
Alt+5	Open Task Monitor
Alt+8	Open Command Tool
Alt+9	Bring focus to current editor
Ctrl+T	Open Axis Tools
Ctrl+E	Open Event Log Monitor
Alt+Enter	View Properties of selected item

Controller Menu Commands

Key	Action
Ctrl+Shift+O	Go Online
Ctrl+Shift+D	Download All to Controller
Ctrl+Shift+U	Upload All to Controller
Ctrl+R	RUN Mode
Ctrl+Shift+R	PROGRAM Mode
Ctrl+I	Enable Controller
Ctrl+K	Fault Controller
Ctrl+Shift+C	Clear All Faults - attempts to clear all error bits on all axes

Programming Menu Commands

Key	Action

F7	Verify Programs
Ctrl+Shift+L	Browse Labels

Plots Menu Commands

Key	Action
Ctrl+Shift+P	Open Plot Manager

Editor Menu Commands (Step Editor for User Programs)

This menu is available when the Step Editor is open.

Key	Action
Insert	Add Step Before
Alt+Insert	Add Step After
Ctrl+Shift+Z	Append Step
Ctrl+Insert	Add Command Before
Alt+Ctrl+Insert	Add Command After
Alt+Ctrl+Z	Append Command
Ctrl+Q	Edit Comment
Ctrl+L	Edit Label
Ctrl+P	Print Current User Program

Editor Menu Commands (User Function Editor)

This menu is available when the User Function Editor is open.

Key	Action
Insert	New User Function
Delete	Delete Selected User Function
Ctrl+P	Print Current User Function

Editor Menu Commands (Curve Tool)

This menu is available when the Curve Tool is open.

Key	Action
Insert	Create New Curve
Delete	Delete Selected Curve
Ctrl+D	Download Curves to Controller
Ctrl+U	Upload Curves to Controller

Window Menu Commands

Key	Action
Ctrl+Tab or Ctrl+F6	Next Window
Ctrl+Shift+Tab or Ctrl+Shift+F6	Previous Window

Help Menu Commands

Key	Action
F1	Open Help

Project Pane

Key	Action
Alt+0	Open Project pane
Right Arrow or Keypad Plus Sign	Expand an Item
Left Arrow or Keypad Minus Sign	Collapse an item
Up Arrow	Move up one item
Down Arrow	Move down one item
Enter	Open an item
Alt+Enter	Properties
Shift+F10	Shortcut menu

Axis Tools and Indirect Data Map

Key	Action
Ctrl+D	Download Parameters
Ctrl+U	Upload Parameters
F6	Switch between Axis Status Registers and Axis Parameter Registers
Right Arrow or Keypad Plus Sign	Expand a section
Left Arrow or Keypad Minus Sign	Collapse a section
Shift+Plus Sign	Expand all sections
Shift+Minus Sign	Collapse all sections

Command Tool

Key	Action
Ctrl+H	Open Command History
Ctrl+Shift+Number	Store current command (number = 0-9)
Enter	Issue a command. If editing a cell, press Enter once to accept the value, then press Enter again to issue the command.

Commands (Global)

Key	Action
Ctrl+I	Initialize all axes
Ctrl+K	Disable all axes
Ctrl+Shift+C	Clear All Axis Faults on all axes
Ctrl+Number	Issue a Stored Command (number = 0-9)

Curve Tool

Key	Action
Insert	Create a new curve
Delete	Delete selected curve
In graph pane:	

Ctrl+Mouse Wheel	Zoom
Ctrl+Plus Sign	Zoom in
Ctrl+Minus Sign	Zoom out
Shift+Mouse Wheel or Mouse Wheel Tilt or Horizontal Mouse Wheel	Pan left and right
Arrow Keys	Move selection between points
Delete	Delete selected point

Plot Manager

Key	Action
Insert	Upload Captured Plot
Ctrl+P	Print Current Plot
Delete	Delete Plot (on History tab)
Ctrl+Page Down	Open the next tab in the Plotting, History, Tuning pane
Ctrl+Page Up	Open the previous tab in the Plotting, History, Tuning pane
In plot pane:	
Ctrl+Mouse Wheel	Horizontal zoom
Ctrl+Plus Sign	Horizontal zoom in
Ctrl+Minus Sign	Horizontal zoom out
Shift+Mouse Wheel or Mouse Wheel Tilt or Horizontal Mouse Wheel	Pan left and right
Ctrl+Shift+Mouse Wheel	Vertical zoom
Ctrl+Shift+Plus Sign	Vertical zoom in
Ctrl+Shift+Minus Sign	Vertical zoom out
Arrow Keys	Move cursor
Page Up	Move cursor to left
Page Down	Move cursor to right
Home	Move cursor to beginning of plot
End	Move cursor to end of plot
Shift+Click	Place second cursor
Ctrl+W	Close selected plot file

Plot Template Editor

Key	Action
Ctrl+N	Add a new plot template
Delete	Delete a plotted data quantity in a custom plot

Ctrl+Down Arrow	Move the selected quantity down in the Plotted Data table
Ctrl+Up Arrow	Move the selected quantity up in the Plotted Data table
Enter	Apply the edits and close the Plot Template Editor

Program Trigger Editor

Key	Action
Ctrl+P	Print Program Trigger

Step Editor

See the **Editor Menu** section above.

User Programs

See the **Editor Menu** section above.

Expression Editor

Key	When Focus is in...	Action
Enter	Multi-line expressions*	Inserts a new line.
	Single-line expressions	Accepts the expression and closes the Expression Editor.
	A list in the Expression Browser	Inserts the selected item in the list.
Shift+Enter	Multi-line expressions*	Accepts the expression and closes the Expression Editor.
Esc	Multi-line expressions*	Accepts the expression and closes the Expression Editor.
	Single-line expressions	Discards the changes and closes the Expression Editor.
	The Expression Browser	Moves the focus to the Expression Editor.
F6		Changes focus between the editor to the Expression Browser.
Ctrl+Tab	The Expression Browser	Opens the next tab.

*Multiline expressions include the Expression (113) command and Link Type conditions.

Variable Table

Key	Action
Ctrl+D	**On Edit tab:** Download Variable Definitions **On Monitor tab:** Download Current Values
Ctrl+P	**On Edit tab only:** Print Variable Table
Ctrl+Page Down	Open the next tab
Ctrl+Page Up	Open the previous tab
Plus Sign	Expand an array or DWORD
Minus Sign	Collapse an array or DWORD

See Also

Shortcut Commands

4.16.11. Copy and Paste

RMCTools supports copying and pasting many objects including text, table cells, user programs, steps, and even controllers using the Windows clipboard. Copying text or table cells will place the copied contents on the Windows clipboard and make them available to be pasted in RMCTools or other applications. Text or table cells copied to the clipboard from other applications, such as Microsoft Excel, can also be pasted in RMCTools.

To copy text or table cells:

Copying means that the selected contents remain unchanged, but are copied to the clipboard.

1. Select the desired text or cells:
    - **With the mouse:** Click and drag the mouse to select text or multiple cells.
    - **With the keyboard:** Hold down the Shift key and use the arrow keys to select the text or cells.
2. Copy the text or cells:
    - **With the menu:** On the **Edit** menu, choose **Copy**.
    - **With the shortcut menu:** Right-click the selection and choose **Copy**.
    - **With the keyboard:** Press Ctrl+C.

To cut text or table cells:

Cutting means that the selected contents are deleted and copied to the clipboard.

1. Select the desired text or cells:
    - **With the mouse:** Click and drag the mouse to select text or multiple cells.
    - **With the keyboard:** Hold down the Shift key and use the arrow keys to select the text or cells.
2. Cut the text or cells:
    - **With the menu:** On the **Edit** menu, choose **Cut**.
    - **With the shortcut menu:** Right-click the desired selection and choose **Cut**.
    - **With the keyboard:** Press Ctrl+X.

To paste text or table cells:

1. Click the desired paste location or move the cursor to the desired paste location.
2. Paste the text or cells:
    - **With the menu:** On the **Edit** menu, choose **Paste**.
    - **With the shortcut menu:** Right-click the desired paste location and choose **Paste**.
    - **With the keyboard:** Press Ctrl+V.

    Pasted text will be inserted at the selection. Pasted cells will overwrite the existing cells.

Pasting to Microsoft Excel

Copying text or table cells from RMCTools and pasting to Microsoft Excel typically works smoothly. However, if the copied content consists of table cells containing quotation marks, you will need to use Excel's Paste Special function to properly paste the cells.

In Excel 2007 or later, on the Home ribbon, click the down arrow on **Paste**, then click **Paste Special.** Choose **CSV** and click **OK**.

In Excel 2003 or earlier, on the **Edit** menu, choose **Paste Special**. Choose **CSV** and click **OK**.

See Also

RMCTools Overview

4.16.12. Find and Replace

Find All

Search for user-entered text in the entire RMCTools project, including the Variable Table, User Programs, Program Triggers, User Functions, Discrete I/O Configuration Editor, Curve Tool and the Indirect Data Map.

1. On the **Edit** menu, choose **Find and Replace**, then click **Find All**.
2. Enter the desired search text in the **Find what** box.
3. Set the **Find options** as desired.
4. Click **Find All**. All occurrences of the search text will be displayed in the **Find Results** window, which will open at the bottom of the RMCTools window.
5. Clicking an occurrence will open the location of that occurrence. Or, press F4 to open the location of the next ocurrence and press Shift + F4 to open the location of the previous occurrence.

Find Text in an Editor

Search for text in a text-based editor, such as the User Function Editor and the Expression (113) command.

1. On the **Edit** menu, choose **Find and Replace**, then click **Find**.
2. Enter the desired search text in the **Find what** box.
3. Set the **Find options** as desired.
4. Click **Find Next** or press F3. The next occurrence of the search text will be highlighted.
5. To find the next occurrence, click **Find Next** or press F3 again.

Replace Text in an Editor

Search for and replace text in a text-based editor, such as the User Function Editor and the Expression (113) command.

1. On the **Edit** menu, choose **Find and Replace**, then click **Replace**.
2. Enter the desired search text in the **Find what** box.
3. Enter the desired replacement text in the **Replace with** box.
4. Set the **Find options** as desired.
5. To find the next occurrence, click **Find Next** or press F3. To replace the currently highlighted occurrence, or the next occurrence, click **Replace**. To replace all occurrences, click **Replace All** again.

See Also

RMCTools Overview

4.16.13. Printing in RMCTools

To print the desired item, on the toolbar within the desired editor, click the Print button  . Or, on the **File** menu, choose **Print**.

RMCTools offers printing of the following parts of the project:

- **User Programs**
- **Program Triggers Editor**
- **Variable Table**
- **User Functions**
- **Plots**

See Also

RMCTools Overview

4.16.14. Installing RMCTools

Installing RMCTools is normally a straight-forward process. Download the installation file from the Download section of Delta's website and run the installer.

By default, with no options or properties, the installer will open the full installation wizard. If RMCTools is not installed, the wizard will guide the user through a new installation. If an older version of RMCTools is installed, the wizard will upgrade that version. And if the same version of RMCTools is already installed, the wizard will offer to Repair or Remove the installation. Installation will include both the RMC70/150 USB drivers and RMC200 USB drivers.

Command Line Options

The following options and properties can be added to the command line to modify this behavior:

Operations (select zero or one):

- **/install**
  Install the product. This is the default.
- **/uninstall**
  Uninstall the product.

Interface options (select zero or one):

By default the full user interface is used.

- **/quiet**
  Quiet mode, no user interaction. Important: This operation will only succeed if it is launched as Administrator, such as from an Administrator command prompt.
- **/passive**
  Unattended mode, progress bar only. The only interaction required is to elevate to administrator privileges or to prompt for permission to install the RMC200 USB driver.

Restart Options (select zero or one):

- **/norestart**
  Do not restart after installation is complete.
- **/promptrestart**
  Prompts the user for restart if necessary. This option cannot be used with **/quiet**.
- **/forcerestart**
  Always restart the computer after installation.

Other options:

- **/log log.txt**
  Saves the log file to the specified log filename. By default, the log is saved as **RMCTools_Install.log** in the user's Temp directory.
- **/help or /?**
  Displays this information.

Available Properties:

The following properties can also be provided from the command line and affect the behavior of the installation as described.

- **EXCLUDE_USB_70_150=1**
  Do not include the RMC70/150 USB drivers in the installation.
- **EXCLUDE_USB_200=1**
  Do not include the RMC200 USB drivers in the installation.
- **INSTALLDIR="installpath"**
  Defines the path to install the product. By default this is **C:\Program Files\RMCTools**. This is most useful in **/quiet** and **/passive** installs, but will also set the default installation path when the full install wizard is used.

Examples:

- **rmctoolsinstall64**
  Install RMCTools using the full standard install wizard including all USB drivers. Equivalent to 'rmctoolsinstall64 /install'.
- **rmctoolsinstall64 /uninstall**
  Uninstall RMCTools.
- **rmctoolsinstall64 /quiet /norestart EXCLUDE_USB_200=1**
  Install RMCTools with no user interface and no restart. Do not include the RMC200 USB driver. This operation will only succeed if running from an Administrator prompt or equivalent.

See Also

Help Overview

4.17. Wizards

4.17.1. Scale/Offset Wizard Overview

The Scale/Offset wizards help you calculate the Scale and Offset parameters. For details on the function of the Scale and Offset parameters, see the Scaling Overview topic.

Accessing the Scale/Offset Wizards

To open a Scale offset Wizard, open Axis Tools. In the Axes Parameters Pane, click the **Setup** tab. Expand the **Tools and Wizards** section. Click **Launch** to open the Scale/Offset Wizard.

Notice that before using the Scale/Offset Wizard, you must define the axes and configure the axis type.

Scale/Offset Wizards

Below is a list of the Scale/Offset Wizards available in RMCTools.

SSI Scale and Offset
MDT Scale and Offset
Quadrature Scaling
Resolver Scale and Offset
Rotary Scale and Offset
Pressure/Force Scale and Offset
Differential Force Scale and Offset
Load Cell Scale and Offset
Generic Scale and offset Using the Position/Counts Method

Scaling Specific Axis Types:

Analog Position Scaling
Analog Velocity Scaling
Analog Acceleration Scaling
Analog Pressure/Force Scaling
MDT Scaling
SSI Scaling
Load Cell Scaling
Quadrature Scaling
Resolver Scaling

See Also

Scaling Overview | Analog Position Scaling | Analog Velocity Scaling | Analog Acceleration Scaling | Analog Pressure/Force Scaling | MDT Scaling | SSI Scaling | Quadrature Scaling | Resolver Scaling | Load Cell Scaling

4.17.2. Autotuning Wizard: Welcome Page

Next Wizard Page

To access this wizard:
In the Axis Parameters Pane, on the **Tune** tab, in the **Tools and Wizards** section, on the **Position Tuning Wizard** row, click **Launch**. Choose **Use Autotuning Wizard** and click **Next**.
Or,
In the Plot Manager, on the **Tuning** tab, click **Tuning Wizard**. Choose **Use Autotuning Wizard** and click **Next**.

This page of the Autotuning Wizard describes the autotuning procedure that the wizard will lead you through.

Read the information and warnings. If you agree, check the check box and click **Next**. If not, you will not be able to complete the wizard.

Additional Information

Activating the Axis

You will use the wizard to generate open loop motion. You specify the position range in which to move and the voltage to apply. For more details on the open loop profile, see the Autotuning Wizard: Move Parameters topic.

The plot(s) captured by the motion will be added to the Plot History just like all plots. After the wizard completes, you can review the plots to see if they represented typical motion on your machine.

Motion systems often do not behave identically at low speeds and high speeds. Therefore, it may be wise to use the Autotuning Wizard as a first pass at tuning. Once the system is controllable, you can capture plots of motion at the intended speed and conditions of normal machine operation. These plots can then be used with the Tuning Wizard to compute a system model that more accurately represents the system during its normal operation.

Building the System Model

Once the plots have been captured, the wizard computes the mathematical model of the system. These model parameters are part of the axis parameters and are accessible in the Axes Parameters Pane, **All** tab, Feedback → Filtering/Modeling section. Once you have saved your project and/or updated Flash, the model parameters generated by the wizard will always be accessible to you.

Choosing Gains

After the system model parameters have been generated, the Gain Calculator will open, where you can choose from a range of appropriate gains based on the system model. See the Gain Calculator topic for more details.

The model parameters are saved as part of the Axis Parameters. If, after creating a model, you have saved the axis parameters, you can open the Gain Calculator at any time and choose a new set of gains based on that model.

See Also

Autotuning

4.17.3. Autotuning Wizard: Enter Move Parameters Page

Previous Wizard Page Next Wizard Page

In the Axis Parameters Pane, on the **Tune** tab, in the **Tools and Wizards** section, on the **Position Tuning Wizard** row, click **Launch**. Choose **Use Autotuning Wizard** and click **Next**.
Or,
In the Plot Manager, on the **Tuning** tab, click **Tuning Wizard**. Choose **Use Autotuning Wizard** and click **Next**.

On this page of the Autotuning Wizard, enter the parameters for the moves you will make. The parameters specify the Control Output profile that will be used. These values you choose here can be very important. For details, see the respective section below for your system type.

Hydraulic System or Electro-servo Motor in Velocity Mode

The Parameters

Starting Position (linear axes): This specifies the position at which you wish to start each move. In the autotuning, you will be given the option of first moving the axis to this starting position before doing the autotuning move. It is not necessary to move the axis to the starting position. If you have selected two moves, each move will be limited by the starting position of the other move. That is, if the Actual Position reaches the other starting position, the axis will be halted. The autotuning will not necessarily use the entire motion range. On a hydraulic system, the autotuning will typically require only a couple inches. Therefore, you may be able to Move the Starting Positions for each move closer to each other to avoid moving the axis so far just to start the autotuning move.

Move Limit (linear axes, single move): This defines the maximum length of the move. If the Actual Position moves this distance from the specified **Starting Position**, the axis will be halted.

Maximum Distance (rotary axes): This specifies the maximum distance the axis is allowed to travel during autotuning. If the Actual Position travels farther than this distance, the axis will be halted.

Move Direction: Specify which direction to move. If you are doing two moves, this specifies the direction of the first move.

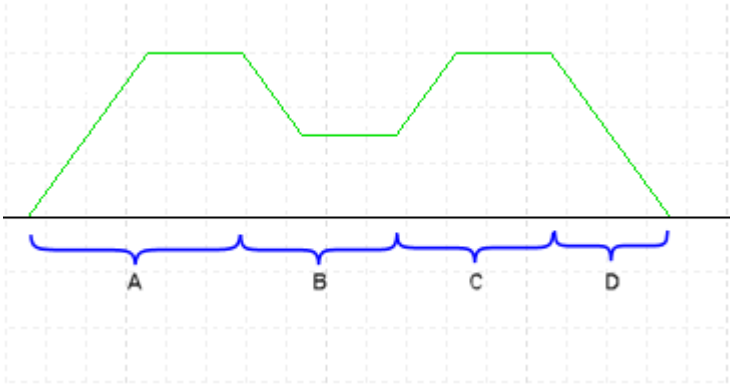
Output Voltage: Specifies the amount of Control Output voltage that will be applied during autotuning. The default value of 3 volts is typically fine, but if you know the level at which your axis will operating normally, you can set this to that value.

Ramp Rate: Specifies how fast the Control Output should ramp up to the specified **Output Voltage**. This needs to be fairly high to provide enough information about the system. For systems with fast response, you may wish to increase this value.

Plot Duration: This must be long enough to capture the entire move. If the entire move is not captured, click back to this page and increase the plot duration. The plot duration should be less than twice as long as the move time.

The Profile

For hydraulic systems or velocity mode motors, when you instruct the wizard to move the axis, it will generate an open loop voltage profile similar to that shown below. Notice that this move will not necessarily take up the entire range between the specified beginning and ending points! On many hydraulic systems, the moves takes only a couple of inches.



The parameters you enter on this page affect the Control Output profile as described below:

Section A: The Control Output ramps up to the user-specified **Output Voltage** at the specified **Ramp Rate**. The Control Output stays at that value until the axis stops accelerating, at which point it goes to section B.

Section B: The Control Output ramps down to half of the user-specified **Output Voltage**. The Control Output stays at that value until the axis stops accelerating, at which point it goes to section C.

Section C: The Control Output ramps back up to the user-specified **Output Voltage**. The Control Output stays at that value until the axis stops accelerating, at which point it goes to section D.

Section D: The Control Output ramps down to zero.

Since the length of sections A, B and C depend on when the axis stops accelerating, the time of this move profile will vary from system to system. Typically, it is less than 0.5 seconds. If the Actual Position gets too close to the specified End Position at any time during the profile, the voltage will go to zero immediately.

Electro-servo Motor in Torque Mode

The Parameters

Maximum Distance: This specifies the maximum distance the axis is allowed to travel during autotuning. If the Actual Position travels farther than this distance, a stop pulse (see profile below, section C) will be applied immediately to halt the axis.

Target Speed: This is the speed at which the axis will move. The Control Output will be ramped to the specified Output Voltage and remain there until the Actual Velocity reaches this value (see profile below, section A).

Move Direction: Specify which direction move. If you are doing two moves, this specifies the direction of the first move.

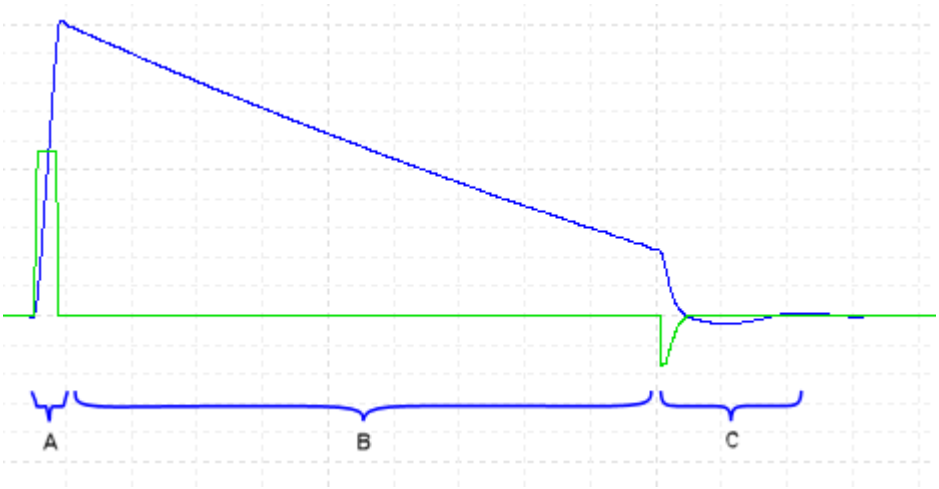
Output Voltage: Specifies the amount of Control Output voltage that will be applied during autotuning. This value should be large enough for the Actual Velocity to reach the **Target Speed**.

Ramp Rate: Specifies how fast the Control Output should ramp up to the specified **Output Voltage**. This needs to be fairly high to provide enough information about the system. For systems with fast response, you may wish to increase this value.

Plot Duration: This must be long enough to capture the entire move. Torque mode systems with low damping may require a very long plot duration. If the entire move is not captured, click back to this page and increase the plot duration. The plot duration should be less than twice as long as the move time.

The Profile

For torque mode autotuning, when you instruct the wizard to move the axis, it will generate an open loop voltage profile similar to that shown below. Notice that this move might not necessarily take up the entire distance specified by the Maximum Distance parameter, but it will typically move much further during autotuning than a velocity mode system will.



The parameters you enter on this page affect the Control Output profile as described below:

Section A: The Control Output (green line above) ramps up to the user-specified **Output Voltage** at the specified **Ramp Rate**. The Control Output stays at that value until the Actual Velocity (blue line above) reaches the specified **Target Speed**, at which point it goes to section B.

Section B: The wizard waits until the Actual Velocity (the blue line above) ramps down to 30% of the specified **Target Speed**.

Section C: A stop pulse is given by setting the Control Output -30% of the user-specified **Output Voltage**. As the Actual Velocity decreases, the Control Output is ramped proportionally down to zero.

The time of this move profile can vary greatly from system to system. If the Actual Position travel approaches the user-specified **Maximum Distance**, or if the time exceeds the user-specified **Plot Duration**, the stop pulse of section C will be applied immediately.

See Also

Tuning Wizard Overview | Autotuning Wizard

4.17.4. Simulator Wizard

To access the Scale/Offset wizard:
In Axis Tools, in the Axes Parameters Pane, click the **Setup** tab. Expand the **Tools and Wizards** section. Click **Launch** to open the Simulator Wizard.

Use the **Simulator Wizard** to quickly set up the axis in simulate mode. The simulator is available on position, position-pressure, and position-force axes. The wizard will fully configure the scale and offset parameters, tuning gains, travel limits, and simulator settings. The simulator wizard does not apply higher-order gains.

Instructions

Use the Simulator Wizard to quickly set up the axis in simulate mode.

Axis parameters such as Linear/Rotary, SSI Data Bits, and Analog Input Type (±5V, ±10V, 4-20mA) should be set before using the wizard.

1. **Enter Desired Position Range**
   Enter the desired range of position travel. For rotary axes, choose the number of position units per rotation.

2. **Enter Maximum Velocity**
   This is the velocity at which the axis will move with 10 V of Control Output.

3. **Enter Maximum Acceleration**
   For best results, set this value significantly higher than the acceleration rates you intend to use on the axis. This is not necessarily a true limit of the acceleration, but helps determine the response of the simulator and the tuning gains. This value is typically at least an order of magnitude (10x) greater than the maximum velocity.

4. **Enter Pressure or Force Information**
   If the axis is position-pressure or position-force, enter the maximum force of the simulator, and the desired range of the force area at the ends of the travel.

   The tuning of the pressure or force is affected by the position settings, especially the **Maximum Acceleration**. If the pressure or force tuning is poor, or pressure or force following errors occur, run the Simulator Wizard again and increase the **Maximum Acceleration**.

5. **Review Parameters**
   Click **Next** and review the axis parameter settings in the **Proposed** column. Any parameters with blue text can be changed. When you are satisfied with the values, click **Finish**.

6. **Download Axis Parameters**

   In the Axis Tools window, click the download button  to apply the parameter changes to the controller.

After setting these parameters, make sure to enable the axis. This can be done by sending the Enable Controller (7) to the RMC.

See Also

Simulating Motion

4.17.5. Firmware Update Wizard

To access this wizard:
Go Online with the controller. In the Project pane, expand the Modules folder, double-click the desired module and click Firmware. Click Update Firmware.

Delta regularly updates the RMC firmware to add features and fix bugs. To update the firmware in the RMC:

- Go to Delta's website at https://deltamotion.com.
- Find the RMC firmware for your RMC on the **Downloads** page, and save it on your computer.
- Start this Firmware Update Wizard, and follow the steps below.

Refer to the Firmware topic for a list of firmware for various RMC modules.

Note:
To update the firmware, the RMC must be connected to the computer running RMCTools via USB, Ethernet, or the RS-232 Monitor Port. The second serial port on the RMC75S will not work for updating firmware. To update firmware via Ethernet, the **Allow updating firmware over Ethernet** box must be checked in the Ethernet Settings Page.

Introduction Page

1. Click **Next** to proceed to the next page.

Select a File Page

1. Click **Browse** to select the firmware file. You must have previously downloaded the RMC firmware file from Delta's download web page and saved it to your computer.
2. Click **Next** to proceed to the next page.

Back Up Firmware Page

This page asks if you upload the firmware from the controller before updating the new firmware. This is recommended in case you need to revert to the old firmware.

1. Choose **Yes** or **No**.
2. Click **Next** to proceed to the next page.

Confirm Page

1. Verify that the information is correct.
2. When the controller begins the firmware update, it cannot control motion. Make sure your system is in a safe state.
3. Click **Next** to proceed to the next page and begin the update.

In Progress Page

1. Wait for the firmware update to complete. If an error occurs, remedy the problem, and then click **Back**. Follow the Confirm page instructions.
2. Click **Finish** to exit the wizard.

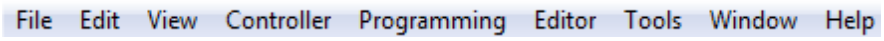
See Also

Firmware

4.18. Menu and Toolbars

4.18.1. RMCTools Menu Bar

The RMCTools menu bar consists of the following menus:



Components

File

New

Controller: Opens the New Controller Wizard which adds a new controller to the project. The controller must first be connected to the computer's serial port via a cable.

Project: Opens a new project. You may be prompted to first save the current project.

Open: Open an existing project. This will close the current project. You may be prompted to first save the current project.

Close: Close the project. You may be prompted to first save the current project. RMCTools will remain open.

Save: Save the project in its current location.

Save As: Save the project in a selected location.

Save and Update Flash: Save the project in its current location and update the flash memory in the currently selected online controller.

Page Setup: Setup the page margins for printing.

Print: Print the current editor.

Exit: Close the project and close RMCTools. You may be prompted to first save the current project.

Edit

Undo: Undo the last action. This is only available in the Step Editor and within expression boxes.

Redo: Redo the previously undone action. This is only available in the Step Editor and within expression boxes.

Cut: Copy and delete the current selection to the clipboard.

Copy: Copy the current selection to the clipboard.

Paste: Paste the current item in the clipboard to the selection.

Delete: Delete the current selection.

Select All: Select all the items in the current editor.

Rename: Rename the currently-selected object.

Find and Replace

Find: Find the specified text in the current editor.

Replace: Replace specific text with different text.

Find All: Find all occurrences of the specified text in the project.

Find Next: Find the next instance of the specified text.

View

Toolbars

Standard: Select whether to display the Standard Toolbar.

Shortcut Commands: Select whether to display the Shortcut Command Toolbar.

Status Bar: Select whether to display the Status Bar.

Project: Opens the Project Pane.

Output: Opens the Output window.

Verify Results: Opens the Verify Results window.

Discrete I/O Monitor: Opens the Discrete I/O Monitor window.

Find Results: Opens the **Find Results** window.

Task Monitor: Opens the Task Monitor window.

Command Tool: Opens the Command Tool window.

Axis Tools: Opens the Axis Tools window.

Event Log: Opens the Event Log Monitor.

Select Columns: Choose which columns are to be displayed in the Axis Tools. The column selection can also be made from the Axis Tools window. The Command Tool will display the same axes that are displayed in the Axis Tools.

Properties: Opens the Properties dialog for the currently selected item in the Project pane.

Controller

Go Online: Connects to the currently selected controller. If the communication is broken, RMCTools will not attempt to restart communication. See Go Online for details.

Go Offline: Closes communication with the currently selected controller. See Go Online for details.

Connection Path: Opens the Connection Path dialog to choose the method of connection to the controller.

Download All to Controller: Downloads all the parameters from the project to the currently selected controller. This will overwrite the controller values.

Upload All from Controller: Uploads all the parameters from the currently selected controller to the project. This will overwrite the project values.

Save Communication Log: Saves the log of RMCTools communications with the RMC. This is only useful when requested by Delta technical support.

RUN Mode: Puts the controller in RUN Mode.

PROGRAM Mode: Puts the controller in PROGRAM Mode.

Enable Controller: Issues an Enable Controller (7) command.

Fault Controller: Issues a Fault Controller (8) command.

Clear All Faults: Issues a Clear Faults (4) command to every axis.

Update Flash: Updates the Flash memory.

View/Change Modules: Opens the RMC Hardware Configuration Dialog dialog.

View/Change Axis Definitions: Opens the Axis Definitions dialog.

Reset Controller to Defaults: Resets the controller to its default settings.

Restart Controller: Restarts the controller. Performs a cold restart, which is the same as cycling power. Anything not saved to Flash will be lost.

View Communication Statistics: Opens the Communication Statistics window.

Programming

Download Programs to Controller: Downloads all the contents of the **Programming** folder to the controller.

Upload Programs from Controller: Uploads all of the contents of the **Programming** folder from the controller.

New User Program: Creates a new User Program.

Export User Program: Exports a User Program.

Import User Program: Imports a User Program.

Verify Programs: Verifyies the programs (all the contents of the **Programming** folder).

Lock Programming: Locks the user programs so changes cannot be made. See Programming Security for details.

Unlock Programming: Unlocks the user programs so changes can be made. See Programming Security for details.

Browse Labels: Displays a list of all the Labels in the user programs and their locations.

Programming Properties: Opens the Programming Properties dialog.

Plots

Open Plot Manager: Opens the Plot Manager.

Edit Plot Templates: Opens the Plot Template Editor.

Download Plot Templates to Controller: Downloads the Plot Templates to the controller.

Upload Plot Templates from Controller: Uploads the Plot Templates from the controller.

Save Plots: Save any of the plots listed on the **History** tab.

Open Plot File: Open a saved plot file.

Recent Plot Files: List of recently saved or opened plot files.

Select Active Plot: Select which of the Plot Templates is active. The Capture and Trend buttons on the Plot Manager Toolbar apply to the active plot template.

Start Trend: Start a trend using the active plot template.

Upload Captured Plot: Upload a captured plot using the active plot template.

Stop Trend/Upload: Stop the current upload or trend in progress.

Trigger Plot: Start capturing a plot immediately. This will not upload the plot; to do so, click the **Capture** button.

Re-arm Plot: Rearm the plot. This is the same as sending the Rearm Plot (103) command.

Zoom: Choose a zoom method.

View: Choose to view various Plot Manager items.

Tools

Plot Manager: Opens the Plot Manager.

Tuning Tools: Opens the Tuning Tools.

Options: Opens the RMCTools Options dialog.

Editor

This menu applies to currently open editor.

Axis Tools - see the Axis Tools topic for details.

Axis Status Registers: Displays only the Axis Status Registers pane in Axis Tools.

Axis Parameter Registers: Displays only the Axis Parameters pane in Axis Tools.

Both - Side-by-Side: Displays both the Axis Status Registers and Axis Parameters pane side-by-side in Axis Tools.

Both - Above-Below: Displays both the Axis Status Registers and Axis Parameters pane above-and-below in Axis Tools.

0 - Axis0: Choose whether or not to display this column in Axis Tools.

View Project Values: View the Axis Parameters values as they are in the project.

View Controller Values: View the Axis Parameters values as they are in the controller.

Download Axis Parameters to Controller: Downloads everything in the Axis Parameters of the Axis Tools to the controller.

Upload Axis Parameters from Controller: Uploads everything in the Axis Parameters of the Axis Tools from the controller.

Step Editor - see the Step Editor topic for details.

Add Step Before: Add a step above the currently selected step.

Add Step After: Add a step below the currently selected step.

Append Step: Add a step to the end of the program.

Delete Step: Delete the currently selected step.

Move Step Up: Move the currently selected step up one step.

Move Step Down: Move the currently selected step down one step.

Add Command Before: Add a command above the currently selected command.

Add Command After: Add a command below the currently selected command.

Append Command: Add a command after the other commands in the step.

Remove Command: Delete the selected command.

Move Command Up: Move the currently selected command up one.

Move Command Down: Move the currently selected command down one.

Add Condition Before: Add a condition above the currently selected condition.

Add Condition After: Add a condition below the currently selected condition.

Append Condition: Add a condition after the other conditions in the Conditional Jump link type.

Remove Condition: Delete the selected condition.

Move Condition Up: Move the currently selected condition up one.

Move Condition Down: Move the currently selected condition down one.

Edit Step Label: Edit the label of the selected step.

Edit Step Comment: Edit the comment of the selected step.

Show Comments: Show all the comments of all steps.

Hide Comments: Hide all the comments of all steps.

Edit Program Comment: Edit the comment of the user program.

Window

Close: Close the current window.

Close All: Close all the open windows.

Close All But This: Close all the open windows except the currently active window.

Next: Display the next open window.

Previous: Display the previous open window.

Split Side-by-Side: Split the active window into a new vertical tab group.

Split Above-Below: Split the active window into a new horizontal tab group.

Move to Previous Tab Group: Move the active window to the previous tab group.

Move to Next Tab Group: Move the active window to the next tab group.

Reset to Basic Layout: Resets the windows and dockable panes to the basic layout. This is useful if the windows become jumbled.

Reset to Small-Screen Layout: Resets the windows and dockable panes to the layout optimized for smaller screens. This sets the Project pane and Command Tool to auto-hide, freeing up screen space.

Help

Help Topics: Opens the help.

Video Tutorials: Links to the videos page on Delta's website for tutorials on the startup procedure, basic motion, communications, and more.

Product Support: Links to the support page of Delta's website.

Check for Updates: Checks if there is a newer version of RMCTools available. If there is, provides a link to the RMCTools download page on Delta's website. Requires an Internet connection.

What's New: Opens a window describing recent important changes to RMCTools.

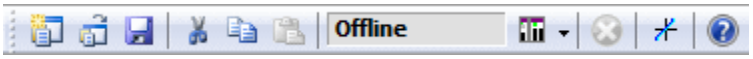
Show Startup Wizard: Opens the Startup Wizard for creating a new project or opening an existing project.

About RMCTools: Shows version information for the RMCTools software and technical support contact information.

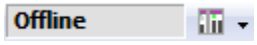
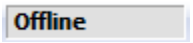
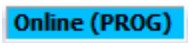
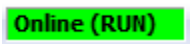
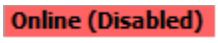
See Also

Using the RMCTools Interface | Standard Toolbar | Status Bar | Shortcut Commands

4.18.2. Standard Toolbar



The following buttons are available on the Standard Toolbar:

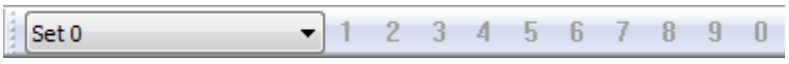
New Project	Creates a new .
Open Project	Opens an RMCTools project.
Save Project	Saves the RMCTools project.
Cut	Cuts the selection and places it in the clipboard.
Copy	Copies the selection to the clipboard.
Paste	Pastes the clipboard contents.
Controller 	Indicates the state of the controller.
	RMCTools is not communicating with the controller.
	RMCTools is connected to the controller and the controller is in PROGRAM Mode.
	RMCTools is connected to the controller and the controller is in RUN Mode.
	RMCTools is connected to the controller

	and the controller is in the Disabled state (RMC200 Only).
	Click the controller button  to do any of the following: • Go Online • Go Offline • Change the Connection Path • Download All to Controller • Upload All from Controller • Enter RUN Mode, PROGRAM Mode, or the Disabled state (RMC200 Only). • View Controller Properties
Fault Controller	Sends the Fault Controller (8) command to the controller. This command halts all the axes and puts the RMC75 and RMC150 in PROGRAM mode. For the RMC200, this command puts the RMC in the Disabled state.
Axis Tools	Opens the Axis Tools window.
Help	Opens the help.

See Also

Using the RMCTools Interface | RMCTools project | Status Bar | Shortcut Command Toolbar

4.18.3. Shortcut Command Toolbar



Use Shortcut Command Toolbar to issue Shortcut Commands.

To Issue a Shortcut Command

- Click on one of the numbered buttons to issue the commands assigned to that button for the selected shortcut set. The grayed buttons do not have not been assigned any commands.
- Or, press Ctrl + *num* on the keyboard, where *num* is the number of the button, e.g. Ctrl+2.

To Select a Shortcut Set

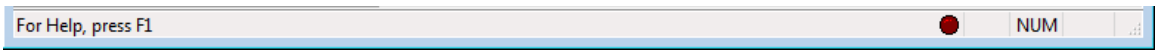
- Choose the active shortcut set from the box.

See Also

Using the RMCTools Interface | Standard Toolbar | Status Bar | Shortcut Commands

4.18.4. RMCTools Status Bar

The status bar appears at the bottom of the RMCTools window. It may look something like this:



The Status bar displays the following information:

- **Help Text**
  This is a simple message on the left side of the status bar that describes the currently selected menu item or toolbar button. It generally displays the same information as the mouse tooltip.
- **Error Icon**  **,**  **,** 
  The error icon indicates whether an error has occurred, and will pop up a bubble when an error occurs. See the Error Icon topic for details.
- **CAP / NUM / SCROLL**
  Indicates whether the Caps Lock, Num Lock, and Scroll Lock are active.

See Also

Using the RMCTools Interface | Standard Toolbar | Shortcut Command Toolbar

5. Programming

5.1. Programming Overview

The RMC has a rich set of pre-programmed high-level motion commands. In addition, it can easily be programmed to perform simple motion sequences or complex actions. With the RMC features listed below, motion applications can be done entirely within the RMC, or in conjunction with a PLC.

Note:
In order to run User Programs or the Program Triggers, the RMC must be in RUN mode. See the RUN/Program Mode topic for details.

Programming Examples

See the Programming Examples topic for tips and examples of programming the RMC.

Commands

The commands are the building blocks of RMC programming. Commands tell the RMC what to do. For a list of commands, see the Command List topic. Commands may be issued from the following places:

- From RMCTools using the Command Tool.
- From a PLC or other host controller via the communication port.
- From a User Program.

See the Issuing Commands topic for details on issuing commands.

User Programs

A User Program carries out a sequence of commands without requiring a PLC or other controller. This allows the RMC to respond to events within its control-loop time rather than the scan rate of the PLC. It also reduces the controller programming required.

A User Program consists of multiple steps. Each step can issue a command on one or several axes. The series of steps are linked together in sequences. The link types allow branching and looping, waiting for conditions and many other features. An RMC controller may execute several User Programs simultaneously.

User Programs run on Tasks. One Task can run one User Program at a time. The RMC75 has 4 tasks and can run 4 User Programs simultaneously. The RMC150 has 10 tasks and can run 10 User Programs simultaneously.

For details on creating and running User Programs, see the User Program topic.

Mathematical Expressions in User Programs

Advanced math operations using expressions are also possible in user programs. RMCTools provides a rich set of functions and operators, and allows for user-defined functions.

Program Triggers

You can set up program triggers to automatically start user programs when some condition becomes true. For example, you can trigger a user program when a discrete input turns on, or a variable becomes a certain value. This is a good way to start user programs from a PLC.

See the Program Triggers topic for details.

Variables

Variables make the User Programs very flexible. Variables can be used to effortlessly change programs, make programs readable, and easily influence User Programs via a PLC. Variables may be used in command parameters, the Expression (113) command, and several Link Types.

Variables can be individually selected to be retentive. The Current Value of retentive variables will be retained between power cycles without requiring a Flash update. This is useful for retaining data such as setpoint positions, machine cycle counters, and recipe data.

Retentive variables are only available on the RMC75E (version 1.4A or newer) and the RMC150E and require firmware 3.30.0 or newer.

See the Variables topic for details.

Discrete I/O

The discrete I/O on the RMC can be used to control the programming, and the programming can control the discrete I/O. See the Discrete I/O topic for details.

Programming from External Systems

The RMC can be controlled from a PLC or other host controller. It can be done in the following ways:

Entirely with a PLC

A PLC can exercise complete control over the RMC by issuing commands to it. The RMC supports many communication protocols, such as DF1, Modus/RTU and PROFIBUS.

With the RMC and PLC (or HMI)

The RMC can be programmed using a combination of a host PLC (or HMI) and User Programs. This allows fast and time-critical sequences to be implemented in the User Program (which executes at the RMC loop time), while less time critical functions are handled via the PLC.

For details on using the RMC with an HMI, see the Communicating with HMIs topic.

Program Size and Time

See the Program Capacity and Time Usage topic for details.

See Also

User Program Overview | Program Triggers | Variables

Show All

5.2. Issuing Commands to the RMC

The commands are the building blocks of RMC programming. Commands tell the RMC what to do. For a list of commands, see the Command List topic. There are several methods of sending commands to the RMC:

Issuing Commands from RMCTools

When you are setting up, tuning, programming and troubleshooting the RMC, use the Command Tool in RMCTools to issue commands to the RMC. See the Command Tool topic for details on issuing commands from RMCTools.

For Shortcut commands, see the Shortcut Commands topic.

Issuing Commands from within a User Program

User Programs carry out complex sequences of commands on the RMC without requiring intervention from a PLC or other controller.

See the User Programs topic for details on how to create and run user programs to issue to commands to the RMC.

Issuing Commands from a Host Controller, such as a PLC or HMI

You can issue commands directly to the RMC from a host controller, such as a PLC or HMI. It is possible to issue commands to multiple axes simultaneously.

Via PROFIBUS

See the PROFIBUS Overview topic and read the topic of the mode you are using.

Via PROFINET IO

See the Using a PROFINET I/O Connection topic.

Via EtherNet/IP I/O

See the Using an EtherNet/IP I/O Connection topic.

Via Ethernet (Modbus/TCP, FINS/UDP, CSP or DF1 over Ethernet, Mitsubishi Procedure Exist)
or Serial RS-232/485 (Modbus RTU, DF1, Mitsubishi Bidirectional Protocol)

Use the following procedure to issue a command:

1. **Determine which Axes**
   Determine which axis or axes you are going to issue the command to.
2. **Find the Address**
   Use the following topics to locate the addresses of the Command Registers for the communication protocol and RMC you are using:
   DF1 Addressing
   Modbus Addressing
   FINS Addressing
3. **Write to the Command Registers**
   Write values to the Command Registers of the axis or axes you wish to issue the command to. You need only write to as many Command Parameters as are used by the command you will issue.

   Note:
   The RMC will process the command as soon as a write is made to the Command register. Most PLCs or HMIs can write to the these in one write, such as a block write.
   If your host controller can only write to one register at a time, it is important that you write to the Command register last. If your system can write to multiple registers at a time, you can write to all the Command Area registers at once.

   Note:
   If you are using a 16-bit addressing protocol, such as Modbus, you must write 2 words for each register you wish to write to in the RMC. All command registers are 32-bit floating-point values.

   Caution: For each command you wish to issue, write it only once so that it is issued only once. If you issue the same motion command multiple times, it can cause the target position to overshoot the requested position.

4. **Optional: Use the Command Request and Acknowledge Bits**
   If your system requires tight synchronization, you should use the Command Request and Acknowledge bits. These bits indicate that the RMC has received the command. Typically, these bits are only needed when using a PLC or other highly capable host controller. See the Command Request and Acknowledge bits topic for details.
   For example, if you issue a Move command and then wait for the In Position Status bit to be set, it is important that you do not start checking the In Position status bit until the RMC has received the command. The Command Request and Acknowledge bits tell you when there RMC has received command.

Examples

1. RMC75: Move Absolute Command to Axis 1

The user wishes to issue the following move absolute command to Axis 1 of the RMC75:

Move Absolute(20):

- Command Parameter 1: Position = 5.2 in.
- Command Parameter 2: Speed = 14 in./sec
- Command Parameter 3: Acceleration = 100
- Command Parameter 4: Deceleration = 100
- Command Parameter 5: Direction = 0

With Allen-Bradley DF1 Addresses:

From the RMC75 Register Map - File 25 Commands, we see that the addresses for the Axis 1 command registers are F25:10-19. To issue this command to the RMC75 via DF1, CSP, or EtherNet/IP, do the following:

1.

1. Write 20 to F25:10
2. Write 5.2 to F25:11
3. Write 14 to F25:12
4. Write 100 to F25:13
5. Write 100 to F25:14
6. Write 0 to F25:15

With Modbus/RTU or Modbus/TCP Addresses:

From the RMC75 Register Map - File 25 Commands, we see that the addresses for the Axis 1 command registers start at register offset 12821. The offsets must be prefixed with a 4. To issue this command to the RMC75 via Modbus/RTU or Modbus/TCP, do the following:

7. Write 20 to 412821
8. Write 5.2 to 412823
9. Write 14 to 412825
10. Write 100 to 412827
11. Write 100 to 412829
12. Write 0 to 412831

2. RMC150: Start User Program 8

To start user program 8, the user decides to issue the following command to Axis 0 of the RMC150:

Start Task (90):

- Command Parameter 1: Task = 0
- Command Parameter 2: Program = 8

With Allen-Bradley DF1 Addresses:

From the RMC150 Register Map - File 40 Commands, we see that the addresses for the Axis 0 command registers are F40:0-9. To issue this command to the RMC75 via DF1, CSP, or EtherNet/IP, do the following:

1.

1. Write 90 to F40:0
2. Write 0 to F40:1
3. Write 8 to F40:2

With Modbus/RTU or Modbus/TCP Addresses:

From the RMC150 Register Map - File 40 Commands, we see that the addresses for the Axis 0 command registers start at register offset 20481. The offsets must be prefixed with a 4. To issue this command to the RMC75 via Modbus/RTU or Modbus/TCP, do the following:

4. Write 90 to 420481
5. Write 0 to 420483
6. Write 8 to 420485

2. RMC200: Start User Program 2

To start user program 2, the user decides to issue the following command to Axis 0 of the RMC200:

Start Task (90):

2.

- Command Parameter 1: Task = 0
- Command Parameter 2: Program = 8

With Allen-Bradley DF1 Addresses:

From the DF1 Addressing topic, we see that the RMC200 addresses for the Axis 0 command registers are F12:0-9. To issue this command to the RMC200 via DF1, CSP, or EtherNet/IP, do the following:

2.

1. Write 90 to F12:0
2. Write 0 to F12:1
3. Write 8 to F12:2

With Modbus/RTU or Modbus/TCP Addresses:

From the Modbus Addressing, we see that the RMC200 addresses for the Axis 0 command registers start at register offset 20481. The offsets must be prefixed with a 4. To issue this command to the RMC75 via Modbus/RTU or Modbus/TCP, do the following:

4. Write 90 to 42049
5. Write 0 to 42051
6. Write 8 to 42053

Command Format

Each RMC command consists of a command number and command parameters.

- **Command Number**
  Each RMC command has a number associated with it. You must use this number when you issue a command to the RMC. The number is typically included in parentheses whenever the command is mentioned. For example, the Move Absolute Command (20) has a number of 20.
- **Command Parameters**
  Some commands have command parameters. For example, the Move Absolute Command (20) has 5 command parameters: Position, Velocity, Accel rate, Decel Rate, and Direction. When you issue a command, you must include any command parameters.

Notice that the Command Number and Command Parameters are all floating point numbers, as seen from the PLC or HMI issuing the command. However, the command number must be a whole number (e.g. 20, or 20.0, *not* 20.1*)*), and certain command parameters must be integers or are limited to a certain range, depending on the command.

For more details on commands, see the Commands Overview topic.

Advanced Details

Immediate Commands

Certain commands in the RMC are *immediate* commands. There is no limit to the number of immediate commands that can be issued to an axis per loop time, whereas a maximum of one non-immediate command per loop time can be issued to each axis.

Immediate commands are usually not motion commands. The ability to issue multiple immediate commands in one loop time affects primarily the user programs, since it is difficult to issue many simultaneous commands via the external communications.

Commands Issued Simultaneously

Commands sent simultaneously to the RMC in the same communication block will always be issued simultaneously. For all communications methods, the RMC waits until the entire communication block is received before issuing any of the commands in the block. The commands received in this block are referred to as a *command set*. Such a command set can contain at most one command per axis, with the exception of immediate commands in user programs, which are unlimited.

Command Queue

As command sets are received from various sources, they are placed in a command queue. Each control loop, the controller processes command sets from the queue on a first-in first-out basis, stopping when it reaches a command set that would issue a non-immediate command to an axis that has already processed a non-immediate command this control loop. Command sets remaining in the command queue will be processed in subsequent control loops. Notice that commands issued as part of the same command set will never be split between different control loops. That is, either all of the commands in the command set will be processed in the control loop, or none of them will be and the entire command set will be deferred to the next control loop.

The command queue has a finite length. The RMC75 command queue allows up to 6 command sets, each set with up to one command per axis. The RMC150 allows up to 12 command sets. If the command queue is full when a new command set is received, the command set will be discarded, and Runtime Error 201 (Command block dropped) will be logged in the Event Log. This condition can only practically occur in cases where non-immediate commands are being issued continuously from a user program. Notice that Expression commands are immediate commands and are therefore safe to issue continuously from user programs.

See Also

Command Request and Acknowledge Bits | Communications Overview

---

Show All

5.3. Tasks

Tasks are for running User Programs. Each task is an execution engine that can run one User Program at a time. Each RMC has a maximum number of tasks.

	RMC75	RMC150	RMC200 CPU20L	RMC200 CPU40
Maximum # Tasks	4	10	32	64

The RMC can run User Programs simultaneously by running them on different tasks. For example, the RMC200 CPU20L can run up to 32 User Programs simultaneously. Use the Task Monitor to monitor

what each task is doing. To change the number of tasks, see the **Changing the Number of Tasks** below.

To start running a User Program on a task, use the Start Task (90) command.

Example

You have created a User Program called MyProgram. You want to run MyProgram. To do this, you need to run it on a task. Use the Start Task (90) command to start running MyProgram on any task, for example task 0.

Example

You have created two User Programs called MyProgram1 and MyProgram2. You want to run MyProgram1 and MyProgram2 simultaneously. To do this:

1. Use the Start Task (90) command to start running MyProgram1 on any task, for example task 0.
2. Since task 0 is already running, you must start MyProgram2 on another task, for example on task 1. If you were to start MyProgram2 on task 0, then task 0 would immediately stop running MyProgram1, and then start running MyProgram2. Then they would not be running simultaneously.

Note:
In order to run tasks or the Program Triggers, the RMC must be in RUN mode. See the RUN/PROGRAM Mode topic for details.

Starting Tasks

Tasks can be started in several ways:

- **Start Task (90) command**
  Send a command to specify which user program to run on a certain task.
- **Program Trigger**
  Automatically start a task based on a user-defined condition. See the Program Triggers topic for details.
- **Task Monitor in RMCTools**
  In the Task Monitor, right-click a task and click **Start Task**.
- **Project Pane**
  In the Project pane, in the User Programs node, right-click a user program, choose **Run Program**, and click the task to run the program on.

Note:
If a user program is started on a task that is already running, the task will stop the user program it is already running and immediately start at the User Program you specify.

Stopping Tasks

Tasks can stop in several ways:

- **End Link Type**
  If a User Program is done running, the task it is running on will also stop. A User Program is done running when it encounters an End link type.
- **Stop Task (91) command**
  The Stop Task (91) command immediately stops the specified task. The User Program currently running on the task will immediately stop.
- **Program Triggers**
  A trigger in the Program Triggers can stop a task. See the Program Triggers topic for details.
- **Axis Halt**
  If a halt occurs on any axis, the RMC immediately stops all tasks by default. This default setting can be changed on the Programming Properties dialog.

- **Program Mode**
  When the RMC enters Program mode, all task will stop.
- **Task Fault**
  A task will be halted if any of the following runtime errors occur:
  - Command overwritten within a Task step for Axis n.
  - Array Index out of range.
  - Internal Interpreter Error.
  - Task not allowed to run this program.
  - Unsupported address range in COPY function.
  - Divide by zero.
  - Undefined numerical operation.
- **Task Monitor**
  In the Task Monitor, right-click a task and click **Stop Task**.

Task Monitor

Use the Task Monitor to see which tasks are currently running User Programs. The Task Monitor shows each task, its state, and which program and step it is currently on or the last step it was on. See the Task Monitor topic for details.

Changing the Number of Tasks

By default, the RMC has 2 tasks.

To increase the number of tasks:

- In the Project pane, right-click **Programming** and click **Properties**.
- On the **General** page, select a number in the **Number of User Tasks** box and click **OK**.
- Apply the changes to the RMC by right-clicking **Programming** and clicking **Download Programs to Controller**.

Assigning User Programs to Tasks

By default, a user program is allowed to run on only one task at a time. This setting is usually sufficient for most applications. You can change this to specify a certain task that the user program is allowed to run on, or allow the user program to run on any number of tasks simultaneously. Restricting the tasks that a program can run on will help detect program errors and help reduce the maximum programming execution time.

To change this setting, in the Project pane, right-click the user program, choose **Properties**, and choose the **Tasks** page.

Default Axis

Each task has a **Default Axis** associated with it. The Default Axis is used in a User Program when a command's Commanded Axes field is set to "Default Axis" or when an expression in a user program contains an axis register, but does not specify the axis (for example, _Axis[].StatusBits.InPos).

When a task is started using a Start Task (90) command, the Default Axis for the task will be automatically set to the axis that received the Start Task command.

When a task is started from the Program Triggers, the Default Axis for the task is left unchanged. On power-up, the Default Axis for task #*n* is Axis *n*, up to the number of available axes. For any additional tasks, the Default Axis is Axis 0 on power-up. However, the Default Axis for a task can change through Start Task commands (see above) or the Current Axis register (see below).

The Default Axis for a task can be changed at any time by changing the Current Axis register for the task (see _CurAxis below). For example, to restore the Default Axis for a task to its power-on default axes on an 8-axis controller, you could use the following expression:

```
_CurAxis := SEL(_CurTask < 8, _CurTask, 0);
```

The Default Axis is typically used only in advanced applications and is not recommended for new users.

Why Bother?

Consider a multi-axis application where each axis runs the exact same sequence. Instead of creating one user program for each axis, you can create a single user program that issues commands to the Default Axis, and does not specify an axis in the link condition. Simply issue the Start Task commands to each axis. The user program can run independently on separate tasks simultaneously.

Using the Task Tags in Expressions

This section is for advanced users only.

The _Task[] tag has several members that can be used to manage tasks with expressions. The "_Task[]" tag can be used without any value in the brackets to reference the task that is running that user program.

_Task[].CurAxis

_Task[].CurProg

_Task[].CurStep

_Task[].Status

_Task[].CurProgStep

Using the Default Axis

In expressions in the user programs, the "_Axis[]" tag can be used without any value in the brackets to reference the default axis.

Changing the Default Axis

To change the default task, you can use the _Task[].CurAxis tag. The tag _CurAxis is equivalent to _Task[].CurAxis, which references the current axis of the current task.

Example 1

"_Task[0].CurAxis := 2" will change the default axis of the task zero to Axis 2.

Example 2

"_Task[].CurAxis := 0" will change the default axis of the current task to Axis 0. The current task is whatever task is running the user program.

See Also

User Program Overview

5.4. Variables

This topic is about the Variable Table variables. For details on local variables in a user program step, refer to Local Variables in Expressions. For details on local variables in a user functions, refer to Declaring Variables in User Functions.

	Number of Variables
RMC75S	1024
RMC75P	1024
RMC75E	1024
RMC150E	1024
RMC200	4096

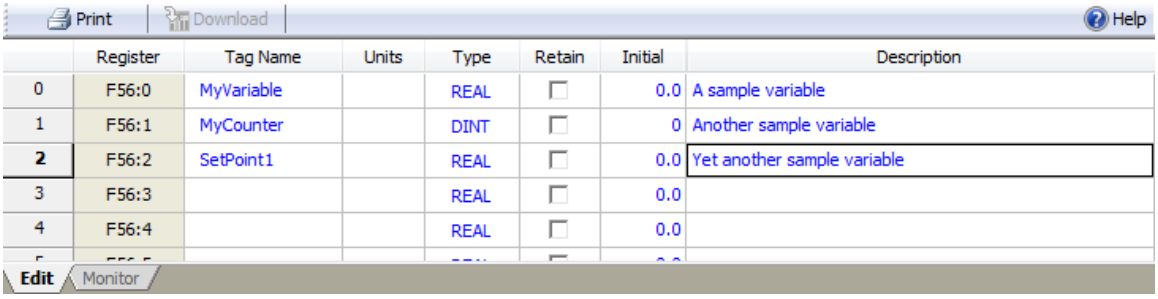
Variables may be named by the user, and can be set to any value at any time. Variables make the User Programs very flexible. They can be used to effortlessly change programs, make programs readable, and easily influence User Programs via a PLC. Variables may be used in command parameters, the Expression (113) command, and several Link Types.

A limited number of variables can be individually selected to be retentive, if supported by the RMC. The Current Value of retentive variables will be retained between power cycles without requiring a Flash update. This is useful for retaining data such as setpoint positions, machine cycle counters, and recipe data.

PROFIBUS uses some of the variables for communications. PROFINET and EtherNet/IP may also use some variables if configured to do so. Therefore, before using any variables, you may first wish to determine which variables you will be using for the communciations to avoid having to move variables later.

The Variable data types can be REAL, DINT, or DWORD. See the Boolean variables topic for special instructions regarding how to create boolean variables. Arrays with indexed addressing are also supported.

Example Variable Table:



The **Edit** tab is for creating variables. The **Monitor** tab is for monitoring and changing the current value in real-time.

Column Details

Register	The address of the current value of the variable. This address is used for accessing the current value of the variable externally, such as from a PLC. For the address of the initial value, see the Register Map topic.
Tag Name	The name of the variable. Use the Variable name to refer to the variable. You can also reference the variable by its register address, especially if communicating to the RMC with a PLC. Tag names are limited to 64 characters.
Units	The units of a variable is purely for the user's own reference. It has no affect on the usage of the variable.
Type	This can be any of the RMC data types, or an array of variables with the same data type. See the Boolean variables topic for special instructions regarding how to create boolean variables.
Retain	The current value of retained, or retentive, variables will automatically be saved in non-volatile memory. A variable can be set to be retained by checking the **Retain** cell in the Variable Table Editor. Not all RMCs support retained variables, nor can all variables be retained. See **Retained Variables** below for more details.
Initial	You can specify an initial value for a variable in the Variable Table Editor. The Current Value of a variable will be set to the Initial Value when:

	• The RMC starts up. If the variable is marked as retentive, the Current Value will then be set to the retained value. • The Initial Value is changed. The Initial Value can be saved to Flash. See the Register Map topic to find the addresses of the Initial Values.
Current	The current value of the variable is displayed only on the **Monitor** tab. There are several ways to change the current value: • On the **Monitor** tab • Using the Expression (113) command • Using a PLC or other host controller See the Register Map topic to find the addresses of the Current Values.
Description	The description is for the user's own reference.

Creating Variables

To define variables and set the initial value, use the **Edit** tab of the Variable Table Editor. All 1024 variables already exist and do not need to be declared or created. However, it is good practice to give a Tag Name, Description, and Initial Value to any variable before you use it.

After making changes on the Edit tab, you must download the Programming node to apply the changes to the RMC.

See the Boolean variables topic for special instructions regarding how to create boolean variables.

Monitoring Variables

Use the **Monitor** tab of the Variable Table Editor to monitor or change the current value of the variable in real time from RMCTools.

To change a current value:

- On the **Monitor** tab, edit the Current Value of the desired variable(s).
- In the toolbar, click the **Download Current Values**  button to apply the changes to the controller.

Assigning a Value to a Variable

There are several ways to assign a value to a variable:

- Use the Edit tab of the Variable Table Editor to set the initial value. The Initial value is the value the variable will take on when the RMC is reset.
- Use the Monitor tab of the Variable Table Editor to set the current value of the variable.
- Use the Expression (113) command in a User Program to assign a value to a variable.
- Write to the Variable register via a PLC or other host controller. See **Variable Locations - Using Variables with a PLC or other Host Controller** below for details.

Using Variables

Variables can be used for many purposes:

- Store a value for later use.
- To influence User Programs.
- Count how many times an event occurs.

To use the values assigned to variables, do the following:

- Enter variables in the Command parameters in User Programs.
- Use variables in Link Type expressions in Use Programs.
- Use variables in the Expression (113) command in User Programs.
- Use variables in the Program Triggers.

Examples:

1. If you create a variable called MySpeed, and enter it as the Speed command parameter in several Move commands in a User Program, then you can easily change the speed for all those commands at once by simply changing the value of the Speed variable.
2. To count how many times an event occurs, you can add 1 to a variable each time the event occurs.

Variable Locations—Using Variables with a PLC or other Host Controller

You can read or write to variables when communicating to the RMC with a PLC or other host controller. See the Register Map topic to find the addresses of the variables. Notice that each variable has two locations—one for the Current Value, and one for the Initial Value.

Retained Variables

Retained means that the Current Value of the variable is automatically saved, or *retained*, in non-volatile memory and will be preserved even when power is cycled to the RMC. Retained variables are saved to non-volatile memory approximately every 100 msec.

Any variable can be set to be retained by checking the **Retain** cell in the Variable Table Editor.

Variables that are not retained will not retain the Current Value. When power is cycled to the RMC, the Current Value of non-retained variables will be set to the Initial Value.

Support for Retentive Variables

	Maximum Retained Variables	Board revisions	Firmware version
RMC75S	none	-	-
RMC75P	none	-	-
RMC75E	986	1.4A and newer	3.30.0 and newer
RMC150E	986	all	3.30.0 and newer
RMC200	2048	all	all

Boot-up Details

When the RMC boots up, the Current Values for all variables will start at zero, and then if a variable has an Initial Value stored in Flash, the Current Value will be overwritten with this Initial Value. Further, if the variable is marked as retentive and was retained to non-volatile memory, the Current Value will be set to the retained value. This is all done before the first loop time of the RMC. Therefore, the precedence on boot-up is (1) NVRAM, (2) Flash, (3) zero.

This precedence order means that a project can communicate what the initial default values should be for each variable, even retained variables. For example, a sample project with retentive variables will be able to specify what the values should be initially, even if they are retained. In addition to sample projects, this also allows a project to be copied to a new machine with meaningful initial values.

See Also

Programming Overview | Arrays | Data Types

5.5. Program Triggers

To access the Program Triggers: In the Project pane, expand **Programming**, and double-click **Program Triggers**.

The Program Triggers start user programs when user-specified events occur. For example, you can set up the Program Triggers to start a user program when an input turns on, or to start a user program when a variable becomes a certain value. This is a good way to start RMC user programs from a PLC or host controller.

For a real programming example of using the Program Triggers, see the Example: Jogging an Axis topic.

Triggers

The Program Triggers can contain the following number of **Triggers:**

	Number of Triggers
RMC75S	64
RMC75P	64
RMC75E	64
RMC150E	64
RMC200	128

A trigger is one complete row in the table, consisting of a **Condition** and **Task Actions**. The RMC checks all the conditions every loop time. When a condition becomes true, the user program(s) specified by the user are started on the specified tasks. The user programs are started only on the rising edge of the entire condition becoming true (one-shot). The user program will be started on that task even if the task is already running a user program. The task will stop running the current user program an run the new one.

The Task Actions are performed only on the rising edge of the entire condition. Therefore, when the entire condition becomes true, the specified Task Actions will be performed. While the condition remains true, the trigger will not start or stop anything.

Important!
The Task Actions are performed ***only*** on the rising edge of the ***entire condition***. Therefore, when the entire condition ***becomes*** true, the specified Task Actions will be performed ***once***. While the condition ***remains*** true, the trigger will ***not*** start or stop anything.

Conditions

The Program Triggers can handle simple or complex conditions. For example, a condition can check for certain inputs to be on *and* the Actual position to be greater than a certain value. Conditions are created using expressions and are therefore very flexible. For details on expressions, see the Expressions topic. To find out how to create a condition, see **Creating a Trigger** below.

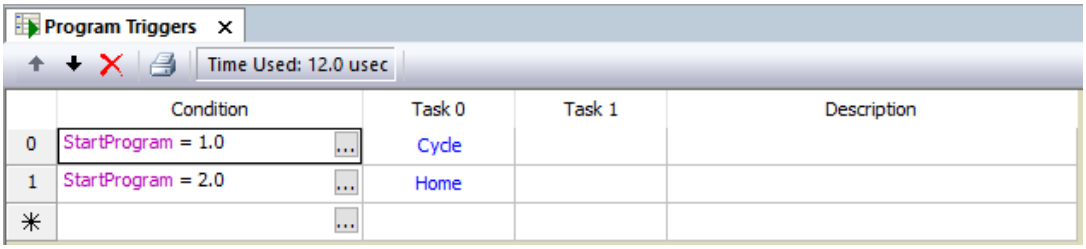
Task Actions

Task Actions can start or stop Tasks. Tasks run User Programs. For each trigger, you can define which Tasks to start, stop, or do nothing to. The Program Triggers will allow a Task to be started only at a label in a User Program. To find out how to set up the Task Action, see **Creating a Trigger** below.

The Program Triggers has one column for each Task. To increase the number of tasks, use the **General** page of the Programming Properties dialog.

Example

This is an example of two triggers in the Program Triggers:



The first trigger will cause the user Program "Cycle" to be started on Task 0 when the variable StartProgram becomes 1, and the second trigger will cause the user Program "MoveHome" to be started on Task 0 when the variable StartProgram becomes 2. As illustrated here, writing to a variable that triggers a user program is an easy way for a PLC to start a user program in the RMC.
Notice that the user program will be started when the condition *becomes* true. It will not restart while the condition remains true.

Order of Execution

If there are multiple triggers (rows), they will be executed in order from top to bottom. Therefore, if two triggers become true in the same loop time, and they start different user programs on the same task, the user program specified by second of these two triggers will end up running on the task. This is because the RMC will always do the last thing it is instructed to do.

Typically, in well-structured programs, this type of conflict does not occur.

How to Create and Manage Triggers

Create a Trigger

1. In the last row of the Program Triggers, click the **Condition** cell, then click the ellipsis button ... . The New Condition Wizard will open.
2. Select the type of condition you want to create, and continue through the wizard to complete the condition.

> **Note:**
> If you want to create a complex condition, double-click the **Condition** cell. It will let you create a custom condition. See the Expressions topic for details on creating expressions.

3. In each **Task** cell for the condition you just entered, click the cell and select one of the following options from the drop-down box:
    1.
    - **User Program**
      Choose a user program from the drop-down box.
    - **No Entry**
      If you do not want the trigger to affect the Task, do not enter anything in the cell. To remove an entry, click the cell and press Delete.
    - **<StopTask>**
      The Task will be stopped immediately when the condition becomes true.

> **Note:**
> The Program Triggers Editor has one column for each Task. To increase the number of tasks, use the **General** page of the Programming Properties dialog.

Move Triggers

To move rows up or down, select one or more rows, then use the **Move Up** and **Move Down** buttons. To delete rows, select one or more rows, then click the **Delete Row** button.
The order of the triggers can be important. If several conditions become true at the same time and multiple actions are triggered on the same Task, the last action for that Task will be the one that runs.

Download the Program Triggers

In the Project Pane, right-click **Programming** and click **Download Programs**.

After the Program Triggers are downloaded, the RMC must be put into RUN mode before the Task Actions will be performed whenever when the associated conditions becomes true.

Print the Program Triggers

To print the Program Triggers, on the Program Triggers Editor toolbar, click the **Print** button. Or, on the **File** menu, choose **Print**.

Special Case: When the RMC Enters RUN Mode

With older firmware, the Program Triggers will trigger when a condition was true when the RMC enters Run mode. Newer firmware has a selection for this behavior, and defaults to triggering only when a rising edge of the condition occurs, and therefore, will not trigger simply because a condition happens to be true when the RMC enters Run mode. RMC75/150 firmware versions 3.64.0 and and later and RMC200 firmware 1.01.0 and later support the selection.

The Program Trigger behavior can be selected in the Programming Properties dialog. The following two options are available:

- **Trigger on rising edge of condition only** will start the specified action only when the condition experiences a rising edge, which means the condition changes from false to true. This is the recommended option.
- **Trigger on rising edge of condition OR when the condition is true upon entering Run mode** will start the specified action when the condition changes from false to true. It will also start the specified action if the condition happens to be true when the RMC enters Run mode, even though there was not a specific rising edge of the condition at that time.
  This option is not recommended, but exists to be compatible with the functionality of older firmware.

Legacy Behavior

With older firmware, when the RMC transitions from PROGRAM mode to RUN mode, it assumes all the Program Trigger conditions were previously false. Therefore, when the RMC enters RUN mode, all Program Trigger conditions that evaluate to true will run the corresponding Task actions, because the RMC thinks their states just transitioned from true to false.

Example

Consider a trigger with the condition **MyInput = False**. If MyInput is off when the RMC enters RUN mode, the Task action for that loop time will run.

If several conditions become true at the same time and multiple actions are triggered on the same Task, the last action for that Task will be the one that runs.

If your Program Triggers has any conditions that may be true when the RMC enters RUN mode, but you do not wish those Task actions to run when you enter RUN mode, you can do the following:

1. Use the **FirstScan** bit (see below) in the condition in the last line of the Program Triggers.
2. For each task that you do not wish to start when entering RUN mode, choose **<StopTask>**. Alternatively, if you have a user program that you want to run when the RMC enters RUN Mode, you can enter that user program for the desired Task.

The method above works because the FirstScan condition will become true at the same time as the other items when you enter RUN mode, and since it is the last item in the Program Triggers, it will override the other items.

First Scan Bit

The _FirstScan bit is true during the first loop time after the RMC enters RUN Mode. The _FirstScan bit is false otherwise. This bit can be used in a trigger condition to start programs when the RMC starts up. In order to do so, you must configure the RMC to start in RUN mode.

The _FirstScan bit is a Controller Status Bit.

To use the FirstScan bit:

1. In the last row of the Program Triggers, double-click in the **Condition** cell. The Expression Editor will open below the Condition cell.

2. In the **Tags** list, expand **Controller** and double-click **FirstScan** so that it appears in the box at the top.
3. Click **OK**.
4. In the **Task** columns, choose the user program you wish to run. You must have created a user program first.
5. To apply the changes to the RMC, right-click **Programming** and click **Download Programs**.

Starting a User Program when the RMC Powers Up

To automatically start a user program when the RMC powers up:

1. Create a user program that contains the commands you would like to run when the RMC powers up.
2. In the Program Triggers, create a condition using the _FirstScan bit as described above. The condition should be "_FirstScan".
3. For the condition you entered in the Program Triggers, choose the user program on a task.
4. On the **RUN/PROGRAM** page of the Program Properties dialog, set the RMC to start up in RUN Mode.
5. Update Flash.
6. Cycle power to the controller. You can view the Event Log Monitor to see if the user program did start immediately after the RMC powered up.

See the Example: Closed Loop Motion on Startup for a detailed walk-through example.

Running a User Program on Startup Only

If you wish to have the Program Triggers start a user program only when the RMC starts up, and not each time it enters RUN mode thereafter (such as after you download the Programming node), do the following:

1. Create a variable called **Running**, and set the Default Value to 0.
2. Create the following Program Triggers condition:
   _FirstScan and Running = 0
3. In the first step of the user program, add an Expression (113) command with the following expression:
   Running := 1

When the RMC starts up in RUN mode, it will run the user program. However, it if exits and enters RUN mode after that, the user program will not start because the variable **Running** is not zero.

Disabling the Program Triggers Task

Sometimes when troubleshooting or setting up the RMC, you may wish to be in RUN Mode but have the Program Triggers disabled.

To disable the Program Triggers:

1. In the **Project** pane, expand the desired controller.
2. Right-click **Programming** and click **Properties**.
3. On the **General** page, in the **Program Triggers** section, set the Program Triggers to **Disabled**. Click **OK**.
4. In the **Project** pane, right-click **Programming** and click **Download Programs** to apply the changes to the RMC.

> **Note:**
> The Program Triggers will never run when the RMC is in PROGRAM Mode.

See Also

Programming Overview | User Program Overview

5.6. Downloading the Programming

After changing any items under the **Programming** node in the Project Pane, you must download the programs to the RMC to apply the changes to the RMC.

To download the programs:

1. In the Project Pane, right-click **Programming** and click **Download Programs**.

See Also

User Program Overview

5.7. Tag Names

Each register in the RMC is considered a **tag**. In RMCTools, each tag has a **tag name**. Tag names can only be used for programming in the RMCTools. Most tags have a fixed name, also referred to as **system tags**. Some tag names can be defined by the user, such as the variables tags and discrete I/O.

Use tags names instead of register addresses to make your user programs more readable. For example, the Axis 0 Actual Position tag name is **_Axis[0].ActPos**, which is much more readable than its IEC address **%MD8.8**.

The tag names for each register are given in the Register Description topics. Use the Registers Maps to find the description topic for the register you need. The tag name details will be given there.

When using tag names in expressions in RMCTools, the Expression Editor includes a Tags box from which you can choose the register you need and double-click to insert the tag name in the expression. Therefore, you do not need to memorize or look up tag names.

Using Tag Names

In general tag names can be used wherever an expression can be entered in RMCTools. Tags names can be used in the following places in RMCTools:

- **In User Program command parameter boxes**

  Example 1: Consider in a User Program, a Move Absolute command is issued to Axis 0. In the Position parameter of the Move Absolute command, entering the tag name **_Axis[1].ActPos** would make Axis 0 move to the position of Axis 1.

  Example 2: Assume a variable is defined with a tag name **MySpeed**. In a User Program, in the Speed parameter of a Move Absolute command, entering the tag name **MySPeed** would make the move at the speed defined by the variable **MySpeed**.

- **In the Expression (113) command**

  Example: The assignment expression **_Axis[0].OutputBias := 5.0** sets the Axis 0 Output Bias to 5.0.

- **In the Wait For, Conditional Jump, and Delay link types**

  Example: The logical expression **_Axis[0].TarPos > _Axis[1].ActPos** in the Wait For Link Type compares the Axis 0 Target Position with the Axis 1 Actual Position.

Tag Names and Bits

Individual bits in any DWORD register can be accessed by appending a period and the bit name or bit number after the tag name.

Examples:

_Axis[0].StatusBits.InPos accesses the Axis 0 In Position bit.

_Task[2].Status.Running accesses the bit that indicates whether task 2 is running.

MyDWORD.MyBit accesses the bit named MyBit in the variable named MyDWORD.

MyDWORD.4 accesses bit 4 in the variable named MyDWORD.

Entering Tag names

You do not have to remember the tag names in order to use them. The Address Selection Tool can be used to find tag names.

To access the Address Selection Tool from a User Program command parameter box, or from a link type, click the ellipsis button ().

In the Expression Editor, the **Tags** box lists all the tags.

Special Tag Names

Controller

The following controller tag names are useful for certain advanced programming applications.

Tag Name	Description
_FirstScan	This BOOL value goes high (1) on first scan after entering RUN Mode, otherwise it is low (0). See the Program Triggers topic for usage details.
_SysMS	System Milliseconds. Contains the number of milliseconds since the RMC powered up. This DINT value will wrap around to -2147483648 after it reaches its maximum value of 2147483647.
_SysTicks	System Ticks. Contains the number of control loops since the RMC powered up. This DINT value will wrap around to -2147483648 after it reaches its maximum value of 2147483647.

Tasks

The following Task tag names are useful for certain advanced programming applications. See the Tasks topic for more details.

Tag Name	Description
_CurTask	The current task that is running the user program.
_CurAxis	The current axis for the task that is running the user program.

Variables

Variables can be given a user-defined tag name in the Variable Table Editor, which is the preferred method of referencing variables. If they are not given a user-defined tag name, they can still be accessed with the default tag names:

Tag Name	Description
_VarTbl.CurVal[n]	Current Value of variable n.
_VarTbl.Initial[n]	Initial value of variable n.

Tag Name List - RMC75/150

Tag Name List - RMC200

See Also
Registers Maps

5.8. Program Capacity and Time Usage

The RMC has a fixed amount of memory for user programs, and a fixed amount of time allocated for processing user programs. When verifying or downloading the programming, RMCTools calculates the required memory size and time usage and displays them in the Verify Results Window.

Exceeding Program Memory Size or Time Usage

Program Memory Size

If the memory size limit is exceeded, an error is logged in the Verify Results window, and RMCTools will not allow the programs to be downloaded to the controller. See **Reducing the Program Size** below for tips on how to fit your programming into the memory.

Program Time Limit

If the time limit is exceeded, an error is logged in the Verify Results window, and RMCTools will not allow the programs to be downloaded to the controller. See **Reducing the Programming Time Usage** below for tips on how to reduce the required programming time. You can also choose to override the time limits by enabling **High Control Loop Utilization** in the Control Loop Page.

Reducing the Program Size

The RMCs have the following amount of memory allocated for the entire Programming node which includes the user programs, Program Triggers, and plot templates. Notice that for RMC75/150 controllers, the Tuning Wizard borrows about 8KB when it needs to run. Therefore, if a controller has less than 8KB of free program memory, the Tuning Wizard may not be able to download its program. On RMC200 controllers, the Tuning Wizard does not affect the program memory.

CPU	Firmware	Program Memory Size	Approximate Capacity in Steps
RMC75E (1.1G or newer)	3.40.0+	496KB	1000-2000
	3.32.0-3.39.4	240KB	500-1000
	2.03-3.31.3	112KB	300-500
	1.00-2.02	64KB	150-250
RMC75E (1.1F or older)	3.32.0+	240KB	500-1000
	2.03-3.31.3	112KB	300-500
	1.00-2.02	64KB	150-250
RMC75S	All	64KB	150-250

RMC75P	All	64KB	150-250
RMC150E	3.40.0+	496KB	1000-2000
	Pre-3.40.0	240KB	500-1000
RMC200	All	1024 KB	2000-4000

If the user programs exceed the amount of memory available, it means that you simply have too much programming. You may be able to slightly reduce the size by disabling expression logging as described below, but you may need to reduce the amount of programming.

Tips for Reducing the Programming Size

If you have disabled expression logging and your program image is still too large, you must reduce the amount of programming. Here are some tips:

- **Offload to the PLC**
  If your machine has a master controller, such as a PLC, offload some of the programming to it.
- **Use arrays**
  If you are performing many consecutive similar steps, consider using arrays and loops instead.
- **Disable expression assignment Logging**
  By default, the RMC logs the results of assignment expressions and COPY statements in the Expression (113) command. If your programs use this command, you may be able to decrease the program size by disabling expression logging. You can still log the results of certain calculations by using the LOG_EVENT function.
  To disable the expression logging, in the Programming Properties dialog, on the **Verify** page, uncheck **Log Expression Statements.**

Reducing the Programming Time Usage

RMCTools calculates the worst-case scenario of how much time of any control loop may be taken up by program calculations, which includes the user programs and the Program Triggers. Each task can process one step per loop, and conceivably, a step can be run on more than one task simultaneously. RMCTools finds the steps that take the longest to process, and multiplies that time by the number of tasks the step could possibly run on. The time allocated for the user programs varies with RMC as described in the **Time Allocation** section below.

When verifying programs, you can view the execution times that RMCTools calculates for the user programs. This can help you determine what steps are causing problems:

1. On the **Tools** menu, choose **Options**.
2. On the **Programming** page, in the **Verify Results** section, uncheck **Automatically close Verify Results...**
3. Click **OK**.
4. On the **Programming** menu, choose **Verify Programs**. If no errors exist, the Output pane will list the maximum estimated loop time that the user programs and Program Triggers will take, and break it down into each user program step.

There are several ways of reducing the programming time usage:

- **Decrease the number of tasks allocated**
  This may significantly reduce the worst-case calculations of the time usage.
- **Assign user programs to tasks**
  By assigning user programs to only run on certain tasks, or only one task at a time, you can significantly reduce the worst-case time usage. To assign user programs to tasks, use the **Tasks** page on the User Program Properties dialog.
- **Disable expression assignment logging**
  This is described in the **Reducing the Program Size** section above. If your programs use the Expression (113) command heavily, this may yield a large improvement. Notice that this will reduce both memory size and time usage.

- **Disable Immediate Command logging**
  This may significantly reduce the time if your programs use many immediate commands. To disable immediate command logging, in the Programming Properties dialog, on the **Verify** page, uncheck **Log Immediate Commands.** If this option is disabled, immediate commands issued from user programs will not be reported in the Event Log.
- **Break down complicated steps into multiple smaller steps**
  When verifying the user programs, the Verify Results window can provide timing details on the longest steps. Break these steps into smaller steps.
- **Increase the loop time**
  Increasing the Loop Time on the Control Loop Page will allocate more time for the user programs.
- **Simplify the programming**
  In general, the simpler the programming, the less time it takes. Simplifying the programming may include such things as revisiting your overall programming approach, or offloading some of the programming to the machine's master controller, such as a PLC.

Time Allocation

The time allocated for the user programs varies with RMC and the selected loop time. The time allocated is displayed in the Verify Results window after verifying the programming.

		Loop Time					
CPU	**Firmware**	**4000 µs**	**2000 µs**	**1000 µs**	**500 µs**	**250 µs**	**125 µs**
RMC75E	All	3000 µs	1200 µs	600 µs	210 µs	75 µs	—
RMC75S	Pre-3.31.0	3000 µs	1000 µs	500 µs	210 µs	—	—
RMC75S	3.31.0+	3000 µs	1000 µs	500 µs or 250 µs (note 1)	210 µs or 25 µs (note 2)	—	—
RMC75P	Pre-3.31.0	3000 µs	1000 µs	500 µs	210 µs	—	—
RMC75P (2.1F or newer)	3.31.0+	3000 µs	1000 µs	500 µs or 250 µs (note 1)	210 µs or 25 µs (note 2)	—	—
RMC75P (2.1E or older)	3.31.0+	3000 µs or 2500 µs (note 3)	1000 µs or 600 µs (note 4)	250 µs or 25 µs (note 5)	25 µs	—	—
RMC150E	All	3200 µs	1200 µs	500 µs or 400 µs (note 6)	200 µs	75 µs	—
RMC200 CPU40	All	3200 µs	1200 µs	500 µs or 400 µs or 300 µs (note 7)	200 µs or 150 µs (note 8)	75 µs	50 µs
RMC200 CPU20L	All	3200 µs	1200 µs	500 µs	200 µs or 150 µs (note 9)	75 µs	50 µs

Note 1: If the RMC75S/RMC75P has 1 control axis with up to 3 reference axes, or 2 control axes with no reference axes, 500 µs is allocated. Otherwise, 250 µs is allocated.

Note 2: If the RMC75S/RMC75P has 1 control axis and up to 1 reference axis, 210 µs is allocated. Otherwise, 25 µs is allocated.

Note 3: If the RMC75P has 1 control axis with up to 1 reference axis, 3000 µs is allocated. Otherwise, 2500 µs is allocated.

Note 4: If the RMC75P has 1 control axis with up to 1 reference axis, 1000 µs is allocated. Otherwise, 600 µs is allocated.

Note 5: If the RMC75P has 1 control axis and up to 1 reference axis, 250 µs is allocated. Otherwise, 25 µs is allocated.

Note 6: If the RMC150 has 3 or fewer axes with any number of reference axes, or 4 control axes with up to 2 reference axes, 500 µs is allocated. Otherwise 400 µs is allocated.

Note 7: The allocated user program is dependent on total estimated axis processing time assuming full load:

Total Axis Processing Time	Allocated User Program Time
0 - 294 µs (approx. 0-28 control axes)	500 µs
295 - 394 µs (approx. 29-38 control axes)	400 µs
> 394 µs (approx. 39 or more control axes)	300 µs

See the Loop Time topic for details on how the total axis processing time is determined.

Note 8: The allocated user program is dependent on total estimated axis processing time assuming full load:

Total Axis Processing Time	Allocated User Program Time
0 - 94 µs (approx. 0-9 control axes)	200 µs
> 94 µs (approx. 10 or more control axes)	150 µs

See the Loop Time topic for details on how the total axis processing time is determined.

Note 9: The allocated user program is dependent on total estimated axis processing time assuming full load:

Total Axis Processing Time	Allocated User Program Time
0 - 86 µs (approx. 0-7 control axes)	200 µs
> 86 µs (approx. 8 or more control axes)	150 µs

See the Loop Time topic for details on how the total axis processing time is determined.

See Also

Verifying | Programming Overview | Plot Capacities | Curve Storage Capacity

5.9. Programming Security

Programming Security is intended to be a means of password-protecting your intellectual property. If you wish to prevent others from accessing the user programs you write, you can enable the Programming Security feature.

Delta reserves the right to disable or bypass security features enabled by the copyright holder of user programming in RMC series controllers. Delta's responsive customer support is an important service that benefits the customers. In order to ensure responsive service, Delta reserves the right to disable or bypass any and all security features applied to the user programming downloaded to an RMC series controller. For details on when and how Delta will do this, refer to the RMCTools Security Policy and Agreement.

The Programming Security applies to the items in the **Programming** node of the Project Pane. The protection can be applied to an online or offline project. When password-protected programming is downloaded to the RMC, it will be protected in the RMC also. This means that when a password-protected project is uploaded from the RMC, the programming will be protected as specified in the original project.

RMCTools offers three levels of User Source Code Security:

1. **Copyright Notice**
   This level will apply a copyright notice to the Programming. The contents of the Programming node will still be accessible and editable, but the copyright notice will appear to the user. Copying and exporting of user programs will be disabled. By entering the password, the user can disable the copyright notice.
2. **Protect Programming Source Code**
   This level will password-protect the contents of the Programming node in the project. The programming contents can only be viewed or edited if the user enters the correct password. The programming contents can be uploaded to an RMCTools project, but cannot be viewed or edited unless the correct password is entered.
3. **Copy Protection**
   In addition to password-protecting the programming, this level will prevent the programming from being uploaded to an RMCTools project, thereby preventing the project from being copied from one controller to another. Copy Protection also prevents the controller image from being uploaded via the Controller Image Upload/Download area.

Enable Programming Security

If you wish to apply security to the user programming source code, do the following:

1. In the Project Pane, right-click **Programming** and choose **Properties**.
2. On the **Security** page, click the **Display Copyright notice** checkbox. The RMCTools Security Policy and Agreement will open.
3. Read the RMCTools Security Policy and Agreement. If you accept the agreement, click **I accept this agreement** and click **Close**, and the copyright notice will be enabled. If you do not agree to the policy, click **I do not accept this agreement** and click **Close**, and the copyright notice will not be enabled.
4. **Enter Copyright Info**
   a. If you agreed to the policy, check the **Edit Copyright** button.
   b. Enter a copyright notice and click **OK**. If you plan on password-protecting the user programming source, make sure to enter contact information.
5. **Source Protection**
   If you wish to password-protect the user programming source in addition to having the copyright notice:
   a. Check the **Protect programming source code** box. Make sure that the copyright notice contains contact information.
   b. Enter a password in the **Password** box and re-enter it in the **Confirm Password** box. Passwords are case sensitive; therefore, you may want to make sure the CAPS LOCK is not on.
6. **Copy Protection**
   If you wish to copy-protect the programming:

a. Check the **Copy Protection** box.

7. Click **OK** to close the Programming Properties dialog.
8. If you are online with the RMC, right-click **Programming** and choose **Download Programs to Controller**.
9. Remember to update Flash, or the security settings will not be retained after cycling power.

Once programming security is enabled, a notice will appear each time the Programming node is downloaded. This is intended to prevent the user from unknowingly downloading a project with programming security enabled.

After applying the copyright notice and/or user program password protection and downloading to the RMC, the security settings in the RMCTools project itself is not password-protected. Anyone who uses that project can change the security settings and the password. If you wish to password protect the security settings in the project itself, lock the copyright notice and/or programming as described in the **Locking the Copyright Notice or Programming** section below.

Once password-protected programming is downloaded to the RMC, it is protected in the RMC. Therefore, if the controller contents are uploaded into a new RMCTools project, the copyright notice and/or user programming will be password protected as specified in the original project, and the security settings in the project will be locked. To unlock them, see the **Unlocking the Copyright Notice or Programming** section below.

Locking the Copyright Notice or Programming

Enabling a copyright notice, password protection, or copy protection and downloading to the RMC will password protect those items in the RMC, but the security settings in the RMCTools project itself is not password-protected. Anyone who uses that project can change the security settings and the password. To password-protect the security settings in the project, you can **lock** the copyright notice or programming. If a project with a locked copyright notice or programming is saved, the Programming portion will be saved in encrypted format.

A project with a locked copyright notice or programming can still be downloaded to an RMC.

To lock the copyright notice or programming:

- On the Programming menu, choose **Lock Copyright** or **Lock Programming.** Now the security settings cannot be changed.

Unlocking the Copyright Notice or Programming

If the copyright notice or programming has been locked, the security settings and user programs in the project will be password protected. In order to change the security settings or access the user programs in the project, the programming must be unlocked.

Unlocking the programming will unlock the project. Notice that the security settings will still be set, and if downloaded to the controller, will be applied to the controller.

To completely disable the security features in a locked project, you must first unlock the project, then disable the security settings on the Programming Security Dialog.

To unlock the copyright notice or programming:

- On the Programming menu, choose **Unlock Copyright** or **Unlock Programming**.
- Enter the correct password and click **OK**. Now the security settings can be changed.
- If you wish to disable the security, you can now do so on the Programming Security Dialog.

Required Firmware

The copyright notice and password protection features require RMCTools version 3.00.0 or newer and firmware version 3.00 or newer. Copy protection requires RMCTools version 3.43.0 or newer and firmware version 3.43 or newer.

See Also

RMCTools Security Policy and Agreement

5.10. RMCTools Security Policy and Agreement

The following security policy applies to the Programming Security feature. In order to enable the programming security feature, you must agree to this policy.

RMCTools Security Policy (RSP)

Purpose

This policy specifies the circumstances under which Delta Computer Systems, Inc. (Delta) will disable or bypass the programming security feature enabled by the copyright holder ("you" or "User") of programming downloaded to RMC series controllers. Delta's responsive customer support is an important service that benefits you and your customers. In order to ensure responsive service, Delta reserves the right to disable or bypass any and all security features applied to User programming.

When will the RSP be used?

This policy is intended to apply in customer support situations, including product repairs and phone or field service support, where an end user is unable to contact you or obtain customer support services from you in a timely manner.

How will the RSP be used?

Delta will disable or bypass programming security if Delta is contacted by an end user and determines that access, copying, or alteration of User programming is a necessary or desirable measure in assisting the end user.

- If you have provided the contact information in the copyright notice field during User programming, Delta will make a good faith effort to contact you and request instructions on assisting the end user. If contact by Delta is made by electronic means rather than in-person, Delta will allow for a sixty (60) minute response time.
- If you do not respond to Delta in writing within sixty (60) minutes, or if you have not provided Delta with effective contact information, or if Delta is unable to contact you for any other reason, Delta will proceed to assist the end user.

Therefore, in accordance with the RSP, and in consideration of your use of the programming security feature, you agree to the following:

RSP Agreement

You hereby grant to Delta a license to access, copy, and alter any source code you may program into any RMC series controller ("User Source Code"). Delta may access, copy, or alter User Source Code where Delta determines, in its sole discretion, that such action is necessary or desirable to assist an end user. Delta will make a good faith effort to avoid disclosure of User Source Code to the end user or any third parties while providing customer support. You hereby grant the end user a license to use User Source Code altered by Delta in accordance with the RSP, on the same terms and conditions as all User Source Code.

NO WARRANTY; LIMITATION OF LIABILITY. DELTA DOES NOT WARRANT THAT USE OF ANY PROGRAMMING SECURITY FEATURE WILL PROTECT USER SOURCE CODE FROM ANY UNAUTHORIZED USE OR DISCLOSURE. BY ENABLING THE PROGRAMMING SECURITY FEATURE, YOU EXPRESSLY AGREE THAT DELTA IS NOT LIABLE FOR ANY DAMAGES WHATSOEVER ARISING OUT OF OR RELATED TO YOUR USE OF, OR DELTA DISABLING OR BYPASSING, ANY PROGRAMMING SECURITY FEATURE, INCLUDING WITHOUT LIMITATION, LIABILITY FOR ACCESS, COPYING, ALTERATION, OR OTHER USE OR FOR DISCLOSURE OF USER SOURCE CODE. MOREOVER, YOU AGREE TO INDEMNIFY DELTA AGAINST ANY AND ALL LIABILITY, LOSS, AND EXPENSE ARISING OUT OF USER SOURCE CODE OR ANY USE OF THE PROGRAMMING SECURITY FEATURE, WHETHER OR NOT DELTA PROVIDES ASSISTANCE TO AN END USER.

See Also

Programming Security

5.11. User Programs

5.11.1. User Programs Overview

User Programs carry out complex sequences of commands on the RMC without requiring intervention from a PLC or other controller. This allows the RMC to respond to events within the control-loop time rather than the scan rate of a PLC. It also reduces the PLC programming required. The number of User Programs is only limited by the memory capacity of the RMC. Use the Verify Results window to find how much memory the User Programs are using.

Each User Program can contain any number of steps. Each step can issue commands, and link to another step. Wizards make creating links easy, or you can create your own complex link types.

Use variables to make User Programs flexible, and to easily influence them from a PLC. Perform math operations and assign values to tags and variables with expressions. You can use tag names in the Command Parameters, Link Types, and in the Expression (113) command to make the user programs very readable.

User Programs can be started by issuing the Start Task (90) command, or from the Program Triggers.

Tasks

User Programs run on **Tasks**. One User Program can run per Task. The RMC75 has up to four Tasks and therefore, up to four User Programs can run simultaneously. The RMC150 has ten Tasks and therefore, ten User Programs can run simultaneously. The RMC200 CPU20L has 32 tasks and the CPU40 has 64 tasks. To start a User Program, issue the Start Task (90) command. It starts the specified User Program on the specified Task. A User Program can also be started on a Task from within a User Program or from the Program Triggers. For more details, see the Running User Programs topic.

Steps for Creating and Running a User Program

- Create and edit a User Program
- Verify the User Program
- Download the User Program
- Run the User Program

Structure of User Programs

A User Program consists of multiple steps. Each step can execute several commands on one or several axes. The series of steps are linked together in sequences. Each step takes one loop time to execute. Therefore, the RMC controller can process a maximum of one step per User Program per loop time.

Each step consists of **Actions** and a **Link:**

Step Actions

The step actions are performed as soon as the task processes the step. The following actions can be performed in a user program step:

- **Command(s)**
  A step can issue any of the RMC commands. A maximum of one non-immediate command per axis can be issued in a single step (motion commands are an example of non-immediate commands). There is no limit to the number of immediate commands (aside from timing constraints), such as the Expression (113) command, that can be issued per step.

- **Commanded Axes**
  Each command in a step may be issued to one or several axes simultaneously. Each command in a User Program (except the Expression command, as described below), has a **Commanded Axes** section to define which axes the command should be issued to.

- **Expressions**
  The Expression (113) command can be used to perform mathematical calculations within the user program steps.

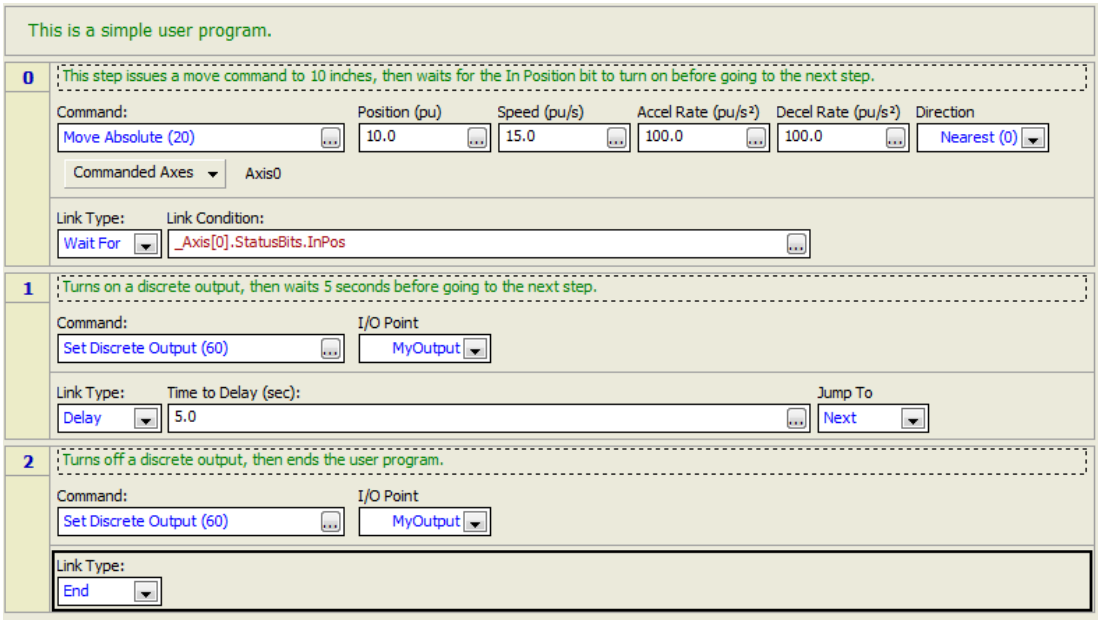
- **Declarations**
  Local variables can be declared in a step. These variables can only be accessed within the step and are not retained after the user program jumps to another step, even if it is to the same step. Local variables can be used in an expression command, in command parameters, or in link conditions.

Step Link Type

The **Link Type** specifies when to jump to the next step and which step to jump to. For example, a link type can wait until the axis is in position before going to the next step, or wait a certain amount of time. You can also define a complex link condition by entering a mathematical expression. See the Link Type topic for details.

Example:

This User Program makes Axis 0 move to 10 in, waits until it gets into position, turns on a discrete output for 5 seconds, then ends. The link condition in step 0 is easily created with a wizard.



Step and Condition Execution Details

When a step is executed, the commands in that step are issued in that loop time. Then, in the next loop time of the controller, the Link Type is evaluated. When the Link Type condition becomes true or tells the program to jump, the program will jump to the specified step and issue the commands in that step *in the same loop time* that the condition became true.

Expressions

The Expression command makes the User Programs very powerful. The Expression (113) command can only be used in User Programs. It can be used for mathematical calculations and to assign values to tags or variables.

There is no limit to the number of Expression commands that can be issued per step in the User Programs.

Local variables can be declared in a step. These variables can only be accessed within the step and are not retained after the user program jumps to another step, even if it is to the same step. Local variables can be used in an expression command, in command parameters, or in link conditions.

Importing and Exporting User Programs

User Programs can be exported to a file to be imported later into another project. When exporting a User Program, all the variables and tags it uses are included. When importing the User Program, a dialog guides you in assigning the variables and tags to the new project.

See the Exporting and Importing User Programs for details.

If you simply wish to copy a User Program from one project into another without including the variables and tags, you can use standard copy and paste functions.

Disabling User Programs

User programs can be disabled. Disabled user programs will be downloaded to the RMC, but will not be able to be run. Disabling user programs is useful if you want to keep a user program in the project for reference or later use even if it has errors.

To enable or disable a user program, right-click the program in the project pane and choose **Disable program** or **Enable program**. You can also use the Programming Properties dialog to disable and enable programs.

5.11.2. Creating User Programs

Follow the steps below to create a User Program. For example user programs, see the Programming Examples Overview topic.

Note:
In order to run User Programs or the Program Triggers, the RMC must be in RUN mode. See the RUN/PROGRAM Mode topic for details.

Basics of Creating a User Program

Create a User Program

1. In the Project Pane, expand the desired controller, expand the **Programming** node, right-click **User Programs** and choose **New Programs**.
2. Enter a name for the program. Names are limited to 64 characters.
3. If desired, change the **Program Number**.
4. Click **Finish**. The Step Editor will open with the new user program. The user program will appear in the Project pane, under **Programming** and **User Programs**.

Now you can use the instructions below to add steps and commands, do any additional editing, and download the user program.

Add a Program Comment

To add a comment for the entire program, right-click anywhere in the program and choose **Edit Program Comment**. Enter a description of the program and click **Save**.

Add Steps

You may add steps to a User Program at any time. On the Step Editor menu, click the **Insert Step** button. Or, right-click the space below the step number and choose **Add Step Before** or

Add Step After. To add a step to the end of the program, right-click the empty space below the last step and choose **Append Step**.

Edit the Step

See the **Editing a Step** section below.

Verify the Program

After you have created the program you must verify it before downloading it and running it. See the Verify topic for instructions. After creating and verifying a User Program, the next step is to download it before running it on the RMC.

Editing a Step

You may edit the steps at any time. For each step in the User Program, do any of the following:

Add Commands

Each step can have multiple commands and can issue any of the RMC commands. A maximum of one command per axis can be issued in a single step. However, there is no limit to the number of *immediate* commands that can be issued per step. Immediate commands include the Expression (113) and discrete I/O commands. The RMC controller can process a maximum of one step of a User Program per Task per loop time. A step is not required to have a command.

1. **Add a Command to the Step**
   - On the Step Editor toolbar, click the **Insert Command** button. Or, right-click close to a command and choose **Add Before** or **Add After**. Or, select a command and press Ctrl+Insert to add a command before, or Ctrl+Alt+Insert to add a command after.
   - Click the **Command** box and then click the Details button. Choose a command from the Command List and click **OK**. The commands are grouped by type in the hierarchical list to help you find the one you want. Alternately, you can select a command by clicking the **Command** box and begin typing the command name or number. Use the arrow keys or the mouse to select the command.

   Immediate Commands

   A maximum of one non-immediate command per loop time can be issued to each axis. There is no limit to the number of immediate commands that can be issued to an axis per loop time.

   To determine whether a command immediate, see the List of Commands.

2. **Enter the Command Parameters**

   If the command has any parameters, enter their values. In most parameters, you may enter a number, tag, or an expression. If you use a tag name, register, or expression, it must evaluate to the data type required by the parameter, typically a REAL.

   Tip:
   For help on the command's parameters, click the command box and press F1.

   - Click a parameter box.
   - To enter a **number**, type the number.
   - To enter a tag, click the Details button and choose the tag from the list. Or, start typing the tag name and then use the arrow keys to select the address.
   - To enter an expression or a tag, type it in the box. The Expression Browser is not available for command parameters, but the auto-complete will assist you.

3. **Selecting Commanded Axes**

   The Commanded Axes specifies which axes to issue the command to.

- Click the **Commanded Axes** button and choose the axes you wish to issue the command to, then click **OK**. For example, selecting Axis 0 and Axis 1 will make the command be issued to both axes.
- The **Default Axis** and **Use Expression** options are typically used only in advanced applications. If you choose Default Axis, the command will be issued to the same axis as the Start Task command was issued to. The **Use Expression** option can be used to programmatically select the commanded axes.

Important: A maximum of one non-immediate command per axis can be issued in a single step. If you try to issue more than one command to an axis per step, the verify will report an error and you will not be able to download the programs.

Add Expressions

To add expressions to a step, choose the Expression (113) command. Local variables can also be declared in a user programs step. To add a local variable to a user program step, on the Step Editor toolbar, click the **Add Step Declarations** button. Or, right-click the step and choose **Add Step Declarations**. See Local variables for more details.

Select a Link Type

The Link Type specifies when the program will jump to another step, and which step it jumps to. A step can jump to any step in the current user program, or to any labeled step in another user program.

Select one of the Link types in the **Link Type** box:

- **Immediate**
  Immediately jumps to the next step in the User Program. The next step is then executed in the next control loop of the RMC.
- **Jump**
  Jumps to the step specified in the Jump To box. You can enter a step number or label, or choose a step label from the drop-down list. The specified step is then executed in the next control loop of the RMC.
- **Delay**
  Waits the number of seconds specified in the Time to Delay box before jumping to the step specified in the **Jump To** box. You can enter a step number or label, choose a step label from the drop-down list, or choose ”Next” to jump to the next step in the sequence.
- **Wait For**
  Waits for the specified Link Condition to become true before jumping to the next step in the User Program.
- **Conditional Jump**
  The **Conditional Jump** link type can have one or more conditions. The conditions are evaluated in order. For the first condition that evaluates to True, the program jumps to the destination specified in its **Jump On True** box. If all conditions evaluate to False, then the program jumps to the destination specified in the **Jump On False** box.

  The Jump destinations can contain a step number, a step label, ”Wait”, ”Next”, or "Repeat":

  - **Next:** The program will jump to the next step in the sequence.
  - **Wait:** The program will stop evaluating the conditions, and re-evaluate the conditions the next control loop. For Conditional Jumps with multiple link conditions, ”Wait” must only be used for the **Jump On True/False** boxes for the last condition, and only in one of the boxes.
  - **Repeat:** The program will repeat the entire step, which means it will issue all the commands in the step.

  To add, delete, or re-order link conditions in a **Conditional Jump** link type, use the **Condition** buttons on the Step Editor toolbar ( ).
- **End**
  Ends the User Program.

Add a Step Comment

If you wish, you may add a comment to the step. Comments help you keep track of what the step is for and what it does.

- On the Step Editor toolbar, click the **Edit Step Comment** button, or right-click anywhere within the step and click **Edit Step Comment**.
- Type your comment and click **Save**.

Add a Step Label

If you wish, you may add a label to the step. Labels can be used when specifying the step to jump to. See the labels topic for details on labels.

- On the Step Editor toolbar, click the **Insert Step** button, or right-click in area below the step number and choose **Edit Label**. Type the label name and click **OK**.

To browse all the labels in a project, use the Browse Labels dialog.

Assigning labels to every step is not good programming practice.

Additional Tasks

Adding, Deleting and Moving Steps

- **Add a Step:**
  On the Step Editor toolbar, click the **Insert Step** button. Or, right-click the area below the step number and choose **Add Before** or **Add After**, or press Insert to add a step before the current step, or Alt + Insert to add a step after the current step.
  To add a step to the end of the program, right-click the empty space below the last step and choose **Append Step**.
- **Delete a Step:**
  On the Step Editor toolbar, click the **Delete Step** button. Or, select the entire step by clicking the area below the step number, then press Delete, or click **Remove Step** on the Step Editor toolbar. Or, right-click the area below the step number and choose **Delete Step**.
- **Move a Step:**
  Select a step by clicking the area below the step number, then drag the step to the desired location, or use the Move Step Up and Move Step Down buttons on the Step Editor toolbar. Or, right-click the area under the step number, then choose **Move Step Up** or **Move Step Down**.

When adding and deleting steps, the Step Editor automatically updates the Link Jump To numbers so that they jump to the same step that they did before the add or delete. For example, if step 3 has a Jump To number of 4, and you insert a step before step 3, then the old step 3 becomes the new step 4. Its Jump To number is automatically changed from 4 to 5.

If a deleted step was linked to with a number from some other step, a warning will appear in the Output window.

Adding, Deleting and Moving Commands

- **Add a Command:**
  On the Step Editor toolbar, click the **Insert Command** button. Or, right-click a command and choose **Add Before** or **Add After.**
  Or press Ctrl + Insert to add a command before the currently selected command or press Ctrl + Alt + Insert to add a step after the currently selected command.
- **Delete a Command:**
  On the Step Editor toolbar, click the **Delete Command** button. Or, select the entire command by clicking close to —but not inside—a command box. Make sure the entire command, including parameters, is selected, then press Delete, or click **Remove Command** on the Step Editor toolbar.
  Or, right-click the command so that the entire command with parameters is selected and choose **Delete Command**.

- **Move a Command:**
  Click a command such that the entire command with parameters is selected and drag it the desired location.
  Or, select the entire command with parameters and use the **Move Command Up** and **Move Command Up** buttons  on the Step Editor toolbar, or right-click a command and choose **Move Command Up** or **Move Command Down**.

Deleting a Step or Program Comment

- Double-click the comment to open the Edit Comment dialog and click **Delete**.

Showing and Hiding All Comments

- On the Step Editor toolbar click the **Show/Hide all Comments** button  , or right-click a step and click **Show Comments** or **Hide Comments**.

Renaming User Programs

In the Project pane, right-click the User Program and click **Rename**, or select the User Program and press F2.

Assigning Tasks the User Program May Run On

By default, a user program is allowed to run on only one task at a time. This setting is usually sufficient for most applications.

You can change this to specify a certain task that the user program is allowed to run on, or allow the user program to run on any number of tasks simultaneously.

Restricting the tasks that a program can run on will help detect program errors and help reduce the maximum programming execution time.

To change this setting, in the Project pane, right-click the user program, choose **Properties**, and choose the **Tasks** page.

See Also

User Program Overview | Example: Basic User Program

5.11.3. Verifying User Programs

After you have created a User Program, it must be verified before downloading and running it in the RMC. The verify process checks the program for errors that will keep it from running. It does not check for other errors, such as invalid command parameters or bad logic. If the verify finds errors, the verify will fail, and the Programming node will not be allowed to be downloaded to the RMC.

To Verify the User Programs

1. In the Project Pane, expand the desired controller, right-click **Programming** and click **Verify Programs**.
   Or, on the **Programming** menu, click **Verify**.
2. The results of the verify will appear in the **Verify** tab of the **Output** window. If errors are found, you must edit the program to fix the problem and verify again. To locate the error, double-click the error in the Verify Results window. Or, press F4 to open the location of the next error and press Shift + F4 to open the location of the previous error. RMCTools will open the editor in which the error exists and highlight the error.
3. If no errors are found, download the User Programs to the controller. If errors are found, you must correct them before you will be able to download the programs to the RMC.
   If the program size or time exceeds the limits, see the Program Capacity and Time Usage topic for tips on how to fix this.

4. After you have downloaded the User Programs, you may run them. See the Running User Programs topic for instructions.

Note:
The Verify will not verify user programs that have been disabled. Disabled user programs will be downloaded to the RMC, but will not be able to be run. Disabling user programs is useful if you want to keep a user program in the project for reference or later use even if it has errors. See the User Program Properties topic for details on disabling programs.

See Also

User Program Overview

5.11.4. Running User Programs

After you have created, verified and downloaded a User Program, you can run it on the RMC. This topic describes how to start a User Program. See the Stopping User Programs topic also.

Note:
In order to run User Programs or the Program Triggers, the RMC must be in RUN mode. See the RUN/PROGRAM Mode topic for details.

Starting a User Program

There are several ways to start a User Program:

- **Send the Start Task (90) command from RMCTools:**
  1. In the **Command Tool**, click the Axis 0 **Cmd** box. Type "90" and press enter.
  2. In **Program** parameter, enter the number of the User Program you wish to run.
  3. In the **Task Number** parameter, enter the number of the task you wish to run the User Program on.

     Tip:
     For help on determining which task to run a User Program on, see the Tasks and Start Task (90) topics.
  4. Click **Send Command**.
- **Send the Start Task (90) command from a PLC or other host controller:**
  1. See the Issuing Commands topic for details.
- **Send the Start Task (90) command from a User Program:**
  1. In the **Command** box, type "90" and press Enter.
  2. Enter the correct values in the **Task Number** and **Program** parameters. For details on these parameters, see the Start Task (90) command.
  3. Complete the step as described in the Creating User Programs topic.
- **From the Task Monitor**
  - In the Task Monitor, to start a User Program, right-click a Task listed in the Task Monitor and click Start Task.
- **From the Project tree**
  - In the Project pane, right-click a User Program, choose **Run Program**, then choose the Task on which to run the program.
    This option is available only if the Programming node has no differences between the project and controller. Notice that a user program cannot be stopped using this method.

- **Create a Trigger in the Program Triggers**
  - See the Program Triggers for details.

How to Determine if a User Program is Running

Use the Task Monitor to track which User Programs and steps are currently running. See the Tasks topic for details.

To open the Task Monitor, on the **View** menu, click **Task Monitor**.

See Also

User Program Overview | Start Task (90) Command

5.11.5. Stopping User Programs

After you have created, verified and downloaded and started a User Program, you may want to stop it.

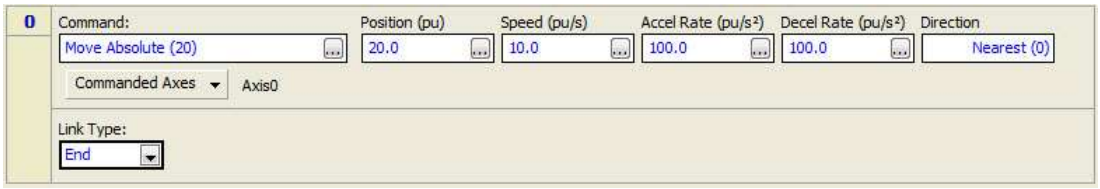
How to Stop a User Program

- **End Link Type:**
  If the user program encounters an End link type, it will stop.
- **Issue the Stop Task (91) Command:**
  Issue the Stop Task (91) command from RMCTools, a PLC, or other host controller. You must know which task the program is running on to stop it. Use the Task Monitor to see which task the user program is running on.
- **Halt an axis:**
  By default, the RMC is configured such that, if an axis halts, all tasks will stop. This setting can be changed on the Programmng Properties dialog.
  See the Halts Overview topic for details on when halts can occur.
- **From the Task Monitor**
  In the Task Monitor right-click a Task listed in the Task Monitor that is running the user program and click **Stop Task**.
- **Create a Trigger in the Program Triggers**
  A trigger in the Program Triggers containing **<StopTask>** in one of the Task columns will stop that task when the condition becomes true.

What Happens When a User Program Stops

A User Program is simply a method of issuing commands to the RMC. When the User Program stops, it stops issuing commands to the RMC. It does *not* mean that the commands it issued are terminated, nor that motion on the axis stops.

For example, consider the following user program:



This user program only runs one step (which takes only one loop time of the RMC), then stops. However, the command that was issued will be processed until it is complete. The axis will move to 20 pu and then stop.

See Also
User Program Overview

Show All

5.11.6. Labeling Steps

You can add a label to any step in a User Program. The label may be used in Link Types when specifying the step to jump to. Labels are global: you can jump to a label in any User Program. Prudent use of labels makes the User Programs more flexible and readable.

It is not necessary to label a step to jump to it from the previous step. Entering "Next" in the **Jump to** box of a link type will jump to the next step without requiring a label.

Using Labels to Jump to Steps

Jumping to a step by specifying a Label is often better than specifying a Step Number. If you use a step number, the number will change if a step is inserted before it. Then the jump will go to the wrong step. If you use a step Label, the jump will always go the step with that label.

To create a jump to a step with a label:

- Add a label to the step you wish to jump to.
- In the step you are jumping from, choose one of the **Jump** Link Types.
- In the **Jump To**, **Jump on True**, or **Jump on False** box, type the name of the label**.**

Add a Label to a Step

- Right-click anywhere in the step and click **Edit Label.**
- Type the name in the **Label** box and click **OK.**

Delete a Label

- On the **Editor** menu, click **Edit Label**.
- Click **Delete**.

Edit a Label

- Right-click anywhere in the step and click **Edit Label**.
- Make any changes and click **OK**.

Finding Labels

In large projects, it may be difficult to keep track of all the labels. Use the Browse Labels dialog to view all the labels in a project and their locations. You can also use it to go to the location of a label.

See Also
User Program Overview

5.11.7. Exporting and Importing User Programs

User Programs can be exported to a file to be imported later into another project. When exporting a User Program, all the tags it uses are included. When importing the User Program, a dialog guides you in assigning the variables and tags to the new project.

Exporting a User Program

To export a User Program, in the project pane, right-click the User Program and click **Export User Program**. Choose a location and name to save the file to and click **Save**.

Importing a User Program

To import a User Program:

- In the project pane, right-click **Programming** and click **Import User Program**. Choose a file and click **Open**.
- If the User Program uses any variables or tags, the **User Program Import** dialog will open. The dialog lists all the variables used by the User Program. In order for the program to verify correctly, these tags must be assigned to locations in the project.
  The User Program Import dialog attempts to automatically allocate the tags. If no space is available for a tag, the user must assign the tag to a new destination address, or choose to skip importing that specific tag. After you are finished assigning or skipping the tags, click **OK**. For more details, see the User Program Import Dialog topic.

See Also

User Program Overview

5.11.8. Using Expressions for Commanded Axes

This topic describes the **Use Expressions** option in the **Commanded Axes** list in user program commands. This allows the commanded axes to be programmatically selected when the user program is running. This feature is useful for machines that occasionally may need to disable an axis but yet run the user programs that normally send commands those axes.

This is an advanced feature that will not typically be used in most applications.

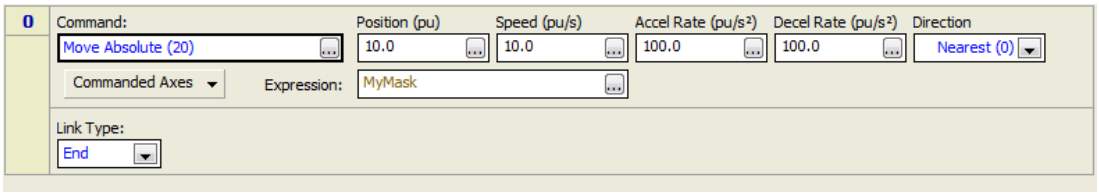
Overview

The **Use Expression** option provides an expression box to provide a mask for selecting the commanded axes. The expression must evaluate to a DWORD. The bits in the DWORD correspond to the axis number. If the bit is set, the command will be sent to the corresponding axis. If the no bits are set, the command will not be sent to any axis. Any bits set for which no axis exists will be ignored. For controllers with more than 32 axes, multiple expression boxes will appear with one expression box per group of 32 axes.

Example 1: Single Command to Multiple Axes

This example demonstrates how to send a single command to any axes as set by a variable.

1. **MyMask** is as a DWORD variable in the Variable Table.
2. This user program sends a Move Absolute (20) command to the axes defined by the bits set in the MyMask variable:



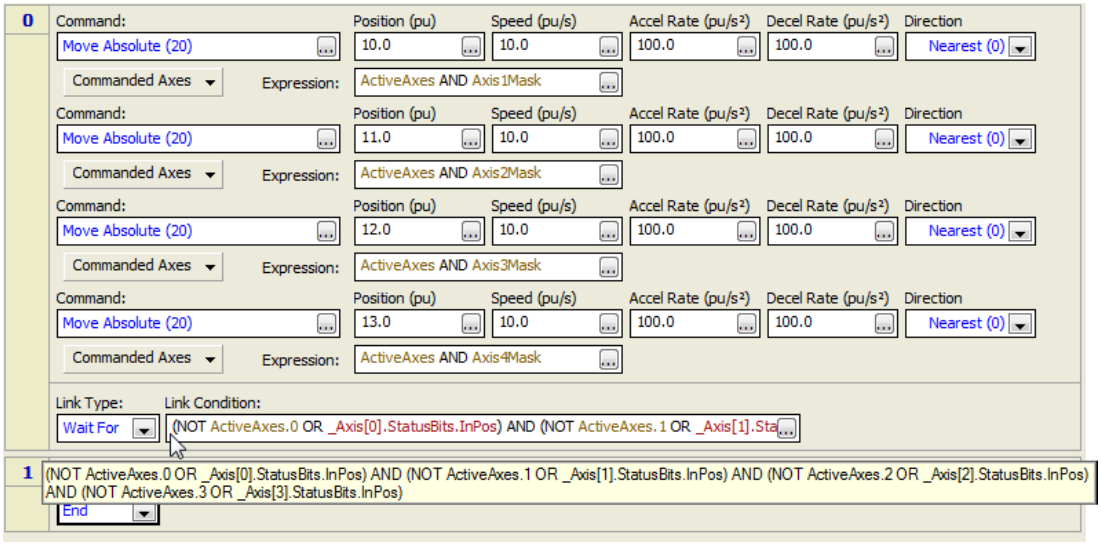
3. When the user program runs, if bits 0, 1, and 3 are set, then the command will be sent to Axis 0, Axis 1, and Axis 3.

Example 2: One Command to Each Axis

This example demonstrates one method of programmatically removing one or more axes when sending a set of commands to a set of axes.

1. The following variables are defined:
   a. **ActiveAxes:** A DWORD that will be define which axes should receive commands.
   b. **Axis1Mask, Axis2Mask, Axis3Mask, Axis4Mask:** DWORD variables that will be used as constants. Each of these variables is initialized to a value that has the bit set for its respective axis. The values of these variables must always remain the same.

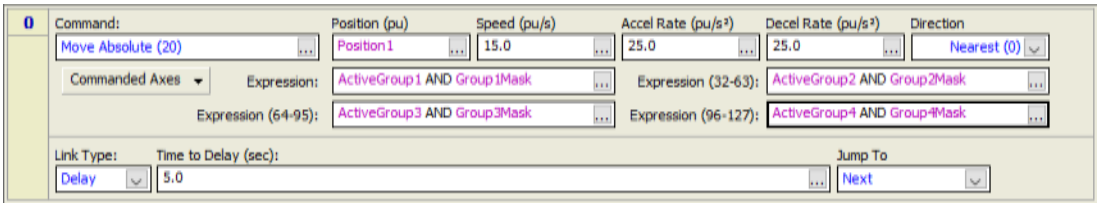
2. This user program step has one Move Absolute (20) command for each axis. Since the **ActiveAxes** variable is ANDed with the axis-specific **Axis*n*Mask** variable, each command can only be sent to the axis specified by the respective **Axis*n*Mask** variable, and only if the respective bit in the **ActiveAxes** variable is set.



3. The **Wait For** Link Condition waits for the In Position bit to turn on for only the selected axes. The expression is too long to be viewed from the Link Condition box, but is shown above in the tooltip window.

Example 3: More than 32 Axes

The RMC200 allows for projects with more than 32 axes. When more than 32 axes are used, the Commanded Axes Expression will have multiple expression boxes. Each box is associated with a respective 32 bit DWORD for the 32 axes specified by each box. All expression boxes must contain an expression in order for the User Program to verify.



See Also

Creating User Programs | Expressions Overview

5.11.9. Link Types

5.11.9.1. Link Types Overview

A Link Type specifies the condition that makes the RMC jump to and start the next step in a User Program. As a User Program runs, the RMC checks the Link Type of the current step every loop time. When the Link Type evaluates to true, it jumps to the specified step. The step it jumps to is specified by a valid step number or a step label. A maximum of one step can be executed per loop time.

For example, you can set a link type to wait until the axis is in position before going to the next step, or you may want to wait a certain amount of time. Complex link conditions can also be specified.

Selecting a Link Type

To select a link type, go to the desired step in a User Program and double-click the **Link Type** box. Choose one of the link types.

List of Link Types

Link Type	Description
Immediate	Jumps immediately to the next step. The next step will be executed in the next loop-time of the RMC.
Jump	Jump immediately to the step specified in the **Jump to** box. The **Jump to** box must contain a valid step number or label.
Delay	Waits for number of seconds specified in the **Link Condition** box and then jumps to the step specified in the **Jump to** box. The **Link Condition** box must evaluate to a REAL value. It can be any of the following: • Constant value • Register • Variable • Numeric Expression
Wait For	Waits for the **Link Condition** to become true and then jumps to the next step. The **Link Condition** box must be a logical expression that evaluates to True or False.
Conditional Jump	Evaluates the **Link Condition** or conditions. If a condition is true, the **Jump On True** is taken. If all conditions are false, the **Jump On False** is taken.

	The **Jump On True** and **Jump On False** boxes must contain a valid step number, label, "Next", "Wait", or "Repeat". The **Link Condition** box must be a logical expression that evaluates to True or False.
End	Ends the sequence. No steps are executed after this step.

5.11.9.2. Link Type: Immediate

Note:
A Link Type specifies the condition that makes the RMC jump to and execute the next step in a User Program.

The Immediate Link Type immediately jumps to the next step in the User Program. The next step is then executed in the next loop time of the RMC.

To select the Immediate Link Type:

- Open or create a User Program.
- Go to the step where you want the Immediate Link Type.
- Double-click the **Link Type** box.
- Click **Immed**.

See Also
Link Type Overview

5.11.9.3. Link Type: Jump

Note:
A Link Type specifies the condition that makes the RMC jump to and execute the next step in a User Program.

The Jump Link Type jumps to the step specified in the **Jump To** box. The Jump destination can contain a step number, a step label, ”Next”, or "Repeat". If the Jump destination is ”Next”, then the program will jump to the next step in the sequence. If the Jump destination is ”Repeat”, then the program will repeat the same step, which means it will issue all the commands in the step. When the user program jumps to a step, that step is executed in the next loop time of the RMC.

Boxes

The Conditional Link Type has the following boxes:

Box	Description
Jump To	Specifies the step to jump to. It must be a valid step number, step label, "Next", or "Repeat".

To select the Advanced Condition Link Type:

- Open or create a User Program.
- Go to the step where you want the Advanced Condition Link Type.
- Double-click the **Link Type** box.
- Click **Jump**.

See Also
Link Type Overview

5.11.9.4. Link Type: Delay

Note:
A Link Type specifies the condition that makes the RMC jump to and execute the next step in a User Program.

The Delay Link Type waits the number of seconds specified in the **Time to Delay** box before jumping to the step specified in the **Jump To** box. The Jump destination can contain a step number, a step label, ”Next”, or "Repeat". If the Jump destination is ”Next”, then the program will jump to the next step in the sequence. If the Jump destination is ”Repeat”, then the program will repeat the same step, which means it will issue all the commands in the step. When the user program jumps to a step, that step is executed in the next loop time of the RMC.

Boxes

The Delay Link Type has the following boxes:

Box	Description
Time to Delay	The Time to Delay specifies the number of seconds to wait before jumping to the next step in the User Program. It must evaluate to a REAL data type and may be any of the following: • Constant value • Register • Variable • Numeric Expression
Jump To	Specifies the step to jump to. It must be a valid step number, step label, "Next", or "Repeat".

To select the Delay Link Type:

- Open or create a User Program.
- Go to the step where you want the Delay Link Type.
- Double-click the **Link Type** box.
- Click **Delay**.

Entering an Expression in the Delay Link Type

If you use an expression in the Delay Link Type, it must evaluate to a REAL value. To enter an expression, double-click the **Time to Delay** box, which will open the Expression Editor. The expression can be one line only. For more details on creating an expression, see the Expressions Overview.

Example Time to Delay Expressions

Example 1

Example 2

Note: MyTimeToDelay must exist as a variable.

Timing Considerations

After the step containing the Delay link type is executed, the RMC will wait the specified number of seconds, then jump to the next step and execute it. Therefore, the next step will be executed exactly at the delay time *plus one loop time* after the first step.

For example, consider a program where step 0 contains the Delay link type with a time of 0.1 seconds and jumps to the next step. Assume the controller loop time is set to 0.001 seconds (1000 µs). If step 0 is executed at time 0 (zero), then step 1 will be executed at time 0 + 0.1 + 0.001, which is 0.101.

If you want the step to be executed at a specific interval, you must specify the delay to be the interval time minus the loop time. For example, if you wish to issue the commands at exactly a 2 second interval and the loop time is 0.0005 seconds (500 µsec), you should set the delay time to 1.9995 seconds.

You can use the **_LoopTime** tag to specify get the loop time. The _LoopTime tag is accessible from the Expression Builder on the **Tags** tab, in the **Controller** section.

See Also

Link Type Overview

5.11.9.5. Link Type: Wait For

Note:
A Link Type specifies the condition that makes the RMC jump to and execute the next step in a User Program.

The Wait For Link Type waits for the **Link Condition** to become true before jumping to the next step in the User Program. If you want the User Program to wait a certain period of time, use the Delay link type instead.

Boxes

The Wait For Link Type has the following boxes:

Box	Description
Link Condition	The Link Condition specifies the condition that must become true before jumping to the next step in the User Program. This condition must be a logical expression that evaluates to True or False.

To select the Wait For Link Type:

- Open or create a User Program.

- Go to the step where you want the Wait For Link Type.
- Double-click the **Link Type** box.
- Click **Wait For**.

Timing Considerations

Beginning with the loop time after the step containing the Wait For link type is executed, the RMC will monitor the Wait For link condition. In the same loop time that the condition becomes true, the program will jump to the next step and execute it. Therefore, the specified step can be executed no sooner than the next loop time after the first step was executed, and the state of the condition in the same loop time that the step is executed is not considered.

Example

Consider a program where Step 0 issues a Move Absolute command and has a Wait For link type with a link condition that waits for a discrete input to be on. In the loop time that step 0 is executed (the Move Absolute command is issued), the link condition is not evaluated. The next loop time, the program will evaluate the link condition. If the condition is true (the discrete input is on), the program will jump to the specified step and execute it in that same loop time. If the condition is false, (the discrete input is off), the program will wait. This will repeat until the condition is true, at which point, the program will jump to the specified step and execute it in the same loop time in which the condition became true.

See Also

Link Type Overview

5.11.9.6. Link Type: Conditional Jump

Note:
A Link Type specifies the condition that makes the RMC jump to and execute the next step in a User Program.

The **Conditional Jump** link type can have one or more conditions. The conditions are evaluated in order. For the first condition that evaluates to True, the program jumps to the destination specified in its **Jump On True** box. If all conditions evaluate to False, then the program jumps to the destination specified in the **Jump On False** box.

The Jump destinations can contain a step number, a step label, ”Wait”, ”Next”, or "Repeat":

- **Next:** The program will jump to the next step in the sequence.
- **Wait:** The program will stop evaluating the conditions, and re-evaluate the conditions the next control loop. For Conditional Jumps with multiple link conditions, ”Wait” must only be used for the **Jump On True/False** boxes for the last condition, and only in one of the boxes.
- **Repeat:** The program will repeat the entire step, which means it will issue all the commands in the step.

When the user program jumps to a step, that step is executed in the next loop time of the RMC.

Boxes

The Conditional Link Type has the following parts:

	Description
Link Condition	The Link Condition specifies the expression that is evaluated. It must be a logical expression that evaluates to True or False.

Jump On True	Specifies the step to jump to if the Link Condition is True. It must be a valid step number, step label, "Wait", "Next", or "Repeat". For Conditional Jumps with multiple link conditions, ”Wait” must only be used for the **Jump On True/False** boxes for the last condition, and only in one of the boxes.
Jump On False	Specifies the step to jump to if the Link Condition is False. It must be a valid step number, step label, "Wait", "Next", or "Repeat". For Conditional Jumps with multiple link conditions, ”Wait” must only be used for the **Jump On True/False** boxes for the last condition, and only in one of the boxes.

To select the Conditional Jump Link Type:

- Open or create a User Program.
- Go to the step where you want the Conditional Jump Link Type.
- Double-click the **Link Type** box.
- Click **Cnd Jump**.

Editing the Link Condition

Each Link Condition box in the Conditional Jump Link Type must contain a single logical (comparison) expression. To begin creating the expression, double-click the Link Condition box. The Expression Editor will open to assist you in entering an expression. See the Expressions Overview topic for details on creating expressions.

Adding and Deleting Link Conditions

- To add a Link Condition, right-click the **Link Condition** box and choose **Add Before** or **Add After**, or click the Add Link Condition After button  on the Step Editor toolbar.
- To delete a Link Condition, right-click the **Link Condition** box and press Delete, or click the Delete Link Condition  button.
- To move a Link Condition, select the **Link Condition** box and click the Move Link Condition Up  or Move Link Condition Down  buttons.

Example

See the Example: Time-out topic for an example of using the Conditional Jump.

Timing Considerations

Beginning with the loop time after the step containing the Conditional Jump link type is executed, the RMC will monitor the Conditional Jump link condition. In the same loop time that the condition indicates a jump should be made, the program will jump to the specified step and execute it. Therefore, the specified step can be executed no sooner than the next loop time after the first step was executed, and the state of the condition in the same loop time that the first step is executed is not considered.

Example 1

Consider a program where Step 0 issues a Move Absolute command and has a Conditional Jump link type with a link condition that waits for a discrete input to be on. The Jump on True contains step 1 and Jump on False contains step 2. In the loop time that step 0 is executed (the Move Absolute command is issued), the link condition is not evaluated. The next loop time, the program will evaluate the link condition. In this case, either step 1 or step 2 will be executed. If the time is 1 millisecond, and step 0 was executed at time 0 (zero), then step 1 or step 2 will be executed at time 0.001 seconds.

Example 2

Consider a program where Step 0 issues a Move Absolute command and has a Conditional Jump link type with a link condition that waits for a discrete input to be on. The Jump on True contains step 1 and Jump on False contains "Wait". In the loop time that step 0 is executed (the Move Absolute command is issued), the link condition is not evaluated. The next loop time, the program will evaluate the link condition. If the condition is true (the discrete in put is on), the program will jump to the specified step and execute it in that same loop time. If the condition is false, (the discrete in put is on), the program will wait. This will repeat until the condition is true, at which point the program will jump to the specified step and execute that step in the same loop time in which the condition became true.

See Also
Link Type Overview

5.11.9.7. Link Type: End

Note:
A Link Type specifies the condition that makes the RMC jump to and execute the next step in a User Program.

The End Link Type stops the task. No steps are executed after this step.

To select the End Link Type:

- Open or create a User Program.
- Go to the step where you want the End Link Type.
- Double-click the **Link Type** box.
- Click **End**.

See Also
Link Type Overview

5.12. Data Types

5.12.1. Data Types

The RMC's expressions are strongly typed. This means that any math or comparison operation in an expression that involves multiple tags must operate on tags of the same data type. This does not mean that each expression must contain only the same data types - rather, each operator within an expression must operate on the same data types. Data types can be converted using specific conversion functions.

The RMC data have the following internal types:

Data Type	Description
REAL	32-bit floating point number.

	When typing a REAL number, it must include a decimal point, or RMCTools will assume it is a DINT.
DINT	32-bit integer number. When typing a DINT number, it must not include a decimal point.
DWORD	32-bit string of bits. Each bit in a DWORD data type is a boolean and can be individually addressed by adding "." and then the bit number. **Example:** MyDWordVariable.12 is a boolean. **Note:** The DWORD and DINT data types are not interchangeable. A function that requires a DINT will not accept a DWORD. Use a conversion function to convert a DWORD to a DINT or a DINT to a REAL.
BOOL	Boolean, 1 bit, either True or False. For discrete I/O, On = True, Off = False. As a bit in a DWORD, 1 = True, 0 = False. Each bit in a DWORD data type is a boolean and can be individually addressed by adding "." and then the bit number. **Example:** MyDWordVariable.12 is a boolean.

To find the data type of a register, use the Address Selection Tool, or the Register Maps.

Arrays

An array is a numerically indexed sequence of elements of the same data type. The RMC supports arrays of each 32-bit data type: REAL, DINT, and DWORD. See the Arrays topic for details.

Internal versus External Data Types

Some RMC75 and RMC150 registers have different data types depending on whether they are accessed from user programs or externally, such as from a PLC. The RMC150 Register Map and RMC75 Register Map topics list the external and internal data types of the registers. When reading from and writing to registers in the RMC75 or RMC150 via one of the communication protocols, use the external data type. The RMC200 does not have different internal and external data types.

The different external and internal data types are intended to make it easier to communicate with the RMC. Some registers that are DINT or DWORD internally are a REAL externally, so that the PLC does not have to deal with may different data types.

Converting Data Types

When used in a math or compare operation in expressions, data types can not be mixed without explicitly converting them. The following functions convert data to different data types:

- REAL_TO_DINT(*a*)
- DINT_TO_REAL(*a*)
- DWORD_TO_DINT(*a*)
- DINT_TO_DWORD(*a*)

See the functions topic for usage details.

See Also

Data Type REAL | Data Type DINT | Data Type DWORD | Data Type BOOL

5.12.2. BOOL Data Type

A BOOL number in the RMC is a single bit. In the RMC, a single bit cannot exist by itself. It is always a part of a of a DWORD value. For details on creating a boolean variable, see the Boolean Variables topic.

Addressing a BOOL Value

In an expression entered in RMCTools:

A boolean number can be addressed in the following ways in an expression. Notice that within the RMC, individual bits cannot be addressed using the DF1 or Modbus/RTU addressing formats.

Tag name: Add a period (.) and the bit number or name after the tag name of the register. The register *must* be a DWORD type.

Examples:	MyVariable.0	Note: MyVariable must be a DWORD
	_Axis[0].StatusBits.InPos	
	_Axis[0].StatusBits.13	

IEC addressing: Add a period (.) and the bit number after the IEC address of the register. The register *must* be a DWORD type.

Examples: %MD8.1.12 - bit 12 in the Error bits register

%MD56.10.16 - bit 16 in the variable 10

From a host controller:

When communicating with the RMC from an external host controller, to address a bit, you must specify the bit in the register that contains it. Not all host controllers allow this, depending on the protocol. For example, addressing a bit in an F file using an Allen-Bradley protocol is not supported in many HMIs.

Typically, specifying bits in any word is possible with most controllers using Modbus/RTU or Modbus/TCP. Notice that you may need to address the upper 16-bit portion of the 32-bit register in order to pick out the bits higher than 15.

See Also

Data Types | Data Type REAL | Data Type DINT | Data Type DWORD

5.12.3. DINT Data Type

A DINT number in the RMC is a signed 32-bit integer number. When typing a DINT number in an expression, it should not include a decimal point, or it may be interpreted as a REAL.

For details on creating a DINT variable, see the Variables topic.

See Also

Data Types | Data Type REAL | Data Type DWORD | Data Type BOOL

5.12.4. DWORD Data Type

A DWORD number in the RMC is a 32-bit string of bits. In RMCTools, DWORDS are displayed with a "16#" or "0x" prefix. For example, 16#1A0002Af.

Each bit in a DWORD data type is a boolean (BOOL data type) and can be individually addressed by adding "." and then the bit number.

Example: MyDWordVariable.12 is a boolean.

Bit Ordering

DWORDs are displayed in RMCTools with the highest byte first. For example, in 16#80000000, bit 31 is set. 16#00000001, bit 0 is set.

For details on creating a DWORD variable, see the Variables topic.

See Also

Data Types | Data Type REAL | Data Type DINT | Data Type BOOL

5.12.5. REAL Data Type

A REAL number in the RMC is a 32-bit floating point number, conforming to the IEEE-754 Floating-Point specification. For details on calculations limits due to the nature of REALs, see the Limitations of 32-bit Numbers topic.

For details on creating a REAL variable, see the Variables topic.

See Also

Data Types | Data Type DINT | Data Type DWORD | Data Type BOOL

5.13. Expressions

5.13.1. Expressions Overview

Expressions are used to assign values to tags or variable, and perform mathematical calculations, both simple and complex. Expressions provide for efficient calculations and compact User Programs. RMCTools provides an Expression Editor, which includes the Expression Browser, to help you create expressions. Expressions can be used in the following places:

- **Expression Command**
  Expressions are used in the Expression (113) command to assign values to variables or other registers.
- **Link Conditions**
  Expressions are used in the Wait For and Conditional Jump Link Types to specify what conditions should be fulfilled before jumping the next step.
  Expressions are used in the Delay Link Type to specify the time delay before jumping the next step.

- **Program Triggers**
  Expressions define the trigger conditions.
- **Command Parameters in User Programs**
  Expressions can be entered in command parameters in user programs, although the Expression Browser cannot be used for this. The expressions must evaluate to the data type required by the parameter, typically a REAL.
- **Commanded Axes in User Programs**
  Expressions can be entered to programmatically select the commanded axes for commands in user programs. See Using Expressions for Commanded Axes for details.

How to Create Expressions

See the Condition Expressions topic for creating expressions in the Program Triggers and in link types.

See the Assignment Expressions topic for creating expressions in user programs with the Expression (113) command.

See the Value Expressions topic for details on creating expressions in the Delay Link Type and in command parameters.

Examples

For examples of expressions, see the Condition Expressions and Assignment Expressions topics.

For examples of user programs that use expressions, see the Programming Examples topic.

Parts of an Expression

The building blocks of all expressions are:

Operators These are symbols that represent an action to be performed, such as +, -, /, etc.

Functions These are predefined, named formulas, such as SIN(), MIN(), LOG_EVENT(), etc.

User Functions User functions are custom functions created or imported by the user. User functions can be edited in the User Function Editor.

Operands These are the elements that the operators and functions work on:

Type	Examples
Constant Value	3, 10.345
Register (tag)	_Axis[0].ActPos, %MD8.3
Variable	SampleVariable1 (must exist in the variable table)

Keywords These are words that have a specific meaning for the code, such as IF, THEN, ELSE, True, False, etc.

Comments Comments are a good way of documenting the code.

Expression Color-Coding

Expressions are color-coded as follows:

Blue: Keywords

Dark Red: System-defined tags

Beige: User-defined tags

Black: Operators and numbers

Magenta: Functions

Red with yellow background: Invalid value

Troubleshooting Expressions

For help on troubleshooting expressions, see the Troubleshooting Expressions topic.

Operators and Functions

For details on the operators and functions used in expressions, see the Operators and Functions topics.

See Also

Assignment Expressions | Condition Expressions | Value Expressions | Functions | User Functions | Operators | Keywords | If Statement | Arrays | Limitations of 32-bit Numbers | Comments

5.13.2. Assignment Expressions

An assignment expression assigns a value to a register. Assignment expressions are used in the Expression (113) command in a User Program, and in User Functions.

Assignment Expression Basics

An assignment expression must follow this format:

Register := Expression;

where:

Register must be writable and may be any of the following:

- Variable
- Local Variable
- Register - specified by its tag name
- Register - specified by an address

The semicolon at the end is not required if the Expression command contains only one assignment.

Expression is a mathematical expression that must evaluate to a numeric value with the same data type as the **Register**.

Sample Expressions

Example 1

```
MyCounter := MyCounter +1;
```

Note: MyCounter must exist as a variable.

Example 2

```
MyVariable := 12.9 + SampleVariable2 / (3.0 + Axis[1].ActPrs);
```

Note: MyVariable and SampleVariable2 must exist as variables.

Example 3

```
F18:2 := 3.0 + ABS(Axis[].ActPos);
```

Example 4

```
IF _Axis[0].StatusBits.0 = True THEN
  MyREAL1 := 34.0;
  MyREAL2 := 70023.0;
ELSIF ABS(_Axis[0].ActPos) > 20.0 THEN
  %QX0.1 := True;
ELSE
  MyDINT := 2;
END_IF
```

Note: MyReal1, MyReal2, and MyDINT must exist as variables.

Entering an Expression

1. In a User Program, choose the Expression (113) command, and double-click the Expression box. The Expression Editor will open.
2. Enter the first register and the assignment operator. Note that the Expression *must* begin with "[register] := ".
   To enter the first part of the expression:
   - In the **Tags** box, find the desired register and double click it. It will be placed in Expression box. Then, in the **Operators** box, double-click the assignment operator (:=). It will also be placed in Expression box.

     Note:
     You can also type the information directly in the **Assignment Expression** box.

   Local Variables:
   If your expression needs temporary variables, you can declare variables in the user program step. See Local Variables for details.
3. Complete the expression:
   - Double-click any item in the lists on the **Tags**, **Functions** or **Operators** tabs to insert into the expression.
   - Type other items, such as register addresses or numeric values.
   - To mix data types, you must explicitly convert them. For example, you cannot directly assign a fractional value to an integer variable. You must use the data type conversion functions.
   - If you enter multiple expressions in the Expression command, each expression must end with a semi-colon. See the examples above.
   - If the expression is invalid, the Expression Builder give an error. When no errors are listed, the expression is valid.
4. Click **OK** to close the dialog and enter the expression in the command.

See Also

Expressions Overview | Constants | Functions | Operators | If Statement | Keywords

5.13.3. Condition Expressions

Condition expressions are used in the **Condition** box of a Program Trigger item, and in the Wait For and Conditional Jump link types. A condition expression must evaluate to True or False.

Simple condition expressions can also be entered with the New Condition Wizard.

Sample Expressions

1. _Axis[0].Status.InPos = True //checks if the in Axis 0 Status Bit named "In Position " is on.

2. _Axis[1].ActPos < 4.0 AND MyInput = True

3. _Axis[0].ActPos > Pos1 AND _Axis[1].ActPos < 4.0

4. MyVariable + SQRT(_Axis[1].ActPrs) >= 12900.1

5. counter >= 300

Entering an Expression

Using a Wizard

In the **Condition** box of a Program Trigger or a link type, click the ellipsis button  . The New Condition Wizard will open. Use the wizard to create the following types of expressions:

- Test a Status bit, such as the In Position bit.
- Test and Error bit, such as the Fault Input bit.
- Soft Limit Switch , such as comparing whether the Actual Position is greater than a ceratin value.
- Discrete I/O, such as checking if a discrete input is on.

Entering Directly

1. In the **Condition** box of a Program Trigger or a link type, double-click the **Condition** box. The Expression Editor will open.
2. Enter the expression:
    - Double-click any item in the lists on the **Tags**, **Functions** or **Operators** tabs to insert into the expression.
    - Type other items, such as register addresses or numeric values.
    - To mix data types, you must explicitly convert them. For example, you cannot directly assign a fractional value to an integer variable. You must use the data type conversion functions.
    - If you enter multiple expressions in the Expression command, each expression must end with a semi-colon. See the examples below.
    - If the expression is invalid, the Expression Builder give an error. When no errors are listed, the expression is valid.
3. Click **OK** to close the dialog and enter the expression in the command.

Using Boolean Values in Condition Expressions

To test a boolean value, such as discrete inputs, it can be compared to True or False, or simply placed by itself, since it is already a boolean value. The two expressions below are identical:

```
MyInput1 = True OR MyInput2 = False

MyInput1 OR NOT MyInput2
```

Testing Bits in DWORD Tags

Appending the Bit Number

To test an individual bit in a DWORD tag, append the bit number or bit name to the tag name. For example:

```
_Axis[0].Status.TGDone
MyDWORD.0
```

Bit Masking

Individual bits in DWORD tags can be tested by using bit masking. The New Condition wizard uses this method for multiple bits. The example below shows how to check if both bits 0 and 7 are set in the variable MyDWORD. The parentheses must be present. Due to the order of precedence, the "=" would be evaluated first if the parentheses were not present.

```
(MyDWORD AND 16#00000081) = 16#00000081
```

See Also

Expressions Overview | Assignment Expressions | Operators | Functions

5.13.4. Value Expressions

A value expression evaluates to a number. Value expressions are used in the Delay Link Type, in command parameters in user programs, and to programmatically select commanded axes in user programs.

Value expressions can be very simple, for example just a tag name, such as Axis[1].ActPos.

Sample Expressions

Example 1

```
MyPosition + 10
```

Note: MyPosition must exist as a variable.

Example 2

```
_Axis[1].ActPrs - _Axis[0].ActPrs
```

Example 3

```
3.0 + ABS(_Axis[].ActPos);
```

Entering an Expression

Delay Link Type

1. In the Delay Link Type, double-click the **Time to Delay** box. The Expression Editor will open.
2. Enter the expression:
    - Double-click any item in the lists on the **Tags**, **Functions** or **Operators** tabs to insert into the expression.
    - Type other items, such as register addresses or numeric values.
    - To mix data types, you must explicitly convert them. For example, you cannot directly assign a fractional value to an integer variable. You must use the data type conversion functions.
    - If you enter multiple expressions in the Expression command, each expression must end with a semi-colon. See the examples below.
    - If the expression is invalid, the Expression Builder give an error. When no errors are listed, the expression is valid.
3. Click **OK** to close the dialog and enter the expression in the command.

Command Parameters

1. In a command parameter in a user program, double-click the box.
   Notice that the Expression Editor is not available when entering expressions in command parameters. it is good practice to only use short expressions in Command Parameters. If you need a long expression in a command parameter, consider creating a variable instead that is calculated in an expression command, and then entered in the command parameter.
2. Enter the expression:
    - Type your expression. The intellisense will offer suggestions for variables and tag names based on what you have typed.
    - If the expression is invalid, the portion in error will be red.
3. Press Enter to finish editing. If the expression is invalid, the entire expression text will be red. When the entire expression is in black text, the expression is valid.

See Also

Expressions Overview | Constants | Functions | Operators | If Statement | Keywords

5.13.5. Local Variables in User Program Steps

Local variables can be declared in user programs steps. Variables declared in this way can only be accessed within that step. The variable values are valid only for the execution of that step. When the task jumps to another step, the local variable values are not retained, even if the task jumps to the same step.

Local variables are different from variables in the variable table. For complex programs, local variables may significantly reduce the number of variables needed in the variable table.

Adding Local Variables to a Step

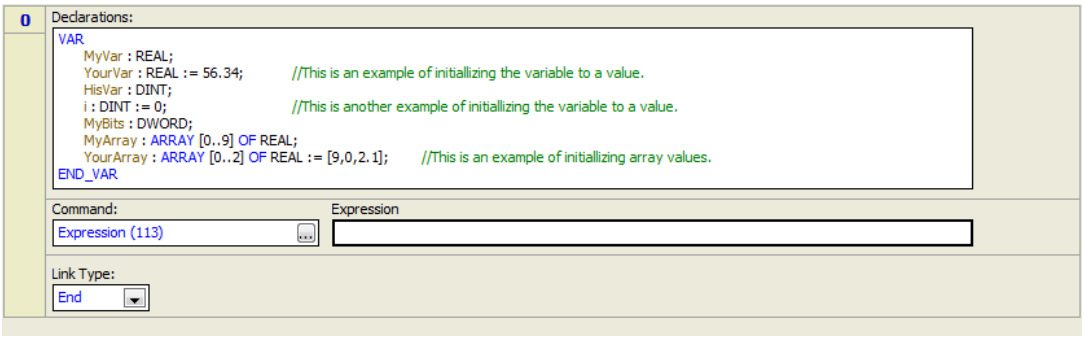
To add local variables to a user program step:

1. On the toolbar, click the **Add Step Declarations** button. Or, right-click the step and choose **Add Step Declarations**.
2. A **Declarations** section will appear.

3. Declare your variable between the **VAR** and **END_VAR** keywords, as shown in the example below.

   Variables can be of REAL, DINT, or DWORD data types, and can be also be fixed-length arrays. Variables can optionally be initialized to a value upon declaration. Variables that are not initialized when declared will default to the value 0.

Example:



Usage

Local variables in a user program step can be used in Expression (113) commands, in command parameters, and in link conditions.

Arrays

Bounds

The lower and upper array bounds are given in brackets. For example, bounds of [0..6] declare an array of 7 items. The upper and lower bounds of the array can be any value, including negative values. For most applications, the lower bound will be zero.

Initializing

Array values can be initialized as shown in the example above. The number of initialized values must equal the number of items in the array.

Parentheses can be used as a repetition factor, where the number preceding the parentheses specifies the number of repetitions.
Examples:

[8,2,3(0),10] is the same as [8,2,0,0,0,10]

[10(0)] initializes an array of length 10 to all zeros.

Size Limits

The maximum length of a local array is 32. A single user program step can have up to 128 items declared, including variables and individual array items.

See Also

Expressions Overview | Expression (113) | Creating User Programs

5.13.6. Arrays

An **array** is a numerically indexed sequence of elements of the same data type. The elements of an array are specified by the index. In the RMC, an array index starts at 0 and extends to the number of elements minus 1. The RMC supports only one-dimensional arrays. For an example on using arrays, see Example: Using Arrays.

This topic treats arrays in the Variable Table. For information on local arrays in user program steps, see Local Variables in Expressions. For information on arrays in User Functions, see the User Functions and Declaring Variables in User Functions topics.

Array indexing can only be used within the RMC expressions. When accessing RMC arrays from a host controller, you must use the register addresses for the protocol you are using.

Creating Arrays

Before declaring an array, make sure there is room in the Variable Table for it. If the new array will overwrite existing variables, RMCTools will warn you and ask whether to continue or not.

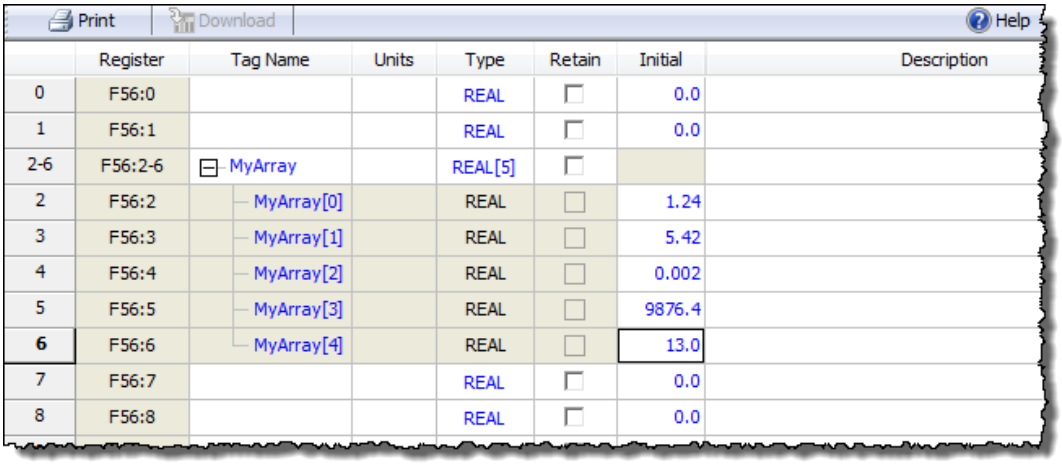
To create an array in the variable table:

1. Open the Variable Table Editor.
2. On the **Edit** tab, click the **Type** cell for some variable, then click the ellipsis button ( ).
3. Choose the **Data Type** and the **Size** of the array, then click **OK**.
4. For each element in the array, you can set the initial value in the same manner as for any variable.

If you plan on using a variable to index through the array, you should create it also. Make sure it is a DINT.

Example

An array of 5 REALs starting at variable 2:



	Register	Tag Name	Units	Type	Retain	Initial	Description
0	F56:0			REAL		0.0	
1	F56:1			REAL		0.0	
2-6	F56:2-6	MyArray		REAL[5]			
2	F56:2	MyArray[0]		REAL		1.24	
3	F56:3	MyArray[1]		REAL		5.42	
4	F56:4	MyArray[2]		REAL		0.002	
5	F56:5	MyArray[3]		REAL		9876.4	
6	F56:6	MyArray[4]		REAL		13.0	
7	F56:7			REAL		0.0	
8	F56:8			REAL		0.0	

Indexing Array Elements

To specify an element of an array, insert the index in brackets after the tag name in the format 'array_name[index]'. The index can be an expression and **must** evaluate to a DINT data type.

Examples

Definitions	Indexing Examples	Description
MyArray as REAL[20]	MyArray[4]	Specifies the element of the array with the index 4. This is actually the fifth element because the indices are zero-based.
MyArray as REAL[10] Myindex as DINT	MyArray[MyIndex]	The value of 'Myindex' specifies the element in 'MyArray'.

MyArray as DINT[5] MyVar as REAL	MyArray[REAL_TO_DINT (MyVar +5.0)]	If you wish to use a REAL value as an index, you can convert it to a DINT as illustrated here.
	_VarTbl.CurVal[2]	You can index variables as shown here, but it is much better to first assign a tag name to a variable, then use the tag name, as shown in the MyArray example above.
	_Axis[0]	Specifies the first axis. Note that '_Axis[0]' is not a valid data type by itself. You must specify a tag, such as _Axis[0].ActPos.
i as DINT	_Axis[i+2]	An index can be an expression, but must evaluate to a DINT.

Special Case: _Axis[] with no specified element

In the user programs, the "_Axis[]" tag can be used without any value in the brackets. In this case, the default axis of the task that is running the user program will be used. This is for advanced users only. See the **Default Axis** section of the Tasks topic for details.

Special Case: _Task[] with no specified element

In the user programs, the "_Task[]" tag can be used without any value in the brackets. In this case, the task that is running that user program will be used. This is for advanced users only.

For example, you can use _Task[].CurAxis to change the default axis of the task.

Array Functions

The following functions are intended for handling arrays:

- **LENGTH(*Array*)**

  Returns the number of elements in the ***Array***.

  For example:

  LENGTH(MyArray)

- **FILL(*To, Val, Len*)**

  Sets ***Len*** registers starting at the ***To*** address to the specified value (***Val***).

  For example, in the array MyArray, to set elements 0-9 to 1:

  FILL(MyArray[0],1,10)

See Also

Expressions Overview | Example: Using Arrays

5.13.7. Operators

This topic describes the operators available in the RMC expressions. For details on using the operators, see the Assignment Expressions and Logical Expressions topics.

Operator	Description	Operates on these Data Types	Notes
:=	Assigns the value of the expression on the right-hand side to the register on the left-hand side. Required in all assignment expressions.	All Data Types	The assignment operator is required in the Expression command.
+	Addition	REAL, DINT	
-	Subtraction	REAL, DINT	
*****	Multiplication	REAL, DINT	
/	Division	REAL, DINT	
MOD	Integer Modulo Gives the remainder of a division of whole numbers.	DINT	Examples: 12 MOD 5 will return 2. 61 MOD 10 will return 1. 5 MOD 5 will return 0.
(	Opening parenthesis.	All Data Types	The number of opening parentheses must match the number of closing parentheses.
)	Closing parenthesis.	All Data Types	The number of opening parentheses must match the number of closing parentheses.
=	Equal To (not for assigning values)	All Data Types	Cannot be used for assignment.
<>	Not Equal	All Data Types	
<=	Less Than or Equal To	REAL, DINT	
<	Less Than	REAL, DINT	
>	Greater Than	REAL, DINT	
>=	Greater Than or Equal To	REAL, DINT	
AND	Logical and Bitwise AND	DWORD, DINT, BOOL	The data types of the operands must match.
OR	Logical and Bitwise OR	DWORD, DINT, BOOL	The data types of the operands must match.
NOT	Logical and Bitwise NOT	DWORD, DINT, BOOL	The data types of the operands must match.
XOR	Logical and Bitwise Exclusive OR	DWORD, DINT, BOOL	The data types of the operands must match.
[	Opening bracket for indexing item in an array, e.g. MyVar[i] or _Axis[0].ActPos.	All Data Types	The number of opening brackets must match the number of closing brackets.
]	Closing bracket for indexing item in an array, e.g. MyVar[i] or _Axis[0].ActPos.	All Data Types	The number of opening brackets must match the number of closing brackets.

**	Exponentiation	REAL	Example: 3**6 is three to the sixth power. If you are calculating polynomials, consider using the POLY function.

Order of Precedence

Precedence	Operation	Symbol
HIGHEST	Parenthesization	(expression)
	Function Evaluation	identifier(argument list) ln(a), max(x,y) etc.
	Negation Complement	- NOT
	Exponent	**
	Multiply Divide Modulo	* / MOD
	Add Subtract	+ -
	Comparison	< , > , <= , >=
	Equality Inequality	= <>
	Boolean AND	AND
	Boolean Exclusive OR	XOR
	Boolean OR	OR
Lowest	Assignment	:=

See Also

Expressions Overview | Functions | Data Types

5.13.8. Keywords

This topic describes the keywords available in the RMC expressions. Keywords are reserved for the uses indicated in the table below and cannot be used as names for user-defined items, such as variable names, input names, etc.

In the Expression Editor, these can be accessed on the **Keywords** tab.

Keyword	Description

IF	Used in an If Statement.
THEN	Used in an If Statement.
ELSE	Used in an If Statement.
ELSIF	Used in an If Statement. Note: ELSEIF is also supported.
ENDIF	Used in an If Statement.
TRUE	boolean true (1 in binary)
FALSE	boolean false (0 in binary)

See Also

Expressions Overview | Functions | Data Types

5.13.9. IF Statement

The IF statement can be used in the Expression (113) command. It is used for conditional branching only within the Expression command.

Format

```
IF condition THEN
  statements
ELSIF condition THEN
  statements
ELSE
  statements
END_IF
```

Notes

Each *statement* must end with a semicolon (;).

An ELSE cannot precede an ELSIF.

The ELSE and ELSIF are optional.

The spelling ELSIF and ELSEIF are both supported and are equivalent.

Example

Example 1

```
IF _Axis[0].ActPos > 20.0 THEN
  MyREAL := 92;
END_IF
```

Example 2

```
IF MyInput1 = True THEN
  MyREAL1 := 34.0;
  MyREAL2 := 70023.0;
ELSE
  MyDINT := 2;
END_IF
```

Example 3

```
IF _Axis[0].StatusBits.InPos = True THEN
  MyREAL1 := 34.0;
  MyREAL2 := 70023.0;
ELSIF ABS(_Axis[0].ActPos) > 20.0 THEN
  %QX0.1 := True;
ELSE
  MyDINT := 2;
END_IF
```

See Also

Expressions Overview

5.13.10. Constants

The term "constants" refers an numeric item in programming that does not change. This can be a number in typical representation, such as 10.345, or a named representation of a number, such as PI or *e*.

Constant Number Representations

The expressions in RMCTools supports the following types of constant number representations:

Constant Type	Description	Notation
Decimal	Base 10 numbers. These can be either DINT (32-bit integers) or REAL (32-bit floating-point decimal) numbers.	*number* **Examples:** 1 156.8902 0.0034
Hexadecimal	Base 16 numbers. These are 32-bit DWORD numbers.	16#*number* **Examples:** 16#F3 16#0000104C Hexadecimal numbers are sometimes represented with a preceding "0x" instead of 16#, for example 0x01006a01.
Boolean	Representing a single bit. 0 = False and 1= True.	0 or 1 or True or False

Special Constants

The expressions in RMCTools include the named constant numbers listed below. In the Expression Editor, these can be accessed on the **Tags** tab, under **Expression Constants**.

Constant	Value

M_PI	3.1415927 (limited to 32-bit floating point accuracy)

See Also

Expressions Overview | Functions | Data Types

5.13.11. Comments

Expressions support comments. Comments are very useful for telling the user (and yourself) what the code is doing. Commented text will not be executed or checked for errors. Comments are saved as part of the expression.

The Expression command supports two types of comments:

Type	Description
//	Comments everything to the right on the same line.
(* *)	Comments everything between the starred brackets. This comment type cannot be nested. For example, (* (* sample comment *) *) will be an error.

Example 1

```
IF _Axis[0].ActPos > 20.0 THEN    //This is a comment.
  MyREAL := 92;                   //This is a comment, too.
END_IF
```

Example 2

```
MyREAL1 := 34.0;        (*This is a comment*)
MyREAL2 := 70023.0;
(*This is a multi-line comment
that spans multiple lines *)
MyDINT := 2;
```

See Also

Expressions Overview

5.13.12. Limitations of 32-bit Numbers

When performing mathematical calculations in the RMC, keep in mind that the numbers are 32-bit numbers. 32-bit numbers do not provide infinite resolution or range. This topic describes some of these limitations and how to work around them.

DINT Numbers

DINT numbers are 32-bit integers. DINTs range from -2,147,483,648 to +2,147,483,647. If a result exceeds these limits, the number will wrap.

REAL Numbers

The RMC's REAL data type conforms to the IEEE-754 Floating-Point specification. Real numbers do not have infinite accuracy. They are limited to a certain number of significant digits. The primary effect is that adding a small number to a large number does not result in great accuracy.

In the RMC75E, RMC75S, RMC75P, RMC150, and RMC200, the 32-bit REAL numbers range is -340.28235E+36 to 340.28235E+36. The smallest number that can be represented is 1.4013E-45.

The RMC75E, RMC150E, and RMC200 support Inf, -Inf, and NaN as described below.

Incrementing a Value

Continually incrementing a value by some small amount in a loop can cause problems. For example, incrementing a variable *i* by 0.001 will eventually cause resolution problems. After reaching a large value, such as 1000, adding 0.001 will not result in exactly 1000.001 due to the resolution limitation.

To avoid this problem, it is possible to increment an integer by 1, then multiply by 1000. Or, if possible, use algorithms that restart the counter periodically, wrap integers, or employ modulo arithmetic.

Not a Number (NaN)

NaN is a symbol that is produced as the result of an operation on invalid input operands.

The RMCs will fault the task if an invalid input to the following functions and operands is detected: SQRT, LN, LOG, ASIN, ACOS, /, and MOD.

Otherwise, the RMC75E, RMC150E, and RMC200 will generate NaN when the result is truly unknown or undefined, which includes the following:

- Any operation where NaN is in at least one operand.
- The divisions ∞/∞, -∞/∞, ∞/-∞, and -∞/-∞.
- The multiplications 0x∞ and 0x-∞.
- The additions ∞+(-∞) and (-∞)+∞ and equivalent subtractions.
- Applying a function arguments outside of its domain, including taking the tangent of an odd multiple of 90 degrees.

The RMC75S and RMC75P do not have NaN. These controllers will return a saturated value (such as -340.28235E+36 for log10(-50)) or 0 when they have an invalid result.

Infinity (Inf)

The RMC75E, RMC150E, and RMC200 will generate Inf or -Inf when the result is infinity or negative infinity or the result overflows the 32-bit limits. Examples: division by zero, multiplication by infinity

The RMC75S and RMC75P do not have Inf or -Inf. These controllers will return a saturated value, either 340.28235E+36 or -340.28235E+36 when the result is infinity or negative infinity.

See Also

Expressions Overview

5.13.13. Troubleshooting Expressions

The Expression Editor displays the Assignment Expression in black text if it is valid, and in red text when it is invalid. Listed below are common errors in the assignment expression syntax:

Mixed Data Types:

Data types cannot be mixed in the assignment expression without explicitly converting them. A number that *does not* contain a decimal point is considered to be a DINT data type. A number that *does* contain a decimal point is considered to be a REAL data type.

Example1:

Assume **MyVariable** is defined to be a REAL type. The following expression *is not* valid, because "3" is a DINT type:

MyVariable:=_Axis[0].ActPos+3

The following expression *is* valid because "3.0" is a REAL type:

MyVariable:=_Axis[0].ActPos+3.0

The following expression *is* valid because the DINT_TO_REAL() function converts "3" to a REAL type:

MyVariable:=_Axis[0].ActPos+DINT_TO_REAL(3)

Example2:

Assume **MyVariable** is defined to be a DINT type. The REAL_TO DINT() function converts the REAL type part of the expression to a DINT. The following expression *is not* valid because "2" is a DINT type and _Axis[0].ActPos is a REAL type:

MyVariable:=REAL_TO_DINT(Abs(_Axis[0].ActPos/2))

The following expression *is* valid because "2.0" is a REAL type and _Axis[0].ActPos is also:

MyVariable:=REAL_TO_DINT(Abs(_Axis[0].ActPos/2.0))

Similarly, the following expression is *not* valid because the Abs() function returns the data type of its arguments, which in this case is REAL, while MyVariable is a DINT type:

MyVariable:=Abs(_Axis[0].ActPos/2.0)

Illegal Expression Order

An expression in the Expression (113) command *must* begin with "[tag]:= ", where *tag* is any register in the RMC and may be given as a register address, tag name, or variable name. See the Assignment Expression topic for details.

Inaccurate Results

If the results of some expressions are not precisely correct, it may be caused by truncation due to the 32-bit limit of RMC data.

See Also

Expressions Overview

5.14. Functions

5.14.1. Functions Overview

Functions can be used in the RMC expressions. Functions typically perform some calculation and return a result.

The RMC supports a number of built-in standard functions. Custom user functions can also be created.

See Also

standard functions | user functions

5.14.2. Standard Functions

5.14.2.1. Standard Functions

The RMC expressions support many built-in standard functions, as listed below. You can also create custom User Functions.

General Math Functions	Description
ABS(*a*)	Returns the absolute value of ***a***.
EXP(*a*)	Returns natural (*e*) raised to the $\boldsymbol{a}^{th}$ power.
LN(*a*)	Returns the natural logarithm (base *e*) of ***a***.
LOG(*a*)	Returns the logarithm (base 10) of ***a***.
SQRT(*a*)	Returns the square root of ***a***.
POLY(*t, a, b, c, d,...*)	Returns the polynomial calculation up to 8 coefficients $\boldsymbol{a} + \boldsymbol{bt} + \boldsymbol{ct}^2 + \boldsymbol{dt}^3$...
SIGNUM(*a*)	Returns -1 if ***a*** is negative, +1 if ***a*** is positive, and 0 if ***a*** is zero.

Rounding Functions	Description
ROUND(*a*) ROUND(*a, n*)	Rounds ***a*** to ***n*** decimal places.
MROUND(*a, multiple*)	Rounds ***a*** to the desired ***multiple***.
TRUNC(*a*)	Rounds ***a*** to an integer towards zero. Return type is DINT.
TRUNC_REAL(*a*)	Rounds ***a*** to an integer towards zero. Return type is REAL.
CEIL(*a*)	Rounds ***a*** to the next greater (most positive) integer.
FLOOR(*a*)	Rounds ***a*** to the next lesser (most negative) integer.

Selection Functions	Description
MIN(*a, b, ...*)	Returns the smallest value of the parameters (up to 10 operands).
MAX(*a, b, ...*)	Returns the largest value of the parameters (up to 10 operands).
LIMIT(*Min, InVal, Max*)	Limits ***InVal*** to the lower limit ***Min*** and the upper limit ***Max***.
SEL(*cond, a, b*)	In-line IF statement. If ***cond*** is true, returns ***a***, otherwise returns ***b***.

Trigonometric and Hyperbolic Functions	Description
SIN(*a*)	Returns the sine of ***a***.

COS(*a*)	Returns the cosine of ***a***.
TAN(*a*)	Returns the tangent of ***a***.
ASIN(*a*)	Returns the arcsine of ***a***.
ACOS(*a*)	Returns the arccosine of ***a***.
ATAN(*a*)	Returns the arctangent of ***a***.
SINH(*a*)	Returns the hyperbolic sine of ***a***.
COSH(*a*)	Returns the hyperbolic cosine of ***a***.
TANH(*a*)	Returns the hyperbolic tangent of ***a***.

Bit String Functions	**Description**
SHR(*a, n*)	Shifts the bits in ***a*** to the right ***n*** times. Zeros are shifted in.
SHL(*a, n*)	Shifts the bits in ***a*** to the left ***n*** times. Zeros are shifted in.
ASHR(*a, n*)	Shifts the bits in ***a*** to the right ***n*** times. The sign bit is shifted in on the left.
ROR(*a, n*)	Rolls the bits in ***a*** to the right ***n*** times.
ROL(*a, n*)	Rolls the bits in ***a*** to the left ***n*** times.

Curve Functions	**Description**
CRV_FIRST_X(*id*)	Returns the x value for the first point in the curve with the specified ***id***.
CRV_LAST_X(*id*)	Returns the x value for the last point in the curve with the specified ***id***.
CRV_INTERP_Y(*id, x, [options]*)	Interpolates the curve with the specified ***id*** at the given value of ***x***, and returns the Y value of the curve at that point.
CRV_INTERP_V(*id, x, [options]*)	Interpolates the curve with the specified ***id*** at the given value of ***x***, and returns the V (velocity) value of the curve at that point.
CRV_INTERP_A(*id, x, [options]*)	Interpolates the curve with the specified ***id*** at the given value of ***x***, and returns the A (acceleration) value of the curve at that point.
CRV_EXISTS(*id*)	Returns TRUE if the curve with the specified ***id*** exists, otherwise returns FALSE.

Array Functions	**Description**
LENGTH(*array*)	Returns the number of elements in the ***array***.
FILL(*to, val, len*)	Fill ***len*** values with ***val*** starting at ***to***.

Type Conversion Functions	**Description**
REAL_TO_DINT(*a*)	Rounds ***a*** to the nearest integer.
DINT_TO_REAL(*a*)	Returns ***a*** in REAL type.
DWORD_TO_DINT(*a*)	***a*** and its bit order remains the same, but type is changed to DINT.

DINT_TO_DWORD(*a*)	***a*** and its bit order remains same, but type is changed to DWORD.
TRUNC(*a*)	Rounds ***a*** to an integer towards zero.

Other Functions	Description
ADDR_OFS(*loc*, *i*)	Returns the address of the ***i***th register after the register ***loc***.
COPY(*src*, *dst*, *len*)	Copies up to 32 variables from ***src*** to ***dst***.
LOG_EVENT(*a, ...*)	Logs the values of the operands ***a***, ... in the Event Log. From 1 to 3 operands.
REG_REAL(*f, e*)	Represents the register at the specified address %MD*f.e*.
REG_DINT(*f, e*)	
REG_DWORD(*f, e*)	

See Also

Expressions Overview | User Functions | Operators | Data Types

5.14.2.2. ABS Function

ABS(*a*)

Returns the absolute value of ***a***.

Parameters

***a* (REAL or DINT)**

The input value.

Return Value

The absolute value of ***a***. The data type will be the same as the ***a*** input parameter.

Examples

ABS(3) returns 3

ABS(-3.0) returns 3.0

See Also

Standard Functions

5.14.2.3. ACOS Function

ACOS(*a*)

Returns the arccosine of ***a***.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL in radians.

Remarks

Notice that the return value is in radians. To convert a value from radians to degrees, multiply the radians value by 180/π. The RMC has an M_PI constant for π. For ***a*** < -1.0 or ***a*** > 1.0, the task will fault. An error will be logged in the Event Log and the user program running on the task will stop.

Examples

ACOS(0.5) returns 1.0471976

ACOS(0.5) * 180 / M_PI returns 60.0

See Also

COS Function | ASIN Function | ATAN Function | Standard Functions

---

5.14.2.4. ADDR_OFS Function

ADDR_OFS(*location, i*)

Returns the address of the ***i***th register after ***location***. This function is intended only for use with the COPY function.

Parameters

***Location* (Address)**
The address of the register location.

***i* (DINT)**
The offset, in 32-bit registers, from the ***location***.

Return Value

Returns the address of the *i*th register after the register ***location***. This address cannot be used directly, but is intended to be passed into one of the COPY function's parameters.

Remarks

The ADDR_OFS function can be used to calculate an address from a base address and an offset when using the COPY function. For example, if the first tag in the block is called MyFirstTag, then the address of a tag located 4 registers after MyFirstTag can be calculated with:

```
ADDR_OFS(MyFirstTag, 4)
```

The ADDR_OF function is especially useful when copying from large blocks of data that are not declared as arrays because it lends itself well to looping and calculating addresses during each loop. However, best programming practice is usually to set up the data in arrays, and use the COPY function on those arrays, which makes the ADDR_OFS unnecessary.

The Address parameter may not be a local variable. Therefore, the ADDR_OFS function may not be used in User Functions.

Examples

See the COPY function for an example.

See Also

COPY | Standard Functions

5.14.2.5. ASHR Function

ASHR(*a*, *n*)

Shifts the bits in ***a*** to the right ***n*** times. The sign bit is shifted in on the left.

Parameters

***a* (DINT or DWORD)**

The value to shift.

***n* (DINT)**

The number of times to shift ***a***. ***n*** must be between 0 and 31.

Return Value

Returns the same data type as ***a***.

Remarks

The result of an invalid ***n*** is undefined.

Examples

ASHR(16#FFFF0008, 2) returns 16#FFFFC0002

See Also

SHR Function | SHL Function | ROL Function | ROR Function | Standard Functions

5.14.2.6. ASIN Function

ASIN(*a*)

Returns the arcsine of ***a***.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL in radians.

Remarks

Notice that the return value is in radians. To convert a value from radians to degrees, multiply the radians value by 180/π. The RMC has an M_PI constant for π. For ***a*** < 1.0 or ***a*** > 1.0, the task will fault. An error will be logged in the Event Log and the user program running on the task will stop.

Examples

ASIN(0.5) returns 0.5235988

ASIN(0.5) * 180 / M_PI returns 30.0

See Also

SIN Function | ACOS Function | ATAN Function | Standard Functions

5.14.2.7. ATAN Function

ATAN(*a*)

Returns the arctangent of ***a***.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL in radians.

Remarks

Notice that the return value is in radians. To convert a value from radians to degrees, multiply the radians value by 180/π. The RMC has an M_PI constant for π.

Examples

ATAN(0.5) returns 0.4636476

ATAN(0.5) * 180 / M_PI returns 26.56505

See Also

TAN Function | ASIN Function | ACOS Function | Standard Functions

5.14.2.8. CEIL Function

CEIL(*a*)

Rounds ***a*** to the next greater (most positive) integer.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL.

Examples

CEIL(12.1) returns 13

CEIL(-5.2) returns -5

See Also

FLOOR Function | Standard Functions

5.14.2.9. COPY Function

COPY(*src*, *dst*, *len*)

Copies up to 32 contiguous registers from ***src*** to ***dst***. Both ***src*** and ***dst*** must be located in the variable table. ***Len*** specifies the number of registers to be copied.

This command is useful when sending large blocks of data to the variable table via some communications types, and that data needs to be moved around. This applies especially to PROFIBUS, PROFINET or EtherNet/IP I/O modes that use the variable table.

Parameters

***Src* (Address)**
The address of the first item of the source data to be copied. The source must a location in the variable table.

***Dst* (Address)**
The address of the first item of the destination to which data will be copied. The destination must be a location in the variable table.

***Len* (DINT)**
The number of registers to be copied. A maximum of 32 registers can be copied. Each register is 32 bits.

Returns

The COPY function does not return a value.

Remarks

The data types of the ***src*** and ***dst*** registers do not need to match. COPY preserves the bits in the registers that are copied. That is, it does not convert the data.

Best programming practice is to set up the data in arrays, and copy from and to arrays. This is usually the cleanest and most manageable method. The RMC also provides the ADDR_OFS function to help copying blocks of individual variables. See **Using COPY with the ADDR_OFS Function** below for details and examples.

The RMC *will not* range-check arrays in the Variable Table when copying to or from them. If the ***len*** is longer than the array, it will copy beyond the end of the array.

The RMC *will* range-check *local* arrays when copying to or from them. This includes arrays declared in user functions or in user program steps.

Notice that although the variable table is broken up into multiple individual files (%MD56, %MD57, etc.), the COPY function can wrap across these file boundaries. For example, copying 30 registers to %MD56.250 will place the first 6 items in %MD56.250 to %MD56.255, and the last 24 items in %MD57.0 to %MD57.23.

An invalid address parameter during compile will trigger a verify error. An invalid address parameter during runtime will trigger a program fault and stop the current task, and the copy will not be performed. This applies to ***src*** and ***dst*** and the entire range as defined by ***len***.

Multiple COPY statements can be used in a single Expression (113) command, each copying up to 32 registers. Like all assignment expressions, the COPY statement will execute completely before going to the next assignment or copy.

Examples

For the examples that follow, use the following variable table. Notice that i and Len are DINTs.

Variable	Tag Name	Type
	ProfiData	DWORD[16]
0	ProfiData[0]	DWORD
1	ProfiData[1]	DWORD
:	:	:
15	ProfiData[15]	DWORD
16	i	DINT
17	Len	DINT
18		REAL
19	Move1ReqPos	REAL
20	Move1Vel	REAL
21		REAL
	CurveData	REAL[500]
22	CurveData[0]	REAL
23	CurveData[1]	REAL
24	CurveData[2]	REAL
:	:	:
521	CurveData[499]	REAL

Example: Copying values

Copy two values from ProfiData[1] and ProfiData[2] to Move1ReqPos and Move1Vel:

```
COPY(ProfiData[1],Move1ReqPos, 2);
```

Example: Writing Curve Data

Writing a curve with 500 registers into CurveData via PROFIBUS. In this example, only 16 registers can be sent to the RMC at a time. A user program would do a loop that transfers the data from ProfiData[2] through ProfiData[15] in successive copies, with some handshaking done between each copy. The variable i must be initialized to zero before starting this loop. The loop assumed to stop when i reaches 500:

```
Len := MAX(500 - i, 14);
COPY(ProfiData[2], CurveData[i], Len);
i := i + Len;
```

Using COPY with the ADDR_OFS Function

The ADDR_OFS function can be used to calculate an address from a base address and an offset. For example, if the first tag in the block is called MyFirstTag, then the address of a tag located 4 registers after MyFirstTag can be calculated with:

ADDR_OFS(MyFirstTag, 4)

This is useful when copying from blocks of data that are not declared as arrays. However, best programming practice is usually to set up the data in arrays, and use the COPY function on those arrays, which makes the ADDR_OFS unnecessary.

Example: Simple

Consider a block of 30 miscellaneous parameters in the variable table, each as individual tags. Suppose the user wants to load these parameters using a 14-register PROFIBUS buffer. Therefore, on each subsequent cycle of receiving PROFIBUS data, the following three copy commands would be issued:

```
COPY(ProfiData[2], MyFirstTag, 14);
COPY(ProfiData[2], MyFifteenthTag, 14);
```

```
COPY(ProfiData[2], MyTwentyNinthTag, 2);
```

This would be difficult to maintain if a tag is inserted. Consider how the following is more maintainable:

```
COPY(ProfiData[2], ADDR_OFS(MyFirstTag,0), 14);
COPY(ProfiData[2], ADDR_OFS(MyFirstTag,14), 14);
COPY(ProfiData[2], ADDR_OFS(MyFirstTag,28), 2);
```

This method can also easily be used to make the length general and put in a loop.

Example: Looping from PROFIBUS Master

This example introduces an advanced and code-efficient manner of using the COPY and ADDR_OFS functions to copy large amounts of data via a small PROFUBUS buffer. Suppose the first data item of the PROFIBUS buffer contained length and offset information, packed into the lower and upper 16 bits of the register. The PROFIBUS master would control the offset and length of the writes. There would be no need to implement looping logic in the RMC, and the handshaking would be reduced.

The code could look like similar to this:

```
CopyLen := DWORD_TO_DINT(SHR(ProfiData[1], 16));
CopyOffset := DWORD_TO_DINT(ProfiData[1] AND 0xFFFF);
COPY(ProfiData[2], ADDR_OFS(MyFirstReg, CopyOffset), CopyLen);
```

This example could be further improved to be more safe. For example:

```
CopyLen := DWORD_TO_DINT(SHR(ProfiData[1], 16));
If ( CopyLen > 0 ) Then
   CopyOffset := DWORD_TO_DINT(ProfiData[1] AND 0xFFFF);
   If ( CopyOffset < MaxOffset AND CopyOffset + CopyLen <= MaxOffset ) Then
      COPY(ProfiData[2], ADDR_OFS(MyFirstReg, CopyOffset), CopyLen);
   Else
      LOG_EVENT(10, CopyOffset, CopyLen);
   End If
End If
```

The Log_Event function would provide the user with some indication of the fault.

See Also

ADDR_OFS | Standard Functions

5.14.2.10. COS Function

COS(*a*)

Returns the cosine of ***a***.

Parameters

***a* (REAL)**

The input value in radians.

Return Value

Returns a REAL.

Remarks

Notice ***a*** is in radians. To convert a value from degrees to radians, multiply the degrees value by π/180. The RMC has an M_PI constant for π.

Due to the 32-bit floating point limitations, COS(M_PI / 2) returns -43.71139E-9 instead of zero. For complex calculations, you may need to check for an exact zero and code the return value as needed.

Examples

COS(M_PI) returns -1.0

COS(45 * M_PI / 180) returns 0.70710677

See Also

SIN Function | TAN Function | ACOS Function | Standard Functions

5.14.2.11. COSH Function

COSH(*a*)

Returns the hyperbolic cosine of ***a***.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL.

See Also

SINH Function | TANH Function | Standard Functions

5.14.2.12. CRV_EXISTS Function

CRV_EXISTS (*id*)

Returns TRUE if the curve with the specified ***id*** exists, otherwise returns FALSE.

Parameters

***id* (DINT)**

The curve ID.

Return Value

Returns a BOOL.

See Also

CRV_INTERP Functions | CRV_FIRST_X Function | CRV_LAST_X Function | Standard Functions

5.14.2.13. CRV_FIRST_X Function

CRV_FIRST_X (id)

Returns the x value for the first point in the curve with the specified ***id***.

Parameters

***id* (DINT)**

The curve ID.

Return Value

Returns the first X value of the curve as a REAL.

Remarks

This is useful for moving to the first point of a curve before sending the Curve Start (86) or Curve Start Advanced (88) command.

If the specified curve ***id*** does not exist, the task will fault. An error will be logged in the Event Log and the user program running on the task will stop. Use the CRV_EXISTS function to programmatically check if the curve ***id*** exists.

See Also

CRV_LAST_X Function | CRV_INTERP Functions | CRV_EXISTS Function | Standard Functions

5.14.2.14. CRV_INTERP Functions

CRV_INTERP_Y (*id, x,* [*options*])

CRV_INTERP_V (*id, x,* [*options*])

CRV_INTERP_A (*id, x,* [*options*])

Interpolates the curve with the specified ***id*** at the given value of ***x***, and returns the Y value, velocity (V), or acceleration (A) at that point in the curve.

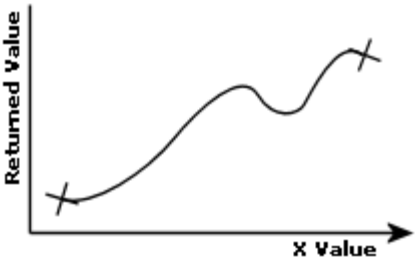
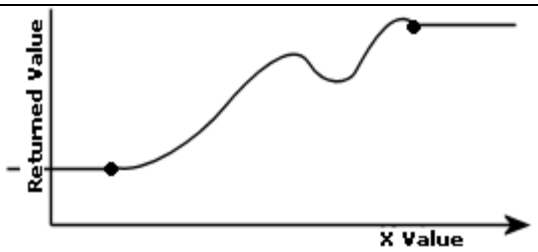
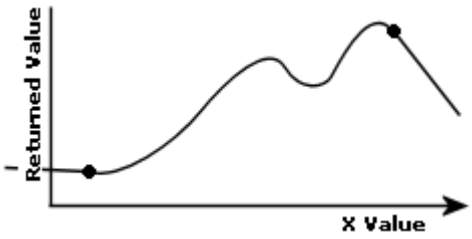
Parameters

***Id* (DINT)**
The curve identification number.

***X* (REAL)**
The x value of the specified curve.

***Options* (DINT)**
Optional parameter. Defines the behavior of the function when ***x*** is beyond the endpoints of the specified curve. The following options exist, similar to the Endpoint Behavior Options of the Curve Start Advanced (88) command:

Option	Description	Image

0: Fault (default)	If ***x*** is beyond the endpoints of the curve, a runtime error will occur, and the task will stop running.	
4: Truncated	If ***x*** is beyond the endpoints of the curve, the curve holds it position at each endpoint: CRV_INTERP_Y will return the Y value of the closest endpoint of the curve. CRV_INTERP_V will return zero. CRV_INTERP_A will return zero.	
8: Extrapolated	If ***x*** is beyond the endpoints of the curve, the curve will be extrapolated linearly, and that value returned: CRV_INTERP_Y will return the extrapolated Y value. The curve will be extrapolated linearly, using the velocity of the curve at the closest endpoint of the curve. CRV_INTERP_V will return the velocity of the closest endpoint of the curve. CRV_INTERP_A will return zero.	

Returns

The interpolated Y, velocity, or acceleration value of the specified curve at the given value of ***x***. The returned data type is REAL.

Remarks

These functions can be used for purposes such as:

- **Gain Scheduling**
  If the application requires different tuning gains based on some value, such as position, a curve can be created that defines the gain based on that value. A simple user program can then use the CRV_INTERP_Y interpolation function to continuously apply the gain. See Gain Scheduling for details.
- **Run a curve on a velocity axis**
  Velocity axes do not currently support curves directly. To run a curve on a velocity axis, first create a curve where the y values represent velocity. Make a simple user program that uses the CRV_INTERP_Y function to continuously update a variable. The user program should then gear the velocity axis to that variable with the Gear Velocity command.

If the specified curve ***Id*** does not exist or the ***Options*** value is invalid, the task will fault. The task will also fault if ***Options*** is zero and the ***X*** value is not within the range of x-values of the curve. When the task faults, an error will be logged in the Event Log and the user program running on the task will stop. Use the CRV_EXISTS function to programmatically check if the curve ***id*** exists.

The CRV_FIRST_X, CRV_LAST_X, AND CRV_EXISTS functions are useful in conjunction with the curve interpolation functions.

Examples

```
MyReal := CRV_INTERP_Y(3, 5.67, 8);
```

See Also

Gain Scheduling | CRV_FIRST_X | CRV_LAST_X | CRV_EXISTS | Standard Functions

5.14.2.15. CRV_LAST_X Function

CRV_LAST_X (id)

Returns the x value for the last point in the curve with the specified ***id***.

Parameters

id **(DINT)**

The curve ID.

Return Value

Returns the last X value of the curve as a REAL.

Remarks

If the specified curve ***id*** does not exist, the task will fault. An error will be logged in the Event Log and the user program running on the task will stop. Use the CRV_EXISTS function to programmatically check if the curve ***id*** exists.

See Also

CRV_FIRST_X Function | CRV_INTERP Functions | CRV_EXISTS Function | Standard Functions

5.14.2.16. DINT_TO_DWORD Function

DINT_TO_DWORD(*a*)
Converts a DINT to a DWORD. ***a*** and its bit order remains the same, but type is changed to DWORD.

Parameters

***a* (DINT)**

Return Value

Returns a DWORD.

See Also

DWORD_TO_DINT Function | DINT_TO_REAL Function | REAL_TO_DINT Function | Standard Functions

5.14.2.17. DINT_TO_REAL Function

DINT_TO_REAL (*a*)
Converts a DINT to a REAL.

Parameters

***a* (DINT)**
The input value.

Return Value

Returns a REAL.

Remarks

For values beyond +/- 16,777,216 some rounding will occur since REAL values are only precise to 24 bits of precision. For example, 17,000,003 will become 17,000,004.0.

Examples

DINT_TO_REAL(5) returns 5.0

See Also

REAL_TO_DINT Function | DINT_TO_DWORD Function | DWORD_TO_DINT Function | Standard Functions

5.14.2.18. DWORD_TO_DINT Function

DWORD_TO_DINT(*a*)
Converts a DWORD to a DINT. ***a*** and its bit order remains the same, but type is changed to DINT.

Parameters

***a* (DWORD)**
The input value.

Return Value

Returns a DINT.

See Also
DINT_TO_DWORD Function | DINT_TO_REAL Function | REAL_TO_DINT Function | Standard Functions

---

5.14.2.19. EXP Function

EXP(*a*)
Returns natural (*e*) raised to the a^{th} power.

Parameters

***a* (REAL)**
The input value.

Return Value

Returns a REAL.

Examples

EXP(1.0) returns 2.7182817
EXP(-1.0) returns 0.36787945

See Also
Standard Functions

5.14.2.20. FILL Function

FILL(*to, value, length*)
Sets ***length*** registers starting at the ***to*** address to the specified value (***value***). Typically used to fill an array with values. A maximum of 32 registers can be filled.

Parameters

***to* (Address)**
The address to begin filling at. Typically an array element.
***value* (REAL, DINT, or DWORD)**
The value to set the elements to.
***length* (DINT)**
The number of elements to fill. A maximum of 32 registers can be filled.

Return Value

The FILL function does not return a value.

Remarks

The FILL function is typically used to fill an array with the same value, for example, setting all array items to zero. The ***to*** location can be any location in the Variable Table or an element in a local array.

If the ***to*** location is a local array or an element of a local array, then if ***length*** exceeds the end of the local array, the task will fault, an error will be logged in the Event Log and the user program running on the task will stop.

If the ***to*** location is in the Variable Table, ***length*** values will be written, even if it exceeds the end of an array. If ***length*** exceeds the end of the Variable Table, the task will fault, an error will be logged in the Event Log and the user program running on the task will stop.

A maximum of 32 registers can be filled. To fill more than 32 registers, use multiple FILL statements.

Examples

Set each of the first 4 elements in an array to one (1):

```
FILL(MyArray, 1, 4);
```

Set all elements in an array to zeroes:

```
FILL(MyArray, 0, LENGTH(MyArray));
```

Set each of elements 5 to 9 in an array to 100:

```
FILL(MyArray[5], 100, 5);
```

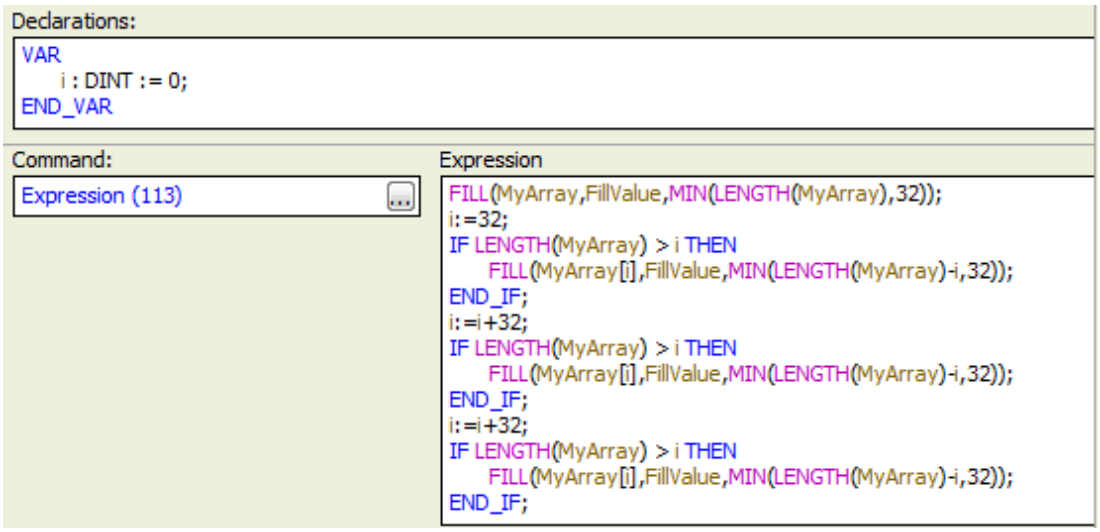
Fill an array of unknown size:

This example assumes the following variable have been declared in the Variable Table:

- **MyArray**: an array
- **FillValue**: the value with which to fill the array

This example is limited to arrays of length 128. To increase the max size, simply duplicate the last 4 lines in the expression as many times as necessary.

Caution: It is possible to exceed the allotted execution time for the step. However, the Verify will catch this prevent downloading to the RMC. If you do exceed the loop time, you will need to spread the code over several steps, in which case the variable **i** needs to be defined in the Variable Table, not locally in the step.



Copyable Text for the Expression Above:

```
FILL(MyArray,FillValue,MIN(LENGTH(MyArray),32));
i:=32;
IF LENGTH(MyArray) > i THEN
	FILL(MyArray[i],FillValue,MIN(LENGTH(MyArray)-i,32));
END_IF;
i:=i+32;
IF LENGTH(MyArray) > i THEN
	FILL(MyArray[i],FillValue,MIN(LENGTH(MyArray)-i,32));
END_IF;
i:=i+32;
IF LENGTH(MyArray) > i THEN
	FILL(MyArray[i],FillValue,MIN(LENGTH(MyArray)-i,32));
END_IF;
```

See Also

Functions Overview | Arrays | LENGTH Function

5.14.2.21. FLOOR Function

FLOOR (*a*)

Rounds ***a*** to the next lesser (most negative) integer.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL.

Examples

FLOOR(5.8) returns 5
FLOOR(-17.8) returns -18

See Also

CEIL Function | Standard Functions

5.14.2.22. LENGTH Function

LENGTH(*array*)
Returns the number of elements in the **Array**.

Parameters

***array* (Array Name)**
The input array.

Return Value

Returns a DINT.

Examples

```
MyVar := Length(MyArray);
```

See Also

Arrays | Standard Functions

5.14.2.23. LIMIT Function

LIMIT(*Min, InVal, Max*)
Limits ***InVal*** to the lower limit ***Min*** and the upper limit ***Max***. If ***InVal*** falls between ***Min*** and ***Max***, returns ***InVal***. If ***InVal*** is greater than ***Max***, returns ***Max***. If ***InVal*** is less than ***Min***, returns ***Min***.

Parameters

***Min* (REAL or DINT)**
The lower limit.
***InVal* (REAL or DINT)**
The input value to be limited.
***Max* (REAL or DINT)**
The upper limit.

All three input parameters must have the same data type.

Return Value

Returns the same data type as the input parameters.

Examples

LIMIT(1, _Axis[0].ActPos, 10) returns the Axis 0 Actual Position if it the position is between 1 and 10. Returns 1 if the position is less than 1. Returns 10 if the position is greater than 10.

See Also

Standard Functions

5.14.2.24. LOG Function

LOG(*a*)

Returns the logarithm (base 10) of ***a***.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL.

Remarks

If ***a*** is less than zero, the task will fault. An error will be logged in the Event Log and the user program running on the task will stop.

Examples

LOG(1000.0) returns 3.0.

LOG(15.0) returns 1.1760913.

See Also

Standard Functions

5.14.2.25. LN Function

LN(*a*)

Returns the natural logarithm (base *e*) of ***a***.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL.

Remarks

If ***a*** is less than zero, the task will fault. An error will be logged in the Event Log and the user program running on the task will stop.

Examples

LN(1.0) returns 0.0

LN(2.718282) returns 1.0.

See Also

Standard Functions

5.14.2.26. LOG_EVENT Function

LOG_EVENT(*a*, ...)

Logs the values of the operands ***a***, ... in the Event Log. This function can have 1 to 3 operands. This function is very useful for troubleshooting.

Parameters

a, etc. **(REAL, DINT, or DWORD)**
The values to record in the Event Log.

Return Value

The LOG_EVENT function does not return a value.

Remarks

The LOG_EVENT function can only be used in the Expression (113) command and user functions. Multiple LOG_EVENT statements can be used in a single Expression (113) command. Do not use with the assignment operator (:=). Use this function by itself only. Other functions can be nested inside.

For calculation-intensive projects, logging of assignment expressions, immediate commands, and the COPY function may be disabled in the Programming Properties. The LOG_EVENT function provides a method of troubleshooting even when the aforementioned logging is disabled.

Examples

```
LOG_EVENT(_Axis[0].ActPos);
LOG_EVENT(MyVariable, _Axis[0].ActPos);
LOG_EVENT(MyVariable, ABS(_Axis[0].ActPos), MyArray[0]);
LOG_EVENT(MyVariable + MyOtherVariable, _Axis[0].StatusBits, SIN(123));
```

See Also

Standard Functions

5.14.2.27. MAX Function

MAX (*a, b, ...*)

Returns the largest (most positive) value of the input parameters (up to 10 operands).

Parameters

***a, b, etc.* (REAL or DINT)**
The input values. All input parameters must be of the same type.

Return Value

Returns the same data type as the input parameters.

Examples

MAX(3, 5, 1) returns 5
MAX(-7.0, 5.1) returns 5.1

See Also

MIN Function | Standard Functions

5.14.2.28. MIN Function

MIN (*a, b, ...*)
Returns the smallest (most negative) value of the input parameters (up to 10 operands).

Parameters

***a, b, etc.* (REAL or DINT)**
All input parameters must be of the same type.

Return Value

Returns the same data type as the input parameters.

Examples

MIN(8, 5, 10) returns 5
MIN(10, 2, -5,8) returns -5

See Also

MAX Function | Standard Functions

5.14.2.29. MROUND Function

MROUND(*a, multiple)*
Rounds ***a*** to the desired ***multiple***.

Parameters

***a* (REAL)**
The input value.
***multiple* (REAL)**
a will be rounded to the closest multiple of this value.

Return Value

Returns a REAL.

Remarks

Rounds ***a*** to the desired ***multiple***, away from zero if dividing gives a remainder greater than or equal to half the value of the ***multiple***.

multiple must be non-zero. If ***multiple*** is zero, the task will fault. An error will be logged in the Event Log and the user program running on the task will stop.

Examples

MROUND(87,12) returns 84

MROUND(0.53,0.15) returns 0.6

See Also

ROUND | Standard Functions

5.14.2.30. POLY Function

POLY(*t, a, b, c, d, ...*)

Returns the polynomial calculation of the equation **a** + ***bt*** + ***ct***^{2} + ***dt***^{3}..., with up to 8 coefficients (up to 7th order).

Parameters

t (REAL)

The polynomial variable.

a, etc. **(REAL)**

The polynomial coefficients, where a is the zero-order coefficient, b is the first order coefficient, c is the third-order coefficient, etc.

Return Value

Returns a REAL.

Remarks

This function supports between 2 and 8 coefficients (3 and 9 total parameters). This function is more efficient than entering the polynomial directly.

Examples

POLY(1, 1, 2, 4, 1) is the equivalent of $1 + 2t + 4t^2 + t^3$, where t is 1.0 and returns 8.0.

POLY(0.5, -10, 5, -0.1) is the equivalent of $-10 + 5t - 0.1t^2$, where t is 0.5 and returns -7.525.

See Also

Standard Functions

5.14.2.31. REAL_TO_DINT Function

REAL_TO_DINT(*a*)

Converts ***a*** to a DINT by rounding to the nearest integer.

Parameters

***a* (REAL)**

The value to convert to a DINT.

Return Value

Returns a DINT.

Remarks

0.5 rounds up to 1.

Rounding -0.5 differs between the RMCs:

- The RMC75E and RMC150E will round -0.5 to -1.
- The RMC75S and RMC75P will round -0.5 to 0.

Examples

REAL_TO_DINT(0.5) returns 1

REAL_TO_DINT(-7.89) returns -8

See Also

TRUNC Function | DINT_TO_REAL Function | DINT_TO_DWORD Function | DWORD_TO_DINT Function | Standard Functions

5.14.2.32. REG_REAL, REG_DINT, REG_DWORD Functions

REG_REAL (*file, element*)

REG_DINT (*file, element*)

REG_DWORD (*file, element*)

Represents the register at the specified IEC address %MD***file***.***element***.

Parameters

***File* (DINT)**

The file portion of the address.

***Element* (DINT)**

The element portion of the address.

Returns

Represents the register at the specified IEC address %MD***file***.***element***. The function determines the returned data type:

REG_REAL: REAL

REG_DINT: DINT

REG_DWORD: DWORD

Remarks

This function is not intended for common use. All RMC registers can be addressed directly, making this function unnecessary in most applications. This function is intended for the rare cases

in which it is necessary to calculate an address mathematically or address a register regardless of the data type or tag name.

The data type of the specified IEC address will be the external data type as specified in the RMC register map. This data type need not match the returned data type. The bits of the register will be preserved. That is, the data will not be converted.

Examples

```
MyReal := REG_REAL(num1 + 10,num2 + 8);
REG_REAL(10,8) := 20.0;
```

Accessing the axis' Scale parameter on an RMC75, given the axis number:

```
REG_REAL(12+MyAxis, 0) := 0.003;
```

See Also

IEC Addressing | RMC75 Register Map | RMC150 Register Map | Standard Functions

5.14.2.33. ROL Function

ROL(*a*, *n*)

Rolls the bits in ***a*** to the left ***n*** times.

Parameters

***a* (DINT or DWORD)**

The value to shift.

***n* (DINT)**

The number of times to shift ***a***. ***n*** must be between 0 and 31.

Return Value

Returns the same data type as ***a***.

Remarks

The result of an invalid ***n*** is undefined.

Examples

ROL(16#8000000F, 2) returns 16#000003C2

See Also

ROR Function | SHR Function | SHL Function | ASHR Function | Standard Functions

5.14.2.34. ROR Function

ROR(*a*, *n*)

Rolls the bits in ***a*** to the right ***n*** times.

Parameters

***a* (DINT or DWORD)**

The value to shift.

***n* (DINT)**

The number of times to shift ***a***. ***n*** must be between 0 and 31.

Return Value

Returns the same data type as ***a***.

Remarks

The result of an invalid ***n*** is undefined.

Examples

ROR(16#0000000F, 2) returns 16#C0000003

See Also

ROL Function | SHR Function | SHL Function | ASHR Function | Standard Functions

5.14.2.35. ROUND Function

ROUND(*a*)

ROUND(*a*, *n*)

Rounds ***a*** to ***n*** decimal places.

Parameters

***a* (REAL)**

The input value.

***n* (DINT)**

The number of decimal places to round ***a*** to. If ***n*** is omitted, it is assumed to be 0, and ***a*** will be rounded to the nearest integer. ***n*** can be between -6 and 6. Negative values specify decimal places to the left of the decimal. Positive values specify decimal places to the right of the decimal.

Return Value

Returns a REAL.

Remarks

0.5 rounds up to 1.0.

Rounding -0.5 differs between the RMCs:

- The RMC75E and RMC150E will round -0.5 to -1.0.
- The RMC75S and RMC75P will round -0.5 to 0.0.

If ***n*** is outside of the range -6 to 6, the task will fault. An error will be logged in the Event Log and the user program running on the task will stop.

Examples

ROUND(0.5) returns 1.0

ROUND(123456,-3) returns 123000

ROUND(0.9876,2) returns 0.99

See Also

MROUND Function | Standard Functions

5.14.2.36. SEL Function

SEL(*cond, a, b*)

Evaluates the boolean condition ***cond***. If ***cond*** is true, the function returns ***a***. If ***cond*** is false, the function returns ***b***. Also known as an in-line IF statement.

Parameters

***cond* (BOOL)**

The condition to be evaluated.

***a* (REAL or DINT)**

The value to return if ***cond*** is true.

***b* (REAL or DINT)**

The value to return if ***cond*** is false.

The ***a*** and ***b*** input parameters must be of the same data type.

Return Value

Returns the same data type as the ***a*** and ***b*** input parameters.

Examples

SEL(_Axis[0].ActPos > 10, 100, SQRT(2)) returns 100 if the Axis 0 Actual Position is greater than 10, otherwise it returns the square root of 2.

See Also

Standard Functions

5.14.2.37. SHL Function

SHL(*a, n*)

Shifts the bits in ***a*** to the left ***n*** times. Zeros are shifted in.

Parameters

***a* (DINT or DWORD)**

The value to shift.

***n* (DINT)**

The number of times to shift ***a***. ***n*** must be between 0 and 31.

Return Value

Returns the same data type as ***a***.

Remarks

The result of an invalid ***n*** is undefined.

Examples

SHL(16#00000008, 2) returns 16#00000020

See Also

SHR Function | ASHR Function | ROL Function | ROR Function | Standard Functions

5.14.2.38. SHR Function

SHR(*a*, *n*)

Shifts the bits in ***a*** to the right ***n*** times. Zeros are shifted in.

Parameters

***a* (DINT or DWORD)**

The value to shift.

***n* (DINT)**

The number of times to shift ***a***. ***n*** must be between 0 and 31.

Return Value

Returns the same data type as ***a***.

Remarks

The result of an invalid ***n*** is undefined.

Examples

SHR(16#00000008, 2) returns 16#00000002

See Also

SHL Function | ASHR Function | ROL Function | ROR Function | Standard Functions

5.14.2.39. SIGNUM Function

SIGNUM(*a*)

Returns -1 if ***a*** is negative, +1 if ***a*** is positive, and 0 if ***a*** is zero.

Parameters

***a* (REAL or DINT)**

The input value.

Return Value

The same data type as ***a***.

Examples

SIGNUM(0.001) returns 1.0
SIGNUM(0.0) returns 0.0
SIGNUM(-100) returns -1

See Also

Standard Functions

5.14.2.40. SIN Function

SIN(*a*)

Returns the sine of ***a***.

Parameters

***a* (REAL)**

The input value in radians.

Return Value

Returns a REAL.

Remarks

Notice ***a*** is in radians. To convert a value from degrees to radians, multiply the degrees value by π/180. The RMC has an M_PI constant for π.

Examples

SIN(1.5707964) returns 1.0
SIN(45 * M_PI / 180) returns 0.70710677

See Also

COS Function | TAN Function | ACOS Function | Standard Functions

5.14.2.41. SINH Function

SINH(*a*)

Returns the hyperbolic sine of ***a***.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL.

See Also

COSH Function | TANH Function | Standard Functions

5.14.2.42. SQRT Function

SQRT(*a*)

Returns the square root of ***a***.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL.

Remarks

If ***a*** is negative, the task will fault. An error will be logged in the Event Log and the user program running on the task will stop.

Examples

SQRT(100.0) returns 10.0.

SQRT(2.0) returns 1.4142135.

See Also

Standard Functions

5.14.2.43. TAN Function

TAN(*a*)

Returns the tangent of ***a***.

Parameters

***a* (REAL)**

The input value in radians.

Return Value

Returns a REAL.

Remarks

Notice ***a*** is in radians. To convert a value from degrees to radians, multiply the degrees value by π/180. The RMC has an M_PI constant for π.

Examples

TAN(M_PI / 4) returns 1.0

TAN(45 * M_PI / 180) returns 1.0

See Also

SIN Function | COS Function | ATAN Function | Standard Functions

5.14.2.44. TANH Function

TANH(*a*)

Returns the hyperbolic tangent of ***a***.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL.

See Also

SINH Function | COSH Function | Standard Functions

5.14.2.45. TRUNC Function

TRUNC (*a*)

Rounds ***a*** to an integer towards zero and returns a DINT data type.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a DINT.

Examples

TRUNC(34.78) returns 34

TRUNC(-3.46) returns -3

TRUNC(-3.99) returns -3

See Also

TRUNC_REAL Function | REAL_TO_DINT Function | ROUND Function | Standard Functions

5.14.2.46. TRUNC_REAL Function

TRUNC_REAL (*a*)

Rounds ***a*** to an integer towards zero.

Parameters

***a* (REAL)**

The input value.

Return Value

Returns a REAL.

Examples

TRUNC_REAL(34.78) returns 34.0
TRUNC_REAL(-3.46) returns -3.0
TRUNC_REAL(-3.99) returns -3.0

See Also

TRUNC Function | Standard Functions

5.14.3. User Functions

5.14.3.1. User Functions

To access the User Function Editor: In the Project pane, expand **Programming**, and double-click **User Functions**.

User functions are custom functions created or imported by the user. User Functions provide flexibility and efficiency for advanced applications. Most applications do not need user functions. User functions can be used anywhere expressions are used in the RMC, including the Expression (113) command, link conditions in user programs, and the Program Triggers.

User functions can have any number of parameters, and return a single value as the result. The parameters can be input, output, or input/output types. Therefore, a function can return many values via the output or input/output type parameters.

Example User Functions

See the Example User Functions for examples that you can copy and paste into your project.

Creating a User Function

1. In the Project Pane, expand **Programming** and double-click **User Functions**.
2. In the User Function Editor toolbar, click the **New User Function**  button.
3. In the New User Function dialog, type a **Name** for the function.
4. Choose the desired **Return Data Type**. This is the data type of the resulting value returned by the function.
5. **Add Parameters (optional)**
   Do the following for each parameter you wish to add. Parameters appear as Input, Output, and Input/Output variables in the user function. Parameters can also be added and modified later when editing the user function, as described in Declaring Variables in User Functions.
   1. Click the **Add** button.

2. Type a **Name** for the function.
3. Choose the desired **Data Type** of the parameter. To create an array parameter, set the **Size** parameter to a value greater than 1.
4. In the **Input/Output** box, choose the parameter type:
   - Input
   - Output
   - Input/Output
5. Click **OK**.

To adjust the order of the parameters, use the **Up** and **Down**  arrows. To delete parameters, use the **Delete**  button.

6. Click **OK**.
7. The new function will be added to the User Functions list, and will appear in the editor.

Editing a User Function

User Function Description

The initial comment in the user function is the function description, enclosed by green parentheses with an asterisk: (* *). This description will appear when browsing the functions in the Expression Editor.

Enter a description that describes what the function does. It is good practice to clearly describe the parameters and return value. You do not need to describe the parameter data types, as this is automatically displayed in the Expression Editor.

Function Name

The function name and return data type is defined by the keyword **FUNCTION**. To change the function name or data type, change the text following the FUNCTION keyword. The entire function must also end with the **END_FUNCTION** keyword.

For example, **mm2inch** is this function's name, which returns a value of type REAL:

```
FUNCTION mm2inch : REAL
```

Variable Declarations

Functions can have input variables, output variables, input/output variables, and local variables. For details, see Declaring Variables in User Functions.

Editing the Function Body

The function body contains the code that the function executes when it is called. The code in the function body must follow the same syntax as the code used in the Expression (113) command. For details, see the assignment expressions topic. The function body can use all the same operators, functions, user functions, and keywords as used in assignment expressions.

Values can be passed into and out of a function via parameters. All variables of type Input, Output, or Input/Output that are declared in the function will be the parameters of the function. See Declaring Variables in User Functions for more details.

The following limitations apply to the user function body:

- Output variables can only be assigned to. That is, Output variables can only be used on the left-hand side of assignment expressions.
- Input variables cannot be assigned to.

Accessing Controller Tags

The code in a user function can directly reference controller tags, such as variables in the Variable Table, axis tags, etc. However, in order to keep a user function portable between RMCTools projects, it is good practice to limit direct referencing of controller tags and instead pass values in and out via the Input, Output, and Input/Output function parameters.

Function Return Value

Each function must have a return value. This is done by assigning a value to the name of the function. For example, consider a function named **Average3.** The return value could be expressed as:

```
Average3 := (Var1 +  Var2 + Var3 ) / 3;
```

Managing User Functions

All user functions in the project are displayed in the User Function list. To add a user function, click the **New User Function** button. To delete a user function, click the **Delete User Function** button.

User Functions Folders

User functions can be organized into folders in the user function list.

1. To create a new folder, click the **New Folder** button on the toolbar. Type the folder name and press Enter.
2. To add new functions to the folder, select the folder, then click the **New User Function** button. You can also drag existing functions into the folder with the mouse.

Click the **Delete Folder** button to delete a folder and it contents.

Importing User Functions

1. Right-click in the function list pane and choose **Import User Functions**.
2. Browse for the RMCTools User Function Library (.rmcflib) file and click **Open**.
3. The user functions available in the RMCTools User Function Library file will be listed. Check the desired functions in the file to import. The **Select All** and **Clear All** buttons may assist you. Imported functions cannot have the same name as a user function already in the project. These functions are labeled **(duplicate)**. Use the **Rename** button to rename a function in the import list so that it can be imported.
4. After selecting the desired user functions to import, click **OK**.

Exporting User Functions

1. Right-click in the function list pane and choose **Export User Functions**.
2. Choose the functions to export, then click **OK**.
3. Browse to the desired location, enter a filename, then click **Save**.

Calling User Functions

To insert a user function into an expression:

1. Begin typing the name of the function and choose the function from the pop-up list.
   Or, browse for the function on the **Function** tab of the Expression Editor and double-click it.
2. Type an opening parenthesis. A pop-up window will open, listing the required parameters for your convenience.
3. Make sure to enter all required parameters, separated with a comma.
4. Type a closing parenthesis.

Using the Function's Return Value

Typically, when calling a user function, the function's return value is assigned to some register. For example, the return value of the user function inch2mm is assigned to the variable MyPos here:

```
MyPos := inch2mm(_Axis[0].ActPos);
```

A user function can also be called without assigning the function's result to a register. This is useful when the function has output parameters. For example, consider the following user function that converts polar coordinates (radius and angle) to cartesian coordinates (x and y). The function has two input parameters and two output parameters:

BOOL **Polar2Cart**([in] REAL **radius**, [in] REAL **angle**, [out] REAL **x**, [out] REAL **y**)

This user function can be used in an expression as follows:

```
Polar2Cart(MyRadius, MyAngle, MyX, MyY);
```

Limitations

The following limitations apply to calling user functions:

- Any Output and Input/Output type parameters can only be step-local variables or variables in the Variable Table. They cannot be system tags. If the result of the function must go to a system tag, you can use a variable for the Output parameter, then use a separate expression to assign the variable value to the system tag.

See Also

Functions Overview | Standard Functions | Example User Functions | Declaring Variables in User Functions | Expressions Overview | Assignment Expressions

5.14.3.2. Declaring Variables and Parameters in User Functions

Variables in user functions are used to assist in calculations, and/or serve as the function parameters.

Variables as Parameters

Variables in a user function that are declared with the **VAR_INPUT**, **VAR_OUTPUT**, and **VAR_IN_OUT** keywords make up the function's parameters. The order in which the parameters are declared in the user function is the order in which the parameters must be entered when calling that user function.

Input Parameters (VAR_INPUT)

Input parameters can only be read from. Values cannot be assigned to input parameters.

Output Parameters (VAR_OUTPUT)

Output parameters can only be assigned to. That is, output parameters can only be used on the left-hand side of assignment expressions.

Input/Output Parameters (VAR_IN_OUT)

Input/output parameters can be read and written.

Variables as Local Variables

Variables declared with the **VAR** keyword are variables that can be used locally in the user function to assist in calculations.

Declaring Variables and Parameters

Use the format in the example below to declare variables in user functions. Keep in mind that the order in which the parameters (VAR_INPUT, VAR_OUTPUT, and VAR_IN_OUT) are declared in the user function is the order in which the parameters must be entered when calling that user function.

Variables can be of any data type, and can be fixed-length arrays.

Local variables (VAR), output parameters (VAR_OUTPUT), and input/output parameters (VAR_IN_OUT) can optionally be initialized upon declaration.

Examples

```
VAR_INPUT
    MyInputVar : REAL;
    MyOtherInputVar : DINT;
    YourVar : Array [0..4] OF REAL;
```

```
END_VAR
VAR_IN_OUT
        MyInOutVar : REAL;
END_VAR
VAR_OUTPUT
        MyOutputVar : REAL :=4;
        MyArray : Array [0..3] OF REAL := [10, 10, 0];
END_VAR
VAR
        MyVar : REAL := 100;
        YourVar : Array [0..9] OF DINT;
END_VAR
```

Declaring Arrays

Array bounds are specified in brackets [], with two intermediate periods (..). The array bounds can be any integer value, including negative values. For most applications, the lower bound is zero. The maximum length of an array in a user function is 32 elements.

Parentheses can be used a repetition factor when assigning an initial value. The value preceding the parentheses specifies the number of repetitions. For example, [4(0)] is equivalent to [0,0,0,0], and [1,3(99),1] is equivalent to [1,99,99,99,1].

Example

This declaration creates an output array of length 5, with initial values of [-1,1,0,0,0].

```
VAR_OUTPUT
        MyArray : Array [0..4] OF REAL := [-1,1, 3(0)];
END_VAR
```

See Also

User Functions | Example User Functions

5.14.3.3. Example User Functions

The text from the following example user functions can be copied and pasted into the User Function Editor.

Example 1

Converts inches to millimeters.

Usage
`MyPos := inch2mm(_Axis[0].ActPos);`
Declaration

`(*`
`Converts inches to millimeters.`
`*)`
`FUNCTION inch2mm : REAL`

```
    VAR_INPUT
        Inches : REAL;
    END_VAR

    inch2mm := Inches * 25.4;
END_FUNCTION
```

Example 2

Returns the average of 4 values.

Usage

```
MyPos := Avg4(_Axis[0].ActPos, _Axis[1].ActPos, _Axis[2].ActPos,
_Axis[3].ActPos);
```

Declaration

```
(*
Returns the average of the 4 input parameters.
*)
FUNCTION Avg4 : REAL

    VAR_INPUT

        Val1 : REAL;

        Val2 : REAL;

        Val3 : REAL;

        Val4 : REAL;

    END_VAR

    Avg4 := (Val1 + Val2 + Val3 + Val4) / 4.0;


END_FUNCTION
```

Example 3

Converts polar coordinates to cartesian coordinates.

Usage

```
Polar2Cart(radius, angle,x,y);
```

Declaration

```
(*
Converts polar coordinates r, theta to cartesian coordinates x, y.
Theta is in the interval [0,360) degrees.
*)
FUNCTION Polar2Cart : BOOL
    VAR_INPUT
```

```
        r : REAL;           //radius
        theta : REAL;       //angle in degrees [0,360)
    END_VAR
    VAR_OUTPUT
        x : REAL;
        y : REAL;
    END_VAR
    IF theta = 0.0 OR theta = 180.0 THEN
        x := r;
        y := 0.0;
    ELSIF theta = 90.0 OR theta = 270.0 THEN
        x := 0.0;
        y := r;
    ELSE
        x := r * COS(theta * (M_PI / 180.0));
        y := r * SIN(theta * (M_PI / 180.0));
    END_IF
    Polar2Cart := TRUE;
END_FUNCTION
```

Example 4

Returns the index of the least significant bit that is set in a DWORD.

Usage

```
MyIndex := LSB(MyDWORD);
```

Declaration

```
(*
LEAST SIGNIFICANT BIT
returns the index of the least significant bit set in a DWORD.
*)
FUNCTION LSB : DINT
    VAR_INPUT
        DW_IN : DWORD;
    END_VAR
    VAR
        DW : DWORD;
        C : DINT;
    END_VAR
    IF DW_IN <> 0 THEN
        DW := DW_IN AND DINT_TO_DWORD(DWORD_TO_DINT(NOT DW_IN) +
1);
        C := 0;
        IF (DW AND 16#FFFF0000) <> 0 THEN C := C + 16; END_IF
        IF (DW AND 16#FF00FF00) <> 0 THEN C := C + 8;  END_IF
        IF (DW AND 16#F0F0F0F0) <> 0 THEN C := C + 4;  END_IF
        IF (DW AND 16#CCCCCCCC) <> 0 THEN C := C + 2;  END_IF
        IF (DW AND 16#AAAAAAAA) <> 0 THEN C := C + 1;  END_IF
        LSB := C;
    ELSE
        LSB := -1;
    END_IF
```

```
END_FUNCTION
```

See Also

User Functions | Expressions

5.15. Discrete I/O

5.15.1. Discrete I/O Overview

Discrete I/O are physical boolean inputs or outputs. General discrete I/O are available on the following modules:

RMC	Module	Details
RMC75	D8 Expansion module	8 discrete I/O, individually programmable as inputs or outputs.
RMC150	DI/O Module	8 outputs, 18 inputs. Fits in slots 0, 2, 3, 4, and 5.
	CPU Module	2 outputs, 2 inputs
RMC200	D24 Module	24 discrete I/O: 0-19 individually programmable as inputs or outputs, 20-23 are inputs.
	CPU20L Module	2 outputs, 2 inputs
	CPU40 Module	2 outputs, 2 inputs
	CA4 Module	4 Fault inputs, 4 Enable outputs
	CV8 Module	8 discrete I/O, individually programmable as inputs or outputs.
	Q4 Module	4 Home inputs, 4 Registration inputs
	U14 Module	4 discrete I/O, individually programmable as inputs or outputs, 2 Reg/Z inputs

Some of the discrete I/O have dedicated functionality but can easily be used as general-purpose I/O, such as those on the CA4 and Q4 modules, and the Reg/Z inputs on the U14 modules. Other modules have dedicated discrete I/O that is more difficult to use a general-purpose I/O, such as:

RMC	Module	Details
RMC75	MA1, AA1, QA1	1 Fault Input, 1 Enable Output
	MA2, AA2, QA2	2 Fault Inputs, 2 Enable Outputs

Using General Discrete I/O

See the Using Discrete I/O topic for details on implementing the uses of discrete I/O listed below. Tag names can be assigned to general discrete I/O.

Discrete inputs can be used for the following:

- To start a User Program.
- As part of the condition for the Conditional Jump link type in User Programs.
- In the Expression (113) command in a User Program.

- Inputs can be forced using the Discrete I/O Monitor in RMCTools.
- If configured to do so, one input can control whether the controller is in Run Mode or Program Mode.
- As Physical Limit Inputs.

Discrete outputs can be used for the following:

- The outputs can be set with the Expression (113) command in a User Program.
- The outputs can be set by writing to the address of the output in the RMC via a PLC, HMI, etc.
- Outputs can be toggled using the Discrete I/O Monitor in RMCTools.
- Outputs can be forced using the Discrete I/O Monitor in RMCTools.

Configuring the Discrete I/O

To configure the discrete I/O, use the Configuring Discrete IO topic.

Viewing the Discrete I/O

There are two ways to view the state of the discrete I/O:

- Use the Discrete I/O Monitor in RMCTools.
- Look at the LEDS on the module, if the module has LEDs.

Addressing Discrete I/O

See the Register Map for details on the addresses of the discrete I/O registers.

Discrete I/O points are represented as bits in the RMC. To address a bit, you must specify the bit in the register that contains it. Not all host controllers allow this, depending on the protocol. For example, addressing a bit in an F file using an Allen-Bradley protocol is not supported in many HMIs. Typically, specifying bits in any word is possible with most controllers using Modbus/RTU or Modbus/TCP. Notice that you may need to address the upper 16-bit portion of the 32-bit register in order to pick out the bits higher than 15.

In RMCTools, the discrete I/O addresses are shown in the IEC 61131-3 format:

RMC	I/O Addressing	Examples
RMC75	**Inputs =** %IX*n* **Outputs** = %QX*n* where *n* is the I/O number as displayed in the Discrete I/O Monitor.	%QX0 is output 0 %IX8 is input 8
RMC150	**Inputs =** %IX*slot.n* **Outputs** = %QX*slot.n* where *slot* numbering starts with 0 for the left-most module in the RMC150. *n* = the number of the input or output on that module.	%QX0.5 is output 5 in slot 0 %IX5.0 is input 0 in slot 5
RMC200	**Inputs =** %IX*slot.n* **Outputs** = %QX*slot.n* where	%QX1.0 is output 0 in slot 1 %IX5.7 is input 7 in slot 5

	slot numbering starts with 0 for the left-most module in the RMC200. *n* = the number of the input or output on that module.	

See Also

Using Discrete I/O | Configuring Discrete I/O | RMC150 Discrete I/O Wiring | EXP70-D8 Wiring

5.15.2. Using Discrete I/O

This topic describes many different ways to use the RMC discrete I/O.

Tag Names

General-purpose discrete I/O can be given tag names. To assign tag names to discrete I/O, use the Discrete I/O Configuration editor.

Setting Outputs

There are several ways to turn discrete outputs on or off:

Using Commands

The commands below operate on a single output. To operate on multiple outs simultaneously, use one of the methods above.

Set Discrete Output (60)

Clear Discrete Output(61)

Toggle Discrete Output(62)

From RMCTools

Use the Discrete I/O Monitor to toggle outputs on or off. See **Toggling Outputs** below.

Use the Discrete I/O Monitor to force the output on or off. See **Forcing Inputs or Outputs** below.

Note:
An output or input that is forced will always remain in the forced state until the force is removed. Toggling an output will change the state of the output if is not forced.

From a User Program

Use the Expression (113) command in a User Program to turn on outputs or get the state of inputs.

A sample expressions is:

```
MyOutput := True
```

The MyOutput tag must previously have been assigned to an output for this expressions to be valid.

From PLC or HMI

Write to the address of the output. See the RMC150 DI/O Register Map, RMC75 DI/O Register Map, or RMC200 DI/O Register Map for address details.

Using Inputs and Outputs in User Programs

You can use inputs and outputs in User Programs for calculations and controlling the flow of the program. Inputs and outputs are boolean Data Types and must match the data types in the expression or must be converted to match the other data types.

All variables, inputs and output tags in the sample expressions below must have been defined for the expression to be valid.

In the Expression (113) command:

Turning on an output:

To turn an output on or off, assign a value to the output.

Sample Expression:

```
MotorOnOutput := ( NOT LowPressureInput ) AND ( NOT ErrorInput )
```

Assigning a value to a variable:

You can assign a value to a variable based on the value of an input.

Sample Expression:

```
SystemOK.0 := LowPressureInput OR (_Axis[0].ActPos > 24.2)
```

> **Note:**
> SystemOK.0 is the first bit in the DWORD type variable SystemOK.

In the Conditional Jump link type:

The Conditional Jump type jumps to a step based on whether the expression evaluates to true or false. To use Discrete I/O, you must enter an expression. To do this, choose the **Cnd Jmp** link type, double-click the **Link Condition** box, choose **Other**, and click **Next**.

Sample expressions:

```
SawLimitInput
PressOnInput OR (_Axis[0].ActPos < 24.2)
```

Forcing Inputs and Outputs and Simulate Inputs

Inputs and Outputs can be forced on or off from RMCTools. The inputs and outputs will always remain in the forced state until the force is removed. To force an input or output, you must first Go Online with the controller, and open the Discrete I/O Monitor.

Forcing an Output

In the Discrete I/O Monitor, right-click the output you want to force, and click **Force On** or **Force Off**. The output will be highlighted with a yellow background to indicate is forced.

Forcing an output will force the physical output on or off. Until the force is removed, the RMC will ignore all other attempts to turn that output on or off. If an output is forced on, its corresponding LED on the expansion module will turn green, and the output will physically be on (conducting). If an output is forced off, its corresponding LED on the expansion module will turn off, and the output will physically be off (not conducting). Forcing an output only affects the physical output, and does not affect the corresponding software bit.

Forcing an Input (Simulate an Input)

In the Discrete I/O Monitor, right-click the input you want to force, and click **Force On** or **Force Off**. The input will be highlighted with a yellow background to indicate is forced.

Forcing an input will force the input on or off. Until the force is removed, the RMC will ignore the physical state of the input. The LED on the expansion module will not be affected by the force. The LED will still reflect the physical state of the input.

Removing a Force

In the Discrete I/O Monitor, right-click the input or output you want to remove the force from, and click **Remove Force**. Or, in the Discrete I/O Monitor, right-click anywhere and click **Remove All Forces**.

Using Inputs to Start a User Program

To start a User Program based on an input:

- Open the Program Triggers Editor.

- Double-click the **Condition** cell in the bottom row to create a Program Trigger.
- In the New Condition Wizard, choose **Other** and click **Next**.
- In the **Tags** box, find the input you want under the **Discrete I/O** node and double-click it.
- If you want a User Program to start when the input goes low, type NOT in front of the input name in the **Expression** box. Otherwise, the User Program will start when the input goes high.
- Click **Finish**.
- In the Tasks columns, choose the User Programs you want to start for each task, or choose which tasks you want to stop.
- To apply the changes to the RMC, right-click the Programming node in the Project pane, and click **Download Programs**.

Reading the Input or Output State

To read the state of an RMC input or output, read the bit in the register or read the entire register. See the Register Map for details.

Control Whether the RMC is in RUN, PROGRAM, or Disabled mode.

See the Configuring Discrete I/O for details on assigning an input to control whether the RMC is in RUN, PROGRAM, or Disabled mode.

Hardware Specifications

For hardware details on the discrete I/O, such as response times and voltage and current limits, see the specifications for the respective modules:

RMC150E CPU Module

RMC150 DI/O Module

RMC75 D8 Module

RMC200 DC24 Module

See Also

Discrete I/O Overview | Configuring Discrete I/O

5.15.3. Configuring Discrete I/O

This topic describes how to configure the Discrete IO on the RMC. Each discrete I/O on the RMC can be configured as described below:

Basic Configuration

To configure Discrete I/O, open the Discrete I/O Configuration window first. Then, you can do any of the actions listed below. To apply the changes to the RMC, right-click the **Programming** node in the Project pane, and click **Download Programs to Controller**.

Action	Instructions
Basic Configuration Options	
Configure I/O as Input or Output	Available only on discrete I/O modules with configurable I/O. In the **Type** column, choose Input or Output.
Assign a Tag Name to the I/O	To assign a tag name to an I/O, type a name in the **Tag Name** column. Tag names are limited to 64 characters.

	Tip: Assigning a Tag name to each discrete I/O makes it much easier to work with.
Add a Description	In the **Description** column, add a description of the I/O point for your own reference.
Advanced Configuration Options	
Set the PROGRAM State for Outputs	To assign the state of an output when the RMC enters or starts up in PROGRAM mode, in the **PROGRAM State** column, choose the desired option: • **Hold** - The output retains its previous state. • **Off** - The output turns off. • **On** - The output turns on.
Set the FAULT State for Outputs	The Fault State defines the state of the output in the following conditions: • The RMC receives a Fault Controller (8) command • The RMC200 is in the Disabled state (RMC200 Only). • The RMC200 powers up, regardless of the startup mode (RMC200 Only). **Options:** • **Hold** - The output retains its previous state. • **Off** - The output turns off. • **On** - The output turns on.
Defining a RUN/PROGRAM Input (RMC75/150) **or** **Defingin a RUN/Disable Input (RMC200)**	To assign a discrete input to control the RMC RUN/PROGRAM/Disable mode: • In the Project Pane, expand the **RMC** node. • Right-click the **Programming** node, click **Properties**, and click the **RUN/PROGRAM** or **RUN/Disable** page. • Select **Define a RUN/PROGRAM discrete input** or **Define a RUN/Disable discrete input** • Choose the desired input and click **OK**. For the RMC75 and RMC150, when the discrete input transitions from low to high, the RMC will enter RUN mode. When the input transitions from high to low, the RMC will enter PROGRAM mode. For the RMC200, When the discrete input transitions from low to high, the RMC will enter RUN mode. When the input is low, the RMC will be in Disabled mode.

See Also
Discrete I/O Overview

5.16. Programming Examples and Tips

5.16.1. Programming Examples and Tips

You have set up your motion system and tuned the axes. Now you are ready to tackle the programming of the motion application, but you aren't sure where to start or how to do it. This topic provides links to complete programming examples and tips to help you get going.

Note:
Before programming your application, it is important that you have set up your system, tuned it, and that you can move it back and forth. For help on doing this, see the Startup Guide that came with the RMC. The Startup Guide is also available on the Download page of Delta's web site, https://deltamotion.com/dloads/.

User Program Examples

Creating a Basic User Program - Shows how to create a basic user program and run it.

Simple User program - A simple user program

Jog Button - Shows how to set up a jog with the RMC. Can be done using discrete I/O or via communications.

Closed Loop Motion on Startup - Shows how to enter closed control (or perform more complex actions) when the RMC starts up.

Timers - Shows how to create timers.

Time-out - Shows how to create a time-out after issuing a move command. If the axis does not reach position before the time-out period, the program takes some action.

Arrays - Shows how to use variable arrays.

Application Tips

Emergency Stops - Discusses the programming options available for handling machine fault situations, such as estops.

Other Examples

Expression (113) Command Examples

User Function Examples

5.16.2. Example: A Basic User Program

This topic describes how to create a user program for a simple press application. The concepts illustrated here can be applied to any application.

Keep in mind that when the user program reaches a step, all the commands will be executed immediately. The Link Type tells the user program when to go to the next step. For example, if a step issues a move command, it is possible to go to the next step immediately (the next loop time), it does not have to wait until the axis reaches the position.

Description

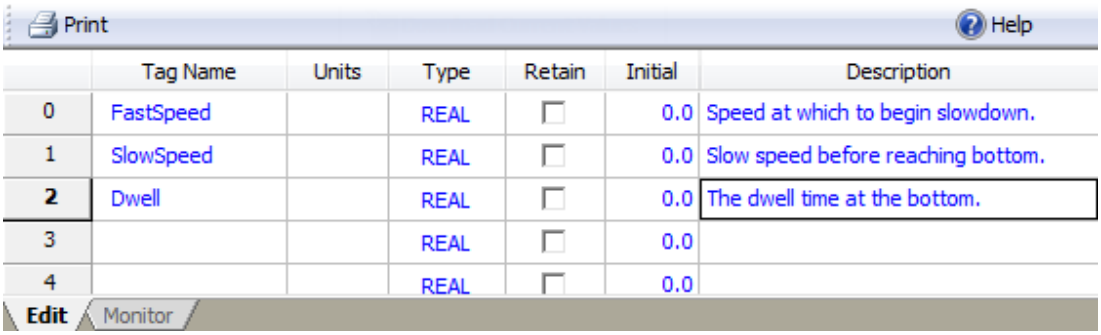
Consider a simple press application where the press has one axis of motion. It starts at the top position of 10 in. It moves down toward zero in. at a certain speed. Once it reaches 9 in., it slows down until it reaches 0 in. It dwells at zero for a certain time, then moves back to 10.

The dwell time and both downstroke speeds may vary. They are selected from an HMI that will communicate with the RMC.

Method

1. Declare Variables

There are three items that can vary. Therefore, declare 3 variables, as shown below. Notice that an HMI or PLC can write to the values of these variables.

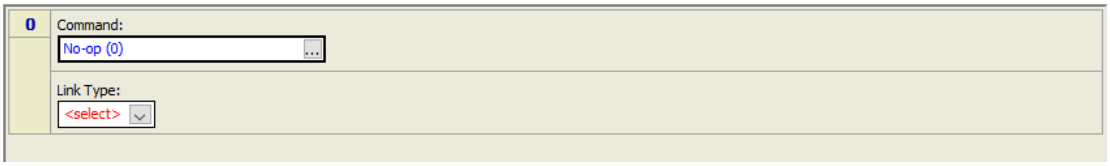


2. Create a New User Program

a. In the **Project Pane**, expand **Programming**, right-click **User Programs**, and click **New Program**.

b. In the **New User Program** dialog, enter a name and click **Finish**.

The user program should now have one step and look like this:move



3. Add Command to First Step

The first step issues a command to move to 0 in. Before it reaches 0 in., it will get a different command that slows it down. This is how to add the command to 0 inches:

a. In the **Command** box, click the ellipsis button . Browse to **Motion Commands** >> **Point-to-Point**, choose **Move Absolute**, and click **OK**.
You can also enter the Move Absolute command into the **Command** box by starting to type "Move Absolute", then clicking it from the pop-up list.

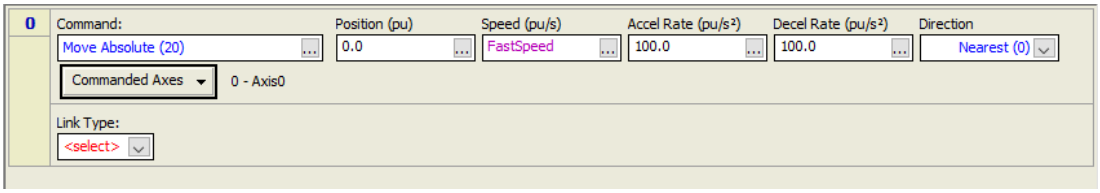
b. In the **Position** box, enter 0.

c. In the **Speed** box, click the ellipsis button . Browse to **Variable Table** >> **Current Values**, choose **FastSpeed**, and click **OK**.
You can also enter the variable by starting to type "FastSpeed", then clicking it from the pop-up list.

d. In the **Accel** and **Decel** boxes, enter 100.

e. Click the **Commanded Axes** button, choose **Axis0**, and click **OK**.

The user program should now look like this:



4. Choose Link Type in First Step

The Link Type in the first step can be set to tell the user program to go to the next step once the axis reaches 9 inches (the next step will issue a Move command with the SlowSpeed).

a. In the **Link Type** box, choose **Wait For**.

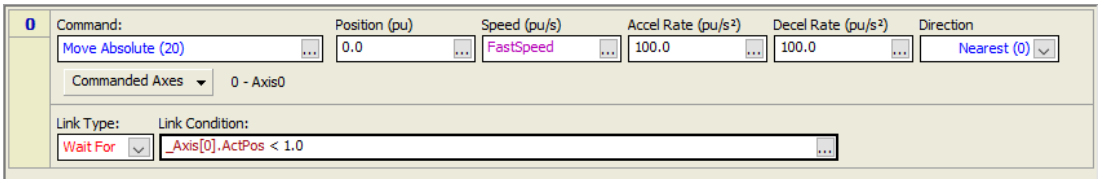
b. In the **Link Condition** box, click the ellipsis button . Choose **Soft Limit Switch** and click **Next**.

c. Choose **One or more specific axes**, check **Axis0**, and click **Next**.

d. In the boxes, choose **Actual Position**, choose **<** , and type "1", as illustrated:

e. Click **Next**, verify that the condition says **Is the Actual Position LESS THAN 1.0 for axis "Axis0"?**, then click **Finish**.

The user program should now look like this:



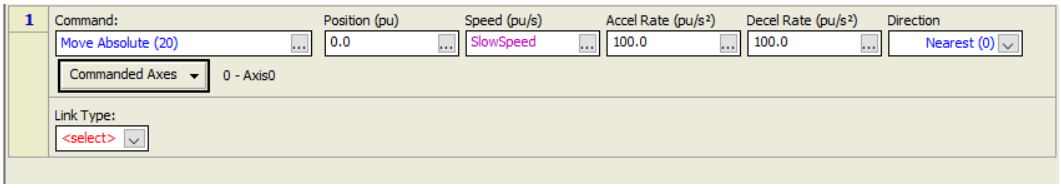
Notice that the Link Type box is red. This indicates an error. In this case, the error is that there is no next step, but one will be created shortly.

5. Add Next Step for Slower Move

The next step will again issue a Move command to 0 inches, this time with a different speed. The step will wait until the axis gets into position before going to the next step.

a. On the Step Editor toolbar, click the **Insert Step** button .

b. Add a Move Absolute command so that step 1 looks like this:



c. In the step 1 **Link Type** box, choose **Wait For**.

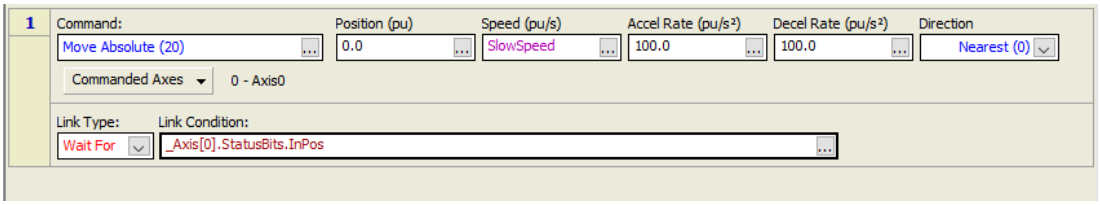
d. In the **Link Condition** box, click the ellipsis button . Choose **Status Bit(s)** and click **Next**.

e. Choose **One or more specific axes**, check **Axis0**, and click **Next**.

f. Check the **On** box for **0: In Position** and click **Next**.

g. Verify that the condition says **Is the Status bit "In Position" ON for axis "Axis0"?**, then click **Finish**.

Step 1 should now look like this:



6. Add Next Step for Delay

This step will not issue a command. It will only delay a certain time.

a. Right-click in the left part of step 1, then click **Add Step After**.

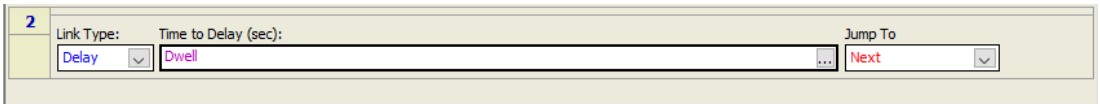
b. This step will not need a command. Therefore, select the No-Op command and click the **Remove Command** button on the toolbar.

c. In the step 2 **Link Type** box, choose **Delay**.

d. In the **Time to Delay** box, click the ellipsis button . Browse to **Variable Table** >> **Current Values**, choose **Dwell**, and click **OK**.

You can also enter the variable by starting to type "Dwell", then clicking it from the pop-up list.

Step 2 should now look like this:



7. Add Last Step for Move Back

This step will move the axis back up to 10 in.

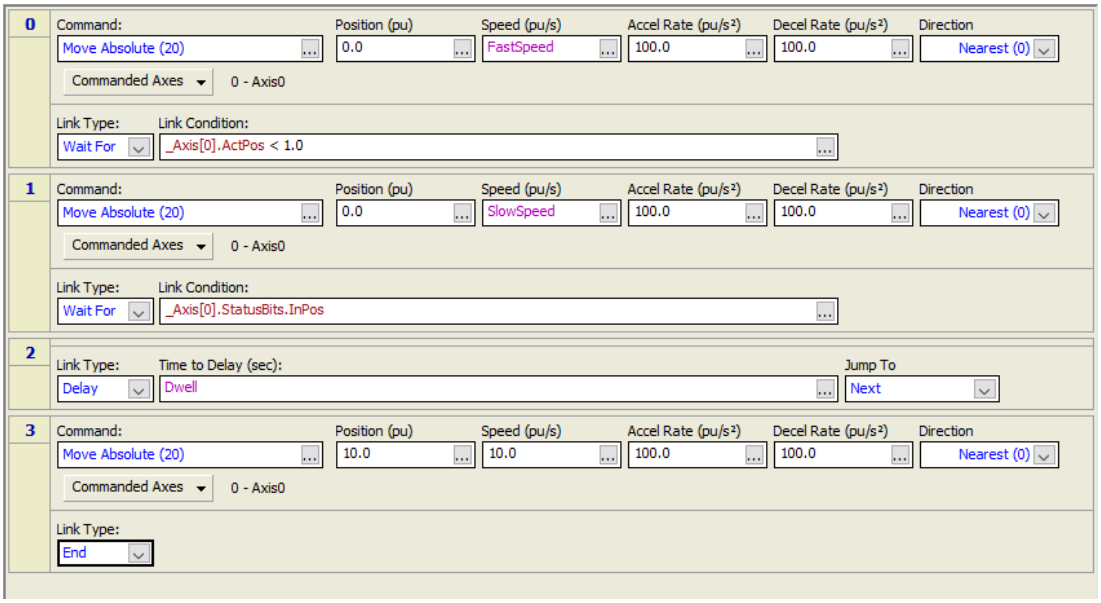
a. Right-click in the left part of step 2, then click **Add Step After**.

b. Add a Move Absolute command with **Position** 10, **Speed** 10, **Accel** 100, and **Decel** 100.

c. Set the Commanded Axes to **Axis 0**.

d. Set the Link Type to **End**.

The user program should now look like this:



8. Download the Programming

To download the programming, in the Project pane, right-click **Programming** and choose **Download Programs to Controller**. If any errors are found, the Verify Results Window will list them and provide links to them. All errors must be fixed before the download can occur.

9. Running the User Program

Before running the user program, use the **Monitor** tab of the Variable Table Editor to set the Current Values of the variables to reasonable values. Make sure to click the **Download Current Values** button to apply the edits to the controller.

To run the user program, choose the **Start Task(90)** command in the command tool. In the **Task** box, choose **0** and in the **Program** box, choose the user program you created. Click **Send**. You can use the Task Monitor to see which step the user program is on.

See Also

Programming Examples Overview

5.16.3. Example: A Simple User Program

This topic shows a simple user program. For a detailed example on how to create a basic user program, see the Example: Creating a Basic User Program topic.

Assume the User Program below is from a system scaled to inches, so that 1 pu (position-unit) is 1 inch. This example User Program does the following:

Step 0:

Command: Moves Axis 0 to 1 in. at 20 in/sec.

Link: Waits for Axis 0 In Position status bit to turn on, which means the axis has reached the requested position. This link type is easily created using the Link Type Wizard.

Step 1:

Command: Sets a Discrete Output.

Link: Waits for 5 seconds, then it goes to the next step.

Step 2:

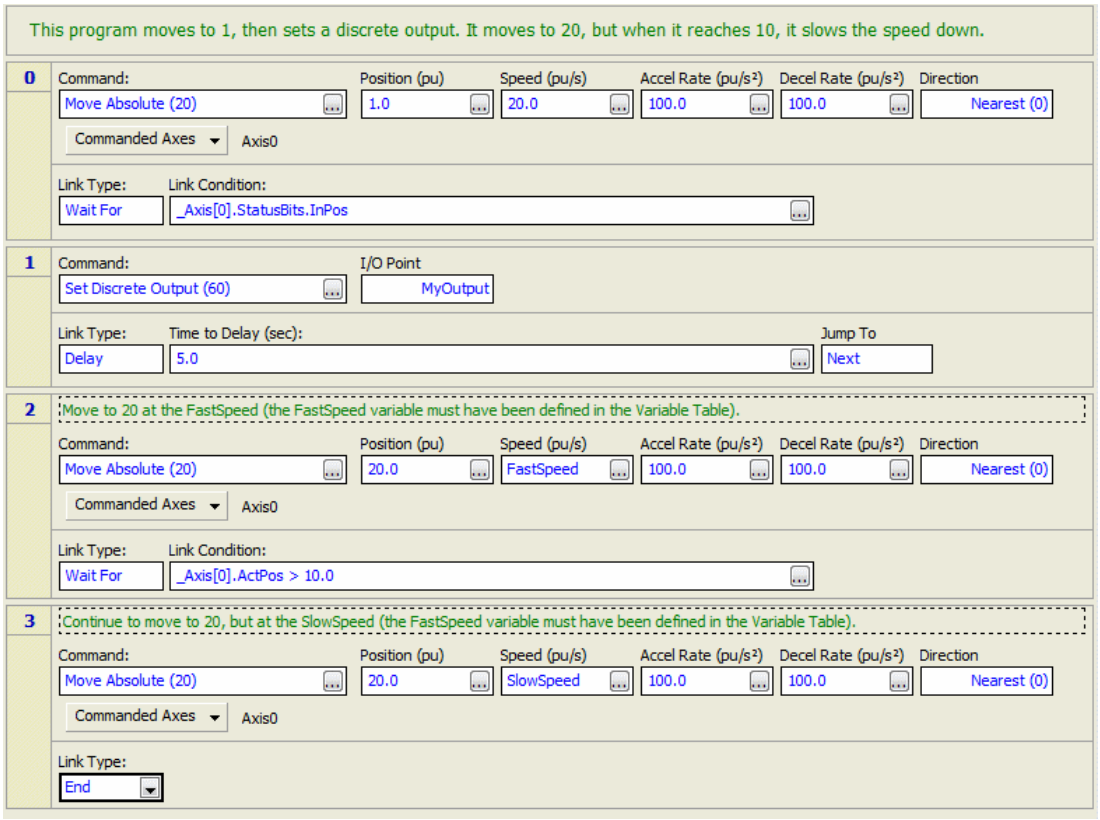
Command: Moves Axis 0 to 20 in. at the speed specified by the FastSpeed variable. This variable must have previously been defined in the Variable Table.

Link: Waits until the Axis 0 Actual Position is greater than 10, then it goes to the next step.

Step 3:

Command: Moves Axis 0 to 20 inch at the speed specified by the SlowSpeed variable. This variable must have previously been defined in the Variable Table. Notice that this command is issued before the axis reaches the Command Position from step 2. This effectively slows down the speed of the move.

Link: Ends the User Program immediately. Notice that the User Program stops running immediately, and the RMC will still finish making the move.



See Also

Programming Examples Overview

5.16.4. Example: Closed Loop Motion on Startup

This topic describes how to program the RMC so that an axis enters closed loop control when the RMC turns on. For example, perhaps you want the RMC to hold the position of the axes as soon as it turns on.

The same method can be used to perform more complicated actions when the RMC starts up.

Description

To automatically hold position in closed loop control when the RMC starts up requires the following steps:

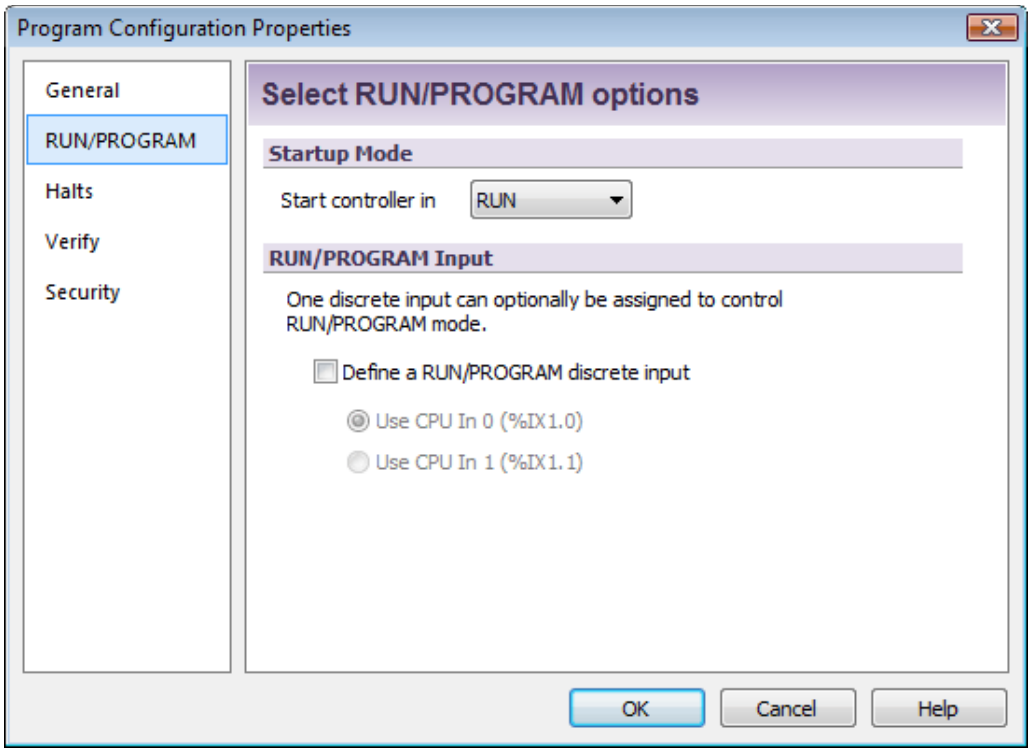
1. Set the RMC to start up in RUN mode. The RMC must be in RUN mode in order for the Program Triggers to run.
2. Create a User Program that issues the Hold Current Position (5) command.
3. Create a Program Trigger that starts the user program when the RMC enters RUN mode. Use the First Scan bit as the condition.

Method

1. Set the RMC to start up in RUN mode

a. In the **Project Pane**, expand **Programming** and choose **Properties**.

b. On the **RUN/PROGRAM** page, in the **Startup Mode** section, choose **RUN**, then click **OK**.



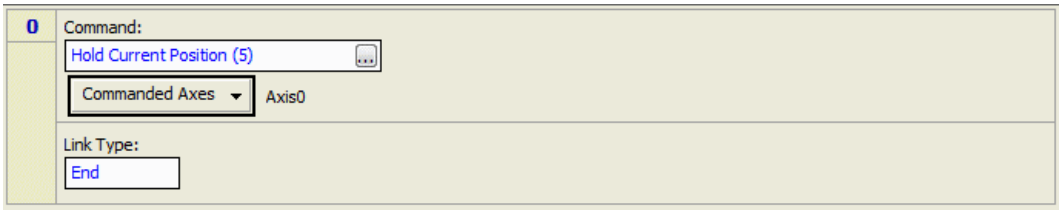
2. Create the User Program

a. Create a new user program. Name it "HoldPosition".

b. In the first step, choose the Hold Current Position (5) command, and specify the desired axes.

c. Set the Link Type to **End**.

d. The user program should now have one step and look like this:



If you are starting up the RMC at the same time as the transducers, there may be a delay before the transducers start sending data to the RMC. Therefore, you may wish to insert a delay before the step that issues the Hold Current Position (5) command.

e. Right-click any open space in step 0 and choose **Add Before**. This will add a step before step 0.

f. Select step 0, then, in the Step Editor toolbar, click the **Remove Command** button .

g. In the step 0 **Link Type** box, choose **Delay**. In the **Time to Delay** box, enter the number of seconds to delay, such as 5. The program should now look like this:

Notice that the first step does not issue a command. It simply waits for 5 seconds.

3. Create the Program Trigger

a. Open the Program Triggers Editor.

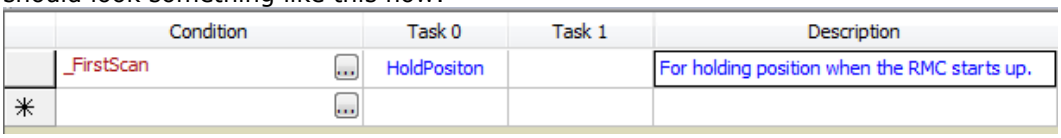
b. In the last row, double-click the **Condition** cell. The Expression Editor will open.

c. On the **Tags** tab, expand **Controller** and double-click **First Scan**. "_FirstScan" will be entered into the Expression box.

d. Click **OK** to accept the expression.

e. In one of the **Task** columns, choose the user program you created in step 2.

f. In the **Description** box, enter a description of the new item. The Program Triggers Editor should look something like this now:



4. Download the Programming and Update Flash.

a. To download the programming, in the Project pane, right-click **Programming** and choose **Download Programs to Controller**. If any errors are found, the Verify Results Window will list them and provide links to them. All errors must be fixed before the download can occur.

b. To update Flash, on the **Controller** menu, choose **Update Flash**. You must update the Flash or all the new changes you have made will be lost when you turn off the RMC.

5. Restart the controller

To test your programming, cycle power to the controller. You can use the Event Log Monitor to see what the controller did when it started up.

See Also

Programming Examples Overview

5.16.5. Example: Jogging an Axis

This topic provides two examples on how to program the RMC to jog on a position axis. Each example uses closed-loop moves to jog the axis, which requires that the axis has already been tuned. The examples can of course be modified to do open-loop motion.

Description

Jogging an axis typically refers to moving an axis forward or backward while a button is pressed. As soon as the button is released, the axis stops.

There are two main types of buttons that can be used for this application:

1. **Momentary "buttons" on a touch panel HMI.**
   In this case, the HMI must be programmed to write a value to a bit or register in the RMC when it is pressed, and write another when is released.

   Note:
   For help on how to communicate with the RMC via an HMI, see the Communicating with HMIs topic.

2. **Momentary voltage switches, buttons, or a joystick.**
   In this case, the switches, buttons, or joystick must be wired to two discrete inputs on the RMC.

Method

Programming the RMC for the jog function consists of the following steps:

1. **Set up the buttons**
   - If you are using an HMI, this requires placing two buttons on the screen; one to move the axis forward, and another to move it backward. Each button press should cause a value to be written to a variable in the RMC.
   - If you are using a joystick, or buttons or switches, the outputs should be wired to two discrete inputs on the RMC.
2. **Create three User Programs:**

1.
   a. User Program 1 will consist of one step that moves the axis forward.
   b. User Program 2 will consist of one step that moves the axis backward.
   c. User Program 3 will consist of one step that stops the axis.

2. **Create four Program Triggers:**
   a. The first condition will continuously monitor the discrete input (or variable register) for moving the axis forward. When it turns "on", it will start User Program 1, which will move the axis forward.
   b. The second condition will continuously monitor the discrete input (or variable register) for moving the axis backward. When it turns "on", it will start User Program 2, which will move the axis backward.
   c. The third condition will continuously monitor the discrete input (or variable register) for moving the axis forward. When it turns "off", it will start User Program 3, which will stop the axis.
   d. The fourth condition will continuously monitor the discrete input (or variable register) for moving the axis backward. When it turns "off", it will start User Program 3, which will stop the axis.

Example 1: Using a "button" on an HMI

The example will follow the method described above.

1. **Set up the buttons**

   First, the user sets up two momentary buttons on the HMI, called **Jog Forward** and **Jog Back**. As with most HMIs, this momentary button can only write to a bit. The RMC supports writing to bits with the Allen-Bradley DF1 protocol or Modbus/RTU or Modbus/TCP. This examples shows using DF1. The momentary button writes a "1" when it is pressed, and a "0" when it is released.

   The user programs the **Jog Forward** button in the HMI to write to bit zero in the RMC variable 0 (address F56:0/0). In the RMC, the user named this variable **Jog_Forward** and defined it as a DINT (double integer).

The user also programs **Jog Back** button in the HMI is programmed to write to bit zero in the RMC variable 1 (address F56:1/0). In the RMC, the user named this variable **Jog_Back**, and defined it as a DINT type (double integer).

It is very important that these variables be declared as DINT type, because later the user will use the value of the entire register. If it were a REAL (floating point), it would be unknown what value it would take on if bit 0 were changed.

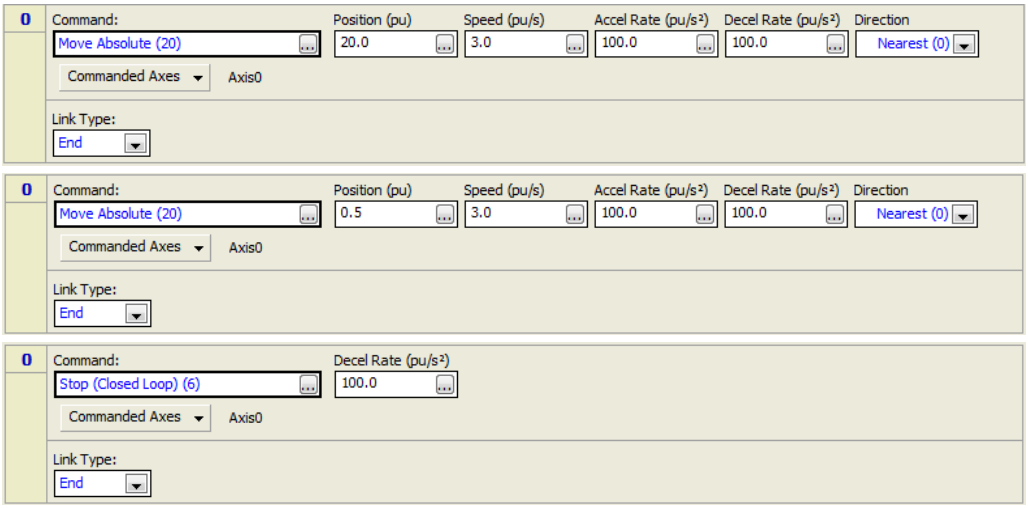
After the user has done this, the variable table looks like this:

	Tag Name	Units	Type	Retain	Initial	Description
0	Jog_Forward		DINT	☐	0	This is for jogging forward.
1	Jog_Backward		DINT	☐	0	This is for jogging backward.
2			REAL	☐	0.0	
3			REAL	☐	0.0	

Edit | Monitor

2. Create three User Programs

The user created three programs. The speeds and accels were chosen based on the system to be controlled. The user chose positions that are shortly before the end of the stroke, so if the operator does not release the button, the axis will still stop before hitting the end.



3. Create four Program Triggers conditions:

Since the HMI writes to bit 0 of the variables, the value of the entire variable will be either zero or one. Therefore, the user made the Program Trigger conditions compare the entire variables to zero or one.

	Condition	Task 0	Task 1	Description
	Jog_Forward = 1	MoveForward		
	Jog_Back = 1	MoveBack		
	Jog_Forward = 0	Stop		
	Jog_Back = 0	Stop		
✳				

Now, when the operator presses and holds the **Jog Forward** button on the HMI, it will write a 1 to bit 0 of the **Jog_Forward** variable. The Program Trigger sees that the **Jog_Forward** variable

became 1, so it starts the User Program **MoveForward**. When the operator releases the **Jog Forward** button on the HMI, it will write a 0 to bit 0 of the **Jog_Forward** variable. The Program Trigger sees that the **Jog_Forward** variable became 0, so it starts the User Program **Stop**.

The **Jog Back** button works like the **Jog Forward** button.

Example 2: Using a button or joystick with a discrete input

The user chooses to use a 3-position switch. When it is pushed to the right, one output goes high. When it is pushed to the left, the other output goes high. When it's in the middle, both outputs are low.

1. Set up the discrete inputs

The user wires the two outputs from the switch to discrete inputs on the RMC. In this example, the user chooses inputs 0 and 1 on the RMC75 D8 expansion module. Any general inputs on the RMC75 or RMC150 will work.

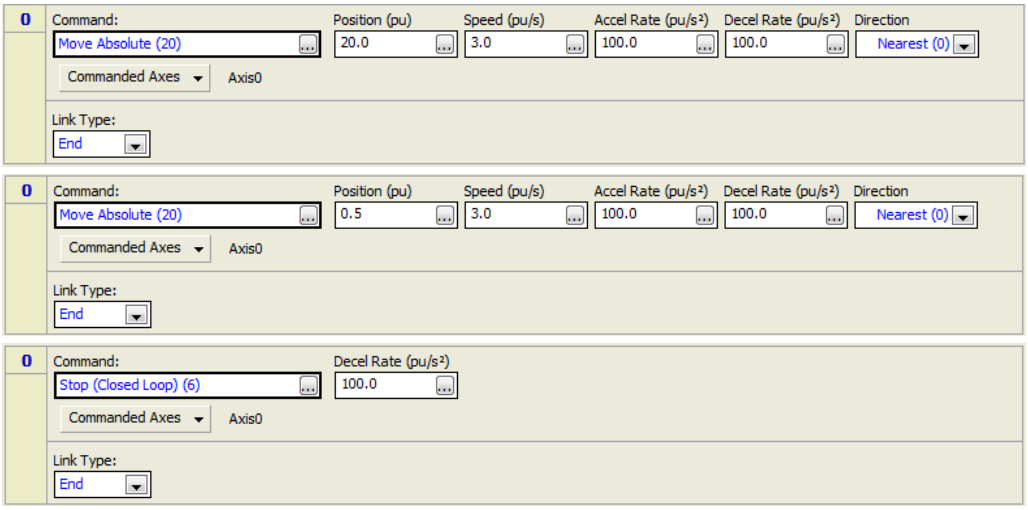
In the Discrete I/O Configuration dialog, the user defines discrete I/O points 0 and 1 as inputs and calls them **Jog_Forward** and **Jog_Back**.

After the user has done this, the Discrete I/O Configuration looks like this:

	Assigned To	Tag Name	Type	PROGRAM State	FAULT State	Description
%IX0	D8[Exp#1] I/O 0	JogForward	Input			
%IX1	D8[Exp#1] I/O 1	JogBack	Input			
%IX2	D8[Exp#1] I/O 2		Input			
%IX3	D8[Exp#1] I/O 3		Input			
%QX4	D8[Exp#1] I/O 4		Output	Hold	Off	
%QX5	D8[Exp#1] I/O 5		Output	Hold	Off	
%QX6	D8[Exp#1] I/O 6		Output	Hold	Off	
%QX7	D8[Exp#1] I/O 7		Output	Hold	Off	

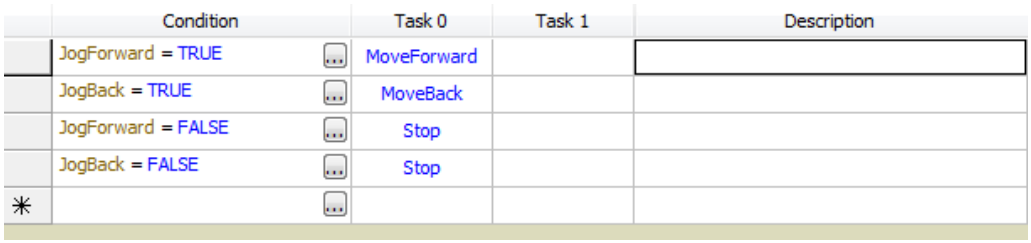
2. Create three User Programs

The user created three programs. The speeds and accels were chosen based on the system to be controlled. The user chose positions that are shortly before the end of the stroke, so if the operator does not release the button, the axis will still stop before hitting the end.



3. **Create four Program Trigger conditions:**

   If an input is on, it is TRUE. If is off, it is FALSE. Each condition compares the state of an input to TRUE or FALSE. The user created four Program Trigger conditions as shown below:



	Condition	Task 0	Task 1	Description
	JogForward = TRUE	MoveForward		
	JogBack = TRUE	MoveBack		
	JogForward = FALSE	Stop		
	JogBack = FALSE	Stop		
✳				

   Now, when the operator pushes the switch to the right, the **Jog_Forward** input becomes TRUE. The Program Trigger sees that the **Jog_Forward** input became TRUE, so it starts the User Program **MoveForward**. When the operator releases the switch, it goes to its center position and the **Jog_Forward** input becomes FALSE. The Program Trigger sees that the **Jog_Forward** input became FALSE, so it starts the User Program **Stop**.

   The **Jog_Back** input works like the **Jog_Forward** input.

See Also

Programming Examples Overview

5.16.6. Example: Creating a Timer

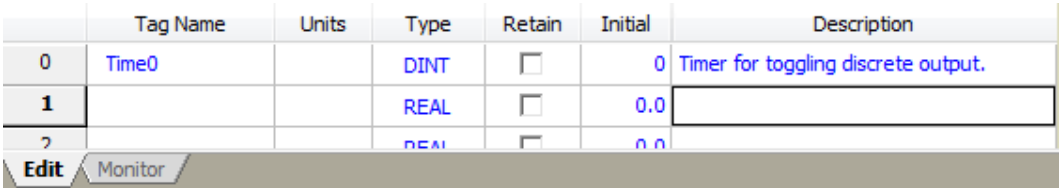
This topic provides an example of how to use the _SysMS tag to create timer functionality in the RMC. The _SysMS tags holds the number of milliseconds since the RMC powered up. It is a 32-bit DINT, and will wrap around to -2147483648 after it reaches its maximum value of 2147483647.

If you need a timer to track values less then one millisecond (e.g. the loop time is less then one millisecond), then you may wish to use the _Controller.SysTime_usec tag instead, which has units of microseconds.

Example

Consider an application that toggles a discrete output every 10 seconds. One method of achieving this is as follows. Notice that this is not necessarily the best way to achieve this application. However, the point here is to illustrate how to make a timer. This is intended to be a starting point for the reader to create more complicated timer applications.

1. **Define a Variable**
   Define a variable, called **Time0**. Define it as a DINT:

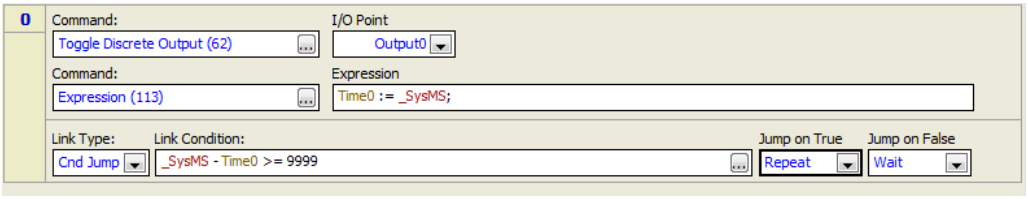


	Tag Name	Units	Type	Retain	Initial	Description
0	Time0		DINT	☐	0	Timer for toggling discrete output.
1			REAL	☐	0.0	
2			REAL	☐	0.0	

Edit | Monitor

2. **Write a User Program**
   The User Program will consist of the following:
   - Toggle the output.
   - Assign **Time0** = **_SysMS**.

- Wait until **_SysMS** minus **Time0** is equal to the time increment before repeating the step. In this case the time increment is 10 seconds, or 10000 milliseconds. One loop time will be lost in jumping to the next step. Assuming the loop time is 1 msec, the comparison will be to 10000 minus 1, which is 9999.



This program will loop forever until the task is stopped.

Handling _SysMS Wrapping

Notice that the _SysMS value in the above example will wrap its 32-bit limit eventually, but the math in the Link Condition will still be correct even through a wrap. However, if the inequality in the link condition is rearranged, it may not handle the wrapping correctly. For example, "_SysMS >= Timer0 + 9999", though mathematically identical, will not handle _SysMS wrapping.

See Also

Programming Examples Overview

5.16.7. Example: Time-out

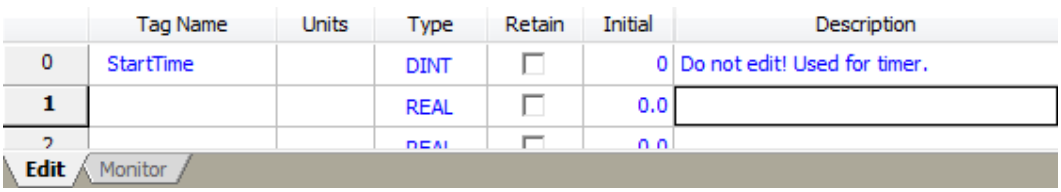
This example illustrates how to create a time-out. After issuing a move command, the user program has a time-out period (this example will use 5000 milliseconds) in which the axis must reach position. If the axis does not reach position within the time-out period, a discrete output turns on. If the axis does reach position within the time-out period, a different discrete output turns on.

This example makes use of the _SysMS tag. The _SysMS tag holds the number of milliseconds since the RMC powered up. It is a 32-bit DINT, and will wrap around to -2147483648 after it reaches its maximum value of 2147483647.

If you need a time-out that tracks values less then one millisecond (e.g. the loop time is less then one millisecond), then you may wish to use the _Controller.SysTime_usec tag instead, which has units of microseconds.

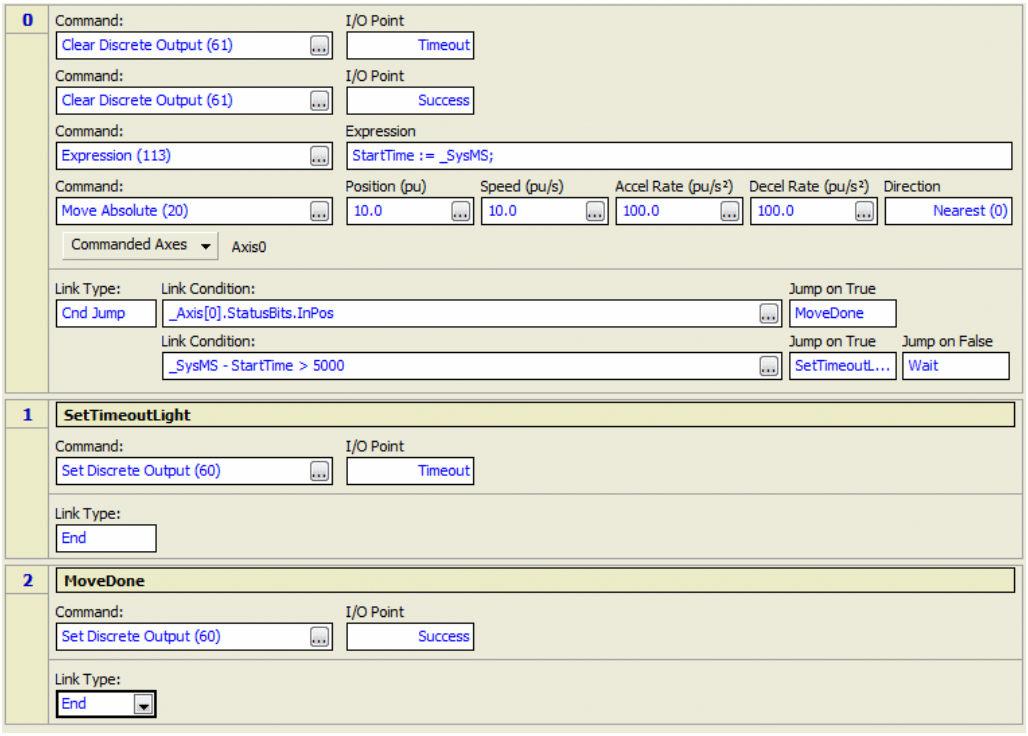
Example

1. **Define a Variable**
   Define a variable, called **StartTime**. Define it as a DINT:



2. **Write a User Program**
   The User Program will consist of the following:

- Turn off the discrete outputs.
- Set the StartTime variable to the current value of the _SysMS.
- Issue the move.
- Wait until either the axis gets into position, or 5000 milliseconds have passed since the move was issued. Jump to the correct step based on which of these events occurs first. The step that is jumped to will set one of the discrete outputs to indicate whether a time-out occurred or the axis reached position successfully.



See Also

Programming Examples Overview

5.16.8. Example: Using Arrays

This topic provides an example of using a variable array.

Description

This example will program the RMC for a cylinder sleeve installation machine for a 6-cylinder in-line engine block.

The RMC needs to control the single axis that moves the cylinder sleeve installation head to each cylinder. When it reaches a cylinder, the RMC needs to toggle a discrete output, wait 1 second, then continue to the next cylinder. The "home" position for the head is 0. The position of the cylinders varies by engine type. The PLC stores these various positions and will write them to the RMC.

Method

1. Declare Variables

Create an array to store the cylinder positions. The PLC can write the values to the array just like it can write values to any variable.

	Register	Tag Name	Units	Type	Retain	Initial	Description
0	F56:0	NextPosition		REAL	☐	0.0	This is the next posiiton to move to.
1	F56:1	Index		DINT	☐	0	This is the index for the array.
2-7	F56:2-7	⊟ Positions		REAL[6]	☐		This is the array of positions.
2	F56:2	Positions[0]		REAL	☐	0.0	
3	F56:3	Positions[1]		REAL	☐	0.0	
4	F56:4	Positions[2]		REAL	☐	0.0	
5	F56:5	Positions[3]		REAL	☐	0.0	
6	F56:6	Positions[4]		REAL	☐	0.0	
7	F56:7	Positions[5]		REAL	☐	0.0	
8	F56:8			REAL	☐	0.0	

Edit | Monitor

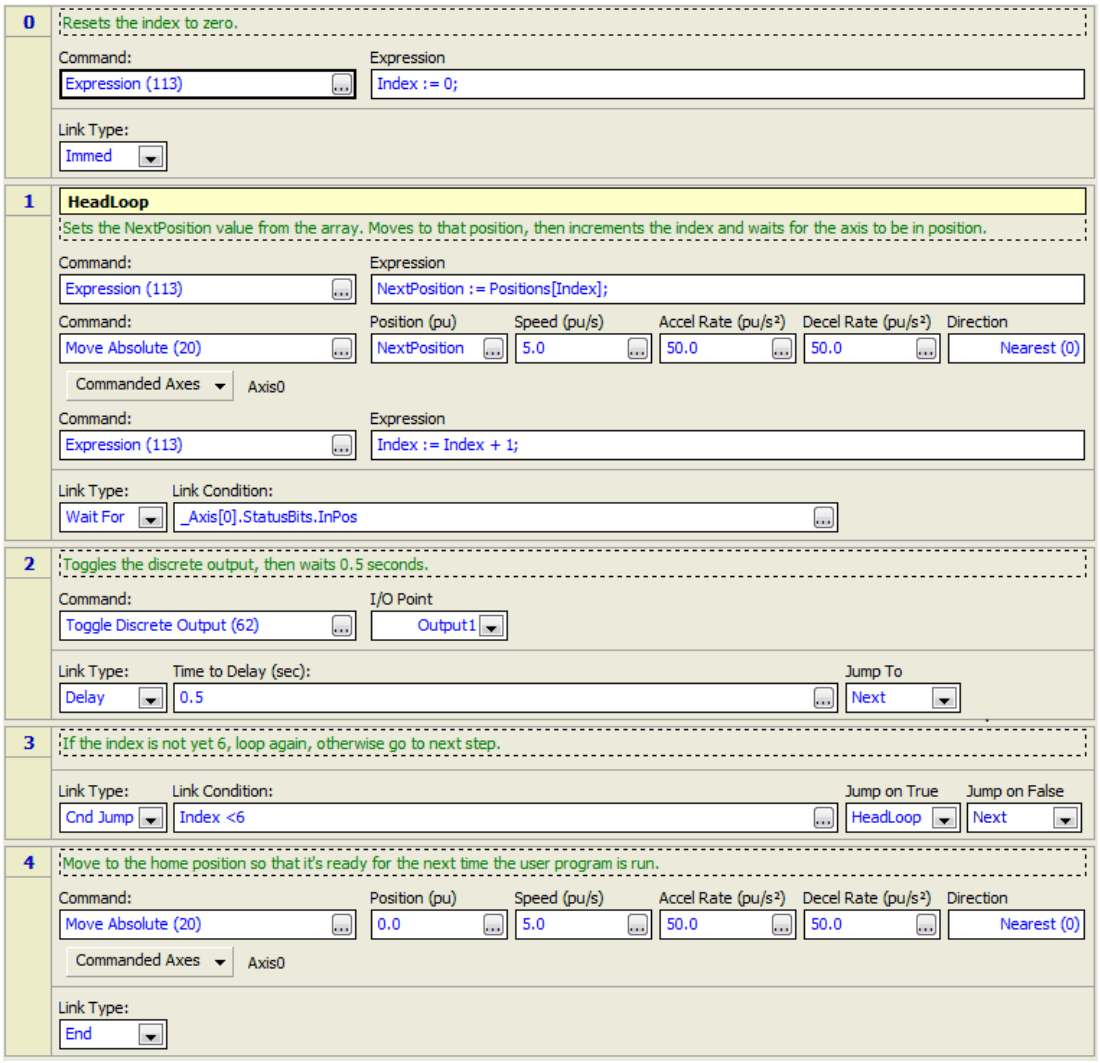
2. Declare Discrete Output

Names a discrete output as "Output1":

	Assigned To	Tag Name	Type	PROGRAM State	FAULT State
%QX1.0	RMC150E [slot 1] Out 0	Output1	Output	Hold	Off
%QX1.1	RMC150E [slot 1] Out 1		Output	Hold	Off
%IX1.0	RMC150E [slot 1] In 0		Input		
%IX1.1	RMC150E [slot 1] In 1		Input		

3. Write the User Program

The user program uses a loop to move to the positions. The label on step 1 is used by step 3 for creating a loop.



3. Downloading the Programmng

To download the programming, in the Project pane, right-click **Programming** and choose **Download Programs to Controller**.

4. Testing the Programming

After downloading the Programming node to the RMC, test it out by setting the Positions array elements to certain values on the Monitor tab of the Variable Table Editor and then running the program.

There are several ways to start a user program. In this case, the PLC may send the Start Task(90) command every time it wants to start the program. See the Running User Programs and Issuing Commands topics for details.

See Also

Programming Examples Overview

5.16.9. Application Tip: Emergency Stops

There are many ways in which emergency stops are implemented in a machine. This topic discusses possible methods of halting the RMC in the event of a machine fault situation.

One option is, of course, to shut off power to the RMC, which may not be desirable. Aside from removing power, the RMC can only programmatically respond to an estop signal.

General Considerations

In a machine fault situation, the following actions are typically desired of the RMC:

1. **Halt all the axes**
   The RMC has several levels of halts.
2. **Turn off any Enable Outputs**
   If any Enable Outputs are wired to a motor drive, or other actuator, these are typically turned off in a fault situation. The Direct Output Halt will halt the axis and turn off the axis' Enable Output.
3. **Stop User Programs**
   Any user programs that are running should be stopped to prevent commands from being issued to the axes.
4. **Stop the Program Triggers**
   The Program Triggers should be stopped to prevent user programs from starting. To stop the Program Triggers, put the RMC in PROGRAM mode.
5. **Turn off discrete outputs**
   Certain discrete outputs may need to be turned off.

Additional machine considerations:

- **Blocking Valve**
  If the RMC halts the hydraulic axis with an Open Loop or Direct Output Halt, the cylinder may drift. If drifting is unacceptable, install a blocking valve that will close when the machine faults.

Using the Fault Input to Halt Axes

If the Fault Input Auto Stop is set to a halt, then when the Fault Input becomes active, that axis will halt, and any new motion will be prevented as long as the Fault Input is active (except for the Direct Output (9) command). The polarity of the Fault Input can be set to Active High or Active Low.

Each Fault Input will only halt its corresponding axis. If you want to use the Fault Input on several axes on the RMC, you will need to wire them all. If you instead want to use only one discrete input to halt the axes, see the **Using a Discrete Input to Halt Axes** section below.

The modules listed below have a dedicated Fault Input.

RMC75	RMC150	RMC200
MA AA QA	Quad (Q)	CA4 - dedicated CV8 - input can be configured as Fault Input

For modules that do not have a Fault Input, you can use a general discrete input to halt the axes, as described in the **Using a Discrete Input** section below.

The Fault Input can also be used to put the RMC in PROGRAM mode, thereby stopping all user programs and the Program Triggers. See the **RUN/PROGRAM Mode** section below.

Using a Discrete Input to Halt Axes

There are several ways to use a single general discrete input to halt the axes:

Use a single discrete input to halt axes and enter PROGRAM mode

For any axis, you can assign the Positive Limit Input and Negative Limit Input (on any number of axes) to the same discrete input. If the AutoStops are set to halt on the Positive Limit Input and

Negative Limit Input errors, this will cause the axis to halt when that discrete input is active. The polarity of the limit inputs can set with the Limit Input Polarity parameter.

In addition, the same discrete input can be used to put the RMC in PROGRAM mode, thereby stopping all user programs and the Program Triggers. See the **RUN/PROGRAM Mode** section below.

Use a single discrete input to trigger a shutdown user program.

You can make a user program that issues the Fault Controller (8) command, and configure the Program Triggers to start the program when a discrete input turns on. This will halt all the axes, and put the RMC in PROGRAM mode. The disadvantage of this method is that the discrete input is not forcing the RMC to remain in an error state, and therefore, any motion command issued to the RMC from another source—such as a PLC or RMCTools—can cause motion.

RUN/PROGRAM/Disabled Mode

A discrete input can be configured to put the RMC in RUN mode or PROGRAM mode (RUN or Disabled for the RMC200). When the input turns on, the RMC will enter RUN mode. When the input turns off, it will be in PROGRAM mode (Disabled mode for the RMC200). This input can be a general discrete input, or any axis' Fault Input, and can simultaneously function as described in the **Using the Fault Input to Halt Axes** and **Using a Discrete Input to Halt Axes** sections above.

To configure the RUN/PROGRAM/Disabled mode input:

a. In the Project Pane, right-click **Programming** and choose **Properties**.
b. On the **RUN/PROGRAM** or **RUN/Disabled** page, check **Define a RUN/PROGRAM discrete input** or **Define a RUN/Disabled discrete input**.
c. Choose the input you wish to use and click **OK**.

Notice that the state of the RUN/PROGRAM/Disabled mode input does not always reflect the RUN/PROGRAM state, because the RUN/PROGRAM mode can be changed by other methods.

Configuring Discrete Outputs

You may wish to set certain discrete outputs either on or off when the machine faults. The Discrete I/O Configuration has options for what states the discrete outputs should take on when the RMC enters PROGRAM mode or receives a Fault Controller command.

See Also

Help Overview

6. Communication

6.1. RMC Communications Overview

The RMC75, RMC150, and RMC200 support many communication protocols, allowing almost any external controller, such as a PLC, HMI, personal computer, etc. to control the RMC and easily integrate it into the rest of the application. The RMC functions as a slave (server) device on the communications. The RMC only responds to communication requests. It does not initiate reads or writes. The host controller (PLC, HMI, etc.) must initiate all communication requests. The RMC does not perform motion control on feedback via any communication channels. It only controls motion on axes that are directly wired to the RMC's modules.

For basic information on how to read from, write to, and send commands to the RMC, see the respective communication type below.

Communication with a PLC, HMI, etc.

The communication types listed below are available in the RMC family. For details on how to set up and use each communication protocol, click on the respective link.

Communication Type	RMC75E	RMC75S	RMC75P	RMC150E	RMC200	Usage
Ethernet (Slave only)						
EtherNet/IP	✓			✓	✓	Supported by many PLCs
PROFINET	✓			✓	✓	Supported by many PLCs, primarily Siemens
Modbus/TCP	✓			✓	✓	Supported by many PLCs and most HMIs
CSP (Allen-Bradley)	✓			✓	✓	Supported by Allen-Bradley PLCs and many HMIs
FINS/UDP (Omron)	✓			✓	✓	Omron PLC protocol

Procedure Exist (Mitsubishi)	✓			✓	✓	Mitsubishi Q-Series QJ71E71-100 module, FX3U
Delta Motion Control Protocol	✓			✓	✓	Custom protocol, many uses, such as embedded
Serial (Slave only)						
DF1 (Full- and Half-Duplex) (Allen-Bradley)		✓				Supported by Allen-Bradley PLCs and many HMIs
Modbus/RTU		✓				Supported by many PLCs and HMIs
Bidirectional Protocol (Mitsubishi Q-Series)		✓				Mitsubishi Q-Series QJ71C24N module
PROFIBUS (Slave only)						
PROFIBUS-DP			✓	✓ (PROFI module)		Supported by many PLCs

Communicating Using Master Controllers

For information on how to communicate with an RMC from various master controllers, see the topics below:

Allen-Bradley Controllers via Message Block
Allen-Bradley Controllers via EtherNet/IP I/O
AutomationDirect PLCs
GE PLCs
Mitsubishi Q-Series
Omron PLCs va FINS
Omron PLCs via EtherNet/IP I/O
RSView
Siemens S7 PLCs via PROFIBUS
Siemens S7 PLCs via PROFINET
Schneider Electric PLCs via EtherNet/IP I/O
Schneider Electric PLCs via Modbus
Wonderware

Communicating from a PC
Other Master Controllers

Communicating with RMCTools

The RMCTools software can communicate with each RMC via the follow methods:

Monitor Port Type	RMC75E	RMC75S	RMC75P	RMC150E	RMC200
USB Monitor port	✓			✓	✓
Ethernet	✓			✓	✓
Serial RS-232 Monitor port		✓	✓		

Sample Projects

Delta provides free example projects for various host controllers, such as PLCs and HMIs, for communicating with the RMC. These example programs will get you up and running quickly. See the downloads page of Delta's website at deltamotion.com for example projects available for the RMC.

Finding Register Addresses

When configuring a host controller to communicate with the RMC, you must enter addresses for the RMC registers that you wish to read from or write to. The address must be in the correct format for the protocol that is being used. To find the RMC addresses in the correct format, use any of the following:

Ways to find RMC addresses:

1. In RMCTools editorss
   RMCTools displays the addresses of many registers, such as in the Axis Tools. To change the address format, right-click any register address cell, choose **Address Formats**, and choose the desired format.
2. Use the Address Maps in RMCTools to browse all the register addresses for any addressing method.
3. Use the Register Map help topic to browse all the register addresses for any addressing method.

Indirect Data Map

The Indirect Data Map is important for several communication types and especially for the RMC200. It maps any data items from anywhere in the RMC to one location. This allows scattered data to be consolidated for efficient I/O and messaging communications.

RMCLink .NET Assembly and ActiveX Control

Use RMCLink to communicate with an RMC from numerous programming languages and applications on a PC. See the RMCLink topic for details.

Discrete I/O

Discrete I/O augments the communications of the RMC. Discrete I/O is typically faster and more deterministic than the communications, and is therefore well-suited for starting a sequence in the RMC at a specific time. For details, see the Discrete I/O for Communications topic.

RMC150-to-RMC150 Communications

The RMC150 Universal I/O Module provides limited inter-RMC150 communications via SSI. The communication transaction is performed each loop time, allowing for tight synchronization between controllers, although only one 32-bit register can be sent and one received. For details, see the Universal I/O Module topic.

See Also

Ethernet Overview | PROFIBUS Overview | Serial Overview

6.2. Communication Statistics

To access the Communication Statistics:
In the **Project** pane, select the desired controller. On the **Controller** menu, click **View Communication Statistics**.

The Communication Statistics window displays a number of diagnostic counters that are useful for monitoring and troubleshooting the communication via the primary communications port and the Monitor Port.

The Communications Statistics contains the following sections:

- **RMCTools Connection Path**
  Provides statistics on the current RMCTools Connection path.

- **Ethernet**
  Provides Ethernet details on controllers with Ethernet.
  - **Summary:** Provides a general summary of the RMC's current Ethernet settings.
  - **Ethernet Statistics:** Provides statistics on various components of the Ethernet communication.
  - **EtherNet/IP Diagnostics:** Provides information specific to the EtherNet/IP protocol.
  - **PROFINET Diagnostics:** Provides information specific to the PROFINET protocol.
  - **IP Statistics:** Provides details on the internet layer of Ethernet.
  - **ICMP Statistics:** Provides details on the ICMP protocol.
  - **UDP Statisitcs:** Provides details on the UDP protocol.
  - **TCP Statisitics:** Provides details on the TCP protocol.

- **PROFIBUS**
  Provides statistics on the **PROFIBUS** communications.

- **Internals**
  Provides statistics on the RMC's internal tasks.

Clearing Statistics

Some of the communication statistics can be cleared (rest to zero). To clear a statistic, select a folder in the navigation tree, then click **Clear Statistics** on the toolbar.

See Also

Ethernet Overview | Communication Statistics | Monitor Port

6.3. Monitor Port — USB or RS-232

The Monitor port is used primarily to communicate from RMCTools to the RMC. The monitor port type varies by RMC:

Controller	Monitor Port
RMC75E	USB
RMC75S	RS-232
RMC75P	RS-232
RMC150E	USB
RMC200	USB

Connecting from RMCTools to the RMC

Use a procedure based on your RMC type and connection method:

USB (RMC75E, RMC150E, RMC200)

1. Make sure RMCTools is installed first.
2. Connect a USB cable from the PC to the USB Monitor port on the RMC.
3. Open a project in RMCTools.
4. In the Project Pane, right-click the controller and click **Connection Path**.
5. Choose **USB** and click **Browse**. RMCTools will list all the RMCs connected via USB. If your RMC does not appear, try disconnecting and reconnecting the USB cable.
6. Choose the desired RMC from the list. If you have multiple RMCs connected via USB, use the Device ID to determine which one to choose from the list. Notice that the Device ID is the serial number for the RMC75 and MAC address for the RMC150. Click **OK**.
7. Click **Go Online.**

Ethernet (RMC75E, RMC150E, RMC200)

Use these steps to connect from RMCTools to the RMC using Ethernet:

1. Connect the RMC75 and the PC to the same Ethernet network. For more information, see the RMC Ethernet Setup and Setting Up a Standalone TCP/IP Network topics.
2. Open a project in RMCTools.
3. In the Project Pane, right-click the controller and click **Connection Path**.
4. Choose **Ethernet** and click **Browse**. RMCTools will list all the RMCs connected via Ethernet. If your RMC does not appear in the box, see the Troubleshooting RMCTools Ethernet Connection topic.
5. Choose the desired RMC. If you have multiple RMCs connected to the Ethernet network, use the part number and MAC address to determine which one to choose from the list.
6. If the RMC does not have an IP address, click the **Configure Device** button to set the IP Address settings.
7. Click **OK**.
8. Click **Go Online.**

RMC75P or RMC75S

Use these steps to connect from RMCTools to the RMC using the serial monitor port:

1. Connect a DB-9 null-modem cable from the PC to the RS-232 Monitor port on the RMC. See the **RS-232 Monitor Port Wiring** section below for cable details.
2. Open a project in RMCTools.
3. In the Project Pane, right-click the controller you wish to connect to and click **Connection Path**.
4. Choose the COM port that the RMC is connected to.
5. Click **Go Online.**

USB Monitor Port Details (RMC75E, RMC150E, RMC200)

The USB Monitor port is a Universal Serial Bus (USB) "B" port. RMCTools communicates with the RMC75E and RMC150E via the USB Monitor port. The USB Monitor port cannot be used for any other purpose.

Use a standard USB "A" to "B" cable to connect your PC to the USB Monitor port.

The USB port is intended for configuration, programming, and troubleshooting purposes only. Do not leave it connected during normal machine operation.

The USB port is sensitive to electrical noise. In some cases communication may be lost. If communication cannot be reestablished, you may need to unplug and reconnect the cable. USB connector shielding is grounded through case ground.

RS-232 Monitor Port (RMC75S, RMC75P)

The RS-232 Monitor port is a DTE DB9 RS-232 serial port. RMCTools communicates with the RMC75S or RMC75P via the Monitor port. The RS-232 Monitor port may also be used to communicate with other devices via the DF1 serial protocol.

Using the RS-232 Monitor Port for Communication with a Host Controller

The RS-232 Monitor Port is primarily intended for communication with RMCTools on a PC. However, the RS-232 Monitor port may be used to communicate with host devices, such HMIs, PLCs, etc. The RS-232 Monitor port can be used at the same time as the other communication port, whether it be Serial or PROFIBUS.

Since the RS-232 Monitor Port communication protocol is fixed at the settings listed below, it can only communicate with devices that support the DF1 protocol. For details on how to use the RS-232 Monitor port for communications, see the Using Serial Communications topic.

RS-232 Monitor Port Settings:

- DF1 Full-Duplex
- 38400 Baud
- 8 Data Bits
- No Parity
- 1 Stop Bit
- CRC

RS-232 Monitor Port Wiring

Cable

Must be a female 9-pin DB9 on the RMC end. The following options are available for the cable between the PC and the RMC75 serial Monitor port:

- Generic null-modem (female on both ends) serial cable
- Allen-Bradley PLC serial cable
- Modicon Modbus cable with a 9-pin male-to-female gender changer
- Siemens SIMATIC 505 Programmable Controller serial cable
- Any cable will work if pins 2, 3, and 5 on the RMC75-end of the cable are connected as shown below.

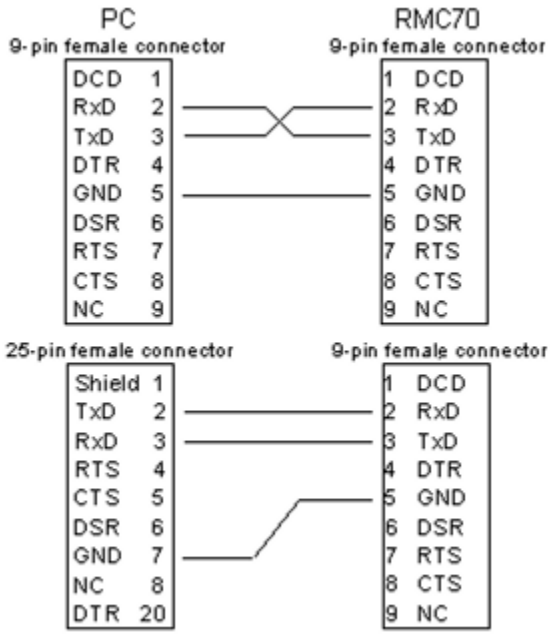
Pin-Out

Pin assignments on the RS-232 Monitor Port:

Pin #	RS-232 Function
1	DCD- Not used by RMC75
2	RxD - Receive Data
3	TxD - Transmit Data

4	DTR - Not used by RMC75
5	GND - Common
6	DSR - Not used by Monitor Port
7	RTS - Not used by Monitor Port
8	CTS - Not used by Monitor Port
9	Not Connected

Wiring



Note:
The DB9 shell is connected to the RMC75 Case.

For more details, see the RS-232 Wiring for the RMC75 topic.

See Also

Communications Overview

6.4. Communicating with HMIs

The RMCs can communicate with many types of HMIs (Human-Machine Interfaces). This topic describes the protocols the HMI must support in order to communicate with each of the RMC series motion controllers. It also describes how to configure the communications and the basics of actually communicating, such as issuing commands, writing and reading, etc.

Example programs for certain HMIs are available on the downloads page of Delta's website. These can help you get up and running quickly.

Requirements of the HMI

Requirement 1: Must Support a Compatible Protocol

The HMI must support a protocol that is compatible with the specific RMC. Notice that certain protocols have specific requirements that the HMI must also fulfill.

Comm Type/ RMC	Supported Protocols	Notes
Ethernet: RMC75E RMC150E RMC200	CSP (Allen-Bradley)	RMC75/150: Must support registers F7 to F255, or L7 to L255
	EtherNet/IP (Allen-Bradley)	RMC75/150: Must support registers F7 to F255, or L7 to L255
	FINS/UDP (Omron)	
	Modbus/TCP	RMC75/150: Preferably support up to address 65535.
Serial: RMC75S RMC75P Monitor Port	DF1 (Full- and Half-Duplex) (Allen-Bradley)	RMC75/150: Must support registers F7 to F255, or L7 to L255
	Modbus/RTU	RMC75/150: Preferably support up to address 65535.
PROFIBUS-DP: RMC75P RMC150E	PROFIBUS-DP	

Requirement 2: 32-bit Floating Point Support

The HMI *must* support 32-floating point numbers. The HMI must be able to write individual floating-point numbers and read individual floating-point numbers. In order to issue commands to the RMC, the HMI must also be able to write to multiple floating-point numbers at once.

Optional Requirement 3: Read Bits in a Register

Since the RMC does not support reading boolean data types directly, the HMI should be able to view individual bits in a register read via Modbus/TCP or Modbus/RTU, or view individual bits in an F or L register read via CSP or DF1. This allows the HMI to display status items such as Axis Enabled, Halted, etc.

Configuring the HMI Communications

Using Ethernet:

For most devices, use Modbus/TCP. If that option is not available, set up the HMI communications as if it were talking to a SLC 500 PLC using CSP or EtherNet/IP.

For more details on setting up ethernet, see the Ethernet Overview topic.

Using Serial RS-232 or RS-485:

For most devices, use Modbus/TCP. If that option is not available, set up the HMI communications as if it were talking to a PLC. When selecting the protocol, choose an Allen-Bradley SLC 500 PLC with DF1. If the SLC500 is not available, choose a PLC-5 or a MicroLogix.

If you are connecting the HMI to the RS-232 Monitor port on the RMC75S or RMC75P, make sure the PLC communication settings are identical to the RS-232 Monitor Port settings: DF1 Full-Duplex, 38400 Baud, 8 Data Bits, No Parity, 1 Stop Bit, CRC.

If you are connecting the HMI to the communications serial port on the RMC75S, make sure you configure the RMC75S serial settings identically to the PLC communication settings.

Using the Communications

Finding and Using Register Addresses

Read and write to registers in the RMC just as you would with a PLC.

Some RMC register addresses are displayed in RMCTools, such as in the Axis Tools, Indirect Data Map, and the Variable Table. To view those addresses in various formats, right-click the address and choose **Address Formats**. To find other RMC addresses, use the Address Maps in RMCTools, or the Register Map help topic.

The RMCs contain only 32-bit registers. Most are floating-point registers, but some are double integers (DINT) or double words (DWORD).

Issuing Commands

Issuing commands consists of writing to the command registers. The HMI must be able to write to several registers at once. See the Issuing Commands topic for more details.

Viewing Individual Bits

The RMC supports reading entire registers, not reading individual boolean data. Therefore, to get bit data from the RMC, the HMI must be able to read an entire word and look at just one bit in the word.

Most HMIs can read individual bits from the RMC if they use Modbus/TCP or Modbus/RTU. With the Allen-Bradley CSP or DF1 protocols, viewing individual bits may be difficult for some HMIs because the RMC registers can only be accessed as F or L register types.

Writing Individual Bits

The RMC supports writing entire registers, not reading individual boolean data. Therefore, to write individual bits, the HMI must be able to read an entire word and look at just one bit in the word. This usually involves reading the register from the RMC, modifying the bit, then writing the register back to the RMC. Many HMIs cannot do this. Therefore, in order to write a bit, you may need to create a user program in the RMC to write the bit, then have the HMI trigger the user program.

Tips on Using the RMC with an HMI as a Stand-alone System

For small machine applications, the RMC is fully capable of acting as a stand-alone system together with an HMI. Several features in the RMC make this possible for many types of systems:

- **Start in RUN Mode**
  The RMC can be set to start in RUN mode. It is immediately ready for motion without requiring the user to enable it from the HMI. See the Run/Program Mode topic for details.
- **Start a User Program on Startup**
  If the RMC is set to start up in RUN mode, it is possible to start a User Program immediately. This is accomplished by using the _FirstScan bit in the Program Triggers. See the Program Triggers topic for details.
- **User Programs**
  User Programs can perform complex motion sequences and mathematical operations. See the User Programs topic for details.
- **Program Triggers**
  The Program Triggers allow the RMC to respond to discrete inputs to any other condition, without requiring that the user explicitly command it to do something. See the Program Triggers topic for details.

Sample Programs

Delta's website provides sample HMI programs for communicating with the RMC. These programs demonstrate reading and writing registers and issuing commands. Using the demo programs can drastically reduce your development time.

The sample programs are available from the downloads page.

See Also

Communications Overview | Using Serial Communications

6.5. Discrete I/O for Communications

Discrete I/O augments the communications of the RMC. Discrete I/O is typically faster and more deterministic than the communications, and is therefore well-suited for starting a sequence in one or more RMCs at a specific time.

Some example uses of discrete I/O for communications:

- **Start or Stop User Programs**
  Use the Program Triggers to start and stop User Programs based on an input.
- **Affect flow of User Programs**
  Use inputs in the Conditional Jump link type to make link conditions that affect the flow of the User Programs.
- **General input and output**
  Turn outputs on and off from within User Programs and use inputs in Expressions.

For details on using these features, see the following topics:

- Using Discrete I/O
- Configuring Discrete I/O

For a list of modules that support discrete I/O, see the Discrete I/O topic.

See Also

Communications Overview | Discrete I/O

6.6. Command Request and Acknowledge Bits

The **Command Request** and **Command Acknowledge** bits provide synchronization of commands and status information between the PLC and RMC. Using these bits is optional.

These bits are available only when writing commands directly to the Command Area registers from an external host controller, such as a PLC.

For many PLC applications, they are very important. When reading status bits and status registers, these bits let the PLC know that the data being requested is valid. For example, after a Move command is issued, the In Position status bit is not valid until the RMC has informed the PLC that it received the command. The RMC does this by matching the Command Acknowledge bit to the Command Request bit.

Note:
The Command Request and Command Acknowledge bits are independent from the Sync Register that is used in EtherNet/IP and PROFINET cyclic I/O.

PROFIBUS

These bits are available only for the Basic/Enhanced modes. The PROFIBUS I/O mode does not use the command area, and instead uses a simple method to handshake data between the PLC and RMC.

For details on using the REQ and ACK bits with PROFIBUS Basic/Enhanced modes, see the Basic/Enhanced PROFIBUS Modes topics.

Ethernet and Serial (RS-232/485)

For Ethernet and serial (RS-232/485) communication, each axis has a Command Request and Command Acknowledge bit.

- **Command Request Bit (REQ)**

  To set the Command Request bit, add 256.0 to the command. For example, writing 20.0 to the Command register would issue the Move Absolute command with the REQ bit cleared. Writing 276.0 to the Command register would issue the Move Absolute command with the REQ bit set.

- **Command Acknowledge Bit (ACK)**

  The Command Acknowledge bit is bit 31 of the Status Bits register. When a command has been received by the RMC, it will update the ACK bit in the Status Bits register to match the REQ bit of the command.

  The ACK bits of all axes whose Command registers are written to in a single transaction will be modified at the same time. This allows the user to take a shortcut and, if they are writing to all axes every time they issue a command to even a single axis, only check the ACK bit on axis 0 (compare examples 2 and 3 below).

The RMC keeps track of the REQ and ACK bits for each communication channel. The REQ and ACK bits of one communication channel will not interfere with the REQ and ACK bits of another channel. For example, if two PLCs are connected to an RMC75E via Ethernet, each can independently use the REQ and ACK bits. Likewise, the REQ and ACK bits of a PLC connected to the RMC75S serial port will not interfere with the REQ and ACK bits of another PLC connected to the Monitor port.

Note:
This means that the Ack bit will NOT be visible in RMCTools, since it uses a different communications channel!

The REQ and ACK bits will work correctly even if the Status Bits register is read via the Indirect Data Map.

The ACK bit defaults to 0 when the RMC powers up. When starting up a PLC program, the PLC can write zeros to all the Command registers, which will immediately set the ACK bit for each axis to zero, giving them a known state.

Examples

The examples below refer to several RMC registers. Their addresses in each RMC are given here for your convenience:

Register Name	RMC75 Address	RMC150 Address	RMC200 Address
Axis 0 Command register	%MD25.0	%MD40.0	%MD16.0
Axis 0 Command registers	%MD25.0-9	%MD40.0-9	%MD16.0-9
Axis 0 Status Bits register	%MD8.0	%MD8.0	%MD256.0
Axis 1 Command register	%MD25.10	%MD40.10	%MD16.10
Axis 1 Command registers	%MD25.10-19	%MD40.10-19	%MD16.10-19
Axis 1 Status Bits register	%MD9.0	%MD9.0	%MD257.0
Axis 0 ACK bit	bit 31 in %MD8.0	bit 31 in %MD8.0	bit 31 in %MD256.0
Axis 1 ACK bit	bit 31 in %MD9.0	bit 31 in %MD9.0	bit 31 in %MD257.0

Example 1: Commanding a Single Axis

1. On startup, the user writes 0.0 to the Axis 0 Command register, and clears an internal coil in the PLC, which we'll call **Axis0Req**. Now the **REQ** and **ACK** bits are set to zero.

2. When it is time to write a new command, the user compares the Axis 0 ACK bit (bit 31 of the Axis 0 Status Bits register) with **Axis0Req**. If they are not equal, he will wait until they are before issuing the next command. Otherwise, a command is still being received, and the synchronization will be lost by not waiting.
3. When the controller is ready (**ACK** = **Axis0Req**), the user will toggle **Axis0Req**, and write a single block of 10 registers to the Axis 0 Command registers, including the command, command parameters, and **Axis0Req** stuffed in at logical bit 8. Since this is typically a FLOAT write, the user will add 256.0 to the command if **Axis0Req** is set, or 0.0 if it is not set.
4. Before looking at the Axis 0 Status Bits register or otherwise assuming that the command has been received, the user must ensure that the Axis 0 **ACK** bit is equal to **Axis0Req** again.
5. Repeat steps 2-4 for issuing additional commands.

Example 2: Commanding Two Axes from the PLC (single ACK/REQ pair):

1. On startup, the user writes 0.0 to the Axis 0 Command register, and clears an internal coil in the PLC, which we'll call **AxisCmdReq**. Now the **REQ** and **ACK** bits are set to zero.
2. When it is time to write a new command to either axis, the user compares the Axis0 **ACK** bit with **AxisCmdReq**. If they are not equal, he will wait until they are before issuing the next command. Otherwise, a command is still being received (on either axis), and the synchronization will be lost by not waiting.
3. When the controller is ready (Axis0 **ACK** = **AxisCmdReq**), the user will toggle **AxisCmdReq**, and write two commands (10 regs each) to the Axis 0 and Axis 1 command registers, including the command, command parameters, and **AxisCmdReq** stuffed in at logical bit 8 of the Axis 0 command (regardless of which axis the command is to). It is important that all 20 registers be written regardless of which of the 2 axes is receiving a command (write zeros to any axis that is not to receive a command).
4. Before looking at EITHER axis's Status Bits register or otherwise assuming that the command has been received, the user must ensure that the Axis0 **ACK** bit is equal to **AxisCmdReq** again.
5. Repeat steps 2-4 for issuing additional commands.

Example 3: Commanding Two Axes from the PLC (one ACK/REQ per axis):

1. On startup, the user writes 0.0 to the Axis 0 and Axis 1 command registers, and clears internal coils in the PLC called **Axis0Req** and **Axis1Req**. Now the **REQ** and **ACK** bits for each axis are set to zero.
2. When it is time to write a new command to an axis, the user compares the **ACK** bit for the axis/axes to be commanded with the corresponding **Axis*n*Req** bit(s). If they are not equal, he will wait until they are before issuing the next command. Otherwise, a command is still being received, and the synchronization will be lost by not waiting. If commands need to be issued simultaneously to both axes, both axes' **ACK**/**REQ** bits should be required to be equal.
3. When the controller is ready (**ACK** = **REQ** for the necessary axis/axes), the user will toggle **Axis*n*Req** for each axis receiving a new command, and write the command(s) (10 registers each) to the command register for that axis, including the command, command parameters, and **Axis*n*Req** stuffed in at logical bit 8 of each Command register.
4. Before looking at any given axis's Status Bits register or otherwise assuming that a command on the axis has been received, the user must ensure that the **ACK** bit in the status register for that axis is equal to **Axis0Req** or **Axis1Req**.
5. Repeat steps 2-4 for issuing additional commands.

See Also

Issuing Commands | Status Bits

6.7. Ethernet

6.7.1. Ethernet Communications Overview

The RMC75E, RMC150E, and RMC200 provide Ethernet communications. The RMC performs as a slave, requiring a master to control it. The RMCs support multiple simultaneous TCP connections. This means multiple devices can communicate with the RMC at the same time, using any of the supported protocols listed below.

Using Ethernet Communications

The RMC Ethernet can be used in several ways:

Communicating from RMCTools

RMCTools can communicate directly with the RMC over Ethernet. This connection provides the fastest update rate for registers and plots in RMCTools. Ethernet also provides the ability of remotely connecting to an RMC. See the Using Ethernet with RMCTools topic for details.

Note: When you first set up the controller, it is easier to use the USB port for communicating with the RMC from RMCTools than Ethernet because USB does not require setting up an IP address.

RMCTools uses port 44818. If your PC has a firewall, make sure it allows connections to port 44818. If the RMC is behind a firewall, make sure the firewall forwards port 44818 to the RMC's IP address.

Communicating from PLCs, HMIs, and Other Devices

Communicating with the RMCs from PLCs, HMIs, or other master Ethernet controllers can be done in one of several methods:

- **PLC Ethernet Emulation**

  The RMC responds to several common industrial Ethernet protocols and can *emulate,* or act like, several common PLCs. If your device supports reading and writing to registers in any of these PLCs, then your device should be able to communicate with the RMC. See Using the RMC's PLC Ethernet Emulation for details.

- **Standard Industrial Ethernet Protocols**

  The RMC controllers support several common industrial Ethernet protocols, including Modbus/TCP, PROFINET and EtherNet/IP. If your device can make requests in any of the RMC's supported industrial Ethernet protocols, then it can likely communicate with the RMC. See the **Supported Ethernet Protocols** section below for details.

- **.NET Assembly or ActiveX Control**

  Applications that are running on a Windows PC and that support .NET Assemblies, ActiveX Controls, or Microsoft Component Object Model (COM) can use Delta's RMCLink .NET Assembly and ActiveX Control. See the RMCLink topic for details.

- **Direct over TCP or UDP**

  Third-party or custom controllers that do not support any of the above three methods but can send and receive either TCP or UDP packets directly can communicate with the RMC by manually building and parsing packets over TCP or UDP, using Delta's simple DMCP protocol. See the Communicating Directly over TCP and Communicating Directly over UDP topics for details.

If your PLC, HMI or other device does not support any of the above methods, please contact a Delta Computer Systems sales engineer to discuss your device. Delta strives to support all major Ethernet devices, and is interested in knowing about devices that the RMC does not support.

Note: The RMC does not support any of the native Ethernet protocols built into Windows. That is, it does not support Web browsers, FTP, e-mail, and browsing through Network Neighborhood.

Configuring the RMC Ethernet Communications

Setting up the RMC Ethernet communications usually requires entering only a few TCP/IP parameters. Additionally, advanced users can choose to disable auto-negotiation and instead manually choose 10 or 100 Mbps and full- or half-duplex.

See the RMC Ethernet Setup topic for details.

Supported Ethernet Protocols

The RMCs support the Ethernet protocols listed below. It is not necessary to select in the RMC the application protocol or device with which you will be communicating. The RMC will automatically respond to all supported protocols.

	RMC75E	RMC150E	RMC200
EtherNet/IP	✓	✓	✓
PROFINET	✓	✓	✓
Modbus/TCP	✓	✓	✓
CSP (also called DF1 over Ethernet)	✓	✓	✓
FINS/UDP (Omron)	✓	✓	✓
Procedure Exist (Mitsubishi Q-Series)	✓	✓	✓
Delta Motion Control Protocol	✓	✓	✓

If your Ethernet device supports one of the protocols listed above, then it may be able to communicate with the RMC. Read more about the protocols to make sure. There may be subtle problems with using some devices with the RMC. For example, a device that proclaims Modbus/TCP support may only be a slave device, and since the RMC is also a slave, neither device will initiate transfers, preventing the devices from being able to work together.

For a list of PLCs known to be compatible with the RMC, see the Using the RMC's PLC Ethernet Emulation topic. For lower-level details on the supported protocols, see the RMC Ethernet Protocols topic.

Troubleshooting Ethernet

The Event Log and Communication Statistics provide information that can help troubleshoot the Ethernet communications.

Unreliable RMCTools Connections

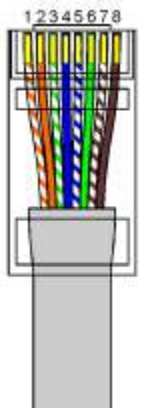
Slow or poor Ethernet connections between RMCTools and an RMC may cause lost connections. Use the **Communications: Ethernet** section of the RMCTools Options dialog to choose Ethernet timeouts intended for slow connections.

Wiring

Ethernet cables that go outside the control cabinet or can be exposed to transients from motor drives or other switched loads should be shielded to avoid communication interruptions.

For the RMC75E, RMC150E, and RMC200 use straight-through Cat5, Cat5e, or Cat6 Ethernet cable with an RJ45 connector.

Ethernet Straight Through (8-wire) Cable



1. Orange/White
2. Orange
3. Green/White
4. Blue
5. Blue/White
6. Green
7. Brown/White
8. Brown

See Also

Communications Overview

6.7.2. Using Ethernet with RMCTools

RMCTools communicates directly with the RMC75E, RMC150E, and RMC200 over Ethernet.

Connecting to the RMC via Ethernet

Use these steps to connect from RMCTools to the RMC75E, RMC150E, or RMC200 using Ethernet:

- Connect the RMC and the PC to the same Ethernet network. For more information, see the RMC Ethernet Setup and Setting Up a Standalone TCP/IP Network topics.
- Open a project in RMCTools. For more details, see the New Project and New Controller wizard help topics.
- In the Project Pane, right-click the controller you wish to connect to and click **Connection Path**.
- Choose **Ethernet** and click **Browse**. RMCTools will list all the RMCs connected via Ethernet. If your RMC does not appear in the box, see the Troubleshooting RMCTools Ethernet Connection topic.
- Choose the desired RMC. If you have multiple RMCs connected, use the part number and MAC address to determine which one to choose from the list.
- If the RMC does not have an IP address, click the **Configure Device** button to set the IP Address settings.
- Click **OK**.
- Click **Go Online**.

Unreliable RMCTools Connections

Slow or poor Ethernet connections between RMCTools and an RMC may cause lost connections. Use the **Communications: Ethernet** section of the RMCTools Options dialog to choose Ethernet timeouts intended for slow connections.

RMCTools Ethernet Port

RMCTools uses TCP port 44818 for the main communications, and port 1324 for browsing for controllers on the Ethernet network. Very old firmware on some controllers may require port 50000. If your PC has a firewall, make sure it allows connections to these ports. If the RMC is behind a firewall, make sure the firewall forwards these ports to the RMC's IP address.

See Also

Troubleshooting RMCTools Ethernet Connection | Monitor Port | Ethernet Overview

6.7.3. Using the RMC's PLC Ethernet Emulation

The RMC Ethernet controllers automatically respond to requests from several common industrial Ethernet protocols, effectively simultaneously emulating several common PLCs. If your device supports reading and writing to registers in any of the PLCs listed below, then your device should be able to communicate with the RMC:

- **Allen-Bradley SLC5/05 and PLC-5**

  The RMC emulates these controllers by responding to DF1 requests over the CSP and EtherNet/IP protocols. See the CSP, EtherNet/IP, and DF1 Addressing topics for details.

- **Omron CS/CJ PLC**

  The RMC emulates these controllers by responding to requests over the FINS/UDP protocol. See the FINS/UDP and FINS Addressing topics for details.

- **Modbus/TCP Controller**

  The RMC emulates a generic Modbus/TCP controller, responding to requests made over Modbus/TCP. See the Modbus/TCP and Modbus Addressing topics for details.

- **Mitsubishi Ethernet Controller**

  The RMC7 and RMC150 emulate a Mitsubishi Ethernet controller listening for Procedure Exist requests. See the Mitsubishi Procedure Exist Protocol topic for details.

Note: Even if your device cannot communicate with any of the above PLCs, it may still be able to communicate with the RMC if it can make requests in any of the RMC's supported industrial Ethernet protocols. See the **Supported Ethernet Protocols** section in the Ethernet Overview topic for details.

Instructions for Specific PLCs

Delta has provided instructions for setting up and using the following master Ethernet controllers for communication with the RMC:

- Allen-Bradley Controllers via Message Block
- Allen-Bradley Controllers via EtherNet/IP I/O
- AutomationDirect PLCs with the RMC
- GE PLCs with the RMC
- Mitsubishi Q-Series PLCs with the RMC
- Omron PLCs via FINS
- Omron PLCs via EtherNet/IP I/O
- RSView with the RMC
- Siemens S7 PLCs via PROFIBUS
- Siemens S7 PLCs via PROFINET
- Schneider Electric PLCs via EtherNet/IP I/O
- Schneider Electric PLCs via Modbus

Online Examples

Delta has several example PLC and HMI programs for communicating with the RMC. These examples will speed up your application development time. Refer to the Downloads section of Delta's website at https://deltamotion.com.

Supported Ethernet Devices

Following is a partial list of devices that are known to communicate with the RMC:

- Allen-Bradley Ethernet PLC-5 (1785-L20E, -L40E, or -L80E)
- Allen-Bradley PLC-5 with the PLC-5 Ethernet Interface Module (1785-ENET)
- Allen-Bradley Ethernet SLC 5/05
- Allen-Bradley ControlLogix with Ethernet Interface Module (1756-ENET, 1756-ENBT, 1756-EN2T)
- Allen-Bradley Ethernet CompactLogix (1769-L32E or 1769-L35E)
- Allen-Bradley CompactLogix with built-in Ethernet or an Ethernet Interface Module (1769-ENBT)
- Allen-Bradley FlexLogix with Ethernet Interface Module (1788-ENBT)
- Allen-Bradley SoftLogix 5
- Allen-Bradley RSLinx
- Automation Direct PAC 1000, PAC 2000, PAC 3000, DoMore, DL05, DL06, DL205, DL405 with the H*n*-ECOM100
- GE Fanuc RX3i with the ETM001 Ethernet Module
- Mitsubishi Q series with the QJ71MT91 or QJ71E71-100
- Mitsubishi Q series with built in Ethernet
- Mitsubishi iQ-R series with the QJ71MT91 or QJ71E71-100
- Modicon Quantum with the 140 NOE 711 00, 711 01, 711 10 and other Ethernet TCP/IP modules
- Modicon Momentum M1E Processor
- Omron CS/CJ PLCs
- Siemens S7-300, S7-400, and S7-1200 controllers with PROFINET support
- SoftPLC Corporation's SoftPLC

Note: If your Ethernet device is not listed above, then Delta recommends contacting one of our sales engineers to discuss the device. There may be subtle problems with using some devices with the RMC. For example, a device that proclaims Modbus/TCP support may only be a slave device, and since the RMC is also a slave, neither device will initiate transfers, preventing the devices from being able to work together. By talking with a sales engineer, Delta receives feedback on the devices that customers want to communicate with and may lead to enhancements in Delta's documentation and device support.

General Communication Details

Communicating with an RMC involves doing the following:

- **Reading and Writing Registers**
  Read and write to registers in the RMC just as you would with a PLC. Use the Address Maps in RMCTools or the Register Map help topics to find the addresses of the RMC registers.

  The RMC contains only 32-bit registers. Most are floating-point registers, but some are double integers (DINT) or double words (DWORD). If your device only supports 16-bit integers, it may still be possible to communicate with the RMC, although the communications will be limited and it will require more setup in the RMC.
- **Issuing Commands**
  Issuing commands consists of writing to the command registers. The device must be able to write to several registers at once. Issuing Commands topic for more details.

See Also

Ethernet Overview

6.7.4. Communicating Directly over TCP

Master Ethernet controllers can communicate with the RMC using several methods. This topic describes only one of those methods—using raw TCP packets—which is appropriate when the master controller cannot use any of the other methods. Please review the other available options in the Ethernet Overview topic before proceeding with this method.

This topic assumes that your controller has an established network platform and that you are familiar with how to send and receive TCP data on that platform. Some PLCs provide access to TCP connections using function blocks, while embedded controllers may require using a socket interface such as Berkeley sockets. Proper use of these interfaces is outside the scope of this documentation.

Protocol Overview

The simplest TCP-based protocol supported by the RMC is the Delta Motion Control Protocol (DMCP). This topic describes how to form DMCP packets to communicate with the RMC75E, RMC150E, and RMC200. The RMC listens for DMCP requests on TCP port 1324. The client port number can be any number. This protocol is a request/response protocol, meaning that for each request packet sent to the RMC, there will be one response packet sent by the RMC.

The DMCP protocol requires using raw TCP packets, which is not the default in some programming languages.

NOTE: Avoid repeatedly closing and re-opening TCP connections. Devices are required to maintain state on TCP connections for two minutes after the connections are closed, and repeatedly opening and closing TCP connections can exhaust resources in the RMC or host device. Instead the connection should be left open by the host device while communicating with the RMC. If your host device is not able to keep the connection open indefinitely, then consider using UDP instead, as described in Communicating Directly over UDP.

Delta has an example C code implementation of DMCP on it's forum: https://forum.deltamotion.com/

For RMC75E and RMC150E firmware versions prior to 3.31.0, Delta recommended using the Mitsubishi Procedure Exist protocol for direct communication over TCP. For details, see the Mitsubishi Procedure Exist topic.

Tip: An example implementation of DMCP in the C programming language is available in the Examples section of Delta's online forum.

Writing Data to the RMC

To send one or more registers to the RMC, send the following packet to the RMC:

Offset	Data (hex)	Description
0-1	*nn nn*	**Packet Length.** Indicates the total length in bytes of this packet, excluding this 2-byte field. This 16-bit value is encoded with the least significant byte first.
2-3	00 02	**Static Values.** These bytes must always hold these values.
4-5	*nn nn*	**Transaction ID.** This value is simply echoed by the RMC in its response packet. It can be used to match responses with requests.
6	15	**Function Code.** This byte must be 15 in a write request.

7	*nn*	**Byte Order.** Determines the byte order for all following fields. Notice that the Packet Length byte order is not affected by this field. • **Least-Significant Byte (LSB) First (00).** For example, the value 0x1122 will be encoded as 22 11. • **Most-Significant Byte (MSB) First (01).** For example, the value 0x1122 will be encoded as 11 22.
8-9	*nn nn*	**Starting Address (File).** Gives the file number (f) for the address (%MDf.e) to start the write at. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.
10-11	*nn nn*	**Starting Address (Element).** Gives the element number (e) for the address (%MDf.e) to start the write at. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.
12-13	*nn nn*	**Write Count.** The number of 32-bit registers to write. This value must be between 0 and 1024. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.
14-15	00 00	**Reserved.** Must be zero.
16-…	…	**Data.** The values of each register to write should follow the above header. Each 32-bit value is encoded in either LSB- or MSB-first byte order, as determined by the **Byte Order** field. For LSB-first byte order, the value 16#11223344 should be encoded as 44 33 22 11. For MSB-first byte order, this would be encoded as 11 22 33 44.

The RMC will respond to this request with the following packet:

Offset	Data (hex)	Description
0-1	06 00	**Packet Length.** Indicates the total length in bytes of this packet, excluding this 2-byte field. This 16-bit value is encoded with the least significant byte first.
2-3	00 02	**Static Values.** These bytes will always have these values in the response.
4-5	*nn nn*	**Transaction ID.** This value will echo the Transaction ID in the request. It can be used to match this response to the corresponding request.
6	95 *-or- 55*	**Function Code (Response).** If the write was processed successfully, this byte will be 95, otherwise it will hold 55.
7	*nn*	**Response Code.** Indicates whether the write was successful or not. See the **DMCP Response Codes** section below.

Reading Data from the RMC:

To read one or more registers from the RMC, send the following request packet to the RMC:

Offset	Data (hex)	Description
0-1	0C 00	**Packet Length.** Indicates the total length in bytes of this packet, excluding this 2-byte field. This 16-bit value is encoded with the least significant byte first.

2-3	00 02	**Static Values.** These bytes must always hold these values.
4-5	*nn nn*	**Transaction ID.** This value is simply echoed by the RMC in its response packet. It can be used to match responses with requests.
6	14	**Function Code.** This byte must be 14 in a read request.
7	*nn*	**Byte Order.** Determines the byte order for the Starting Address fields, the Read Count field, and the Data registers in the response. Notice that the Packet Length byte order is not affected by this field. • **Least-Significant Byte (LSB) First (00).** For example, the value 0x1122 will be encoded as 22 11. • **Most-Significant Byte (MSB) First (01).** For example, the value 0x1122 will be encoded as 11 22.
8-9	*nn nn*	**Starting Address (File).** Gives the file number (f) for the address (%MDf.e) to start the read at. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.
10-11	*nn nn*	**Starting Address (Element).** Gives the element number (e) for the address (%MDf.e) to start the read at. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.
12-13	*nn nn*	**Read Count.** The number of 32-bit registers to read. This value must be between 0 and 1024. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.

The RMC will respond to this request with the following packet:

Offset	Data (hex)	Description
0-1	*nn nn*	**Packet Length.** Indicates the total length in bytes of this packet, excluding this 2-byte field. This 16-bit value is encoded with the least significant byte first.
2-3	00 02	**Static Values.** These bytes will always have these values in the response.
4-5	*nn nn*	**Transaction ID.** This value will echo the Transaction ID in the request. It can be used to match this response to the corresponding request.
6	94 -*or*- 54	**Function Code (Response).** If the read was processed successfully, this byte will be 94, otherwise it will hold 54.
7	*nn*	**Response Code.** Indicates whether the read was successful or not. See the **DMCP Response Codes** section below.
8-…	…	**Data.** The values of each register that was read will follow the above header. Each 32-bit value is encoded in either LSB- or MSB-first byte order, as determined by the **Byte Order** field. For LSB-first byte order, the value 16#11223344 will be encoded as 44 33 22 11. For MSB-first byte order, this will be encoded as 11 22 33 44.

Register Addresses

This protocol uses the RMC's two-level file/element addressing format. The IEC Addressing topic describes how this addressing format works.

DMCP Response Codes

The response packet holds a single-byte Response Code field, indicating to the client whether the transaction was successful or not. Notice that the Response Function Code byte will also indicate whether the Response Code is 00 (success) or not. If the Response Code is 00, then the Response Function Code will match the request's Function Code plus 0x80. Otherwise, the Response Function Code will match the request's Function Code plus 0x40.

Response Code	Description
00	Success.
01	Malformed.
02	Too Long.
03	Invalid Address.

Notice that in certain other error cases, the RMC will not generate an error response, but will instead either discard the incoming packet with no response, or close the TCP/IP connection:

- The Packet Length field is less than 5. The RMC will discard the packet without response.
- The Packet Length field is greater than 4110. The connection will be closed.
- Unable to receive the entire packet from the TCP stream. The connection will be closed.
- Unable to successfully send the response over the TCP stream. The connection will be closed.
- The Static Values (bytes 2-3) are not 00 and 02 respectively. The RMC will discard the packet without response.
- Unsupported function code (byte 6). The RMC will discard the packet without response.

The errors above are also logged in the Event Log.

Example 1: Writing a Single Register

In this example, the client will write the value 0x11223344 to variable 0 (%MD56.0), using the LSB-first byte order (00). The client chooses a Transaction ID of 00 00 for this packet, either because he is not using this field, or because this is the first transaction.

Plugging in the above values in the proper byte order gives us the following packet that should be sent to TCP port 1324 on the RMC:

```
12 00 00 02 00 00 15 00
38 00 00 00 01 00 00 00
44 33 22 11
```

After the RMC has successfully received and processed this write request, it will respond with the following packet:

```
06 00 00 02 00 00 95 00
```

Notice that the last byte is the Response Code, with 00 meaning success.

Example 2: Reading a Single Register

In this example, the client will read a value from variable 0 (%MD56.0). The client chooses a Transaction ID of 0x0001, perhaps because this is the second transaction.

Plugging in the above values in the proper byte order gives us the following packet that should be sent to TCP port 1324 on the RMC:

```
0C 00 00 02 01 00 14 00
38 00 00 00 01 00
```

After the RMC has successfully received and processed this write request, it will respond with the following packet if variable 0 held the value 0x11223344:

0A 00 00 02 01 00 94 00
44 33 22 11

Notice that byte 7 holds the Response Code, and bytes 8-11 hold the data that was read.

See Also

IEC Addressing | Communicating Directly over UDP | RMCLink Component | Delta Motion Control Protocol

6.7.5. Communicating Directly over UDP

Master Ethernet controllers can communicate with the RMC using several methods. This topic describes only one of those methods—using raw UDP packets—which is appropriate when the master controller cannot use any of the other methods. Please review the other available options in the Ethernet Overview topic before proceeding with this method.

This topic assumes that your controller has an established network platform and that you are familiar with how to send and receive UDP packets on that platform. Some PLCs provide access to UDP packets using function blocks, while embedded controllers may require using a socket interface such as Berkeley sockets. Proper use of these interfaces is outside the scope of this documentation.

Protocol Overview

The simplest UDP-based protocol supported by the RMC is the Delta Motion Control Protocol (DMCP). This topic describes how to form DMCP packets to communicate with the RMC75E, RMC150E, and RMC200. The RMC listens for DMCP requests on UDP port 1324. The client port number can be any number. This protocol is a request/response protocol, meaning that for each request packet sent to the RMC, there will be one response packet sent by the RMC.

Delta has an example C code implementation of DMCP on it's forum: https://forum.deltamotion.com/

For RMC75E and RMC150E firmware versions prior to 3.31.0, Delta recommended using the FINS/UDP protocol for direct communication over UDP. For details, see the FINS/UDP topic.

Writing Data to the RMC

To send one or more registers to the RMC, send the following packet to the RMC:

Offset	Data (hex)	Description
0-1	*nn nn*	**Packet Length.** Indicates the total length in bytes of this packet, excluding this 2-byte field. This 16-bit value is encoded with the least significant byte first.
2-3	00 02	**Static Values.** These bytes must always hold these values.
4-5	*nn nn*	**Transaction ID.** This value is simply echoed by the RMC in its response packet. It can be used to match responses with requests, since UDP does not prevent packets from being delivered out-of-order or dropped.
6	15	**Function Code.** This byte must be 15 in a write request.
7	*nn*	**Byte Order.** Determines the byte order for all following fields. Notice that the Packet Length byte order is not affected by this field. • **Least-Significant Byte (LSB) First (00).** For example, the value 0x1122 will be encoded as 22 11.

		• **Most-Significant Byte (MSB) First (01).** For example, the value 0x1122 will be encoded as 11 22.
8-9	*nn nn*	**Starting Address (File).** Gives the file number (f) for the address (%MDf.e) to start the write at. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.
10-11	*nn nn*	**Starting Address (Element).** Gives the element number (e) for the address (%MDf.e) to start the write at. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.
12-13	*nn nn*	**Write Count.** The number of 32-bit registers to write. This value must be between 0 and 256. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.
14-15	00 00	**Reserved.** Must be zero.
16-…	…	**Data.** The values of each register to write should follow the above header. Each 32-bit value is encoded in either LSB- or MSB-first byte order, as determined by the **Byte Order** field. For LSB-first byte order, the value 16#11223344 should be encoded as 44 33 22 11. For MSB-first byte order, this would be encoded as 11 22 33 44.

The RMC will respond to this request with the following packet:

Offset	Data (hex)	Description
0-1	06 00	**Packet Length.** Indicates the total length in bytes of this packet, excluding this 2-byte field. This 16-bit value is encoded with the least significant byte first.
2-3	00 02	**Static Values.** These bytes will always have these values in the response.
4-5	*nn nn*	**Transaction ID.** This value will echo the Transaction ID in the request. It should be used to match this response to the corresponding request.
6	95 *-or- 55*	**Function Code (Response).** If the write was processed successfully, this byte will be 95, otherwise it will hold 55.
7	*nn*	**Response Code.** Indicates whether the write was successful or not. See the **DMCP Response Codes** section below.

Reading Data from the RMC:

To read one or more registers from the RMC, send the following request packet to the RMC:

Offset	Data (hex)	Description
0-1	0C 00	**Packet Length.** Indicates the total length in bytes of this packet, excluding this 2-byte field. This 16-bit value is encoded with the least significant byte first.
2-3	00 02	**Static Values.** These bytes must always hold these values.
4-5	*nn nn*	**Transaction ID.** This value is simply echoed by the RMC in its response packet. It can be used to match responses

		with requests, since UDP does not prevent packets from being delivered out-of-order or dropped.
6	14	**Function Code.** This byte must be 14 in a read request.
7	*nn*	**Byte Order.** Determines the byte order for the Starting Address fields, the Read Count field, and the Data registers in the response. Notice that the Packet Length byte order is not affected by this field. • **Least-Significant Byte (LSB) First (00).** For example, the value 0x1122 will be encoded as 22 11. • **Most-Significant Byte (MSB) First (01).** For example, the value 0x1122 will be encoded as 11 22.
8-9	*nn nn*	**Starting Address (File).** Gives the file number (f) for the address (%MDf.e) to start the read at. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.
10-11	*nn nn*	**Starting Address (Element).** Gives the element number (e) for the address (%MDf.e) to start the read at. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.
12-13	*nn nn*	**Read Count.** The number of 32-bit registers to read. This value must be between 0 and 256. The order of the bytes in this 16-bit value is determined by the **Byte Order** field.

The RMC will respond to this request with the following packet:

Offset	Data (hex)	Description
0-1	*nn nn*	**Packet Length.** Indicates the total length in bytes of this packet, excluding this 2-byte field. This 16-bit value is encoded with the least significant byte first.
2-3	00 02	**Static Values.** These bytes will always have these values in the response.
4-5	*nn nn*	**Transaction ID.** This value will echo the Transaction ID in the request. It should be used to match this response to the corresponding request.
6	94 -*or*- 54	**Function Code (Response).** If the read was processed successfully, this byte will be 94, otherwise it will hold 54.
7	*nn*	**Response Code.** Indicates whether the read was successful or not. See the **DMCP Response Codes** section below.
8-…	…	**Data.** The values of each register that was read will follow the above header. Each 32-bit value is encoded in either LSB- or MSB-first byte order, as determined by the **Byte Order** field. For LSB-first byte order, the value 16#11223344 will be encoded as 44 33 22 11. For MSB-first byte order, this will be encoded as 11 22 33 44.

Register Addresses

This protocol uses the RMC's two-level file/element addressing format. The IEC Addressing topic describes how this addressing format works.

DMCP Response Codes

The response packet holds a single-byte Response Code field, indicating to the client whether the transaction was successful or not. Notice that the Response Function Code byte will also indicate whether the Response Code is 00 (success) or not. If the Response Code is 00, then the Response Function Code will match the request's Function Code plus 0x80. Otherwise, the Response Function Code will match the request's Function Code plus 0x40.

Response Code	Description
00	Success.
01	Malformed.
02	Too Long.
03	Invalid Address.

Notice that in certain other error cases, the RMC will not generate a response to the DMCP request. Such cases include the following:

- UDP datagram is less than 7 bytes in length.
- UDP datagram size does not match the Packet Length field. The UDP datagram length must always be 2 bytes longer than the Packet Length field.
- The Static Values (bytes 2-3) are not 00 and 02 respectively.
- Unsupported Function Code.

Example 1: Writing a Single Register

In this example, the client will write the value 0x11223344 to variable 0 (%MD56.0), using the LSB-first byte order (00). The client chooses a Transaction ID of 00 00 for this packet, either because he is not using this field, or because this is the first transaction.

Plugging in the above values in the proper byte order gives us the following packet that should be sent to UDP port 1324 on the RMC:

12 00 00 02 00 00 15 00
38 00 00 00 01 00 00 00
44 33 22 11

After the RMC has successfully received and processed this write request, it will respond with the following packet:

06 00 00 02 00 00 95 00

Notice that the last byte is the Response Code, with 00 meaning success.

Example 2: Reading a Single Register

In this example, the client will read a value from variable 0 (%MD56.0). The client chooses a Transaction ID of 0x0001, perhaps because this is the second transaction.

Plugging in the above values in the proper byte order gives us the following packet that should be sent to UDP port 1324 on the RMC:

0C 00 00 02 01 00 14 00
38 00 00 00 01 00

After the RMC has successfully received and processed this write request, it will respond with the following packet if variable 0 held the value 0x11223344:

0A 00 00 02 01 00 94 00
44 33 22 11

Notice that byte 7 holds the Response Code, and bytes 8-11 hold the data that was read.

See Also

IEC Addressing | Communicating Directly over TCP | RMCLink Component | Delta Motion Control Protocol

6.7.6. Ethernet Link/Act LED

The RMC75E, RMC150E, and RMC200 have one Link/Activity LED per Ethernet port. The RMC200 has 2 ports. This LED has the following states:

RMC75/150 State	Description
Off	Link is down
Flashing Green	Link is up, with activity
Steady Green	Link is up, no activity

RMC200 State	Description
Off	**Link down** The Ethernet link is down.
Flashing Green	**Link up, activity** The Ethernet link is up, and activity is detected.
Steady Green	**Link up, idle** The Ethernet link is up, but no activity is detected.
Flashing Amber	**Link up, collision** The Ethernet link is up, but there was a collision.
Steady Amber	**Port disabled** The user has disabled this port.
Steady Red	**Major port fault** There was a major fault initializing the port.

The Link/Activity LED reflects the status of the physical Ethernet connection between the RMC and the device on the other end of the Ethernet cable - typically a switch. This LED will come on if (1) both devices are powered up, (2) both devices have enabled the Ethernet port, (3) the cable is wired correctly and inserted securely on both ends, and (4) both devices agreed on Ethernet settings (baud rate, and full/half duplex). Notice that the "Link" portion has no reflection on higher level protocols.

When the LED is blinking, this means that there is activity on this Ethernet connection, whether intended for this device or not. It will blink for packets that the switch sends to the RMC, which may be unicast packets intended for the RMC, multicast packets if the switch doesn't filter them, or broadcast packets. When an Ethernet hub is used instead of a switch, then all traffic intended for any device connected to the hub will affect the LED blinking.

The _Enet.Status.LinkUp tag will be true when the Link/Activity LED is flashing or steady green.

If the Link/Act LED is off, check the following items (the remote device is assumed to be a switch, but it could be a PLC, PC, or HMI):

1. Verify that the RMC is powered up.
2. Verify that the switch is powered up.
3. Verify that the Ethernet cable is properly inserted into the RMC.
4. Verify that the other end of the Ethernet cable is properly inserted into the switch.
5. Verify that the Ethernet cable is a properly constructed Category 5 Ethernet cable. Replace the cable if necessary.

6. Verify the duplex and baud rate settings in the RMC and switch. Either one or both must be set to Auto Negotiate, or both must be set to the same baud rate and duplex settings.

If the Link/Act LED is on, but not blinking when you expect activity, check the following:

1. Verify that the RMC has an IP address. If the RMC has no IP address, the Net LED will be off. If a duplicate IP address has been detected, the Net LED will be steady red.
2. Verify that the RMC's IP address is what you expect. Use the Communication Statistics window to verify the RMC's IP settings.
3. Verify that the remote device (PC, PLC, or HMI) has the IP address that you expect.
4. Verify that the subnet mask is set properly in both the RMC and remote device. See the Understanding IP Addressing for details.
5. If both devices are on the same Ethernet network, then verify that the subnet masks in the RMC and remote device are the same, and that the network portions of the IP addresses match. See the Understanding IP Addressing for details.
6. Verify that the IP address entered for the RMC in the remote device matches the RMC's actual IP address.
7. Verify that the remote device is properly enabled.

See Also

Ethernet Overview | Understanding IP Addressing

6.7.7. Troubleshooting RMCTools Ethernet Connection

This topic describes problems and possible solutions for connecting RMCTools via Ethernet to the RMC75E, RMC150E, or RMC200.

1. My RMC does not appear in the Browse box in the Ethernet Configuration Dialog.

Possible Reason:

The Windows Firewall may be enabled. Also there are other firewall software packages that may be installed on your PC. If a firewall is installed and active on your computer and has not been told to allow RMCTools through, then browsing for controllers over Ethernet will not work. There are two solutions.

Solution A: Allow RMCTools through the Firewall

In many cases, you will only need to tell Windows Firewall to not block RMCTools. This can be done when Windows Firewall notifies you that it is blocking RMCTools, or can be done through the Windows Firewall control panel. Some users may need their network administrator to add this exception to their firewall.

Solution B: Turn off the Firewall

In some cases, it has been found that adding an exception to Windows Firewall or other firewall software does not allow browsing to complete successfully. This is especially true when trying to browse for RMC75E controllers with firmware older than 1.52. In this situation it may be necessary to temporarily disable the firewall. Consult your network administrator before turning off the firewall.

To turn off Windows Firewall in Windows 8.1, 10, or 11:

1. Click **Start**.

2. In the search window, type **Windows Defender Firewall**, and then click it. This will open the **Windows Defender Firewall** control panel.
3. In the menu on the left, click **Turn Windows Defender Firewall on or off**.
4. For each of the network types, click **Turn off Windows Firewall (not recommended)**.
5. Click **OK**.

To turn off Windows Firewall in Windows Vista, Server 2008, and Windows 7:

1. Click **Start**.
2. In the search window, type **firewall.cpl**, and then click **OK**. This will open the **Windows Firewall** control panel.
3. In the menu on the left, click **Turn Windows Firewall on or off**.
4. For each of the network types, click **Turn off Windows Firewall (not recommended)**.
5. Click **OK**.
6. Restart RMCTools.

To turn off Windows Firewall in Windows XP:

1. Click **Start**, and then click **Run**.
2. Type **firewall.cpl**, and then click **OK**. This will open the **Windows Firewall** control panel.
3. On the **General** tab, click **Off (not recommended)**.
4. Click **OK**.
5. Restart RMCTools.

Possible Reason:

Computers with multiple IP interfaces may experience problems browsing for controllers over Ethernet. Examples of IP interfaces include each wired or wireless Ethernet adaptor plus dial-up networking adaptors. Notice that virtualization software such as VMware and Virtual PC also often adds virtual IP interfaces, in addition to any physical interfaces on the computer.

To determine if browsing may not be working due to multiple IP interfaces:

1. **Determine if your computer has multiple IP interfaces**
   From the browsing window, click **Troubleshoot**. This will open the **Troubleshoot Ethernet Browsing** dialog box, which includes a table of all available IP interfaces on your computer. If more than one interface is listed, then proceed to the next step.
2. **Determine which of the interfaces listed is connected to the network containing the RMCs**
   First, determine which physical connector on your computer is connected to the network with the RMC controllers. Next, determine which interface in the list corresponds with that connector. You can rule out any virtual adapters, such as those labeled ”VMware Virtual Ethernet Adapter.” If there is more than one remaining interface, then you may be able to look at the IP address to help decide which interface corresponds to your main wired Ethernet port.
3. **Verify that the interface with access to the RMCs has an asterisk (*) next to its Metric value**
   If the desired interface does not have an asterisk (*) next to its Metric value, then browsing for controllers on that interface will likely not work. See the solutions below.
4. **Verify that the interface with access to the RMCs has a lower Metric value than all other interfaces**
   If the desired interface has the asterisk (*), but its Metric value is not the lowest or is tied for lowest with another interface, then you may experience intermittent problems with browsing for controllers on that interface. It is recommended that you consider one of the solutions below.

Solution A: Adjust Interface Metrics
The least obtrusive method of fixing this problem is to adjust the Metric values for each interface such that the desired interface has the lowest value. Usually, this only requires settings the Metric for the desired interface to 1. However, if other interfaces are already have a Metric value of 1, then you will also have to increase those interfaces' Metric values to 2.

Notice that the Windows operating system controls the interface Metric values, and therefore the **Network Connections** control panel must be used to change these values. After making the changes in the control panel, refer to the **Troubleshoot Ethernet Browsing** window in RMCTools to see if the changes have taken effect. If they have not (particularly if the values changed but the asterisk (*) did not move to the interface with the lowest Metric), then you must restart your computer.

The following procedures describe how to change an interface metric using the Network Connections control panel:

To change an interface metric on Windows Vista, Server 2008, and Windows 7:

1. Click **Start**.
2. In the search window, type **ncpa.cpl**, and press Enter. This will open the **Network Connections** control panel.
3. Right-click the network adapter whose metric you want to change, click **Properties**, and then the click the **Networking** tab.
4. In the **This connection uses the following items** list, click **Internet Protocol Version 4 (TCP/IPv4)**, and then click **Properties**.
5. In the **Internet Protocol Version 4 (TCP/IPv4) Properties** dialog box, click the **General** tab, and then click **Advanced**.
6. In the **Advanced TCP/IP Settings** dialog box, click the **IP Settings** tab, clear the **Automatic metric** checkbox, and then type a value in the **Interface metric** checkbox.
7. Click **OK** in the **Advanced TCP/IP Settings** dialog box.
8. Click **OK** in the **Internet Protocol Version 4 (TCP/IPv4) Properties** dialog box.
9. Click **Close** in the **Connection Properties** dialog box.

To change an interface metric on Windows XP and Server 2003:

1. Click **Start**, and then click **Run**.
2. Type **ncpa.cpl**, and then click **OK**. This will open the **Network Connections** control panel.
3. Right-click the network adapter whose metric you want to change, click **Properties**, and then the click the **General** tab.
4. In the **This connection uses the following items** list, click **Internet Protocol (TCP/IP)**, and then click **Properties**.
5. In the **Internet Protocol (TCP/IP) Properties** dialog box, click the **General** tab, and then click **Advanced**.
6. In the **Advanced TCP/IP Settings** dialog box, click the **IP Settings** tab, clear the **Automatic metric** checkbox, and then type a value in the **Interface metric** checkbox.
7. Click **OK** in the **Advanced TCP/IP Settings** dialog box.
8. Click **OK** in the **Internet Protocol (TCP/IP) Properties** dialog box.
9. Click **OK** in the **Connection Properties** dialog box.

Solution B: Disable Unused Interfaces
If you are not using some of the IP interfaces, you may want to disable them. Disabling interfaces will remove them from the list, thus allowing the desired interface to have the

lowest metric and the asterisk (*). Notice that disabling an interface may affect other applications using that interface, and should only be done if you know that the interface does not need to be used for the time that it is disabled.

To disable an interface:

1. Click **Start**, and then click **Run**.
2. Type **ncpa.cpl**, and then click **OK**. This will open the **Network Connections** control panel.
3. Right-click the network adapter that you want to disable, and click **Disable**.

2. I can't connect to my RMC via the Internet or a VPN.

Solution:

If your PC has a firewall, make sure it allows connections to port 44818. If the RMC is behind a firewall, make sure the firewall forwards port 44818 to the RMC's IP address.

3. The Link LED is off or is not blinking when you expect activity.

Solution:

See the Ethernet Link topic for troubleshooting procedures for the Link LED.

See Also

Monitor Port | Setting Up a Standalone TCP/IP Network | Using Ethernet with RMCTools | Ethernet Overview

6.7.8. Ethernet Setup Topics

6.7.8.1. Setting Up the RMC Ethernet

Setting up the Ethernet communications for the RMC75E, RMC150E, or RMC200 requires entering only a few IP parameters.

Tip: The RMC200 Ethernet address can also be set up from the Display Screen.

1. Go Online with the RMC.
2. In the Project Pane, expand the **Modules** folder, double-click the CPU module and click **Ethernet**.
3. In the **IP Settings** section, click **Use the following IP address**.
4. Enter an **IP address** and **subnet mask** in dotted decimal notation (e.g. 192.168.0.5 and 255.255.255.0). To learn about IP addressing or setting up a stand-alone network, refer to Understanding IP Addressing and Setting Up a Standalone TCP/IP Network.
5. The **Default Gateway** is optional. If you choose not to use it, leave it blank, and the RMC will not be able to communicate with devices on networks other than its own. Otherwise, enter a value in dotted decimal notation (e.g. 192.168.0.1).

In the **Additional Options** section, checking the **Lock IP settings** box will not allow the IP settings to be changed through an Ethernet connection. Changing the IP settings will only be allowed when RMCTools is connected to the RMC via USB. Locking the IP settings prevents accidentally changing the IP settings from RMCTools when connected over Ethernet. If they are accidentally changed, then the connection will be broken, which can be very frustrating if programming the RMC remotely.

See Also

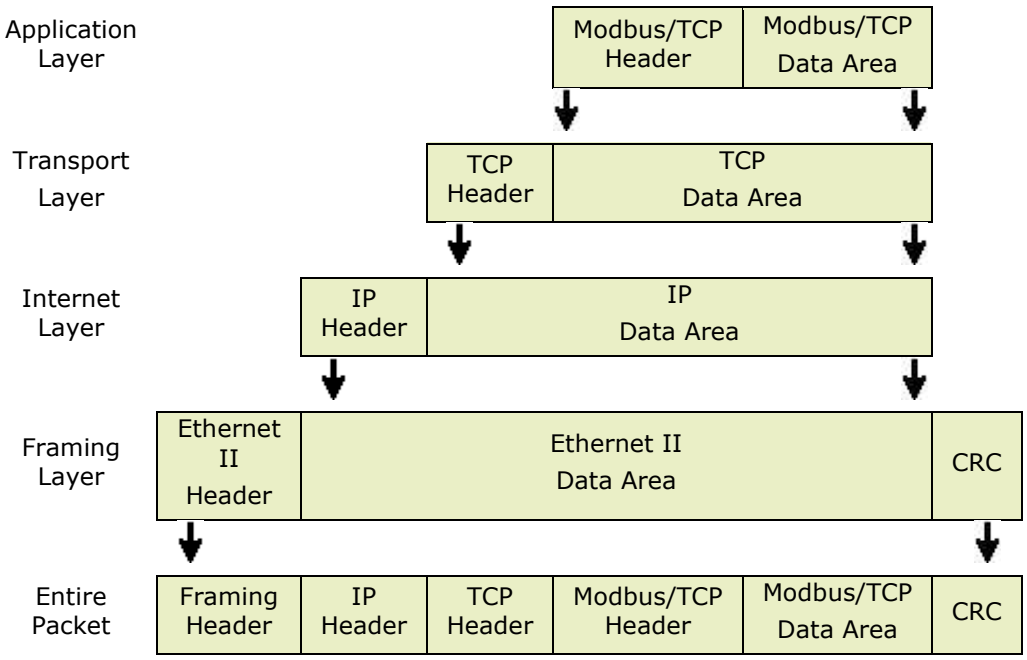
Understanding IP Addressing | Setting Up a Standalone TCP/IP Network | Ethernet Overview

6.7.8.2. RMC Ethernet Protocols

This topic provides a complete list of low-level protocols supported on the RMC75E, RMC150E, and RMC200 for use by network administrators. This topic is not required to be understood for using the RMC module successfully. It is intended to be a brief summary of the architecture and a list of the protocols.

TCP/IP Layers Overview

Networking is often viewed conceptually as layers of protocols. Each layer contains a header, used to fulfill the purpose of that layer, and data. These layers are set up such that each layer contains the header and data from the next higher layer within its data area, as shown in the following diagram:



This diagram shows the four conceptual layers of TCP/IP: application, transport, internet, and framing. A fifth layer—the hardware layer—is often added below these four layers, but is left out of this diagram because it is more of a specification of how the data is physically sent rather than another protocol header. When a device is sending a packet the packet is assembled from the top layer down, but when receiving a packet, it must be processed from the bottom layer up.

Here is how the RMC might look at an incoming packet with this structure:

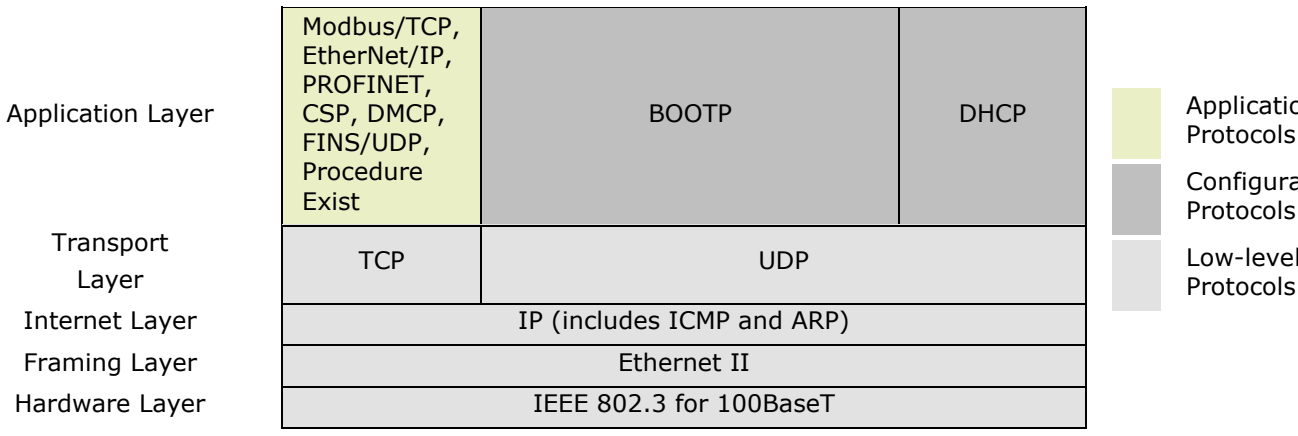
1. **Hardware Layer:**
   A full packet is received and passed to the Framing Layer.
2. **Framing Layer:**
   The CRC (cyclic redundancy check) is verified. If this fails, the packet is discarded. Next, the destination MAC address in the framing header is compared with the RMC's MAC address. If

the addresses do not match and the destination address was not a special broadcast address, the packet is discarded. Otherwise it is passed to the Internet Layer.

3. **Internet Layer:**
   The IP address in the IP header is compared with the RMC's user-selectable IP address. If it does not match, the packet is discarded. Otherwise, it is passed to the Transport Layer.
4. **Transport Layer:**
   The transport layer provides a number of services, but minimally must specify the port that the data should be sent to. A port is an abstract connection point on a device that allows for multiple connections to exist on a single device. It also helps determine which application protocol will follow. The packet may be discarded here too if the destination port is not one that the RMC supports.
5. **Application Layer:**
   In our example, the application protocol is Modbus/TCP, so the Modbus/TCP header contains data such as the RMC register address to begin reading or writing from, the number of registers to access, and whether the operation is a read or write. The Modbus/TCP data area holds the actual words to be written.

Supported Protocols

The diagram below shows many of the protocols supported by the RMC and the layers to which they belong:



Each protocol supported by RMC controllers is briefly described below:

ARP (Address Resolution Protocol)

Ethernet packets can either be broadcast (received by all devices on the network) or sent to a single MAC address. However, applications generally address computers by IP address rather than MAC address. Therefore, this protocol is used to determine the MAC address of the computer owning a given IP address.

BOOTP (BOOTstrap Protocol)

This protocol is used to allow a central database to be maintained with all IP addresses on a network. This single computer is the BOOTP server. When a BOOTP client (such as the RMC, if configured to use BOOTP) starts up, it broadcasts asking a BOOTP server to tell it what its IP address should be. The BOOTP server looks up the MAC address of the BOOTP client in a database and sends a reply with the corresponding IP address.

When the RMC powers up, it broadcasts a request for its IP parameters. If a BOOTP server is available on the network, then the BOOTP server will look into a database of mappings from MAC addresses to IP addresses for the MAC address of the RMC that sent the request. If it finds a match, it will give the RMC an IP address, default gateway, and subnet mask. If no BOOTP server responds—either because no BOOTP server exists, or the RMC's MAC address was not

found in the BOOTP server's database—the RMC's Ethernet communication channel will not be usable.

CIP (Control Information Protocol)

See the **EtherNet/IP** section below.

CSP (Client/Server Protocol)

This is a proprietary protocol developed by Allen-Bradley, Inc. Variants of this are used on Allen-Bradley's SLC 5/05 and PLC-5 controllers. This protocol is also used by the Allen-Bradley's SoftLogix 5 and RSLinx and SoftPLC Corporation's SoftPLC. Allen-Bradley does not publish the specifications for this protocol. See the CSP topic for details on usage with the RMC.

DHCP (Dynamic Host Configuration Protocol)

This protocol is an enhanced version of BOOTP. However, for industrial applications, the enhancements (lease times and dynamic assignment of IP addresses) are generally not usable. The RMCs support both BOOTP and DHCP so that the user may use either type of server.

DLR (Device Level Ring) - RMC200 only

Device Level Ring (DLR) provides media redundancy in a ring topology for devices that have multiple Ethernet ports. The DLR protocol provides for fast network fault detection and reconfiguration. The RMC200 uses DLR with EtherNet/IP.

DMCP (Delta Motion Control Protocol)

A simple Ethernet protocol intended for use with direct communication to the RMC over TCP or UDP. In applications where none of the RMC's other protocols are supported by the master controller, but direct communication over TCP or UDP is allowed, Delta recommends that the DMCP protocol be implemented by the user manually.

Ethernet II

This is the most common framing layer protocol used by Ethernet devices. Other alternatives include IEEE 802.3 SNAP, IEEE 802.5 (Token Ring), and IEEE 802.4 (Token Bus). The RMC only supports Ethernet II framing, and therefore will not work on networks using any of the other framing types. All PLCs currently support Ethernet II framing although the Modicon Quantum allows selecting either Ethernet II (the default) or IEEE 802.3 SNAP.

EtherNet/IP

This is an open application protocol, originally developed by Allen-Bradley, Inc., but now maintained and distributed by ODVA (https://www.odva.org). The specification and source code are available through https://www.ethernet-ip.org. This protocol is an Ethernet-adaptation of the Control Information Protocol (CIP) in the same way that DeviceNet is a CAN adaptation of CIP and ControlNet is a CTDMA-adaptation of CIP. EtherNet/IP is used by the Allen Bradley ControlLogix's Ethernet modules (1756-ENET and 1756-ENBT).
See the EtherNet/IP topic for details on usage with the RMC.

FINS/UDP

This is an open application protocol developed and used by Omron Electronics Inc. This protocol is available over a number of media, including Ethernet and serial. Additional information is available in the CS1 Communications Reference Manual, available on Omron's web site: https://www.omron.com/oei.

For more details, see the FINS/UDP and Communicating Directly over UDP topics.

HEI (Host Engineering Inc)

This is a proprietary protocol controlled by Host Engineering Inc (https://www.hosteng.com). This application protocol can be run on top of UDP/IP, IPX, and Ethernet II. The RMC can respond to HEI requests over UDP/IP and Ethernet II.

ICMP (Internet Control Message Protocol)

This protocol—running on top of the Internet Protocol—is used for sending error messages between routers and also provides the ping service. Only the ping service portion of this protocol

is used by the RMC. Ping is a utility provided on most operating systems that simply asks if a device can be found with a given IP address or name.

IEEE 802.3 for 100BaseT

This is a standard for sending Ethernet data at 100Mb/s through twisted pair wiring rated Category 5 (CAT5) or greater. The connectors used by this standard are RJ45.

IP (Internet Protocol)

This is the main Internet Layer protocol and is used for sending packets between two computers in a network or across two or more networks. It also allows for fragmentation of packets. This is a situation where a packet is larger than a network's maximum packet size, so it is broken into smaller packets that are re-assembled by the recipient.

LLDP (Link Layer Discovery Protocol)

This is an open link-layer protocol used by network devices to advertise their identity, capabilities, and neighbors over Ethernet. All PROFINET devices are required to support LLDP. These devices also are required to use a separate port-specific port MAC address (instead of the device MAC address used by all other protocols) as the source address for LLDP packets that are sent. See Ethernet MAC Address for details.

Modbus/TCP

This is an open protocol developed and used by Modicon of Schneider Electric. Its standard is maintained by Mosbus-IDA (www.modbus.org). See the Modbus/TCP topic for details on usage with the RMC.

MRP (Media Redundancy Protocol) - RMC200 Only

Media Redundancy Protocol (MRP) is a data network protocol standardized as IEC 62439-2. It allows rings of Ethernet switches to overcome any single failure with a fast recovery time. The RMC200 uses MRP with PROFINET.

Procedure Exist

The Mitsubishi protocol is for the Mitsubishi's Q-series QJ71E71-100 Ethernet module and the FX3U PLC. See the Mitsubishi Procedure Exist topic for details.

PROFINET

PROFINET is the open industrial Ethernet standard of PROFIBUS & PROFINET International (PI) for automation and is part of the IEC 61158 and IEC 61784 standards. The RMC75E, RMC150E, and RC200 support PROFINET RT, which allows for data cycles down in the millisecond range. See the PROFINET topic for details on usage with an RMC.

SNMP (Simple Network Management Protocol)

This is an open protocol running on top of UDP for collecting and organizing information about devices on IP networks. It is widely used in network management for network monitoring. Support for SNMP is required by PROFINET devices wishing to comply with the Topology Engineering & Discovery feature, which is especially important for devices with multiple Ethernet ports like the RMC200. See Simple Network Management Protocol (SNMP) for details on support by RMC controllers.

TCP (Transmission Control Protocol)

This is the main Transport Layer protocol in the TCP/IP stack. It provides for maintaining a reliable connection between two devices.

UDP (User Datagram Protocol)

This Transport Layer protocol is used by some application protocols instead of TCP. UDP is lighter weight than TCP and works well in situations where acknowledgements are built into the application protocol or are not required.

If you wish to communicate with the RMC directly over UDP, see the Communicating Directly over UDP topic.

See Also

Ethernet Overview

6.7.9. Ethernet Informational Topics

6.7.9.1. Setting Up a Standalone TCP/IP Control Network

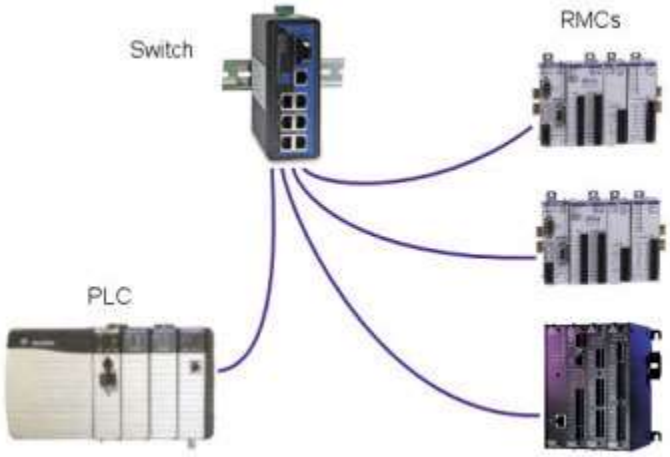
Many industrial applications require a standalone Ethernet network for machine control, for example, a PLC communicating with several RMCs and an HMI. This topic describes how to easily set up a standalone Ethernet network using TCP/IP.

Before reading this topic, make sure you understand the Understanding IP Addressing topic.

Note:
This topic is intended only for new networks that will not be connected via a router to another network. If you will be adding an RMC to an existing network or you will be creating a new network that will be connected via a router to another network, consult your network administrator.

To set up a stand-alone network:

1. **Wire the Network**
   The RMC75E and RMC150E use the IEEE 802.3 100BaseT hardware standard. This means it runs at 100Mbaud on twisted pair wiring rated Category 5 or higher, and uses RJ45 connectors. Twisted pair networks generally use a star topology, which means that each device is wired to a single switch device, as illustrated below:



   Care should be taken to use a high-quality switch that will support the temperature, noise, vibration, and other environmental requirements of the application. It is also important to use a switch rather than a hub to avoid collisions, which reduce the determinism of the network. Both switches and Category 5 (commonly called CAT5) cabling are readily available from network supply companies.

2. **Select a network address and subnet mask.**
   By convention, the address ranges below are intended to be used for private networks. An address from these ranges is a good choice for the network address of a stand-alone control network.
   - 10.0.0.0 - 10.255.255.255
   - 172.16.0.0-172.31.255.255
   - 192.168.0.0 - 192.168.255.255

   Example:
   The user decided to use the 192.168.0 address. Because this network address is 24 bits long, the subnet mask will be 255.255.255.0. This leaves 254 local addresses (remember that addresses 0 and 255 are reserved) for an IP address range of 192.168.0.1 to 192.168.0.254.

3. **Assign local addresses for each device.**
   The IP address range from step 2 provides 254 IP addresses that can be assigned to the network devices. When assigning addresses, ensure that all devices have unique address. To avoid assigning the same IP address twice, record the IP address assignments for use later when you need to add or replace device.
   By convention, local address 1 (IP address 192.168.0.1 in our new example network in the previous step) is used as the default gateway. Even if you do not have gateway/router, it is a good idea to leave that address unassigned. Assign your first device (perhaps a PLC) 192.168.0.2, assign your second device (perhaps an RMC) 192.168.0.3, etc.

4. **Enter the network parameters into each device.**
   The method of assigning the network parameters varies for each type of device. Use the IP address you have assigned, a subnet mask of 255.255.255.0, and leave the default gateway blank for each device. See RMC Ethernet Setup for details on editing these parameters on the RMC, and consult the manuals of the other devices on your network for details on setting up their TCP/IP parameters.

See Also
Understanding IP Addressing | Ethernet Overview

6.7.9.2. Understanding Ethernet IP Addressing

IP Address

A fundamental part of setting up a TCP/IP network is setting up IP addresses. An IP address is a 32-bit number that is generally displayed in dotted decimal format, in which each octet (8 bits) of the address is displayed in decimal format, and each value is separated by period (e.g. 192.168.0.5). A less common, but often useful, way of displaying the address is in hexadecimal. The hexadecimal equivalent of 192.168.0.5 is C0A80005. Every computer on an intranet (one or more networks connected together) must have a unique IP address.

Subnet Mask

To facilitate communicating between multiple interconnected networks, the IP address is broken into two parts. One part is the network address, and the other part is the local address. Each network has a unique network address, and every device on that network has the same network address portion in its IP address. The local address uniquely identifies a computer within a

network. It is expected that local addresses will be duplicated on different networks, but the entire IP address (network address + local address) is always unique.

The method for determining which portion of the IP address is the network address and which portion is the local address is to use a value called a subnet mask. A subnet mask is also a 32-bit number often displayed in dotted decimal format. Each bit of the subnet mask that is a 1 means that the corresponding bit of the IP address is part of the network address. Each bit of the subnet mask that is a 0 means that the corresponding bit of the IP address is part of the local address.

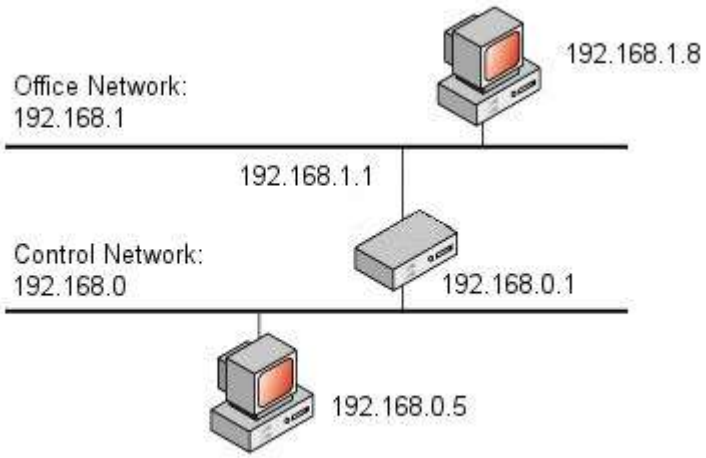
Example:

	Decimal Value	Hexadecimal Value
IP Address	192.168.0.5	C0A80005
Subnet Mask	255.255.255.0	FFFFFF00
Network Address	192.168.0	C0A800
Local Address	5	05

Therefore, from this example, we see that a device with an IP address of 192.168.0.5 and subnet mask of 255.255.255.0 will have a network address of 192.168.0 and a local address of 5. Other devices on this network must have the same network address but different local addresses. Therefore, some possible IP addresses for other nodes on the network include 192.168.0.6, 192.168.0.1, and 192.168.0.25. There are two reserved local addresses: a local address with all zero bits refers to the network (e.g. 192.168.0.0), and a local address with all one bits is the broadcast address for the network (e.g. 192.168.0.255).

Default Gateway

Suppose the device given in the above example must communicate with a device on a connected network with an IP address of 192.168.1.8. Because the device is not on the same network there is no electrical connection between the computers so it cannot send its data directly. Instead it must go through an IP router. An IP router is a device that sends packets it receives from one network that are intended for devices on another network to the other network. Here is the example intranet:



How does 192.168.0.5 send a message to 192.168.1.8? The answer is that it must use a third parameter called the default gateway. This parameter is the IP address of the router who will take care of getting the packet to its destination. The rule for most devices is to send packets to devices with the same network address directly over its network, but to send packets to devices with a different network address to the default gateway. In the above network, the device at

192.168.0.5 would have a default gateway of 192.168.0.1, and the device at 192.168.1.8 would have a default gateway of 192.168.1.1.

The default gateway parameter is optional if the device will be on a network that is not connected to any other networks, or if you have an intranet but do not want to allow the device to communicate with devices on networks other than its own.

See Also

Ethernet Overview

6.7.9.3. Ethernet MAC Address

Every Ethernet device manufactured is required to have a unique MAC address. This address is given in the form of twelve hexadecimal digits (e.g. 0050A0984001) and is often displayed with each digit pair separated by spaces, hyphens or colons (e.g. 00-50-A0-98-40-01). However, all formats are equivalent. Network administrators may require this value when tracking network problems or configuring BOOTP or DHCP protocols. All Ethernet devices manufactured by Delta Computer Systems, Inc. begin with "00 50 A0".

The RMC75, RMC150, and RMC200 Ethernet controllers all use more than one MAC address. The first MAC address is called the device MAC address and is the primary MAC address used for almost all Ethernet communication. In addition to the device MAC address, each physical Ethernet port will also have its own port MAC address. The port MAC address is only used in Link Layer Discovery Protocol (LLDP) packets as required by the PROFINET specification.

Use either of these methods to find the device and port MAC addresses for a controller using RMCTools:

- Open the CPU Properties and select the **General** page. See General Module Properties for details.
- Open Communication Statistics, and select the **Ethernet Summary** page.

Use the appropriate method below to find the device MAC address on the RMC controller itself:

- **RMC75E**
  The device MAC address is printed on the label on the side of the RMC75E CPU module.
- **RMC150E**
  The device MAC address is printed on the label on the front of the CPU module.
- **RMC200**
  The device MAC address is listed in the Display Screen by browsing to **Ethernet** > **View IP Address** and scrolling to show the MAC address.

See Also

Understanding IP Addressing | Ethernet Overview

6.7.10. Application Protocols

6.7.10.1. Modbus/TCP

This topic describes the RMC support of the Modbus/TCP Ethernet protocol. Whether you are using it to communicate with the RMC from a PLC or from a custom application, this information may be useful.

The Modicon Modbus/TCP protocol is one of the simplest protocols and has an open standard. This means the protocol specification is publicly available. See the Modicon Modbus/TCP web site at https://www.modbus.org for complete details on the protocol.

Supported Modbus function codes:

The following function codes are supported. Function codes 3 and 16 are the most common.

- Read Multiple Registers (FC 3) - very common
- Read Input Registers (FC 4)
- Write Single Register (FC 6)
- Get Diagnostics (FC 8)
- Write Multiple Registers (FC 16) - - very common
- Read/Write Registers (FC 23)

Addressing and Data Format

See the Modbus Addressing topic.

Implementation Notes

Word Order

RMC registers are Big-Endian 32-bit words and use two (2) holding registers each. In 32-bit words, the RMC expects that the least significant 16 bits comes first. Most Modbus masters have a setting for selecting this, since the Modbus protocol never specified this.

Zero-Based vs. One-Based

Modbus addresses that the user sees are 1-based. Since all RMC registers are 32-bit and use two Modbus registers, all RMC register addresses are therefore odd, for example 404097. However, internally to the protocol, the addresses are one-based. Some device manufacturers have confused this and mistakenly made their protocol zero-based for the user. In such cases, you will need to enter your addresses in the external device as zero-based. For example, the RMC address 408001 would be entered 408000.

Common Function Codes

Typically, only functions 3 and 16 are used. Function 23 can be used to improve performance by doing a read and write in a single function. FC 4 and 8 were included for compatibility with existing Modbus/TCP masters, but otherwise add no value to the RMC.

Unit Identifier (Slave Address)

The Unit Identifier (Slave Address) field in the Modbus/TCP is only used by routing devices. Since the Slave Address is not used by the RMC, it should be set to zero (0) in requests to the RMC.

Advanced

- To create a connection with the RMC over Modbus/TCP, open a client socket with the RMC on TCP port 502. The RMC will only respond to Modbus/TCP requests on this port. Other ports are reserved or used by other protocols.
- The RMC handles incoming packets on a first-in first-out (FIFO) basis, making it possible to send multiple requests and then wait for the replies.
- The RMC can handle up to 64 open TCP/IP connections at once. Typically each device uses one connection at a time, allowing the RMC to be connected to up to 64 devices at once.
- Multiple-register reads and writes are limited to 100 16-bit words for writes and 125 16-bit words for reads—per the Modbus/TCP specification—even though the length field in the Modbus/TCP header makes it look like larger packets would be supported.
- Floating point numbers are 32-bit values and use two (2) holding registers each. The least significant 16 bits comes first.

Troubleshooting

If the data that is being read from or written to the RMC is appearing, but not correctly, the problem is likely one or more of the following:

- **The 16-bit word order is wrong**
  Find the setting in the Modbus master to swap the 16-bit word order within a 32-bit value.
- **Zero-based instead of one-based**
  The device manufacturer mistakenly made their protocol zero-based for the user. You will need to enter your addresses in the external device as zero-based. For example, the RMC address 408001 would be entered 408000.
- **Data Type**
  Make sure that if you are writing a 32-bit floating point value, that data type is correctly set in the external device to a 32-bit floating point (often called a REAL data tyep). For DINT or DWORD, the data type in the external device is commonly called 32-bit integer.

Tip: To help determine word order or zero-based errors, set an RMC variable (not the first variable, or some problem may not show up easily) to a DWORD type, then in the external device, write a hexadecimal value, such as 16#11223344 to the RMC. You will quickly see in the RMC what the problem is.

See Also

RMC Ethernet Protocols | Ethernet Overview | Modbus Addressing

6.7.10.2. CSP

This is a proprietary protocol developed by Allen-Bradley, Inc. Variants of this are used on Allen-Bradley's SLC 5/05 and PLC-5 controllers. This protocol is also used by the Allen-Bradley's SoftLogix 5 and RSLinx and SoftPLC Corporation's SoftPLC. Allen-Bradley does not publish the specifications for this protocol.

The CSP protocol is also known as "DF1 over Ethernet". It uses the same addressing format and function code types as the serial DF1 protocol. This common component is called PCCC.

CSP can be used to read and write any registers in the RMC. In the Allen-Bradley PLCs, MSG (message) blocks are used to read and write any registers in the RMC using CSP. See Using Allen-Bradley Controllers via Message Block for details.

Supported PCCC function codes:

- Diagnostic Status (06 03)
- Echo (06 00)
- SLC Protected Typed Logical Read with 2 Address Fields (0F A1)
- SLC Protected Typed Logical Read with 3 Address Fields (0F A2)
- SLC Protected Typed Logical Write with 2 Address Fields (0F A9)
- SLC Protected Typed Logical Write with 3 Address Fields (0F AA)
- PLC5 Typed Read (0F 68)
- PLC5 Typed Write (0F 67)
- PLC5 Word Range Read (0F 01)
- PLC5 Word Range Write (0F 00)

Addressing:

Same as DF1 (serial) - see the DF1 Addressing topic.

See Also
RMC Ethernet Protocols | Ethernet Overview | DF1 Addressing

6.7.10.3. FINS/UDP

The FINS/UDP protocol is an Omron protocol which can be used by a PLC program to transfer data and perform other services with a remote PLC connected on an Ethernet Network. The FINS protocol allows Omron PLCs to communicate with the RMC75E, RMC150E, and RMC200. The RMCs support FINS/UDP.

The FINS/UDP protocol is an open application protocol developed and used by Omron Electronics Inc. This protocol is available over a number of media, including Ethernet and serial. Additional information is available in the CS1 Communications Reference Manual, available on Omron's web site: https://www.omron.com/oei.

The FINS protocol uses a three-stage addressing system: network address, node number, and unit number. These three address elements have the following purposes:

- **Network Address**
  This value identifies the network on which the target node resides. The PLC looks up this number in its Local and Remote Network Tables to determine which local unit number should be used to send out the request. For Ethernet systems, typically only the Local Network Table is used to map a network address (for example, 1) to the unit number on the ETN01 Ethernet Unit (for example, 0). See your Omron PLC and/or CX-Programmer documentation for details on setting up the Local and Remote Network Tables.
- **Node Number**
  This value identifies the node on the network given by the network address. For Ethernet networks, this number must be converted to an IP address. This conversion is done by the ETN01 Ethernet Unit. Three methods are available. In the Automatic Address Generation method, the IP address is derived by combining the ETN01's IP Network Address (e.g. 192.168.0.0) with the Node Number (e.g. 5) to get an IP Address (e.g. 192.168.0.5). In the IP Address Table method, each Node Number maps to an IP Address via an entry in an IP Address Table. In the Combined method, the IP Address Table is used first, but if the Node Number is not found in the table, then the Automatic Address Generation method is used. See the Omron ETN01 Ethernet Unit manual for details on these methods.
- **Unit Number**
  This value identifies the unit number on the rack of the remote node identified by the Network Address and Node Number.

The RMC does not require manually setting these values in the RMC. Only the IP address needs to be set in the RMC; see the RMC Ethernet Setup topic for details.

Addressing:

See the FINS Addressing topic for information on the FINS address of RMC registers.

See Also
RMC Ethernet Protocols | Ethernet Overview | Using Omron PLCs with the RMC | FINS Addressing

6.7.10.4. Mitsubishi Procedure Exist Protocol

The Mitsubishi Procedure Exist Ethernet protocol is for Mitsubishi's Q-series QJ71E71-100 Ethernet module and the FX3U PLC. For communication with other Mitsubishi PLC modules, such as CPUs with built-in Ethernet, see Using the Mitsubishi Q-Series PLC with the RMC.

The QJ71E71-100 and FX3U support several communication protocols. The RMC75E, RMC150E and RMC200 support the Fixed Buffer communication with the **Procedure Exist** control method. It allows the Mitsubishi PLC to read and write binary data from an RMC over Ethernet. The RMC requires that the data sent via the Procedure Exist method is formatted as described in this topic. The Procedure Exist protocol is described in chapter 7 of the **Q Corresponding Ethernet Interface Module User's Manual (Basic)**. The manual part number is SH (NA)-080009-I. It is also described in the FX3U manual **User explanations for FX3U-ENET Ethernet Block**, manual no. JY997D18101.

The RMC uses port number 7171 hex (29,041 in decimal) for the Procedure Exist protocol.

A sample program for the Q-Series Procedure Exist method is available on the downloads page of Delta's website at https://deltamotion.com/dloads. This should be used as a starting point for any Mitsubishi Q-series program using the QJ71E71-100 Ethernet module and an RMC.

Note:
The RMC can also communicate with the Mitsubishi Q-series PLC via the QJ71MT91 Ethernet Modbus/TCP module. Modbus/TCP is a complete protocol and therefore using the QJ71MT91 requires less programming than using the QJ71E71-100. Consequently, the QJ71MT91 may be slightly easier to use with the RMC. The disadvantage is that the QJ71MT91 it is not as widely used as the QJ71E71-100. A sample program for using the QJ71MT91 is available on the downloads page of Delta's website.

Configuring the Q-Series

Mitsubishi's "Q Corresponding Ethernet Interface Module User's Manual (Basic)" (manual part number SH (NA)-080009-I) describes how to set up QJ71E71-100. It is very detailed, but is very large and may seem difficult. The sections below describe the settings necessary to use the QJ71E71-100 to communicate with the RMC.

The instructions below are from GX Developer version 8.25B.

1. I/O Assignment Settings

First, set the I/O settings. This is described in section 4.5.1 of the Q Corresponding Ethernet Interface Module User's Manual (Basic). Notice that there are no Switch settings to set for this module.

2. Network Parameter Settings

This is described in section 4.6 of the Q Corresponding Ethernet Interface Module User's Manual (Basic).

In the Project Data List, double-click **Network Param**, then click **MELSEC/Ethernet**. For each QJ71E71-100 module, set the following parameters:

Network Type	Set to **Ethernet**
Starting I/O Number	Set this to the head address of the Ethernet module
Network No.	See section 4.6 of the manual (Set to 1)
Total Stations	—
Group No.	Set to 1
Station No.	Set to 1
Mode	Set to **On line**
Operational Settings	Communication data Code: **Binary Code** Initial Timing: **Do not Wait for Open** IP Address: enter the IP Address of the QJ71E71-100. Send frame setting: **Ethernet(V2.0)**

	Enable Write at RUN time: cleared TCP Existence confirmation setting: **Use the KeepAlive**
Initial Settings	No settings required here.
Open Settings	For each RMC you wish to communicate with, you must set up a paired connection. To do so, on one row, set the following: Protocol: **TCP** Open System: **Active** Fixed Buffer: **Receive** Fixed Buffer communication procedure: **Procedure Exist** Paring open: **Enable** Existence confirmation: **Confirm** Host Station Port No.: **7171 hex (29,041 in decimal)** Transmission device IP address: Set this to the IP Address of the RMC Transmission target device Port No.: **7171 hex (29,041 in decimal)** Because Pairing open is set to Enable, the next row will automatically be set identically, except the Fixed Buffer will be set to **Send**. Notice that each QJ71E71-100 can communicate with up to 8 RMCs.

Opening a Connection on the Q-Series

Before the PLC can communicate with the RMC, the IP connection must be opened using the ZP.OPEN instruction, described in section 10.8 of the Q Corresponding Ethernet Interface Module User's Manual (Basic). Both connections of the pair (Send and Receive) must be opened individually.

The port number must be 7171 hex (29,041 in decimal).

Closing a Connection on the Q-Series

To close the IP connection to the RMC, use the ZP.CLOSE instruction, described in section 10.5 of the Q Corresponding Ethernet Interface Module User's Manual (Basic). Both connections of the pair (Send and Receive) must be closed individually.

Writing to the RMC with the Q-Series

Use the ZP.BUFSND Instruction to write to registers in the RMC. It is described in section 10.4 of the Q Corresponding Ethernet Interface Module User's Manual (Basic). The RMC requires that the Send Data of the BUFSND instruction is in the format shown below.

TxCount (16 bits)	0 (16 bits)	Register File (16 bits)	Register Element (16 bits)	Data Item 1 (32 bits)	Data Item 2 (32 bits)	…	Data Item n (32 bits)

Description:

TxCount	The number of 16-bit words written, not including this word.

0	This word must be 0.
Register File	This is the file number of the register's address in IEC format. For example, for %MD8.12, the file number is 8.
Register Element	This is the element number of the register's address in IEC format. For example, for %MD8.12, the element number is 12.
Data Time 1	The RMC has 32-bit registers. Therefore, you can only write 32-bit words. Most RMC registers are floating-point; a few are integers.
Data Item *n*	To write n 32-bit registers to the RMC, make sure the TxCount is correct. It should be (2 x n) + 3.

Example

A programmer wishes to write 5 values to the variable table in the RMC75 (address %MD56.0). These values are: 32.876, 1.0, 12.0, 5.432, 862.0.

The send data for the ZP.BUFSND instruction would be as follows:

13 (16 bits)	0 (16 bits)	56 (16 bits)	0 (16 bits)	32.876 (32 bits)	1.0 (32 bits)	12.0 (32 bits)	5.432 (32 bits)	862.0 (32 bits)

Reading from the RMC with the Q-Series

To read registers from the RMC, use the ZP.BUFSND instruction to send a read request and then use the ZP.BUFRCV instruction to read the received data. The ZP.BUFRCV instruction is described in section 10.2 of the Q Corresponding Ethernet Interface Module User's Manual (Basic).

Writing the Read Request

To send a request a read from the RMC, the send data of the ZP.BUFSND instruction must be formatted as shown below. Each box is a 16-bit word.

3 (16 bits)	Read Count (16 bits)	Register File (16 bits)	Register Element (16 bits)

Description:

3	This is actually TxCount, the number of 16-bit words written, not including this word. This value must be 3 when requesting a read.
Read Count	This is the number of 32-bit registers to be read from the RMC, starting at the address given by the file and element. The Read Count may be between 1 and 382.
Register File	This is the file number of the address of the first register to be read. For example, for %MD8.12, the file number is 8.
Register Element	This is the element number of the address of the first register to be read. For example, for %MD8.12, the element number is 12.

Receiving the Returned Data

After the write request is sent, the RMC will return the requested data. The QJ71E71 buffer memory location 20485 indicates whether data is being received. The bit in memory location

20485 corresponding to the current connection can be used to indicate when to read the receive buffer.

To read the receive buffer, use the ZP.BUFRCV instruction. The returned data is in the format shown below:

16-bit Count (16 bits)	32-bit Read Count (16 bits)	Data Item 1 (32 bits)	Data Item 2 (32 bits)	…	Data Item *n* (32 bits)

Description:

16-bit Count	This is the number of 16-bit words read from the buffer.
32-bit Read Count	This is the number of 32-bit registers that the RMC returned.
Data Item 1-*n*	This is the returned data.

Communicating Directly over TCP

For RMC 75E/150E firmware versions prior to 3.31.0, in applications where none of the RMC's protocols are supported by the master controller, but direct communication over TCP is allowed, Delta recommended that the Mitsubishi Procedure Exist protocol be implemented by the user manually. The remainder of this topic describes how to manually implement the Mitsubishi Procedure Exist protocol for communication with RMCs. The RMC75E/150E firmware versions 3.31.0 and newer and the RMC200 support the Delta Motion Control Protocol (DMCP), which is the preferred method of manual direct communication over TCP.

The RMC listens for Procedure Exist connections on TCP port hexadecimal 7171 (decimal 29041). The client TCP port number can be any valid port number. This protocol is a request/response protocol, meaning that the RMC will not send any data unless it receives a packet requesting that it do so.

All multiple-byte fields are encoded with the least-significant byte first. For example, a Packet Length value of 3 would be encoded as 03 00, and a 32-bit data value of 0x11223344 would be encoded as 44 33 22 11.

Writing Data to the RMC:

Writing to one or more registers in the RMC requires the following 2-packet sequence:

- First, the client sends the following packet to the RMC:

Offset	Data (hex)	Description
0-1	60 00	**Sub-Header.** These 2 bytes should always have these values in a write request.
2-3	mm nn	**Packet Length.** This value holds the number of 16-bit words in this packet, not including this field and the sub-header. For writing N registers to the RMC, this register will hold 3+2xN, since each RMC register uses 32 bits, or two 16-bit words.
4-5	00 00	**Must be Zero.** These two bytes must be zero (0) to indicate a write.
6-7	ff ff	**Register File.** This value holds the file portion of the address of the first register to write to. For example, for the IED address %MD12.2, this field would be 0C 00.

8-9	ee ee	**Register Element.** This value holds the element portion of the address of the first register to write to. For example, for the IEC address %MD12.2, this field would be 02 00.
10-...	...	**Data.** The values of each 32-bit register to write should follow the above header, with each register encoded in 4 bytes ordered from the least- to most-significant byte.

- The RMC will respond to this request with the following packet:

Offset	Data (hex)	Description
0	E0	**Acknowledge.** Indicates that the packet is an acknowledgement.
1	rr	**Response Code.** Indicates whether the write was successful or not. See the **Response Codes** section below.

Reading Data from the RMC:

Reading from one or more registers in the RMC requires the following 4-packet sequence:

- The client sends the RMC a read request:

Offset	Data (hex)	Description
0-1	60 00	**Sub-Header.** These 2 bytes should always have these values in a read request.
2-3	03 00	**Packet Length.** This value holds the number of 16-bit words in this packet, not including this field and the sub-header. For read requests, this field will always be 03 00.
4-5	mm nn	**Read Count.** This value holds the number of 32-bit RMC registers to read. This value cannot be greater than 382.
6-7	ff ff	**Register File.** This value holds the file portion of the address of the first register to read from. For example, for the IEC address %MD12.2, this field would be 0C 00.
8-9	ee ee	**Register Element.** This value holds the element portion of the address of the first register to read from. For example, for the IEC address %MD12.2, this field would be 02 00.

- The RMC acknowledges the request with the following short packet:

Offset	Data (hex)	Description
0	E0	**Acknowledge.** Indicates that the packet is an acknowledgement.
1	rr	**Response Code.** Indicates whether the read was successful or not. See the **Response Codes** section below.

- If the response above is success (00), then the RMC will then send a second packet, holding the data requested:

Offset	Data (hex)	Description
0-1	60 00	**Sub-Header.** These 2 bytes will always have these values in a read response.

2-3	mm mm	**Packet Length.** This value holds the number of 16-bit words in this packet, not including this field and the sub-header. For reading N registers from the RMC, this register will hold 1+2xN, since each RMC register uses 32 bits, or two 16-bit words.
4-5	nn nn	**Read Count.** This value holds the number of 32-bit RMC registers that were read. This will match the Read Count field in the read request.
6...	...	**Data.** The values of each register that was read will follow the above header, with each register encoded in 4 bytes ordered from the least- to most-significant byte.

- Finally, the client must acknowledge the RMC's read response with the following simple packet:

Offset	Data (hex)	Description
0	E0	**Acknowledge.** Indicates that the packet is an acknowledgement.
1	00	**Response Code.** Indicates success (00).

Register Addresses

This protocol uses the RMC's two-level file/element addressing format. See the IEC Addressing topic describes how this addressing format works.

Response Codes

Each acknowledge packet holds a response code, indicating whether the requested transaction was completed successfully or not. The RMC uses the following response codes:

Response Code (hex)	Description
00	**Success.**
40	**Protocol Error.** Indicates that the request violated the protocol described above in one of the following ways: • The **Packet Length** field was less than 3. • The **Register File** field was greater than 255 (RMC75E/150E) or 4095 (RMC200). • The **Register Element** field was greater than 4095. • The length of the **Data** field in a write request was not a multiple of 4 bytes. • The **Read Count** field in a read request was greater than 382.
50	**Bad Sub-header.** A request packet did not start with 60 00.

Example 1: Writing a Single Register

In this example, the client will write the value 0x11223344 to variable 0 (%MD56.0) on and RMC75E. Therefore, the Register File is 56, the Register Element is 0, and the Packet Length is 3+2xN or 5. Entering these values into the write request packet structure with the least-significant bytes first, gives us the following packet:

60 00 05 00 00 00 38 00

00 00 44 33 22 11

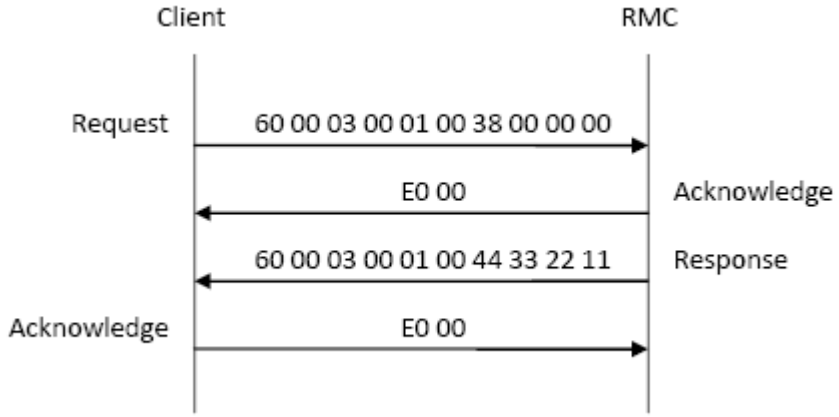
After the RMC has successfully received and processed this write request, it will respond with the following packet:

E0 00

Notice that the last byte is the response code, with 00 meaning success.

Example 2: Reading a Single Register

In this example, the client will read a value from variable 0 (%MD56.0) on an RMC75E. Therefore, the Register File is 56, and the Register Element is 0. Supposing that variable 0 held the value 0x11223344, then the following packets would be sent between the controllers:



See Also

RMC Ethernet Protocols | Ethernet Overview | Using the Mitsubishi Q-Series PLC with the RMC

6.7.10.5. DMCP

The Delta Motion Control Protocol (DMCP) is a simple Ethernet protocol intended for use with direct communication to the RMC75E, RMC150E and RMC200 over TCP or UDP. In applications where none of the RMC's other protocols are supported by the master controller, but direct communication over TCP or UDP is allowed, Delta recommends that the DMCP protocol be implemented by the user manually.

DMCP requires firmware version 3.31.0 or newer.

For details on implementing DMCP, see Communicating Directly over TCP or Communicating Directly over UDP.

Tip:
An example implementation of DMCP in the C programming language is available in the Examples section of Delta's online forum.

See Also

Communicating Directly over TCP | Communicating Directly over UDP

6.7.10.6. EtherNet/IP

6.7.10.6.1. EtherNet/IP Overview

EtherNet/IP is an open application protocol, maintained and distributed by ODVA (https://www.odva.org). EtherNet/IP is used by Ethernet modules for several PLC's including Allen Bradley, Schneider Electric, and Omron. EtherNet/IP is an Ethernet adaptation of the Control Information Protocol (CIP) in the same way that DeviceNet is a CAN adaptation of CIP and ControlNet is a CTDMA adaptation of CIP.

The RMC is a passive EtherNet/IP device. It does not establish its own I/O connections, nor does it initiate messaging transactions. Therefore, an active EtherNet/IP device or client is required to control the RMC or request data from the RMC.

The RMC statements of conformance and EDS files are available for download from the Delta website at https://deltamotion.com.

The RMC75E, RMC150E, and RMC200 support EtherNet/IP Messaging and EtherNet/IP I/O.

EtherNet/IP I/O

EtherNet/IP I/O provides a mechanism of deterministically sending data in both directions between a PLC and remote device. This data is sent on an interval called the Requested Packet Interval (RPI). EtherNet/IP I/O is a very fast and easy-to-use method of communication. It reduces the amount of ladder logic required for communication, and communication occurs even when the PLC is in Program mode.

Ethernet Protocol Mode Selecting PROFINET Protocol Mode (RMC200 Only)

In order to communicate via EtherNet/IP I/O, the RMC200 must be set to **EtherNet/IP Mode**:

1. In the Project pane, expand the RMC, and double-click the **CPU**.
2. On the **Ethernet** page, in the **Ethernet Protocol Mode** section, click **EtherNet/IP Mode**, then click **OK**.

The Ethernet Protocol Mode may also be viewed from the RMC200 Display Screen.

I/O Connections and Data

Connection Type	RMC75E	RMC150E	RMC200
Max Number of Input/Output Connections	1	1	3
Total Number of Connections, including Listen Only and Input Only	4	4	4
Connection#1 Size	125 input registers 124 output registers	125 input registers 124 output registers	360 input registers 360 output registers
Connection#2 Size	n/a	n/a	125 input registers 124 output registers
Connection#3 Size	n/a	n/a	125 input registers 124 output registers

The RMC also supports Input Only and Listening connections, as described in Setting up an EtherNet/IP I/O Connection.

The RMC supports only one type of controlling I/O connection at a time. Therefore, EtherNet/IP I/O cannot be used simultaneously with a PROFINET IO connection. Other protocols can be used simultaneously with EtherNet/IP.

To learn about setting up and using EtherNet/IP I/O, see the following topics:

- Using Allen-Bradley Controllers via EtherNet/IP I/O
- Using Omron Controllers via EtherNet/IP I/O
- Using Schneider Electric PLCs via EtherNet/IP I/O

- Setting up an EtherNet/IP I/O Connection
- Using an EtherNet/IP I/O Connection

Supported RPI (Requested Packet Interval)

	RMC75E	RMC150E	RMC200 CPU20L	RMC200 CPU40
Minimum RPI*	2 ms	2 ms	1 ms**	1ms
Typical RPI	20 ms			
Maximum RPI	10000 ms			

* The RPI may not be less than the loop time to which the RMC is set, and must be a multiple of the loop time.

** Not all CPU20L connections can be 1 ms. The CPU20L supports a bandwidth of 6000 packets per second and the CPU40 supports 8000 packets per second. The bandwidth of each connection is calculated as (2 / RPI). For example, an RPI of 1 ms is 2/0.001 = 2000 packet/sec. The CPU20L can support 3 connections at 1 ms RPI.

EtherNet/IP Messaging

EtherNet/IP explicit messaging allows the originator (PLC or HMI) to request individual services from the target device (RMC). These requests are made explicitly rather than being scheduled cyclically like I/O. Explicit messaging is much more flexible than I/O in terms of what data or services are accessed in the target device, since I/O connections must pre-configure the I/O data to be exchanged.

In most cases, RMC users use EtherNet/IP Explicit messaging to read and write registers in the RMC. Also, some advanced EtherNet/IP users may want to access standard CIP services and attributes.

The three types of EtherNet/IP explicit messaging services users may want to use in the RMC are:

- Read and/or write RMC registers using the Allen-Bradley PCCC/DF1 services.
- Read and/or write RMC registers using the Register Map Object.
- Access standard CIP services and attributes.

See Using EtherNet/IP Explicit Messaging for details on all three.

Supported Number of Connections

The maximum number of each type of connection supported by the RMCs is listed below:

Connection Type	RMC75E	RMC150E	RMC200
TCP	64	64	64
CIP (total of messaging and I/O)	32	32	20
CIP Messaging	up to 32	up to 32	16
CIP I/O	4 (1 with output data from the PLC)	4 (1 with output data from the PLC)	4 (up to 3 with output data from the PLC)

See Also

Ethernet Overview | Setting up an EtherNet/IP I/O Connection | Using an EtherNet/IP I/O Connection | Handling Broken EtherNet/IP I/O Connections | Troubleshooting EtherNet/IP I/O | Multiple EtherNet/IP I/O Connections | EtherNet/IP I/O Performance | Using EtherNet/IP Explicit Messaging | Using Allen-Bradley Controllers via Message Block | Using Allen-Bradley Controllers via EtherNet/IP I/O | Using Omron Controllers via EtherNet/IP I/O

6.7.10.6.2. Setting Up an EtherNet/IP I/O Connection

This topic describes the concepts involved in setting up an EtherNet/IP I/O connection. For step-by-step procedures for Allen-Bradley, Omron, or Schneider Electric PLCs, see the Using Allen-Bradley Controllers via EtherNet/IP I/O, Using Omron Controllers via EtherNet/IP I/O, and Using Schneider Electric PLCs via EtherNet/IP I/O topics. For details on controlling the RMC once a connection has been made, see the Using an EtherNet/IP I/O Connection topic.

Setting up an EtherNet/IP I/O connection involves the following concepts:

- Using the Correct EDS File
- Connection Type
- Requested Packet Interval (RPI)
- Multicast vs. Point-to-Point (Unicast)
- Input Data
- Output Data
- Using a Generic EDS File

Using the Correct EDS File

Electronic Data Sheet files (EDS files) are used by many vendors' EtherNet/IP configuration tools to help configure the communications. For configuration tools that do not use EDS files to configure EtherNet/IP I/O connections, see Using a Generic EDS below.

Several EDS files are available for the RMC controllers. EDS files are included when installing RMCTools to your computer. Current versions of EDS files are also available for download from Delta's website https://deltamotion.com. Each EDS file is packaged in a compressed (.zip) file with its corresponding icon.

From the following table, choose the correct EDS file for your RMC and RMC firmware version:

Product	RMC Firmware	EDS File	Version [1]	Download Location
RMC75E	2.20-3.40.x	rmc75e_v1.eds	1.x	https://deltamotion.com/files/eds/rmc75e_v1_eds.zip
	3.41.0-3.61.x	rmc75e_v2.eds	2.1	https://deltamotion.com/files/eds/rmc75e_v2_eds.zip
	3.62.0 or newer	rmc75e_v3.eds	3.1	https://deltamotion.com/files/eds/rmc75e_v3_eds.zip
RMC150E	2.20-3.40.x	rmc150e_v1.eds	1.x	https://deltamotion.com/files/eds/rmc150e_v1_eds.zip
	3.41.0-3.61.x	rmc150e_v2.eds	2.1	https://deltamotion.com/files/eds/rmc150e_v2_eds.zip
	3.62.0 or newer	rmc150e_v3.eds	3.1	https://deltamotion.com/files/eds/rmc150e_v3_eds.zip
RMC200 CPU20L	1.14.0 or newer	r200-cpu20L_v1.eds	1.1	https://deltamotion.com/files/eds/r200-cpu20L_v1_eds.zip
RMC200 CPU40	1.06.0 or newer	r200-cpu40_v1.eds	1.1	https://deltamotion.com/files/eds/r200-cpu40_v1_eds.zip

[1] This revision refers to the EtherNet/IP Identity Object revision, not the firmware revision. This revision only changes when the EtherNet/IP functionality is changed.

Once the correct EDS compressed file has been downloaded, extract the EDS file and icon to a temporary folder and import them into your EtherNet/IP configuration tool.

The correct EDS file is also embedded in the RMC controller itself and may be uploaded into the EtherNet/IP configuration tool, if supported by that particular tool. The steps required for this are tool-specific. Embedded EDS files are not supported by RMC75/150 firmware 3.40.x or older.

Connection Type

The RMC supports three types of I/O connections:

- **Input/Output**
  This connection is bidirectional: the originator (PLC or HMI) produces data consumed by the RMC and the target (RMC) produces data that is consumed by the originator. This connection type is also called an Exclusive Owner connection or the *controlling connection*.
- **Input Only**
  For this connection type, only the target (RMC) produces data, which is consumed by the originator (PLC or HMI). The originator will only send a heartbeat packet, sometimes at a reduced interval (less frequently than the RPI) used to allow the RMC to identify when the connection is broken.
- **Listen Only**
  This connection type is identical to an Input Only connection type, with one exception: a Listen Only connection can only exist when one of the other I/O connection types has been established. That is, a Listen Only connection cannot be established until an Input/Output or Input Only connection has been established, and conversely when the last non-Listen-Only connection has been closed or has timed out, the Listen Only connection will automatically close as well. This connection type is the least-frequently used. This connection type is not supported by RMC75/150 firmware 3.40.x or older.

Of these three, the Input/Output connection type is by far the most commonly used. The other two are generally only used when multiple I/O connections are used. See the Multiple EtherNet/IP I/O Connections topic for details.

Max Number of Connections and Data

Connection Type	RMC75E	RMC150E	RMC200
Total I/O Connections	4	4	4
Input/Output Connections	1	1	up to 3
Input Only Connections	up to 4	up to 4	up to 4
Listen Only Connections	up to 4	up to 4	up to 4
Connection#1 Size	125 input registers 124 output registers	125 input registers 124 output registers	360 input registers 360 output registers
Connection#2 Size	n/a	n/a	125 input registers 124 output registers
Connection#3 Size	n/a	n/a	125 input registers 124 output registers

When using the RMC's EDS file in your EtherNet/IP configuration tool, you should be able to select the I/O connection type from a list. Notice that in addition to the three listed above, you may also see **Input/Output with Config** and **Input Only with Config**. These connection types are generally not used. If you are using a Generic EDS File (as required by RSLogix 5000 prior to version 20), see the Using a Generic EDS File section below.

Since the RMC200 supports 3 I/O connections, it is possible for several PLCs to communicate with the RMC at once, each with a controlling connection. This is not possible to for the RMC75 and RMC150, as they support only one I/O connection.

Requested Packet Interval (RPI)

EtherNet/IP I/O sends data between the communicating devices at the **Requested Packet Interval** (RPI). The RPI is configured in the EtherNet/IP controller (for example, RSLogix 5000), not in RMCTools. The RMC supports the following RPIs, based on the number of simultaneous I/O connections established (see the Multiple EtherNet/IP I/O Connections topic):

I/O Connections	Minimum RPI RMC75/150	Minimum RPI RMC200 CPU20L	Minimum RPI RMC200 CPU40	Typical RPI	Maximum RPI

1 (typical)	2 ms	1ms*	1ms	20 ms	10,000 ms
2	3 ms				
3	4 ms				
4	4 ms				

* Not all CPU20L connections can be 1 ms. The CPU20L supports a bandwidth of 6000 packets per second and the CPU40 supports 8000 packets per second. The bandwidth of each connection is calculated as (2 / RPI). For example, an RPI of 1 ms is 2/0.001 = 2000 packet/sec. The CPU20L can support 3 connections at 1 ms RPI.

The RPI must be both an integer number of milliseconds and a multiple of the RMC's Loop Time. For example, an RPI of 9 ms would not be allowed for a loop time of 2 ms or 4 ms, but would be for 1 ms or lower loop times.

Delta recommends using the slowest RPI that meets the requirements of your application in order to reduce network requirements. See the EtherNet/IP I/O Performance topic for more details.

Multicast vs. Point-to-Point (Unicast)

The RMC supports both multicast and unicast (point-to-point) I/O connections. In almost all cases, unicast I/O connections should be selected whenever supported by the EtherNet/IP I/O controller, in order to reduce network traffic.

The RMC75/150 require multicast I/O connections only when multiple I/O connections will be established at once, which is quite rare. The RMC200 allows multiple unicast I/O connections.

Input Data

The Input Data is the data produced by the RMC and sent to the PLC. By default, the source of the Input Data is the Indirect Data Map, beginning with Indirect Data Map item 0. First, you must set up the Indirect Data Map in RMCTools to map to the RMC registers you want to transfer. Typically, you would include Actual Positions, Status and Error Bits for each axis of control, task status registers so the PLC can tell which user programs are running, and any other registers of your choice. The Input Data source can be changed to something other than the Indirect Data Map, as described below, but the Indirect Data Map is used in nearly all EtherNet/IP I/O applications.

The size of the Input Data is specified in the PLC when configuring the EtherNet/IP I/O connection. This should be set to the number of registers you have set up in the Indirect Data Map, plus the Sync Register, if used. For details on the Sync Register, see the Using an EtherNet/IP I/O Connection topic.

Once an EtherNet/IP I/O connection is established, the Input Data will automatically be sent from the RMC to the PLC each RPI. The PLC can use the data whenever it needs it.

Setting the Input Data Source in RMCTools

To set the source in the RMC for the Input Data:

1. In the Project pane, expand the **Modules** folder, double-click the CPU module, and choose **EtherNet/IP**.
2. Under **I/O Connection Settings**, in the **Outgoing Data** box, use the browse button to use the **Address Selection Tool** to find the location you want to use.
3. If using multiple connections, repeat for the other connections.

Notice that the length of the Input Data is specified in the PLC, not in RMCTools.

Output Data

The Output Data is the data sent from the PLC and consumed by the RMC. By default, the destination of the Output Data is the Indirect Data Map. This allows the PLC to easily write to whatever registers you have mapped into the Indirect Data Map, such as variables and axis command registers. The destination of the Output Data can be changed to something other than the Indirect Data Map registers, as described below. However, the Indirect Data Map works very well for most applications.

The size of the Output Data is specified in the PLC when configuring the EtherNet/IP I/O connection. The Output Data size must be the number of registers you wish to write, plus the Sync Register, if used. For details on the Sync Register, see the Using an EtherNet/IP I/O Connection topic.

Once an EtherNet/IP I/O connection is established, the Output Data is used to write to the RMC. Refer to the Using an EtherNet/IP I/O Connection topic for details.

Setting the Output Data Destination in RMCTools

To set the destination in the RMC for the Output Data:

1. In the Project pane, expand the **Modules** folder, double-click the CPU module, and choose **EtherNet/IP**.
2. Under **I/O Connection Settings**, in the **Incoming Data** box, use the browse button to use the **Address Selection Tool** to find the location you want to use.
3. If using multiple connections, repeat for the other connections.

Notice that the length of the Output Data is specified in the PLC, not in RMCTools.

Using a Generic EDS File

Some PLC EtherNet/IP configuration tools do not support importing third-party EDS files. In these cases, additional connection configuration information usually must be entered into a generic device template. The following charts provides the correct values for various connection settings that may be required by your EtherNet/IP configuration tool:

RMC75/150 Connections:

Setting	Connection Type		
	Input/Output	**Input Only**	**Listen Only**
Input Connection Point[1,2]	1		
Output Connection Point[1,2]	2	32	33
Config Data Assembly Instance[2]	4		
Connection Path[2]	20 04 24 04 2C 02 2C 01	20 04 24 04 2C 20 2C 01	20 04 24 04 2C 21 2C 01
Transport Class	1[3]		
Trigger Type	Cyclic[3]		
O->T Format	32-bit Run/Idle[3]	Heartbeat	
O->T Fixed/Variable	Fixed[3]		
O->T Connection Type	Point-to-point[3]		
O->T Priority	Scheduled[3]		
O->T Size	4, 8, …, 496 bytes	0 bytes	
T->O Format	Modeless[3]		
T->O Fixed/Variable	Fixed[3]		
T->O Connection Type	Point-to-point or Multicast		
T->O Priority	Scheduled[3]		
T->O Size	4, 8, …, 500 bytes		
O->T RPI	2ms to 10000ms		
T->O RPI	2ms to 10000ms		
Configuration Data Size	0 bytes		

RMC200 Input/Output Connections:

Setting	Input/Output Connection #1	Input/Output Connection #2	Input/Output Connection #3
Input Connection Point[1,2]	1	5	7
Output Connection Point[1,2]	2	6	8
Config Data Assembly Instance[2]	4	4	4
Connection Path[2]	20 04 24 04 2C 02 2C 01	20 04 24 04 2C 06 2C 05	20 04 24 04 2C 08 2C 07
Transport Class	1[3]		
Trigger Type	Cyclic[3]		
O->T Format	32-bit Run/Idle[3]		
O->T Fixed/Variable	Fixed[3]		
O->T Connection Type	Point-to-point[3]		
O->T Priority	Scheduled[3]		
O->T Size	4, 8, …, 1440 bytes	4, 8, …, 496 bytes	4, 8, …, 496 bytes
T->O Format	Modeless[3]		
T->O Fixed/Variable	Fixed[3]		
T->O Connection Type	Point-to-point or Multicast		
T->O Priority	Scheduled[3]		
T->O Size	4, 8, …, 1440 bytes	4, 8, …, 500 bytes	4, 8, …, 500 bytes
O->T RPI	1ms to 10000ms		
T->O RPI	1ms to 10000ms		
Configuration Data Size	0 bytes		

RMC200 Input Only Connections:

Setting	Input Only Connection #1	Input Only Connection #2	Input Only Connection #3
Input Connection Point[1,2]	1	5	7
Output Connection Point[1,2]	32	32	32
Config Data Assembly Instance[2]	4	4	4
Connection Path[2]	20 04 24 04 2C 20 2C 01	20 04 24 04 2C 20 2C 05	20 04 24 04 2C 20 2C 07
Transport Class	1[3]		
Trigger Type	Cyclic[3]		
O->T Format	Heartbeat		
O->T Fixed/Variable	Fixed[3]		
O->T Connection Type	Point-to-point[3]		
O->T Priority	Scheduled[3]		
O->T Size	0 bytes		
T->O Format	Modeless[3]		
T->O Fixed/Variable	Fixed[3]		

T->O Connection Type	Point-to-point or Multicast		
T->O Priority	Scheduled[3]		
T->O Size	4, 8, …, 1440 bytes	4, 8, …, 500 bytes	4, 8, …, 500 bytes
O->T RPI	1ms to 10000ms		
T->O RPI	1ms to 10000ms		
Configuration Data Size	0 bytes		

RMC200 Listen Only Connections:

Setting	Listen Only Connection #1	Listen Only Connection #2	Listen Only Connection #3
Input Connection Point[1,2]	1	5	7
Output Connection Point[1,2]	33	33	33
Config Data Assembly Instance[2]	4	4	4
Connection Path[2]	20 04 24 04 2C 21 2C 01	20 04 24 04 2C 21 2C 05	20 04 24 04 2C 21 2C 07
Transport Class	1[3]		
Trigger Type	Cyclic[3]		
O->T Format	Heartbeat		
O->T Fixed/Variable	Fixed[3]		
O->T Connection Type	Point-to-point[3]		
O->T Priority	Scheduled[3]		
O->T Size	0 bytes		
T->O Format	Modeless[3]		
T->O Fixed/Variable	Fixed[3]		
T->O Connection Type	Multicast		
T->O Priority	Scheduled[3]		
T->O Size	4, 8, …, 1440 bytes	4, 8, …, 500 bytes	4, 8, …, 500 bytes
O->T RPI	1ms to 10000ms		
T->O RPI	1ms to 10000ms		
Configuration Data Size	0 bytes		

[1] The term 'Assembly Instance' is used by some applications instead of 'Connection Point'.

[2] The Input and Output Connection Points and Config Data Assembly Instance are implied by the Connection Path. Therefore, configuration tools will generally either ask for the Connection Points and Instances or the Connection Path but not both.

[3] This value is the common default value for this setting and the option to change this field may not be offered by the configuration tool.

Unfortunately which fields are required and how they are presented varies significantly from one tool to the next. For this reason, most EtherNet/IP configuration tools are moving toward supporting importing EDS files to reduce the complexity of setting up an I/O connection.

Advanced Multicast Settings

In addition to the standard settings described above, the RMC provides advanced multicast I/O connection settings. These include the ability to override the default multicast address and Time To Live (TTL) value. See the EtherNet/IP Settings Page topic for details on these settings.

See Also

EtherNet/IP Overview | Using an EtherNet/IP I/O Connection | Using Allen-Bradley Controllers via EtherNet/IP I/O | Using Omron Controllers via EtherNet/IP I/O

6.7.10.6.3. Using an EtherNet/IP I/O Connection

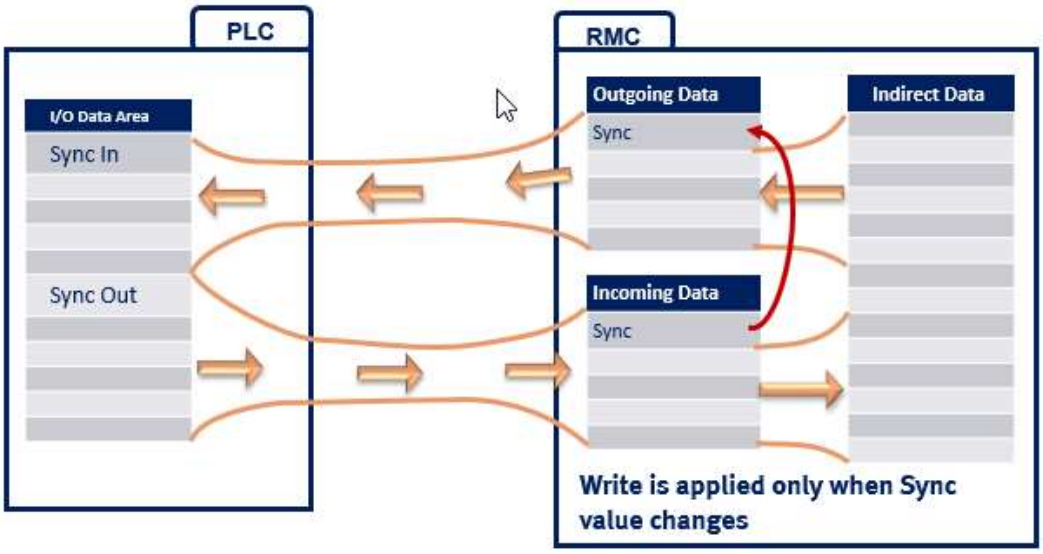
This topic describes how to control the RMC over an EtherNet/IP I/O connection. For details on setting up a connection, see the Setting up an EtherNet/IP I/O Connection topic.

Understanding the Sync Register

A cyclic I/O data connection with the RMC can operate in one of two modes: *without* a Sync Register and *with* a Sync Register. Delta normally recommends using the Sync Register.

Without a Sync Register:
The RMC processes incoming data each time any register in the block changes.

With a Sync Register:
The first Input Data register and first Output Data register are each reserved as a Sync Register, and the RMC processes incoming data only when the Sync Register changes. Once the incoming data has been processed, the incoming Sync Register value is echoed in the outgoing data. For RMCs that support multiple I/O connections, each connection has a Sync Register.
The Sync Registers show up in the Input and Output Data images in the PLC. In the RMC, they are only visible in the Event Log. See Troubleshooting EtherNet/IP I/O for details.



Why Use a Sync Register?

The Sync Register gives the user tight control over synchronization between the PLC and RMC and helps prevent synchronization issues in the communication logic. In applications that do not require this level of synchronization, the user can choose to not include the Sync Register.

The synchronization provided by the Sync Register is useful when:

- **Treating writes to a block of registers as a consistent block.**
Not all PLCs synchronize their I/O with the PLC scan. Therefore, while data is being placed in the Output Data by the PLC program, a copy of the Output Data could prematurely be sent to the RMC, mixing some old data with some new data. When the Sync Register is used, this problem is avoided by the simple convention of having the PLC program update the Sync Register after all other registers have been set to the desired values. The RMC is then guaranteed to receive all the data in a single block.

- **Coordinating when Input Data has been updated to reflect an issued command.**
  This benefit is best demonstrated with an example. Suppose that an axis is holding position with its **In Position** status bit set. If the PLC issues a new move command to a new position, then we will see that the **In Position** bit will go off when the command is first processed and remain off until the new move completes and the axis is at the new position. How then does the PLC know when its copy of the **In Position** bit received in the Input Data reflects the new command having been received? When the Sync Register is used, the PLC can simply wait until the Sync Register coming from the RMC matches the one it sent to the RMC when issuing the command, and it can then safely examine the **In Position** bit.

  Setting the Sync Register Mode

  To select whether the Sync Register is used:

  1. In the Project pane, expand the **Modules** folder, double-click the CPU module, and choose the **EtherNet/IP** page.
  2. In the **I/O Connection Settings** section, check the **Use a Sync Register** box to use the sync register, or clear the box to not use it. If the RMC supports multiple I/O Connections, make sure to set the Sync Register setting as desired for each connection. Not using the Sync Register is not supported by RMC75/150 firmware 3.38.0 or older.

Using Input Data in the PLC

If the Sync Register is used for a connection, then the first register of the Input Data is reserved as the Sync Register. All other registers in the Input Data are defined by the Outgoing Cyclic I/O Data address specified in the EtherNet/IP Settings Page. When the I/O connection is established, the Input Data is automatically updated in the PLC at the Requested Packet Interval (RPI). The PLC can simply use the data whenever it needs it. The RMC does not wait for the Sync Register to change to update its Outgoing Cyclic I/O Data.

Care must be taken to consider the data types of each Input Data register. The RMC includes registers of type REAL, DINT, and DWORD, and when you assign the Outgoing Cyclic I/O Data, you have the ability to mix registers of different types in the Input Data. It may be necessary to copy the data from one type of memory to another in the PLC to ensure that it gets interpreted correctly. For example, on Allen-Bradley PLCs, the COP and CPS instructions can be used to copy the data from an array of one type to tags of the correct type.

Writing Output Data with a Sync Register from the PLC

To write data in the RMC controller using I/O data when the Sync Register is used, do the following in the PLC:

1. Wait for the Sync Registers in the Input Data and Output Data to match.
2. Update any Output Data register values, except the Sync Register.
3. Change the value of the Output Data Sync Register.
   The easiest way to do this is to add one to it. However, since the Sync register is a 32-bit floating point value, one cannot be added to it after it reaches 16,777,216. One method of handling this is to add one and then MOD it with some large number, such as 1,000. This will make the register count from 0 to 999, and then wrap back down to 0 without an error. Take care to ensure that you only update the Sync Out register once so that the commands do not get re-issued.
4. Do not change any Output Data registers until the Input and Output Sync Registers match again.

The RMC will apply the Output Data register contents only once.

For multiple connections to one RMC, each connection has an individual Sync Register. The connections are independent of each other, and the data may not be transferred at exactly the same time, even if the Sync Registers are changed simultaneously. Therefore, it is best practice that for data that must be sent simultaneously, that they are sent in the same Output Data block.

Writing Output Data without a Sync Register from the PLC

To write data in the RMC controller using I/O data without the Sync Register, simply change the Output Data registers in the PLC and they will get applied in the RMC. Often care must be taken

as to the order that the registers are updated to ensure that actions triggered by changes to registers in the RMC have updated values for any parameters they use. It is often necessary to also understand whether the I/O data is updated synchronously to the PLC scan or not. For multiple connections to one RMC, the connections are independent of each other, and the data may not be transferred at exactly the same time.

Writing to the Command Area from the PLC

The Incoming Cyclic I/O Data area can be set up to point to the Command Area registers directly, or indirectly through the Indirect Data Map. The Sync Register should always be used if the PLC will be writing commands to the Command Area. The following sequence is recommended:

1. **Wait Until the Sync In and Sync Out Registers Match**
   If they do not match, then this means that another command or set of commands is in progress.
2. **Clear Old Commands from the Command Registers**
   Clear old commands from the command registers for each axis. Otherwise, when the Sync Out register is changed, the commands would be re-issued. One method of clearing the old commands is to fill the Output Data array with zeroes.
3. **Write to the Command Registers**
   Write the Command registers and all required command parameters to the Output Data for all commands you want to issue. You can issue up to one command per axis. Leave the Command register set to 0 for each axis that will not receive a command.
4. **Change the Sync Out Register**
   The easiest way to do this is to add one to it. However, since the Sync register is a 32-bit floating point value, one cannot be added to it after it reaches 16,777,216. One method of handling this is to add one and then MOD it with some large number, such as 1,000. This will make the register count from 0 to 999, and then wrap back down to 0 without an error. Take care to ensure that you only update the Sync Out register once so that the commands do not get re-issued.
5. **Wait Until the Sync In and Sync Out Registers Match**
   It is important to wait until this occurs before using the status bits in the Input Data (if the Input Data includes any status bits).

For multiple connections to one RMC, each connection has an individual Sync Register. The connections are independent of each other, and the data may not be transferred at exactly the same time, even if the Sync Registers are changed simultaneously. Therefore, it is best practice that if commands need to be sent simultaneously, that they are sent in the same Output Data block.

Triggering Commands through the Variable Table

If you have set the Output Data to write to Variable Table registers directly or via the Indirect Data Map, you can use them together with the Program Triggers to start User Programs that issue commands. For advanced applications that require issuing commands and writing to variables, this can be a very efficient method of communication.

To issue a command with this method, you will first need to do the following:

1. Create a user program that contains the commands you need.
2. Define a variable in the Variable Table that will be used to start the program. Name it StartProgram, for example.
3. Create a Program Trigger condition that looks at the value of StartProgram. When the StartProgram variable becomes a certain value, it will start the specified user program. This StartProgram variable can be used to start different programs based on its value.
4. The user programs should always reset the StartProgram variable to some starting value (such as 0). This is so the Program Triggers will always see the number change when the PLC writes to it. If the number already was 6, and the PLC writes 6, the value doesn't change, and the Program Triggers do not know to start a user program. Typically, the first step in a program should reset the value of the variable.

Once you have created the Program Triggers and User Programs, make sure the RMC is in RUN Mode, and use one of the procedures below to start a User Program:

With a Sync Register:

1. Wait until the Sync In and Sync Out Registers match.
2. Write a number to the StartProgram variable. The number will specify which user program to run.
3. If you need to write to other variables (e.g. for defining speeds, setpoints, etc.), do so.
4. Change the Sync Out Register.
   The easiest way to do this is to add one to it. However, since the Sync register is a 32-bit floating point value, one cannot be added to it after it reaches 16,777,216. One method of handling this is to add one and then MOD it with some number, such as 1,000. This will make the register count from 0 to 999, and then wrap back down to 0 without an error. Take care to ensure that you only update the Sync Out Register once so that the commands do not get re-issued.
5. Wait until the Sync In and Sync Out Registers match.
   When the Sync In Register has changed, you know the data has been applied to the RMC. It is important to wait until this occurs before using the status bits in the Input Data (if the Input Data includes any status bits).
   After the Sync registers match, you can look at the Task Status and Current Program registers to determine if the task is running the program, and when it stops. These registers should be included in the Input Data.

Without a Sync Register

1. If you need to write to other variables that are used by the user program (e.g. for defining speeds, setpoints, etc.), do so first.
2. Write a number to the StartProgram variable. The number will specify which user program to run.

Handling Broken Connections

Many applications require some type of action if the EtherNet/IP I/O connection breaks during machine operation. For information on how to handle a broken connection in either the RMC or PLC, see the Handling Broken EtherNet/IP Connections topic.

Troubleshooting

See the Troubleshooting EtherNet/IP I/O topic.

How the I/O Data is Applied to the RMC

When the Sync register is used:
When the RMC sees that the Sync Register changes for a given connection, the RMC applies all the data from the PLC to the specified Incoming Cyclic I/O Data area, for that connection.

When the Sync Register is not used:
When the RMC sees that any register value changes for a given connection, the data for that connection is immediately written to the specified Incoming Cyclic I/O Data area.

In certain cases, it may take the RMC more than one loop time to apply the incoming data. Therefore, not all of the data may be valid at the same time. For example, if you are using any of the data to trigger a user program that also uses some of the data, it is important that the user program is not triggered before *all* the data is valid, otherwise, the user program may use old data. The RMC takes at least one loop time per connection to apply the data.

Data Consistency

When all the incoming data is applied in the same RMC loop time, the data is called **consistent**. If the data is consistent, you can use all the data immediately. If the data is not consistent, you need to make sure that you do not use the data until it has all been updated. For multiple connections to one RMC, the data between connections should be assumed to not be consistent.

The following rules apply to data consistency within a single given connection based on where the consuming address is in the RMC:

Variable Table - Current Values

- **1 ms or longer loop time:** Up to 256 variables can be written at once within a single motion loop. If more than 256 variables are being written to by the I/O connection, then the first 256 variables will be written on the first motion loop, and the remaining variables will be written on the next motion loop.
- **500 us loop time:** Up to 128 variables can be written at once within a single motion loop. If more than 128 variables are being written to by the I/O connection, then 128 variables will be written each motion loop until the entire transaction is done.
- **250 us or shorter loop time:** Up to 64 variables can be written at once within a single motion loop. If more than 64 variables are being written to by the I/O connection, then 64 variables will be written each motion loop until the entire transaction is done.

Command Area

Writing the data to the Command Area may take up to 3 ms. After the write is complete, the entire command block is submitted as a single consistent block. Therefore the data is treated as a single consistent block, although it may take several loop times before the block is submitted.

Indirect Data Map

The only data guaranteed to be consistent are variables, if they are placed first in the Incoming Data area of the Indirect Data Map. The size of guaranteed consistent data depends on the loop time:

- **1 ms or longer loop time:** The first 100 registers will be consistent if they are all variables.
- **500 us loop time:** The first 50 registers will be consistent if they are all variables.
- **250 us or shorter loop time:** The first 25 registers will be consistent if they are all variables.

As soon as the first register is encountered that is not a current or initial variable register, the guarantee for data consistency no longer applies. Remaining registers in the I/O block will be processed as time allows on subsequent motion loops, taking up to a maximum of 10 ms. For this reason, variables should be included first in the Incoming I/O Data Area of the Indirect Data Map in order to ensure their consistency.

Notice that if commands are mapped to the Indirect Data, the commands will all be submitted at the same time, after all values have been written.

Other Locations

No data consistency is guaranteed when consuming in other areas (such as axis parameters). The entire write can take as long as 10 ms, with the write being deferred to the next control loop at any point as required to meet loop time requirements.

See Also

EtherNet/IP Overview | Setting up an EtherNet/IP I/O Connection | Handling Broken EtherNet/IP I/O Connections | Troubleshooting EtherNet/IP I/O | Multiple EtherNet/IP I/O Connections | EtherNet/IP I/O Performance | Using Allen-Bradley Controllers via EtherNet/IP I/O | Using Omron Controllers via EtherNet/IP I/O

6.7.10.6.4. Handling Broken EtherNet/IP I/O Connections

It is important to detect loss of an EtherNet/IP I/O connection quickly. EtherNet/IP supports a variable timeout value, which is expressed in terms of Requested Packet Intervals (RPIs). The RMC supports

multipliers ranging from 4x to 512x. The PLC may allow you to select this timeout multiplier value, or may fix it to a certain value. For example, the ControlLogix establishes its EtherNet/IP I/O connections with a timeout of 32 RPIs. Therefore, an RPI of 4.0 ms will have a timeout of 32 x 4.0 ms or 128 ms.

When either device in an I/O connection does not receive a packet from the other device for the timeout interval, it closes the connection and typically indicates this condition to the main program. The method of indicating this condition depends on the actual device. This topic describes the methods used by the RMC and several common PLCs.

Handling Broken I/O Connections in the RMC

The RMC has tags that indicate the state of the I/O connections. The user can use these tags to qualify whether certain operations in the User Programs can be done, or they can use these tags in the Program Triggers to respond to the change in state of the connection.

To find these tags when editing the Program Triggers or a User Program, use the Address Selection tree and browse to:

Controller > Communication Settings > Ethernet

RMC75/150 Tag Name	RMC200 Tag Name	Description
_Enet.CCStatus.Active	_Enet.IOConn1Status.Active _Enet.IOConn2Status.Active _Enet.IOConn3Status.Active	**I/O Connection Active** This bit is set as long as an I/O connection is currently active. If the connection is closed or timed out, this bit will be cleared.
_Enet.CCStatus.TimedOut	_Enet.IOConn1Status.TimedOut _Enet.IOConn2Status.TimedOut _Enet.IOConn3Status.TimedOut	**I/O Connection Timed Out** This bit is set when an I/O connection timed out. Notice that this is only one method of a connection being closed (another example is the PLC intentionally closing it). This bit is cleared when the I/O connection is re-opened. The user can look for the rising edge of this bit in the Program Triggers to respond to a time-out. A time-out can occur when the cable is disconnected, or when the client is powered off or reset.
_Enet.PLCStatus	_Enet.IOConn1PLCState _Enet.IOConn2PLCState _Enet.IOConn3PLCState	**I/O Connection PLC Status** This register indicates the state of the controlling PLC. This register (DINT data

		type) can hold one of three values: Unknown (0), Run (1), and Program (2). The Unknown state applies whenever no controlling EtherNet/IP connection is active.

Handling Broken I/O Connections in the PLC

The method of detecting and handling a broken EtherNet/IP I/O connection varies from PLC to PLC. Review the documentation included with your PLC or EtherNet/IP communication card for details. Some PLC detection procedures are shown below. However, these are not comprehensive procedures and do not cover all PLCs.

Allen-Bradley ControlLogix and CompactLogix PLCs

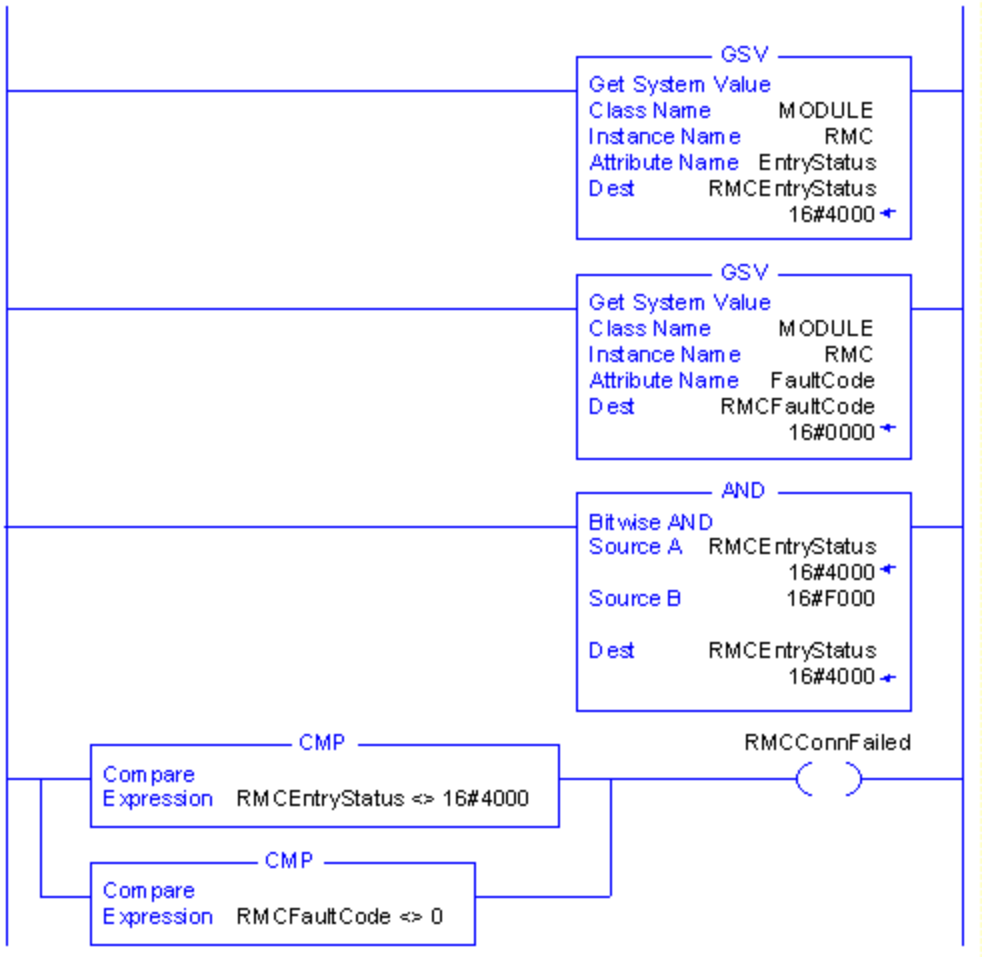
The ControlLogix and CompactLogix controllers have two methods of handling a broken connection with an EtherNet/IP device such as the RMC:

- **Major Fault if Connection Fails:**
  In RSLogix 5000, in the Module Properties dialog box for the RMC, on the **Connection** tab, the **Major Fault On Controller If Connection Fails In Run Mode** check box can be checked to fault the ControlLogix when the EtherNet/IP connection to the RMC is broken.
- **Connection Status Tags:**

  For EDS File:
  If an EDS file was used to configure the connection, the input tag for each connection has a ConnectionFaulted boolean tag that can be used to read the status of the connection in ladder logic:

MyRMC150:I	{...}	{...}		024E:RMC150E...
MyRMC150:I.ConnectionFaulted	0		Decimal	BOOL
MyRMC150:I.Data	{...}	{...}	Float	REAL[64]
MyRMC150:I.Data[0]	0.0		Float	REAL
MyRMC150:I.Data[1]	0.0		Float	REAL

  For Generic Ethernet Module:
  If the connection was configured as a Generic Ethernet Module, the Get System Value (GSV) block must be used to read the status of the connection in ladder logic. The following ladder logic demonstrates this:



The core of this ladder segment is reading the EntryStatus and FaultCode attributes from the RMC MODULE object using the GSV blocks. The MODULE objects are internal to the ControlLogix and represent external modules. In the Instance Name field of the GSV blocks, type the name you selected for the particular RMC module.

If the connection to the module is running, then the high four bits of the EntryStatus will be equal to 4 and the FaultCode will be equal to 0. This is described in the RSLogix 5000 online help's "Accessing the MODULE Object" topic.

The above ladder masks off the low 12 bits of the EntryStatus using an AND block, and then sets the RMCConnFault coil to indicate whether or not the connection is faulted.

Omron CS1/CJ1/CJ2 PLCs

Refer to section 6-3-1 **Ladder Programming Related to Tag Data Links** in the **SYSMAC CS and CJ Series EtherNet/IP Units Operation Manual** (document W-465-E1-06) from Omron. This section describes using four flags to determine that the I/O communications are operating normally with a specific target:

1. The Unit Error Occurred Flag (n+10, bit 00) must be OFF.
2. The Online Flag (n+11, bit 00) must be ON.
3. The Tag Data Link Operating Flag (n+11, bit 01) must be ON.
4. The Normal Target Node Flag (in words n+20 to n+23) for the corresponding target node must be ON.

Notice that the Target Node PLC Operating Flag and Target Node PLC Error Flag also discussed in this section are not used with RMC controllers.

Please refer to the above-referenced section of the Omron manual for additional information including differences between Omron EtherNet/IP Unit revisions and example ladder logic.

Schneider Electric Quantum, Premium, and Modicon M340 PLCs

Detecting a broken EtherNet/IP I/O connection on the Schneider Quantum and Premium PLCs involves testing the appropriate connection health status bit in the EtherNet/IP controller's Derived Data Type (DDT) Variables.

The procedure for determining the location of the appropriate bit differs between the first and second generation of communication modules from Schneider Electric:

Quantum 140 NOC 771 00 and Premium TSX ETC 100 Modules:

These status bits are located in the DDT input structure in a field called Status, which is an array of 16 BYTEs. To determine the correct bit to check for a given target device, you must locate the **Connection Bit Health Offset** field for the device:

1. Open the Unity Pro EtherNet/IP Configuration Tool.
2. Open the properties for the desired target device.
3. Select the **Connection** tab.
4. Under **Configured Connections**, select the **General** item for the connection whose status you want to monitor. In most cases, there will only be a single connection in this list.
5. Under **Connections Parameters**, locate the item named **Connection Bit Health Offset**. This is usually the first item in the list. This should be a value in the range of 0..127.

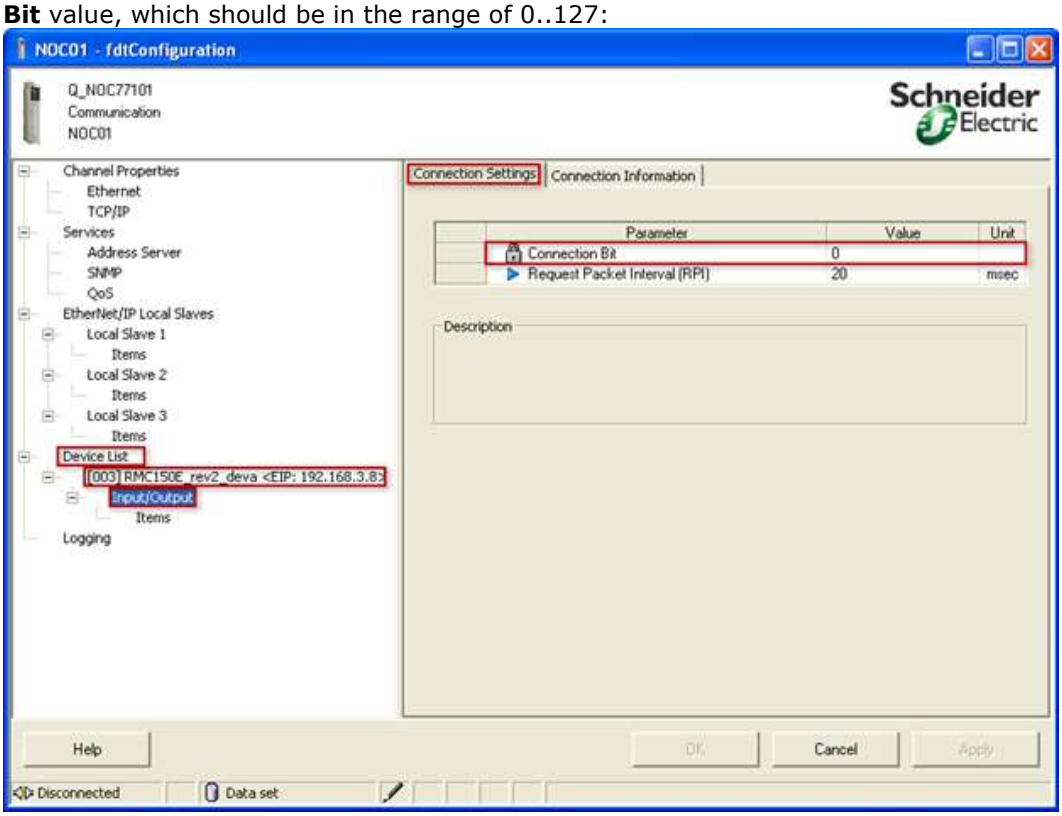
The value of the **Connection Bit Health Offset** is the bit offset from the start of the 16-byte Status[] array located in the Derived Variables for the EtherNet/IP controller (140 NOC 711 00 or TSX ETC 100). Offsets 0-7 correspond to Status[0].0 to Status[0].7, offsets 8-15 correspond to Status[1].0 to Status[1].7, and so on. If this bit is 1, then the connection is healthy. If this bit is 0, then the connection has been lost.

Quantum 140 NOC 771 01, Premium TSX ETC 101, and Modicon M340 BMX NOC 0401 Modules:

These status bits are located in the DDT input structure in a field called Status, which is an array of 16 BYTEs. To determine the correct bit to check for a given target device, you must locate the Connection Bit field for the device:

1. Start **Unity Pro**.
2. In the **DTM Browser**, right click on the Ethernet communication module (named 'NOC01' in this example) and click Open.
3. In the **NOC01 – fdtConfiguration** window, in the left-side tree, expand the **Device List** group, expand the target device (RMC) that you want to look at, and select the connection node named **Input/Output**, which will generally be the only connection listed. Then in the right part of the window, select the **Connection Settings** tab, and look at the **Connection**

Bit value, which should be in the range of 0..127:



The value of the **Connection Bit** is the bit offset from the start of the 16-byte HEALTH_BITS_IN [] array located in the input Derived Variable created for the EtherNet/IP communication module. For example, for a module named NOC01, offsets 0-7 correspond to NOC01_IN.HEALTH_BITS_IN[0].0 to NOC01_IN.HEALTH_BITS_IN[0].7, offsets 8-15 correspond to NOC01_IN.HEALTH_BITS_IN[1].0 to NOC01_IN.HEALTH_BITS_IN[1].7, and so on. If this bit is 1, then the connection is healthy. If this bit is 0, then the connection has been lost.

See Also

EtherNet/IP Overview | I/O Connection Status | I/O Connection PLC Status

---

6.7.10.6.5. Troubleshooting EtherNet/IP I/O

Using the Event Log

The Event Log is the primary troubleshooting tool in the RMC for EtherNet/IP I/O. It can record every change in the EtherNet/IP I/O data received (consumed) by the RMC. This is the Output Data from the PLC. It does not record the data produced by the RMC (the Input Data in the PLC).

The Event Log can report the following events based on Incoming I/O Data:

- **Initial Data**
  This entry is logged only if the Sync Register is used and the **Ethernet I/O Logging** filter option for the Event Log is set to **All**. It is reported when the first incoming I/O data is received after a new controlling I/O connection is established, showing both the data and the value of the Sync In register. Notice that the data is not yet applied to the RMC.
- **Data Changed**
  This entry is logged only if the Sync Register is used and the **Ethernet I/O Logging** filter option for the Event Log is set to **All**. It is reported when any Incoming I/O Data register has

changed, but the Sync In register has not. This entry shows both the data and the value of the Sync In register. Notice that the data is not yet applied to the RMC.

- **Request Made**
  This entry is logged each time the incoming I/O data is applied to the RMC. If the Sync Register is used, then this is each time the Sync In register changes. If the Sync Register is not used, then this occurs when the I/O data is first received, and each time after that when any register in the Incoming Data changes. This log entry will be shown by default, but can be disabled by setting the **Ethernet I/O Logging** filter option for the Event Log to **None**.

Using the Communications Statistics

To open the Communication Statistics window, in the Project pane, select the desired controller. On the **Controller** menu, choose **View Communication Statistics**.

The Communications Statistics window provides information on open EtherNet/IP I/O connections, such as the producing and consuming registers and the RPI, along with information on timed-out connections. The statistics also provide advanced information on the Ethernet traffic.

Incompatibilities with Older Allen-Bradley Ethernet Modules

The following problems have been seen when using the RMC controllers with Allen-Bradley Ethernet modules:

Problem #1:

I/O connections drop repeatedly resulting in error 0x0203 being reported in the Allen-Bradley Ethernet module

Cause

A number of Rockwell Automation Ethernet modules with older firmware are incompatible with the Quality-of-Service object in EtherNet/IP. The Quality-of-Service (QoS) object allows managed Ethernet switchgear to prioritize EtherNet/IP packets to provide higher levels of determinism. The QoS object was defined by ODVA (https://www.odva.org) to be backward compatible with devices that did not support the QoS object.

Some devices incorrectly handle the default QoS object behavior of marking I/O data with non-zero DSCP values, including some Rockwell Automation Ethernet modules with older firmware, causing the EtherNet/IP I/O connection to time out with error code 16#0203 shortly after it is established.

This does not apply to RMC75/150 firmware 3.40.1 or older, as that firmware did not support the QoS object.

Rockwell Automation is aware of the problem, and has a knowledge base article that discusses the issue (#63904 - https://rockwellautomation.custhelp.com/app/answers/detail/a_id/63904/) and suggests that users update the firmware in their Ethernet modules to fix the issue. The following firmware revisions fix the problem in the following Rockwell modules:

Rockwell Module	Firmware Revision
1788-ENB	2.004.1
L2xE/L3xE	V17
1756-ENBT	4.005.1
1756-EWEB	4.005.3
1768-ENBT	2.001.0
1768-EWEB	1.002.9
1794-AENT	4.001.1

If you have one of these modules with an older revision, then you should update the Rockwell Ethernet module firmware, if possible.

Solutions

There are two possible solutions:

1. **Update Rockwell Module firmware (recommended)**
   Update the firmware in the affected Rockwell Ethernet module.
2. **Disable Non-Zero DSCP Values in RMCTools**
   This is not necessary if you followed the recommended solution #1.
   In RMCTools, do the following:
   a. In the Project tree, expand the **Modules** folder and double-click the RMC CPU.
   b. On the **EtherNet/IP** page, in the **Quality of Service (QoS)** section, uncheck the **Enable DiffServ Code Point (DSCP) marking** box.
   c. Click **OK** to apply the changes to the RMC. Ensure that the settings are saved to flash and the module is restarted.

Problem #2:

I/O connections are lost approximately 2 minutes after they are established, are successfully re-established, but continue to be lost every two minutes.

Cause

RMC75/150 firmware 3.63.0 (February 2016) and newer, and RMC200 firmware, include an "Encapsulation Inactivity Timeout" parameter. This is an EtherNet/IP requirement, and must default to closing an inactive TCP/IP connection used by EtherNet/IP after 2 minutes.

Older Allen-Bradley 1756-ENBT (prior to 4.007) and 1756-EN2T (prior to 1.004) Ethernet bridge modules will incorrectly close any EtherNet/IP I/O connection that was opened using the TCP/IP connection. This is seen in the RMC event log by the CIP IO connection timing out or being closed nearly exactly when the EtherNet/IP inactivity timeout occurs, which is also reported in the Event Log.

Solutions

There are two possible solutions:

1. **Update Rockwell Module firmware (recommended)**
   Update the firmware in the affected Rockwell Ethernet module.
2. **Disable Encapsulation Inactivity Timeout**
   This is not necessary if you followed the recommended solution #1.
   In RMCTools, do the following:
   a. In the Project tree, expand the **Modules** folder and double-click the RMC CPU.
   b. On the **EtherNet/IP** page, in the **Encapsulation Inactivity Timeout** section, set the **Timeout** to 0. This disables the timeout.
   c. Click **OK** to apply the changes to the RMC. Ensure that the settings are saved to flash and the module is restarted.

See Also

EtherNet/IP Overview

6.7.10.6.6. Multiple Types of EtherNet/IP I/O Connections

This topic discusses having different types of EtherNet/IP I/O connections open with a single RMC, where one connection is Input/Output, and the others receive the same Input Data as that Input/Output connection. For details on multiple Input/Output connections on the RMC200, see the Setting up an EtherNet/IP I/O Connection topic.

An RMC can have different types of I/O connections open simultaneously with multiple EtherNet/IP clients such as PLCs and HMIs. For the RMC75/150, when using different types of connections, each

connection will share the same I/O data produced by the RMC, but only one client will send I/O data to be consumed by the RMC. Each simultaneous RMC75/150 I/O connection must have the same RPI and input data size, and all connections must use multicast for the RMC's produced data. The RMC200 supports unicast with multiple connections and supports differing data sizes and update data rates.

Using multiple types of I/O connections with an RMC requires that you understand the three types of connections supported by the RMC:

- **Input/Output**
  This connection is bi-directional: the originator (PLC or HMI) produces data consumed by the RMC and the target (RMC) produces data that is consumed by the originator. This connection type is also called an Exclusive Owner connection or the *controlling connection*.
- **Input Only**
  For this connection type, only the target (RMC) produces data, which is consumed by the originator (PLC or HMI). The originator will only send a heartbeat packet, sometimes at a reduced interval (less frequently than the RPI) used to allow the RMC to identify when the connection is broken.
- **Listen Only**
  This connection type is identical to an Input Only connection type, with a single exception: a Listen Only connection can only exist when one of the other I/O connection types has been established. That is, a Listen Only connection cannot be established until an Input/Output or Input Only connection has been established, and conversely when the last non-Listen-Only connection has been closed or has timed out, the Listen Only connection will automatically close as well. This connection type is the least-frequently used type.

The RMC75/150 support only one Input/Output connection. The RMC200 supports up to 3 Input/Output connections simultaneously.

Configuring Multiple Types of Connections

To establish multiple types of I/O connections with an RMC, you simply create I/O connections from each originator (PLC or HMI) to the RMC as you would normally. However, you must obey the following rules in order for all of the connections to be successful:

- RMC75/150 only: All connections must be set up to use multicast (not unicast) for target-to-originator (T->O) packets.
- For Multicast:
  - All connections must use the same RPI value. See the **Requested Packet Interval (RPI)** section of the Setting up an EtherNet/IP I/O Connection topic for details on the minimum RPI supported based on the number of open connections.
  - All connections must have the target-to-orginator (T->O) data size set the same.
- Number of Input/Output Connections:
  - RMC75/150: No more than one of the connections can be an Input/Output connection.
  - RMC200: Up to three connections can be Input/Output connections.
- Listen Only connections will not connect until at least one Input/Output or Input Only connection has been established.
- Each connection must be set up correctly for its desired connection type, as described in detail in Setting up an EtherNet/IP I/O Connection.

Using Multiple Types of Connections

When performing communications with multiple types of EtherNet/IP I/O connections, the Input/Output connection (or controlling connection) behaves as normal. The Input Only and Listen Only connections can only view the data from the RMC—they can't write to the RMC. If the Sync Register is being used, then the first register in the Input Data is the Sync register and is of no use for the Input Only devices.

See Also

EtherNet/IP Overview

6.7.10.6.7. EtherNet/IP Performance Considerations

There are several areas to consider with regard to EtherNet/IP Performance:

- General Recommendations
- Considerations for RMC EtherNet/IP Performance
- Considerations for Master EtherNet/IP Performance
- Considerations for Network Performance

General Recommendations

The following recommendations will improve the performance and determinism of the network in general.

- Always use unicast I/O connections instead of multicast whenever possible. Notice that Allen-Bradley products only supported multicast connections prior to RSLogix 5000 v18.
- Do not use a lower RPI than you need. Usually an RPI of 10 ms or 20 ms is sufficient. Lowering the RPI will increase the load on every Ethernet component in the system, and beyond a certain point has no benefit. For example, if the RPI is lower than the scan time of the controlling PLC, then additional packets will not benefit the system.
- Never use hubs or other half-duplex switchgear. This will introduce collisions, which greatly reduce the scalability and determinism of a network.
- When using a multicast connection, never connect the EtherNet/IP I/O network to a general purpose network, except through a router. The multicast traffic on the EtherNet/IP network will be noticed by your IT department. Conversely, glitches on the general purpose network could affect your factory network. If only unicast I/O connections are used, then connecting to a general purpose network through a switch may be acceptable in some applications.
- Read Rockwell Automation's **EtherNet/IP Performance** application guide (Publication ENET-AP001D-EN-P). It covers this subject in greater detail.

Considerations for RMC EtherNet/IP Performance

The RMC controllers support the following EtherNet/IP I/O bandwidth:

Controller	**Total Bandwidth** (packets/second)	**I/O Connections**[1]	**Minimum RPIs and Bandwidth**
RMC75E	1250	1	2 ms (1000 pps)
		2	3 ms (1000 pps)
		3-4	3-4 ms (1000-1250 pps)
RMC150E	1250	1	2 ms (1000 pps)
		2	3 ms (1000 pps)
		3-4	4 ms (1000-1250 pps)
RMC200 CPU20L	6000	1-3	1 ms (2000-6000 pps)
		4	4 at 2 ms (4000 pps), or 2 at 2 ms and 2 at 1 ms (6000 pps)
RMC200 CPU40	8000	1-4	1 ms (2000-8000 pps)

[1] The RMC75E and RMC150E only support multiple I/O connections in *multicast* mode. The RMC200 supports multiple I/O connections in *multicast* or *unicast* mode. The first I/O connection always

requires 2 packets per RPI. Subsequent multicast connections require 1 packet per RPI and subsequent unicast connection require 2 packets per RPI.

Considerations for Master EtherNet/IP Performance

The PLC EtherNet/IP I/O module must often support a large number of devices, including RMCs and other devices. Therefore, it is important to recognize the limits in bandwidth of your EtherNet/IP master controller, and to know how to determine the amount of bandwidth being utilized. This entire subject is covered in more detail in Rockwell Automation's **EtherNet/IP Performance** application guide (Publication ENET-AP001D-EN-P), but will be addressed narrowly here for RMC connections.

Here are some published bandwidth limits for EtherNet/IP I/O controllers available as of this writing:

Manufacturer	**Module**	**Total Bandwidth** (packets/second)
Allen-Bradley	1756-ENET/B (obsolete)	900
Allen-Bradley	1756-ENBT	5000
Allen-Bradley	1756-EN2T, 1756-EN2TR, 1756-EN2F, 1756-EN2TXT, 1756-EN3TR,	10000
Allen-Bradley	1768-ENBT	5000
Allen-Bradley	1769-L23E	2000
Allen-Bradley	1769-L3xE	4000
Allen-Bradley	1788-ENBT	4000
Schneider Electric	Quantum 140 NOC 711 00	7500
Schneider Electric	Premium TSX ETC 100	7500
Omron	CS1W-EIP21	6000
Omron	CJ1W-EIP21	6000
Omron	CJ2H-CPUxx-EIP	6000
Omron	CJ2M-CPU3x	3000

Notice that it is recommended that not more than 90% of this bandwidth be used on I/O connections.

The bandwidth required for each RMC depends on the RPI and the number of connections. The first connection requires 2 / RPI packets per second (where RPI is in seconds), and each additional connection requires 1 / RPI additional packets per second. The following chart summarizes the requirement:

Number of Connections	**Multicast Required for Bandwidth** (packets/second)	**Required Bandwidth for Unicast (RMC200 Only)** (packets/second)
1	2 / RPI	2 / RPI
2	3 / RPI	4 / RPI
3	4 / RPI	6 / RPI
4	5 / RPI	8 / RPI

Therefore, to determine the total bandwidth required for any group of RMCs being serviced by a single master EtherNet/IP I/O controller, simply add up the bandwidth requirement of each RMC connection. Usually the RPI for all RMCs will be set the same, so the total utilization can be calculated as shown:

Total Required Bandwidth = (# of RMC connections) x (2 / RPI)

Example

For 12 RMCs with an RPI of 20 ms each and one connection each, the total required bandwidth is 12 x (2 / 0.020), which is 1200 packets/second.

Using this formula, we can also calculate the maximum number of RMCs that can be supported by your controlling EtherNet/IP module:

Maximum Supported RMCs = Total Bandwidth x 90% / (2 / RPI)

Example

How may RMC connections can the 1756-EN2T support at a 10 ms RPI? The Total Bandwidth for the 1756-EN2T is 10000 packets/second, so the maximum number of RMC connections is found by 10000 x 90% / (2 / 0.010), which is 45 RMCs.

If the total bandwidth required exceeds 90% of the master EtherNet/IP I/O controller's bandwidth, then you will have to consider one of the following options:

- **Increase the RPI for one or more of the RMC connections**
  Increasing the RPI reduces the required bandwidth. For example, doubling the RPI will cut the bandwidth in half. Notice that in many cases, increasing the RPI comes at no cost to the system performance because the PLC may be unable to scan its ladder logic as frequently as the RPI.
- **Replace the EtherNet/IP controller with a faster model**
  If an EtherNet/IP controller with higher bandwidth is available for the same platform, then you may be able to upgrade the EtherNet/IP controller. For example, on the ControlLogix family, the 1756-ENET/B can be replaced by the 1756-ENBT or 1756-EN2T, or the 1756-ENBT can be replaced by the 1756-EN2T.
- **Use multiple EtherNet/IP modules to divide the network**
  By adding one or more additional EtherNet/IP communication modules to the PLC, each with its own isolated network with a smaller number of RMCs, the bandwidth requirement on each device is reduced. Notice that it is important that the individual networks are not connected to one another directly or through a switch, as this will largely defeat the benefits of dividing the network. It is acceptable to connect the networks using routers. However, it is important that someone with IT experience is consulted before doing so.

Example

Suppose one ControlLogix 1756-ENBT module will be controlling twenty-five (25) RMC connections. The intended RPI is 10.0 ms. Therefore the total bandwidth in packets/second is computed as follows:

Packets/Second	=	Number of RMC connections x (2 / RPI)
	=	25 x (2 / 0.010s)
	=	5000

The maximum allowable I/O load on the ENBT is 90% of 5000, which is 4500. Since 5000 is greater than 4500 packets/second, one of the following alternatives must be considered:

- Increase the RPI of one or more of the RMCs until the bandwidth is below 4500. Raising each to 12.0 ms or higher does this.
- Replace the 1756-ENBT with the 1756-EN2T. The bandwidth on the 1756-EN2T is 10,000 packets/second, which easily handles this load.

- Use two 1756-ENBT modules and divide the RMC connections between the two. For example, one could control 15 RMC connections (3000 packets/second), and the second could control 10 RMC connections (2000 packets/second), putting both within the recommended limits.

Considerations for Network Performance

The maximum amount of information that 100 Mbps Ethernet can carry in one direction is 100 million bits per second. If all packets on the network were EtherNet/IP I/O packets of the largest size supported by the RMC (125 four-byte REALs), then this would be a maximum of 21,600 packets/second (notice that packet overhead such as headers and inter-frame spacing are included in this estimate). The network utilization should be kept well below this maximum. As the utilization approaches the maximum, it becomes more and more likely that a packet will have to wait to get onto the wire, introducing small delays.

In addition to the limits of the wire itself, many devices have a maximum sustained rate that they can receive packets without falling behind in packet processing and eventually dropping packets. The RMC is one such device, since priority must be given to controlling motion above handling packets. No pre-determined maximum rate is currently available, but by reviewing the Ethernet Statistics in the devices, you can look to see if packets are being discarded.

Notice that EtherNet/IP multicast I/O connections can greatly impact the performance of an Ethernet network since—unlike unicast packets, which are sent directly to a single device—multicast packets are sent to all devices on the network by default. This increases the load on every device, including the switches, and each individual network segment.

If your network has a high utilization and/or network components are dropping packets due to high packet rates, you will want to look at ways of reducing the network utilization. There are three ways to reduce the network utilization:

- **Change multicast I/O connections to Unicast**
  As described above, multicast I/O connections can significantly increase the network load. Most applications do not require multicast I/O connections, but unfortunately because Rockwell did not support unicast I/O connections until RSLogix 5000 v18, most current EtherNet/IP installations use multicast. If you are using a EtherNet/IP controller that supports unicast and are not using multiple I/O connections per RMC, then you should make sure that all I/O connections are set up as unicast.
- **Increase the RPI for one or more EtherNet/IP I/O connections**
  The simplest way to reduce network utilization is to increase the RPI of EtherNet/IP I/O connections. Of course, this is only a possibility if the RPIs are unnecessarily low. This topic has been discussed above, so no more needs to be said here.
- **Use multiple EtherNet/IP modules to divide the network**
  By adding one or more additional EtherNet/IP communication modules to the PLC, each with its own isolated network with a smaller number of RMCs, the utilization on each sub-network is reduced. Notice that if multicast I/O connections are used, then it is important that the individual networks are not connected to one another directly or through a switch, as this will largely defeat the benefits of dividing the network. It is acceptable to connect the networks using routers. However, it is important that someone with IT experience is consulted before doing so.
- **If using multicast I/O connections, use smarter switchgear**
  As noted above, currently EtherNet/IP I/O connections generate a lot of multicast Ethernet traffic. This multicast traffic is treated like broadcast traffic by default, which means that every port on every switch in the network will send out a copy of each multicast packet, even though in most cases only one device on one port is interested in the packet. However, it is possible to be more efficient than this default behavior by using smarter switchgear to direct the multicast traffic only to those ports that are listening to the multicast traffic.

  The current way of doing this uses a protocol called IGMP (Internet Group Management Protocol). Using IGMP to direct packets in switches to only the interested ports requires that (1) the switch supports IGMP Snooping, and (2) at least one switch or router connected to the network supports IGMP Query. Notice that some switches are now available that provide both IGMP Snooping and IGMP Query in the same switch.

See Also

EtherNet/IP Overview

6.7.10.6.8. Using EtherNet/IP Explicit Messaging

EtherNet/IP explicit messaging allows the originator (PLC or HMI) to request individual services from the target device (RMC). These requests are made explicitly rather than being scheduled cyclically like I/O. Explicit messaging is much more flexible than I/O in terms of what data or services are accessed in the target device, since I/O connections must pre-configure the I/O data to be exchanged.

In most cases, RMC users use EtherNet/IP explicit messaging to read and write registers in the RMC. The RMC provides two methods for doing so. Also, some advanced EtherNet/IP users may want to access standard CIP services and attributes. The following sections describe how to use EtherNet/IP explicit messaging in each of these cases.

Reading and Writing RMC Registers using the Allen-Bradley PCCC/DF1 Services

When using an Allen-Bradley PLC with EtherNet/IP, the PLC's MSG (message) block is used to read and write registers in the RMC. See Using Allen-Bradley Controllers via Message Block for details. Requests made using this method use Allen-Bradley file addressing, which is described in the DF1 Addressing topic.

Other PLCs or HMIs that can read or write from Allen-Bradley file addresses (e.g. F7:0, L8:16) may also be able to use this method. The underlying object in the RMC is the Allen-Bradley PCCC/DF1 Object. This object provides services that encapsulate the PCCC/DF1 commands and functions. The RMC implements the following PCCC function codes:

- Diagnostic Status (06 03)
- Echo (06 00)
- SLC Protected Typed Logical Read with 2 Address Fields (0F A1)
- SLC Protected Typed Logical Read with 3 Address Fields (0F A2)
- SLC Protected Typed Logical Write with 2 Address Fields (0F A9)
- SLC Protected Typed Logical Write with 3 Address Fields (0F AA)
- PLC5 Typed Read (0F 68)
- PLC5 Typed Write (0F 67)
- PLC5 Word Range Read (0F 01)
- PLC5 Word Range Write (0F 00)

Reading and Writing RMC Registers using the Register Map Object

For PLCs that support explicit messaging, but do not support the PCCC/DF1 services above, the RMC's Register Map Object can be used to read and write registers. The Register Map Object provides relatively simple services for reading and writing registers. The PLC must build and issue a CIP Service Request and then extract the results from the CIP Service Response. The method for issuing and receiving CIP service requests and responses are PLC-specific.

To invoke a CIP service, you must know the object class, object instance, service ID, and format of the request and response data. For all services in the Register Map Object, the object class and instance will be as follows:

Field	Value
Object Class	0xC0 (192)
Object Instance	0x01 (1)

The Register Map Object supports the following services:

Service ID	Service Name	Description
0x4B (75)	Read (LSB First)	Read one or more registers from the RMC. All multi-byte values are encoded least-significant byte (LSB) first.
0x4C (76)	Write (LSB First)	Writes one or more registers to the RMC. All multi-byte values are encoded least-significant byte (LSB) first.
0x4D (77)	Read (MSB First)	Read one or more registers from the RMC. All multi-byte values are encoded most-significant byte (MSB) first.
0x4E (78)	Write (MSB First)	Writes one or more registers to the RMC. All multi-byte values are encoded most-significant byte (MSB) first.

Notice that there are really only two services: read and write. However, because there are two standards for encoding multi-byte data, both LSB-first and MSB-first versions of each service are provided.

The following sections describe the format of the CIP requests and responses for the service types. Notice that the first six bytes of the requests and the first four bytes of the response in the charts that follow make up the standard CIP message routing header. Therefore your PLC or HMI may already build this part of the packet for you.

Read (LSB or MSB First) Request:

Offset	Type[1]	Size	Field Name	Description
0	USINT	1	Service	Must be 0x4B (LSB First) or 0x4D (MSB First).
1	USINT	1	Path Size	Must be 0x02.
2	USINT[4]	4	Path	Must be 0x20 0xC0 0x24 0x01.
6	UINT	2	File	Register file in the RMC to read.
8	UINT	2	Element	First element to read from the specified file.
10	UINT	2	Count	Number of DINTs or REALs to read.

[1] The byte order of multi-byte fields is determined by the Read service selected. For service 0x4B (Read LSB First), the least-significant byte must be sent first. For service 0x4D (Read MSB First), the most-significant byte must be sent first.

Read (LSB or MSB First) Response:

Offset	Type[1]	Size	Field Name	Description
0	USINT	1	Reply Service	Will be 0xCB (LSB First) or 0xCD (MSB First).
1	USINT	1	Reserved	Will be 0
2	USINT	1	General Status	See below. Zero (0) means success.
3	USINT	1	Additional Status Size	Will be 0
4	DINT or REAL	4	Register0	Value of first register
8	DINT or REAL	4	Register1	Value of second register
:	:	:	:	:
4+4xN	DINT or REAL	4	RegisterN	Value of Nth register

[1] The byte order of multi-byte fields is determined by the Read service selected. For service 0x4B (Read LSB First), the least-significant byte will be sent first. For service 0x4D (Read MSB First), the most-significant byte will be sent first.

Write (LSB or MSB First) Request:

Offset	Type[1]	Size	Field Name	Description
0	USINT	1	Service	Must be 0x4C (LSB First) or 0x4E (MSB First).
1	USINT	1	Path Size	Must be 0x02.
2	USINT[4]	4	Path	Must be 0x20 0xC0 0x24 0x01.
6	UINT	2	File	Register file in the RMC to write to.
8	UINT	2	Element	First element in the specified file to write to.
10	UINT	2	Count	Number of DINTs or REALs to write.
12	DINT or REAL	4	Register0	Value of first register
16	DINT or REAL	4	Register1	Value of second register
:	:	:	:	:
12+4xN	DINT or REAL	4	RegisterN	Value of Nth register

[1] The byte order of multi-byte fields is determined by the Write service selected. For service 0x4C (Write LSB First), the least-significant byte must be sent first. For service 0x4E (Write MSB First), the most-significant byte must be sent first.

Write (LSB or MSB First) Response:

Offset	Type	Size	Field Name	Description
0	USINT	1	Reply Service	Will be 0xCC (LSB First) or 0xCE (MSB First).
1	USINT	1	Reserved	Will be 0
2	USINT	1	General Status	See below. Zero (0) means success.
3	USINT	1	Additional Status Size	Will be 0

In the CIP service response, the General Status field can hold one of several values:

Gen Status	Description
0x00 (0)	**Success.** The request succeeded.
0x03 (3)	**Invalid service data.** This error will occur if the File or Element values are invalid, or for a write, if the length of the request does not match the expected length based on the Count value.
0x11 (17)	**Reply data too large.** This error will occur if the response generated is too large to be sent. If you receive this error, you must decrease the Count value for a Read request. The maximum value varies based on the PLC, but is generally around 120 registers.
0x13 (19)	**Request length too small.** This error will occur on a Read or Write request if the length of the request is smaller than expected.

0x15 (21)	**Request length too large.** This error will occur on a Read request if the length of the request is larger than expected.

Accessing Standard CIP Services and Attributes

The RMC supports a number of standard CIP objects. These objects have various services and attributes that can be accessed through CIP Service requests. The RMC Statement of Conformance documents—available for download on Delta's web site (https://deltamotion.com)—list the objects, services, and attributes supported in the these controllers. Advanced EtherNet/IP users can use these documents, together with the EtherNet/IP specification and PLC documentation to access services in the supported objects.

See Also

EtherNet/IP Overview

6.7.10.7. PROFINET

6.7.10.7.1. PROFINET Overview

PROFINET IO is an Ethernet-based standard for connecting distributed I/O with controllers for fast data exchange, parameterization, and diagnostics. The bus cycle times for data exchange are down in the millisecond range. Configuring a PROFINET IO system has the same look and feel as in PROFIBUS.

The RMC75E, RMC150E, and RMC200 controllers support PROFINET IO as an IO-Device. Therefore, an IO-Controller must be configured to establish cyclic I/O data connections with the RMC and to initiate acyclic data record transactions:

- **Cyclic I/O data** is automatically exchanged between the PLC and RMC at the specified update time. For example, status information from the RMC, and variables to be written to the RMC would typically be part of the cyclic data. Cyclic I/O data is defined by the Incoming and Outgoing Cyclic I/O Data as described in Setting up a PROFINET I/O Connection.
  The RMC75E supports up to 128 registers in each direction, and the RMC150 and RMC200 support up to 256 registers in each direction. Notice that some PLCs may be limited to fewer than the full sized supported by the RMC controllers.
- **Acyclic data transactions** are performed only when needed and typically for infrequent or large data transfers. For example, if the PLC creates a curve and sends it to the RMC, that data would typically be sent via the acyclic data. Or, if the PLC needs to read a captured plot from the RMC, that is also best done via the acyclic data. Acyclic I/O data is defined by the Record Data. The RMC75E and RMC150E support acyclic transfer of up to 2048 32-bit registers (8192 bytes) in each direction. The RMC200 supports acyclic transfer of up to 4096 32-bit registers (16,384 bytes) in each direction.

The RMC supports only one type of controlling IO connection at a time. Therefore, PROFINET cannot be used simultaneously with an EtherNet/IP IO connection. Other protocols can be used simultaneously with PROFINET.

Further Information

For further information on setting up and using PROFINET IO with RMC controllers, refer to the following topics:

- Setting up a PROFINET I/O Connection
- Using a PROFINET I/O Connection
- Using PROFINET Record Data
- Troubleshooting PROFINET

RMC PROFINET IO Specifications

The following chart summarizes the PROFINET IO specifications for the RMC products:

Item	RMC75E	RMC150E	RMC200 CPU20L	RMC200 CPU40
Supported Update Times	2, 4, 8, 16, 32, 64, 128, 256, 512ms	2, 4, 8, 16, 32, 64, 128, 256, 512ms	1, 2, 4, 8, 16, 32, 64, 128, 256, 512 ms	1, 2, 4, 8, 16, 32, 64, 128, 256, 512 ms
Minimum Input Data	32 bytes (8 registers)			
Maximum Input Data	512 bytes (128 registers)	1024 bytes (256 registers)	1024 bytes (256 registers)	1024 bytes (256 registers)
Minimum Output Data	32 bytes (8 registers)			
Maximum Output Data	512 bytes (128 registers)	1024 bytes (256 registers)	1024 bytes (256 registers)	1024 bytes (256 registers)
Data Consistency	All items			
Vendor ID	0x0192			
Vendor Name	Delta Computer Systems, Inc.			
Device ID	0x0002	0x0003	0x0004	0x0004
Device Name	RMC75E	RMC150E	RMC200 CPU20L	RMC200 CPU40
Certificate No.	Z10245	Z10244	Z12911	Z12353
Node Type	IO-Device			
Device Family	I/O, Delta Motion Controllers			
PNIO Version	V2.2	V2.2	V2.4	V2.34
Conformance Class	A	A	B	B
Netload Class	III*	III*	III	III
Media Class (Type)	Wired (100BaseTX, Full Duplex)			
Redundancy Class	--		MRP Client	MRP Client
Communication Class	RT_CLASS_1			
Application Relations	one IO AR, one Device Access AR, implicit AR			
Communication Relations	2 IO CR: one input, one output 1 Alarm CR 1 Record Data CR			

*Not tested as part of the certificate

See Also

Setting up a PROFINET I/O Connection | Using a PROFINET I/O Connection | Using PROFINET Record Data |Handling Broken PROFINET Connections | Troubleshooting PROFINET | Using Siemens S7 PLCs via PROFINET

6.7.10.7.2. Setting up a PROFINET I/O Connection

This topic describes the steps generally required to set up a PROFINET I/O connection with the RMC. For a step-by-step procedure for doing these steps on a Siemens S7 PLC, see the Using Siemens S7 PLCs via PROFINET topic. For details on controlling the RMC once a connection has been made, see the Using a PROFINET I/O Connection topic.

Setting up a PROFINET I/O connection requires steps in RMCTools and in the IO-Controller's setup software (such as TIA Portal for Siemens S7 PLCs).

Cyclic and Acyclic Data

The PROFINET IO connection can be configured for both cyclic and acyclic I/O data. Understanding these methods of data transfer will help you determine how you wish to configure the PROFINET connection.

- **Cyclic I/O data** is always sent between the PLC and RMC at the specified update time. For example, status information from the RMC, and variables to be written to the RMC would typically be part of the cyclic data. Cyclic I/O data is defined by the Incoming and Outgoing Cyclic I/O Data.
- **Acyclic data** is sent only when it is needed. It is useful for infrequent data transfers, or for very large data transfers. For example, if the PLC creates a curve and sends it to the PLC, that data would typically be sent via the acyclic data. Or, if the PLC needs to read a captured plot from the RMC, that is also best done via the acyclic data. Acyclic I/O data is defined by the Data Records. The maximum length is 2048 registers (8192 bytes).

PROFINET Settings in RMCTools

Selecting PROFINET Protocol Mode (RMC200 Only)

In order to communicate via PROFINET I/O, the RMC200 must be set to **PROFINET Mode**:

1. In the Project pane, expand the RMC, and double-click the **CPU**.
2. On the **Ethernet** page, in the **Ethernet Protocol Mode** section, click **PROFINET Mode**, then click **OK**.

The Ethernet Protocol Mode may also be viewed on the RMC200 Display Screen.

Setting the PROFINET Device Name

The primary identifier for PROFINET nodes is the device name. To set the device name for the RMC using RMCTools, use the PROFINET Settings page.

Setting up the Outgoing and Incoming Cyclic I/O Data Areas

Each PROFINET I/O connection will cyclically transfer Input data and Output data at the specified update time. The location of the Outgoing Cyclic I/O Data, sent from the RMC to PLC, and the location where the Incoming Cyclic I/O Data sent from the PLC to the RMC is stored must be set in RMCTools using the PROFINET Settings page.

Notice that only the starting address for each data area is specified in RMCTools. The length of each data block is determined by which input and output module is selected in the IO-Controller's PROFINET setup. See Selecting the Input and Output Modules below for details.

For almost all applications, the starting location for the Outgoing Cyclic I/O Data is left at its default of the start of the Indirect Data Map area. For most applications, the starting location for where the Incoming Cyclic I/O Data is stored is also left at its default location in a separate Indirect Data Map area, but the Variable Table and Command Area are also plausible destinations.

Selecting the Synchronization Method

The user has the option of including or not including a Sync Register at the beginning of the Incoming and Outgoing I/O Data. This option is selected in RMCTools on the PROFINET Settings page. For a description of the benefits of each method, see the Using a PROFINET I/O Connection topic.

Configuring Data Records

The configuration of Data Records does not need to be completed in order to establish a PROFINET I/O connection. Data Records apply to acyclic read and writes and are described fully in the Using PROFINET Record Data topic.

Setting the Byte Order

The RMC supports both Most Significant Byte (MSB) First and Least Significant Byte (LSB) First. The default is MSB First, which should be correct for most PROFINET controllers, including Siemens S7 PLCs.

Setting the IP Address from RMCTools

PROFINET supports setting the IP address from the PROFINET controller or configuration tool. However, it is recommended that RMCTools be used to set the IP address in order to ensure that the RMC is always accessible from RMCTools. To set the IP address for the RMC, use the RMC Ethernet Settings Page.

PROFINET Settings in the PROFINET Configuration Software

Once the PROFINET Device Name, Outgoing and Incoming Cyclic I/O Data areas, and Synchronization Method have been set in RMCTools, the rest of the PROFINET configuration is done in the PROFINET configuration software, which is specific to the IO-Controller being used. For Siemens S7 controllers, the Devices and Networks tool within Siemens TIA Portal is typically used.

Installing the GSD File

Before an RMC device can be added to the PROFINET configuration, the GSD files for the RMC must be installed in the PROFINET configuration software. The PROFINET GSD files for the RMC controllers are available for download from Delta's website, and are installed in a PROFINET subfolder under the RMCTools install folder, for example C:\Program Files\RMCTools\PROFINET.

Delta provides GSDML files for each product:

- GSDML-V*n*.*n*-Delta-RMC75E-*yyyymmdd*.xml
- GSDML-V*n*.*n*-Delta-RMC150E-*yyyymmdd*.xml
- GSDML-V*n*.*n*-Delta-RMC200-*yyyymmdd*.xml

The most recent GSD versions are included with the RMCTools installation. Contact Delta for older versions if required for use with older PROFINET configuration software.

Notice that each of the RMCs has an associated image file. These should be the in same folder as the GSD file when it is installed into the PROFINET configuration software.

Selecting the Device Access Point (DAP)

After the GSD files have been installed into the PROFINET configuration software, the RMC can be added to the PROFINET network. This generally involves browsing through the device library to find the desired device and selecting the Device Access Point (DAP) to use:

- RMC75E V1.0
- RMC150E V1.0
- RMC200 CPU20L V1.0
- RMC200 CPU40 V1.0

To add the RMC to the PROFINET network, the user generally drags the desired DAP onto the PROFINET network.

Using the PROFINET Flash LED Feature

PROFINET allows the configuration software to flash an LED on a node to help identify which nodes correspond to which physical device:

- **RMC150E:** the CPU LED will flash when instructed to do so by PROFINET.
- **RMC75E:** the Controller LED will flash. Notice that the Controller LED will also flash when in PROGRAM mode. Therefore, you may need to set the RMC75E to RUN mode before verifying a device using the PROFINET flash LED feature.
- **RMC200:** all LEDs on the CPU will flash in unison.

Setting the Device Name

Each IO-Device added to a PROFINET configuration must have a Device Name. Set the Device Name for the RMC in the PROFINET configuration software to match the Device Name set in RMCTools.

Notice that many PROFINET configuration software packages can also set the Device Name, which the RMC does support. However, it is generally easiest to set it directly in RMCTools so that the Device Name can also be saved in the RMCTools project file.

Setting the IP Address

PROFINET allows several methods of allocating the IP address in IO-Devices. The traditional method is to set the IP address for the IO-Devices when configuring the PROFINET network, and then the IO-Controller will find the IO-Devices by device name and configure the IP addresses as part of connecting to them. This method is not recommended because it can result in the IP address being changed or cleared at times when RMCTools is connected to the RMC.

An alternative method is to set the IP address settings in RMCTools, and then instruct the IO-Controller through the PROFINET Configuration Software to leave the IP address as it is when connecting to the IO-Device. For example, Siemens TIA Portal V14 includes an **IP address is set directly at the device** checkbox in the IO device's **Ethernet Addresses** properties that should be checked to choose this option. This is the recommended method for RMC controllers.

Selecting the Input and Output Modules

The RMC PROFINET devices are set up to have 3 virtual slots. The first slot is the DAP, and will be named RMC150E, RMC75E, RMC200 CPU20L, or RMC200 CPU40. The second slot is the Input Slot, and the third slot is the Output Slot. In the PROFINET IO device library, under the selected DAP, you will find a folder of input modules and a folder of output modules. Select exactly one Input module and one Output module to install in slots 1 and 2 respectively.

The Input and Output modules to choose from differ only in the number of registers to be transferred. The Input module corresponds to the Outgoing Cyclic I/O Data area in the RMC, and the Output module corresponds to the Incoming Cyclic I/O Data area. Notice that if the Sync Register is used, the first register in each block will be reserved as the Sync Register.

Setting the Update Time

One additional item configured in the PROFINET configuration software that affects the PROFINET communication with the RMC is the Update Time. The RMC75E and RMC150E controllers support update times of 2, 4, 8, 16, 32, 64, 128, 256, and 512 ms. The RMC200 supports these same update times, plus 1 ms. Notice that the update time cannot be set lower than the Loop Time of the RMC. Therefore, if a 4 ms loop time is used, then the minimum PROFINET update time will be 4 ms. A typical update time is 16 msec. Choosing a faster update time than necessary may use excessive Ethernet bandwidth.

Related to the Update Time is the Watchdog Time. This is specified as a multiple of the Update Time. For example, with a Watchdog Time multiplier of 3 and an Update Time of 16 ms, the Watchdog Time will be 48 ms. Most IO controllers fix this value at 3, but some may allow configuration of this value.

See Also

PROFINET Overview | Using a PROFINET I/O Connection | Using Siemens S7 PLCs via PROFINET

6.7.10.7.3. Using a PROFINET I/O Connection

This topic describes the how to control the RMC via the PROFINET I/O connection's *cyclic* data. For detail using the *acyclic* data, see Using PROFINET Record Data. For details on setting up a connection, see the Setting up a PROFINET I/O Connection topic.

Understanding the Sync Register

The cyclic I/O data can operate in two modes: *with* a Sync Register and *without* a Sync Register. When using a Sync Register, the first Input Data register and first Output Data register are each reserved as a Sync Register, and the RMC processes incoming data only when the Sync Register changes. When not using a Sync Register, the RMC processes incoming data each time any register in the block changes.

The Sync Registers show up in the Input and Output Data images in the PLC. In the RMC, they are only visible in the Event Log. See Troubleshooting PROFINET for details.

Why Use a Sync Register?

The Sync Register gives the user tight control over synchronization between the PLC and RMC. In applications that do not require this level of synchronization, the user can choose to not include the Sync Register.

The synchronization provided by the Sync Register is useful when:

- **Treating writes to a block of registers as a consistent block.**
  Not all PLCs synchronize their I/O with the PLC scan. Therefore, while data is being placed in the Output Data by the PLC program, a copy of the Output Data could prematurely be sent to the RMC, mixing some old data with some new data. When the Sync Register is used, this problem is avoided by the simple convention of having the PLC program update the Sync Register after all other registers have been set to the desired values. The RMC is then guaranteed to receive all the data in a single block.
- **Coordinating when Input Data has been updated to reflect an issued command.**
  This benefit is best demonstrated with an example. Suppose that an axis is holding position with its **In Position** status bit set. If the PLC issues a new move command to a new position, then we will see that the **In Position** bit will go off when the command is first processed and remain off until the new move completes and the axis is at the new position. How then does the PLC know when its copy of the **In Position** bit received in the Input Data reflects the new command having been received? When the Sync Register is used, the PLC can simply wait until the Sync Register coming from the RMC matches the one it sent to the RMC when issuing the command, and it can then safely examine the **In Position** bit.

Setting the Sync Register Mode

To select whether the Sync Register is used:

1. In the Project pane, expand the **Modules** folder, double-click the CPU module, and choose **PROFINET**.
2. Under Sync Register, select either **Use a Sync Register** or **Do not use a Sync Register**.

Using Input Data in the PLC

If the Sync Register is used, then the first register is reserved as the Sync Register. All other registers in the Input Data are controlled by the Outgoing Cyclic I/O Data address specified in the PROFINET Settings Page. When the I/O connection is established, the Input Data is automatically updated in the PLC at the specified PROFINET update time. The PLC can simply use the data whenever it needs it. The RMC does not wait for the Sync Register to change to update its Outgoing Cyclic I/O Data.

Notice that the data types of the variables in the RMC must match the data types of the structure receiving the Input Data from the RMC. Specifically, registers of type REAL in the RMC must correspond to REAL data types in the PLC, and all other RMC data types (DINT and DWORD) should correspond to tags of type DINT or DWORD in the PLC.

Writing Output Data with a Sync Register from the PLC

To write data in the RMC controller using I/O data when the Sync Register is used, do the following in the PLC:

1. Wait for the Sync Registers in the Input Data and Output Data to match.
2. Update any Output Data register values, except the Sync Register.
3. Change the value of the Output Data Sync Register.
4. Do not change any Output Data registers until the Input and Output Sync Registers match again.

The RMC will apply the Output Data register contents only once.

Writing Output Data without a Sync Register from the PLC

To write data in the RMC controller using I/O data without the Sync Register, simply change the Output Data registers in the PLC and they will get applied in the RMC. Often care must be taken as to the order that the registers are updated to ensure that actions triggered by changes to registers in the RMC have updated values for any parameters they use. It is often necessary to also understand whether the I/O data is updated synchronously to the PLC scan or not.

Writing to the Command Area from the PLC

The Incoming Cyclic I/O Data area can be set up to point to the Command Area registers directly, or indirectly through the Indirect Data Map. The Sync Register should always be used if the PLC will be writing commands to the Command Area. The following sequence is recommended:

1. **Wait Until the Sync In and Sync Out Registers Match**
   If they do not match, then this means that another command or set of commands is in progress.
2. **Clear Old Commands from the Command Registers**
   Clear old commands from the command registers for each axis. Otherwise, when the Sync Out register is changed, the commands would be re-issued. One method of clearing the old commands is to fill the Output Data array with zeroes.
3. **Write to the Command Registers**
   Write the Command registers and all required command parameters to the Output Data for all commands you want to issue. You can issue up to one command per axis. Leave the Command register set to 0 for each axis that will not receive a command.
4. **Change the Sync Out Register**
   The easiest way to do this is to add one to it. However, you must take care to handle overflowing this register (the Sync register is a REAL). One method is to add one and then MOD it with some large number, such as 10000. This will make the register count from 0 to 9,999, and then wrap back down to 0 without an error. Take care to ensure that you only update the Sync Out register once so that the commands do not get re-issued.
5. **Wait Until the Sync In and Sync Out Registers Match**
   It is important to wait until this occurs before using the status bits in the Input Data (if the Input Data includes any status bits). See the example above for how problems can occur if this step is ignored.

Triggering Commands through the Variable Table

If you have set the Output Data to write to Variable Table registers directly or via the Indirect Data Map, you can use them together with the Program Triggers to start User Programs that issue commands. For advanced applications that require issuing commands and writing to variables, this can be a very efficient method of communication.

To issue a command with this method, you will first need to do the following:

1. Create a user program that contains the commands you need.
2. Define a variable in the Variable Table that will be used to start the program. Name it StartProgram, for example.
3. Create a Program Triggers condition that looks at the value of StartProgram. When the StartProgram variable becomes a certain value, it will start the specified user program. This StartProgram variable can be used to start different programs based on its value.
4. The user programs should always reset the StartProgram variable to some starting value (such as 0). This is so the Program Triggers will always see the number change when the PLC writes to it. If the number already was 6, and the PLC writes 6, the value doesn't change, and the Program Triggers do not know to start a user program. Typically, the first step in a program should reset the value of the variable.

Once the Program Triggers and User Programs have been made, make sure the RMC is in RUN Mode, and use one of the procedures below to start a User Program:

With a Sync Register:

1. Wait until the Sync In and Sync Out Registers match.
2. Write a number to the StartProgram variable. The number will specify which user program to run.
3. If you need to write to other variables (e.g. for defining speeds, setpoints, etc.), do so.
4. Change the Sync Out Register.
   The easiest way to do this is to add one to it. However, you must take care to handle

overflowing this register (the Sync register is a REAL). One method is to add one and then MOD it with some large number, such as 10,000. This will make the register count from 0 to 9,999, and then wrap back down to 0 without an error. Take care to ensure that you only update the Sync Out Register once so that the commands do not get re-issued.

5. Wait until the Sync In and Sync Out Registers match.
   When the Sync In Register has changed, you know the data has been applied to the RMC. It is important to wait until this occurs before using the status bits in the Input Data (if the Input Data includes any status bits). See the example above for how problems can occur if this step is ignored.
   After the Sync registers match, you can look at the Task Status and Current Program registers to determine if the task is running the program, and when it stops. These registers should be included in the Input Data.

Without a Sync Register

1. If you need to write to other variables that are used by the user program (e.g. for defining speeds, setpoints, etc.), do so first.
2. Write a number to the StartProgram variable. The number will specify which user program to run.

Handling Broken Connections

Many applications require some type of action if the PROFINET I/O connection breaks during machine operation. For information on how to handle a broken connection in either the RMC or PLC, see the Handling Broken PROFINET Connections topic.

Troubleshooting

See the Troubleshooting PROFINET topic.

How the I/O Data is Applied to the RMC

When the Sync register is used, and the RMC sees that the incoming Sync Register changes, the RMC applies all the data from the PLC to the specified Incoming Cyclic I/O Data area. When the Sync Register is not used, then when any register changes its value, then all data is written to the specified Incoming Cyclic I/O Data area.

In certain cases, it may take more than one loop time to apply the data. Therefore, not all the data may be valid at the same time. For example, if you are using any of the data to trigger a user program that also uses some of the data, it is important that the user program is not triggered before *all* the data is valid, otherwise, the user program may use old data.

Data Consistency

When all the incoming data is applied in the same RMC loop time, the data is called **consistent**. If the data is consistent, you can use all the data immediately. If the data is not consistent, you need to make sure that you do not use the data until it has all been updated.

The following rules apply to data consistency based on where the consuming address is in the RMC:

Variable Table - Current Values

- **1 ms or longer loop time:** Up to 256 variables can be written at once within a single motion loop. All variables will be written within a single motion loop.
- **500 us loop time:** Up to 128 variables can be written at once within a single motion loop. If more than 128 variables are being written to by the I/O connection, then 128 variables will be written each motion loop until the entire transaction is done.
- **250 us or shorter loop time:** Up to 64 variables can be written at once within a single motion loop. If more than 64 variables are being written to by the I/O connection, then 64 variables will be written each motion loop until the entire transaction is done.

Command Area

Writing the data to the Command Area may take up to 3 ms. After the write is complete, the entire command block is submitted as a single consistent block. Therefore the data is treated as a single consistent block, although it may take several loop times before the block is submitted.

Indirect Data Map

The only data guaranteed to be consistent are variables, if they are placed first in the Incoming Data area of the Indirect Data Map. The size of guaranteed consistent data depends on the loop time:

- **1 ms or longer loop time:** The first 100 registers will be consistent if they are all variables.
- **500 us loop time:** The first 50 registers will be consistent if they are all variables.
- **250 us or shorter loop time:** The first 25 registers will be consistent if they are all variables.

As soon as the first register is encountered that is not a current or initial variable register, the guarantee for data consistency no longer applies. Remaining registers in the I/O block will be processed as time allows on subsequent motion loops, taking up to a maximum of 10 ms. For this reason, variables should be included first in the Incoming I/O Data Area of the Indirect Data Map in order to ensure their consistency.

Notice that if commands are mapped to the Indirect Data Map, the commands will all be submitted at the same time, after all values have been written.

Other Locations

No data consistency is guaranteed when consuming in other areas (such as axis parameters). The entire write can take as long as 10 ms, with the write being deferred to the next control loop at any point as required to meet loop time requirements.

See Also

PROFINET Overview | Setting up a PROFINET I/O Connection | Handling Broken PROFINET Connections | Troubleshooting PROFINET

6.7.10.7.4. Using PROFINET Record Data

PROFINET defines Record Data in addition to the cyclic I/O data exchange. The PLC can read or write these records using RDREC and WRREC or similar function blocks. Each data record within a PROFINET device is addressed through the *Slot*, *Subslot*, and *Index*. The RMC supports three types of data records: fixed, custom, and PROFINET-specific. Only the first two are typically of interest to RMC users.

PROFINET allows the caller to specify the length of the data being read, but does not specify a starting offset within the given data record. Therefore, the PLC can only read or write starting at the beginning of a data record. The RMC75 and RMC150 maximum read/write length per record is 2048 registers (8,192 bytes). The RMC200 maximum read/write length per record is 4096 registers (16,384 bytes).

RMC75 and RMC150 Data Records

The RMC75 and RMC150 have many pre-defined data records and four configurable data records.

The records limit you to read and write starting at the *beginning* of each defined range. Each read or write of an RMC75 or RMC150 data record can be up to 2048 registers (8,192 bytes).

Fixed Data Records

The RMC75 and RMC150 define one data record for each RMC register file with the index of each match the number of the register file. Fixed data records are addressed as *Slot* 0, *Subslot* 1, and *Index* of 7-255:

Index	Starting Address	Description

7	%MD7.0	Contents of RMC register file 7
8	%MD8.0	Contents of RMC register file 8
:	:	:
255	%MD255.0	Contents of RMC register file 255

For a list of the register files and their content see the RMC75 Register Map or the RMC150 Register Map.

Notice that because PROFINET always starts its reads and writes at the beginning of the data record, fixed data records can only be used to read or write starting at the beginning of an RMC register file. Custom Data Records get around this limitation.

Custom Data Records

In order to allow users to read and write registers in the RMC starting at locations other than the beginning of a register file, the RMC75 and RMC150 define four Custom Data Records, which are addressed as *Slot* 0, *Subslot* 1, and *Index* 1000-1003:

Index	Starting Address	Description
1000	Configurable	The starting address is configurable in RMCTools.
1001	Configurable	The starting address is configurable in RMCTools.
1002	Configurable	The starting address is configurable in RMCTools.
1003	Configurable	The starting address is configurable in RMCTools.

The starting addresses of each of the Custom Data Records are set up in the PROFINET Settings Page.

For example, to read or write a block of registers starting at variable 50 (%MD56.50), the user can set Custom Data Record 1000 to have a starting address of %MD56.50, and then read or write to Index 1000.

RMC200 Data Records

Use the PROFINET Data Records Address Map to view and configure the RMC200 Data Records. Data records 0-3 are pre-defined and you can configure up to 128 additional data records to access registers in the RMC200.

The records limit you to read and write starting at the *beginning* of each defined range. Each read or write of an RMC200 data record can be up to 4096 registers (16,384 bytes).

Fixed Data Records

The following RMC200 data records are fixed:

Index	Starting Address	Register Count	Description
0	%MD8.0	256	Indirect Data Map
1	%MD16.0	1280	Command Area
2	%MD1024.0	256	Variables 0-255
3	%MD23.0	4096	Image Upload/Download Area

Adding Data Records

1. In the bottom row, in the **Start Address** column, enter the data record number you wish to use. The data record number must be a whole number between 4.0 and 32767.0, with a decimal point and a zero. The first number, before the decimal point, is the Record Index. The second

number, after the decimal point, is the starting register offset within the data record, which must be zero. For example, 4.0 or 100.0.

2. In the **Registers** column, enter the number of registers. This is the number of 32-bit registers in the RMC.

3. In the **Map To** column, enter the area in the RMC that this address range will apply to.

4. Repeat for additional data records. The rows will automatically re-order entries to keep them in numerical address order.

5. Click the Download button  to download the changes to the RMC.

PROFINET-specific Record Data

In addition to the user data records described above, the RMC also supports PROFINET-specific data records as defined in the PROFINET standard. These records are generally only of interest to the PROFINET configuration software. They have Index values in the range of 0x8000-0xFFFF. Refer to the PROFINET specification for details.

Record Data Format and Length

The contents of the Fixed and Custom Data Records match the contents of the corresponding RMC registers. Therefore, each register will be a 32-bit (4-byte) DINT, DWORD, or REAL data value. However, be aware that the MLEN and LEN parameters on the RDREC and WRREC function blocks are in bytes. Therefore, these values must be 4x the number of registers you want to read or write. By default each 32-bit value is transmitted most-significant byte first, which should work for most IO controllers. Otherwise, the byte order can be changed to least-significant byte first on the PROFINET Settings Page.

The RMC75 and RMC150 maximum read/write length per data record is 2048 registers (8,192 bytes). The RMC200 maximum read/write length per data record is 4096 registers (16,384 bytes).

See Also

PROFINET Overview | PROFINET Settings Page | Setting up a PROFINET I/O Connection

6.7.10.7.5. Handling Broken PROFINET Connections

It is important in many industrial applications to detect faults quickly. One such fault is losing PROFINET I/O communication. PROFINET supports a configurable timeout value, which is expressed in terms of a multiple of the update time. The default multiplier is 3 for most IO Controllers. For example, for an update time of 16 ms and a timeout multiplier of 3, the connection timeout will be 3 x 16 ms or 48 ms.

When either the IO-Controller or IO-Device does not receive a packet from the other device for the timeout interval, it closes the connection and typically indicates this condition to the main program. The method of indicating this condition depends on the actual device. This topic describes the methods used by the RMC and Siemens TIA Portal.

Handling Broken I/O Connections in the RMC

The RMC has tags that indicate the state of the I/O connections. The user can use these registers to qualify whether certain operations in the User Programs can be done, or they can use these registers in the Program Triggers to respond to the change in state of the connection.

To find these tags when editing the Program Triggers or a User Program, use the Address Selection tree and browse to:

Controller > Communication Settings > Ethernet

RMC75/150 Tag Name	RMC200 Tag Name	Description

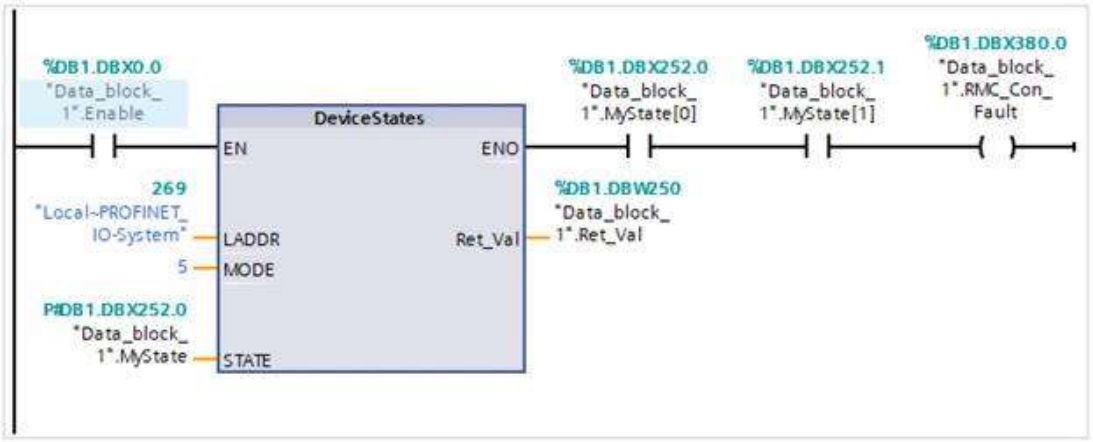
_Enet.CCStatus.Active	_Enet.PNIOConnStatus.Active	**I/O Connection Active** This bit is set as long as an I/O connection is currently active. If the connection is closed or timed out, this bit will be cleared.
_Enet.CCStatus.TimedOut	_Enet.PNIOConnStatus.TimedOut	**I/O Connection Timed Out** This bit is set when an I/O connection timed out. Notice that this is only one method of a connection being closed (another example is the PLC intentionally closing it). This bit is cleared when the I/O connection is re-opened. The user can look for the rising edge of this bit in the Program Triggers to respond to a time-out. A time-out can occur when the cable is disconnected, or when the client is powered off or reset. This bit is not used for PROFINET on the RMC75/150.
_Enet.PLCStatus	_Enet.PNIOConnPLCState	**I/O Connection PLC Status** This register indicates the state of the controlling PLC. This register (DINT data type) can hold one of three values: Unknown (0), Run (1), and Program (2). The Unknown state applies whenever no PROFINET I/O connection is active.

Handling Broken I/O Connections in Siemens TIA Portal

One method of handling broken connections in the Siemens TIA software is via the **DeviceStates** instruction.

To set up the DeviceStates instruction:

1. Set the **LADDR** input to the hardware identifier of the PROFINET IO.
2. Set the **MODE** input to 5 (you may wish to use 2 instead - see the TIA help).
3. Connect the **STATE** input to an array of booleans capable of holding the state for each IO device.
4. Connect the **Ret_Val** output to a data block to record any errors in executing the instruction.



Once this block has been set up, the first boolean in the State array will indicate whether a connection fault is present, and the remaining portion of the array will indicate which connection has a fault.

Example:

A PROFINET IO system has connections to three IO devices with numbers 1, 2, and 3. The connection with device number 1 is faulty.

The following will be written to the state array.

Bit 0 = **1**: A fault exists for at least one of the IO slaves

Bit 1 = **1**: IO device with device number 1 is faulty

Bit 2 = **0**: IO device with device number 2 is not faulty

Bit 3 = **0**: IO device with device number 3 is not faulty

See Also

PROFINET Overview | I/O Connection Status | I/O Connection PLC Status

6.7.10.7.6. Troubleshooting PROFINET

Using the Event Log

The Event Log is the primary troubleshooting tool in the RMC for PROFINET I/O. It will report each time a PROFINET I/O connection is opened or closed and report errors with the connection as well. It can also record every change in the incoming (consumed) PROFINET I/O data (the Output Data from the PLC). It does not record the outgoing data sent by the RMC (the Input Data in the PLC).

The Event Log can report the following events based on Incoming I/O Data:

- **Initial Data**
  This entry is logged only if the Sync Register is used and the **Ethernet I/O Logging** filter option for the Event Log is set to **All**. It is reported when the first incoming I/O data is received after a new controlling I/O connection is established, showing both the data and the value of the Sync In register. Notice that the data is not yet applied to the RMC.
- **Data Changed**
  This entry is logged only if the Sync Register is used and the **Ethernet I/O Logging** filter option for the Event Log is set to **All**. It is reported when any Incoming I/O Data register has changed, but the Sync In register has not. This entry shows both the data and the value of the Sync In register. Notice that the data is not yet applied to the RMC.
- **Request Made**
  This entry is logged each time the incoming I/O data is applied to the RMC. If the Sync Register is used, then this is each time the Sync In register changes. If the Sync Register is not used, then this occurs when the I/O data is first received, and each time after that when any register in the Incoming Data changes. This log entry will be shown by default, but can be disabled by setting the **Ethernet I/O Logging** filter option for the Event Log is set to **None**.

Using the Communications Statistics

To open the Communication Statistics window, in the Project Pane, select the desired controller. On the **Controller** menu, choose **View Communication Statistics**.

The Communications Statistics window provides information on open PROFINET I/O connections, including producing and consuming registers and the update time. The statistics also provide advanced information on the Ethernet traffic.

See Also

PROFINET Overview

6.7.11. Other Protocols

6.7.11.1. Simple Network Management Protocol (SNMP)

This topic only applies to the RMC200, which is the only RMC that supports SNMP.

SNMP is an open protocol running on top of UDP for collecting and organizing information about devices on IP networks. It is widely used in network management for network monitoring. Support for SNMP is required by PROFINET devices wishing to comply with the Topology Engineering & Discovery feature, which is especially important for devices with multiple Ethernet ports like the RMC200.

Use of SNMP is beyond the scope of this documentation. This topic includes details that will be useful to an experienced IT professional for utilizing the SNMPv1/v2c agent included in the RMC200 controller.

SNMP Versions

The RMC200 supports SNMPv1 and SNMPv2c. It does not support SNMPv3. The RMC200 operates as an SNMP agent.

SNMP Community Strings

The RMC200 supports the following two SNMP community strings:

- **Read-only community string:** "public"
  This community string only supports reading objects over SNMP.
- **Read-write community string:** "private"
  This community string supports reading and writing object values over SNMP. However, the

only writable objects are sysName, sysLocation, and sysContact, which do not affect the operation of the motion controller.

The RMC does not allow changing the community strings. Both of these community strings are required in order for the PROFINET Topology Engineering & Discovery feature to work.

SNMP Traps

The RMC200 does not support generating traps and therefore does not support setting trap destinations.

Supported Management Information Bases (MIBs)

SNMP groups managed data into multiple Management Information Bases (MIBs). The RMC200 supports two main MIBs: MIB-2 and LLDP-MIB. Each of these MIBs also includes sub-MIBs. Here is a full list of the major object identifiers (OIDs) supported in the RMC200 and the associated MIBs and standards:

Object Identifier (OID)	MIB	Standard
1.3.6.1.2.1 (mib-2)	RFC1213-MIB	RFC 1213
1.3.6.1.2.1.1 (system)	SNMPv2-MIB	RFC 3418
1.3.6.1.2.1.2 (interfaces)	IF-MIB	RFC 2863
1.3.6.1.2.1.4 (ip)	IP-MIB	RFC 4293
1.3.6.1.2.1.5 (icmp)	IP-MIB	RFC 4293
1.3.6.1.2.1.6 (tcp)	TCP-MIB	RFC 4022
1.3.6.1.2.1.7 (udp)	UDP-MIB	RFC 4113
1.3.6.1.2.1.10.7 (dot3)	EtherLike-MIB	RFC 3635
1.3.6.1.2.1.11 (snmp)	SNMPv2-MIB	RFC 3418
1.3.6.1.2.1.31 (ifMIB)	IF-MIB	RFC 2863
1.3.6.1.6.3.1 (snmpMIB)	SNMPv2-MIB	RFC 3418
1.0.8802.1.1.2 (lldpMIB)	LLDP-MIB	IEEE 802.1AB
1.0.8802.1.1.2.1.5.4623 (lldpXdot3MIB)	LLDP-EXT-DOT3-MIB	IEEE 802.1AB
1.0.8802.1.1.2.1.5.3791 (lldpXPnoMIB)	LLDP-EXT-PNO-MIB	PROFINET Specification

See Also

RMC Ethernet Protocols

6.8. PROFIBUS

6.8.1. PROFIBUS-DP Slave Communications Overview

PROFIBUS-DP Slave is the communication protocol available on the RMC75P and on the RMC150E PROFIBUS module. PROFIBUS-DP is an open industrial fieldbus and is vendor independent, allowing a large range of PLCs and other PROFIBUS masters to control the RMC. The RMC performs as a PROFIBUS-DP slave, requiring a PROFIBUS-DP master to control it. The type of the actual master—whether a PLC or other controller—is unimportant.

RMC PROFIBUS-DP Specifications

	RMC75P	RMC150E PROFIBUS
Baud Rates	9.6Kbaud to 12Mbaud	
Product Identifier Number	0x07E1	0x0AC6
Supported	Freeze mode, sync mode, automatic baud-rate detect	
Modularity	The RMC is a modular station that can have one module selected at a time.	
Station Addresses	0-99 (selected by rotary switches)	1-126 (set in RMCTools or by Set_Slave_Address function from a PROFIBUS Class 2 Master)

PROFIBUS-DP Modes

RMC150E and RMC75P

The following modes are available on the RMC150E and RMC75P:

- I/O Mode (4 registers)
- I/O Mode (8 registers)
- I/O Mode (16 registers)
- I/O Mode (32 registers)

Each of these modes specifies a number of 32-bit registers, *n*. For each mode, *n* registers are continuously sent from the RMC's first *n* Indirect Data Map registers to the PLC, and *n* registers are always sent from the PLC to the first *n* registers in the RMC's Variable Table.

Choose the mode with the number of registers you need for your application. For details on how to use these modes, see the PROFIBUS Mode: I/O Modes topic.

RMC75P Only

In addition to the I/O modes described above, the RMC75P supports Basic/Enhanced PROFIBUS Modes. These modes are generally more difficult to use. Therefore, most users will prefer the cyclic modes. For details on these additional modes, see the Basic/Enhanced PROFIBUS Modes topic.

PROFIBUS Configuration

For details on configuring the PROFIBUS, see the PROFIBUS Configuration topic.

Connection Status

The PROFIBUS Connection Status register provides information on the PROFIBUS connection. This register is located at:

RMC75P: %MD21.6

RMC150: %MD45.6

This register has one bit defined, which can be used within the RMC user programs as part of handling broken connections:

Bit	Description
0	**Connection Established.** This bit will be set when the PROFIBUS interface is in the Data Exchange mode.

Troubleshooting

For details on how to troubleshoot the RMC PROFIBUS, see the Troubleshooting PROFIBUS topic.

See Also

PROFIBUS Configuration | Communications Overview | Troubleshooting PROFIBUS

6.8.2. Troubleshooting PROFIBUS

The RMC and RMCTools provide several tools for troubleshooting the PROFIBUS communications.

LED

An LED on the RMC indicates that a PROFIBUS connection has been made and the RMC is exchanging data with the PROFIBUS master. On the RMC75P, this LED is labeled **Net**. On the RMC150 PROFIBUS module, this LED is labeled **Comm Status**.

This LED should be *solid* green. If it is flashing or flickering green, it indicates an intermittent connection, generally caused by a parameterization or configuration error.

This LED may be off for any of the following reasons:

- Wrong slave address set in the RMC
- Wrong slave address set in the PROFIBUS master
- Incorrect GSD selected in the PROFIBUS master
- PROFIBUS master has not been configured to include this slave
- Incorrect parameterization sent by the PROFIBUS master
- Incorrect configuration sent by the PROFIBUS master
- PROFIBUS master not active
- Improper PROFIBUS cable or connector
- Incorrect PROFIBUS cable termination
- RMC is powered off
- PROFIBUS master is off

PROFIBUS Connection Status Register

The PROFIBUS Connection Status register provides PROFIBUS connection information to user programs. This register is located at:

RMC75P: F21:6

RMC150: F45:6

This register has one bit defined, which can be used within the RMC user programs as part of handling broken connections:

Bit	Description
0	**Connection Established.** This bit will be set when the PROFIBUS interface is in the Data Exchange mode.

Event Log

The Event Log Monitor can be used to view the PROFIBUS data. By default, the Event Log only records the changes in the PROFIBUS connection state or address. However, the Event Log Filter can be configured to make a log each time there is a change in the data sent from the PLC to the RMC.

Using the Event Log for the I/O Modes

To log the incoming PROFIBUS data, set the Event Log Filter as follows:

1. In the Project pane, right-click **Event Log** and choose **Properties**.
2. Click the **Communications** folder and click **PROFIBUS**.

3. Check the **I/O Mode Data Logging** and click **OK**. The Event Log will now log an entry each time the PROFIBUS data changes and will display the data.

Using the Event Log for the Basic/Enhanced Modes

See the topic for the respective Basic/Enhanced mode for details.

Communication Statistics

The Communication Statistics Window provides information on the PROFIBUS settings and connection, including which parameterization and configuration data was last sent to the RMC.

PROFIBUS Timeouts

How long will it take for the RMC to detect that the PROFIBUS connection is broken?

This is controlled by the PROFIBUS master.

Specifically, when the master sends the Set_Parameters message as one of the required steps to enter DATA-EXCHANGE mode with the slave device, the second and third bytes in that parameterization are called WD_Fact_1 and WD_Fact_2. These two 8-bit values are multiplied together and then by 10ms to give a timeout value. Therefore, the master can make this value as short as 10ms and as long as 650 seconds (>10 minutes!).

The user must determine how to set this value for their particular PROFIBUS master.

See Also

Help Overview

6.8.3. Configuration

6.8.3.1. PROFIBUS Configuration

The following steps are required to connect an RMC unit to a PROFIBUS network:

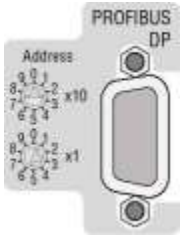
1. Set the RMC's station address

RMC75P

On the front panel of the RMC75P, use a small screwdriver to turn the rotary switches to the desired values. The address is the sum of the value of each switch times its respective multiplier. Use any station address between 1 and 99. Zero (0) is not allowed as a station address. Since the station address 1 is normally used for the master, Delta recommends not setting the address to 1.

The station address on the RMC75P must match the station address as expected by the master. Step 3 explains how to set the station address in the master.

Example



The station address is:
(0 x 10) + (3 x 1) = 3

RMC150E

a. In RMCTools, go online with the RMC.

b. In the Project Pane, expand the **Modules** folder and double-click the **PROFIBUS** module.
c. On the **PROFIBUS** page, enter the desired station address. Use any station address between 1 and 125. Since the station address 1 is normally used for the master, Delta recommends not setting the address to 1. The value 126 is the default address and indicates that a station address has not yet been assigned.
d. Click **OK**. The change will automatically be downloaded to the RMC.

2. Determine the Appropriate GSD Configuration Module

A GSD file is a device description file used to inform the PROFIBUS master how to communicate with the device. The master requires a GSD file from each device in the network. The RMC's GSD files are is installed by the RMCTools software package in the RMCTools folder. The GSD file is also available from Delta's website https://deltamotion.com.

- RMC75P: DELT07E1.GSD
- RMC150E: DELT0AC6.GSD

The GSD file contains several configuration module entries. When adding an RMC to a PROFIBUS network (Step 3), you will need to select exactly one of these configuration modules. Each of these modules is assigned a title, which is often referred to by PROFIBUS configuration software as an *Order Number*.

To determine the correct GSD order number, first determine the operation mode. The following operation modes are available on the RMCs:

RMC75P	RMC150E PROFIBUS
I/O Mode (32 regs)	I/O Mode (32 regs)
I/O Mode (16 regs)	I/O Mode (16 regs)
I/O Mode (8 regs)	I/O Mode (8 regs)
I/O Mode (4 regs)	I/O Mode (4 regs)
Basic	
Basic+	
Enhanced	
Enhanced+	

If you do not know the difference between these modes or have not yet determined which is right for your application, please read the PROFIBUS Overview topic. Most users prefer the I/O modes.

Delta has found several cases where the GSD file is not supported correctly or manual setup is otherwise required. In these cases, you will need to manually set up the PROFIBUS. See the topic for the mode you will be using for parameterization and configuration values.

3. Configure the PROFIBUS Network

The PROFIBUS network configuration is stored in the master PROFIBUS device. Creating this configuration is the most difficult step in making the network work. This step requires a device description file (GSD file) from each device in the network, including the RMC.

The following steps outline configuring your PROFIBUS-DP network. The steps are general because various manufacturers handle these steps differently.

1. Open your PROFIBUS-DP master configuration program.
2. If you are modifying an existing PROFIBUS-DP network, open your current configuration file.
3. If you are creating a new PROFIBUS-DP network, you must create a new network, add a master device to the network, and select the baud rate of the network.
4. Add the RMC's GSD file to your configuration programs GSD database if it is not already there.

5. Add and configure an **RMC75P Motion Controller** or **RMC150E Motion Controller** slave device to the network. This involves selecting the correct configuration module for your RMC module and application. Refer to Step 2, **Determining the Appropriate GSD Configuration Module,** for details on selecting the correct configuration module.
6. Select the addresses that the master will use for the registers read from the RMC.
7. The PROFIBUS configuration should include a Word Order setting. Because PROFIBUS-DP does not specify the word order for 32-bit values, Delta allows the order to be swapped through this setting. You will probably not know which is correct until later in the process. If you receive data over the PROFIBUS network, but the values are not what you expect or do not make sense, return to this step and try swapping the word order.
8. Add any other RMC devices you want on the same network. To do this, repeat steps 5, 6 and 7.
9. Save your configuration.
10. Send the configuration to the master device. This step varies greatly depending on the type of master you use.

If you are using *COM PROFIBUS*, *SyCon*, or *SST Profibus Configuration*, you should select one of the following topics for more detailed instructions:

Configuring a PROFIBUS-DP Network with COM PROFIBUS

Configuring a PROFIBUS-DP Network with SST Profibus Configuration

Configuring a PROFIBUS-DP Network with SyCon

4. Ready to use PROFIBUS!

Once you have configured the PROFIBUS, can begin the communications. For details on how to use it, see the topic for the mode you have chosen:

I/O Mode (4, 8, 16, or 32 regs)

Basic (RMC75P only)

Basic+ (RMC75P only)

Enhanced (RMC75P only)

Enhanced+ (RMC75P only)

See Also

PROFIBUS-DP Overview

6.8.3.2. Configuring a PROFIBUS-DP Network with COM PROFIBUS

Before reading this topic, you should read and understand PROFIBUS Configuration. This topic only gives a specific example of doing one step of the configuration process. In addition, Siemens may, and probably will, change the steps taken here slightly with each version of *COM PROFIBUS*.

The following steps have been tested with *COM PROFIBUS* versions 3.0 and 3.3:

1. Start *COM PROFIBUS*.
2. If you are modifying an existing PROFIBUS-DP network, open your current configuration file.

3. If you are creating a new PROFIBUS-DP network, you must create a new network and add a master device to the network.
    - On the **File** menu, click **New**.
    - In the **Master & Host Selection** dialog, enter the Address of the master device (in most cases, 1 is a good choice), the Master Station Type, and the appropriate Host Station Type if available. Click **OK**.
    - On the **Configure** menu, click **Bus Parameters**.
    - In the **Bus Parameters** dialog, set the **Bus Profile** to **PROFIBUS DP** and the **Baud Rate** to the desired rate. The RMC75P is capable of speeds up to 12000kBaud (12MBaud); check the rates supported by your master and other slaves. Click **OK**.
4. If it is not already in the database, add the GSD file to your configuration programs GSD database.
    - Copy the GSD file to the GSD folder under the COM PROFIBUS folder. The GSD file can be found in the folder where RMCTools is installed.
        - For the RMC75P, this file is DELT07E1.GSD
        - For the RMC150E PROFIBUS, this file is DELT0AC6.GSD
    - Copy the BMP file to the BITMAPS folder under the COM PROFIBUS folder. The GSD file can be found in the folder where RMCTools is installed.
        - For the RMC75P, this file is DELTR70N.GSD
        - For the RMC150E PROFIBUS, this file is DELTR15N.GSD
    - On the **File** menu, click **Scan GSD Files**.
5. Add and configure an **RMC Motion Controller** slave device to the network.
    - On the **Configure** menu, click **New Slave**.
    - In the **PROFIBUS Address** dialog, select the desired station address, and click **OK**.
    - In the **Family** list, click **Other**.
    - In the **Station Type** list, click **RMC75P Motion Controller** or **RMC150E Motion Controller**.
    - Click **OK**. In most cases you will be prompted at this time to select a **Preset Configuration**. Refer to the **Determining the Appropriate GSD Configuration Module** section in the PROFIBUS Configuration topic for guidance on selecting the correct option.
    - If you are not prompted to select a **Preset Configuration**, then continue with the following steps:
        - Right-click on the RMC slave device icon, and click **Configure** from the shortcut menu.
        - In the **Configure: Delta RMC Family** dialog, click **Order No.**
        - In the **Select by Order Number** dialog, click the order number determined in the previous steps. Click **Accept**. Click **Close**.
    - Click **OK**.
6. You may wish to assign register addresses within the master at this time.
   This is optional for many masters, because it can be done in the master configuration software. In fact, at times it is not possible to edit the register addresses.
    - Right-click on the RMC slave device icon, and select **Configure** from the shortcut menu.
    - Move the cursor to the top row's **I Addr** cell, and either enter the offset that you wish to access the data at, or click the **Auto Addr.** button. Do the same for the **O Addr** cell.
    - Click **OK**.
7. Select the data word order.
   Because PROFIBUS-DP does not specify the word order for 32-bit values, we allow the order to be swapped. You will probably not know which is correct until later in the process. If you receive data over the PROFIBUS network, but the values are not what you expect or do not make sense, return to this step and try swapping the word order:

- Right-click the RMC slave device icon and click **Parameterize**.
- Select the cell in the Value column for the **Word Order** parameter.
- Click **Select**.
- Choose **LSW First** or **MSW First**, and click **OK**.

8. Add any other RMC devices you want on the same network. To do this, repeat steps 5, 6 and 7.
9. Save your configuration.
10. Send the configuration to the master device. This step varies greatly depending on the type of master you use.

See Also

PROFIBUS-DP Overview

6.8.3.3. Configuring a PROFIBUS-DP Network with SST Profibus Configuration

Before reading this topic, you should read and understand PROFIBUS Configuration. This topic only gives a specific example of doing one step of the configuration process. In addition, SST may, and probably will, change the steps taken here slightly with each version of *SST Profibus Configuration*.

The following steps have been tested with *SST Profibus Configuration* versions 1.9.9:

1. Start *SST Profibus Configuration*.
2. If you are modifying an existing PROFIBUS-DP network, open your current configuration file.
3. If you are creating a new PROFIBUS-DP network, you must create a new network and add a master device to the network:
   - On the **File** menu, click **New**.
   - The tree-type control on the left is the GSD library. It contains all devices known about. In this tree, expand **Masters** and **SST** and double-click on the SST master you will be using (e.g. SST-PFB-SLC).
   - In the **Station** box, enter the station address of your master. Click **OK**.
   - On the **Browse** menu, click **Network Properties**.
   - In the **Baud Rate** field, select your desired baud rate. The RMC is capable of speeds up to 12000kBaud (12MBps); check the rates supported by your master and other slave devices. Click **OK**.
4. Add the GSD file to your configuration programs GSD database.
   - On the **Library** menu, click **Add GSD**.
   - Browse to the GSD file. It is located in the folder where RMCTools is installed.
     - RMC7P: DELT07E1.GSD
     - RMC150E PROFIBUS: DELT0AC6.GSD
   - Select the file, and click **Open**.
5. Add and configure an **RMC Motion Controller** slave device to the network.
   - In the GSD library tree, expand **Slaves** and **Delta Computer Systems, Inc.** and double-click on **RMC75P Motion Controller** or **RMC150E Motion Controller**.
   - On the **General** tab, select the station address of your master in the **Station** field of the dialog that is displayed.

- In the **Modules** tab, click **Add**. In the dialog that is displayed, select the appropriate module from the **Available Modules** list and click **OK**. For help on determining the appropriate configuration module, refer to the **Determining the Appropriate GSD Configuration Module** section of the PROFIBUS Configuration topic.
- In the **Address** (or **SLC Address**) tab, review the default addressing and change it if necessary.
- On the **Ext. Prms** tab, notice the **Word Order** parameter. Because PROFIBUS-DP does not specify the word order for 32-bit values, we allow the order to be swapped through this setting. You will probably not know which is correct until later in the process. If you receive data over the PROFIBUS network, but the values are not what you expect or do not make sense, return to this step and try swapping the word order.
- Click **OK**.

6. Add any other RMC devices you want on the same network. To do this, repeat step 5.
7. Save your configuration.
8. Send the configuration to the master device. This step varies depending on the master you selected.

See Also
PROFIBUS-DP Overview

6.8.3.4. Configuring a PROFIBUS-DP Network with SyCon

Before reading this topic, you should read and understand PROFIBUS Configuration. This topic only gives a specific example of doing one step of the configuration process. In addition, Hilscher may, and probably will, change the steps taken here slightly with each version of *SyCon System Configurator*.

The following steps came from *SyCon System Configurator 2.710*:

1. Start *SyCon System Configurator*.
2. If you are modifying an existing PROFIBUS-DP network, open your current configuration file.
3. If you are creating a new PROFIBUS-DP network, you must create a new network and add a master device to the network.
   - On the **File** menu, click **New**. If you have multiple networks installed you will need to then select the network type: select PROFIBUS. If you are not given the option of selecting PROFIBUS, you may not have installed the PROFIBUS driver for SyCon.
   - On the **Insert** menu, click **Master**. Move the cursor to the top device slot in the window area (the cursor will change to an arrow with an M next to it), and click to place the master device.
   - In the **Insert Master** dialog, select the desired master and click the **Add>>** button. Enter the station address, and click **OK**.
   - On the **Settings** menu, click **Bus Parameter**.
   - In the **Bus Parameter** dialog, set the **Baud Rate** to the desired rate. The RMC is capable of speeds up to 12000kBits/s (12MBaud); check the rates supported by your master and other slaves. Click **OK**.
4. Add the GSD file to your configuration programs GSD database.
   - You must first have an open PROFIBUS project file.
   - On the **File** menu, click **Copy GSD**.
   - In the **Open** dialog, navigate to the folder where the DELT07E1.GSD file is located (by default it is installed in the main RMCTools folder)

        - RMC75P: DELT07E1.GSD
        - RMC150E PROFIBUS: DELT0AC6.GSD
    - Select the file and click **Open**.

> **NOTE:**
> Depending on the version, this may not add the bitmap file. If you wish to use the bitmap file; copy the DELTR70N.BMP or DELTR15N.BMP file to the Fieldbus\Profibus\BMP folder under the SyCon folder (by default, the total path will be C:\Program Files\Hilscher GmbH\SyCon\Fieldbus\Profibus\BMP).

5. Add an **RMC Motion Controller** slave device to the network.
    - On the **Insert** menu, click **Slave**. Move the cursor to the next available device slot in the window area (the cursor will change to an arrow with an *S* next to it), and click to place the device.
    - In the **Insert Slave** dialog, click **RMC75P Motion Controller** or **RMC150E Motion Controller**, and then click the **Add>>** button. Enter the station address, and click **OK**.
6. Configure the RMC Slave Device.
   This involves selecting the correct configuration module for your RMC module and application. Refer to the sub-topics **Determining the Appropriate GSD Configuration Module** in the PROFIBUS Configuration topic for details on selecting the correct configuration module.
    - Right-click on the RMC slave device icon, and click **Slave Configuration** from the shortcut menu.
    - In the **Slave Configuration** dialog, from the list of order numbers, select the configuration module you selected in the previous steps.
    - Click **Append Module**.
    - You may change the **I Addr.** and **O Addr.** fields for the added module to set the offsets of the data.
    - Click **Parameter Data**, and then **Common**. Select **LSW First** or **MSW First** and click **OK**. Because PROFIBUS-DP does not specify the word order for 32-bit values, we allow the order to be swapped through this setting. You will probably not know which is correct until later in the process. If you receive data over the PROFIBUS network, but the values are not what you expect or do not make sense, return to this step and try swapping the word order.
    - Click **OK**.
7. Add any other RMC devices you want on the same network. To do this, repeat steps 5 and 6.
8. Save your configuration.
9. Send the configuration to the master device. This step varies depending on the master you selected.

See Also
PROFIBUS-DP Overview

6.8.4. Using I/O Modes

6.8.4.1. PROFIBUS Mode: I/O Modes

The RMC's PROFIBUS I/O modes continuously pass blocks of data between the PLC and RMC. The following modes are available on the RMC150E and RMC75P:

- I/O Mode (4 registers)
- I/O Mode (8 registers)
- I/O Mode (16 registers)
- I/O Mode (32 registers)

Each of these modes specifies a number of 32-bit registers, *n*. For each mode, *n* registers are continuously sent from the RMC's first *n* Indirect Data Map registers, and *n* registers are always sent from the PLC to the first *n* registers in the RMC's Variable Tables.

For your application, choose the mode with the number of registers you need. The PROFIBUS I/O modes are very simple and easy to use. No handshaking is required, and you are free to use the data as you wish. The PROFIBUS data is updated each Loop Time of the RMC.

Put in other words, the PLC's PROFIBUS Input Data comes from the RMC's Indirect Data Map, and the PLC's PROFIBUS Output Data goes to the RMC's Variable Tables.

The RMC's Indirect Data Map is an area in which any register in the RMC can be mapped to. Therefore, to have the PLC read any register in the RMC via one of the I/O modes, just put it in the Indirect Data Map.

The RMC's Variable Table contains variable registers that can be used for any purpose. Writing to these variables via the I/O modes can do such things as changing settings or profiles, or starting user programs.

The I/O data is consistent. A block of PROFIBUS data is called 'consistent' if it is consistent over the length of the block, rather than just over a single 8- or 16-bit data item. Consistent blocks of data will stay together through the communication, from the time it was captured in one device until it is delivered to the other device, whereas data from different consistent blocks could have been sampled at different times.

The I/O modes require firmware version 3.10 or newer.

Parameterization

The I/O modes require the PROFIBUS configuration and parameterization listed below. The GSD file directs the PROFIBUS master setup software to automatically set up these values, but Delta has found several cases where it is not supported correctly or manual setup is otherwise required.

I/O Mode:	32 registers	16 registers	8 registers	4 registers
Configuration	C0 FF FF	C0 DF DF	FF	F7
Parameters (bytes 1-7) **(Prm_Data)**	See the PROFIBUS DP specification for details.			
Parameters (bytes 8-13) **(User_Prm_Data)**	00 00 00 xx* 02 20	00 00 00 xx* 02 10	00 00 00 xx* 02 08	00 00 00 xx* 02 04

* The xx parameter can be 00 or 01 and selects whether the least-significant word comes first (00) or most-significant word comes first (01).

Notice that some masters will not support the larger I/O modes.

Using the I/O Modes

1. Configure the PROFIBUS communications

Configure the PROFIBUS communications as described in the PROFIBUS configuration topic.

2. Set up the Indirect Data Map

Once you have chosen the mode and configured the PROFIBUS communications, you will need to set up the RMC's Indirect Data Map for the data you wish to continuously send to the PLC. For details, see the Indirect Data Map topic.

Keep in mind that the first *n* registers of the Indirect Data Map will continuously be sent to the PLC.

Example

A user decides to use I/O Mode (4 regs). This means that the first four items of the Indirect Data Map will continuously be sent to the PLC. The user wants the PLC to monitor the following:

- Axis 0 Status bits
- Axis 0 Actual Position
- Task 0 Status
- Task 0 Current Program

Therefore, the user would add these items to the Indirect Data Map. The Indirect Data Map editor will then look like this:

	Reg #	Map To	Description	Current
0	F18:0	%MD8.0	Axis0 Status Bits	16#00002800
1	F18:1	%MD8.8	Axis0 Actual Position (pu)	19.785192
2	F18:2	%MD24.0	Task 0 Status	16#00000003
3	F18:3	%MD24.3 ...	Task 0 Current Program	1
4	F18:4			
5	F18:5			

It is important to check the External Data Types of the registers used in the register map, and treat them accordingly in the master controller.

3. Set Up the Variable Table

The PLC's Output Data will be continuously sent to the first *n* registers in the RMC's Variable Table. You should add tag names to each data item and make sure the Data Type is set correctly.

Example

A user decides to use I/O Mode (4 regs). This means that the PLC's Output Data will consist of four registers that will be continuously sent to the first four items in the RMC's Variable Table. The user wants the PLC to send the following four data items to the RMC:

- Command Position (REAL Data Type)
- FastSpeed (REAL Data Type)
- SlowSpeed (REAL Data Type)
- DwellTime (REAL Data Type)

Therefore, the user sets the tag names and Data Types of the first four items in the Variable Table like this:

	Register	Tag Name	Units	Type	Retain	Initial	Description
0	F56:0	CommandPosition		REAL	☐	0.0	
1	F56:1	FastSpeed		REAL	☐	0.0	
2	F56:2	SlowSpeed		REAL	☐	0.0	
3	F56:3	DwellTime		REAL	☐	0.0	
4	F56:4			REAL	☐	0.0	
5	F56:5			REAL	☐	0.0	

4. Perform Communications

After configuring the PROFIBUS and setting up the Indirect Data Map and Variable Table, start the PROFIBUS communications. Once the PROFIBUS connection is established, the RMC's Indirect Data Map registers will continuously be sent to the PLC's Input Data, and the PLCs' Output Data will continuously be sent to the RMC's Variable Table. The RMC's variables can be used for anything that they normally are used for. However, trying to change the PROFIBUS variables from user programs is not very useful, because the variables will be overwritten by the PROFIBUS.

Issuing Commands via I/O Mode

To issue commands, or write to RMC registers (other than the I/O Mode variables), or do anything else via the I/O modes, you must create a user program. You can then use the Program Triggers to start a user program when an I/O Mode variable becomes a certain value.

To program the RMC to do this, you must:

1. Create user programs to do the desired actions.
2. Define a variable for starting the programs.
3. Edit the Program Triggers to start the user programs based on the value of the variable.

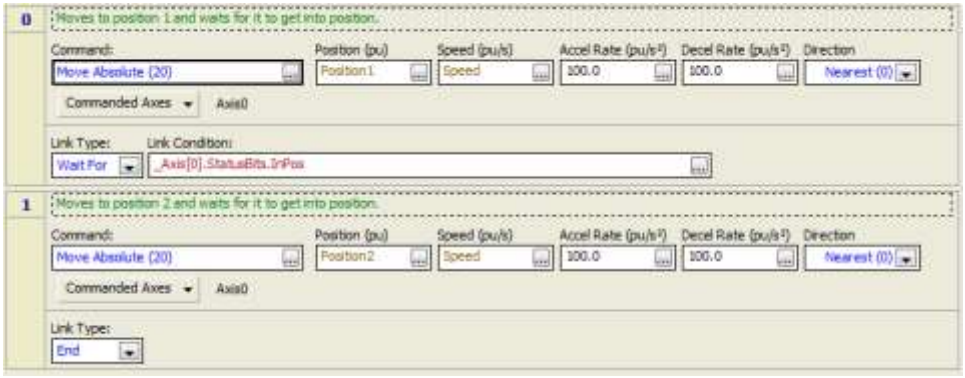
Example

Bruce is using I/O Mode (8 regs). He wants to run two different user programs on the machine. The first program will move back and forth once, and the second will move to a home position. The PLC will send the two cycle positions, the speed, and the home position to the RMC.

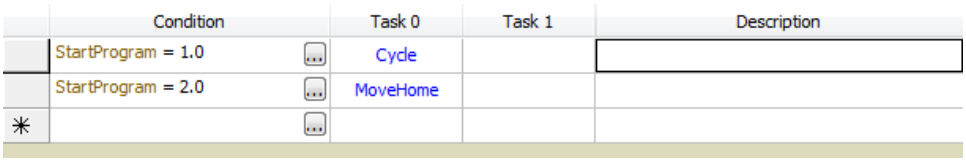
1. Bruce defines variable zero as **StartProgram**, and defines 4 more variables for his cycle positions, speed, and home position.

	Register	Tag Name	Units	Type	Retain	Initial	Description
0	F56:0	StartProgram		REAL		0.0	
1	F56:1	Position1		REAL		0.0	
2	F56:2	Position2		REAL		0.0	
3	F56:3	Speed		REAL		0.0	
4	F56:4	HomePosition		REAL		0.0	
5	F56:5			REAL		0.0	

2. Bruce creates the user programs and calls them **Cycle** and **MoveHome**:



3. Bruce edits the Program Triggers to start the **Cycle** program when **StartProgram** is 1, and to start the **MoveHome** program when **StartProgram** is 2.



4. Now Bruce is ready to perform the communications. To run the Cycle program, the PLC first sets up the position and speed variables, then sets the StartProgram variable to 1. To run the program again, the PLC must first set the StartProgram variable to some other number (such as 0), then set it to 1.

   To run the MoveHome program, the PLC must first set the HomePosition variable, then set the StartProgram variable to 2.

Handshaking

The I/O Modes do not require any handshaking, but you may wish to make some simple handshaking so the PLC knows the RMC is running normally and is performing the requested actions. Here are some methods of doing simple handshaking:

- **Put the StartProgram Variable in the Indirect Data Map**
  In the example above, the StartProgram variable could be put in the Indirect Data Map, so the PLC could see that it changed correctly and knows the communications is working properly.
- **Use the Task Status Registers.**
  The Task status registers indicate whether a task is running, and which user program it is running. You can add these registers to the Indirect Data Map so the PLC can see if the correct user program is running.
- **Create a Program Done Variable**
  You can create a variable for reporting when a user program is complete. For example, at the beginning of the user program, add a step with an Expression (113) command that sets the variable to zero. Then, at the end of the user program, add a step with an Expression (113) command that sets the variable to a certain number indicating that the user program is complete. Make sure to include this variable in the Indirect Data Map so the PLC can see it.

See Also

Help Overview

6.8.5. Using Basic/Enhanced Modes (RMC75P Only)

6.8.5.1. Basic/Enhanced PROFIBUS Modes (RMC75P Only)

The RMC75P supports four additional modes that are not supported by the RMC150E: Basic, Basic+, Enhanced, and Enhanced+.

These Basic/Enhanced modes require more programming in the PLC and are typically more difficult to use than the I/O modes. Therefore, most users will prefer the I/O modes. However, the I/O modes typically require more programming in the RMC. The following table lists pros and cons of the Basic/Enhanced modes to help you choose which mode is appropriate for your application:

I/O Modes	Basic/Enhanced Modes
Advantages • Easy to understand • Requires little programming in the PLC.	**Advantages** • Directly issue commands • Directly write to any register(s)

• Can start user programs to issue commands or read and write any registers. • Supported by both the RMC75P and RMC150E. Therefore, programs made for either RMC easily transfer to other RMC.	• Directly read from any registers(s) • Does not require using the Program Triggers and User Programs
Disadvantages • Requires using the Program Triggers and/or user programs in the RMC.	**Disadvantages** • Can be difficult to understand • Requires a lot of programming in the PLC. • Not supported in the RMC150E. Therefore, programs made for these modes for the RMC75P do not directly transfer to the RMC150E.

Basic/Enhanced Mode Details

If you decide to use a Basic/Enhanced mode, choose from one of the following. For more details, see the topic for each respective mode.

Mode:	Basic	Basic+	Enhanced	Enhanced+
Commands* per Cycle	1	1	1	1
Simultaneous Commands	✓	✓	✓	✓
Cyclic Read Registers	8	16	8	16
Explicit Read Registers	1	1	1+7	1+7
Explicit Write Registers	1	1	1+7	1+7
Bandwidth Required (16-bit words)	16 consistent** I/O words	16 consistent** I/O words + 16 consistent** Input words	2 blocks of 16 consistent** I/O words	2 blocks of 16 consistent** I/O words + 16 consistent** Input words

*For commands with more than 5 command parameters, only the first 5 parameters will be issued. The remaining parameters will be 0. To issue such commands completely, include them in a user program and issue a command via PROFIBUS to start the user program.

**A block of PROFIBUS data is called 'consistent' if it is consistent over the length of the block, rather than just over a single 8- or 16-bit data item. Consistent blocks of data will stay together through the communication, from the time it was captured in the slave device until it is delivered to the master application, whereas data from different consistent blocks could have been sampled at different times.

For details on reading, writing and issuing commands, see the mode type you are using. For instructions on how to select a mode, see the PROFIBUS Configuration topic.

RMC Register Addresses for Basic/Enhanced Modes

When using the Basic/Enhanced modes, the RMC register addresses use the same *file*:*element* addresses as the RMC's Allen-Bradley IEC-61131 addresses.

For example, to read the Axis 0 Actual Position via PROFIBUS, its address is %MD8.8. Therefore, the address via PROFIBUS is file 8, element 8.

See Also

PROFIBUS Overview | PROFIBUS Mode: Basic | PROFIBUS Mode: Basic+ | PROFIBUS Mode: Enhanced | PROFIBUS Mode: Enhanced+

6.8.5.2. PROFIBUS Mode: Basic

The Basic mode is one of the Basic/Enhanced PROFIBUS Modes available only on the RMC75P. Most users will prefer the I/O Modes instead.

Features

This mode has the following features:

- Eight (8) user-selectable cyclic read registers are continuously read for instant access by the PLC or PC.
  See **Response Block** and **Setting up the Indirect Data Map** below.
- One (1) register anywhere in the RMC75 can be explicitly written or read.
  See **Read from the RMC75** and **Write to the RMC75** below.
- Commands can be issued to any number of axes simultaneously.
  See **Command Block**, **Issue a Single Command** and **Issue Simultaneous Commands** below.
- PROFIBUS bandwidth used: 16 consistent I/O words (16-bits each)

RMC75 Register Addresses for PROFIBUS

When communicating over PROFIBUS, the RMC75 registers addresses use the same *file*:*element* addresses as the RMC75 IEC-61131 addresses.

For example, to read the Axis 0 Actual Position via PROFIBUS, notice that its address is %MD8.8. Therefore, the address via PROFIBUS is file 8, element 8.

Parameterization

Basic mode requires the PROFIBUS configuration and parameterization listed below. The GSD file does direct the PROFIBUS master setup software to automatically set up these values, but Delta has found several cases where it is not supported correctly or manual setup is otherwise required.

Configuration: FF

Parameters:

Prm_Data (bytes 1-7): See the PROFIBUS DP specification for details.

User_Prm_Data (bytes 8-14): 00 00 00 xx* 00 08

* The xx parameter can be 00 or 01 and selects whether the least-significant word comes first (00) or most-significant word comes first (01).

Data Blocks

The Basic mode uses two fixed-length blocks of data: the **Command Block** and the **Response Block**.

Command Block

The Command block is a block of 8 contiguous 32-bit output registers. These registers are sent from the PLC or PC to the RMC.

The Command Block has the following structure:

Register Number	Data Type	Description
Command Area: Registers 0 - 5 are used for issuing commands to the RMC75. See **Issue a Single Command** and **Issue Simultaneous Commands** below for details on using these registers.		
0	Integer	Command Register **Bit** — **Bit Description** 31 — Command Request 30 — Deferred Command 29 — Deferred Command 20-28 — Reserved 19 — Axis 3 Select 18 — Axis 2 Select 17 — Axis 1 Select 16 — Axis 0 Select 8-15 — Reserved 7-0 — Command
1	Float	Command Parameter 1
2	Float	Command Parameter 2
3	Float	Command Parameter 3
4	Float	Command Parameter 4
5	Float	Command Parameter 5
Data Channel 0: Registers 6 and 7 are used for reading and writing to any register in the RMC75. See **Read from the RMC75** and **Write to the RMC75** for details on using these registers.		
6	Integer	Read/Write Register **Bit** — **Bit Description** 31 — Read/Write 30 — Read/Write Request 16-29 — Reserved 15-8 — R/W Address File 7-0 — R/W Address Element
7	Float*	Explicit Write Value

*These registers are typically REAL data type (floating point), but in some cases may be DINT or DWORD integers, such as variables declared as such.

Response Block

The Response Block is a block of 8 contiguous 32-bit input registers (cyclic read registers), corresponding to the Indirect Data Map registers 0 to 7 in the RMC75. These registers are continuously sent from the RMC75 to the PLC or PC. See **Configuring the Data** below for details on setting up this data.

Each of the 8 contiguous input registers always reads from the same register in the RMC75. However, one of the registers in the Response Block can be set up to return the value of a read from *any* single register in the RMC. This allows you to read the value of any single register at any time. See **Read from the RMC75** below.

The Response Block has the following structure:

Register Number	Data Type	Description
0	Integer	Indirect Data 0 - ***must be Axis 0 Status bits!***
		Bit / **Bit Description**: 31 Command Acknowledge; 30 Read/Write Acknowledge; 0-29 Axis 0 Status Bits
1	Float*	Indirect Data 1
2	Float*	Indirect Data 2
3	Float*	Indirect Data 3
4	Float*	Indirect Data 4
5	Float*	Indirect Data 5
6	Float*	Indirect Data 6
7	Float*	Indirect Data 7

*These registers are typically REAL data type (floating point), but in some cases may be DINT or DWORD integers, such as variables declared as such.

Configuring the Data

Setting up the Indirect Data Map

The Response Block continuously returns the values from the RMC75 Indirect Data registers 0-7. These registers, in turn, can be mapped to any registers in the RMC75. Thereby, the values from the selected registers in the RMC75 can be read from and written to by writing to and reading from the Indirect Data registers.

To set up the Indirect Data Map:

1. In the Project pane, double-click **Address Maps**, then click Indirect Data Map.
2. In the **Register** column of the first Indirect Data Map entry, type "%MD8.0" and press Enter. This will map Axis 0 Status Bits register to the first item in the Indirect Data Map. Basic mode requires that the first item in the Indirect Data Map contains the Axis 0 Status Bits register.
3. For each of the remaining Indirect Data Map entries 1-7, enter the desired register to map to each. To do this, click the cell in the **Register** column, click the ellipsis button (), then browse to the desired register.
4. If you wish to add additional read capability, one of the Indirect Data Map registers should be mapped to the Read Response register. Then, the corresponding register in the Response Block will return the value of a read from *any* single register in the RMC75 at any time. See **Read from the RMC75** below.

Example

Requirements

The user would like to read the following registers:

- Axis 0 Status Bits
- Axis 0 Actual Position
- Axis 1 Status Bits
- Axis 1 Actual Position
- Task 0 Current Step
- Task 1 Current Step

In addition, the user would like to read some other registers occasionally.

Implementation

- **First**, PROFIBUS communications requires that Axis 0 Status Bits register must be in the first Response Block register, which is entry 0 in the Indirect Data Map.
- **Second**, the Read Response register is needed in order to read other registers occasionally.
- **Third**, the rest of the registers listed above can be put anywhere in the remaining Indirect Data Map registers 0-7.

The user chose to set up the Indirect Data Map like this:

	Reg #	Map To	Description	Current
0	%MD18.0	%MD8.0	Axis0 Status Bits	N/A
1	%MD18.1	%MD8.8	Axis0 Actual Position (pu)	N/A
2	%MD18.2	%MD9.0	Axis1 Status Bits	N/A
3	%MD18.3	%MD9.8	Axis1 Actual Position (pu)	N/A
4	%MD18.4	%MD24.1	Task 0 Current Program/Step	N/A
5	%MD18.5	%MD24.17	Task 1 Current Program/Step	N/A
6	%MD18.6	%MD8.4	Axis0 Read Response	N/A
7	%MD18.7			

Using the Data Blocks

Issue a Single Command

To issue a command, set up the contents of the first six registers of the Command Block, and when complete, toggle the Command Request bit in the first Command Block register.

Notice that commands with more than 5 command parameters cannot be issued via PROFIBUS. To issue such commands, include them in a user program and issue a command via PROFIBUS to start the user program.

To issue a single command to the RMC75, use the following steps:

1. Wait until the **Command Request** bit in the Command Register (0) of the Command Block is equal to the **Command Acknowledge** bit in register 0 of the Response Block. If they are not equal, the RMC is currently processing a command request.
2. Enter the command number in bits 0-7 of the Command Register (0) of the Command Block.
3. If the command has any parameters, put them in registers 1-5 of the Command Block.
4. Clear the **Deferred Command** bits.
5. Set the desired **Axis Select** bit in the Command Register. The command will be sent simultaneously to each axis you select.

> **Note:**
> Using this method, you can send a single command to multiple axes simultaneously. You cannot send different commands to multiple axes simultaneously. To send different

commands to multiple axes simultaneously, see the **Issue Simultaneous Commands** section below.

6. Toggle the **Command Request** bit.
7. Wait until the **Command Request** bit is equal to the **Command Acknowledge** bit. When they are equal, the RMC75 has received the command.

Note:
Until the **Command Acknowledge** bit matches the **Command Request** bit, the Input Data registers, including the Status Bits registers, do not reflect having received the command.

Example

A Move Absolute (20) command is issued using the PROFIBUS Command Block. Until the **Command Request**bit matches the **Command Acknowledge**bit after the **Command Request**bit has been toggled, the **In Position**bit should not be checked as it may still be set for the previously requested move. Once the **Acknowledge**toggles to match, the **In Position**bit will have been cleared and when it is set, it is due to the new command being complete. Similar synchronization issues are resolved in the same way with other status bits and registers.

Issue Simultaneous Commands

Although only one command may be sent at a time to the RMC75P via PROFIBUS, it is possible to simultaneously issue different commands to several axes by using deferred commands. Deferred commands are stored in the PROFIBUS command buffer until all deferred commands are received. They are then executed simultaneously. Bits 30 and 29 in the Command Data Register of the Command Block define the deferred status of each command issued. The bits are used as follows:

Bit 30	Bit 29	Action
0	0	**Single Command:** When both bits are zero, the command is not deferred. The command is executed normally. If the PROFIBUS command buffer contains any commands, an error is logged in the Event Log and the commands are removed from the command buffer without being executed. The new command is still issued.
0	1	**Last Deferred:** This command and any deferred commands in the PROFIBUS command buffer are executed simultaneously.
1	0	**First Deferred:** This command is placed as a deferred command in the PROFIBUS command buffer, but is not otherwise processed. If the command buffer already contains commands, an error is logged in the Event Log and the commands are removed from the command buffer without being executed. The new deferred command is still placed in the command buffer.
1	1	**Middle Deferred:** This command is placed as a deferred command in the PROFIBUS command buffer, but is not otherwise processed. This deferred command type allows other deferred commands to be in the command buffer, although they are not required to be there. Note that for a 2-axis controller, this deferred setting will not be used because there can only be a first and last deferred command.

Multiple deferred commands cannot be issued to the same axis. That is, if a deferred command is issued to an axis that already has a deferred command, an error is logged in the Event Log and the previous command is overwritten without being executed.

Read from the RMC75

The Response Block only returns the values from 8 registers, which must be determined when setting up the communications. However, it is possible to set up one of the registers in the Response Block to return the value of a read from *any* single register in the RMC75.

When a read is requested from any single register in the RMC75, the response from this single-register read will be placed in the Axis 0 Read Response register. In order to see the response from the PROFIBUS, you must map the Axis 0 Read Register into one of the Indirect Data Map registers.

Notice that the copy from the requested register into the Axis 0 Read Response register only occurs once, and therefore you will not see the value continuously updating like the other Response Block registers.

To read any single register from the RMC, use the following steps:

- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Clear the **Read/Write** bit.
- Set the Read/Write Address File and Read/Write Address Element. For example, for address %MD8.12, the file is 8, and the element is 12. See the RMC75 Register Map topic for a description of all RMC75 registers and their addresses.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 will have updated the Axis 0 Read Response register with the requested data, and the corresponding Response Block register.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the read address and **Read/Write** bit before toggling the **Read/Write Request** bit.
- **Do not** change the **Read/Write Request** bit after a read request until you have processed the data in the Read Response register.
- **Do not** change the read address or **Read/Write** bit when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Write to the RMC75

To write to the RMC75, use the following steps:

- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Copy the value you wish to write to the RMC75 into the Write Value register (7) of the Command Block.
- Enter the Read/Write Address file and element. For example, for address %MD56.0, the file is 56, and the element is 0. See RMC75 Register Map topic for a description of all RMC75 registers and their addresses.
- Set the **Read/Write** bit.
- Toggle the **Read/Write Request** bit.
- Wait until the **Write Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 has received the data written to it.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the **Read/Write** bit, write address, and write value before toggling the **Read/Write Request** bit.
- **Do not** change the **Read/Write** bit, write address, or write value when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Note:
The RMC75 sets the **Read/Write Acknowledge** bit equal to the **Read/Write Request** to acknowledge that the write was processed. In addition, the RMC75 also places the write value in the Read Response register. This provides a simple method of verifying that the write was completed.

Debugging

Using the Event Log for PROFIBUS

The Event Log can record every change in the PROFIBUS data received by the RMC75P. This is the data in the Command Block. It does not record the data in the Response Block, which is sent by the RMC75P. The Event log displays the received data in hexadecimal format.

The Event Log can log an entry when any of the following occurs:

- **Data is Initialized**
  (the **Configuration Information** box must be checked in the Event Log filter for PROFIBUS)
  This typically occurs when the RMC75 is restarted. The Event Log entry will be labeled **initial data**. It provides the user with a reference of what the initial data is.
  For example, assume a user wrote a 1 to the Command Request bit to issue a command immediately after starting the PROFIBUS communications, but the command was not issued. The user then looked in the Event Log and found out that the initial data showed that the Command Request bit already was 1, which explains why the command was not issued. The bit must be toggled to send a command, so he should have written a 0.
- **Data Changes**
  (the **Command Channel Logging** and **Data Channel Logging** boxes must be set to **All** in the Event Log filter for PROFIBUS)
  Each time the data in the Command Block changes, the data will be logged with an entry labeled **changed**. The entry will also specify which data changed, as described below.

- **A Request is Made**
  (the **Command Channel Logging** and **Data Channel Logging** boxes must be set to **All or Requests** in the Event Log filter for PROFIBUS)
  Each time a Command Request bit or Read/Write Request bit is toggled, the entry will be labeled **request made**. This indicates a command was requested to be issued, or a read or write was made.

  Tip:
  In some cases, a request can be made, but nothing happens. This is probably caused by one of the following:
  - A command was requested, but no Selected Axis bit was set.
  - A read or write of multiple registers was requested, but the Count was set to 0.

The Event Log labels the logged entries in the following manner:

- **Command Area**
  These 6 registers are the data used to issue commands to the RMC75.
- **Data Channel 0**
  These 2 registers contain the data for reading or writing a single RMC75 register. Data Channel 1 is not used with Basic Mode.

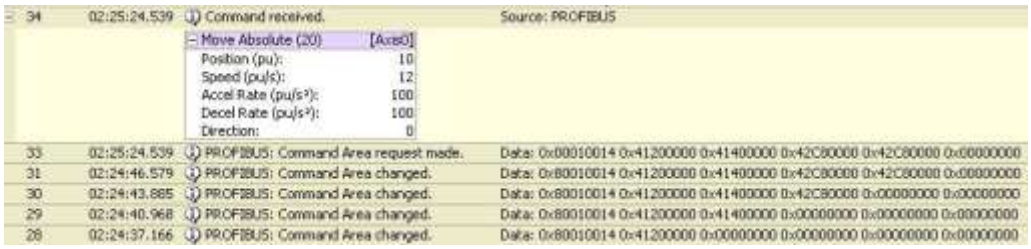
Debugging the Command Area Data

The Event Log displays the Command Area data in the following order:

Cmd Register, Cmd Parameter 1, Cmd Parameter 2, Cmd Parameter 3, Cmd Parameter 4, Cmd Parameter 5

Example:

Assume a Move Absolute command has been issued to the RMC75 via PROFIBUS. The Event Log may look like this:



Steps 28-31 show how the command parameters are changing. The command word shows the command that will be issued, (hexadecimal 14 is 20 in decimal), and the command select bit (the 1 in the middle of the word).

In step 33, bit 31 of the Command Register changed, which then issued the move command.

Debugging Data Channel 0

The Event Log displays the Data Channel 0 data in the following order:

Read/Write Register (register 6), Explicit Write Value (register 7)

Example:

Assume a value of 46.2 was written to %MD56.0 via PROFIBUS. The Event Log may look like this:

Step 38 shows that the File is 56 (38 in hexadecimal), bit 31 is set to 1 for a write. The Explicit Write register contains the write value 46.2, but it is very difficult to decipher a float value from its hexadecimal representation.

Step 39 shows that the Read/Writer Request bit changed, which requested the write.

See Also

Basic/Enhanced PROFIBUS Modes | PROFIBUS Configuration | PROFIBUS-DP Overview

6.8.5.3. PROFIBUS Mode: Basic+

The Basic+ mode is one of the Basic/Enhanced PROFIBUS Modes available only on the RMC75P. Most users will prefer the I/O Modes instead.

Features

This mode has the following features:

- Sixteen (16) user-selectable cyclic read registers are continuously read for instant access by the PLC or PC.
  See **Response Block** and **Setting Up the Indirect Data Map** below.
- One (1) register anywhere in the RMC75 can be explicitly written or read.
  See **Read from the RMC75** and **Write to the RMC75** below.
- Commands can be issued to any number of axes simultaneously.
  See **Command Block**, **Issue a Single Command**, and **Issue Simultaneous Commands** below.
- PROFIBUS bandwidth used: 16 consistent I/O words (16 bits each), plus 16 consistent Input words (16 bits each).

RMC75 Register Addresses for PROFIBUS

When communicating over PROFIBUS, the RMC75 registers addresses use the same *file*:*element* addresses as the RMC75 IEC-61131 addresses.

For example, to read the Axis 0 Actual Position via PROFIBUS, notice that its address is %MD8.8. Therefore, the address via PROFIBUS is file 8, element 8.

Parameterization

Basic+ mode requires the PROFIBUS configuration and parameterization listed below. The GSD file does direct the PROFIBUS master setup software to automatically set up these values, but Delta has found several cases where it is not supported correctly or manual setup is otherwise required.

Configuration: FF DF

Parameters:

Prm_Data (bytes 1-7): See the PROFIBUS DP specification for details.

User_Prm_Data (bytes 8-14): 00 00 00 xx* 00 10

* The xx parameter can be 00 or 01 and selects whether the least-significant word comes first (00) or most-significant word comes first (01).

Data Blocks

The Basic+ mode uses two fixed-length blocks of data: the **Command Block** and the **Response Block**.

Command Block

The Command block is a block of 8 contiguous 32-bit output registers. These registers are sent from the PLC or PC to the RMC.

The Command Block has the following structure:

<table>
<tr><th>Register Number</th><th>Data Type</th><th>Description</th></tr>
<tr><td colspan="3">Command Area:
Registers 0 - 5 are used for issuing commands to the RMC75. See Issue a Single Command and Issue Simultaneous Commands below for details on using these registers.</td></tr>
<tr><td>0</td><td>Integer</td><td>Command Register
<table><tr><th>Bit</th><th>Bit Description</th></tr><tr><td>31</td><td>Command Request</td></tr><tr><td>30</td><td>Deferred Command</td></tr><tr><td>29</td><td>Deferred Command</td></tr><tr><td>20-28</td><td>Reserved</td></tr></table></td></tr>
</table>

		19 Axis 3 Select 18 Axis 2 Select 17 Axis 1 Select 16 Axis 0 Select 8-15 Reserved 7-0 Command Number
1	Float	Command Parameter 1
2	Float	Command Parameter 2
3	Float	Command Parameter 3
4	Float	Command Parameter 4
5	Float	Command Parameter 5

Data Channel 0:
Registers 6 and 7 are used for reading and writing to any register in the RMC75. See **Read from the RMC75** and **Write to the RMC75** for details on using these registers.

6	Integer	Read/Write Register **Bit** **Bit Description** 31 Read/Write 30 Read/Write Request 16-29 Reserved 15-8 R/W Address File 7-0 R/W Address Element
7	Float*	Explicit Write Value

*These registers are typically REAL data type (floating point), but in some cases may be DINT or DWORD integers, such as variables declared as such.

Response Block

The Response Block is a block of 16 contiguous 32-bit input registers (cyclic read registers), corresponding to the Indirect Data Map registers 0 to 15 in the RMC75. These registers are continuously sent from the RMC75 to the PLC or PC. See **Configuring the Data** below for details on setting up this data.

Each of the 16 contiguous input registers always reads from the same Indirect Data Map register in the RMC75, as listed in the **Description** column below. However, one of the registers in the Response Block can be set up to return the value of a read from *any* single register in the RMC75. This allows you to read the value of any single register at any time. See **Read from the RMC75** below.

The Response Block has the following structure:

Register Number	Data Type	Description
0	Integer	Indirect Data 0 - ***must be Axis 0 Status bits***! **Bit** **Bit Description** 31 Command Acknowledge

		30 Read/Write Acknowledge 0-29 Axis 0 Status Bits
1	Float*	Indirect Data 1
2	Float*	Indirect Data 2
3	Float*	Indirect Data 3
4	Float*	Indirect Data 4
5	Float*	Indirect Data 5
6	Float*	Indirect Data 6
7	Float*	Indirect Data 7
Note: Registers 8-15 are not consistent with registers 0-7. See explanation below.		
8	Float*	Indirect Data 8
9	Float*	Indirect Data 9
10	Float*	Indirect Data 10
11	Float*	Indirect Data 11
12	Float*	Indirect Data 12
13	Float*	Indirect Data 13
14	Float*	Indirect Data 14
15	Float*	Indirect Data 15

*These registers are typically REAL data type (floating point), but in some cases may be DINT or DWORD integers, such as variables declared as such.

A Note about PROFIBUS Consistency

Registers within a consistent block are all updated at the same time. Notice that the Response Block area is divided into two consistent blocks. Therefore, the first eight (8) registers may have been updated at a different time than the last eight (8) registers. This is important because command and read/write synchronization use the first register, and therefore only the following seven (7) registers are guaranteed to have been updated at the same time as this synchronization register.

For example, suppose a PLC issues a command to axis 1 and then needs to wait for it to get in position. To do this, the PLC must issue the command, wait for the command to be received, and finally check the axis's In Position status bit. However, if the Axis 1 Status Bits register is placed in the second block of registers, then even after the **Command Acknowledge** bit matches the **Command Request** bit, indicating that the command was received, we have no way of knowing whether the Axis 1 Status Bits register was read from the controller before or after the command was issued, and thus could provide the In Position bit from *before* the command was issued.

In short, do not put any registers that depend on a command being issued—such as axis Status Bits, Error Bits, or Command Position—or the Read Response—which is tightly coupled to the **Read/Write Acknowledge** bit in register 0—in the second block of registers.

Configuring the Data

Setting up the Indirect Data Map

The Response Block continuously returns the values from the RMC75 Indirect Data registers 0-15. These registers, in turn, can be mapped to any registers in the RMC75. Thereby, the values from the selected registers in the RMC75 can be read from and written to by writing to and reading from the Indirect Data registers.

To set up the Indirect Data Map:

1. In the Project pane, double-click **Address Maps**, then click Indirect Data Map.
2. In the **Register** column of the first Indirect Data Map entry, type "%MD8.0" and press Enter. This will map Axis 0 Status Bits register to the first item in the Indirect Data Map. Basic mode requires that the first item in the Indirect Data Map contains the Axis 0 Status Bits register.
3. For each of the remaining Indirect Data Map entries 1-15, enter the desired register to map to each. To do this, click the cell in the **Register** column, click the ellipsis button (...), then browse to the desired register.

   Note:
   Response Block registers 8-15 are not consistent with registers 0-7. Because of this, registers 8-15 should not be used for tight synchronization with registers 0-7. The following registers should not be placed in Indirect Data Map registers 8 to 15:
   - Read Response - this is tightly coupled with the **Read/Write Acknowledge** bit in register 0.
   - Status and Error bits, Actual Position, Command Position - these depend on the command being issued.
4. If you wish to add additional read capability, one of the Indirect Data Map registers should be mapped to the Read Response register. Then, the corresponding register in the Response Block will return the value of a read from *any* single register in the RMC75 at any time. See **Read from the RMC75** below.

Registers within a consistent block are all updated at the same time. Notice that the Response Block area is divided into two consistent blocks. Therefore, the first eight (8) registers may have been updated at a different time than the last eight (8) registers. This is important because command and read/write synchronization use the first register, and therefore only the following seven (7) registers are guaranteed to have been updated at the same time as this synchronization register.

For example, suppose a PLC issues a command to axis 1 and then needs to wait for it to get in position. To do this, the PLC must issue the command, wait for the command to be received, and finally check the axis's In Position status bit. However, if the Axis 1 Status Bits register is placed in the second block of registers, then even after the **Command Acknowledge** bit matches the **Command Request** bit, indicating that the command was received, we have no way of knowing whether the Axis 1 Status Bits register was read from the controller before or after the command was issued, and thus could provide the In Position bit from *before* the command was issued.

In short, do not put any registers that depend on a command being issued—such as axis Status Bits, Error Bits, or Command Position—or the Read Response—which is tightly coupled to the **Read/Write Acknowledge** bit in register 0—in the second block of registers.

Example

Requirements

First, the user lists the desired registers to read from the RMC75:

- Axis 0 Status Bits
- Axis 0 Actual Position
- Axis 1 Status Bits
- Axis 1 Actual Position
- Task 0 Current Step
- Task 1 Current Step
- The first 8 registers of the Variable Table.

In addition, the user would like to read some other registers occasionally.

Implementation

- **First**, PROFIBUS communications requires that Axis 0 Status Bits register must be in the first Response Block register, which is entry 0 in the Indirect Data Map.

- **Second**, the Read Response register is needed in order to read other registers occasionally.
- **Third**, the user determines which registers must be in registers 0-7 to preserve consistency. The rest of the registers can then be placed in the remaining registers.

The user chose to set up the Indirect Data Map like this:

	Reg #	Map To	Description	Current
0	%MD18.0	%MD8.0	Axis0 Status Bits	N/A
1	%MD18.1	%MD8.8	Axis0 Actual Position (pu)	N/A
2	%MD18.2	%MD9.0	Axis1 Status Bits	N/A
3	%MD18.3	%MD9.8	Axis1 Actual Position (pu)	N/A
4	%MD18.4	%MD24.1	Task 0 Current Program/Step	N/A
5	%MD18.5	%MD24.17	Task 1 Current Program/Step	N/A
6	%MD18.6	%MD8.4	Axis0 Read Response	N/A
7	%MD18.7	%MD56.0	0 - (NoName)	N/A
8	%MD18.8	%MD56.1	1 - (NoName)	N/A
9	%MD18.9	%MD56.2	2 - (NoName)	N/A
10	%MD18.10	%MD56.3	3 - (NoName)	N/A
11	%MD18.11	%MD56.4	4 - (NoName)	N/A
12	%MD18.12	%MD56.5	5 - (NoName)	N/A
13	%MD18.13	%MD56.6	6 - (NoName)	N/A
14	%MD18.14	%MD56.7	7 - (NoName)	N/A
15	%MD18.15			

Using the Data Blocks

Issue a Single Command

To issue a command, set up the contents of the first six registers of the Command Block, and when complete, toggle the Command Request bit in the first Command Block register.

Notice that commands with more than 5 command parameters cannot be issued via PROFIBUS. To issue such commands, include them in a user program and issue a command via PROFIBUS to start the user program.

To issue a single command to the RMC75, use the following steps:

1. Wait until the **Command Request** bit in the Command Register (0) of the Command Block is equal to the **Command Acknowledge** bit in register 0 of the Response Block. If they are not equal, the RMC is currently processing a command request.
2. Enter the command number in bits 0-7 of the Command Register (0) of the Command Block.
3. If the command has any parameters, put them in registers 1-5 of the Command Block.
4. Clear the **Deferred Command** bits.
5. Set the desired **Axis Select** bit in the Command Register. The command will be sent simultaneously to each axis you select.

> **Note:**
> Using this method, you can send a single command to multiple axes simultaneously. You cannot send different commands to multiple axes simultaneously. To send different commands to multiple axes simultaneously, see the **Issue Simultaneous Commands** section below.

6. Toggle the **Command Request** bit.
7. Wait until the **Command Request** bit is equal to the **Command Acknowledge** bit. When they are equal, the RMC75 has received the command.

NOTE:
Until the **Command Acknowledge** bit matches the **Command Request** bit, the Input Data registers, including the Status Bits registers, do not reflect having received the command.

Example

A Move Absolute (20) command is issued using the PROFIBUS Command Block. Until the **Command Request**bit matches the **Command Acknowledge**bit after the **Command Request**bit has been toggled, the **In Position**bit should not be checked as it may still be set for the previously requested move. Once the **Acknowledge**toggles to match, the **In Position**bit will have been cleared and when it is set, it is due to the new command being complete. Similar synchronization issues are resolved in the same way with other status bits and registers.

Issue Simultaneous Commands

Although only one command may be sent at a time to the RMC75P via PROFIBUS, it is possible to simultaneously issue different commands to several axes by using deferred commands. Deferred commands are stored in the PROFIBUS command buffer until all deferred commands are received. They are then executed simultaneously. Bits 30 and 29 in the Command Data Register of the Command Block define the deferred status of each command issued. The bits are used as follows:

Bit 30	Bit 29	Action
0	0	**Single Command:** When both bits are zero, the command is not deferred. The command is executed normally. If the PROFIBUS command buffer contains any commands, an error is logged in the Event Log and the commands are removed from the command buffer without being executed. The new command is still issued.
0	1	**Last Deferred:** This command and any deferred commands in the PROFIBUS command buffer are executed simultaneously.
1	0	**First Deferred:** This command is placed as a deferred command in the PROFIBUS command buffer, but is not otherwise processed. If the command buffer already contains commands, an error is logged in the Event Log and the commands are removed from the command buffer without being executed. The new deferred command is still placed in the command buffer.
1	1	**Middle Deferred:** This command is placed as a deferred command in the PROFIBUS command buffer, but is not otherwise processed. This deferred command type allows other deferred commands to be in the command buffer, although they are not required to be there. Note that for a 2-axis controller, this deferred setting will not be used because there can only be a first and last deferred command.

Multiple deferred commands cannot be issued to the same axis. That is, if a deferred command is issued to an axis that already has a deferred command, an error is logged in the Event Log and the previous command is overwritten without being executed.

Read from the RMC75

The Response Block only returns the values from 16 registers, which must be determined when setting up the communications. However, it is possible to set up one of the registers in the Response Block to return the value of a read from *any* single register in the RMC75.

When a read is requested from any single register in the RMC75, the response from this single-register read will be placed in the Axis 0 Read Response register. In order to see the response from the PROFIBUS, you must map the Axis 0 Read Register into one of the Indirect Data Map registers.

Notice that the copy from the requested register into the Axis 0 Read Response register only occurs once, and therefore you will not see the value continuously updating like the other Response Block registers.

To read any single register from the RMC75, use the following steps:

- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Clear the **Read/Write** bit.
- Set the Read/Write Address File and Read/Write Address Element. For example, for address %MD8.12, the file is 8, and the element is 12. See the RMC75 Register Map topic for a description of all RMC75 registers and their addresses.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 will have updated the Axis 0 Read Response register with the requested data, and the corresponding Response Block register.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the read address and **Read/Write** bit before toggling the **Read/Write Request** bit.
- **Do not** change the **Read/Write Request** bit after a read request until you have processed the data in the Read Response register.
- **Do not** change the read address or **Read/Write** bit when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Write to the RMC75

To write to the RMC75, use the following steps:

- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Copy the value you wish to write to the RMC75 into the Write Value register (7) of the Command Block.
- Enter the Read/Write Address file and element. For example, for address %MD56.0, the file is 56, and the element is 0. See the RMC75 Register Map topic for a description of all RMC75 registers and their addresses.
- Set the **Read/Write** bit.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 has received the data written to it.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the **Read/Write** bit, write address, and write value before toggling the **Read/Write Request** bit.

- **Do not** change the **Read/Write** bit, write address, or write value when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Note:
The RMC75 sets the **Read/Write Acknowledge** bit equal to the **Read/Write Request** to acknowledge that the write was processed. In addition, the RMC75 also places the write value in the Read Response register. This provides a simple method of verifying that the write was completed.

Debugging

Using the Event Log for PROFIBUS

The Event Log can record every change in the PROFIBUS data received by the RMC75P. This is the data in the Command Block. It does not record the data in the Response Block, which is sent by the RMC75P. The Event log displays the received data in hexadecimal format.

The Event Log can log an entry when any of the following occurs:

- **Data is Initialized**
  (the **Configuration Information** box must be checked in the Event Log filter for PROFIBUS)
  This typically occurs when the RMC75 is restarted. The Event Log entry will be labeled **initial data**. It provides the user with a reference of what the initial data is.
  For example, assume a user wrote a 1 to the Command Request bit to issue a command immediately after starting the PROFIBUS communications, but the command was not issued. The user then looked in the Event Log and found out that the initial data showed that the Command Request bit already was 1, which explains why the command was not issued. The bit must be toggled to send a command, so he should have written a 0.
- **Data Changes**
  (the **Command Channel Logging** and **Data Channel Logging** boxes must be set to **All** in the Event Log filter for PROFIBUS)
  Each time the data in the Command Block changes, the data will be logged with an entry labeled **changed**. The entry will also specify which data changed, as described below.

- **A Request is Made**
  (the **Command Channel Logging** and **Data Channel Logging** boxes must be set to **All or Requests** in the Event Log filter for PROFIBUS)
  Each time a Command Request bit or Read/Write Request bit is toggled, the entry will be labeled **request made**. This indicates a command was requested to be issued, or a read or write was made.

  Tip:
  In some cases, a request can be made, but nothing happens. This is probably caused by one of the following:
  - A command was requested, but no Selected Axis bit was set.
  - A read or write of multiple registers was requested, but the Count was set to 0.

The Event Log labels the logged entries in the following manner:

- **Command Area**
  These 6 registers are the data used to issue commands to the RMC75.
- **Data Channel 0**
  These 2 registers contain the data for reading or writing a single RMC75 register. Data Channel 1 is not used with Basic+ Mode.

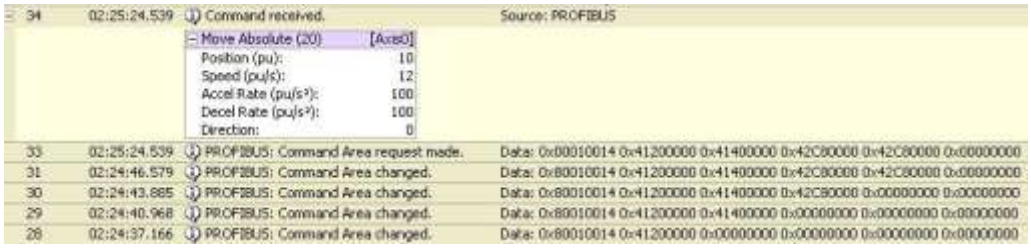
Debugging the Command Area Data

The Event Log displays the Command Area data in the following order:

Cmd Register, Cmd Parameter 1, Cmd Parameter 2, Cmd Parameter 3, Cmd Parameter 4, Cmd Parameter 5

Example:

Assume a Move Absolute command has been issued to the RMC75 via PROFIBUS. The Event Log may look like this:



Steps 28-31 show how the command parameters are changing. The command word shows the command that will be issued, (hexadecimal 14 is 20 in decimal), and the command select bit (the 1 in the middle of the word).

In step 33, bit 31 of the Command Register changed, which then issued the move command.

Debugging Data Channel 0

The Event Log displays the Data Channel 0 data in the following order:

Read/Write Register (register 6), Explicit Write Value (register 7)

Example:

Assume a value of 46.2 was written to %MD56.0 via PROFIBUS. The Event Log may look like this:

Step 38 shows that the File is 56 (38 in hexadecimal), bit 31 is set to 1 for a write. The Explicit Write register contains the write value 46.2, but it is very difficult to decipher a float value from its hexadecimal representation.

Step 39 shows that the Read/Writer Request bit changed, which requested the write.

See Also

Basic/Enhanced PROFIBUS Modes | PROFIBUS Configuration | PROFIBUS-DP Overview

6.8.5.4. PROFIBUS Mode: Enhanced

The Enhanced mode is one of the Basic/Enhanced PROFIBUS Modes available only on the RMC75P. Most users will prefer the I/O Modes instead.

Features

This mode has the following features:

- Eight (8) user-selectable cyclic read registers are continuously read for instant access by the PLC or PC.
  See **Response Block** and **Setting Up the Indirect Data Map** below.

- One (1) register anywhere in the RMC75 can be explicitly written or read.
  See **Read Any Single RMC75 Register** and **Write to Any Single RMC75 Register** below.
- Seven (7) contiguous registers in the RMC75 can be explicitly written or read.
  See **Read Any Contiguous RMC75 Registers** and **Write to Any Contiguous RMC75 Registers** below.
- Commands can be issued to any number of axes simultaneously.
  See **Command Block**, **Issue a Single Command**, and **Issue Simultaneous Commands** below.
- PROFIBUS bandwidth used: 2 blocks each of 16 consistent I/O words (16 bits each).

RMC75 Register Addresses for PROFIBUS

When communicating over PROFIBUS, the RMC75 registers addresses use the same *file*:*element* addresses as the RMC75 IEC-61131 addresses.

For example, to read the Axis 0 Actual Position via PROFIBUS, notice that its address is %MD8.8. Therefore, the address via PROFIBUS is file 8, element 8.

Parameterization

Enhanced mode requires the PROFIBUS configuration and parameterization listed below. The GSD file does direct the PROFIBUS master setup software to automatically set up these values, but Delta has found several cases where it is not supported correctly or manual setup is otherwise required.

Configuration: FF FF

Parameters:

Prm_Data (bytes 1-7): See the PROFIBUS DP specification for details.

User_Prm_Data (bytes 8-14): 00 00 00 xx* 01 08 08

* The xx parameter can be 00 or 01 and selects whether the least-significant word comes first (00) or most-significant word comes first (01).

Data Blocks

The Enhanced mode uses two fixed-length blocks of data: the **Command Block** and the **Response Block**.

Command Block

The Command block is a block of 16 contiguous 32-bit output registers. These registers are sent from the PLC or PC to the RMC.

The Command Block has the following structure:

<table>
<tr><th>Register Number</th><th>Data Type</th><th>Description</th></tr>
<tr><td colspan="3">Command Area:
Registers 0 - 5 are used for issuing commands to the RMC75. See Issue a Single Command and Issue Simultaneous Commands below for details on using these registers.</td></tr>
<tr><td>0</td><td>Integer</td><td>Command Register
<table><tr><th>Bit</th><th>Bit Description</th></tr><tr><td>31</td><td>Command Request</td></tr><tr><td>30</td><td>Deferred Command</td></tr><tr><td>29</td><td>Deferred Command</td></tr><tr><td>20-28</td><td>Reserved</td></tr></table></td></tr>
</table>

		19 Axis 3 Select 18 Axis 2 Select 17 Axis 1 Select 16 Axis 0 Select 8-15 Reserved 7-0 Command Number
1	Float	Command Parameter 1
2	Float	Command Parameter 2
3	Float	Command Parameter 3
4	Float	Command Parameter 4
5	Float	Command Parameter 5
Data Channel 0: Registers 6 and 7 are used for reading and writing to any register in the RMC75. See **Read any Single RMC75 Register** and **Write to any Single RMC75 Register** for details on using these registers.		
6	Integer	Read/Write Register **Bit** **Bit Description** 31 Read/Write 30 Read/Write Request 16-29 Reserved 15-8 R/W Address File 7-0 R/W Address Element
7	Float*	Explicit Write Value
Data Channel 1: Registers 8-15 are used for explicit reads and writes. See **Read Any Contiguous RMC75 Registers** and **Write Any Contiguous RMC75 Registers** below.		
8	Integer	Read/Write Command Register **Bit** **Bit Description** 31 Read/Write 30 Read/Write Request 24-29 Reserved 16-23 Count 15-8 R/W Address File 7-0 R/W Address Element
9	Float*	Explicit Write Value 0
10	Float*	Explicit Write Value 1
11	Float*	Explicit Write Value 2
12	Float*	Explicit Write Value 3
13	Float*	Explicit Write Value 4
14	Float*	Explicit Write Value 5

15	Float*	Explicit Write Value 6

*These registers are typically REAL data type (floating point), but in some cases may be DINT or DWORD integers, such as variables declared as such.

Response Block

The Response Block is a block of 16 contiguous 32-bit input registers (cyclic read registers). Registers 0-7 correspond to the Indirect Data Map registers 0 to 7 in the RMC75. These registers are continuously sent from the RMC to the PLC or PC. Registers 8-15 are for explicit reads or writes.

The Response Block has the following structure:

Register Number	Data Type	Description
Note: Register 0 supplies the reading and writing acknowledge for the Command Block register 6. **Note:** Registers 0-7 are **consistent**. See explanation below.		
0	Integer	Indirect Data 0 - ***must be mapped to Axis 0 Status bits***! **Bit** / **Bit Description**: 31 Command Acknowledge; 30 Read/Write Acknowledge; 0-29 Axis 0 Status Bits
1	Float*	Indirect Data 1
2	Float*	Indirect Data 2
3	Float*	Indirect Data 3
4	Float*	Indirect Data 4
5	Float*	Indirect Data 5
6	Float*	Indirect Data 6
7	Float*	Indirect Data 7
Note: Register 8 supplies the reading and writing acknowledge for registers 9-15.		
8	Integer	Read/Write Acknowledge Register. **Bit** / **Bit Description**: 30 Read/Write Acknowledge. **Note:** The entire Command Register 8 is echoed to this Read/Write Acknowledge Register. Typically, bit 30 is the only bit used.
9	Float*	Explicit Read Data 0
10	Float*	Explicit Read Data 1
11	Float*	Explicit Read Data 2

12	Float*	Explicit Read Data 3
13	Float*	Explicit Read Data 4
14	Float*	Explicit Read Data 5
15	Float*	Explicit Read Data 6

*These registers are typically REAL data type (floating point), but in some cases may be DINT or DWORD integers, such as variables declared as such.

Consistent and Non-Consistent Registers

Registers 0-7 of the Response Block are consistent. That is, they are all updated at the same time. These registers are suitable for tight synchronization with the commands, read/write requests, or each other.

Registers 8-15 of the Response block are non-consistent. That is, each register is not guaranteed to be updated at the same time as the other registers in the block, nor at the same time as the registers in the first Response Block. They are updated at the same rate as the first Response block, but each register's update may differ slightly.

Because they are non-consistent, registers 8-15 should not be used for tight synchronization. The following registers should **not** be placed in Indirect Data Map registers 8 to 15 so that they will not be in registers 8-15 of the Response Block:

- Read Response register
- Status and Error bits, because these bits are often checked immediately after a command is issued.

Configuring the Data

Setting up the Indirect Data Map

The Response Block continuously returns the values from the RMC75 Indirect Data registers 0-7. These registers, in turn, can be mapped to any registers in the RMC75. Thereby, the values from the selected registers in the RMC75 can be read from and written to by writing to and reading from the Indirect Data registers.

To set up the Indirect Data Map:

1. In the Project pane, double-click **Address Maps**, then click Indirect Data Map.
2. In the **Register** column of the first Indirect Data Map entry, type "%MD8.0" and press Enter. This will map Axis 0 Status Bits register to the first item in the Indirect Data Map. Basic mode requires that the first item in the Indirect Data Map contains the Axis 0 Status Bits register.
3. For each of the remaining Indirect Data Map entries 1-7, enter the desired register to map to each. To do this, click the cell in the **Register** column, click the ellipsis button (), then browse to the desired register.
4. If you wish to add additional read capability, one of the Indirect Data Map registers should be mapped to the Read Response register. Then, the corresponding register in the Response Block will return the value of a read from *any* single register in the RMC75 at any time. See **Read any Single RMC75 Register** below. Notice that this is probably unnecessary because this mode already enables you to read from seven contiguous registers. See **Read any Contiguous RMC75 Registers** below.

Example

Requirements

The user would like to read the following registers:

- Axis 0 Status Bits
- Axis 0 Actual Position
- Axis 1 Status Bits
- Axis 1 Actual Position

- Task 0 Current Step
- Task 1 Current Step

In addition, the user would like to read some other registers occasionally.

Implementation

- **First**, PROFIBUS communications requires that Axis 0 Status Bits register must be in the first Response Block register, which is entry 0 in the Indirect Data Map.
- **Second**, the Read Response register is needed in order to read other registers occasionally.
- **Third**, the rest of the registers listed above can be put anywhere in the remaining Indirect Data Map registers 0-7.

The user chose to set up the Indirect Data Map like this:

	Reg #	Map To	Description	Current
0	%MD18.0	%MD8.0	Axis0 Status Bits	N/A
1	%MD18.1	%MD8.8	Axis0 Actual Position (pu)	N/A
2	%MD18.2	%MD9.0	Axis1 Status Bits	N/A
3	%MD18.3	%MD9.8	Axis1 Actual Position (pu)	N/A
4	%MD18.4	%MD24.1	Task 0 Current Program/Step	N/A
5	%MD18.5	%MD24.17	Task 1 Current Program/Step	N/A
6	%MD18.6	%MD8.4	Axis0 Read Response	N/A
7	%MD18.7			

Using the Data Blocks

Issue a Single Command

To issue a command, set up the contents of the first six registers of the Command Block, and when complete, toggle the Command Request bit in the first Command Block register.

Notice that commands with more than 5 command parameters cannot be issued via PROFIBUS. To issue such commands, include them in a user program and issue a command via PROFIBUS to start the user program.

To issue a single command to the RMC75, use the following steps:

1. Wait until the **Command Request** bit in the Command Register (0) of the Command Block is equal to the **Command Acknowledge** bit in register 0 of the Response Block. If they are not equal, the RMC is currently processing a command request.
2. Enter the command number in bits 0-7 of the Command Register (0) of the Command Block.
3. If the command has any parameters, put them in registers 1-5 of the Command Block.
4. Clear the **Deferred Command** bits.
5. Set the desired **Axis Select** bit in the Command Register. The command will be sent simultaneously to each axis you select.

> **Note:**
> Using this method, you can send a single command to multiple axes simultaneously. You cannot send different commands to multiple axes simultaneously. To send different commands to multiple axes simultaneously, see the **Issue Simultaneous Commands** section below.

6. Toggle the **Command Request** bit.
7. Wait until the **Command Request** bit is equal to the **Command Acknowledge** bit. When they are equal, the RMC75 has received the command.

NOTE:
Until the **Command Acknowledge** bit matches the **Command Request** bit, the Input Data registers, including the Status Bits registers, do not reflect having received the command.

Example

A Move Absolute (20) command is issued using the PROFIBUS Command Block. Until the **Command Request**bit matches the **Command Acknowledge**bit after the **Command Request**bit has been toggled, the **In Position**bit should not be checked as it may still be set for the previously requested move. Once the **Acknowledge**toggles to match, the **In Position**bit will have been cleared and when it is set, it is due to the new command being complete. Similar synchronization issues are resolved in the same way with other status bits and registers.

Issue Simultaneous Commands

Although only one command may be sent at a time to the RMC75P via PROFIBUS, it is possible to simultaneously issue different commands to several axes by using deferred commands. Deferred commands are stored in the PROFIBUS command buffer until all deferred commands are received. They are then executed simultaneously. Bits 30 and 29 in the Command Data Register of the Command Block define the deferred status of each command issued. The bits are used as follows:

Bit 30	Bit 29	Action
0	0	**Single Command:** When both bits are zero, the command is not deferred. The command is executed normally. If the PROFIBUS command buffer contains any commands, an error is logged in the Event Log and the commands are removed from the command buffer without being executed. The new command is still issued.
0	1	**Last Deferred:** This command and any deferred commands in the PROFIBUS command buffer are executed simultaneously.
1	0	**First Deferred:** This command is placed as a deferred command in the PROFIBUS command buffer, but is not otherwise processed. If the command buffer already contains commands, an error is logged in the Event Log and the commands are removed from the command buffer without being executed. The new deferred command is still placed in the command buffer.
1	1	**Middle Deferred:** This command is placed as a deferred command in the PROFIBUS command buffer, but is not otherwise processed. This deferred command type allows other deferred commands to be in the command buffer, although they are not required to be there. Note that for a 2-axis controller, this deferred setting will not be used because there can only be a first and last deferred command.

Multiple deferred commands cannot be issued to the same axis. That is, if a deferred command is issued to an axis that already has a deferred command, an error is logged in the Event Log and the previous command is overwritten without being executed.

Read Any Single RMC75 Register

Registers 0-7 of the Response Block return the values from 8 registers, which must be determined when setting up the communications. However, it is possible to set up one of the

registers 1-7 in the Response Block to return the value of a read from *any* single register in the RMC75.

When a read is requested from any single register in the RMC75, the response from this single-register read will be placed in the Axis 0 Read Response register. In order to see the response from the PROFIBUS, you must map the Axis 0 Read Register into one of the Indirect Data Map registers.

Notice that the copy from the requested register into the Axis 0 Read Response register only occurs once, and therefore you will not see the value continuously updating like the other Response Block registers.

To read any single register from the RMC75, use the following steps:

- Wait until the Response Block register 0 **Read/Write Request** bit is equal to the Command Block register 6 **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Clear the Command Block register 6 **Read/Write** bit.
- Set the Command Block Register 6 Read/Write Address File and Read/Write Address Element. For example, for address %MD8.12, the file is 8, and the element is 12. For a description of all RMC75 registers and their addresses, see the RMC75 Register Map topic.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 will have updated the Axis 0 Read Response register with the requested data, and the corresponding Response Block register.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the read address and **Read/Write** bit before toggling the **Read/Write Request** bit.
- **Do not** change the **Read/Write Request** bit after a read request until you have processed the data in the Read Response register.
- **Do not** change the read address or **Read/Write** bit when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Read Any Contiguous RMC75 Registers

To read any contiguous RMC75 registers, use the Command Block register 8 and the Response Block registers 8-15.

To read any contiguous RMC75 registers, use the following steps:

- Wait until the Command Block register 8 **Read/Write Request** bit is equal to the Response Block register 8 **Read/Write Acknowledge** bit. If they are not equal, the RMC is currently processing a read or write request.
- Clear the Command Block register 8 **Read/Write** bit.
- In the Command Block register 8, set the Read/Write Address File and Read/Write Address Element to the first RMC75 address you wish to read. For example, for address %MD8.12, the file is 8, and the element is 12. Set the Count to the number of register to read, up to 7. For a description of all RMC75 registers and their addresses, see the RMC75 Register Map topic.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 will have updated the Response Block registers 8-15 with the requested data.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the read address and **Read/Write** bit before toggling the **Read/Write Request** bit.

- **Do not** change the **Read/Write Request** bit after a read request until you have processed the data in the Read Response register.
- **Do not** change the read address or **Read/Write** bit when the **Read/Write** Request bit does not match the **Read/Write Acknowledge** bit.

Write to Any Single RMC75 Register

To write to a single RMC75 register, use the Command Block register 6 and the Response Block register 0. Register 0 of the RMC75 Indirect Data Map must be mapped to the Axis 0 Status Bits register.

To write to the RMC75, use the following steps:

- Wait until the Command Block register 6 **Read/Write Request** bit is equal to the Response Block register 0 **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Copy the value you wish to write to the RMC75 into the Write Value register (7) of the Command Block.
- In the Command Block register 6, enter the Read/Write Address file and element. For example, for address %MD56.0, the file is 56, and the element is 0. For a description of all RMC75 registers and their addresses, see the RMC75 Register Map topic.
- Set the Command Block register 6 **Read/Write** bit.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 has received the data written to it.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the **Read/Write** bit, write address, and write value before toggling the **Read/Write Request** bit.
- **Do not** change the **Read/Write** bit, write address, or write value when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Note:
The RMC75 sets the **Read/Write Acknowledge** bit equal to the **Read/Write Request** to the acknowledge that the write was processed. In addition, the RMC75 also places the write value in the Read Response register. This provides a simple method of verifying that the write was completed.

Write to Any Contiguous RMC75 Registers

To write to any contiguous RMC75 registers, use Command Block registers 8-15 and Response Block register 8. To write to the RMC75, use the following steps:

- Wait until the Command Block register 8 **Read/Write Request** bit is equal to the Response Block register 8 **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Set the Command Block register 8 **Read/Write** bit.
- In the Command Block register 8, set the Read/Write Address File and Read/Write Address to the first RMC75 address you wish to write to. Set the Count to the number of register to write, up to 7. For a description of all RMC75 registers and their addresses, see the RMC75 Register Map topic.
- In the Command Block registers 9-15, put the values you wish to write.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 has received the data written to it.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the **Read/Write** bit, write address, and write value before toggling the **Read/Write Request** bit.
- **Do not** change the **Read/Write** bit, write address, or write value when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Debugging

Using the Event Log for PROFIBUS

The Event Log can record every change in the PROFIBUS data received by the RMC75P. This is the data in the Command Block. It does not record the data in the Response Block, which is sent by the RMC75P. The Event log displays the received data in hexadecimal format.

The Event Log can log an entry when any of the following occurs:

- **Data is Initialized**
  (the **Configuration Information** box must be checked in the Event Log filter for PROFIBUS)
  This typically occurs when the RMC75 is restarted. The Event Log entry will be labeled **initial data**. It provides the user with a reference of what the initial data is.
  For example, assume a user wrote a 1 to the Command Request bit to issue a command immediately after starting the PROFIBUS communications, but the command was not issued. The user then looked in the Event Log and found out that the initial data showed that the Command Request bit already was 1, which explains why the command was not issued. The bit must be toggled to send a command, so he should have written a 0.
- **Data Changes**
  (the **Command Channel Logging** and **Data Channel Logging** boxes must be set to **All** in the Event Log filter for PROFIBUS)
  Each time the data in the Command Block changes, the data will be logged with an entry labeled **changed**. The entry will also specify which data changed, as described below.
- **A Request is Made**
  (the **Command Channel Logging** and **Data Channel Logging** boxes must be set to **All or Requests** in the Event Log filter for PROFIBUS)
  Each time a Command Request bit or Read/Write Request bit is toggled, the entry will be labeled **request made**. This indicates a command was requested to be issued, or a read or write was made.

  Tip:
  In some cases, a request can be made, but nothing happens. This is probably caused by one of the following:
  - A command was requested, but no Selected Axis bit was set.
  - A read or write of multiple registers was requested, but the Count was set to 0.

The Event Log labels the logged entries in the following manner:

- **Command Area**
  These 6 registers are the data used to issue commands to the RMC75.
- **Data Channel 0**
  These 2 registers contain the data for reading or writing a single RMC75 register.
- **Data Channel 1**
  These 8 registers contain the data for reading or writing to contiguous RMC75 registers.

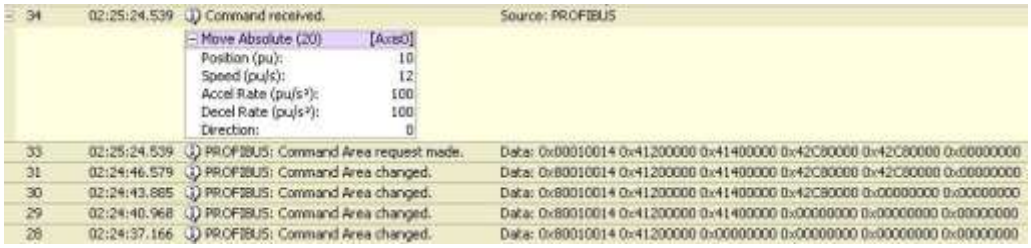
Debugging the Command Area Data

The Event Log displays the Command Area data in the following order:

Cmd Register, Cmd Parameter 1, Cmd Parameter 2, Cmd Parameter 3, Cmd Parameter 4, Cmd Parameter 5

Example:

Assume a Move Absolute command has been issued to the RMC75 via PROFIBUS. The Event Log may look like this:



Steps 28-31 show how the command parameters are changing. The command word shows the command that will be issued, (hexadecimal 14 is 20 in decimal), and the command select bit (the 1 in the middle of the word).

In step 33, bit 31 of the Command Register changed, which then issued the move command.

Debugging Data Channel 0

The Event Log displays the Data Channel 0 data in the following order:

Read/Write Register (register 6), Explicit Write Value (register 7)

Example:

Assume a value of 46.2 was written to %MD56.0 via PROFIBUS. The Event Log may look like this:

Step 38 shows that the File is 56 (38 in hexadecimal), bit 31 is set to 1 for a write. The Explicit Write register contains the write value 46.2, but it is very difficult to decipher a float value from its hexadecimal representation.

Step 39 shows that the Read/Writer Request bit changed, which requested the write.

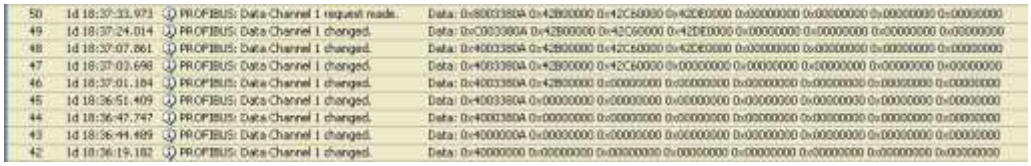
Debugging Data Channel 1

The Event Log displays the Data Channel 1 data in the following order:

Read/Write Command Register (register 8), Explicit Write Value 0, Explicit Write Value 1, Explicit Write Value 2, Explicit Write Value 3, Explicit Write Value 4, Explicit Write Value 5, Explicit Write Value 6

Example:

Assume 3 values were written to 3 registers beginning at %MD56.10. The Event Log may look like this:



In step 43, the Element is 10 (A in hexadecimal).

In step 44, the File is set to 56 (38 in hexadecimal).

In step 45, the count is set to3.

In steps 46 to 48, the write values are entered.

In step 49, bit 31 is set to 1 for a write.

In step 50, bit 30 is toggled to request the write.

See Also

Basic/Enhanced PROFIBUS Modes | PROFIBUS Configuration | PROFIBUS-DP Overview

6.8.5.5. PROFIBUS Mode: Enhanced+

The Enhanced+ mode is one of the Basic/Enhanced PROFIBUS Modes available only on the RMC75P. Most users will prefer the I/O Modes instead.

Features

This mode has the following features:

- Sixteen (16) user-selectable cyclic read registers are continuously read for instant access by the PLC or PC.
  See the **Response Block** and **Setting Up the Indirect Data Map** sections below.
- One (1) register anywhere in the RMC75 can be explicitly written or read.
  See **Read Any Single RMC75 Register** and **Write to Any Single RMC75 Register** below.
- Seven (7) contiguous registers in the RMC75 can be explicitly written or read.
  See **Read Any Contiguous RMC75 Register** and **Write to Any Contiguous RMC75 Register** below.
- Commands can be issued to any number of axes simultaneously.
  See **Command Block**, **Issue a Single Command**, and **Issue Simultaneous Commands** below.
- PROFIBUS bandwidth used: 2 blocks of 16 consistent I/O words (16 bits each), plus 16 consistent Input words (16 bits each).

RMC75 Register Addresses for PROFIBUS

When communicating over PROFIBUS, the RMC75 registers addresses use the same *file*:*element* addresses as the RMC75 IEC-61131 addresses.

For example, to read the Axis 0 Actual Position via PROFIBUS, notice that its address is %MD8.8. Therefore, the address via PROFIBUS is file 8, element 8.

Parameterization

Enhanced+ mode requires the PROFIBUS configuration and parameterization listed below. The GSD file does direct the PROFIBUS master setup software to automatically set up these values, but Delta has found several cases where it is not supported correctly or manual setup is otherwise required.

Configuration: FF DF FF

Parameters:

Prm_Data (bytes 1-7): See the PROFIBUS DP specification for details.

User_Prm_Data (bytes 8-14): 00 00 00 xx* 01 10 08

* The xx parameter can be 00 or 01 and selects whether the least-significant word comes first (00) or most-significant word comes first (01).

Data Blocks

The Enhanced+ mode uses two fixed-length blocks of data: the **Command Block** and the **Response Block**.

Command Block

The Command block is a block of 16 contiguous 32-bit output registers. These registers are sent from the PLC or PC to the RMC.

The Command Block has the following structure:

Register Number	Data Type	Description
Command Area: Registers 0 - 5 are used for issuing commands to the RMC75. See **Issue a Single Command** and **Issue Simultaneous Commands** below for details on using these registers.		
0	Integer	Command Register **Bit** — **Bit Description** 31 — Command Request 30 — Deferred Command 29 — Deferred Command 20-28 — Reserved 19 — Axis 3 Select 18 — Axis 2 Select 17 — Axis 1 Select 16 — Axis 0 Select 8-15 — Reserved 7-0 — Command Number
1	Float	Command Parameter 1
2	Float	Command Parameter 2
3	Float	Command Parameter 3
4	Float	Command Parameter 4
5	Float	Command Parameter 5
Data Channel 0: Registers 6 and 7 are used for reading and writing to any register in the RMC75. See **Read any Single RMC75 Register** and **Write to any Single RMC75 Register** for details on using these registers.		
6	Integer	Read/Write Register **Bit** — **Bit Description** 31 — Read/Write 30 — Read/Write Request 16-29 — Reserved 15-8 — R/W Address File 7-0 — R/W Address Element
7	Float*	Explicit Write Value

Data Channel 1: Registers 8-15 are used for explicit reads and writes. See **Read Any Contiguous RMC75 Registers** and **Write Any Contiguous RMC75 Registers** below.		
8	Integer	Read/Write Command Register **Bit** **Bit Description** 31 Read/Write 30 Read/Write Request 24-29 Reserved 16-23 Count 15-8 R/W Address File 7-0 R/W Address Element
9	Float*	Explicit Write Value 0
10	Float*	Explicit Write Value 1
11	Float*	Explicit Write Value 2
12	Float*	Explicit Write Value 3
13	Float*	Explicit Write Value 4
14	Float*	Explicit Write Value 5
15	Float*	Explicit Write Value 6

*These registers are typically REAL data type (floating point), but in some cases may be DINT or DWORD integers, such as variables declared as such.

Response Block

The Response Block is a block of 24 contiguous 32-bit input registers (cyclic read registers). Registers 0-15 correspond to the Indirect Data Map registers 0 to 15 in the RMC75. These registers are continuously sent from the RMC75 to the PLC or PC. Registers 16-23 are for explicit reads or writes.

The Response Block has the following structure:

Register Number	Data Type	Description
Note: Register 0 supplies the reading and writing acknowledge for the Command Block register 6.		
0	Integer	Indirect Data 0 - ***must be mapped to Axis 0 Status bits***! **Bit** **Bit Description** 31 Command Acknowledge 30 Read/Write Acknowledge 0-29 Axis 0 Status Bits
1	Float*	Indirect Data 1
2	Float*	Indirect Data 2
3	Float*	Indirect Data 3
4	Float*	Indirect Data 4

5	Float*	Indirect Data 5
6	Float*	Indirect Data 6
7	Float*	Indirect Data 7
Note: Registers 8-15 are not consistent with registers 0-7. See explanation below.		
8	Float*	Indirect Data 8
9	Float*	Indirect Data 9
10	Float*	Indirect Data 10
11	Float*	Indirect Data 11
12	Float*	Indirect Data 12
13	Float*	Indirect Data 13
14	Float*	Indirect Data 14
15	Float*	Indirect Data 15
Note: Register 16 supplies the reading and writing acknowledge for registers 17-23.		
16	Integer	Read/Write Acknowledge Register **Bit** **Bit Description** 30 Read/Write Acknowledge **Note:** The entire Command Register 8 is echoed to this Read/Write Acknowledge Register. Typically, bit 30 is the only bit used.
17	Float*	Explicit Read Data 0
18	Float*	Explicit Read Data 1
19	Float*	Explicit Read Data 2
20	Float*	Explicit Read Data 3
21	Float*	Explicit Read Data 4
22	Float*	Explicit Read Data 5
23	Float*	Explicit Read Data 6

*These registers are typically REAL data type (floating point), but in some cases may be DINT or DWORD integers, such as variables declared as such.

6.8.5.5.1.1. A Note about PROFIBUS Consistency

Registers within a consistent block are all updated at the same time. Notice that the Response Block area is divided into two consistent blocks. Therefore, the first eight (8) registers may have been updated at a different time than the last eight (8) registers. This is important because command and read/write synchronization use the first register, and therefore only the following seven (7) registers are guaranteed to have been updated at the same time as this synchronization register.

For example, suppose a PLC issues a command to axis 1 and then needs to wait for it to get in position. To do this, the PLC must issue the command, wait for the command to be received, and finally check the axis's In Position status bit. However, if the Axis 1 Status Bits register is placed in the second block of registers, then even after the **Command Acknowledge** bit matches the **Command Request** bit, indicating that the command was received, we have no way of knowing

whether the Axis 1 Status Bits register was read from the controller before or after the command was issued, and thus could provide the In Position bit from *before* the command was issued.

In short, do not put any registers that depend on a command being issued—such as axis Status Bits, Error Bits, or Command Position—or the Read Response—which is tightly coupled to the **Read/Write Acknowledge** bit in register 0—in the second block of registers.

Configuring the Data

Setting up the Indirect Data Map

The Response Block continuously returns the values from the RMC75 Indirect Data registers 0-15. These registers, in turn, can be mapped to any registers in the RMC75. Thereby, the values from the selected registers in the RMC75 can be read from and written to by writing to and reading from the Indirect Data registers.

To set up the Indirect Data Map:

1. In the Project pane, double-click **Address Maps**, then click Indirect Data Map.
2. In the **Register** column of the first Indirect Data Map entry, type "%MD8.0" and press Enter. This will map Axis 0 Status Bits register to the first item in the Indirect Data Map. Basic mode requires that the first item in the Indirect Data Map contains the Axis 0 Status Bits register.
3. For each of the remaining Indirect Data Map entries 1-15, enter the desired register to map to each. To do this, click the cell in the **Register** column, click the ellipsis button (...), then browse to the desired register.

   Note:
   Response Block registers 8-15 are not consistent with registers 0-7. Because of this, registers 8-15 should not be used for tight synchronization with registers 0-7. The following registers should not be placed in Indirect Data Map registers 8 to 15:
   - Read Response, because it is tightly coupled with the **Read/Write Acknowledge** bit in register 0.
   - Status and Error bits, Actual Position, Command Position, because these depend on the command being issued.
4. If you wish to add additional read capability, one of the Indirect Data Map registers should be mapped to the Read Response register. Then, the corresponding register in the Response Block will return the value of a read from *any* single register in the RMC75 at any time. See **Read any Single RMC75 Register** below. Notice that this is probably unnecessary because this mode already enables you to read from seven contiguous registers. See **Read any Contiguous RMC75 Registers** below.

Example

Requirements

First, the user lists the desired registers to read from the RMC75:

- Axis 0 Status Bits
- Axis 0 Actual Position
- Axis 1 Status Bits
- Axis 1 Actual Position
- Task 0 Current Step
- Task 1 Current Step
- The first 8 registers of the Variable Table.

In addition, the user would like to read some other registers occasionally.

Implementation

- **First**, PROFIBUS communications requires that Axis 0 Status Bits register must be in the first Response Block register, which is entry 0 in the Indirect Data Map.

- **Second**, the Read Response register is needed in order to read other registers occasionally.
- **Third**, the user determines which registers must be in registers 0-7 to preserve consistency. The rest of the registers can then be placed in the remaining registers.

The user chose to set up the Indirect Data Map like this:

	Reg #	Map To	Description	Current
0	%MD18.0	%MD8.0	Axis0 Status Bits	N/A
1	%MD18.1	%MD8.8	Axis0 Actual Position (pu)	N/A
2	%MD18.2	%MD9.0	Axis1 Status Bits	N/A
3	%MD18.3	%MD9.8	Axis1 Actual Position (pu)	N/A
4	%MD18.4	%MD24.1	Task 0 Current Program/Step	N/A
5	%MD18.5	%MD24.17	Task 1 Current Program/Step	N/A
6	%MD18.6	%MD8.4	Axis0 Read Response	N/A
7	%MD18.7	%MD56.0	0 - (NoName)	N/A
8	%MD18.8	%MD56.1	1 - (NoName)	N/A
9	%MD18.9	%MD56.2	2 - (NoName)	N/A
10	%MD18.10	%MD56.3	3 - (NoName)	N/A
11	%MD18.11	%MD56.4	4 - (NoName)	N/A
12	%MD18.12	%MD56.5	5 - (NoName)	N/A
13	%MD18.13	%MD56.6	6 - (NoName)	N/A
14	%MD18.14	%MD56.7	7 - (NoName)	N/A
15	%MD18.15			

Using the Data Blocks

Issue a Single Command

To issue a command, set up the contents of the first six registers of the Command Block, and when complete, toggle the Command Request bit in the first Command Block register.

Notice that commands with more than 5 command parameters cannot be issued via PROFIBUS. To issue such commands, include them in a user program and issue a command via PROFIBUS to start the user program.

To issue a single command to the RMC75, use the following steps:

1. Wait until the **Command Request** bit in the Command Register (0) of the Command Block is equal to the **Command Acknowledge** bit in register 0 of the Response Block. If they are not equal, the RMC is currently processing a command request.
2. Enter the command number in bits 0-7 of the Command Register (0) of the Command Block.
3. If the command has any parameters, put them in registers 1-5 of the Command Block.
4. Clear the **Deferred Command** bits.
5. Set the desired **Axis Select** bit in the Command Register. The command will be sent simultaneously to each axis you select.

> **Note:**
> Using this method, you can send a single command to multiple axes simultaneously. You cannot send different commands to multiple axes simultaneously. To send different commands to multiple axes simultaneously, see the **Issue Simultaneous Commands** section below.

6. Toggle the **Command Request** bit.
7. Wait until the **Command Request** bit is equal to the **Command Acknowledge** bit. When they are equal, the RMC75 has received the command.

NOTE:
Until the **Command Acknowledge** bit matches the **Command Request** bit, the Input Data registers, including the Status Bits registers, do not reflect having received the command.

Example

A Move Absolute (20) command is issued using the PROFIBUS Command Block. Until the **Command Request**bit matches the **Command Acknowledge**bit after the **Command Request**bit has been toggled, the **In Position**bit should not be checked as it may still be set for the previously requested move. Once the **Acknowledge**toggles to match, the **In Position**bit will have been cleared and when it is set, it is due to the new command being complete. Similar synchronization issues are resolved in the same way with other status bits and registers.

Issue Simultaneous Commands

Although only one command may be sent at a time to the RMC75P via PROFIBUS, it is possible to simultaneously issue different commands to several axes by using deferred commands. Deferred commands are stored in the PROFIBUS command buffer until all deferred commands are received. They are then executed simultaneously. Bits 30 and 29 in the Command Data Register of the Command Block define the deferred status of each command issued. The bits are used as follows:

Bit 30	Bit 29	Action
0	0	**Single Command:** When both bits are zero, the command is not deferred. The command is executed normally. If the PROFIBUS command buffer contains any commands, an error is logged in the Event Log and the commands are removed from the command buffer without being executed. The new command is still issued.
0	1	**Last Deferred:** This command and any deferred commands in the PROFIBUS command buffer are executed simultaneously.
1	0	**First Deferred:** This command is placed as a deferred command in the PROFIBUS command buffer, but is not otherwise processed. If the command buffer already contains commands, an error is logged in the Event Log and the commands are removed from the command buffer without being executed. The new deferred command is still placed in the command buffer.
1	1	**Middle Deferred:** This command is placed as a deferred command in the PROFIBUS command buffer, but is not otherwise processed. This deferred command type allows other deferred commands to be in the command buffer, although they are not required to be there. Note that for a 2-axis controller, this deferred setting will not be used because there can only be a first and last deferred command.

Multiple deferred commands cannot be issued to the same axis. That is, if a deferred command is issued to an axis that already has a deferred command, an error is logged in the Event Log and the previous command is overwritten without being executed.

Read Any Single RMC75 Register

Registers 0-15 of the Response Block return the values from 16 registers, which must be determined when setting up the communications. However, it is possible to set up one of the registers 1-7 in the Response Block to return the value of a read from *any* single register in the RMC75.

When a read is requested from any single register in the RMC75, the response from this single-register read will be placed in the Axis 0 Read Response register. In order to see the response from the PROFIBUS, you must map the Axis 0 Read Register into one of the Indirect Data Map registers.

Notice that the copy from the requested register into the Axis 0 Read Response register only occurs once, and therefore you will not see the value continuously updating like the other Response Block registers.

To read any single register from the RMC75, use the following steps:

- Wait until the Response Block register 0 **Read/Write Request** bit is equal to the Command Block register 6 **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Clear the Command Block register 6 **Read/Write** bit.
- Set the Command Block Register 6 Read/Write Address File and Read/Write Address Element. For example, for address %MD8.12, the file is 8, and the element is 12. For a description of all RMC75 registers and their addresses, see the RMC75 Register Map topic.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 will have updated the Axis 0 Read Response register with the requested data, and the corresponding Response Block register.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the read address and **Read/Write** bit before toggling the **Read/Write Request** bit.
- **Do not** change the **Read/Write Request** bit after a read request until you have processed the data in the Read Response register.
- **Do not** change the read address or **Read/Write** bit when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Read Any Contiguous RMC75 Registers

To read any contiguous RMC75 registers, use the Command Block register 8 and the Response Block registers 16-23.

To read any contiguous RMC75 registers, use the following steps:

- Wait until the Command Block register 8 **Read/Write Request** bit is equal to the Response Block register 16 **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Clear the Command Block register 8 **Read/Write** bit.
- In the Command Block register 8, set the Read/Write Address File and Read/Write Address Element to the first RMC75 address you wish to read. For example, for address %MD8.12, the file is 8, and the element is 12. Set the Count to the number of register to read, up to 7. For a description of all RMC75 registers and their addresses, see the RMC75 Register Map topic.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 will have updated the Response Block registers 16-23 with the requested data.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the read address and **Read/Write** bit before toggling the **Read/Write Request** bit.
- **Do not** change the **Read/Write Request** bit after a read request until you have processed the data in the Read Response register.
- **Do not** change the read address or **Read/Write** bit when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Write to Any Single RMC75 Register

To write to a single RMC75 register, use the Command Block register 6 and the Response Block register 0. Register 0 of the RMC75 Indirect Data Map must be mapped to the Axis 0 Status Bits register.

To write to the RMC75, use the following steps:

- Wait until the Command Block register 6 **Read/Write Request** bit is equal to the Response Block register 0 **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Copy the value you wish to write to the RMC75 into the Write Value register (7) of the Command Block.
- In the Command Block register 6, enter the Read/Write Address file and element. For example, for address %MD56.0, the file is 56, and the element is 0. For a description of all RMC75 registers and their addresses, see the RMC75 Register Map topic.
- Set the Command Block register 6 **Read/Write** bit.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 has received the data written to it.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the **Read/Write** bit, write address, and write value before toggling the **Read/Write Request** bit.
- **Do not** change the **Read/Write** bit, write address, or write value when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Note:
The RMC75 sets the **Read/Write Acknowledge** bit equal to the **Read/Write Request** to the acknowledge that the write was processed. In addition, the RMC75 also places the write value in the Read Response register. This provides a simple method of verifying that the write was completed.

Write to Any Contiguous RMC75 Registers

To write to any contiguous RMC75 registers, use Command Block registers 8-15 and Response Block register 16.

To write to the RMC75, use the following steps:

- Wait until the Command Block register 8 **Read/Write Request** bit is equal to the Response Block register 16 **Read/Write Acknowledge** bit. If they are not equal, the RMC75 is currently processing a read or write request.
- Set the Command Block register 8 **Read/Write** bit.
- In the Command Block register 8, set the Read/Write Address File and Read/Write Address to the first RMC75 address you wish to write to. For example, for address %MD56.0, the file is

56, and the element is 0. Set the Count to the number of register to write, up to 7. For a description of all RMC75 registers and their addresses, see the RMC75 Register Map topic.

- In the Command Block registers 9-15, put the values you wish to write.
- Toggle the **Read/Write Request** bit.
- Wait until the **Read/Write Request** bit is equal to the **Read/Write Acknowledge** bit. When they are equal, the RMC75 has received the data written to it.

To further clarify the ordering, keep these basic rules in mind:

- **Do** change the **Read/Write** bit, write address, and write value before toggling the **Read/Write Request** bit.
- **Do not** change the **Read/Write** bit, write address, or write value when the **Read/Write Request** bit does not match the **Read/Write Acknowledge** bit.

Debugging

Using the Event Log for PROFIBUS

The Event Log can record every change in the PROFIBUS data received by the RMC75P. This is the data in the Command Block. It does not record the data in the Response Block, which is sent by the RMC75P. The Event log displays the received data in hexadecimal format.

The Event Log can log an entry when any of the following occurs:

- **Data is Initialized**
  (the **Configuration Information** box must be checked in the Event Log filter for PROFIBUS)
  This typically occurs when the RMC75 is restarted. The Event Log entry will be labeled **initial data**. It provides the user with a reference of what the initial data is.
  For example, assume a user wrote a 1 to the Command Request bit to issue a command immediately after starting the PROFIBUS communications, but the command was not issued. The user then looked in the Event Log and found out that the initial data showed that the Command Request bit already was 1, which explains why the command was not issued. The bit must be toggled to send a command, so he should have written a 0.
- **Data Changes**
  (the **Command Channel Logging** and **Data Channel Logging** boxes must be set to **All** in the Event Log filter for PROFIBUS)
  Each time the data in the Command Block changes, the data will be logged with an entry labeled **changed**. The entry will also specify which data changed, as described below.

- **A Request is Made**
  (the **Command Channel Logging** and **Data Channel Logging** boxes must be set to **All or Requests** in the Event Log filter for PROFIBUS)
  Each time a Command Request bit or Read/Write Request bit is toggled, the entry will be labeled **request made**. This indicates a command was requested to be issued, or a read or write was made.

  Tip:
  In some cases, a request can be made, but nothing happens. This is probably caused by one of the following:
  - A command was requested, but no Selected Axis bit was set.
  - A read or write of multiple registers was requested, but the Count was set to 0.

The Event Log labels the logged entries in the following manner:

- **Command Area**
  These 6 registers are the data used to issue commands to the RMC75.
- **Data Channel 0**
  These 2 registers contain the data for reading or writing a single RMC75 register.
- **Data Channel 1**
  These 8 registers contain the data for reading or writing to contiguous RMC75 registers.

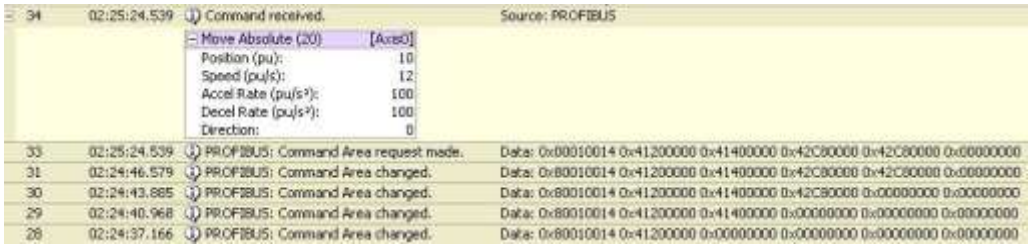
Debugging the Command Area Data

The Event Log displays the Command Area data in the following order:

Cmd Register, Cmd Parameter 1, Cmd Parameter 2, Cmd Parameter 3, Cmd Parameter 4, Cmd Parameter 5

Example:

Assume a Move Absolute command has been issued to the RMC75 via PROFIBUS. The Event Log may look like this:



Steps 28-31 show how the command parameters are changing. The command word shows the command that will be issued, (hexadecimal 14 is 20 in decimal), and the command select bit (the 1 in the middle of the word).

In step 33, bit 31 of the Command Register changed, which then issued the move command.

Debugging Data Channel 0

The Event Log displays the Data Channel 0 data in the following order:

Read/Write Register (register 6), Explicit Write Value (register 7)

Example:

Assume a value of 46.2 was written to %MD56.0 via PROFIBUS. The Event Log may look like this:

Step 38 shows that the File is 56 (38 in hexadecimal), bit 31 is set to 1 for a write. The Explicit Write register contains the write value 46.2, but it is very difficult to decipher a float value from its hexadecimal representation.

Step 39 shows that the Read/Writer Request bit changed, which requested the write.

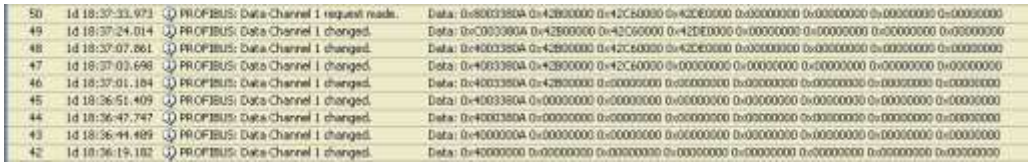
Debugging Data Channel 1

The Event Log displays the Data Channel 1 data in the following order:

Read/Write Command Register (register 8), Explicit Write Value 0, Explicit Write Value 1, Explicit Write Value 2, Explicit Write Value 3, Explicit Write Value 4, Explicit Write Value 5, Explicit Write Value 6

Example:

Assume 3 values were written to 3 registers beginning at %MD56.10. The Event Log may look like this:



In step 43, the Element is 10 (A in hexadecimal).

In step 44, the File is set to 56 (38 in hexadecimal).

In step 45, the count is set to3.

In steps 46 to 48, the write values are entered.

In step 49, bit 31 is set to 1 for a write.

In step 50, bit 30 is toggled to request the write.

See Also

Basic/Enhanced PROFIBUS Modes | PROFIBUS Configuration | PROFIBUS-DP Overview

6.9. Serial (RS-232/485) (RMC70)

Show All

6.9.1. Serial Communications Overview

Serial RS-232 or RS-485 communication is supported by the RMC75S CPU module, allowing the RMC75S to interface to other devices such as HMIs and PLCs. The RMC performs as a slave, requiring a master to control it. It always waits for a request from a master serial device before responding. It does not initiate communications.

The RMC150 does not support serial RS-232 or RS-485. However, the RMC150E can communicate to serial RS-232/485 devices via serial-to-Ethernet converters, available from manufacturers such as Lantronix (UDS100-IAP) or Allen-Bradley (1761-NET-ENI). Contact Delta for more information.

The RMC75S has two communication ports, which can be used simultaneously:

- **Monitor Port (Port 0)**
  The RMC75S Monitor Port (port 0) is intended to be used for communication with RMCTools, but may also be used with other devices. This port supports only RS-232 at fixed settings. For details, see the Monitor Port topic.
  Notice that the RMC75P also has a serial RS-232 serial Monitor port.
- **Port 1**
  Port 1 on the RMC75S module is intended to be used for communication with devices such as PLCs and HMIs. Port 1 supports either RS-232 and RS-485 (2-wire only). See the Configuring Serial Communications topic for instructions on how to configure port 1.

For basics on using serial communications, such as reading and writing registers and issuing commands, see the Using Serial Communications topic.

Features

The RMC75S serial port 1 has a great amount of flexibility, as summarized below:

- **Protocols:**
  - Allen-Bradley DF1 (Full- and Half-duplex)
  - Modbus/RTU
  - Mitsubishi Bidirectional Protocol
- **Baud Rate:** 9,600 to 115,200
- **Parity:** Odd, Even, or None
- **Line Drivers:** RS-232 and RS-485
- **Termination and Biasing (RS-485 only):** Both jumper-selectable on the connector

Setup

Because of the large number of options offered by the RMC75, setting it up can be intimidating to users new to serial communications. Read each of the following topics carefully before designing your serial network:

Configuring the RMC7*x*S Serial Communications

Line Drivers: RS-232/485

Serial Network Topologies

RS-232 Wiring for the RMC75

RS-485 Wiring for the RMC75

RS-485 Termination and Biasing

Protocols

The following topics describe how to use each protocol supported by the RMC75S:

DF1 (Full- and Half-Duplex)

Modbus/RTU

Mitsubishi Bidirectional Protocol

Using Serial Communications

See the Using Serial Communications topic.

See Also

Communications Overview

6.9.2. Using Serial Communications

Serial RS-232 or RS-485 communication is only available on the RMC75S, and the Monitor port of the RMC75P.

Communicating via serial RS-232 or RS-485 from a PLC or HMI to the RMC75 consists of the following:

- Reading RMC Registers
- Writing to RMC Registers
- Issuing Commands

This topic describes the basics of doing this with serial communications. All data in the RMC is stored in registers. Therefore, to communicate is to write to and read from registers in the RMC.

For information on setting up serial communications, see the Serial Overview topic.

Reading and Writing RMC Registers

To read from or write to RMC registers, simply write to the address of the register. The exact method will vary depending on the host controller you use, but the concept is the same.

Finding Addresses of RMC Registers

Use any of the following methods to find the address of a register in the RMC:

- Use the Register Map topic. This topic lists all the user-accessible registers in the RMC. Make sure you use the correct address format for your communication protocol.
- Use the address that is displayed in RMCTools. Many editors and windows in RMCTools display the address of the registers. Make sure you use the correct address format for your communication protocol.

Issuing Commands

See the Issuing Commands topic.

Reading Plots

For details on reading plots from the RMC, see the Reading Plots with a Host Controller topic.

See Also

Serial Overview

6.9.3. Configuration and Wiring

6.9.3.1. Line Drivers: RS-232/485

Port 1 on the RMC75S module supports two different line drivers: RS-232 and RS-485 (2-wire only). The RMC75S Monitor Port (port 0) supports only RS-232. See the Configuring Serial Communications topic for instructions on how to choose a line driver for port 1.

Features

The following chart compares these line drivers as implemented on port 1 of the RMC75S:

	RS-232	RS-485(2-wire)
Duplex	Full	Half
Differential?	No	Yes
Topology	Point-to-point	Point-to-point, Multi-drop
Wires	3	2 + CMN
Max Length[2]	50-100 ft	4000 ft

Note:
The maximum cable lengths vary depending on the baud rate, termination (for RS-485), and capacitance of the cable. See the RS-485 Termination and Biasing topic for details.

Each of the above features is described below:

Duplex (Full or Half)

Full-duplex means that each device on a serial network can send and receive at the same time, effectively doubling the bandwidth of the network. Half-duplex means that only one device on the

network can send data at one time. For the above drivers, full-duplex requires separate send and receive wires.

Differential

Differential wiring uses two wires per signal which allows common mode noise rejection. RS-232 does not use differential wiring, but instead has one wire per signal plus a ground. Differential wiring allows for longer cable distances and greater noise immunity.

Topology

Topology describes the layout of the network. Point-to-point means that exactly two devices are wired together. Multi-drop means that two or more devices are chained together. Notice that "multi-drop" with only two devices becomes point-to-point. See Serial Network Topologies for details.

Wires

This item refers to how many wires need to be connected between nodes. Notice that 2-wire actually requires three because of the Common in addition to the differential signal wires. See RS-232 Wiring for the RMC SERIAL and RS-485 Wiring for the RMC SERIAL for details.

Max Length

The maximum cable lengths vary depending on the baud rate, termination (for RS-485), and capacitance of the cable. See the RS-232 Wiring for the RMC75 and RS-485 Wiring for the RMC75 topics for details.

Flow Control

Flow control can be used with RS-232 to ensure that one device does not overrun the other device. That is, if one device is sending data and the receiving device's buffers get full, then it can use flow control to pause the first device's sending until it has room in its buffers.

See Also

Serial Overview

6.9.3.2. Configuring the RMC75S Serial Communications

The RMC75S port 1 supports most standard serial port options, several protocols, and several line drivers. These settings can be changed using RMCTools and saved to Flash memory.

The Monitor port (port 0) settings are fixed. See the Monitor Port topic for details.

Changing Serial Settings

To change the serial communication settings:

1. In the **Project** pane, expand the **Modules** folder, double-click the module with serial communications, and then click **Serial**.
2. Make the desired changes. For a description of the options, see the **Configuration Options** section in this topic.
3. Click **OK**.

Configuration Options

The following options are available under the **Serial** page in the serial module's **Properties** dialog:

Serial Port

Select port 1.

Note:
The **RS232 Monitor** (port 0) port is used by RMCTools. It can be used for other serial communication as well, but its settings cannot be changed.

Protocol Settings

Select the protocol supported by your master device and the address of the RMC75S. The address must match the address that master device expects.

Serial Port Settings

The following settings define how data is sent over the wire:

- **Line Driver:** Select the line driver you wish to use. For further details on the line drivers, see Line Drivers: RS-232/485.
- **Baud Rate:** Select the baud rate from 9,600 to 115,200. This must match the other device(s) on the network.

NOTE:
Some of these options may be disabled depending on which protocol was selected. For example, many protocols require eight data bits, and as such seven data bits is not available when these protocols are selected.

Advanced...

These are advanced settings for the line drivers:

- **Data Length:** Select seven (7) or eight (8) data bits. Most protocols require eight data bits. This must match the other device(s) on the network.
- **Parity:** Select which type of parity error checking you want to include. This must match the other device(s) on the network.
- **Stop Bits:** Select the number of stop bits. This must match the other device(s) on the network.
- **Flow Control:** Choose **None** or **Hardware (RTS/CTS)**. This must match the other device(s) on the network.

See Also

Serial Overview

Show All

6.9.3.3. Serial Network Topologies

The RMC75S supports two network topologies: point-to-point and multi-drop. Which topologies are available depend on the line driver (RS-232 or RS-485) used. See Line Drivers: RS-232/485 for details on choosing the appropriate line driver.

Point-to-Point

Point-to-point means that exactly two devices are wired together. For the RMC, this means that there will be one RMC wired to one host. Both line drivers support point-to-point, as shown in Figures 1 and 2:

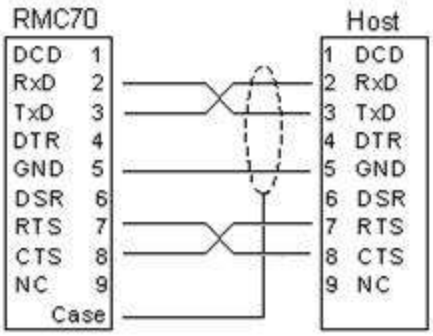


Fig. 1: Point-to-Point RS-232 Network

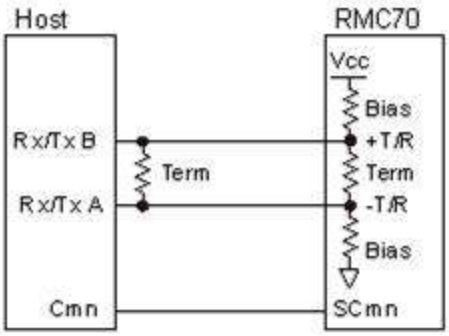


Fig. 2: Point-to-Point RS-485 Network

Figure 2 shows biasing and termination. Termination and biasing can be left out of networks at the expense of maximum cable distance and noise immunity. See RS-485 Termination and Biasing for details.

Multi-Drop

Only RS-485 supports multi-drop. Multi-drop is the connecting of multiple slaves with a single master. Slaves should be chained together. Neither a star topology nor a chain with long stubs (wires from the main chain to the device) should be used. These topologies will cause excessive ringing on the network and unreliable data transmission.

The number of devices that can be connected to the network is dictated by the number of unit loads that each represents. According to the TIA/EIA-485-A specification, there can be a maximum of 32 unit loads connected to a single network. Each RMC represents unit load for a total of 124 RMCs on the network, assuming the host is a unit load.

Figure 3 shows a typical multi-drop chain:

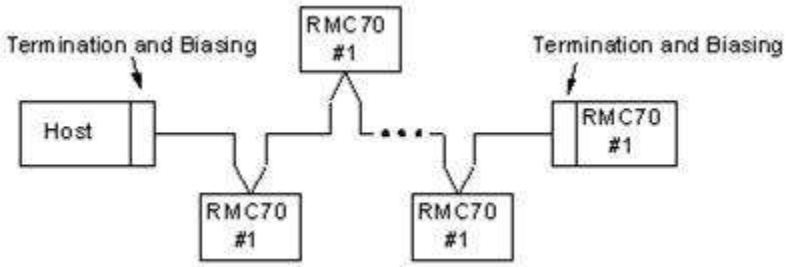


Fig. 3: Daisy-Chained RS-485 Network

Note: Termination should only be located at the extreme ends of the network:

Figure 4 shows one host with two RMC controllers in a daisy-chained two-wire RS-485 configuration:

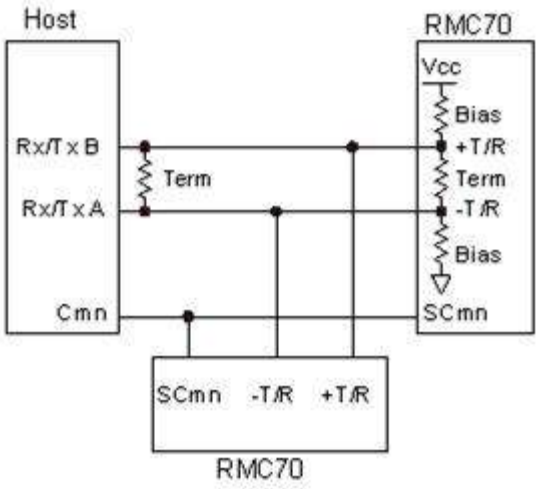


Fig. 4: Two-Wire Multi-drop RS-485 Network

See Also

Serial Overview

6.9.3.4. RS-232 Wiring for the RMC75

This topic describes the wiring of the RS-232 serial port on the RMC75S. For details on wiring the RS-232 Monitor Port, see the Monitor Port topic.

Connectors

Both of the 9-pin male DB connectors on the RMC7xS are used for RS-232 communications. A 9-pin female connector is required on the RMC75 end of the connecting cable. The other end of the cable must match the master/host hardware, which may be a 9-pin, 25-pin, RJ-11, connector block, or other connector.

Pin-Out

Pin assignments on the RS-232 9-pin connector:

Pin #	RS-232 Function
1	DCD- Not used by RMC
2	RxD - Receive Data
3	TxD - Transmit Data
4	DTR - Not used by RMC
5	GND - Common
6	DSR - Not used by RMC
7	RTS - Ready to Send
8	CTS - Clear to Send
9	Not Connected

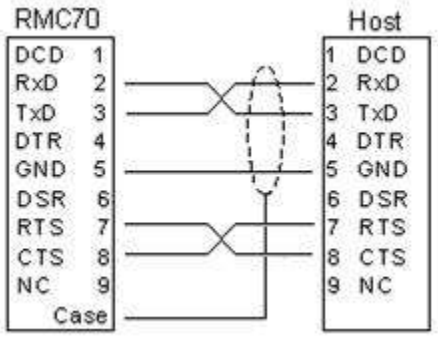
Note:
The RTS and CTS pins need only be connected if the RMC75S serial port has been configured for Hardware Flow Control.

Cabling

A null-modem or crossover cable is typically used for RS-232 communications. The RMC75 RS-232 communications require only three conductors in the cable: RxD, TxD, and GND. See the following wiring diagram for details.

Delta recommends that a shielded cable be used to limit susceptibility to outside electrical interference.

Cable Wiring



Note:
The RTS and CTS wires need only be connected if the RMC75S serial port has been configured for Hardware Flow Control.

Cable Length

One of the characteristics that limit the length of an RS-232 cable is capacitance. Most cables have a capacitance rating in pF/ft. Use the following formula to calculate the maximum distance signals can be reliably transmitted for a given cable capacitance.

MaxLength = 2400 pF / (C + (Shield * C)),

where

Shield = 2 for shielded cable and 0.5 for unshielded cable

C = capacitance rating of the cable in pF/ft

2400 pF = derived by taking the maximum capacitance specified by ANSI/TIA/EIA-232-F (2500 pF) and subtracting 100 pF for the input capacitance of the receiver

Example:

The cable that is being used is shielded and has a capacitance of 15 pF/ft.

MaxLength = 2400 pF / (15 pF/ft + (2 * 15 pF/ft))

= 2400 pF / (45 pF/ft)

= 53.3 ft

Note:
If you need to run your communications farther than allowed by this formula, then you must either use a cable with lower capacitance or use RS-485.

See Also

Serial Overview | Wiring Guidelines

Show All

6.9.3.5. RS-485 Wiring for the RMC75S

Connectors

The RMC75S serial port 1 supports 2-wire RS-485. It does not support 4-wire RS-485. RS-485 uses the 8-pin connector block on the RMC75S CPU module (pins 6, 7 and 8 are for power to the controller). The pin-out is as follows:

Pin	RMC75 Label	RS-485 (2-wire) Function
1	+T/R	Rx/Tx B (+)
2	Trm Jpr	See RS-485 Termination and Biasing
3	-T/R	Rx/Tx A (-)
4	SCmn	Common
5	Bias Jumper	See RS-485 Termination and Biasing

NOTE:
Pins 2 and 4 are for termination and biasing. See the RS-485 Termination and Biasing topic for details.

NOTE:
Some manufacturers use A and B labeling, while others use + and - labeling. If you need to interface to equipment that uses an alternate labeling scheme, keep in mind that A corresponds to - and B corresponds to +.

Cabling

All cabling for balanced or differential communications should consist of twisted pairs. Because the RMC's RS-485 interface is isolated, the signal common must be run alongside or in the cable. Therefore, for a two-wire network the cable must be either a one-pair cable with a separate ground line that is run externally or a two-pair cable in which one pair is used as the common. For a clean cabling solution, Delta recommends the option using an additional wire pair.

Another consideration when selecting communication cabling is the impedance of the cable. This impedance should match the termination resistance that is used. See RS-485 Termination and Biasing to determine whether or not your network will require termination.

No cable characteristics are specified in the TIA/EIA-422-B and TIA/EIA-485-A standards, but the RS-422-B standard does recommend 24AWG twisted pair cable with capacitance of 16 pF/ft and 100Ω characteristic impedance. These specifications will work well for RS-485 as well. One good choice would be to use Category 5 Ethernet cable. Category 5 Ethernet cable has a capacitance of 17 pF/ft max with 100Ω characteristic impedance. It is commonly available as shielded twisted pair (STP) or unshielded twisted pair (UTP). If this is not suitable then there are a number of manufacturers of communications cable such as Alpha and Beldon Wire and Cable.

Cable Length

The maximum cable length for RS-485 depends on the baud rate and termination. At higher baud rates, termination allows longer cable lengths. For details on the effects of termination and how to apply it, see RS-485 Termination and Biasing.

The following chart shows the maximum cable length for RS-485 with and without termination:

Maximum RS-485 Cable Length:

Baud Rate	Max Unterminated Cable Length (ft)	Max Terminated Cable Length (ft)
115,200	475	3250
57,600	950	4000
38,400	1900	4000
19,200	3750	4000
9,600	4000	4000
4,800	4000	4000
2,400	4000	4000

See Also

Serial Overview | Wiring Guidelines

6.9.3.6. RS-485 Termination and Biasing

Termination and Biasing are concepts that only apply to differential wiring. As such, they only apply to RS-485 and not RS-232.

Note:
Delta recommends you always use biasing on a RS-485 network. To determine whether you need termination, read the **Termination Concept** section in this topic.

Selecting Termination and Biasing on the RMC75

The RMC7xS termination and biasing can be independently selected with jumpers on the 8-pin RS-485 connector. The locations where the jumpers should be installed are marked on the label.

To select termination: Insert a jumper between pins **+T/R** and **Trm Jpr**.

To select biasing: Insert a jumper between pins **SCmn** and **Bias Jumper**.

The Termination Concept

Cable termination is a way of absorbing transmitted energy at the end of a network. This prevents signal reflections from bouncing back towards the transmitter and potentially upsetting signal quality and communications.

The termination resistor should match the characteristic impedance of the cable being terminated. The effective impedance of the RMC7*x*S's termination resistor and biasing resistors is 114Ω. Therefore, cabling with impedance of 100Ω to 120Ω is recommended.

Termination should be placed at the end of the network for each wire pair. For RS-485 (2-wire, point-to-point or multi-drop), terminate the wire pair at each end of the network. The diagram in Serial Network Topologies shows the correct location of the termination.

Termination and Cable Length

Termination is not required on all differential networks, but it does typically extend the maximum cable length. The following chart shows the maximum cable lengths at various baud rates with and without termination:

NOTE:
The maximum cable length is the length of the entire network and not just the distance between nodes on the network.

Termination vs. Cable Length:

Baud Rate	Max Unterminated Cable Length (ft)	Termination Requirements	Max Terminated Cable Length (ft)
115,200	475	Required beyond 475 ft	3250
57,600	950	Required beyond 950 ft	4000
38,400	1900	Required beyond 1900 ft	4000
19,200	3750	Required beyond 3750 ft	4000
9,600	4000	Not Required	4000
4,800	4000	Not Required	4000
2,400	4000	Not Required	4000

Cable Length Derivation

The values presented in the chart above are based on 24AWG cable with capacitance of 16 pF/ft and the following reasoning. Signals travel through a cable at approximately 66% of c or 0.66 ft/ns. It is assumed that a signal transition will dampen out after three round trips in the cable. This damping must occur before the bit is sampled or within half a bit time. One bit time is equal to the reciprocal of the baud rate.

Example:

Compute the cable length for 115,200 baud RS422.

First, we compute a half bit time at this baud rate.

Half Bit Time = 0.5 * 1 / 115200
= 4,340 ns

Next, we convert this time to the distance the signal would travel in this time, assuming a speed of 0.66 ft/ns as described above:

Distance = 4,340 ns * 0.66 ft / ns
= 2890 ft

Since three round trips are required for the signal transition to dampen and each round trip is twice the length of the cable, the total distance in feet is divided by six to get the final unterminated cable length:

Length = 2890 ft / 6
= 482 ft

This value is then rounded down to allow for inexact cable velocities and damping rates, giving us 475 ft.

The Biasing Concept

RS-485 indicates a binary 1 when the A line is at least 200 mV negative with respect to B, and a binary 0 when A is at least 200 mV positive with respect to B. It is important that the lines always be in a known state, not only when being driven. Biasing forces the network into a known state when the lines are idle and therefore otherwise not driven.

A known state is forced by allowing current to flow across the termination resistor. Therefore, biasing is usually selected on the RMC that also has a termination. However, some masters only

have termination, in which case the user may want to only select biasing on an RMC close to the master. The current will then flow across the master's termination resistor.

The RMC7*x*S requires biasing in order to be in a known state when the lines are idle. The biasing forces a binary 1.

Example:

This example assumes that there is a single master and two RMCs on the network. Compute the voltage across a 120Ω termination resistor when using 1150Ω biasing resistors.

First, we calculate how much DC resistance will be between the biasing resistors. Calculating the parallel resistance of all DC terminations and node input impedances does this. For a single master and two RMCs we have the following components:

- Master load: 1 unit load, which is defined as 12 kΩ.
- RMC loads: unit load each, which is 48 kΩ.
- Termination Resister in the RMC: 120Ω

Therefore, putting all of the resistances in parallel yields the following:

Termination Resistance = 120Ω || 12kΩ || 48kΩ || 48kΩ
= 118Ω

Then, we calculate how much DC resistance the network has between power rails:

Total Resistance = 1150Ω + 118Ω + 1150Ω
= 2418Ω

Next, we calculate how much current is flowing through this DC resistance:

Current = 5VDC / 2418Ω
= 2.068mA

Finally, we calculate the voltage drop across the termination resistor:

Voltage = 2.068mA * 118Ω
= 244mV

This value is greater than the 200mV difference required by the TIA/EIA standards and constitutes a valid binary 0 state.

See Also

Serial Overview

6.9.4. RMC75S Serial Protocols

6.9.4.1. DF1 Protocol (Full- and Half-Duplex)

DF1 is one of the serial RS-232 and RS-485 protocols supported by the RMC75S. This topic describes the DF1 protocol as it applies to the RMC75. For details on configuring the RMC75 for serial communications, including DF1, see the Serial RMC75S Configuration topic.

General

Allen-Bradley' DF1 Protocol and Command Set Reference Manual (pub. no. 1770-6.5.16) is the authority on the DF1 full- and half-duplex protocols. This manual is available on Allen-Bradleys web site (https://www.ab.com). As of this writing, the following URL contains this document: https://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1770-rm516_-en-p.pdf. When this link is out-of-date, try searching for the above publication number.

Full-duplex DF1 is used for peer-to-peer communication. Therefore, only two devices can communicate with one another. The RMC75 only supports full-duplex when used with RS-232.

Half-duplex DF1 is used for master-slave communication with one or more slaves. When more than two devices communicate with one another, 2-wire RS-485 is used. Otherwise, any line driver can be used.

RMC75 Support for DF1

Both full- and half-duplex DF1 use the same application protocol, which consists of commands and functions for the slave or peer to execute.

The RMC75 supports the following DF1 functions:

- SLC Protected Typed Write with 2 Address Fields (CMD=0x0F, FNC=0xA9)
- SLC Protected Typed Read with 2 Address Fields (CMD=0x0F, FNC=0xA1)
- SLC Protected Typed Write with 3 Address Fields (CMD=0x0F, FNC=0xAA)
- SLC Protected Typed Read with 3 Address Fields (CMD=0x0F, FNC=0xA2)
- Echo (CMD=0x06, FNC=0x00)
- Diagnostic Status (CMD=0x06, FNC=0x03)
- PLC-5 Word Range Read (CMD=0x0F, FNC=0x 68)
- PLC-5 Word Range Write (CMD=0x0F, FNC=0x 67)
- PLC-5 Typed Read (CMD=0x0F, FNC=0x 01)
- PLC-5 Typed Write (CMD=0x0F, FNC=0x 00)

PLC Support for DF1

DF1 is a major industrial serial protocol supported by a large number of devices, both those built by Allen-Bradley and other companies. Any DF1 master implementation that uses the above blocks should also be able to read and write from the RMC75.

Using DF1 with Allen-Bradley PLCs

See the Using Allen-Bradley Controllers via Message Block for details on communicating with Allen-Bradley PLCs.

See Also

Serial Overview

6.9.4.2. Mitsubishi Bidirectional Protocol

The Mitsubishi Bidirectional protocol is one of the serial RS-232 and RS-485 protocols supported by the RMC75S. The Mitsubishi Bidirectional protocol is for use with Mitsubishi's Q-series QJ71C24N serial communication module. This protocol requires RMC75 firmware version 1.60 or newer and RMCTools version 1.60.0 or newer.

The QJ71C24N supports several communication protocols. The one that matches the needs of the RMC75S is the Bidirectional protocol. It allows the Q-series PLC to read and write binary data from an RMC75S over serial RS-232 or RS-485. The RMC75S requires that the data sent via the Bidirectional protocol be formatted as described in this topic.

The Bidirectional protocol is described in Mitsubishi's "Q Corresponding Serial Communication Module User's Manual (Basic)". The manual part number is SH (NA)-080006-J. Search for 080006 on www.meau.com in the Downloads>>Manuals section.

A sample program for using the Bidirectional protocol is available on the downloads page of Delta's website at https://deltamotion.com/dloads. This should be used as a starting point for any Mitsubishi Q-series program using the QJ71C24N module and an RMC75S.

Configuring

The instructions below are with GX Developer version 8.25B.

Switch Settings

Set the intelligent function module switches for the desired serial settings. See section 4.5.2 of the following Mitsubishi manual: Q Corresponding Serial Communication Module (User's Manual).

The data bits must be 8 and the protocol must be Bidirectional. Make sure you set the baud rate, parity and stop bits identically in the QJ71C24N and RMC75S.

Writing to the RMC75S

Use the GP.BIDOUT instruction to write to registers in the RMC75S. It is described in section 9.5 of the "Q Corresponding Serial Communication Module User's Manual (Basic)". The RMC75S requires that the data sent to it via this instruction is in the format shown below.

0 (16 bits integer)	Register File (16 bits integer)	Register Element (16 bits integer)	Data Item 1 (32 bits, typically floating-point)	Data Item 2 (32 bits, typically floating-point)	…	Data Item n (32 bits, typically floating-point)

Description:

0	This 16-bit word must be 0.
Register File	This 16-bit word is the file number of the register's address in IEC format. For example, for %MD8.12, the file number is 8.
Register Element	This 16-bit word is the element number of the register's address in IEC format. For example, for %MD8.12, the element number is 12.
Data Time 1	The RMC75 has 32-bit registers. Therefore, you can only write 32-bit words. Most RMC75 registers are floating-point; a few are integers.
Data Item *n*	To write n 32-bit registers to the RMC75, make sure the TxCount is correct. It should be (2 x n) + 3.

Example

A programmer wishes to write 5 values to the variable table (address %MD56.0). These values are: 32.876, 1.0, 12.0, 5.432, 862.0.

The send data for the GP.BIDOUT instruction would be as follows:

0 (16 bits)	56 (16 bits)	0 (16 bits)	32.876 (32 bits)	1.0 (32 bits)	12.0 (32 bits)	5.432 (32 bits)	862.0 (32 bits)

Reading from the RMC75S

To read registers from the RMC75S, use the GP.BIDOUT instruction to send a read request and then use the GP.BIDIN instruction to read the received data. The GP.BIDIN instruction is described in section 9.6 of the "Q Corresponding Serial Communication Module User's Manual (Basic)".

Writing the Read Request

To send a request a read from the RMC75S, the send data of the GP.BIDOUT instruction must be formatted as shown below. Each box is a 16-bit word.

Read Count (16 bits)	Register File (16 bits)	Register Element (16 bits)

Description:

Read Count	This is the number of 32-bit registers to be read from the RMC75, starting at the address given by the file and element.
Register File	This is the file number of the address of the first register to be read. For example, for %MD8.12, the file number is 8.
Register Element	This is the element number of the address of the first register to be read. For example, for %MD8.12, the element number is 12.

Receiving the Returned Data

After the write request is sent, the RMC75S will return the requested data. The QJ71C24N X3 input indicates whether data has been received. To read the receive buffer, use the GP.BIDIN instruction. The returned data is in the format shown below:

Count (16 bits)	Data Item 1 (32 bits)	Data Item 2 (32 bits)	…	Data Item *n* (32 bits)

Description:

Count	This is the number of 32-bit words read from the buffer.
Data Item 1-*n*	This is the returned data.

See Also

Communications Overview | Using the Mitsubishi Q-Series PLC with the RMC

6.9.4.3. Modbus/RTU Protocol

Modbus/RTU is one of the serial RS-232 and RS-485 protocols supported by the RMC75S. This topic describes the Modbus/RTU protocol as it applies to the RMC75. For details on configuring the RMC75 for serial communications, see the Serial Configuration topic.

General

Modbus/RTU is a standard protocol managed by Schneider Automation (Modicon). The specification is available through Schneider Automation (https://www.modicon.com/openmbus) and Modbus.org (https://www.modbus.org). The Modicon Modbus Protocol Reference Guide (PI-MBUS-300) from Modicon is a good reference for this protocol. It is available through the Modicon web site. At the time of this writing it was necessary to select the MODBUS/TCP Protocol documents link to find an Acrobat Reader (.pdf) format version of this document.

RMC75 Support for Modbus/RTU

Modbus, in its various forms such as Modbus/ASCII, Modbus/RTU, Modbus Plus, and Modbus/TCP, is a request/response protocol. That is, a Modbus master makes a request from a Modbus slave, and the slave responds. A number of functions are defined under Modbus. The following functions are supported by the RMC75:

- Read Multiple Registers (FC 3)
- Read Input Registers (FC 4)
- Write Single Register (FC 6)
- Get Diagnostics (FC 8)
- Write Multiple Registers (FC 16)
- Read/Write Registers (FC 23)

Each of the above functions acts on 4X or Holding registers. These 4X registers are mapped in the RMC75 as described in the following topic:

- RMC Register Map (Modbus/RTU and /TCP)

See Also

Serial Overview

6.10. Using Master Controllers

6.10.1. Using Allen-Bradley Controllers via Message Block

Most Allen-Bradley PLCs and PC-based controllers (ControlLogix, CompactLogix, SLC5/05, PLC-5, SoftLogix, etc.) support serial RS-232 and Ethernet communications, either built-in or through an add-on module.

This topic describes how to communicate with the RMC from the Allen-Bradley PLCs via the Message (MSG) block. For information on using EtherNet/IP I/O with the Logix PLCs, see the Using Allen-Bradley Controllers via EtherNet/IP topic. For details on using Allen-Bradley's RSView operator interface with the RMC, see the RSView with the RMC topic.

The Allen-Bradley PLCs can read or write from registers in compatible remote devices such as other Allen-Bradley PLCs or the RMC. The RMC contains floating point (F) files, all of which are accessible over serial or Ethernet from the Allen-Bradley PLCs. See DF1 Addressing for details on finding those registers and their addresses.

If you need help setting up your Ethernet network, either consult your network administrator, or for simple stand-alone networks, see Setting Up a Standalone TCP/IP Network.

For setting up the RMC75S serial settings, see the Configuring Serial Communications topic. For setting up the RMC75E,RMC150E, or RMC200 Ethernet, see the RMC Ethernet Setup topic.

Example Programs

Delta provides example PLC programs to help you quickly set up the communications between your PLC and the RMC. See the downloads section of Delta's website at https://deltamotion.com.

Message (MSG) Block

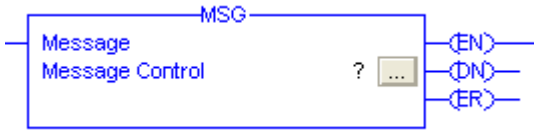
The Allen Bradley PLCs and controllers listed above all use the same ladder logic block to communicate over Ethernet: the Message (MSG) block. This block takes a number of parameters, which are briefly described below. For a complete description of the parameters, refer to Allen-Bradley's Instruction Set Reference Manual for the appropriate PLC.

MSG Parameters

The MSG block parameters differ slightly depending on the controller and programming software. The parameters used by RSLogix 5 version 3.2.0.0, RSLogix 500 version 6.30.00, and RSLogix 5000 version 17.00.00 for the PLC-5, SLC 5/05, ControlLogix and CompactLogix controllers respectively are described below. The SoftLogix 5 parameters are similar.

ControlLogix and CompactLogix MSG Block Parameters

The ControlLogix or CompactLogix MSG block is displayed as follows. More detailed information on the MSG block is available in Allen Bradley publication 1756-RM003L-EN-P, Logix5000 Controllers General Instructions.



The **Message Control** portion of the MSG requires a tag.
To create a tag:

- Right-click the "?" and choose **NewTag**.
- Type a name in the **Name** field.
- Make sure the **Data Type** is MESSAGE.
- Click **Configure**. The Message Configuration dialog will open.
- Enter the following on the **Configuration** tab:
    - **Message Type:** From this drop-down list, select PLC-5 Typed Read to read values from the RMC, or PLC-5 Typed Write to write values to the RMC. The SLC Typed Read or Write will also work, but is limited to 58 registers.
    - **Source Element (reads only):** Enter the address of the first RMC register you want to read. See the Registers Maps for help on addresses.
    - **Source Element (writes only):** Enter the tag in the ControlLogix or CompactLogix that you want to send to the RMC. In most cases, the element should be an array or structure composed entirely of REAL data types, since this is the type of the RMC data.
    - **Number of Elements:** Enter the number of RMC registers to read or write in this field.
    - **Destination Element (reads only):** Enter the tag in the ControlLogix or CompactLogix into which you want to read the RMC data. See the Source Element bullet above for suggestions on the data type of the tag.
    - **Destination Element (writes only):** Enter the address of the first RMC register you want to write. See the Registers Maps for help on addresses.
- Enter the following on the **Communication** tab:
    - **For Ethernet:**
      Path:

1. Add the RMC as an Ethernet Device to the Logix Designer PLC project using the EDS file. For details, see Using Allen-Bradley Controllers via EtherNet/IP I/O.
2. In the **Path** box, enter the name of the RMC as it was added to the project.

   For old PLCs, you may need to use the non-preferred manual method

   - Enter the path from the ControlLogix or CompactLogix CPU to the RMC. The format of the path usually is as follows:

     [PLC Ethernet module name], 2, [RMC IP Address]

     To get the PLC's Ethernet module name, click the Browse button and select the PLC's Ethernet module. The "2" specifies the Ethernet port on the module.

     Therefore, if a ControlLogix 1756-ENBT is in named **LocalEnb**, and the RMC is at address 192.168.0.5, then the following path would be used:

     LocalENB, 2, 192.168.0.5

     Note: If you are also using EtherNet/IP I/O cyclic messaging, you can use the name of the name you assigned to the Generic Ethernet Module that references the RMC. This name will constitute the entire **Path**.
     For example, if you assigned the name MyRMC, then the Path would be: MyRMC.

     If the Logix5000 project I/O configuration does *not* include the module name, you will need to give the path of going over the Logix backplane to the Logix ethernet communication module and then going over the Ethernet to the RMC. The backplane is indicated by the number "1". The path would then be:

     1, [Ethernet module slot number], 2, [RMC IP Address]

     Therefore, if the 1756-ENBT is in slot 3 (the third slot on the rack), and the RMC is at address 192.168.0.5, then the following path would be used:

     1, 3, 2, 192.168.0.5

   - **For Serial:**
     Path: Enter the path from the ControlLogix or CompactLogix CPU to the RMC. The format of the path usually is as follows:

     2, *station address*

     where *station address* is the node address of the RMC. For example, if the node address is 3, the path is:

     2, 3

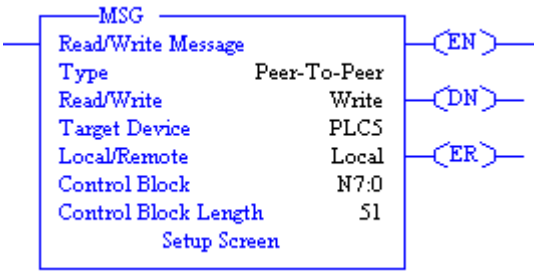
     Note:
     The "2" is the port number for the serial port on the ControlLogix CPU.

   - **Communication Method:** Select the **CIP** option.

- Click **OK**.
- To make changes to the Message Configuration dialog later, click the ellipsis button in the MSG block.

SLC 5/05 MSG Block Parameters

The SLC 5/05 MSG block is displayed as the following:



Parameter	Description
Type	This parameter is always set to Peer-To-Peer for serial or Ethernet communication channels.
Read/Write	This parameter should be set to **Read** to read registers from the RMC and to **Write** to write registers to the RMC.
Target Device	This should be set to **500CPU** for communicating with the RMC.
Local/Remote	This parameter has possible values of Local and Remote. It should be set to **Local** for communicating with the RMC.

Control Block	This parameter points to a block of integer-file registers (51 registers for Ethernet, 12 for serial). Set this to a block of registers, and then use the Setup Screen option in the MSG ladder logic block to modify those register values:
	• **This Controller:** This section holds parameters for the SLC 5/05.
	  ○ **Communication Command:** This parameter will be set to 500CPU Read, or 500CPU Write, depending on what was selected in the MSG block itself (as described above).
	  ○ **Data Table Address:** Enter the address of the first Allen-Bradley PLC register to read RMC registers into, or to write to RMC registers from.
	  ○ **Size in Elements:** Enter the number of RMC registers to read or write in this field. The range enforced by the SLC is 1 to 256 values.
	  ○ **Channel:** Set this to the channel number of the serial Ethernet channel. For the SLC 5/05, this should be channel #1 for both serial and Ethernet.
	• **Target Device:** This section holds parameters for the target device.
	  ○ **Message Timeout:** Indicate the number of seconds to wait for the RMC to respond before determining that the attempt failed. This can be set as low as a few seconds.
	  ○ **Data Table Address:** Enter the address of the first RMC register to read or write in this field. See the RMC Register Map (Allen-Bradley) for help on addresses.

- **For Ethernet:**
  Ethernet (IP) address: Set this to the IP address of the RMC you wish to communicate with.

MultiHop: This parameter should be set to **No**.

- **For Serial:**
  Local Node Addr (dec): Enter the node address of this RMC. The node address of the RMC is set up in the Serial Port Settings Page. The node address entered on that screen is in decimal, so it is recommended that you enter the same number in the dec field.

Local/Remote and Bridge Parameters: In most applications these will be set to Local and there will be no bridge parameters. If you are using a bridge then these parameters will need to be used. However, this is beyond the scope of RMC documentation. Refer to your Allen-Bradley documentation for instructions on using these fields.

Note:
For the SLC505, if the start of communications with the RMC exhibits several minutes of delay after power-up, it is because your SLC processor and/or firmware is old. Newer SLC processors do not have this problem.

PLC-5 MSG Block Parameters

The PLC-5 MSG block is displayed as follows:

The **Control** parameter points to a block of 51 N-file (integer) registers or two (2) MG-file (message) registers. Set this to an unused block of registers, and then use the **Setup Screen** option in the MSG ladder logic block to modify those register values:

Register	Description
This PLC-5	This section holds parameters for the PLC-5: • **Communication Command:** From this drop-down list, select PLC-5 Typed Read to read values from the RMC, or PLC-5 Typed Write to write values to the RMC. • **Data Table Address:** Enter the address of the first Allen-Bradley PLC register to read RMC registers into, or to write to RMC registers from. • **Size in Elements:** Enter the number of RMC registers to read or write in this field. Transfers are limited to 1000 bytes for PLC-5 Typed Reads and Writes. Therefore, this limit is 500 integers, 250 floats, etc. • **Port Number:** Set this to the Ethernet channel number. For the PLC-5, this should be channel #2.

Target Device

This section holds parameters for the target device:

- **Data Table Address:** Enter the address of the first RMC register to read or write in this field. See the Registers Maps for help on addresses.
- **For Ethernet:**

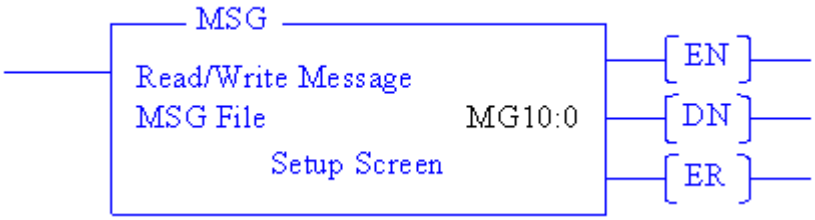
  MultiHop: This parameter should be set to **No**.

  Ethernet (IP) address: Set this to the IP address of the RMC you wish to communicate with.
- **For Serial:**

  Local Node Addr (dec): Enter the node address of this RMC. The node address of the RMC is set up in the Serial Port Settings Page. The node address entered on that screen is in decimal, so it is recommended that you enter the same number in the dec field.

MicroLogix MSG Parameters

The MicroLogix MSG block is displayed as follows:



Serial Communications:

To edit the parameters of the message block, select the MSG block, enter an unused MSG file in the MSG File parameter, and double-click Setup Screen. This brings up a dialog with the following options:

This Controller: This section holds parameters for the MicroLogix.

Communication Command: From this drop-down list, select PLC-5 Typed Read to read values from the RMC, or PLC-5 Typed Write to write values to the RMC. The SLC Typed Read or Write will also work, but is limited to 58 registers.

Data Table Address: Enter the address of the first Allen-Bradley PLC register to read RMC registers into, or to write to RMC registers from.

Size in Elements: Enter the number of RMC registers to read or write in this field. The MicroLogix can transfer 1 to 41 integers.

Channel: Set this to the channel number of the serial port you want to use. The MicroLogix 1500 LRP Series B can use either channel 0 or 1, but all other MicroLogix PLCs will always use channel 0.

Target Device: This section holds parameters for the target device.

Message Timeout: Indicate the number of seconds to wait for the RMC to respond before determining that the attempt failed. This can be set as low as a few seconds.

Data Table Address: Enter the address of the first RMC register to read or write in this field. See the Registers Maps topic for help on addresses.

Local Node Addr (dec): Enter the node address of this RMC. The node address of the RMC is set up in the Serial Port Settings Page. The node address entered on that screen is in decimal, so it is recommended that you enter the same number in the dec field.

Local/Remote and Bridge Parameters: In most applications these will be set to Local and there will be no bridge parameters. If you are using a bridge then these parameters will need to be used. However, this is beyond the scope of RMC documentation. Refer to your Allen-Bradley documentation for instructions on using these fields.

Ethernet Communications (MicroLogix 1100):

1. In the RSLogix project tree, in the Data Files, create a message File (MG) data file and a Routing Information (RI) data file.
2. Add the MSG block to the ladder logic. Select the MSG block, enter the an address in the Message data file you just created (e.g MG9:0), and double-click Setup Screen.

   This brings up a dialog with a **General** tab with the following options:

   This Controller: This section holds parameters for the MicroLogix.

   Channel: To use Ethernet on the MicroLogix 1100, choose **1 (integral)**.

   Communication Command: This parameter can be set to PLC5 Read, PLC5 Write, 500CPU Read, or 500CPU Write. The type of PLC selected is not important, but the Read or Write determines whether you will read registers from the RMC or write registers into the RMC.

   Data Table Address: Enter the address of the first Allen-Bradley PLC register to read RMC registers into, or to write to RMC registers from.

   Size in Elements: Enter the number of RMC registers to read or write in this field.

   Target Device: This section holds parameters for the target device.

   Message Timeout: Indicate the number of seconds to wait for the RMC to respond before determining that the attempt failed. This can be set as low as a few seconds.

   Data Table Address: Enter the address of the first RMC register to read or write in this field. See the Registers Maps topic for help on addresses.

   Local / Remote: Choose **Local**.

   Routing Information File (RI): Enter the address in the Routing Information File that you intend to use (e.g. RI10:0).

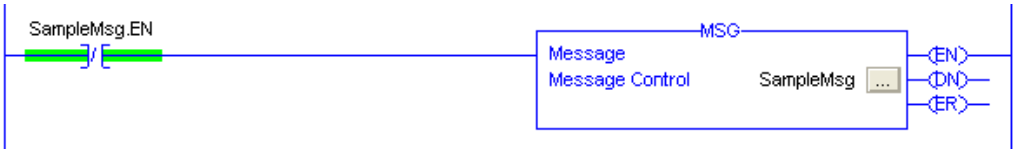
3. On the **Multi-hop** tab, in the **To Address** box, enter the IP address of the target RMC. For example, 192.168.0.219.

Using the MSG Block in Ladder Logic

The Allen-Bradley MSG block may take multiple ladder scans to complete. Therefore, it is important to enable the MSG block for the correct amount of time. Specifically, the MSG block must be energized until the message control's enable (EN) bit turns on. Following the ladder samples to ensure proper functionality:

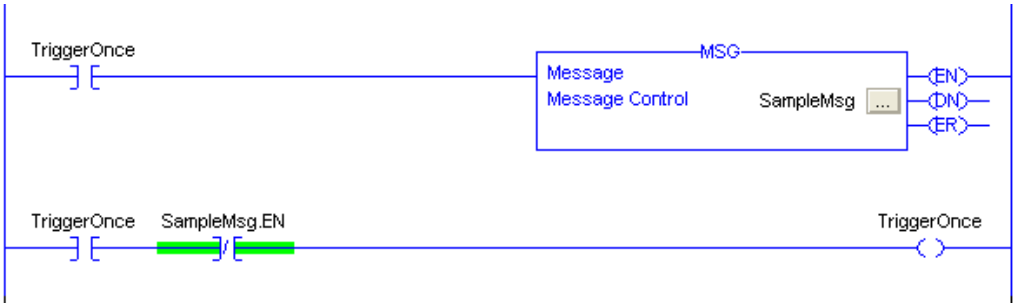
Read or Write Continuously

Using the Examine If Open instruction as shown below fulfills two requirements of continuous MSG transactions. First, it will keep the block energized until the EN turns on, and second, it de-energizes the MSG block once the transactions is started so that when the transaction is completed (EN goes low again), the MSG block sees a rising edge on its input, thus repeating the transaction:



Read or Write Once

This sample takes care to keep the MSG block energized until the MSG block starts, as indicated by the enable (EN) bit turning on. Once this happens, the application-controlled TriggerOnce coil is turned off. The message control's Done (DN) or Error (ER) bits can be used to process the results of the transaction.



Reading DWORDs from the RMC

All items in the RMC have F-file addresses. Allen-Bradley defines F file data as 32-floating point values. All the RMC registers have F-file addresses, even if they are DWORDs or DINTs. For example, the Status Bits and Error Bits in the RMC are DWORDS. To read these values, read them using their F addresses as given in the RMC. Then, in the PLC, use the COP instruction (RSLogix5000), or CPW instruction (RSlogix500) to copy the data to a register or tag of the correct data type, for example, an N register, L register, or DINT tag. The COP instruction preserves all the bits correctly, and the resulting values will be correct.

If the PLC must write a DINT or DWORD in the RMC, use a similar method.

See Also

Ethernet Overview | RMC75 Register Map | RMC150 Register Map | RSView with the RMC

6.10.2. Using Allen-Bradley Controllers via EtherNet/IP I/O

This topic describes how to configure and use Allen-Bradley ControlLogix and CompactLogix PLCs to communicate with the RMC via EtherNet/IP I/O. If you are instead using the MSG block command, see the Using Allen-Bradley Controllers via Message Block topic.

The download section of Delta's website contains Studio 5000 programs that are already set up to communicate with an RMC via EtherNet/IP I/O. These programs can be used as a starting point for your application.

The setup steps for RMC to Allen-Bradley communication include:

- Determine Data Size and Number of I/O Connections
- Determine I/O Data Locations in the RMC
- Set Up the RMC for EtherNet/IP I/O
- Configure the PLC Connection Using an EDS File or Configuring the PLC Connection Using a Generic Ethernet Module
- Export RMCTools Tags to Logix Designer
- Perform Communications

Determine Data Size and Number of I/O Connections

The first step is to determine the amount of data to be transferred and the number of I/O connections to set up between the PLC and the RMC:

1. If you have an RMC75E or RMC150E, you will use one (1) I/O connection that transfers data back and forth between the PLC. See the chart below for the maximum number of registers that can be transferred.

2. If you have an RMC200, and will be sending 124 registers or less to the RMC, and receiving 125 registers or less from the RMC, then you will use one (1) connection, otherwise, you will need to use multiple connections (maximum of 3), for as much data as you need, as shown in the chart below. See the chart below for the maximum number of registers that can be transferred with each connection. Keep in mind that the connections are independent of each other, and the data will not be transferred at the same time, even when the RPI may be set to the same value.

I/O Connections and Max Data Size (input is to PLC, output is from PLC)

Connection	RMC75E	RMC150E	RMC200
Connection#1	125 input registers 124 output registers	125 input registers 124 output registers	125 input registers* 124 output registers*
Connection#2	n/a	n/a	125 input registers 124 output registers
Connection#3	n/a	n/a	125 input registers 124 output registers

*The RMC200 supports more registers on the first connection, but the Allen-Bradley controllers support only the listed amount.

Determine I/O Data Locations in the RMC

EtherNet/IP I/O transfers data back and forth between the RMC and PLC at the Requested Packet Interval (RPI). The user must specify which data items in the RMC should be sent and received. Typically, this is data in the Indirect Data Map.

For each connection to the RMC, set up the Indirect Data map so that one part contains all the data coming from the PLC (Incoming Data), and another part contains all the data going to the PLC (Outgoing Data). Make sure the Incoming and Outgoing Data areas in the Indirect Data Map do not overlap, and make sure the areas for a connection do not overlap that for another connection.

The Outgoing Data typically includes RMC status items that the PLC always needs to keep track of, such as actual positions and status bits.

The Incoming Data consists of items that the PLC needs to write to in the RMC. This is typically variables and possibly command registers.

Note:
The Incoming and Outgoing Data locations need not be in the Indirect Data Map. However, the Indirect Data Map is usually the best choice. Other options are the Variable Table and the command area.

Setting Up the RMC for EtherNet/IP I/O

Do the following in the RMC:

1. **Set the RMC's IP Address**
   Set up the RMC's IP Address as for any Ethernet connection. See Setting Up the RMC Ethernet for details.
2. **Set Up the Indirect Data Map**

   In the Project pane, double-click **Address Maps**, then click **Indirect Data Map**.

   Beginning at item 0 in the Indirect Data Map, choose the items for the Outgoing Cyclic I/O Data.

   At some location in the Indirect Data Map after the Outgoing Cyclic I/O Data area, choose the items for the Incoming Cyclic I/O Data, that is, the items that will be sent to the RMC from the PLC. If you are using another location for your Incoming Data, such as the Variable Table or Command Area, you need not set up the Indirect Data Map for the Incoming Data.

   If you are using multiple connections to the RMC, repeat this for each connection.

Example

If you have a 4-axis controller, you may wish to set up the Outgoing Data at the beginning of the Indirect Data Map to include the Actual Position and the Status Bits for each axis, in addition to information on Task 0, which perhaps runs your user programs. You may also wish to set the Incoming Data, starting at item 12 in the Indirect Data Map, to go to 5 variables and some of the Axis 0 command registers. You could then set up the Indirect Data Map like this:

	Reg #	Map To	Description	Current
0	%MD42.0	%MD8.0	Axis0 Status Bits	N/A
1	%MD42.1	%MD9.0	Axis1 Status Bits	N/A
2	%MD42.2	%MD10.0	Axis2 Status Bits	N/A
3	%MD42.3	%MD11.0	Axis3 Status Bits	N/A
4	%MD42.4	%MD8.8	Axis0 Actual Position (pu)	N/A
5	%MD42.5	%MD9.8	Axis1 Actual Position (pu)	N/A
6	%MD42.6	%MD10.8	Axis2 Actual Position (pu)	N/A
7	%MD42.7	%MD11.8	Axis3 Actual Position (pu)	N/A
8	%MD42.8	%MD48.0	Task 0 Status	N/A
9	%MD42.9	%MD48.3	Task 0 Current Program	N/A
10	%MD42.10			
11	%MD42.11			
12	%MD42.12	%MD56.0	Current Value - 0 - (NoName)	N/A
13	%MD42.13	%MD56.1	Current Value - 1 - (NoName)	N/A
14	%MD42.14	%MD56.2	Current Value - 2 - (NoName)	N/A
15	%MD42.15	%MD56.3	Current Value - 3 - (NoName)	N/A
16	%MD42.16	%MD56.4	Current Value - 4 - (NoName)	N/A
17	%MD42.17	%MD40.0	Axis0 Command Area	N/A
18	%MD42.18	%MD40.1	Axis0 Command Parameter 1	N/A
19	%MD42.19	%MD40.2	Axis0 Command Parameter 2	N/A
20	%MD42.20	%MD40.3	Axis0 Command Parameter 3	N/A
21	%MD42.21	%MD40.4	Axis0 Command Parameter 4	N/A
22	%MD42.22	%MD40.5	Axis0 Command Parameter 5	N/A
23	%MD42.23			

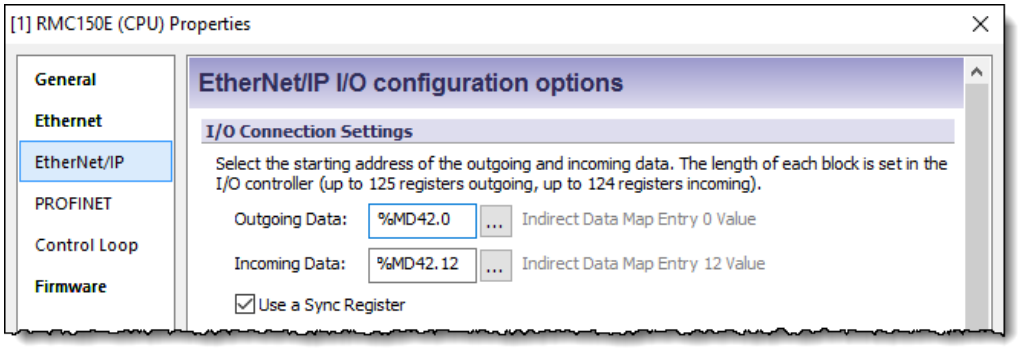
3. **Set the Cyclic I/O Data Locations in the RMC**
   In the Project pane, expand the **Modules** folder, double-click the CPU module, and choose **EtherNet/IP**.

   Under **Outgoing Data**, enter the starting location for the outgoing cyclic I/O data. In our example, verify that the location is the Indirect Data Map Entry 0 Value.

   Under **Incoming Data**, enter the starting address for the incoming cyclic I/O data. This should be a location in the Indirect Data Map, the Variable Table, or Command Area as discussed in the **Determine I/O Data Locations in the RMC** section above.

   If you are using multiple connections to the RMC, repeat this for each connection, otherwise, ignore the other connection settings.

For example, the EtherNet/IP Settings Page below shows an RMC150 with the Outgoing Data coming from the Indirect Data map starting at item 0 and the Incoming Data going to the Indirect Data starting at item 12.



4. **Choose Whether to Use a Sync Register**

   The Sync Register provides a method for the PLC to synchronize the Input Data and Output Data. If you will be writing to the Command Area directly or indirectly via the Indirect Data Map, Delta recommends using the Sync Register.

   With a Sync Register, the Incoming Data is not written to the RMC until the Sync Register changes. If you prefer to have the Incoming Data be written whenever any value in the Incoming Data changes, choose the option to not use a Sync Register.

   For multiple connections, each connection has an individual Sync Register. Keep in mind that the connections are independent of each other, and the data may not be transferred at exactly the same time, even if the Sync Registers are changed simultaneously.

   For more details, see Using an EtherNet/IP I/O Connection.

Configuring the PLC Connection Using an EDS File

Configuring the connection using an EDS file is the preferred method, and supports multiple connections. For RSLogix version 19 and earlier, see **Configuring the PLC Connection Using a Generic Ethernet Module** below.

Install the EDS File

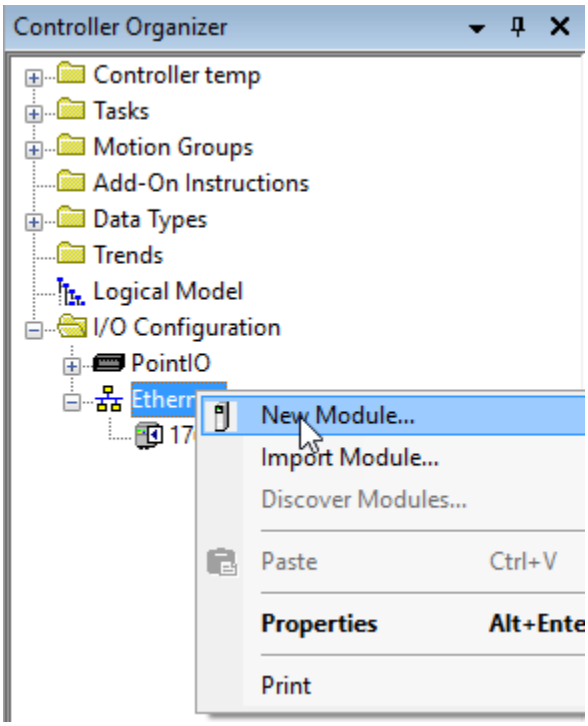
1. In Studio 5000, on the **Tools** menu, choose **EDS Hardware Installation Tool**.
2. In the wizard:
   1. Choose **Register an EDS file**.
   2. Browse to the EDS folder in the RMCTools installation location, which is typically **C:\Program Files\RMCTools\EDS**.
   3. Choose the highest version number of the **.eds** files for your RMC (e.g. rmc75e_v3.eds).
   4. Complete the wizard.

If you do not have the most recent version of RMCTools, or have older firmware, see Using the Correct EDS File for more information.

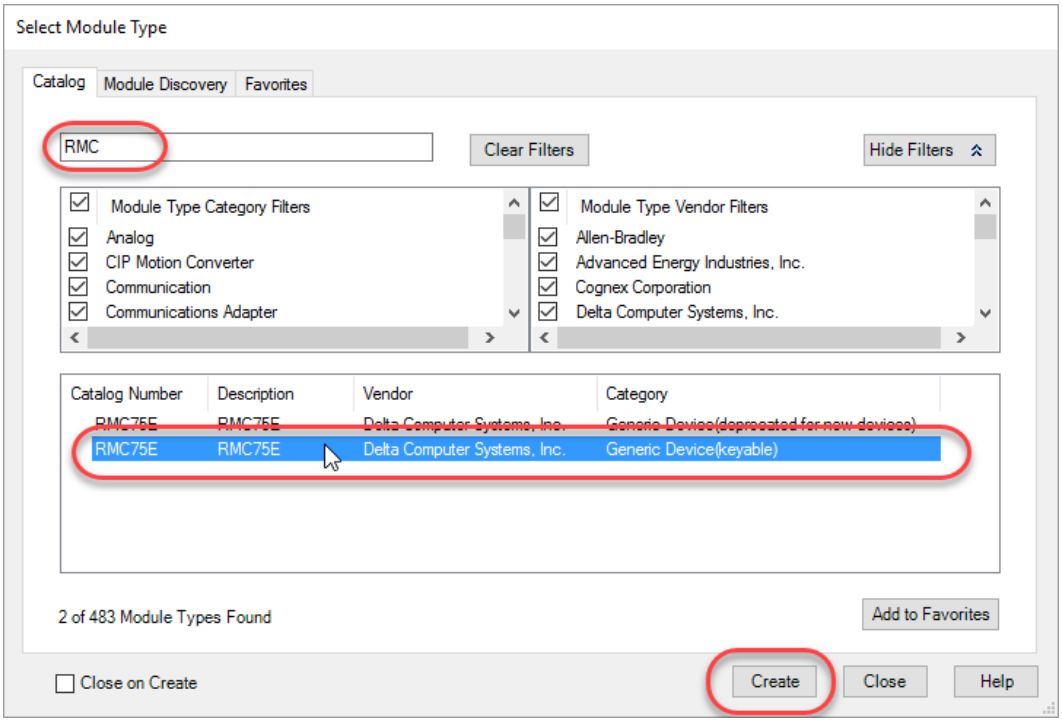
Configure the Connection

1. In Studio 5000, open your project.
2. In the Controller Organization window, in the **I/O Configuration** folder, if there is not already an Ethernet module, add your Ethernet module.

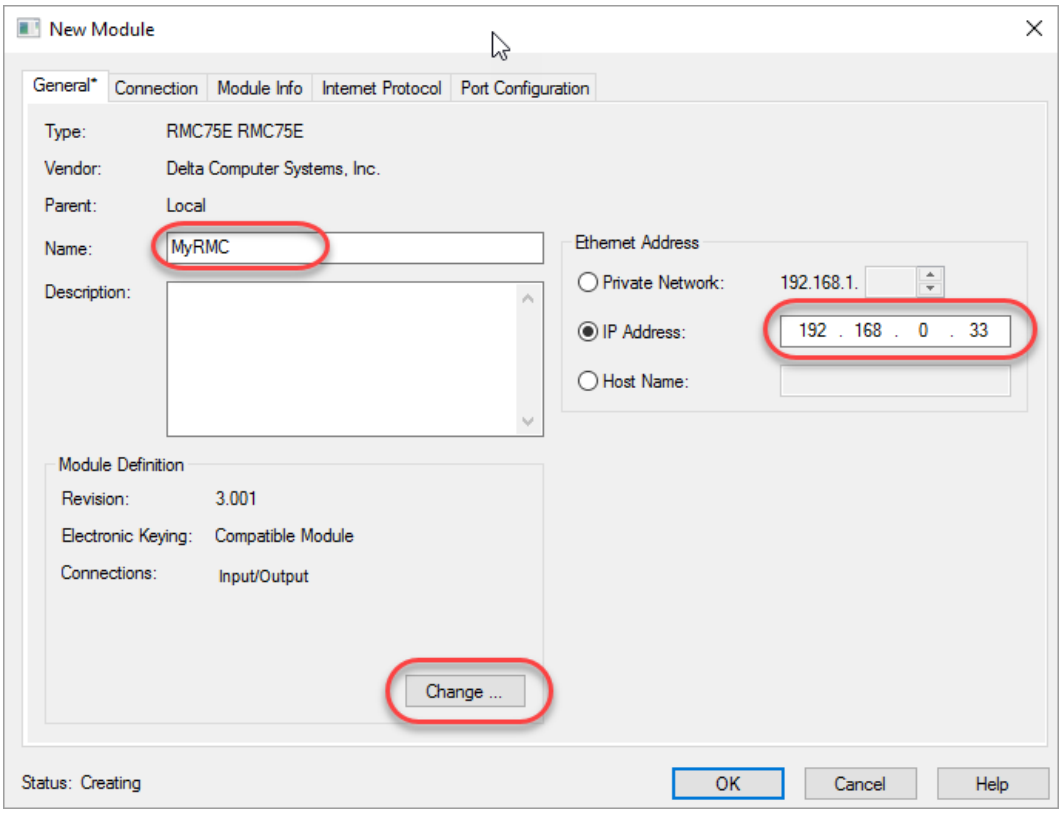
3. Right-click the Ethernet module under which you want to add the RMC, and choose **New Module**.



4. In the search box, type **RMC**. Select the RMC from the list, and click **Create**. For the RMC75E or RMC150E, you may see two options. If this occurs, select the one with Category "Generic Device (keyable)" for RMC firmware 3.62.0 or newer, and select the option Category "Generic Device (deprecated for newer devices)" for older firmware versions.

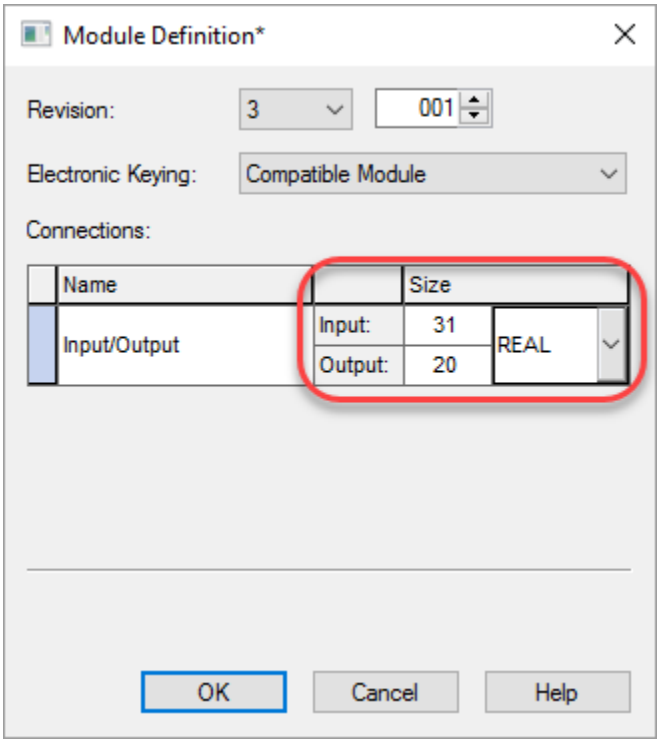


5. The **New Module** dialog opens. On the **General** tab, enter the **Name** and **IP Address**, then click **Change**, which opens the **Module Definition** dialog.

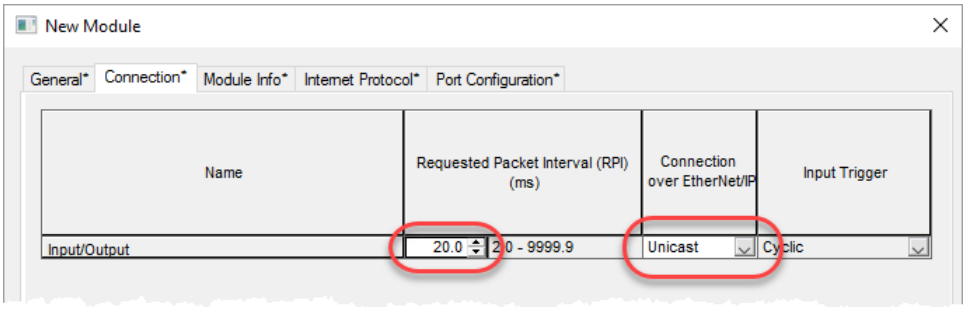


6. In the **Module Definition** dialog:
   a. Under **Connections**, in the **Name** column, select **Input/Output**.
   b. For the **Input/Output** connection, set the data type to **REAL**.
   c. Set the **Input** to the number of registers to transfer from the RMC to the PLC. If you selected to use the Sync Register, the size should be one plus the number of registers of the RMC's Outgoing Cyclic I/O Data. If you are using multiple connections, this value applies to the first connection.
   d. Set the **Output** to the number of registers to transfer from the PLC to the RMC. If you selected to use the Sync Register, the size should be one plus the number of registers of the RMC's Incoming Cyclic I/O Data. If you are using multiple connections, this value applies to the first connection.
   e. If you are using multiple connections (RMC200 only), add up to two additional **Input/Output** connections in the next rows. If you only need input data (from the RMC to the PLC), choose an **Input Only** connection. For each connection, set the data type to REAL, and set the number of **Input** and **Output** registers.
   f. The **Revision** and **Electronic Keying** values can generally be left set to their defaults. Notice that the Revision refers to the EtherNet/IP Identity Object revision, not the RMC firmware revision. See Using the Correct EDS File for details on which Revisions apply to which firmware revisions.

g. Click **OK**.



7. In the **New Module** dialog, click the **Connection** tab and make the following settings for each connection:
   a. **RPI**: Select the desired update rate. A commonly used RPI is 20.0 ms. Very low RPIs may flood the network, and reduce network reliability.
   b. **Connection over EtherNet/IP:**
   Set this to **Unicast**. This option is very important for reducing network bandwidth required.



8. Set the remaining items in the **New Module** dialog as required for your application, and click **OK**.

The above steps will allocate two tags per connection in the Controller Tags database in Studio 5000, corresponding to the Input and Output data that was set up in the Module Definition dialog section above. Each tag has a Data array that holds the actual data. The input tag also includes a boolean tag that indicates whether the I/O connection is faulted. The following table summarizes the tags created for each connection:

Tag Name	Description

<table>
<tr><td>[Name]:I.ConnectionFaulted</td><td>This boolean indicates the status of the I/O connection.</td></tr>
<tr><td>[Name]:I.Data</td><td>This is the Input Data, where the Outgoing Cyclic I/O Data from the Indirect Data in the RMC will appear. The data type of each array item is REAL.
If you selected to use a Sync Register, the first item in this array is the SyncIn Register, followed by the Indirect Data[0], Indirect Data[1], and so on:

<table><tr><th>Tag</th><th>With Sync Register</th><th>Without Sync Register</th></tr><tr><td>[name]:I.Data[0]</td><td>SyncIn Register</td><td>Indirect Data [0]</td></tr><tr><td>[name]:I.Data[1]</td><td>Indirect Data [0]</td><td>Indirect Data [1]</td></tr><tr><td>[name]:I.Data[2]</td><td>Indirect Data [1]</td><td>Indirect Data [2]</td></tr><tr><td>[name]:I.Data[3]</td><td>Indirect Data [2]</td><td>Indirect Data [3]</td></tr><tr><td>[name]:I.Data[4]</td><td>Indirect Data [3]</td><td>Indirect Data [4]</td></tr><tr><td>etc.</td><td>etc.</td><td>etc.</td></tr></table></td></tr>
<tr><td>[Name]:O.Data</td><td>This is the Output Data, which will be sent to the RMC's Incoming Cyclic I/O Data area. The data type of each array item is REAL.
If you selected to use a Sync Register, the first item in this array is the SyncOut register, and the data that will be written to the RMC begins at [name]:O.Data[1].

<table><tr><th>Tag</th><th>With Sync Register</th><th>Without Sync Register</th></tr><tr><td>[name]:O.Data[0]</td><td>SyncOut Register</td><td>Incoming Data Item[0]</td></tr><tr><td>[name]:O.Data[1]</td><td>Incoming Data Item[0]</td><td>Incoming Data Item[1]</td></tr><tr><td>[name]:O.Data[2]</td><td>Incoming Data Item[1]</td><td>Incoming Data Item[2]</td></tr><tr><td>[name]:O.Data[3]</td><td>Incoming Data Item[2]</td><td>Incoming Data Item[3]</td></tr><tr><td>[name]:O.Data[4]</td><td>Incoming Data Item[3]</td><td>Incoming Data Item[4]</td></tr><tr><td>etc.</td><td>etc.</td><td>etc.</td></tr></table></td></tr>
</table>

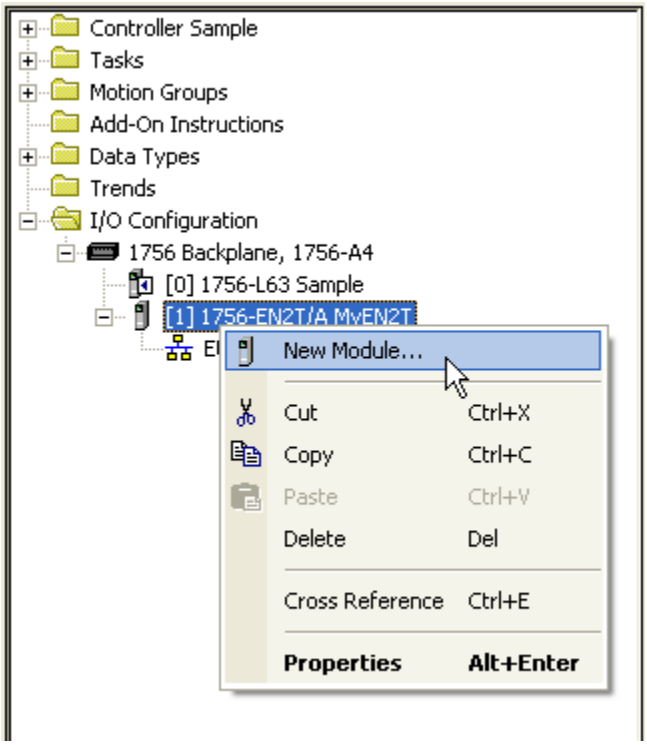
Now the communications are set up. See the **Performing Communications** section below for details on how to use the communications.

Configuring the PLC Connection Using a Generic Ethernet Module

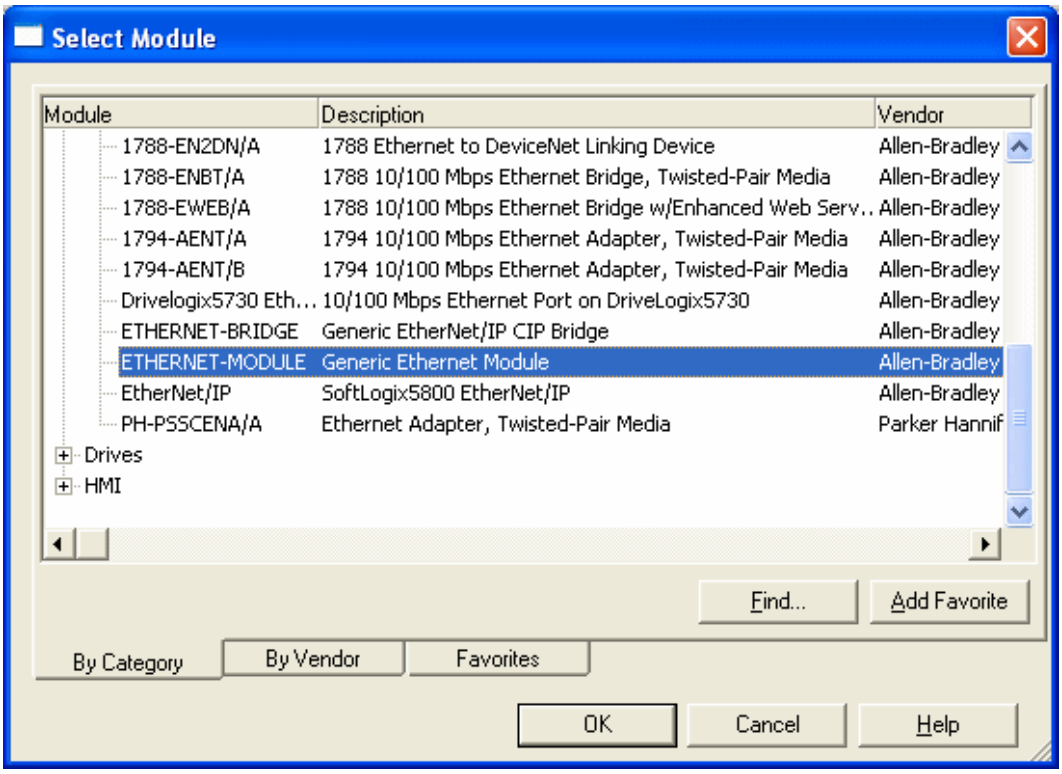
This method supports only a single connection and is required for RSLogix 5000 versions 19 and earlier. For later versions, or for multiple connections, use the procedure listed above using an EDS file.

Do the following in RSLogix 5000:

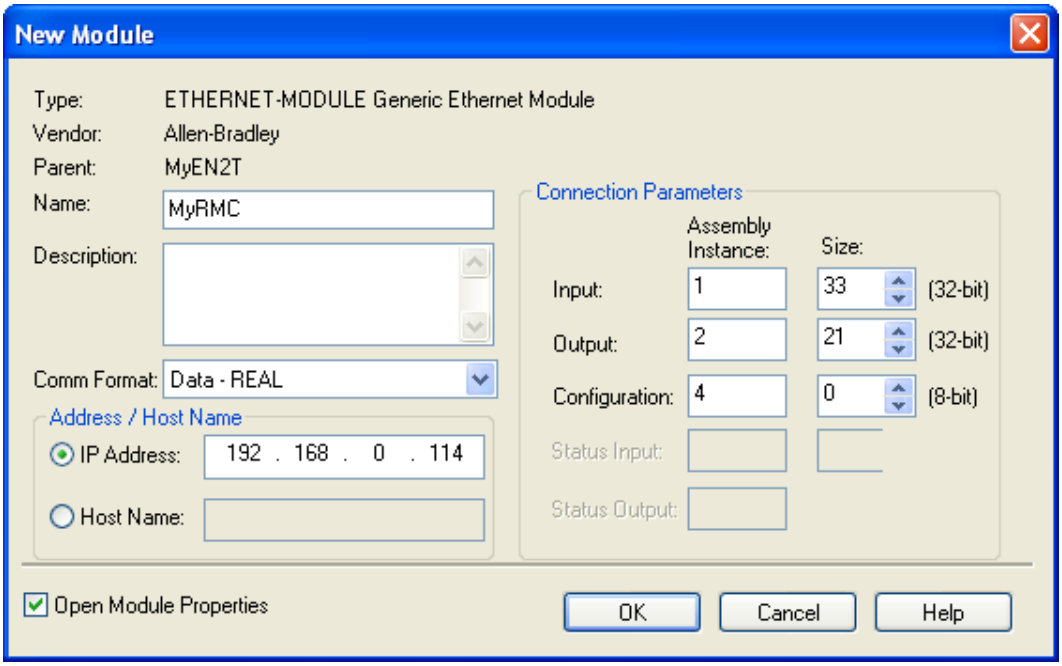
1. Start RSLogix 5000 and open the project to which you want to add an RMC I/O connection.
2. In the Controller Organization window, add a 1756-ENET/B, 1756-ENBT/A, or 1756-EN2T/A module under the I/O Configuration item. If the ControlLogix Ethernet module that you want to use already exists, then skip this step. Otherwise, refer to the Ethernet module's manual for details on adding the module.
3. In the Controller Organization window, right click on the 1756-ENET/B, 1756-ENBT/A, or 1756-EN2T/A under which you want to add the RMC. The following shortcut menu will be displayed:



4. In the shortcut menu that appears, click **New Module**. The following dialog box will be displayed:



5. Expand the **Communications** node, click the **ETHERNET-MODULE** type and click **OK**. The following dialog box will be displayed:



6. Fill in the fields in this dialog box as follows:

   Name: Type a valid module name for the RMC.

   Description: Type a description.

   Comm Format: Select **Data - REAL**.

Address/Host Name: Choose **IP Address** and enter the IP Address or host name of the RMC. The RMC must have its IP address set up to match this address.

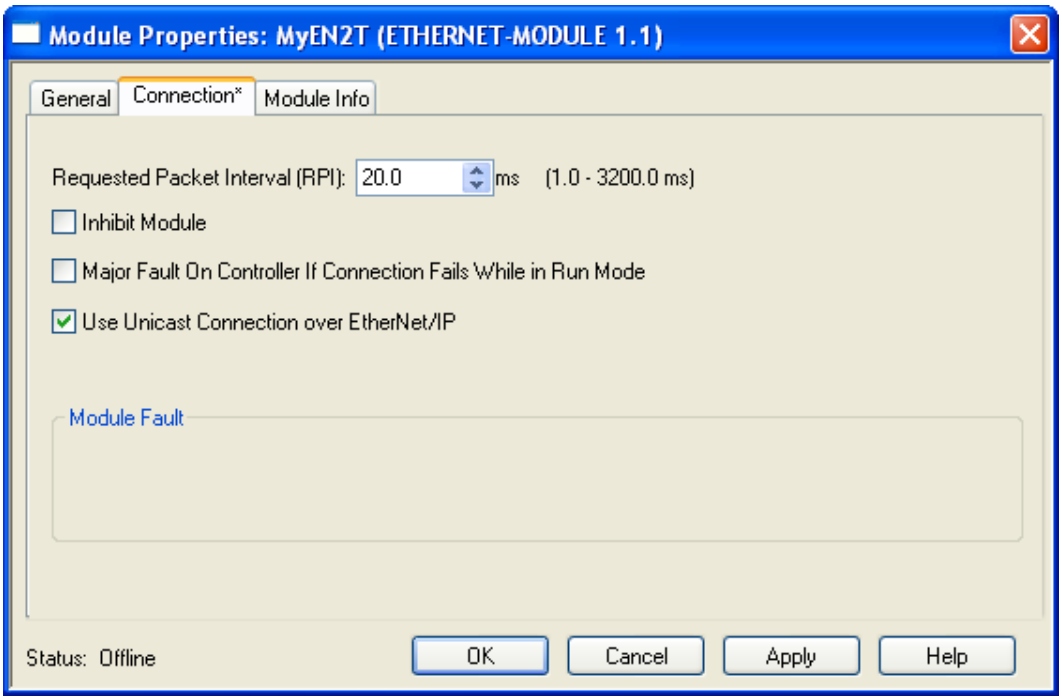
Input: Set the Assembly Instance to 1 and set the Size to the number of registers to transfer. If you selected to use the Sync Register, the Size should be one plus the number of registers of the RMC's Outgoing Cyclic I/O Data.

Output: Set the Assembly Instance to 2 and set the Size to the number of registers to transfer. If you selected to use the Sync Register, the Size should be one plus the number of registers of the RMC's Incoming Cyclic I/O Data.

Configuration: Set the Assembly Instance to 4 and set the size to 0.

Note: The Input and Output Assembly instances access the RMC75/150 I/O data, or Connection #1 I/O data on the RMC200.
To access Connection #2 data on the RMC200, set the Input Assemble Instance to 5, and the Output Assembly Instance to 6.
To access Connection #3 data on the RMC200, set the Input Assemble Instance to 7, and the Output Assembly Instance to 8.
See Setting up an EtherNet/IP I/O Connection for more details.

7. Click **Next**. The following dialog box will be displayed:



8. Enter the desired **Requested Packet Interval (RPI)**. This is the desired update rate. A commonly used RPI is 20.0 ms. Very low RPIs may flood the network, and reduce network reliability.
9. Make sure to check **Use Unicast Connection over EtherNet/IP**. This option is very important for reducing network bandwidth required, but is available only in RSLogix version 18.00.00 and newer.
10. Clear the **Inhibit Module** check box and set the **Major Fault On Controller if Connection Fails While in Run Mode** check box as required by your application.
11. Click **Finish**.

The above steps will allocate three tags in the Controller Tags database in RSLogix 5000. These tags correspond to the Input, Output, and Configuration data set up for the module under the Connection Parameters section above. The type of each tag is a special module-defined type created by RSLogix. Each special type has a Data field that holds the actual data. The following table summarizes the tags created for each module:

<table>
<tr><th>Tag Name</th><th>Type Of Data</th><th>Description</th></tr>
<tr><td>[Name]:I.Data</td><td>REAL[size]</td><td>This is the Input Data, where the Outgoing Cyclic I/O Data from the Indirect Data in the RMC will appear.
If you selected to use a Sync Register, the first item in this array is the SyncIn Register, followed by the Indirect Data[0], Indirect Data[1], and so on:
<table>
<tr><th>Tag</th><th>With Sync Register</th><th>Without Sync Register</th></tr>
<tr><td>[name]:I.Data[0]</td><td>SyncIn Register</td><td>Indirect Data [0]</td></tr>
<tr><td>[name]:I.Data[1]</td><td>Indirect Data [0]</td><td>Indirect Data [1]</td></tr>
<tr><td>[name]:I.Data[2]</td><td>Indirect Data [1]</td><td>Indirect Data [2]</td></tr>
<tr><td>[name]:I.Data[3]</td><td>Indirect Data [2]</td><td>Indirect Data [3]</td></tr>
<tr><td>[name]:I.Data[4]</td><td>Indirect Data [3]</td><td>Indirect Data [4]</td></tr>
<tr><td>etc.</td><td>etc.</td><td>etc.</td></tr>
</table></td></tr>
<tr><td>[Name]:O.Data</td><td>REAL[size]</td><td>This is the Output Data which will be sent to the RMC's Incoming Cyclic I/O Data area.
If you selected to use a Sync Register, the first item in this array is the SyncOut register, and the data that will be written to the RMC begins at [name]:O.Data[1].
<table>
<tr><th>Tag</th><th>With Sync Register</th><th>Without Sync Register</th></tr>
<tr><td>[name]:O.Data[0]</td><td>SyncOut Register</td><td>Incoming Data Item[0]</td></tr>
<tr><td>[name]:O.Data[1]</td><td>Incoming Data Item[0]</td><td>Incoming Data Item[1]</td></tr>
<tr><td>[name]:O.Data[2]</td><td>Incoming Data Item[1]</td><td>Incoming Data Item[2]</td></tr>
<tr><td>[name]:O.Data[3]</td><td>Incoming Data Item[2]</td><td>Incoming Data Item[3]</td></tr>
<tr><td>[name]:O.Data[4]</td><td>Incoming Data Item[3]</td><td>Incoming Data Item[4]</td></tr>
<tr><td>etc.</td><td>etc.</td><td>etc.</td></tr>
</table></td></tr>
<tr><td>[Name]:C</td><td>SINT[400]</td><td>This is not used. Notice that a full 400 bytes is allocated by the ControlLogix regardless of how many are actually configured to be sent to the RMC.</td></tr>
</table>

Export RMCTools Tags to Logix Designer

This step is not required but may reduce the time spent creating tags in the PLC and improve the readability of the EtherNet/IP data in Logix Designer. See Using Logix Designer Export Components for more details.

The Logix Designer Export feature does the following:

- Exports the RMCTools input and output tag names into a structure with a tag name for each register.
- Includes example synchronization ladder logic.
- Supports Sync register or no Sync register. See **Performing Communications** below for an explanation of the Sync register.
- Supports multiple connections (RMC200 only).

The exported components can then be imported to Logix Designer.

To export tags to a Logix Designer Import/Export file:

1. Before starting, make sure to have the following information:
    1. The name of the RMC as used in your Logix Designer project.
    2. If using the RMC200, the number of I/O connections.
    3. The number of PLC Input Data registers.
    4. The number of PLC Output Data registers.
2. In the RMCTools Project pane, expand the **Modules** folder, double-click the CPU module, and choose the **EtherNet/IP** page.
3. In the **Logix Designer Export** section, click **Export to Logix Designer**.
4. Complete the Logix Designer Export wizard. It will prompt you to save an .L5K file.

To import the .L5K file to Logix Designer:

1. In Logix Designer, in a ladder logic routine, right-click a rung and choose **Import Rungs**.
2. Browse to the .L5K file and click **Open**.
3. The example ladder logic, tags, and user-defined types will be imported.
4. For a detailed explanation of the logic, see Using Logix Designer Export Components.

Performing Communications

This section explains the EtherNet/IP cyclic communication at its basic level. For details on how to use the logic resulting from the Logix Designer Export feature, see Using Logix Designer Export Components.

Once the EtherNet/IP connection is configured and applied to the PLC, the communications will automatically start, assuming the PLC and RMC are both on a properly set up Ethernet network. Notice that the ControlLogix will communicate even when it is in Program mode.

Reading Data from the RMC

The data for each connection will automatically update each Requested packet interval, and you can use the data as you wish. You cannot write to the Input Data. The Input Data will contain the Indirect Data from the RMC. However, if you selected to use the Sync Register, the first item in the Input Data array will be the SyncIn value, followed by the Indirect Data from the RMC.

The SyncIn is used only for synchronizing commands with the logic, as explained below. The SyncIn and SyncOut registers are only visible in the PLC. They are not visible in RMCTools. If you are using multiple connections, keep in mind that the connections are independent of each other, and the data may not be transferred at exactly the same time, even if the Sync Registers are used and are changed simultaneously.

Writing to the RMC - General

For each connection *without* a Sync Register, the Output Data is written to the RMC when any value in the Output Data changes.

For each connection *with* a Sync Register, the Output Data for each connection is sent to the RMC at each Requested Packet Interval, but the RMC ignores it until the SyncOut register for that connection changes. The first item in the Output Data array is the SyncOut register, followed by the registers that will be sent to the Incoming Data location in the RMC. Use the following procedure to write to the RMC with a Sync Register:

1. **Wait Until the Sync In and Sync Out Registers Match**
   If they do not match, then this means that another write is in progress.
2. **Write to the Output Data**
   Write the desired data to the Output Data array.
3. **Change the Sync Out Register**
   An easy way to do this is to add one to it. However, you must take care to handle overflowing this register (the Sync register is a REAL). One method is to add one and then MOD it with some large number, such as 10000. This will make the register count from 0 to 9,999, and then wrap back down to 0 without an error.
4. **Wait Until the Sync In and Sync Out Registers Match**
   This indicates that the RMC has received the data and processed it.

If you are using multiple connections with sync registers, you must change the Sync Register independently for each connection. The connections are independent of each other, and the data may not be transferred at exactly the same time, even if the Sync Registers are changed simultaneously.

Sending Commands to the RMC

If you are writing to the Command Area, either directly or via the Indirect Data Map, Delta recommends using a Sync Register and following the procedure below to send commands to the RMC. The Output Data is sent to the RMC each RPI, but the RMC ignores it until the SyncOut register changes. The first item in the Output Data array is the SyncOut register, followed by the registers that will be sent to the Incoming Cyclic I/O Data location in the RMC.

1. **Wait Until the Sync In and Sync Out Registers Match**
   If they do not match, then this means that another write is in progress.
2. **Clear Old Commands from the Command Registers**
   Clear old commands from the command registers in the Output Data. Otherwise, when the Sync Out register is changed, the commands would be re-issued. One method of clearing the old commands is to fill the Output Data array with zeroes (except the SyncOut value).
3. **Write to the Command Registers**
   Write the Command registers and all required command parameters to the Output Data for all commands you want to issue. You can issue up to one command per axis. Leave the Command register set to 0 for each axis that will not receive a command.

   For example, if a portion of the Output Data is going to the Command Area for Axis 0, and you wish to issue a Move Absolute Command (20) to Axis 0 with a position of 6.7, a Speed of 3, and Accel and Decel of 50, you would write the following:

Value
20 (for a move Absolute Command)
6.7 (Position)
3 (Speed)
50 (Accel)
50 (Decel)
0 (Direction)

   Use the online help for each command to find out how many parameters a command has and what they mean. Make sure to write to all the parameters that the command uses. You do not need to write to command parameters that are not used by the command.
4. **Change the Sync Out Register**
   The easiest way to do this is to add one to it. However, you must take care to handle overflowing this register (the Sync register is a REAL). One method is to add one and then

MOD it with some large number, such as 10000. This will make the register count from 0 to 9,999, and then wrap back down to 0 without an error. Take care to ensure that you only update the Sync Out Register once so that the commands do not get re-issued.

5. **Wait Until the Sync In and Sync Out Registers Match**
   This indicates that the RMC has received the command and issued it. It is important to wait until the SyncIn and SyncOut match before using the status bits in the Input Data (if the Input Data includes any status bits). See the Using an EtherNet/IP I/O Connection topic for how problems can occur if this step is ignored.

For multiple connections to one RMC, each connection has an individual Sync Register. The connections are independent of each other, and the data may not be transferred at exactly the same time, even if the Sync Registers are changed simultaneously. Therefore, it is best practice that if commands need to be sent simultaneously, that they are sent in the same Output Data block.

See Using an EtherNet/IP I/O Connection for further details.

Reading and Writing other registers

To read and write other registers in the RMC that are not included in the Incoming or Outgoing Cyclic I/O Data, you can use MSG blocks (see Using Allen-Bradley Controllers via Message Block). EtherNet/IP I/O and MSG blocks can be used simultaneously.

Another option for writing to other RMC registers is to create user programs that move data from the variable table to other RMC registers.

Reading DWORDs from the RMC

EtherNet/IP I/O will exchange data between the PLC and RMC as if they are all REAL values. However, some data items are of type DWORD or DINT, such as the Status Bits and Error Bits, which are of DWORD type. These values will not be displayed properly in the PLC's I/O Input and Output arrays.

To view DWORD or DINT values properly in the PLC, use the COP instruction to copy them from the Input array (:I) into a tag of DINT type. To write DINT values to the RMC, use the COP instruction to write the values to the Output array (:O).

Handling Broken Connections

See the Handling Broken EtherNet/IP I/O Connections topic for details.

See Also

EtherNet/IP Overview | Setting up an EtherNet/IP I/O Connection | Using an EtherNet/IP I/O Connection | Handling Broken EtherNet/IP I/O Connections | Troubleshooting EtherNet/IP I/O | Multiple EtherNet/IP I/O Connections | EtherNet/IP I/O Performance

---

6.10.3. Using Logix Designer Export Components

This topic explains how to use the communication data tags and example ladder logic generated by the RMCTools' Logix Designer Export feature. This is only available for the Rockwell Studio 5000 Logix Designer.

The EtherNet/IP cyclic I/O data can be difficult to decipher and use. The exported tags and logic from RMCTools are designed to make the data easy to read and simplifies the logic for the user.

How to Export and Import

Before starting:

1. Configure the EtherNet/IP connection in RMCTools as described in Using Allen-Bradley Controllers via EtherNet/IP I/O.

2. Make sure to have the following information:
    1. The name of the RMC as used in your Logix Designer project. The name must be less than 32 characters, but may need to be even shorter, as explained in item 3 below.
    2. If using the RMC200, the number of I/O connections.
    3. The number of PLC Input Data registers.
    4. The number of PLC Output Data registers.
3. Logix Designer limits the length of imported tag names. This means the lengths of the RMC name, axis names and variable tag names must limited such that any tag does not exceed 40 characters. For example, the exported Axis 0 Status Bits tag name is MyRMCNameMyAxisNameStatusBits and the Axis 0 Command Parameter 1 is MyAxisName_Command_Parameter_1. The wizard allows you to preview the tags to be exported.

To export from RMCTools to a Logix Designer Export/Import file:

1. In the RMCTools Project pane, expand the **Modules** folder.
2. Double-click the CPU module, and choose the **EtherNet/IP** page.
3. In the **Logix Designer Export** section, click **Export to Logix Designer**.
4. Complete the Logix Designer Export wizard. It will prompt you to save an .L5X file.

To import the .L5X file to Logix Designer:

1. In Logix Designer, in a ladder logic routine, right-click a rung and choose **Import Rungs**.
2. Browse to the .L5X file and click **Open**.
3. In the Import Configuration dialog, click **OK** (if you are re-importing a modified file, see **Re-importing Logic** below).

Imported Components

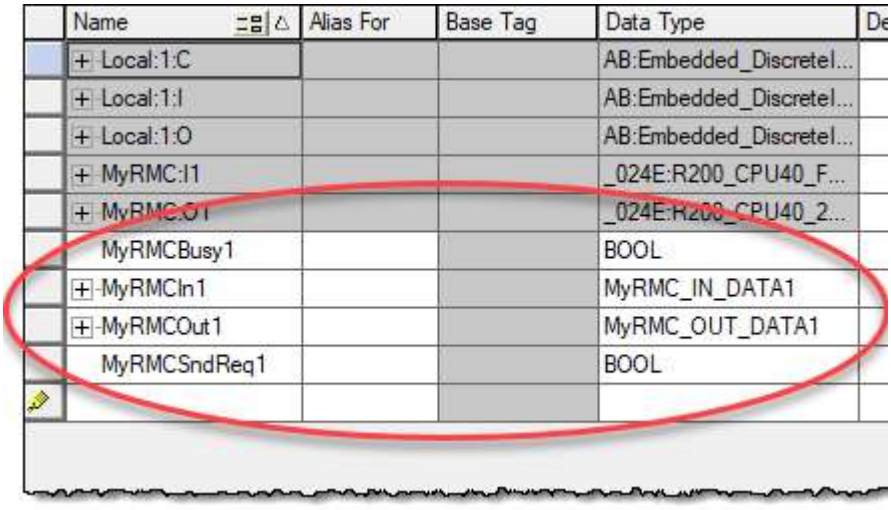
The following components are imported into Logix Designer:

Controller Tags:
One each of the following per connection:

- **Input Data Tag** (in the example, **MyRMCIn1**) Contains all the tags for the input data coming from the RMC. This tag uses a special user-defined data type (MyRMC_IN_DATA1). See **User-Defined Data Types** below.
- **Output Data Tag** (in the example, **MyRMCOut1**) Contains all the tags for the output data going to the RMC. The data type is an imported user-defined data type.

Example:

The tags circled in the image below are imported for an RMC with the module name **MyRMC**.



This tag uses a special user-defined data type (MyRMC_OUT_DATA 1).
See **User-Defined Data Types** below.

- **Send Request bit** (in the example, **MyRMCSndReq1**)
  Used to apply the output data to the RMC. First it copies the output data to the output buffer. If the Sync Register is used, it then changes the sync register to trigger the RMC to apply the data.
- **Busy bit** (in the example, **MyRMCBusy1**)
  Indicates that the Sync Register write transaction is in process. This bit is included only if the connection uses the Sync Register.

Ladder Logic

- **For input data**

  One rung is imported for copying input data. If multiple connections are used, the rung uses multiple branches for each connection. See **Input Data Logic** below for details.

- **For output data**

  If the connection uses a Sync Register, two rungs per connection are imported for copying output data and managing the Sync register. See **Output Data Logic With a Sync Register** below for details.

  If the connection does not use a Sync Register, one rung per connection is imported for applying output data. See **Output Data Logic Without a Sync Register** below for details.

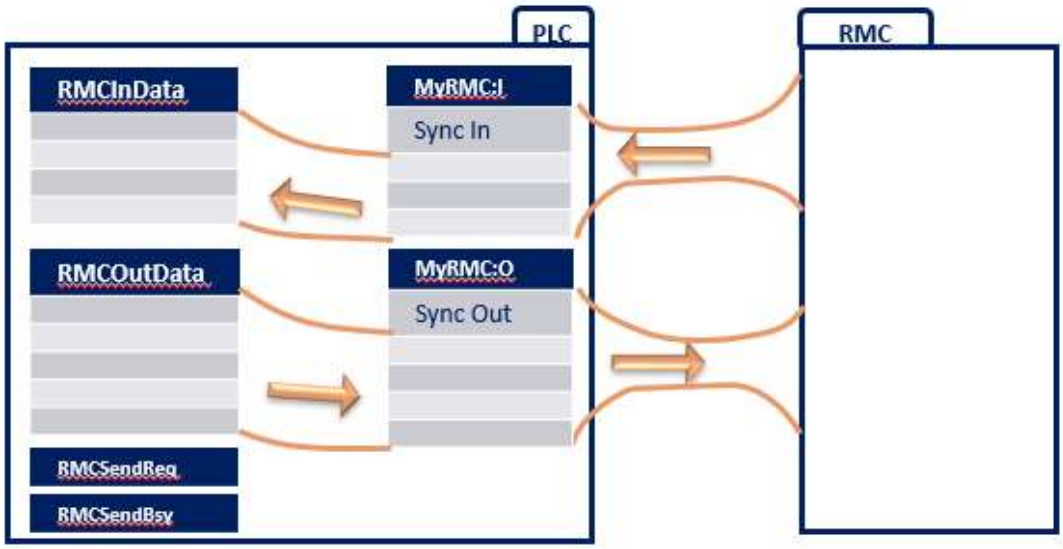
User-Defined Data Types:

These user-defined data types (UDTs) are applied to the imported input and output tags. All the UDTs are prefaced by the module name of the RMC as used in Logix Designer.

- ***RMCname*_IN_DATA**
  Input data type
- ***RMCname*_OUT_DATA**
  Output data type
- ***RMCname_axisname*StatusBits**
  Axis Status Bits data types. Only if Axis Status Bits registers are included in the input data. One Status Bits data type per axis will be created, since the valid status bits may vary between axes.
- ***RMCname_axisname*ErrorBits**
  Axis Error Bits data types. Only if Axis Error Bits registers are included in the input data. One

Error Bits data type per axis will be created, since the valid status bits may vary between axes.

The imported data tags and the data flow can be visualized as follows:



Input Data Logic

The input data rung continuously copies the difficult-to-decipher EtherNet/IP input data array **MyRMC:I1.Data** to the easy-to-read input tag **MyRMCIn1**.

This example was generated for an RMC with the module name **MyRMC**, using a connection with a Sync Register. The Sync Register at index 0 in the array. The copy starts at index 1, not index 0, because the Sync Register at index 0 need not be copied to the input tag. If the connection did not use the Sync Register, the copy would start at index 0.

Output Data Logic With a Sync Register

When the RMC EtherNet/IP connection has been set to use the Sync Register, two rungs are imported to handle the output data going to the RMC.

The two rungs work together to apply data to the RMC and keep the communications synchronized with the Sync Register. The first rung copies the data and increments the Sync Register to trigger the RMC to apply the data. The second rung waits for the Sync Register value to be echoed by the RMC, to ensure that the data was applied to the RMC before releasing control.

When using these rungs in an application, the PLC must do the following to apply data to the RMC:

1. Set the desired values in the output tag.
2. Set the Send Request bit.

When the Send Request bit and Send Busy bits clear, the data has been applied to the RMC. This indicates that the transaction is complete, and any PLC logic depending on it can now proceed.

The following rung analysis applies to the example ladder logic below, which was generated for an RMC with the module name **MyRMC**.

First Rung Details

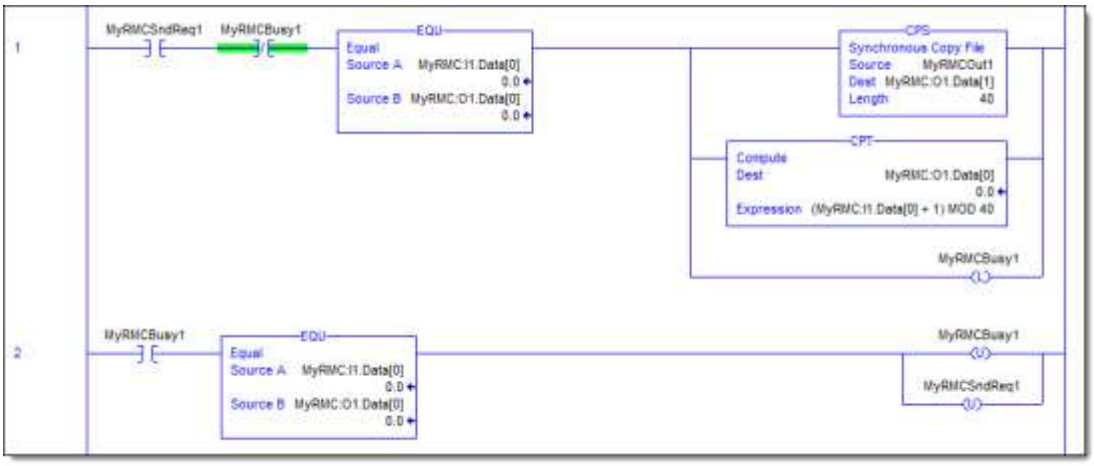
When the Send Request bit (**MyRMCSndReq1**) is set, the first rung copies the easy-to-read output tag **MyRMCOut1** to the difficult-to-decipher EtherNet/IP output data array **MyRMC:O1.Data**. It does so by performing these steps:

- Checks to make sure the Busy bit is not set, which means a previous transaction is still in process.
- Waits for the SyncIn and SyncOut registers to be equal. If they are not equal, that means a previous transaction is still in process.
- Copies the easy-to-read output tag **MyRMCOut1** to the difficult-to-decipher EtherNet/IP output data array **MyRMC:O1.Data**. The copy starts at index 1, not index 0, because the Sync Register at index 0 need not be copied to the input tag.
- Changes the SyncOut register by adding 1 to it. It uses a modulo so that the value does not increase beyond its valid range.
- Latches the Busy bit.

Second Rung Details

When the Busy bit **MyRMCBusy1**) is set, the second rung ensures that the data was applied to the RMC before releasing control. It does so by performing these steps:

- Waits for the SyncIn and SyncOut bits to be equal. This means the transaction has completed.
- Clears the Send Request and Busy bits.



Output Data Logic Without a Sync Register

When the RMC EtherNet/IP connection has been set to *not* use the Sync Register, one rung is imported to handle the output data going to the RMC.

On the rising edge of the Send Request bit, the output rung copies the easy-to-read output tag **MyRMCOut1** to the difficult-to-decipher EtherNet/IP output data array **MyRMC:O1.Data**.

When using this rung in an application, the PLC must do the following each time the data is to be applied to the RMC:

1. Set the desired values in the output tag.
2. Set the Send Request bit.

The Send Request bit will clear as soon as the copy is complete. Without a Sync register, there is no indication from the RMC to know when the data was actually applied. The user should take care in programming the PLC logic to account for this.

This example was generated for an RMC with the module name **MyRMC**. This example does not use the Sync Register.

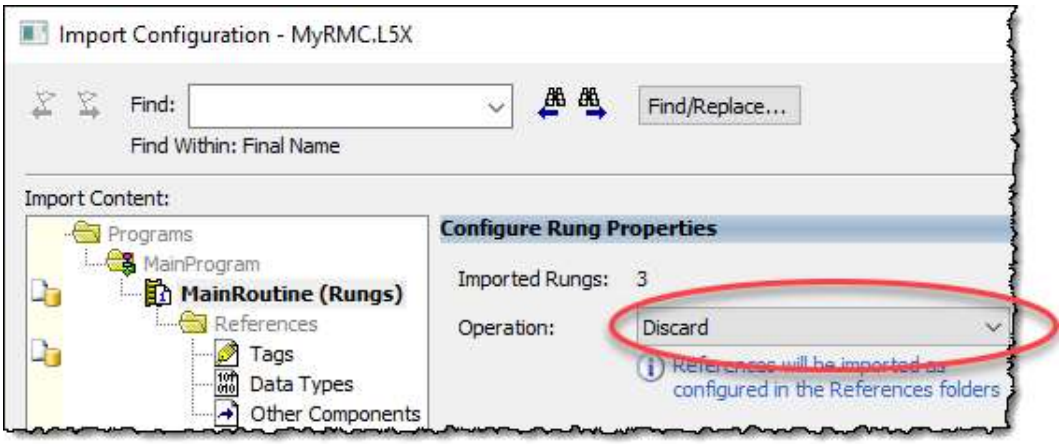
Re-importing Logic

If you change the EtherNet/IP data tags in the RMC after initially exporting tags and ladder logic, you can re-import the logic to Logix Designer. In the Logix Designer Import Configuration dialog, you must specifically select which imported tags you want to overwrite the existing tags. The Import Configuration dialog defaults to use the existing Logix Designer tags, not the imported tags. When changing data, you will need to re-import both tags and the user-defined data types. If you change the data length, you will need to make sure to change the data lengths both in RMCTools before exporting, and in the defined RMC module in Logix Designer.

Example

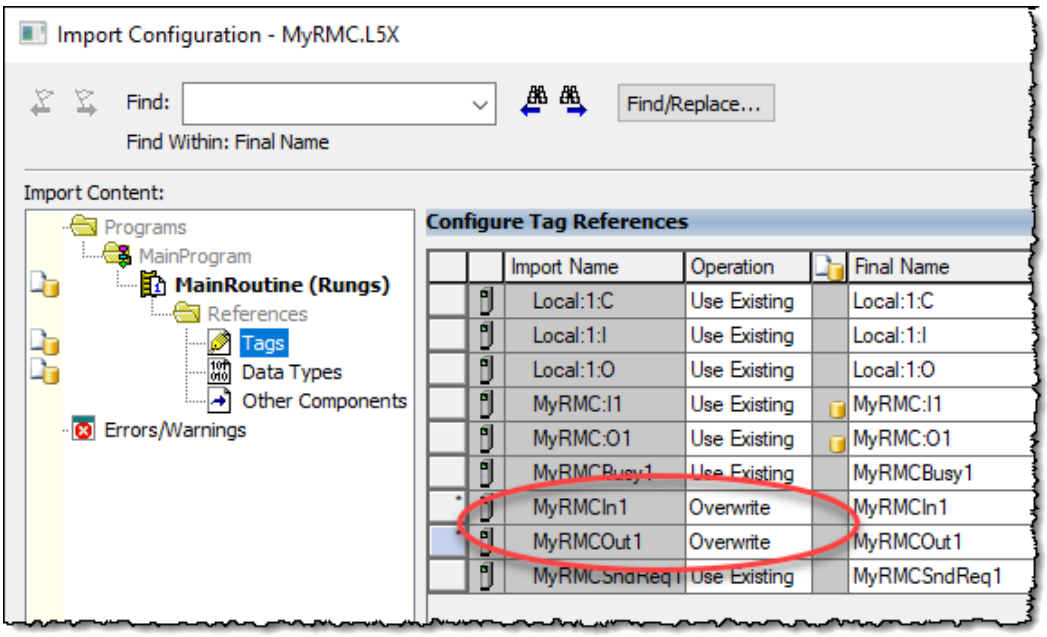
If you change some tag names in the RMC, you may wish to import and overwrite the tags in Logix Designer, but not overwrite the ladder logic rungs. You would do the following:

1. In Logix Designer, in a ladder logic routine, right-click a rung and choose **Import Rungs**.
2. Browse to the .L5X file and click **Open**.
3. In the Import Configuration dialog, in the **Import Content** box, choose **Main Routine**.
4. In the **Operation** box, choose **Discard**.

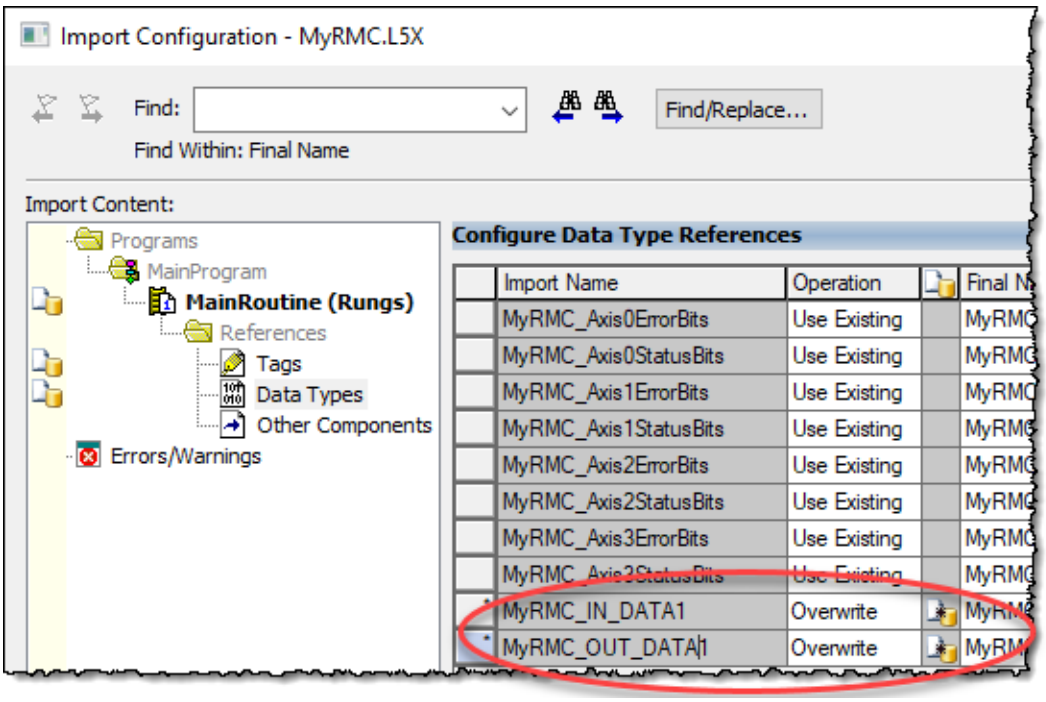


5. In the **Import Content** box, choose **Tags**.

6. For the tags you wish to use (typically the **In** tag and the **Out** tags), choose **Overwrite**.



7. In the **Import Content** box, choose **Data Types**.
8. For the data types you wish to use (typically the **In** tag and the **Out** datatypes), choose **Overwrite**.



9. Click **OK** to import the selected items.

See Also

Using Allen-Bradley Controllers via EtherNet/IP I/O

6.10.4. Using AutomationDirect DL-series PLCs

The RMC family of controllers can communicate with AutomationDirect DirectLogic PLCs via Ethernet or serial RS-232/485.

RMC and AutomationDirect DirectLogic PLC Compatibility

AutomationDirect PLC	RMC75E, RMC150E, RMC200	RMC75S
DL05/06	Requires **H0-ECOM100** module. Use Modbus/TCP master.	Requires **D0-DCM** Module. Use Modbus/RTU master.
DL105	Not Compatible	Not Compatible
DL205	Requires **H2-ECOM100** module. Use Modbus/TCP master.	DL250-1 and DL260 support Modbus/RTU master on the CPU. No support for 230 and 240.
DL305	Not Compatible	Not Compatible
DL405	Requires **H4-ECOM100** module. Use Modbus/TCP master.	Not Compatible

Example Programs

Delta provides example PLC programs to help you quickly set up the communications between your PLC and the RMC. See the downloads section of Delta's website at https://deltamotion.com.

Communicating via Ethernet

Use the AutomationDirect **Hx-ECOM100** modules to communicate with the RMC via Ethernet. Refer to the AutomationDirect HX-ECOM-M manual for instructions on how to use it.

Setting up the Hx-ECOM100

For each RMC that the PLC will communicate with, use NetEdit to add a Peer-to-Peer configuration to the Hx-ECOM100 for Modbus/TCP. Specify the Modbus/TCP protocol and enter the IP Address of the RMC. The Port number must be 502. The value in the Unit ID is not important.

Using the to the Hx-ECOM100

Use the Hx-ECOM100 as a Network Client (master), as described in the **MODBUS TCP/IP for H0/H2/H4–ECOM100** chapter, **Network Client (master) Operation** section of the Hx-ECOM100 manual.

Calculating RMC Register Addresses

Use the Modbus Addressing topic for instructions on finding the Modbus/TCP addresses of the desired registers in the RMC. Then convert the addresses to octal as described in the Hx-ECOM100 manual.

The AutomationDirect PLCs cannot access all the registers in the RMC. Therefore, use the Indirect Data Map to map the desired registers to one location that the PLC can read. This also makes the communications more efficient.

Communicating with the RMC75S

For the DL05/06 series, set up the **D0-DCM** module as a Modbus/RTU master. For the 250-1 or 260, configure the second CPU port as a Modbus/RTU master.

Calculating RMC75S Register Addresses

Use the RMC75 Register Map to find the Modbus/TCP addresses of the desired registers in the RMC75S. Then convert the addresses to octal as described in the Hx-ECOM100 manual.

The AutomationDirect PLCs cannot access all the registers in the RMC75S. Therefore, use the Indirect Data Map to map the desired registers to one location that the PLC can read. This also makes the communications more efficient.

See Also

Communications Overview | Register Maps

6.10.5. Using Factory Talk View with the RMC

FactoryTalk View can communicate with the RMC75E, RMC75S, RMC150E, and RMC200. An example FactoryTalk View and RMC75 project is available on the Downloads page Delta's website at https://deltamotion.com/dloads. The example project can communicate with either an RMC75S or RMC75E. The only difference is in setting up the communications, as described in this topic.

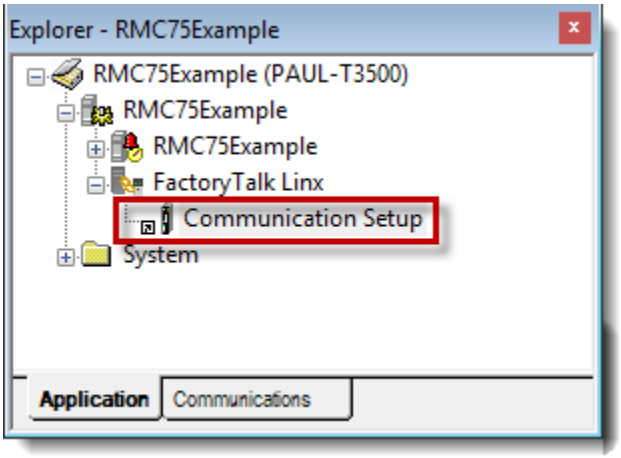
FactoryTalk View Studio uses RSLinx Enterprise. RSLinx Enterprise is different from RSLinx Classic. RSLinx Enterprise is the runtime data collection engine used by FactoryTalk View Studio. Therefore, you cannot use an EDS file or a Topic as in RSLinx Classic.

To communicate with the RMC from FactoryTalk View, you will need to add the RMC to RSLinx Enterprise as a MicroLogix device, then create a Device Shortcut that references it. Notice that you cannot browse tags in the RMC via RSLinx. Once you have created a Device Shortcut, you will use it to enter direct addresses in FactoryTalk View.

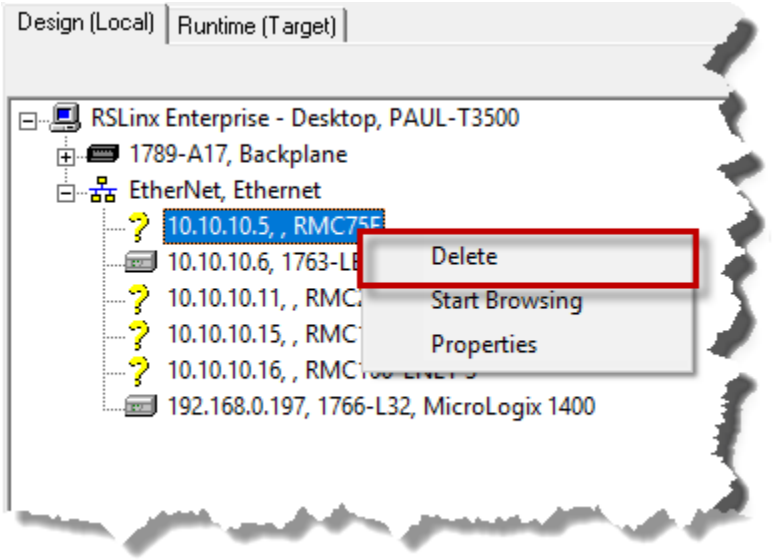
The instructions below are for **FactoryTalk View Studio** release number 10.00.00.

To add the RMC75E, RMC150E, or RMC200 (Ethernet) Device Shortcut to RSView:

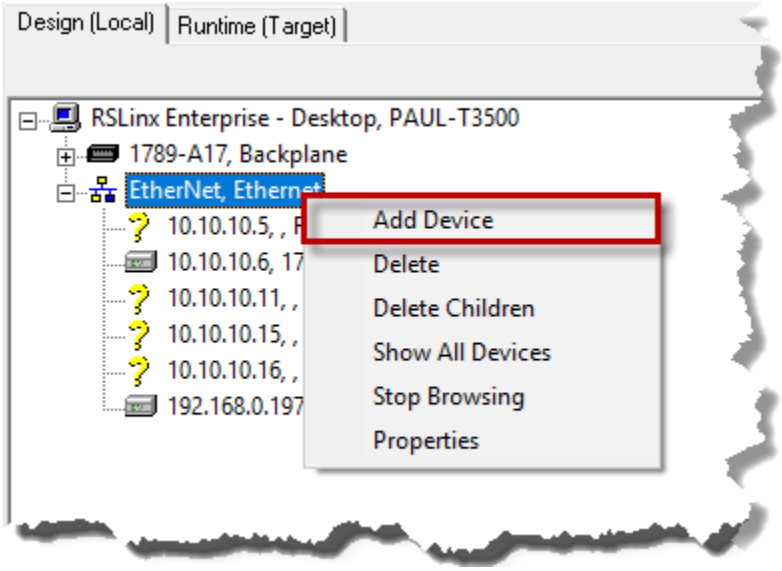
1. Determine the IP address of the RMC.
2. Disconnect the RMC from your network. This is so RSLinx Enterprise will not see it while you are configuring it.
3. Open an existing project or start a new project.
4. In the Explorer pane, on the **Application** tab, expand the **FactoryTalk Linx** node and double-click **Communication Setup**.



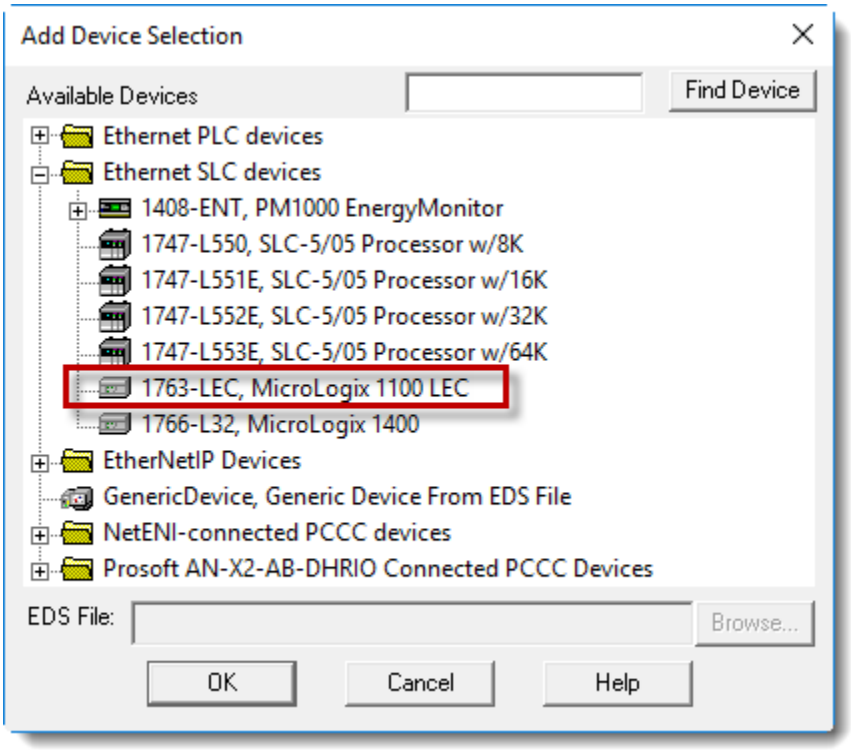
5. On the **Design (Local)** tab, expand the Ethernet node. If your RMC is listed, right-click the RMC and choose **Delete** and click **OK**.



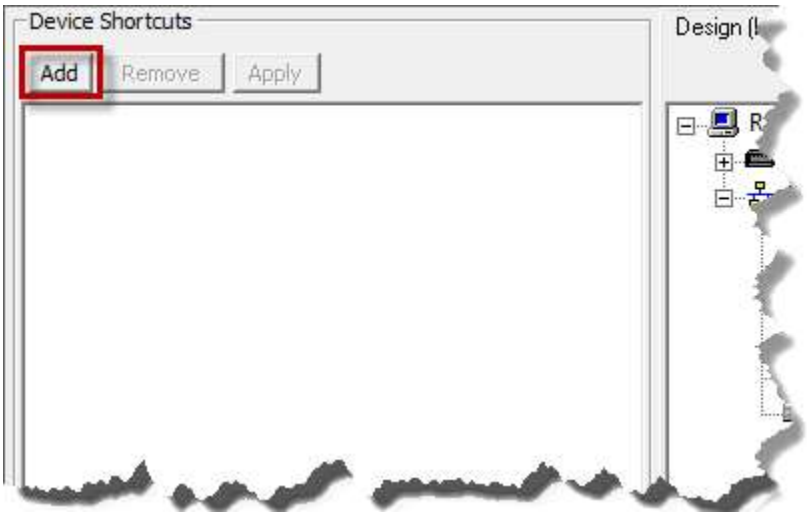
6. Right-click the **Ethernet** node and choose **Add Device**.



7. In the **Add Device Selection** dialog, expand **Ethernet SLC Devices**, choose the MicroLogix 1100 LEC and click **OK**.



8. In the **Device Properties** dialog, enter a name, such as "MYRMC", enter the IP address of your RMC, and click **OK**.
   The RMC should now appear as a 1763 series device in the list.
9. If you are using a Target, repeat the previous three steps on the **Runtime (Target)** tab to add the same RMC to the list.
10. In the **Device Shortcuts** area of the **Communication Setup** dialog, click **ADD**. Enter a name, such as "RMC".

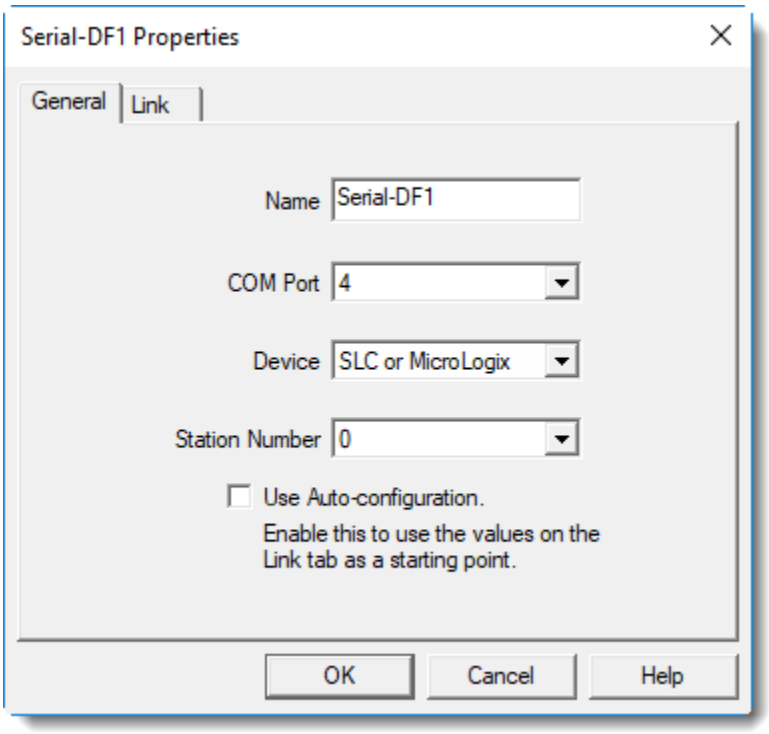


11. Now, you must make sure the shortcut is assigned to the RMC on the **Runtime (Local)** and **Runtime (Target)** tabs. In the **Device Shortcuts** box, make sure the shortcut you added is selected. Then, on the **Runtime (Local)** tab, select the RMC, and similarly, on the **Runtime (Target)** tab, select the RMC.
12. In the **Device Shortcuts** box, click **Apply** to save the changes to the shortcuts.

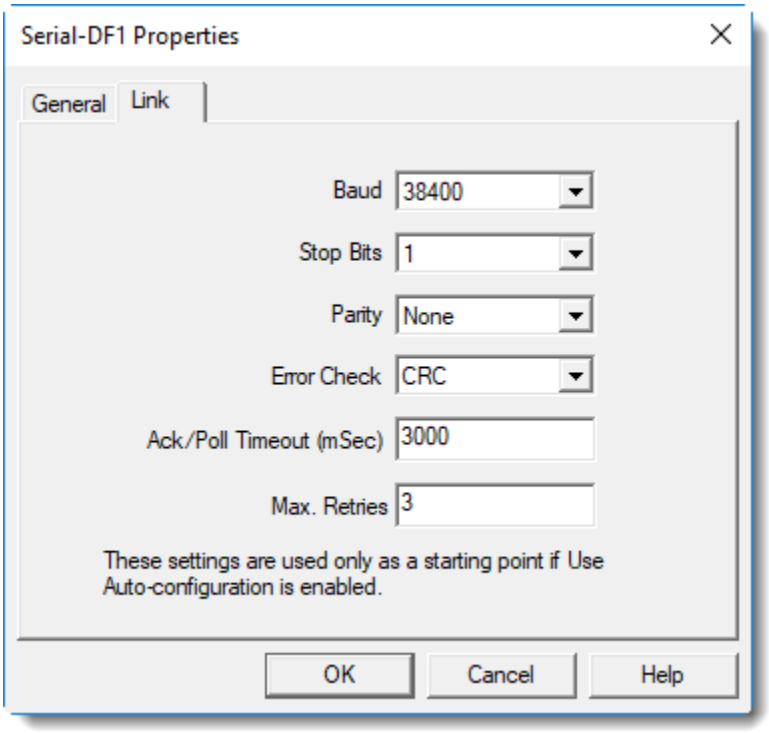
13. Click **OK** to close the **Communication Setup**.
14. Connect the RMC to the network.

To add the RMC75S (serial) Device Shortcut to RSView:

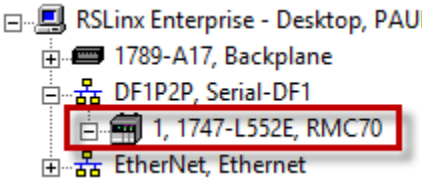
1. Determine which serial port will be used to connect to the RMC75S.
2. Open an existing project or start a new project.
3. In the Explorer pane, expand the **FactoryTalk Linx** node and double-click **Communication Setup**.
4. On the **Runtime (Local)** tab, right-click **RSLinx Enterprise** and choose **Add Driver**. Choose **Serial DF1** and click **OK**.
5. Enter a **Name**, such as "Serial-DF1 1".
6. In the **COM Port** box, choose the COM port you will be using.
7. In the **Device** box, choose **SLC or MicroLogix**. The **Station Number** is unimportant. The configured General tab is shown below:



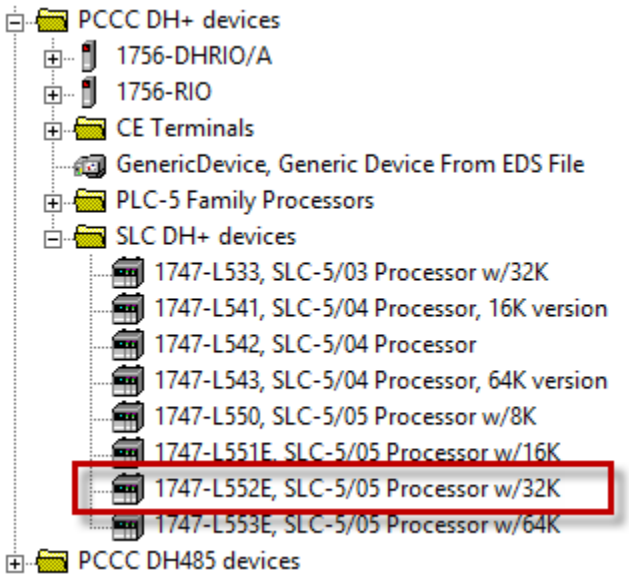
8. On the **Link** tab, set the Baud to **38400**, and the Error Check to **CRC**. The **Link** tab should now be configured as below:



9. Click **OK**.
10. RSLinx Enterprise should start browsing and find the RMC75 on the newly created device:



11. If you are using a Target, repeat these steps on the **Runtime (Target)** tab. You must manually add the RMC75 on the **Runtime (Target)** tab. Select the PLC boxed in the below image.

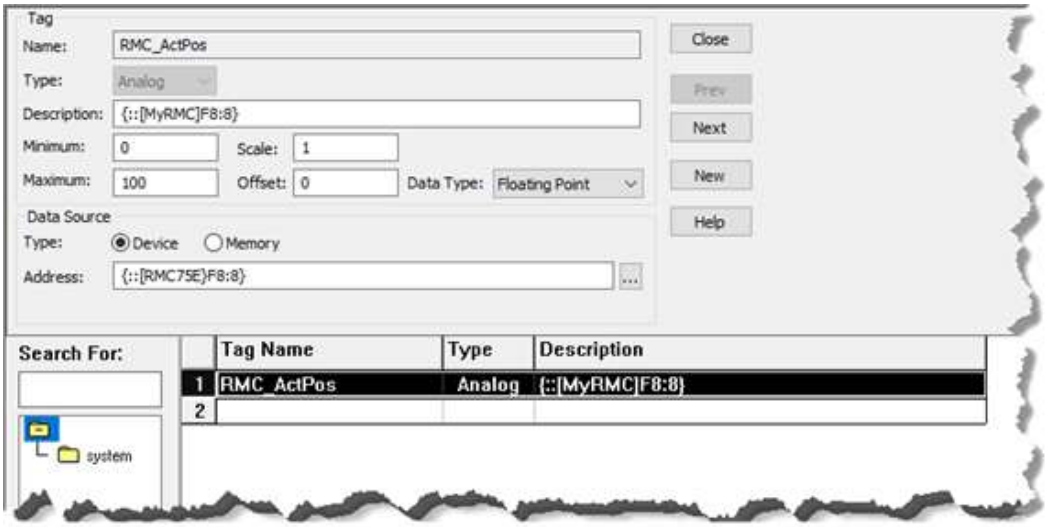


12. In the **Device Shortcuts** area of the **Communication Setup** dialog, click **ADD**. Enter a name, such as "RMC75S".
13. Now, you must make sure the shortcut is assigned to the RMC75S on the **Runtime (Local)** and **Runtime (Target)** tabs. In the **Device Shortcuts** box, make sure the shortcut you added is selected. Then, on the **Runtime (Local)** tab, select the RMC75S, and similarly, on the **Runtime (Target)** tab, select the RMC75S.
14. Click **OK** to close the **Communication Setup**.

To add a tag using the Device Shortcut (RMC75S, RMC75E, RMC150E, or RMC200):

1. To add a tag, expand the **HMI Tags** node, and double-click **Tags**. Select the bottom line in the Tag Name list.
2. In the **Name** box, enter a name, such as RMC_ActPos.
3. In the **Type** box, choose **Analog**. In the **Data Type** box, choose **Floating Point**.
4. In the **Data Source** area, choose **Device**.
5. The Address must be entered in direct format. The format for REAL values is {::[*ShortcutName*]F*n*:*x*} , where n is the file number and x is the element number.
   For example, with a device shortcut of "RMC75E" and the Axis 0 Actual Position tag (address F8:8), the Address would be:
   {::[RMC75E]F8:8}

6. Click **Accept**. An example of a configured tag is below.



Note:
Using tags with direct addressing in a Numeric Display does not seem to work properly. When using a Numeric Display, enter the direct address as the Value. The direct address format is {::[*ShortcutName*]%MD*n.x*} , where n is the file number and x is the element number, as described above.

See Also

Registers Maps | Allen-Bradley Controllers with the RMC

6.10.6. Using GE PLCs with the RMC

The RMC75E and RMC150E can communicate with the GE FANUC RX3i, RX7i, and 90-30 PLCs via the Modbus/TCP Ethernet protocol. The RX3i requires the IC695ETM001 Ethernet module.

Example Programs

Delta offers a sample RX3i PLC program to help you quickly set up the communications between your PLC and the RMC. See the downloads section of Delta's website at https://deltamotion.com.

Using the GE Ethernet Communications

The GE Fanuc manual "TCP/IP Ethernet Communications for PACSystems", number GFK-2224D, provides very good detail on how to use the IC695ETM001 Ethernet module. The RX3i must be used as a Modbus/TCP client, which is described in Chapter 8 of the manual. Use the example PLC program on Delta's website as a starting point for your application.

See Also

Communications Overview

6.10.7. Using LabVIEW with RMCs

Delta provides instrument drivers for use with National Instruments LabVIEW™ software. These instrument drivers include full-fledged examples that are ready to run.

Installing the Instrument Drivers

Delta recommends installing the instrument drivers directly from the LabVIEW™ software:

1. In LabVIEW, on the **Tools** menu, choose **Instrumentation** and click **Find Instrument Drivers**.
2. Choose **Delta Computer Systems** and click **Search**.
3. Choose the driver for your RMC and your RMC LabVIEW version, and click **Install**. Wait for the installation to complete.

After completing the installation, the RMC instrument drivers will be available on the Instrument Driver pallete. You can find examples by searching for "RMC" in the LabVIEW Example Finder.

See Also

Help Overview

6.10.8. Using Mitsubishi PLCs with the RMC

Q-Series

The RMC can communicate with the Mitsubishi Q-series PLC in the following ways:

- **Q and L-Series Built-in Ethernet:**
  Delta has created function blocks for communicating with the RMC via the built-in Ethernet port of Q and L-series PLCs. See https://deltamotion.com/company/alliances/alliance_mitsubishi.php for details.
- **QJ71E71-100 module (Ethernet)**
  Using the Procedure Exist control method. See the Mitsubishi Procedure Exist topic for details. A sample PLC project is available on the downloads page of Delta's website.
- **QJ71MT91 module (Ethernet Modbus/TCP)**.
  Using the Modbus/TCP protocol. A sample PLC project is available on the downloads page of Delta's website. The download includes a document explaining how to set up and use the QJ71MT91.
- **QJ71C24N module (Serial RS-232/485)**
  Supported by the RMC75S only, using the Bidirectional protocol. See the Mitsubishi Bidirectional Protocol topic for details. A sample PLC project is available on the downloads page of Delta's website.

FX3U

The RMC can communicate with the Mitsubishi FX3U PLC in the following ways:

- **Ethernet (RMC75E or RMC150E):**
  Using the Procedure Exist control method. See the Mitsubishi Procedure Exist topic for details.

See Also

Communications Overview

6.10.9. Using Schneider Electric (Modicon) PLCs via Modbus/TCP

The RMC75E, RMC150E, and RMC200 can communicate with any Schneider Electric Modicon PLCs that support Modbus/TCP (Ethernet) or Modbus RTU (serial RS-232). The RMCs *cannot* communicate via Modbus Plus.

The RMCs can also communicate with Schneider Electric Modicon PLCs via EtherNet/IP I/O.

See Also

Modbus/TCP | Modbus RTU | Using Schneider Electric PLCs via EtherNet/IP I/O

6.10.10. Using Schneider Electric PLCs via EtherNet/IP I/O

Several Schneider Electric PLCs support EtherNet/IP I/O communication via plug-in Ethernet communication modules. This topic describes how to use Quantum 140 NOC 771 01, Premium TSX ETC 101, and Modicon M340 BMX NOC 0401 Ethernet communication modules to communicate with the RMC via EtherNet/IP I/O.

Determine I/O Data Locations in the RMC

EtherNet/IP I/O transfers data back and forth between the RMC and PLC at the Requested Packet Interval (RPI). The user must specify which data items in the RMC should be sent and received. Typically, this is data in the Indirect Data Map.

Set up the Indirect Data map so that one part contains all the data coming from the PLC (Incoming Data), and another part contains all the data going to the PLC (Outgoing Data). Make sure the Incoming and Outgoing Data areas in the Indirect Data Map do not overlap.

The Outgoing Data typically includes RMC status items that the PLC always needs to keep track of, such as actual positions and status bits.

The Incoming Data consists of items that the PLC needs to write to in the RMC. This is typically variables and possibly command registers.

Note:
The Incoming and Outgoing Data locations need not be the Indirect Data Map. However, the Indirect Data Map is usually the best choice. Other options are the Variable Table and the command area.

Setting Up the RMC for EtherNet/IP I/O

Do the following in the RMC:

1. **Set the RMC's IP Address**
   Set up the RMC's IP Address as for any Ethernet connection. See Setting Up the RMC Ethernet for details.
2. **Set Up the Indirect Data Map**

   In the Project pane, double-click **Address Maps**, then click **Indirect Data Map**.

   Beginning at item 0 in the Indirect Data Map, choose the items for the Outgoing Cyclic I/O Data.

At some location in the Indirect Data Map after the Outgoing Cyclic I/O Data area, choose the items for the Incoming Cyclic I/O Data, that is, the items that will be sent to the RMC from the PLC. If you are using another location for your Incoming Data, such as the Variable Table or Command Area, you need not set up the Indirect Data Map for the Incoming Data.

Example

If you have a 4-axis controller, you may wish to set up the Outgoing Data at the beginning of the Indirect Data Map to include the Actual Position and the Status Bits for each axis, in addition to information on Task 0, which perhaps runs your user programs. You may also wish to set the Incoming Data, starting at item 12 in the Indirect Data Map, to go to 5 variables and some of the Axis 0 command registers. You could then set up the Indirect Data Map like this:

	Reg #	Map To	Description	Current
0	%MD42.0	%MD8.0	Axis0 Status Bits	N/A
1	%MD42.1	%MD9.0	Axis1 Status Bits	N/A
2	%MD42.2	%MD10.0	Axis2 Status Bits	N/A
3	%MD42.3	%MD11.0	Axis3 Status Bits	N/A
4	%MD42.4	%MD8.8	Axis0 Actual Position (pu)	N/A
5	%MD42.5	%MD9.8	Axis1 Actual Position (pu)	N/A
6	%MD42.6	%MD10.8	Axis2 Actual Position (pu)	N/A
7	%MD42.7	%MD11.8	Axis3 Actual Position (pu)	N/A
8	%MD42.8	%MD48.0	Task 0 Status	N/A
9	%MD42.9	%MD48.3	Task 0 Current Program	N/A
10	%MD42.10			
11	%MD42.11			
12	%MD42.12	%MD56.0	0 - (Var1)	N/A
13	%MD42.13	%MD56.1	1 - (Var2)	N/A
14	%MD42.14	%MD56.2	2 - (Var3)	N/A
15	%MD42.15	%MD56.3	3 - (Var4)	N/A
16	%MD42.16	%MD56.4	4 - (Var5)	N/A
17	%MD42.17	%MD40.0	Axis0 Command Area	N/A
18	%MD42.18	%MD40.1	Axis0 Command Parameter 1	N/A
19	%MD42.19	%MD40.2	Axis0 Command Parameter 2	N/A
20	%MD42.20	%MD40.3	Axis0 Command Parameter 3	N/A
21	%MD42.21	%MD40.4	Axis0 Command Parameter 4	N/A
22	%MD42.22	%MD40.5	Axis0 Command Parameter 5	N/A

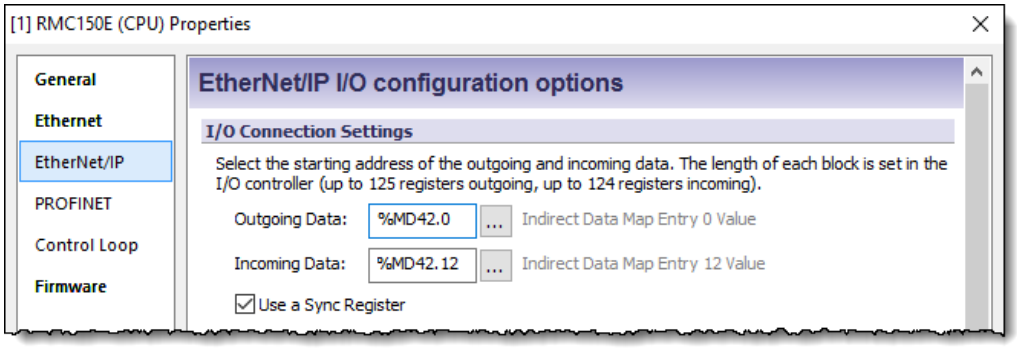
3. **Set the Cyclic I/O Data Locations in the RMC**
   In the Project pane, expand the **Modules** folder, double-click the CPU module, and choose **EtherNet/IP**.

   Under **Outgoing Data**, enter the starting location for the outgoing cyclic I/O data. In our example, verify that the location is the Indirect Data Map Entry 0 Value.

Under **Incoming Data**, enter the starting address for the incoming cyclic I/O data. This should be a location in the Indirect Data Map, the Variable Table, or Command Area as discussed in the **Determine I/O Data Locations in the RMC** section above.

Notice that the RMC200 supports up to three I/O connections. However, for Schneider Electric PLCs, it is recommended that only the first I/O connection is used, which supports up to 360 registers in each direction. The settings for I/O Connection 2 and I/O Connection 3 will be ignored.

For example, the EtherNet/IP Settings Page below shows an RMC150 with the Outgoing Data coming from the Indirect Data map starting at item 0 and the Incoming Data going to the Indirect Data starting at item 12.



4. **Choose Whether to Use a Sync Register**

   The Sync Register provides a method for the PLC to synchronize the Input Data and Output Data. If you will be writing to the Command Area directly or indirectly via the Indirect Data Map, Delta recommends using the Sync Register.

   With a Sync Register, the Incoming Data is not written to the RMC until the Sync Register change. If you prefer to have the Incoming Data be written whenever any value in the Incoming Data changes, choose the option to not use a Sync Register.

   For more details, see Using an EtherNet/IP I/O Connection.

Configuring the Unity PLC Connection

The following procedures describe setting up the EtherNet/IP I/O connection between a Schneider Electric PLC with an RMC using Unity Pro software. The corresponding Schneider Electric documentation for these procedures is the following:

- **Quantum 140 NOC 771 01 Ethernet Communication Module User Manual** (S1A33985.xx)
- **Premium TSX ETC 101 Ethernet Communication Module User Manual** (S1A34003.xx)
- **Modicon M340 BMX NOC 0401 Ethernet Communication Module User Manual** (S1A34009.xx)

These procedures were built using Unity Pro XL V13. They describe using the Quantum 140 NOC 771 01 Ethernet module. The procedures for other versions of Unity Pro and other EtherNet/IP-capable Schneider Electric Ethernet modules are expected to be similar.

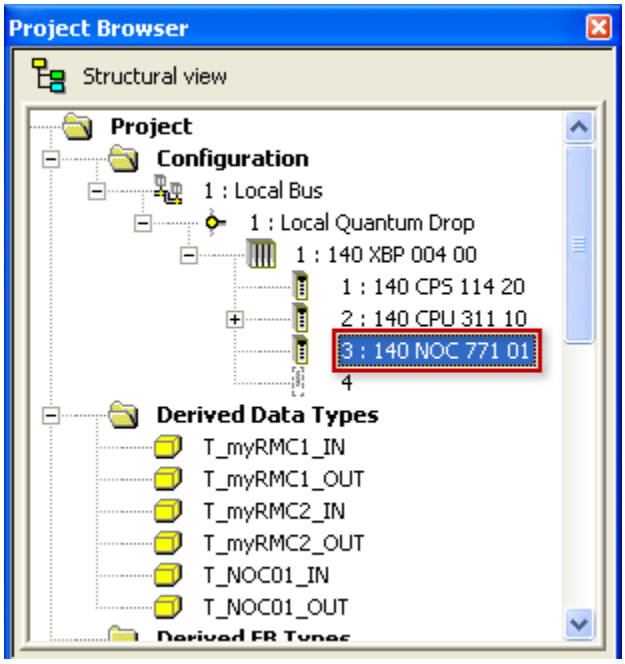
1. Start Unity Pro and ensure the following:
   - Ensure that the configuration includes the Schneider Electric Ethernet Communication Module. In this example, this module is a 140 NOC 771 01 with the Alias Name of "NOC01".
   - Ensure that the IP settings in the Ethernet module are configured and compatible with your network.

- Ensure that you are able to go on- and offline with the PLC and to connect and disconnect with the Ethernet module in the DTM Browser.
- Ensure that Unity Pro is not in Simulation Mode.

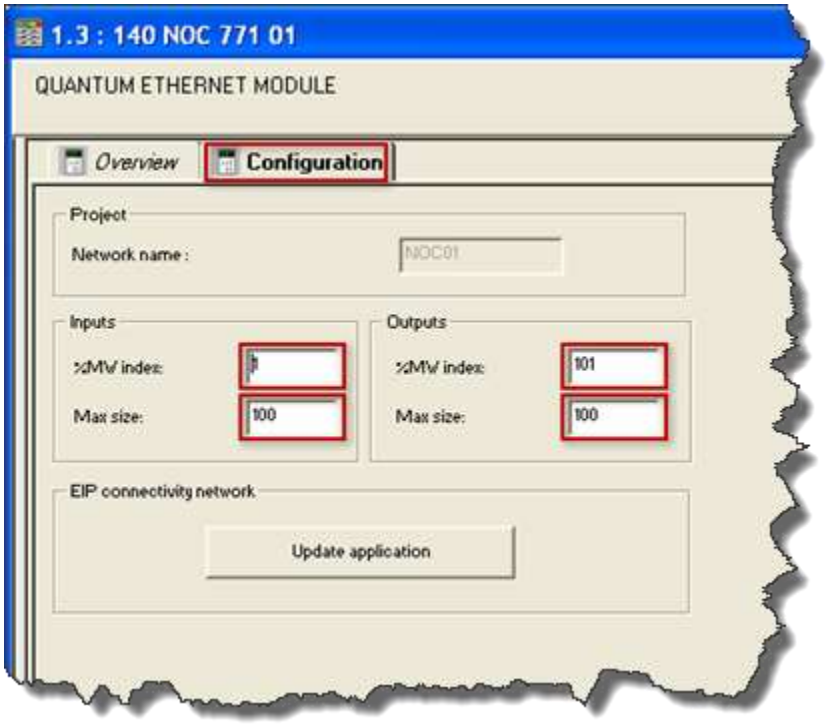
2. In the main Unity Pro window, ensure that the PLC is disconnected. If you are connected, on the **PLC** menu, click **Disconnect**.
3. Reserve a memory area for the Ethernet module that includes room for all the slave devices you will attach to the module.

2.

a. In the **Project Browser**, expand the **Configuration** node and double-click on the Ethernet communication module, which is a **140 NOC 771 01** in this example:



b. In the property window that opens, select the **Configuration** tab, and review the **Inputs and Outputs** sections:



These sections define two blocks of %MW registers, one for inputs into the PLC and one for outputs from the PLC. Refer to the manual for your particular Ethernet module to check how much data is required for the module itself, and then add the data that will be required by the slave devices.

In our example, we will need 11 incoming registers (produced by the RMC) and 12 outgoing registers (consumed by the RMC). Because the RMC uses 32-bit registers and the %MW registers are 16-bit, we need to double each. The revision of the 140 NOC 711 01 Ethernet module used in this example requires 16 %MW registers each for inputs and outputs in addition to the data transmitted with slave devices. Therefore the minimum number of inputs are 38 (11x2 + 16) and the minimum number of outputs are 40 (12x2 + 16)

In this example, we set both sizes to 100 to allow room for adding future devices.

c. To save these new address and size settings, on the **Edit** menu, click **Validate**.

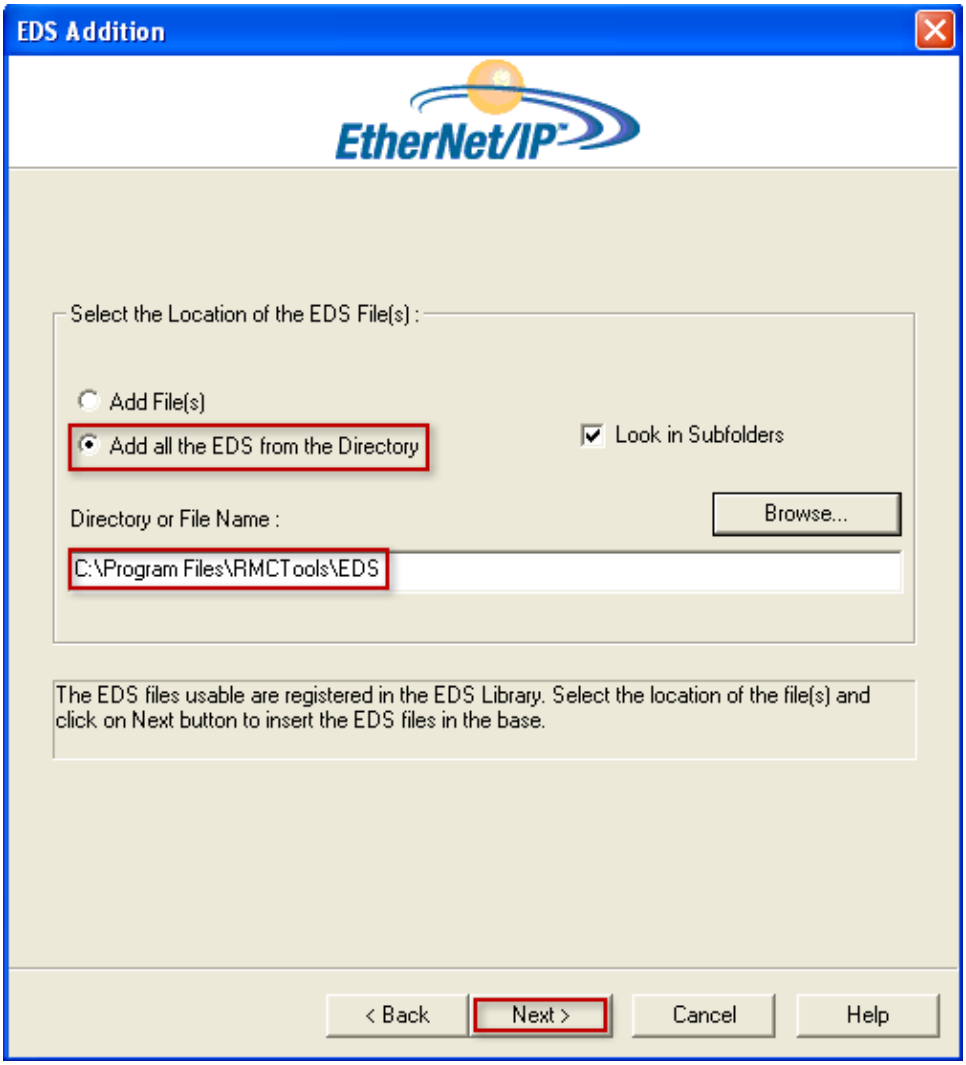
3.

a.

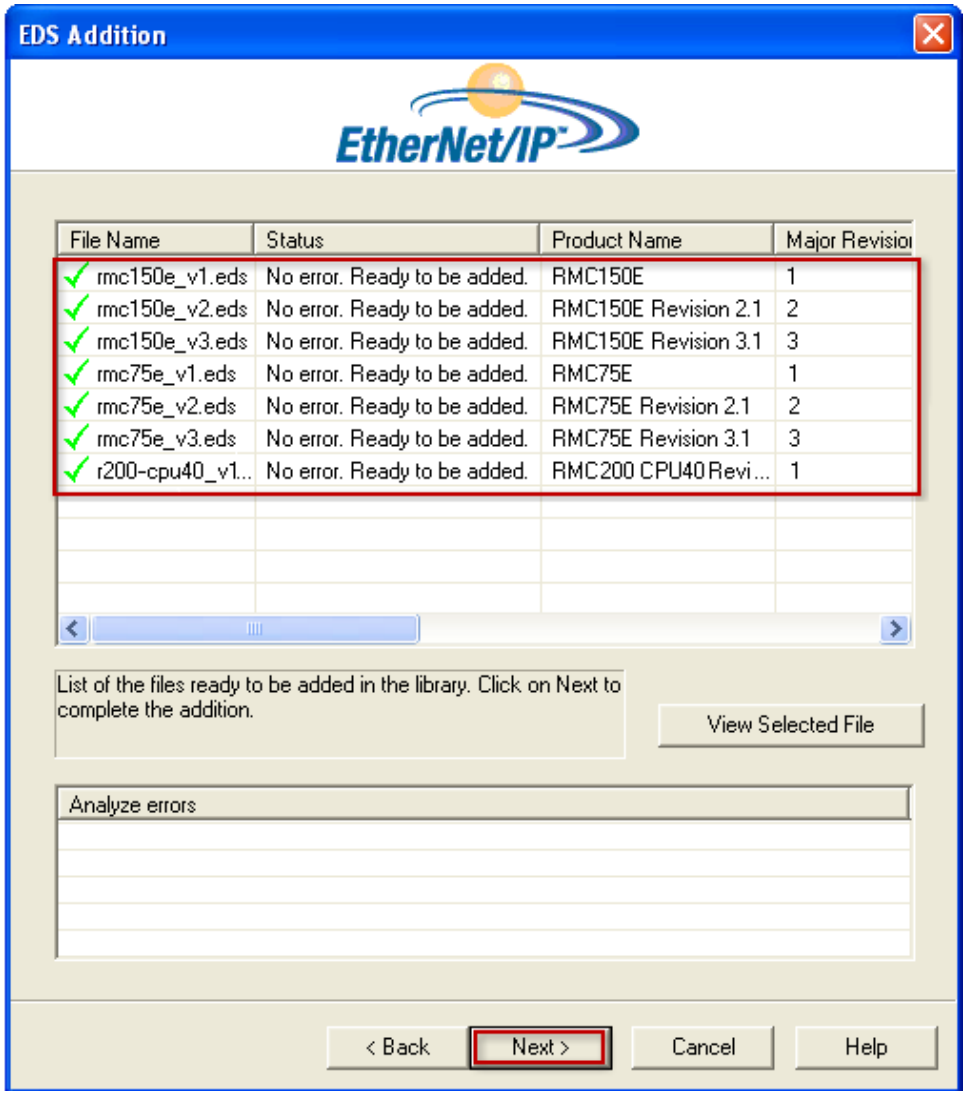
b.

c.

d. Complete the **EDS Addition** wizard that follows. When prompted for the location of the EDS files, select the contents of the EDS folder under the RMCTools program folder as shown below:



The wizard should confirm that the RMC EDS files will be added, similar to that shown below:



Complete the wizard using the Next and Finish buttons. At this point, the EDS files have been added to the EDS database, but are not yet added to the Hardware Catalog.

e.

f.

g.

h.

i.

4.

a.

b.

c.

d.

b.
c.

d.

e.

f.

5.

a.
b.
c.

6.

a.
b.

6.

a.

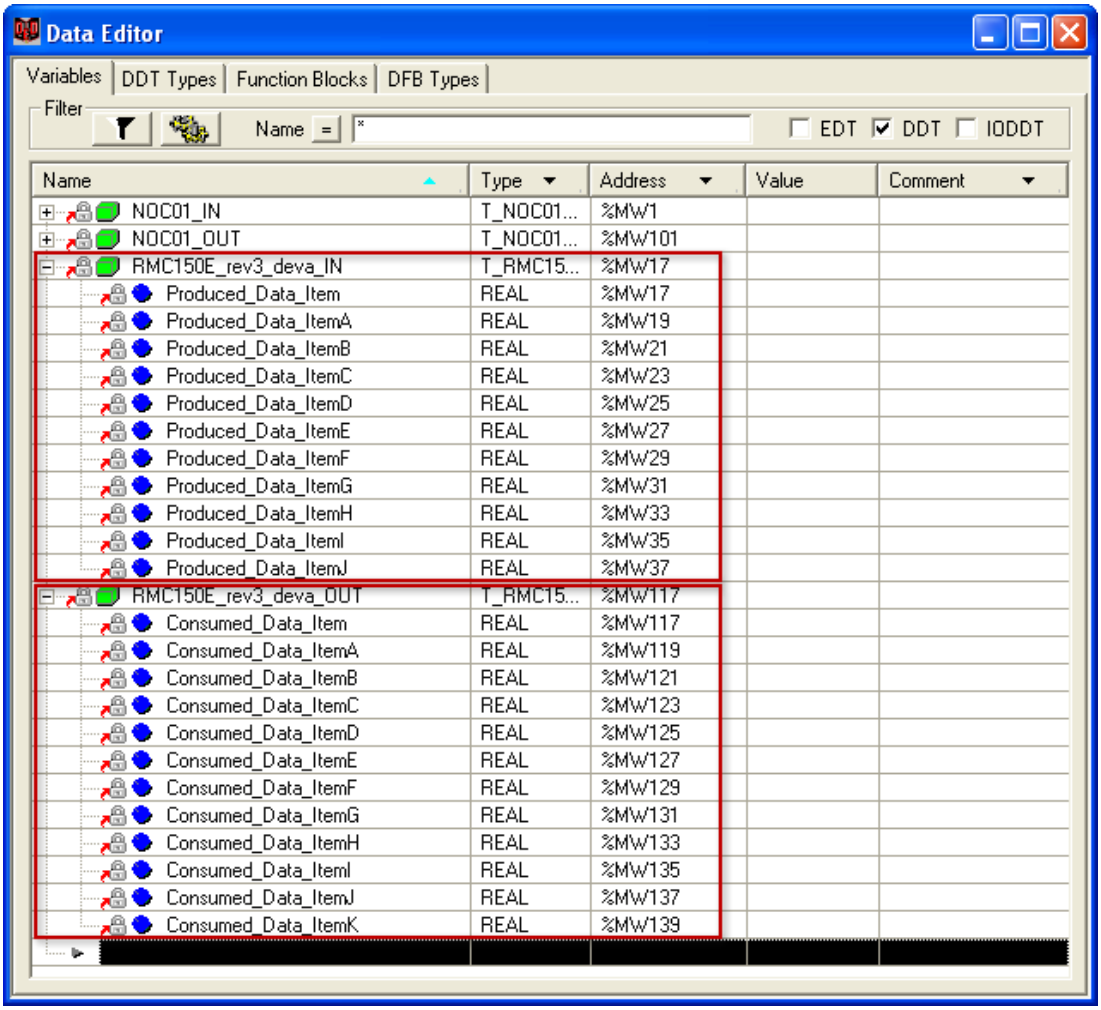
- **RPI:** Select the desired update rate. A commonly-used RPI is 20.0 ms. Very low RPIs may flood the network and reduce network reliability.
- **Input size:** This is the size of the data being produced by the RMC. In this example, we'll send 10 registers from the Indirect Data Map, plus one Sync register. Each RMC register is 32 bits, or 4 bytes, for a total of 44 bytes.
- **Input mode:** Most EtherNet/IP applications do not share the data produced by the RMC and, therefore, the Input mode should be set to Point to Point to reduce network traffic, but this can be set to Multicast if required.
- **Output size:** This is the size of the data being consumed by the RMC. In this example, we'll consume 11 registers by the Indirect Data Map, plus one Sync register. Each RMC register is 32 bits, or 4 bytes, for a total of 48 bytes.

Click **OK** when done.

b.

7. Add additional RMC devices to your EtherNet/IP network.
   a.
8. Download the configuration to the PLC.
   a. On the **PLC** menu, click **Connect**.
   b. On the **PLC** menu, click **Transfer Project to PLC**.

The above steps will register EtherNet/IP I/O connections for each RMC with the Schneider Electric Ethernet communication module and create derived variables in the Schneider Electric PLC representing the data for each RMC. The following screen shot shows the derived variables created for the example device in the steps above:



The RMC150E_rev3_deva_IN structure represents data produced by the RMC, starting with the Sync Register followed by the Outgoing Cyclic I/O Data in the RMC, which in this example is %MD42.0-9 from the Indirect Data Map. The RMC150E_rev3_deva_OUT structure represents data consumed by the RMC, starting with the Sync Register followed by the Incoming Cyclic I/O Data in the RMC, which in this example is %MD42.12-22 in the Indirect Data Map.

Performing Communications

Once the EtherNet/IP connection is configured and applied to the PLC, the communications will automatically start, assuming the PLC and RMC are both on a properly set up Ethernet network. Notice that the Omron will communicate even when it is in Program mode.

Reading Data from the RMC

This data will automatically update each Requested packet interval, and you can use the data as you wish. You cannot write to the Input Data. The Input Data will contain the Indirect Data from the RMC. However, if you selected to use the Sync Register, the first item in the Input Data array will be the SyncIn value, followed by the Indirect Data from the RMC.

The SyncIn is used only for synchronizing commands with the logic, as explained below. The SyncIn and SyncOut registers are only visible in the PLC. They are not visible in RMCTools.

Writing to the RMC - General

If you selected to *not* use a Sync Register, the Output Data is written to the RMC when any value in the Output Data changes.

If you selected to *use* a Sync Register, the Output Data is sent to the RMC at each Requested Packet Interval, but the RMC ignores it until the SyncOut register changes. The first item in the Output Data array is the SyncOut register, followed by the registers that will be sent to the Incoming Data location in the RMC. Use the following procedure to write to the RMC with a Sync Register:

1. **Wait Until the Sync In and Sync Out Registers Match**
   If they do not match, then this means that another write is in progress.
2. **Write to the Output Data**
   Write the desired to the Output Data array.
3. **Change the Sync Out Register**
   The easiest way to do this is to add one to it. However, you must take care to handle overflowing this register (the Sync register is a REAL). One method is to add one and then MOD it with some large number, such as 10000. This will make the register count from 0 to 9,999, and then wrap back down to 0 without an error.
4. **Wait Until the Sync In and Sync Out Registers Match**
   This indicates that the RMC has received the data and processed it.

See Using an EtherNet/IP I/O Connection for further details.

Sending Commands to the RMC

If you are writing to the Command Area, either directly or via the Indirect Data Map, Delta recommends using a Sync Register and following the procedure below to send commands to the RMC. The Output Data is sent to the RMC each RPI, but the RMC ignores it until the SyncOut register changes. The first item in the Output Data array is the SyncOut register, followed by the registers that will be sent to the Incoming Cyclic I/O Data location in the RMC.

1. **Wait Until the Sync In and Sync Out Registers Match**
   If they do not match, then this means that another write is in progress.
2. **Clear Old Commands from the Command Registers**
   Clear old commands from the command registers in the Output Data. Otherwise, when the Sync Out register is changed, the commands would be re-issued. One method of clearing the old commands is to fill the Output Data array with zeroes (except the SyncOut value).
3. **Write to the Command Registers**
   Write the Command registers and all required command parameters to the Output Data for all commands you want to issue. You can issue up to one command per axis. Leave the Command register set to 0 for each axis that will not receive a command.

   For example, if a portion of the Output Data is going to the Command Area for Axis 0, and you wish to issue a Move Absolute Command (20) to Axis 0 with a position of 6.7, a Speed of 3, and Accel and Decel of 50, you would write the following:

Value
20 (for a move Absolute Command)
6.7 (Position)
3 (Speed)
50 (Accel)
50 (Decel)
0 (Direction)

   Use the online help for each command to find out how many parameters a command has and what they mean. Make sure to write to all the parameters that the command uses. You do not need to write to command parameters that are not used by the command.
4. **Change the Sync Out Register**
   The easiest way to do this is to add one to it. However, you must take care to handle overflowing this register (the Sync register is a REAL). One method is to add one and then MOD it with some large number, such as 10000. This will make the register count from 0 to 9,999, and then wrap back down to 0 without an error. Take care to ensure that you only update the Sync Out Register once so that the commands do not get re-issued.

5. **Wait Until the Sync In and Sync Out Registers Match**
   This indicates that the RMC has received the command and issued it. It is important to wait until the SyncIn and SyncOut match before using the status bits in the Input Data (if the Input Data includes any status bits). See the Using an EtherNet/IP I/O Connection topic for how problems can occur if this step is ignored.

See Using an EtherNet/IP I/O Connection for further details.

Reading and Writing other Registers

To read and write other registers in the RMC that are not included in the Incoming or Outgoing Cyclic I/O Data, you can use the MSTR blocks to read/write registers in the RMC either using Modbus/TCP or the Register Map Object with EtherNet/IP Explicit Messaging. EtherNet/IP I/O and MSTR blocks can be used simultaneously.

Another option for writing to other RMC registers is to create user programs that move data from the variable table to other RMC registers.

See Also

EtherNet/IP Overview | Using Schneider Electric PLCs via Modbus

6.10.11. Using Omron Controllers via the FINS Protocol

This topic describes communication via the FINS protocol. For EtherNet/IP, see Using Omron Controllers via EtherNet/IP I/O.

The RMC75E, RMC150E, and RMC200 can communicate with Omron CS/CJ/CP series PLCs via the FINS Ethernet protocol. This requires an Omron module such as the CS/CJ ETN21 module. When communicating with the RMC from an Omron PLC, use the RMC's FINS Addresses for addressing the RMC's registers.

Example Programs

Delta provides example PLC programs to help you quickly set up the communications between your PLC and the RMC. See the downloads section of Delta's website at https://deltamotion.com.

Procedure for Using the ETN21

Omron's "Ethernet Units Construction of Networks" manual (catalog number W420-E1-01) describes how to set up and use the ETN21. It provides very good detail, but is very large. The information below describes the steps necessary for using the ETN21 with the RMC.

1. **Set the ETN21 IP Address**
   See section 2-7 in the "Ethernet Units Construction of Networks" manual.
2. **Pair the FINS node address to the RMC75E IP Address**
   See section 5 in the "Ethernet Units Construction of Networks" manual. The Automatic Generation Method is the easiest.
3. **Using the SEND and RECV Instructions**
   The SEND and RECV instructions are described in detail in Omron's "INSTRUCTIONS REFERENCE MANUAL" (catalog nr W340-E1-13). You will also need to refer to the "Ethernet Units Construction of Networks" manual (catalog number W420-E1-04), section 6-6-3 for information on how to use these commands for Ethernet.

Some notes about setting up the control word for the SEND and RECV instructions for the RMC75E are given below:

Word	Bits 08 to 15	Bits 00 to 07
C	**Number of 16-bit words to read or write** (1 to 512). The PLC supports reads up to 990 words, but the RMC is limited to 512 words. You will need to read or write two 16-bit words for each 32-bit register in the RMC.	
C + 1	**Always 00 for RMCs.**	**Remote Network Address.** This is typically 0. It appears that if you are going from network through a PLC, to a network, then it may need to be set to some other number.
C + 2	**Remote Node Number.** For RMCs, this value should be set to whichever Node Number will be mapped to the RMC's IP address. If you use the automatic method, then it will be the same as the last byte of the IP address. For example, if the RMC's IP address is 192.168.0.34, then this would be 34 (22 in hexadecimal).	**Remote Unit Number.** For RMCs, this value should be 0.
C + 3	**Port Number:** 00 to 07. The Port Number is used to allow simultaneous communications in the PLC. Use a different number for each communication that may be requested simultaneously.	**No. of Retries:** 00 to 0F (0 to 15). For RMC communications, this value should be between 2 and 5.
C + 4	**Timeout:** 0001 to FFFF (0.1 to 6553.5 seconds).The default setting of 0000 sets a monitoring time of 2 seconds. For RMC communications, this value should be set to 0001 or 0002 for 0.1 or 0.2 second timeouts.	

See Also

Communications Overview | FINS Addressing | Using Omron Controllers via EtherNet/IP I/O

6.10.12. Using Omron Controllers via EtherNet/IP I/O

Several Omron controllers support EtherNet/IP I/O communication. This topic describes how to configure and use the Omron CS1 and CJ2 PLCs to communicate with the RMC via EtherNet/IP I/O. For instructions on using the FINS protocol to communicate with an RMC from an Omron PLC, see the Using Omron Controllers via FINS topic.

The download section of Delta's website contains Omron programs that are already set up to communicate with an RMC via EtherNet/IP I/O. These programs can be used as a starting point for your application.

Determine Data Size and Number of I/O Connections

The first step is to determine the amount of data to be transferred and the number of I/O connections to set up between the PLC and the RMC:

1. If you have an RMC75E or RMC150E, you will use one (1) I/O connection that transfers data back and forth between the PLC. See the chart below for the maximum number of registers that can be transferred.
2. If you have an RMC200, the I/O connection supports 360 registers in each direction. This will typically be enough data for most applications. If you need more data, you will need to use multiple connections (maximum of 3), for as much data as you need, as shown in the chart below. Or, if your PLC is limited to 125/124 registers for the first connection, you may need to use multiple connections. See the chart below for the maximum number of registers that can

be transferred with each connection. Keep in mind that the connections are independent of each other, and the data will not be transferred at the same time, even when the RPI may be set to the same value.

I/O Connections and Max Data Size (input is to PLC, output is from PLC)

Connection	RMC75E	RMC150E	RMC200
Connection#1	125 input registers 124 output registers	125 input registers 124 output registers	360 input registers 360 output registers
Connection#2	n/a	n/a	125 input registers 124 output registers
Connection#3	n/a	n/a	125 input registers 124 output registers

Determine I/O Data Locations in the RMC

EtherNet/IP I/O transfers data back and forth between the RMC and PLC at the Requested Packet Interval (RPI). The user must specify which data items in the RMC should be sent and received. Typically, this is data in the Indirect Data Map.

Set up the Indirect Data map so that one part contains all the data coming from the PLC (Incoming Data), and another part contains all the data going to the PLC (Outgoing Data). Make sure the Incoming and Outgoing Data areas in the Indirect Data Map do not overlap.

The Outgoing Data typically includes RMC status items that the PLC always needs to keep track of, such as actual positions and status bits.

The Incoming Data consists of items that the PLC needs to write to in the RMC. This is typically variables and possibly command registers.

Note:
The Incoming and Outgoing Data locations need not be the Indirect Data Map. However, the Indirect Data Map is usually the best choice. Other options are the Variable Table and the command area.

Setting Up the RMC for EtherNet/IP I/O

Do the following in the RMC:

1. **Set the RMC's IP Address**
   Set up the RMC's IP Address as for any Ethernet connection. See Setting Up the RMC Ethernet for details.
2. **Set Up the Indirect Data Map**

   In the Project pane, double-click **Address Maps**, then click **Indirect Data Map**.

   Beginning at item 0 in the Indirect Data Map, choose the items for the Outgoing Cyclic I/O Data.

   At some location in the Indirect Data Map after the Outgoing Cyclic I/O Data area, choose the items for the Incoming Cyclic I/O Data, that is, the items that will be sent to the RMC from the PLC. If you are using another location for your Incoming Data, such as the Variable Table or Command Area, you need not set up the Indirect Data Map for the Incoming Data.

   If you are using multiple connections to the RMC, repeat this for each connection.

   Example

   If you have a 4-axis controller, you may wish to set up the Outgoing Data at the beginning of the Indirect Data Map to include the Actual Position and the Status Bits for each axis, in addition to information on Task 0, which perhaps runs your user programs. You may also wish to set the Incoming Data, starting at item 12 in the Indirect Data Map, to go to 5 variables and some of the Axis 0 command registers. You could then set up the Indirect Data Map like this:

	Reg #	Map To	Description	Current
0	%MD42.0	%MD8.0	Axis0 Status Bits	N/A
1	%MD42.1	%MD9.0	Axis1 Status Bits	N/A
2	%MD42.2	%MD10.0	Axis2 Status Bits	N/A
3	%MD42.3	%MD11.0	Axis3 Status Bits	N/A
4	%MD42.4	%MD8.8	Axis0 Actual Position (pu)	N/A
5	%MD42.5	%MD9.8	Axis1 Actual Position (pu)	N/A
6	%MD42.6	%MD10.8	Axis2 Actual Position (pu)	N/A
7	%MD42.7	%MD11.8	Axis3 Actual Position (pu)	N/A
8	%MD42.8	%MD48.0	Task 0 Status	N/A
9	%MD42.9	%MD48.3	Task 0 Current Program	N/A
10	%MD42.10			
11	%MD42.11			
12	%MD42.12	%MD56.0	Current Value - 0 - (NoName)	N/A
13	%MD42.13	%MD56.1	Current Value - 1 - (NoName)	N/A
14	%MD42.14	%MD56.2	Current Value - 2 - (NoName)	N/A
15	%MD42.15	%MD56.3	Current Value - 3 - (NoName)	N/A
16	%MD42.16	%MD56.4	Current Value - 4 - (NoName)	N/A
17	%MD42.17	%MD40.0	Axis0 Command Area	N/A
18	%MD42.18	%MD40.1	Axis0 Command Parameter 1	N/A
19	%MD42.19	%MD40.2	Axis0 Command Parameter 2	N/A
20	%MD42.20	%MD40.3	Axis0 Command Parameter 3	N/A
21	%MD42.21	%MD40.4	Axis0 Command Parameter 4	N/A
22	%MD42.22	%MD40.5	Axis0 Command Parameter 5	N/A
23	%MD42.23			

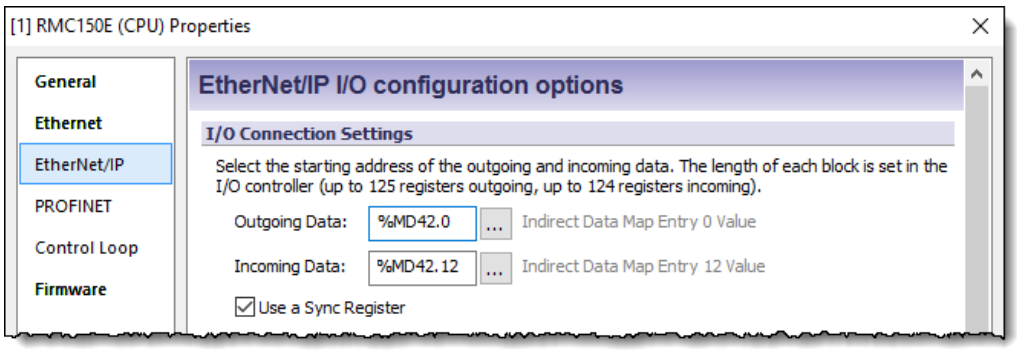
3. **Set the Cyclic I/O Data Locations in the RMC**
   In the Project pane, expand the **Modules** folder, double-click the CPU module, and choose **EtherNet/IP**.

   Under **Outgoing Data**, enter the starting location for the outgoing cyclic I/O data. In our example, verify that the location is the Indirect Data Map Entry 0 Value.

   Under **Incoming Data**, enter the starting address for the incoming cyclic I/O data. This should be a location in the Indirect Data Map, the Variable Table, or Command Area as discussed in the **Determine I/O Data Locations in the RMC** section above.

   If you are using multiple connections to the RMC, repeat this for each connection, otherwise, ignore the other connection settings.

   For example, the EtherNet/IP Settings Page below shows an RMC150 with the Outgoing Data coming from the Indirect Data map starting at item 0 and the Incoming Data going to the Indirect Data starting at item 12.



4. Choose Whether to Use a Sync Register

The Sync Register provides a method for the PLC to synchronize the Input Data and Output Data. If you will be writing to the Command Area directly or indirectly via the Indirect Data Map, Delta recommends using the Sync Register.

With a Sync Register, the Incoming Data is not written to the RMC until the Sync Register change. If you prefer to have the Incoming Data be written whenever any value in the Incoming Data changes, choose the option to not use a Sync Register.

For multiple connections, each connection has an individual Sync Register. Keep in mind that the connections are independent of each other, and the data may not be transferred at exactly the same time, even if the Sync Registers are changed simultaneously.

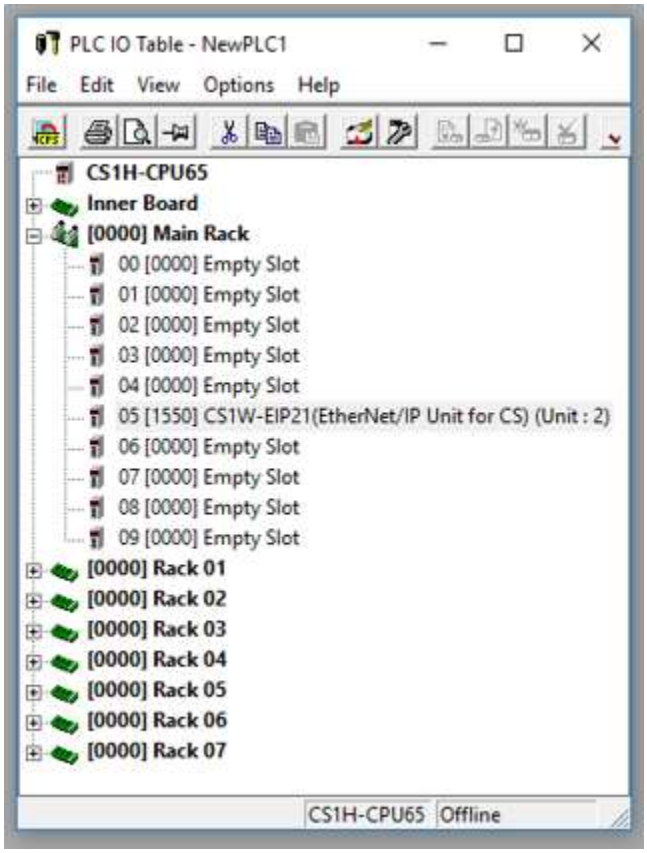
For more details, see Using an EtherNet/IP I/O Connection.

Configuring the Omron Connection

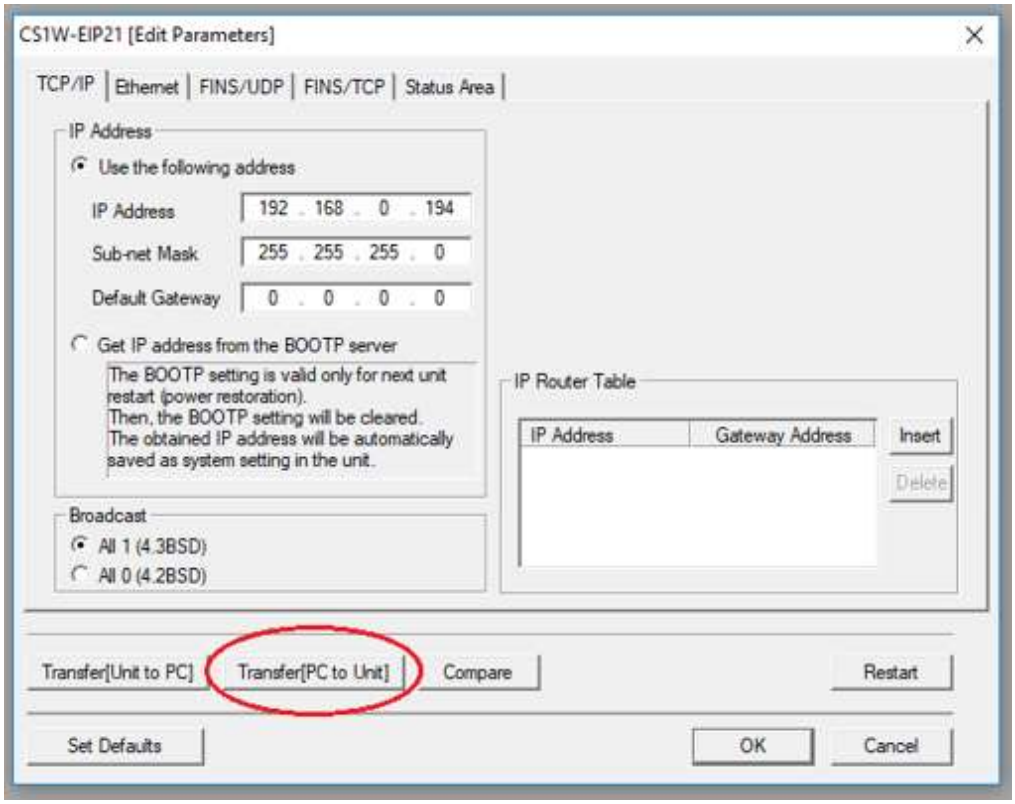
The following procedures describe setting up an EtherNet/IP I/O connection between an Omron communication adapter and an RMC using Omron's Network Configurator software. These procedures are based off of Network Configurator 3.20. The corresponding Omron documentation for these procedures is in section 6 (Tag Data Link Functions) in the **Omron EtherNet/IP Units Operational Manual (W465)** manual.

1. 1. In CX-Programmer, configure the the module with the EtherNet/IP port as follows:
   2. a. In the project view, double-click **IO Table and Unit Setup**.

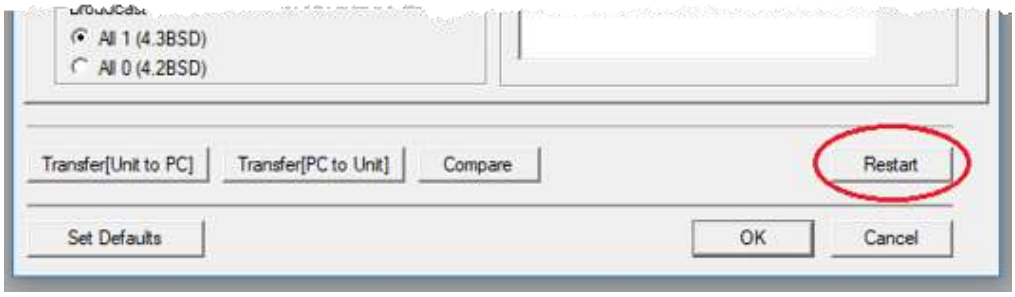
3. b. Expand **Main Rack** and double-click the **EIP21** module:



4. c. Set the **IP Address**, **Sub-net Mask**, and **Default Gateway**, then click **Transfer[PC to Unit]:**



5. d. Click **Restart** to restart the EIP21 module:

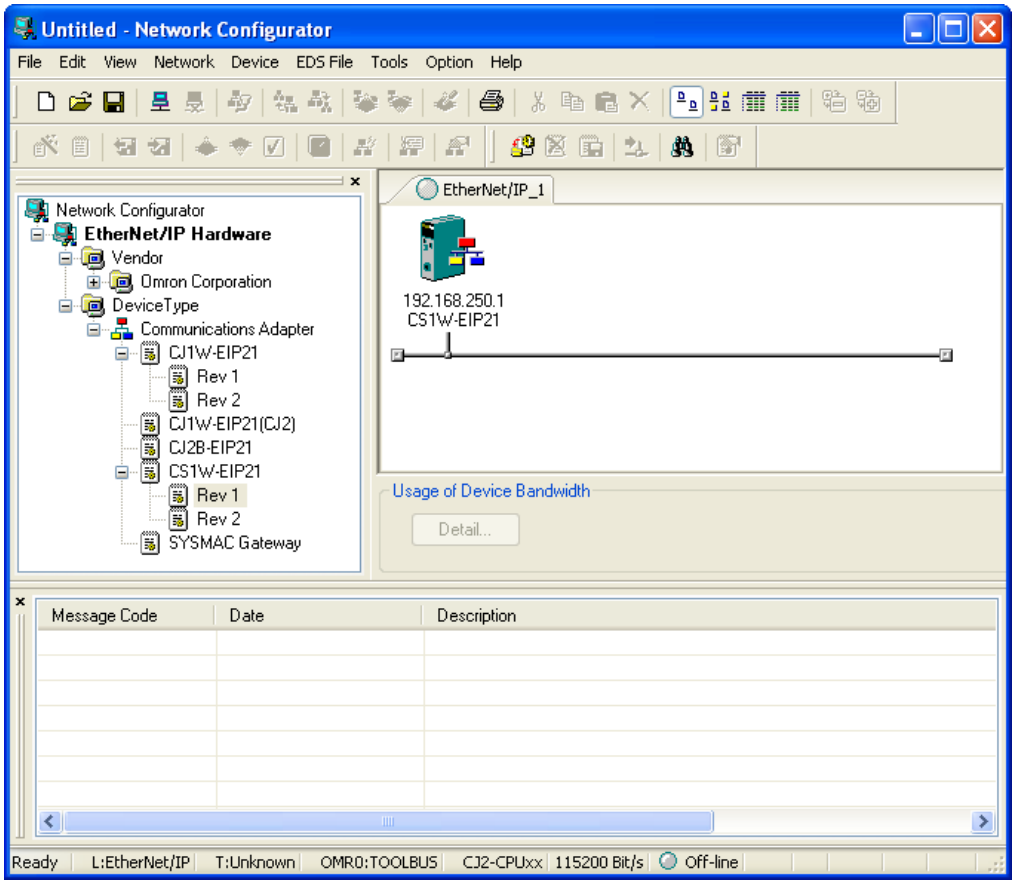


2. Start the Network Configurator.

3. If you have an existing EtherNet/IP network that you are adding the RMCs to, then do the following:
   a. Open your existing Network Configurator file.
   b. Skip to step 5.

4. If this is a new EtherNet/IP network, then do the following:
   a. From the list of EtherNet/IP Hardware on the left side, drag the appropriate communications adapter to the network line on the right side.

6.



b. Right-click on the new node in the diagram, and click **Change Node Address**.

c. Enter the new Communication Adapter's IP address to match the address set up in the PLC IO Table and click **OK.**

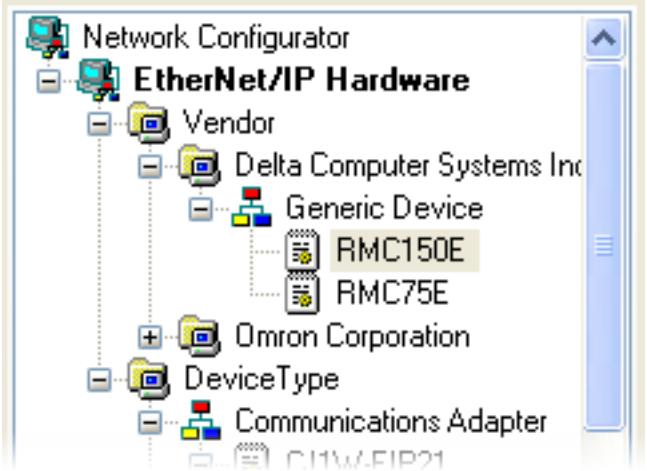
5. Install the RMC EDS files into the Network Configurator.

a. In Network Configurator, on the **EDS File** menu, click **Install**.

b. Browse to the EDS folder in the RMCTools installation location, which is typically **C:\Program Files\RMCTools\EDS**, and click **Open** to add the EDS file.

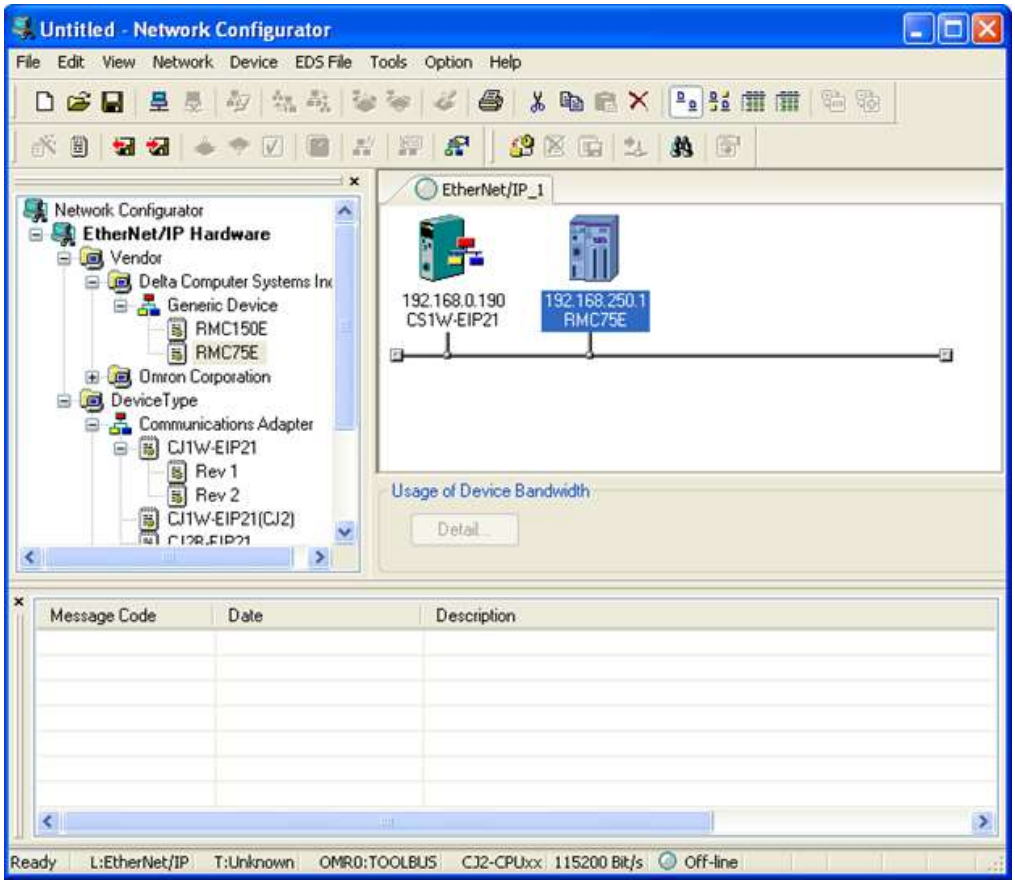
c. Repeat step b to install additional EDS files.

The EtherNet/IP Hardware list should now include the RMC devices you added:



7.
8.

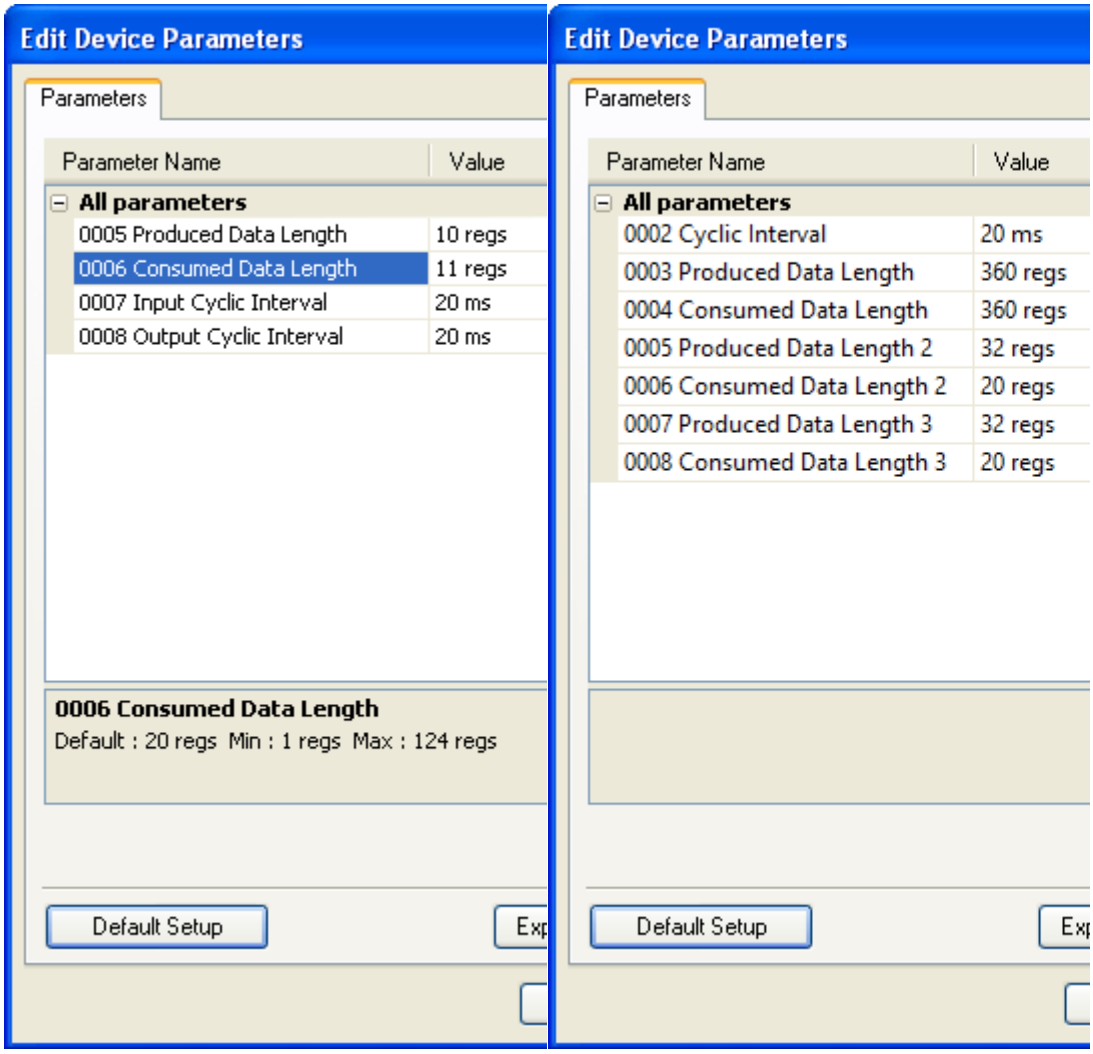
6. Add the RMC device to your EtherNet/IP network.
   a. From the list of EtherNet/IP Hardware on the left side, drag the appropriate RMC device to the network line on the right side.



   b. Right-click on the new node in the diagram, and click **Change Node Address**.
   c. Enter the RMC's actual IP address, as configured in RMCTools and click **OK**.

7. Edit the Device Parameters for the RMC.
   a. Double-click the RMC controller in the network diagram. This will open the **Edit Device Parameters** window.

RMC75 and RMC150: **RMC200:**



b. RMC75 and RMC150:

   i. For the **0005 Produced Data Length** parameter, enter the number of registers that you want the RMC to send cyclically to the PLC. If you chose to use the Sync Register, then this value must include the Sync Register.

   ii. For the **0006 Consumed Data Length** parameter, enter the number of registers that you want the PLC to send cyclically to the RMC. If you chose to use the Sync Register, then this value must include the Sync Register.

c. RMC200:

   i. For the **0003 Produced Data Length** parameter, enter the number of registers that you want the RMC to send cyclically to the PLC. If you chose to use the Sync Register, then this value must include the Sync Register.

   ii. For the **0004 Consumed Data Length** parameter, enter the number of registers that you want the PLC to send cyclically to the RMC. If you chose to use the Sync Register, then this value must include the Sync Register.

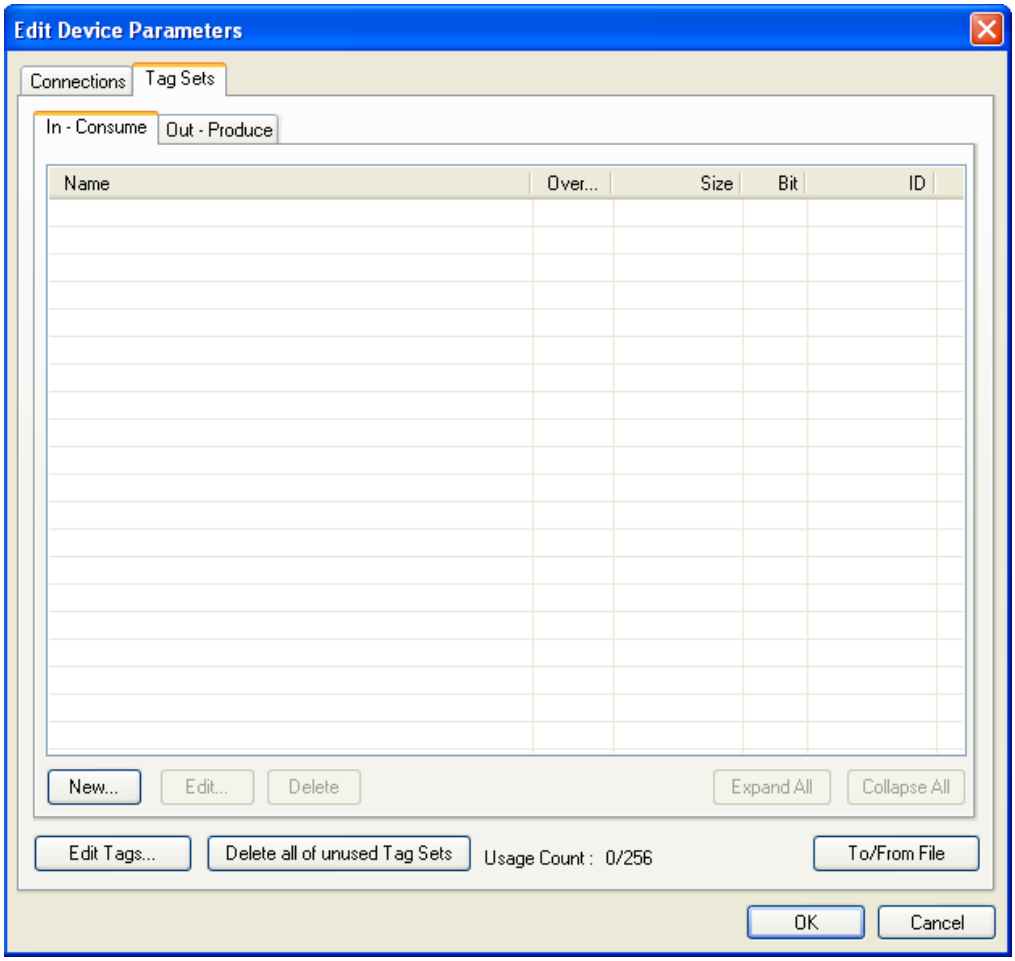
   iii. If you are using multiple connections, set the remaining items for the second and third connections, otherwise other items will be ignored.

d. Click **OK**. The cyclic intervals will be set later in the process.

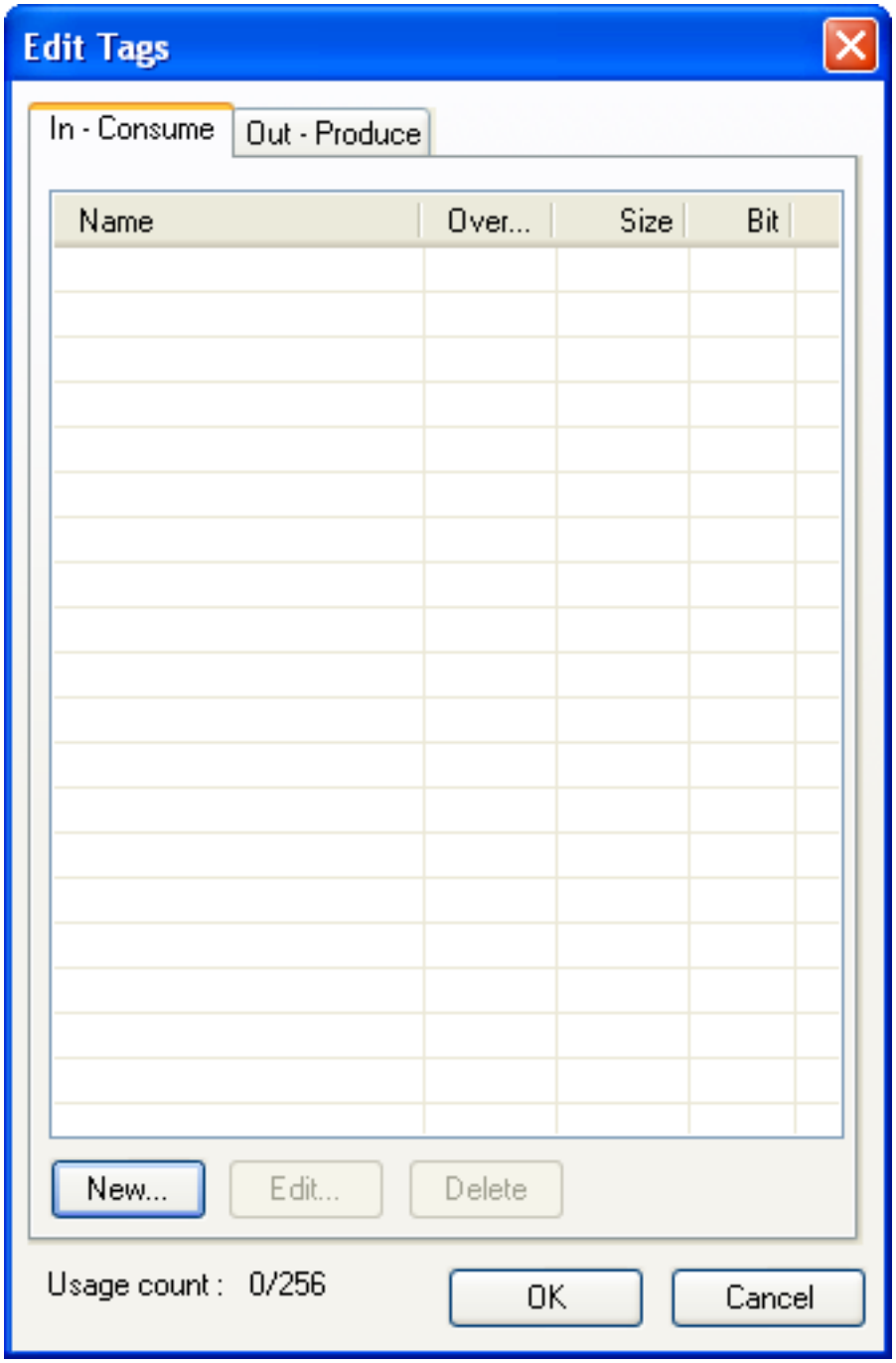
8. Create Tag Sets for the Input and Output Data

Before establishing an I/O connection, the location for the I/O data in the Omron PLC must be configured. The following procedure describes one way of doing this. However, be aware that the tags must be known about both by CX-Programmer and Network Configurator. **Section 6-2-4 Creating Tags and Tag Sets** of the **Omron EtherNet/IP Units Operational Manual (W465)** manual describes other methods of creating tag sets.

a. Double-click the Communication Adapter in the network diagram (CS1W-EIP21 in our example), and click the **Tag Sets** tab. This opens the **Edit Device Parameters** for the communication adapter to the **Tag Sets** tab:



b. Click **Edit Tags**. This opens the **Edit Tags** window:



c. Add a single tag to the **In – Consume** tab:

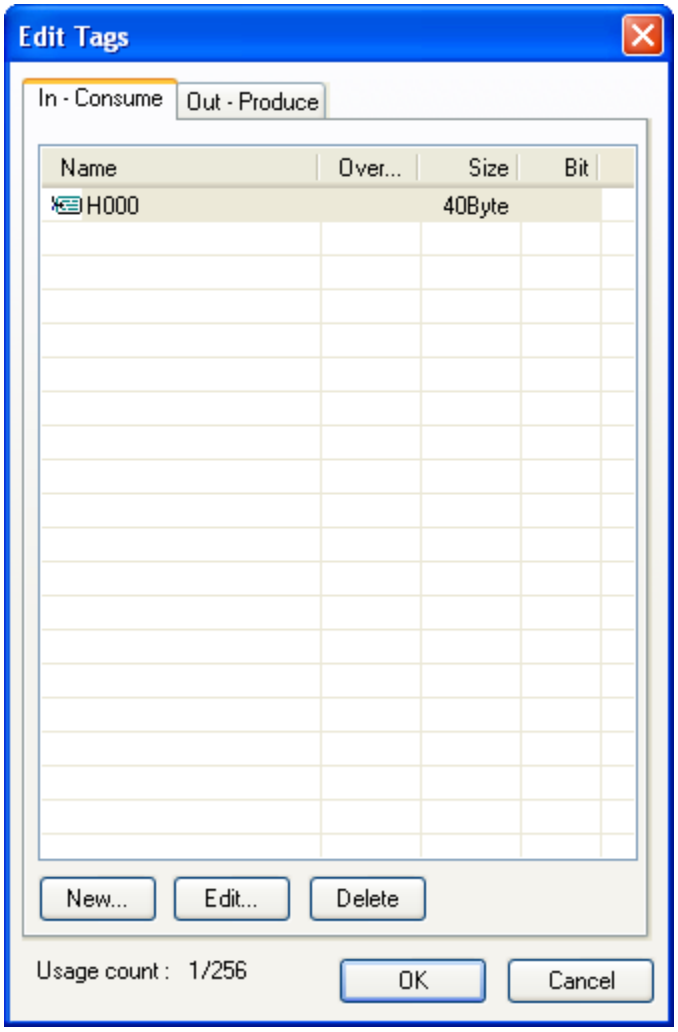
9. i. Click the **In – Consume** tab.

ii. At the bottom of the **In – Consume** tab, click **New.**

iii. In the **Edit Tag** dialog box, enter the Omron destination tag name in the **Name** box, and enter the length of the RMC's produced data in bytes in the **Size** box. This value must be four times the RMC's **0005 Produced Data Length** device parameter above. Notice that for Omron CS1 and CJ1 PLCs, the tag name must an Omron address. An address in the H memory range is recommended to allow direct bit access.

iv. Click **OK**. If this is the only connection, then click **Close** if prompted to add another tag. If you have multiple connections, choose to add another tag for each connection, following steps i-iv above.

The **In – Consume** tab should look like this:



d. Add a single tag to the **Out – Produce** tab:

i. Click the **Out – Produce** tab.

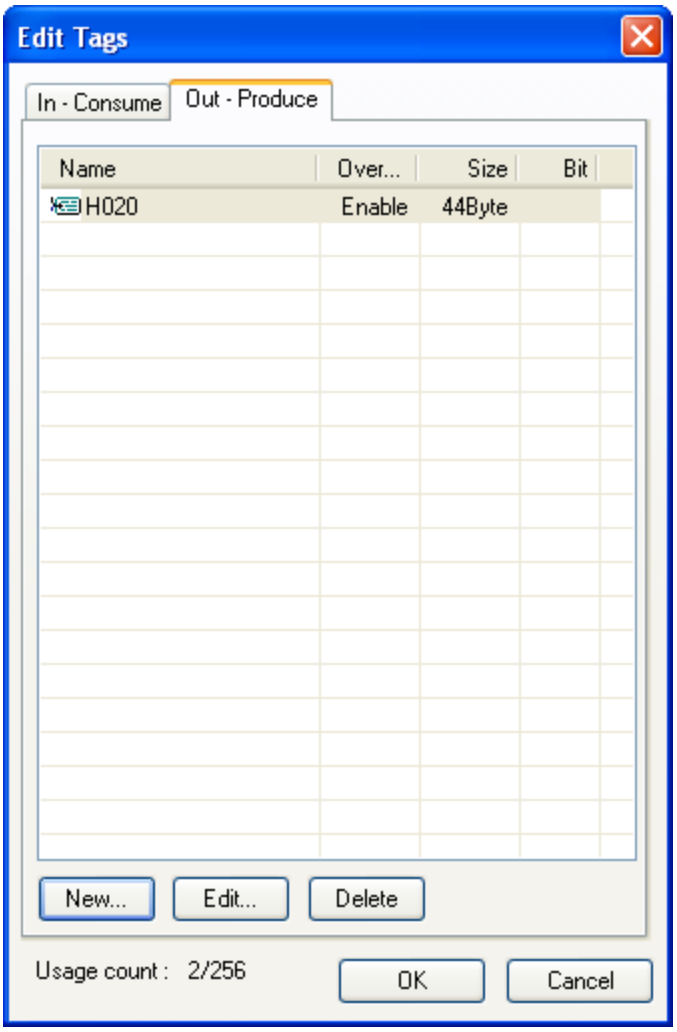
ii. At the bottom of the **Out – Produce** tab, click **New**.

iii. In the **Edit Tag** dialog box, enter the Omron source tag name in the **Name** box, and enter the length of the RMC's consumed data in bytes in the **Size** box. This value must be four times the RMC's **0006 Consumed Data Length** device parameter above. Notice that for Omron CS1 and CJ1 PLCs, the tag name must be an Omron address. An address in the H memory range is recommended to allow direct bit access.

iv. Click **OK**.

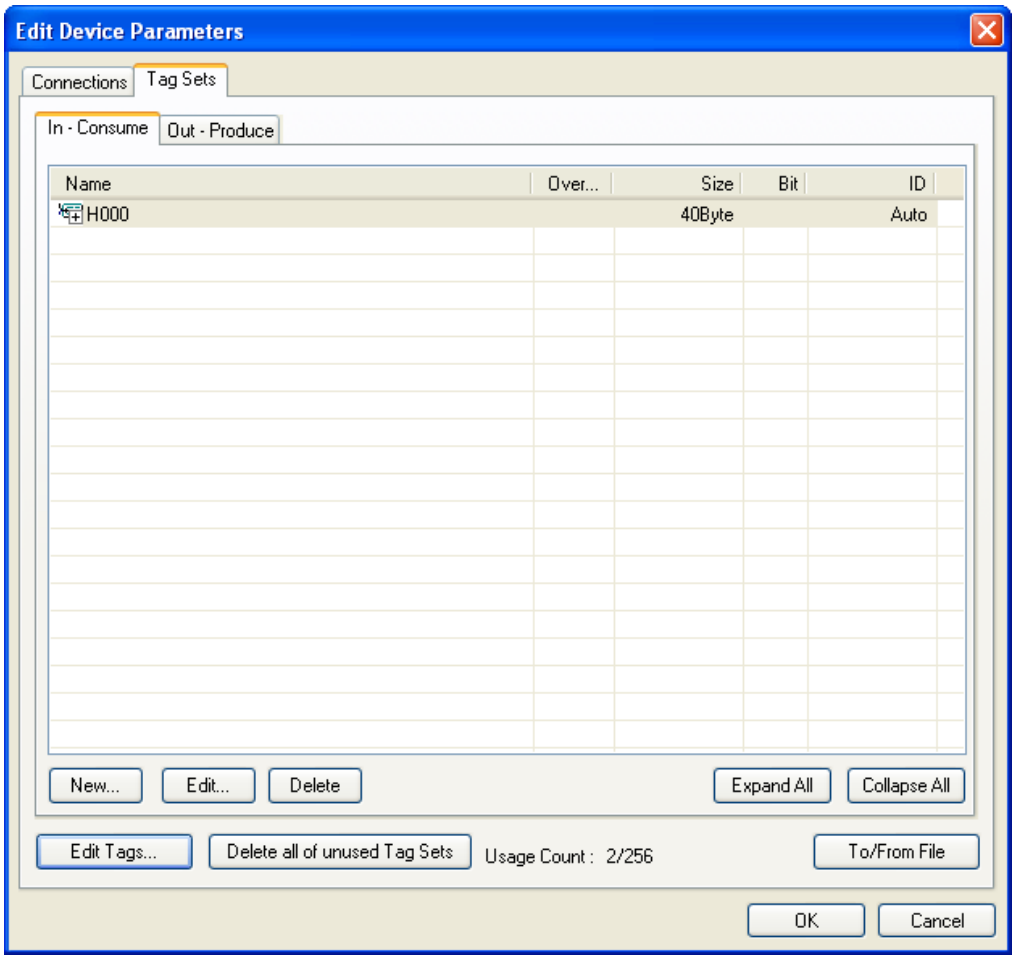
v. If this is the only connection, then click **Close** if prompted to add another tag. If you have multiple connections, choose to add another tag for each connection, following steps i-iv above.

The **Out – Produce** tab should now look like this:



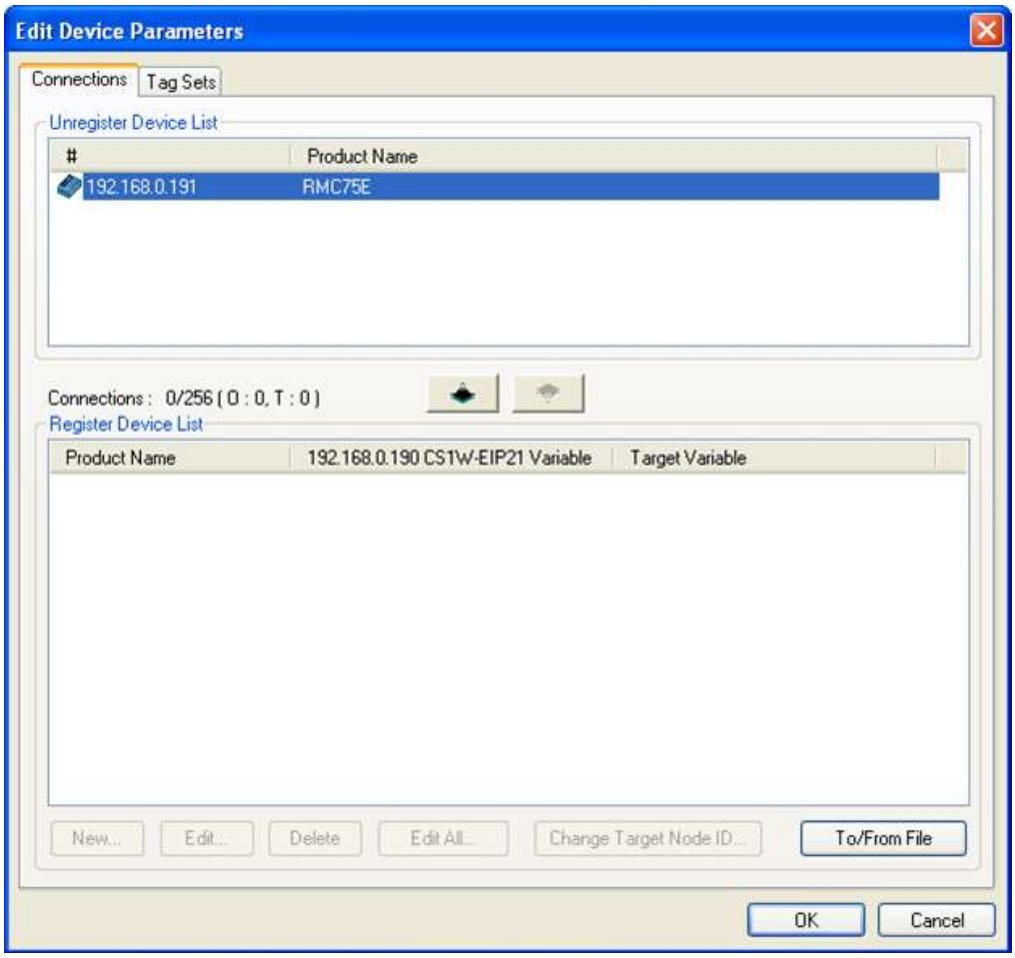
e. In the **Edit Tags**, window, click **OK**.

f. When prompted to register the new Tags as Tag sets, click **Yes**. This will add the new tags as tag sets:



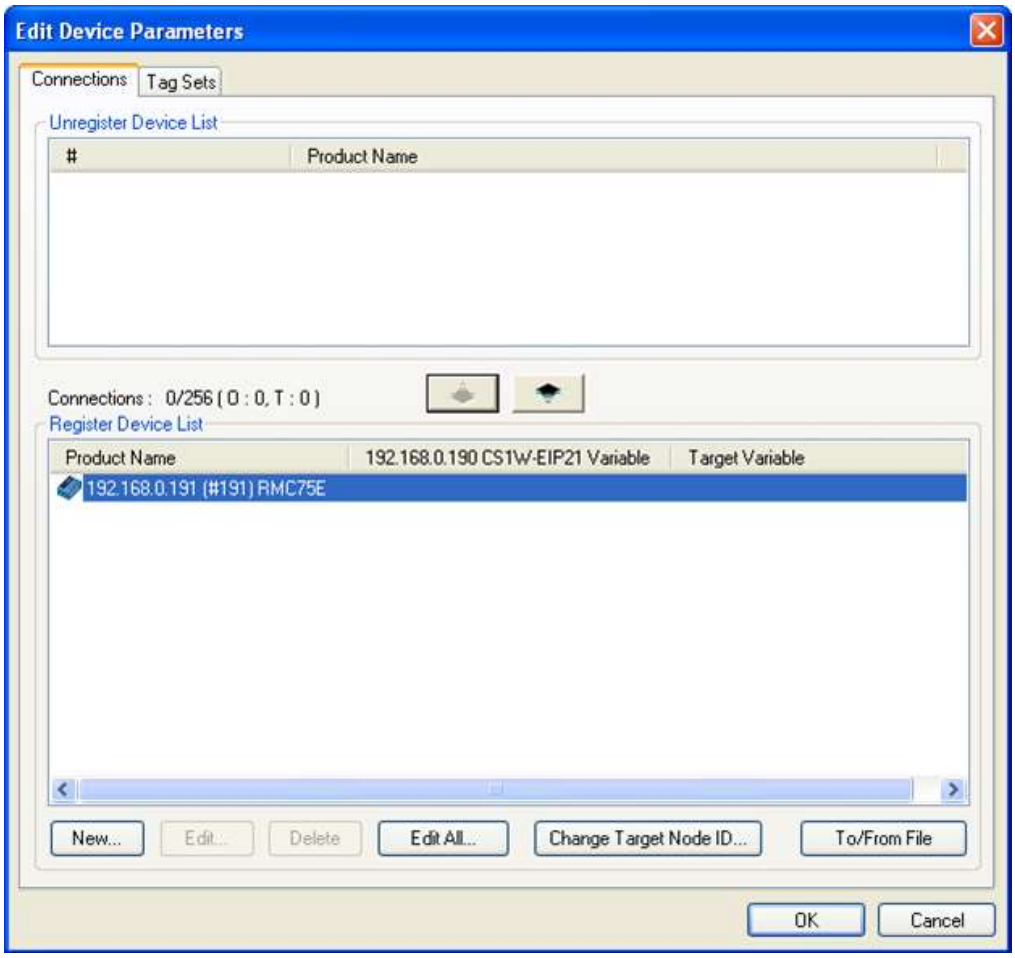
There are now two tag sets that have been registered. These tag sets will be used when creating the connection in the next step.

9. Create a connection to the RMC
   a. In the **Edit Device Parameters** window for the Communication Adapter (CS1W-EIP21 in this example), click the **Connections** tab.

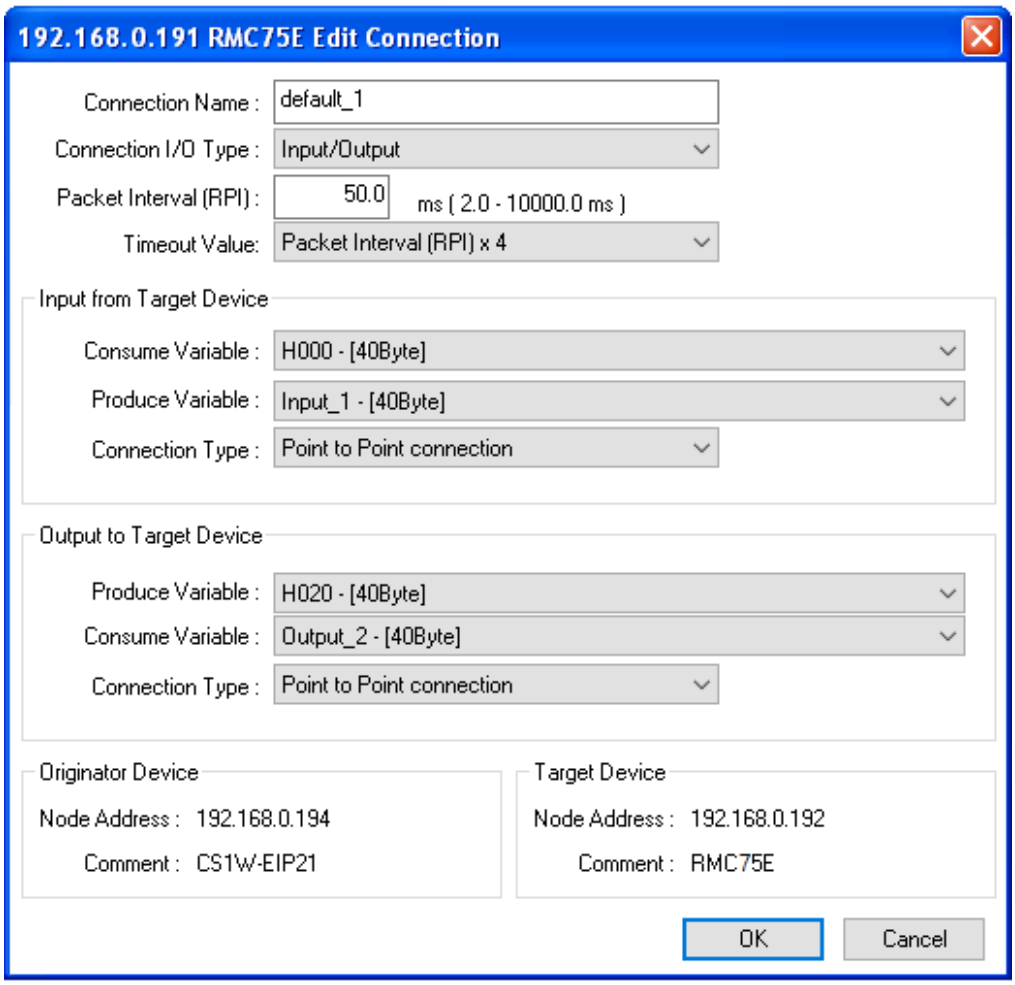


The RMC is listed in the **Unregister Device List**. In order to establish an I/O connection with this device, we must register the device, and then add a new connection.

b. Select the RMC in the **Unregister Device List**, and click the down arrow button  to register it. The RMC will now appear in the **Register Device List**:



c. Select the RMC in the **Register Device List**, and click **New**. This opens the Edit Connection window:



d. Fill in the fields in this dialog box as follows:

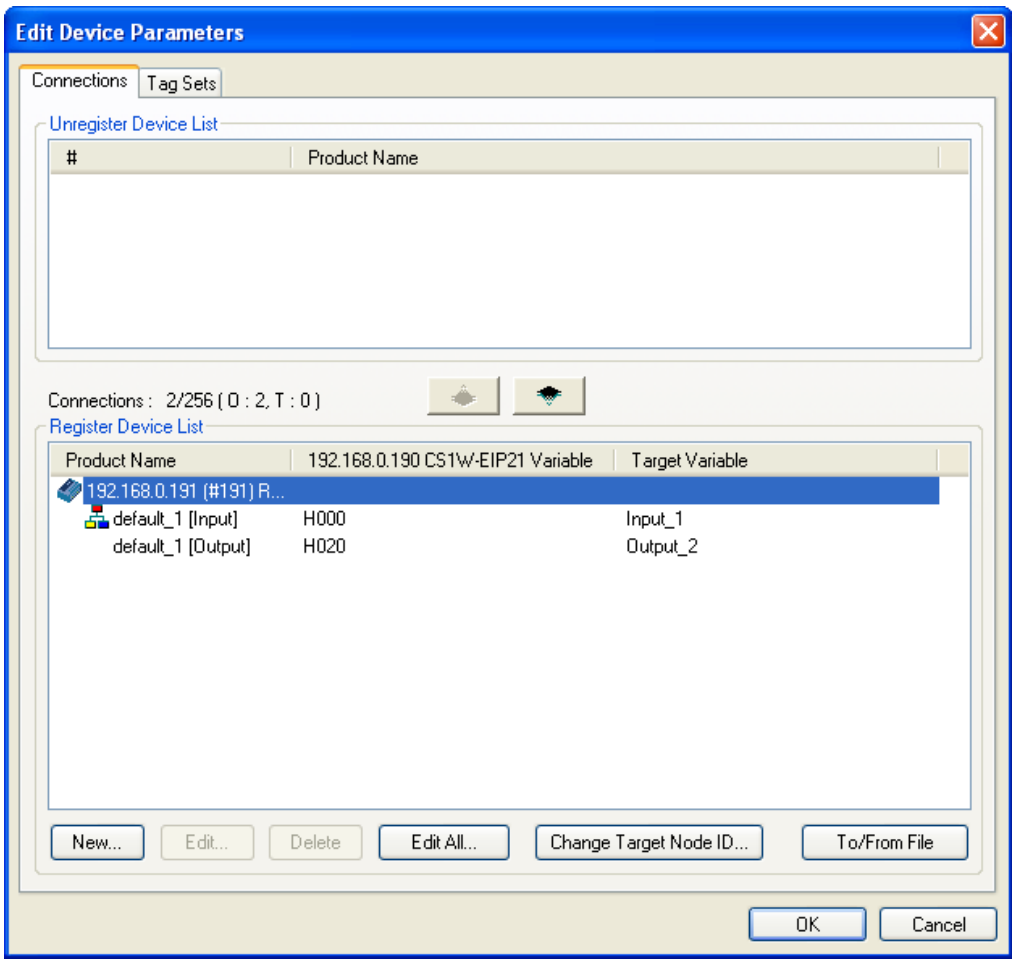
Field	Value
Connection Name	Optionally assign a user-readable name for this connection.
Connection I/O Type	Select **Input/Output**. The other options are only for use by advanced users.
Packet Interval (RPI)	Select the desired update rate. A commonly used RPI is 20.0 ms. Very low RPIs may flood the network, and reduce network reliability.
Timeout Value	Select **Packet Interval (RPI) x 4**, although for the minimum RPI of 2.0 ms, the Omron only supports **Packet Interval (RPI) x 8** or higher.
Input from Target Device **Consume Variable**	Select the input tag set created in the previous steps, **H000 – [40Byte]** in this example.
Produce Variable	Leave this set to the default, **Input_1 – [40Byte]** in this example.
Connection Type	In most applications this should be set to **Point to Point connection**. In advanced applications that

	will share the data in multiple controllers, **Multi-cast connection** must be used.
Output to Target Device **Produce Variable**	Select the output tag set created in the previous steps, **H020 – [44Byte]** in this example.
Consume Variable	Leave this set to the default, **Output_2 – [44Byte]** in this example.
Connection Type	Select **Point to Point connection**.

e. Click **OK**.

f. If you have multiple connections, choose to add another tag for each connection, following steps c - e above.

The Connections tab may now look something like this:

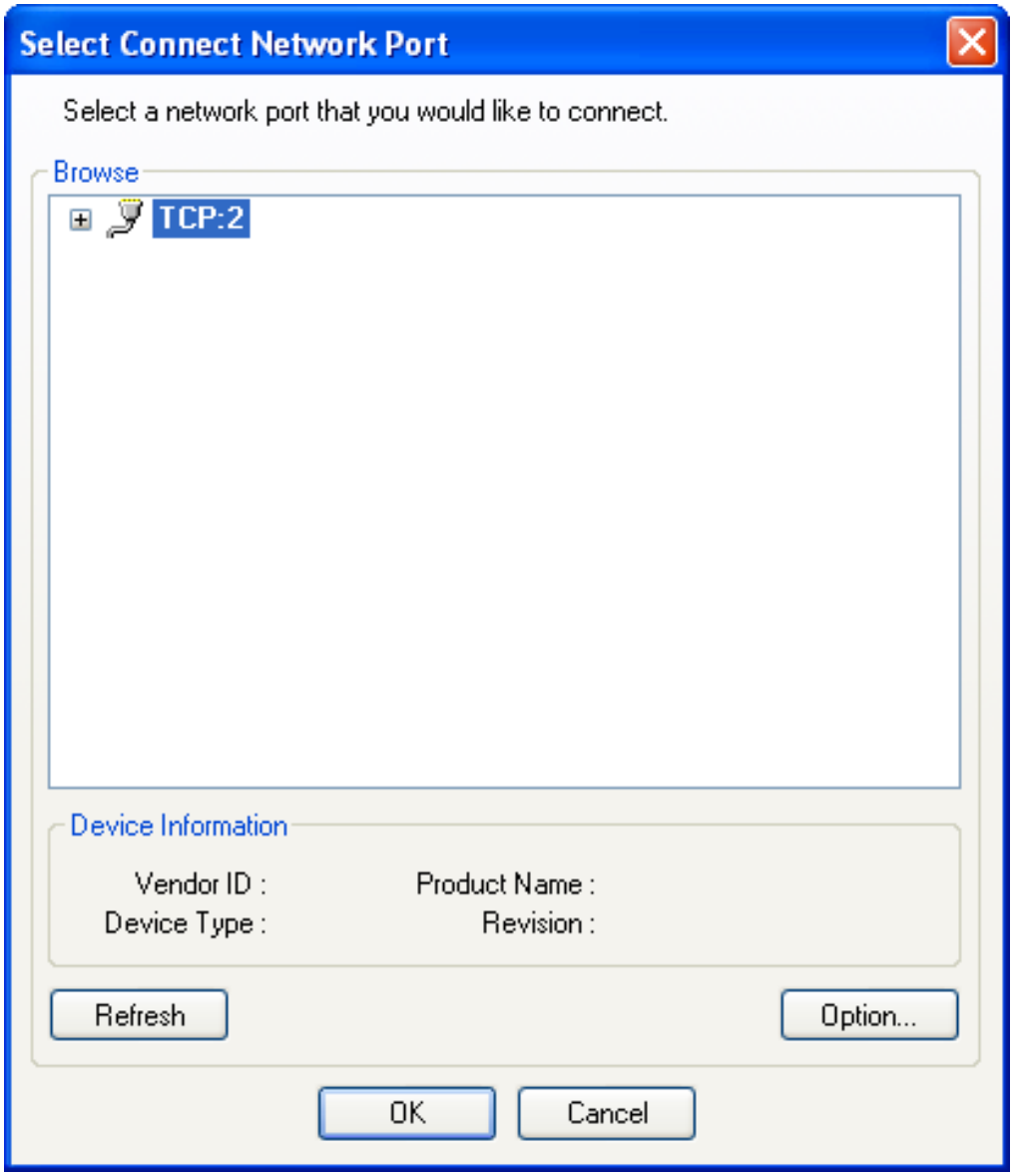


10. Add additional RMC devices to your EtherNet/IP network.
    a. Repeat steps 6-9 above for each additional RMC device.

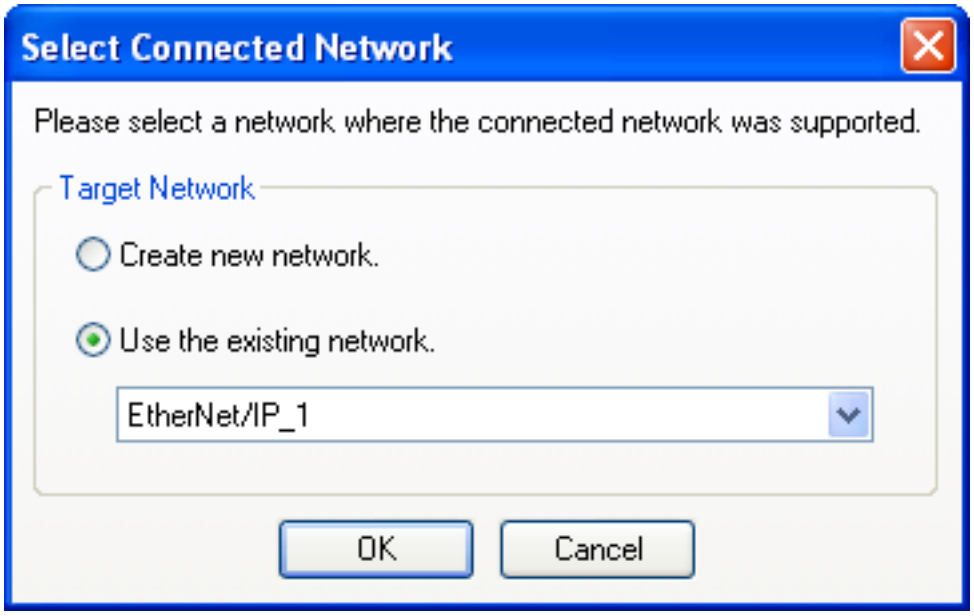
11. Download the Network Configuration
    a. On the **Option** menu, point to **Select Interface**, and then click **Ethernet I/P**.

b. On the **Network** menu, click **Connect**.

c. In the **Select Connect Network Port** window, select the Ethernet interface to use (which will most often be just one), and click **OK**.



d. In the Select Connected Network, select to use the existing network you just created and click **OK**.



e. On the **Network** menu, click **Download.** Answer **Yes** to any prompts.

12. Save the Network Configuration
    a. On the **File** menu, click **Save**. Follow the instructions to save the network configuration.

Performing Communications

Once the EtherNet/IP connection is configured and applied to the PLC, the communications will automatically start, assuming the PLC and RMC are both on a properly set up Ethernet network. Notice that the Omron will communicate even when it is in Program mode.

Reading Data from the RMC

This data will automatically update each Requested packet interval, and you can use the data as you wish. You cannot write to the Input Data. The Input Data will contain the Indirect Data from the RMC. However, if you selected to use the Sync Register, the first item in the Input Data array will be the SyncIn value, followed by the Indirect Data from the RMC.

The SyncIn is used only for synchronizing commands with the logic, as explained below. The SyncIn and SyncOut registers are only visible in the PLC. They are not visible in RMCTools. If you are using multiple connections, keep in mind that the connections are independent of each other, and the data may not be transferred at exactly the same time, even if the Sync Registers are used and are changed simultaneously.

Writing to the RMC - General

For each connection *without* a Sync Register, the Output Data is written to the RMC when any value in the Output Data changes.

For each connection *with* a Sync Register, the Output Data is sent to the RMC at each Requested Packet Interval, but the RMC ignores it until the SyncOut register changes. The first item in the Output Data array is the SyncOut register, followed by the registers that will be sent to the Incoming Data location in the RMC. Use the following procedure to write to the RMC with a Sync Register:

1. **Wait Until the Sync In and Sync Out Registers Match**
   If they do not match, then this means that another write is in progress.
2. **Write to the Output Data**
   Write the desired to the Output Data array.

3. **Change the Sync Out Register**
   The easiest way to do this is to add one to it. However, you must take care to handle overflowing this register (the Sync register is a REAL). One method is to add one and then MOD it with some large number, such as 10000. This will make the register count from 0 to 9,999, and then wrap back down to 0 without an error.
4. **Wait Until the Sync In and Sync Out Registers Match**
   This indicates that the RMC has received the data and processed it.

If you are using multiple connections with sync registers, you must change the Sync Register independently for each connection. The connections are independent of each other, and the data may not be transferred at exactly the same time, even if the Sync Registers are changed simultaneously.

See Using an EtherNet/IP I/O Connection for further details.

Sending Commands to the RMC

If you are writing to the Command Area, either directly or via the Indirect Data Map, Delta recommends using a Sync Register and following the procedure below to send commands to the RMC. The Output Data is sent to the RMC each RPI, but the RMC ignores it until the SyncOut register changes. The first item in the Output Data array is the SyncOut register, followed by the registers that will be sent to the Incoming Cyclic I/O Data location in the RMC.

1. **Wait Until the Sync In and Sync Out Registers Match**
   If they do not match, then this means that another write is in progress.
2. **Clear Old Commands from the Command Registers**
   Clear old commands from the command registers in the Output Data. Otherwise, when the Sync Out register is changed, the commands would be re-issued. One method of clearing the old commands is to fill the Output Data array with zeroes (except the SyncOut value).
3. **Write to the Command Registers**
   Write the Command registers and all required command parameters to the Output Data for all commands you want to issue. You can issue up to one command per axis. Leave the Command register set to 0 for each axis that will not receive a command.

   For example, if a portion of the Output Data is going to the Command Area for Axis 0, and you wish to issue a Move Absolute Command (20) to Axis 0 with a position of 6.7, a Speed of 3, and Accel and Decel of 50, you would write the following:

Value
20 (for a move Absolute Command)
6.7 (Position)
3 (Speed)
50 (Accel)
50 (Decel)
0 (Direction)

   Use the online help for each command to find out how many parameters a command has and what they mean. Make sure to write to all the parameters that the command uses. You do not need to write to command parameters that are not used by the command.
4. **Change the Sync Out Register**
   The easiest way to do this is to add one to it. However, you must take care to handle overflowing this register (the Sync register is a REAL). One method is to add one and then MOD it with some large number, such as 10000. This will make the register count from 0 to 9,999, and then wrap back down to 0 without an error. Take care to ensure that you only update the Sync Out Register once so that the commands do not get re-issued.
5. **Wait Until the Sync In and Sync Out Registers Match**
   This indicates that the RMC has received the command and issued it. It is important to wait until the SyncIn and SyncOut match before using the status bits in the Input Data (if the Input Data includes any status bits). See the Using an EtherNet/IP I/O Connection topic for how problems can occur if this step is ignored.

For multiple connections to one RMC, each connection has an individual Sync Register. The connections are independent of each other, and the data may not be transferred at exactly the same time, even if the Sync Registers are changed simultaneously. Therefore, it is best practice that if commands need to be sent simultaneously, that they are sent in the same Output Data block.

See Using an EtherNet/IP I/O Connection for further details.

Reading and Writing other registers

To read and write other registers in the RMC that are not included in the Incoming or Outgoing Cyclic I/O Data, you can use the SEND and RECV instructions as described in Using Omron Controllers via FINS. EtherNet/IP I/O and the SEND and RECV instructions can be used simultaneously.

Another option for writing to other RMC registers is to create user programs that move data from the variable table to other RMC registers.

See Also

EtherNet/IP Overview | Using Omron Controllers via FINS

6.10.13. Using Siemens S7 PLCs via PROFINET

Siemens offers several CPUs and Communication Processors (CPs) that support PROFINET, including products in the S7-300, S7-400, S7-1200 and S7-1500 lines. The procedures below describe using TIA Portal version 15 with the S7-1200 CPU, but the steps will be similar for other S7-family products and versions of TIA Portal.

The PROFINET IO connection can be configured for both cyclic and acyclic I/O data:

- **Cyclic I/O data** is always exchanged between the PLC and RMC at the specified update time. For example, status information from the RMC, and variables to be written to the RMC would typically be part of the cyclic data. Cyclic I/O data is defined by the Incoming and Outgoing Cyclic I/O Data. The RMC's support the following number of cyclic I/O registers:

	Max Cyclic I/O Registers in each direction
RMC75E	128
RMC150E	256
RMC200	256

  Some PLCs may be limited to fewer than the full size supported by the RMC controllers.

- **Acyclic data** is sent only when it is needed. For example, if the PLC creates a curve and sends it to the PLC, that data would typically be sent via the acyclic data. Or, if the PLC needs to read a captured plot from the RMC, that is also best done via the acyclic data. Acyclic I/O data is defined by the Data Records. The maximum length is 2048 registers (8192 bytes) for the RMC75 and RMC150 and 4096 registers (16,384 bytes) for the RMC200.

Determine I/O Data Locations in the RMC

PROFINET IO transfers data back and forth between the RMC and PLC at the specified update time. The user must specify which data items in the RMC should be sent and received. Typically, this is data in the Indirect Data Map.

Set up the Indirect Data map so that one part contains all the data coming from the PLC (Incoming Data), and another part contains all the data going to the PLC (Outgoing Data). Make sure the Incoming and Outgoing Data areas in the Indirect Data Map do not overlap.

The Outgoing Data typically includes RMC status items that the PLC always needs to keep track of, such as actual positions and status bits.

The Incoming Data consists of items that the PLC needs to write to in the RMC. This is typically variables and possibly command registers.

Note:
The Incoming and Outgoing Data locations need not be the Indirect Data Map. However, the Indirect Data Map is usually the best choice. Other options are the Variable Table and the command area.

Setting Up the RMC for PROFINET

Do the following in RMCTools:

1. **Select PROFINET Protocol Mode (RMC200 Only)**

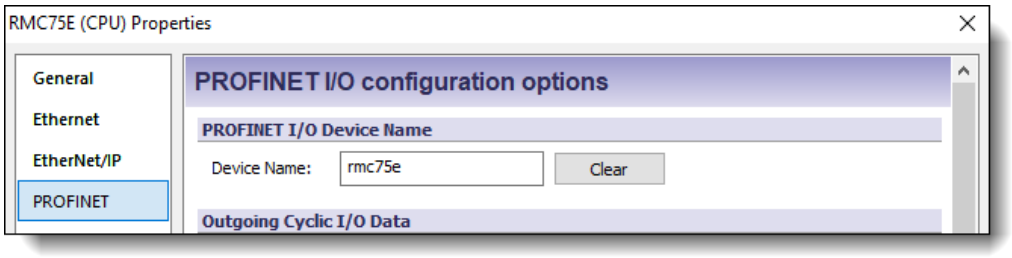
   In order to communicate via PROFINET I/O, the RMC200 must be set to PROFINET Mode:

   1. In the Project pane, expand the RMC, and double-click the **CPU**.
   2. On the **Ethernet** page, in the **Ethernet Protocol Mode** section, click **PROFINET Mode**, then click **OK**.

   The Ethernet Protocol Mode may also be viewed from the CPU20L or CPU40 Display Screen.

2. **Set the RMC's Device Name**

   In the Project pane, expand the **Modules** folder, double-click the CPU module, and choose **PROFINET**. In the **PROFINET I/O Device Name** section, enter the desired unique device name that your IO controller will use to refer to this device:



3. **Set the RMC's IP Address**

   With PROFINET you can set up the IP address either through RMCTools and instruct the PLC to not change the IP address, or you can allow the IO controller to set the IP address. If you are using Ethernet to connect RMCTools to the controller, then be sure that the IO controller and RMCTools do not conflict on the IP address the module will use.

4. **Set Up the Indirect Data Map**

   In the Project pane, double-click **Address Maps**, then click **Indirect Data Map**.

   Beginning at item 0 in the Indirect Data Map, choose the items for the Outgoing Cyclic I/O Data.

   At some location in the Indirect Data Map after the Outgoing Cyclic I/O Data area, choose the items for the Incoming Cyclic I/O Data, that is, the items that will be sent to the RMC from the PLC. If you are using another location for your Incoming Data, such as the Variable Table or Command Area, you need not set up the Indirect Data Map for the Incoming Data.

   Example

   If you have a 2-axis RMC75E controller, you may wish to set up the Outgoing Data at the beginning of the Indirect Data Map to include the Actual Position and the Status Bits for each axis, in addition to information on Task 0, which perhaps runs your user

programs.
You may also wish to set the Incoming Data, starting at item 32 in the Indirect Data Map, to go to the Axis 0 and Axis command registers, as well as some variables. Starting at 32 will simplify expanding the Outgoing Data later if the application ever requires it.
You could then set up the beginning of the Indirect Data Map like this for the Outgoing Data:

	Reg #	Map To	Description	Current
0	%MD18.0	%MD8.0 ...	Axis0 Status Bits	N/A
1	%MD18.1	%MD8.1	Axis0 Error Bits	N/A
2	%MD18.2	%MD8.8	Axis0 Actual Position (pu)	N/A
3	%MD18.3	%MD9.0	Axis1 Status Bits	N/A
4	%MD18.4	%MD9.1	Axis1 Error Bits	N/A
5	%MD18.5	%MD9.8	Axis1 Actual Position (pu)	N/A
6	%MD18.6	%MD24.0	Task 0 Status	N/A
7	%MD18.7	%MD24.3	Task 0 Current Program	N/A
8	%MD18.8			
9	%MD18.9			

You could then set up the Outgoing Data further on in the Indirect Data Map like this:

	Reg #	Map To	Description	Current
32	%MD18.32	%MD16.0	Axis0 Command Area	0.0
33	%MD18.33	%MD16.1	Axis0 Command Parameter 1	0.0
34	%MD18.34	%MD16.2	Axis0 Command Parameter 2	0.0
35	%MD18.35	%MD16.3	Axis0 Command Parameter 3	0.0
36	%MD18.36	%MD16.4	Axis0 Command Parameter 4	0.0
37	%MD18.37	%MD16.5	Axis0 Command Parameter 5	0.0
38	%MD18.38	%MD56.27	27 - (Var1)	0.0
39	%MD18.39	%MD56.28	28 - (Var2)	1.0
40	%MD18.40	%MD16.6	Axis1 Command Area	0.0
41	%MD18.41	%MD16.7	Axis1 Command Parameter 1	0.0
42	%MD18.42	%MD16.8	Axis1 Command Parameter 2	0.0
43	%MD18.43	%MD16.9	Axis1 Command Parameter 3	0.0
44	%MD18.44	%MD16.10	Axis1 Command Parameter 4	0.0
45	%MD18.45	%MD16.11	Axis1 Command Parameter 5	0.0
46	%MD18.46	%MD56.29	29 - (Var3)	0.0
47	%MD18.47	%MD56.30	30 - (Var4)	0.0
48	%MD18.48			

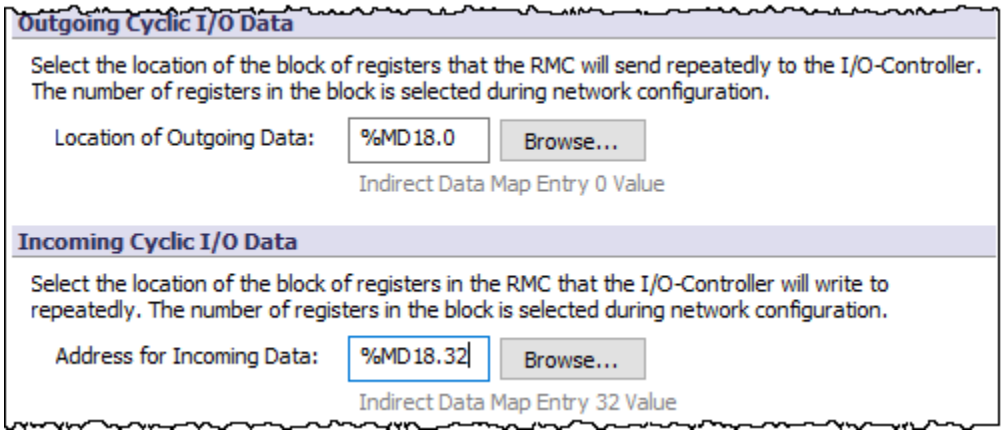
5. Set the Cyclic I/O Data Locations in the RMC

In the Project pane, expand the **Modules** folder, double-click the CPU module, and choose **PROFINET**.

In the **Outgoing Cyclic I/O Data** section, enter the starting location for the Outgoing Cyclic I/O Data. In our example, verify that the location is the Indirect Data Map Entry 0 Value.

In the **Incoming Cyclic I/O Data** section, enter the starting address for the Incoming Cyclic I/O Data. Click the **Browse** button and browse to the desired RMC location for the outgoing data. This should be a location in the Indirect Data Map, the Variable Table, or Command Area as discussed in the **Determine I/O Data Locations** in the RMC section above.

For example, the PROFINET I/O Configuration Options page below shows an RMC75E with the Outgoing Data coming from the Indirect Data map starting at item 0 and the Incoming Data going to the Indirect Data starting at item 32.

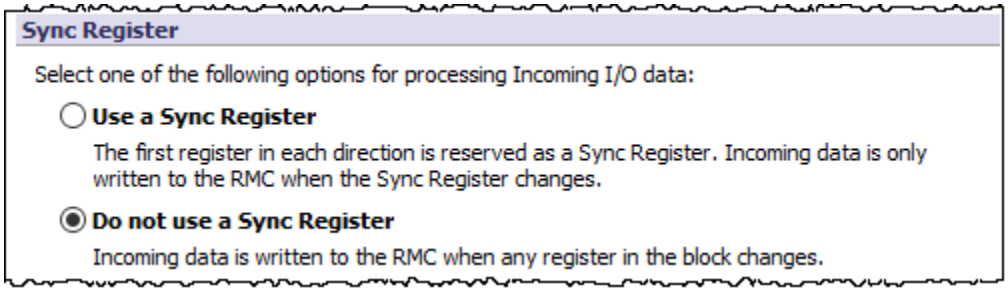


6. Choose Whether to Use a Sync Register

The Sync Register provides a method for the PLC to synchronize the Input Data and Output Data. With a Sync Register, the Incoming Data is not written to the RMC until the Sync Register changes. If you prefer to have the Incoming Data be written whenever any value in the Incoming Data changes, choose the option to not use a Sync Register.

Because the S7 PLCs use the DPRD_DAT (SFC14) and DPWR_DAT (SFC15) system functions to control when the data is copied into and out of the S7's DBs, the Sync Register generally does not need to be used with the S7.

To select whether or not to use the Sync Register, in the Project pane, expand the **Modules** folder, double-click the CPU module, and choose **PROFINET**. Then select the desired option under Sync Register:



For more details, see the Using a PROFINET I/O Connection topic.

7. Configure Data Records

While Data Records can be set up at this time, they are not required to establish the PROFINET connection, and are generally better to set up later in the process, once you are

planning the acyclic reads and writes the PLC will need to do. See the Using PROFINET Record Data topic for details.

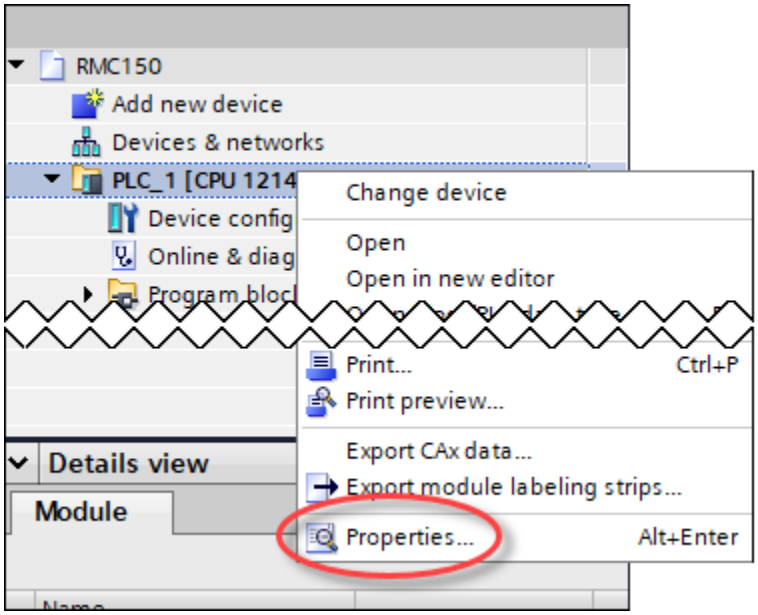
8. **Set the Byte Order**

   In the Project pane, expand the **Modules** folder, double-click the CPU module, and choose **PROFINET**. In the **Data Encoding** section, make sure the **Byte Order** is set to **MSB First (BE)**.

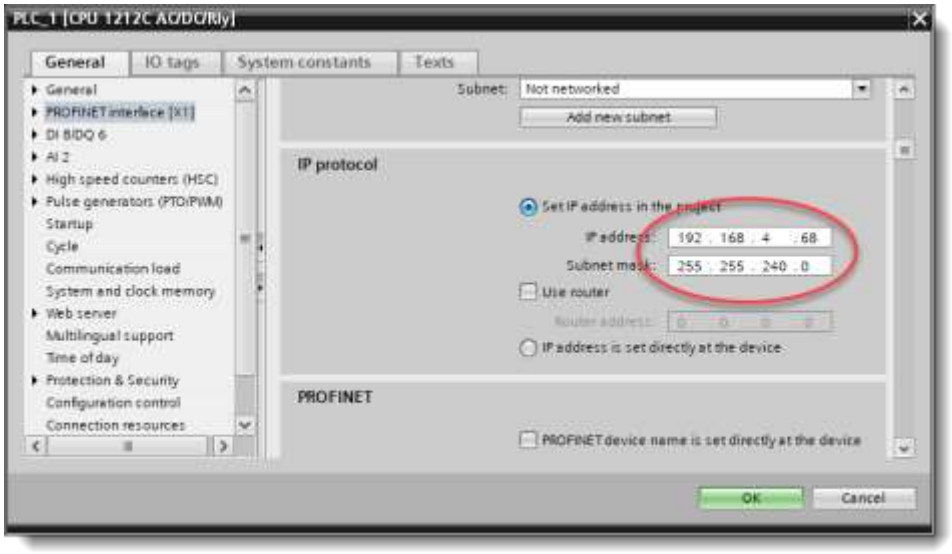
Creating the PROFINET Connection in the S7

After the RMC has been set up using RMCTools, the TIA Portal software is used to configure the PROFINET network for the S7 PLC. These steps describe using TIA Portal version 15 with the S7-1200 as the IO controller.

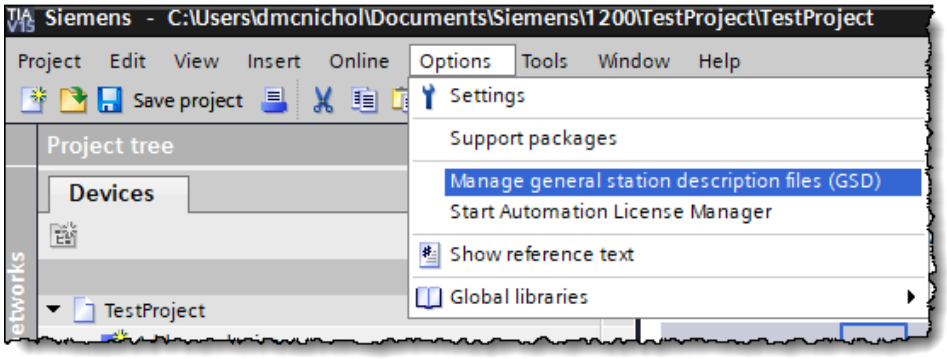
1. Start TIA Portal and open the project to which you will add an RMC to the PROFINET network.
2. Set the Ethernet IP address of your PLC:
   a. In the project tree, right-click the PLC's CPU module and select **Properties**.



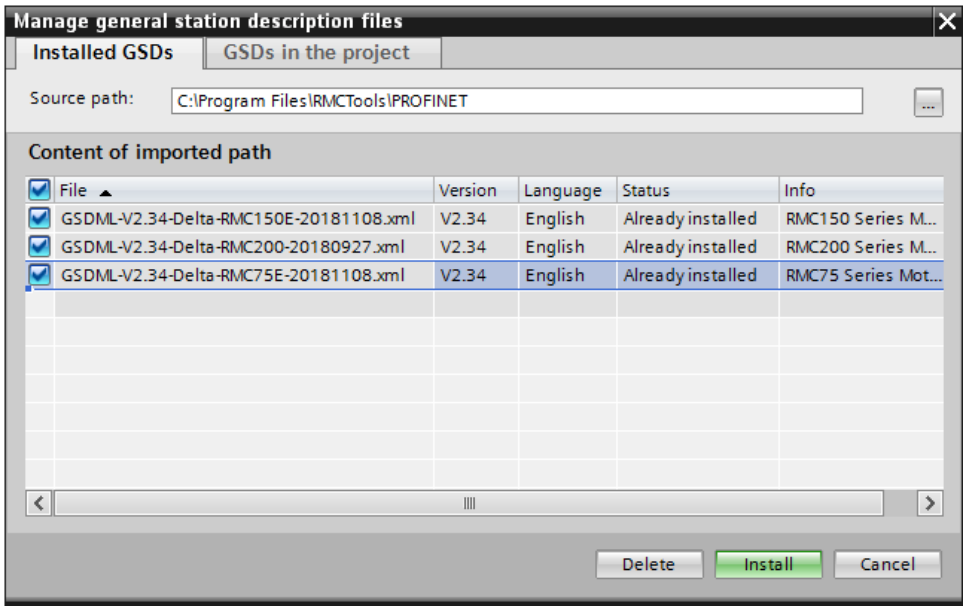
   b. On the **General** tab, select **PROFINET interface**.
   c. Choose **Set IP address in this project**, set the **IP address** and **Subnet mask**, then click **OK**.



3. If the RMC GSD files have not yet been added to the hardware catalog, then use the following steps to add them:
    a. On the **Options** menu, choose **Manage general station description files (GSD)**.

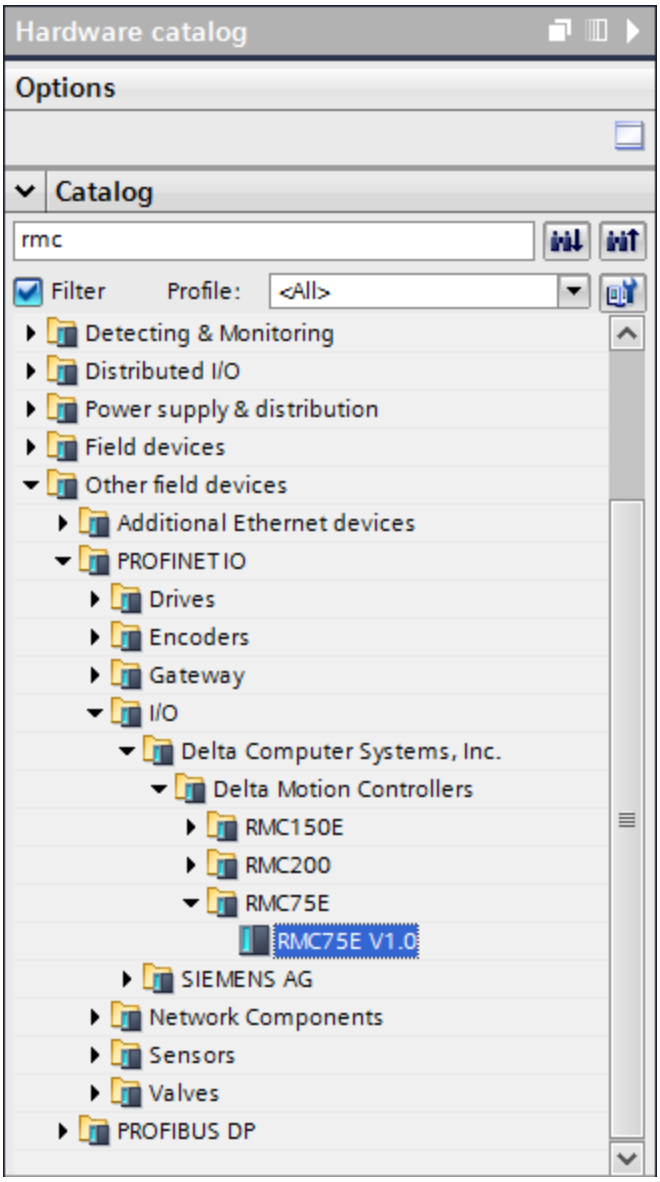


    b. On the **Installed GSDs** tab, in the **Source Path**, browse to the PROFINET folder under the RMCTools install folder. This is typically C:\Program Files\RMCTools\PROFINET.
    c. Select the most recent GSD file versions for the RMCs, as shown below. If your software does not accept the most recent version, contact Delta for older versions.

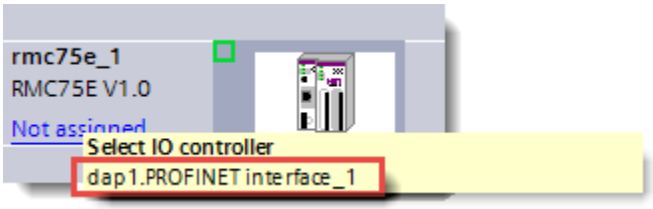


d. Click **Install** and follow the instructions to install these GSD files. Click **Close** when done.

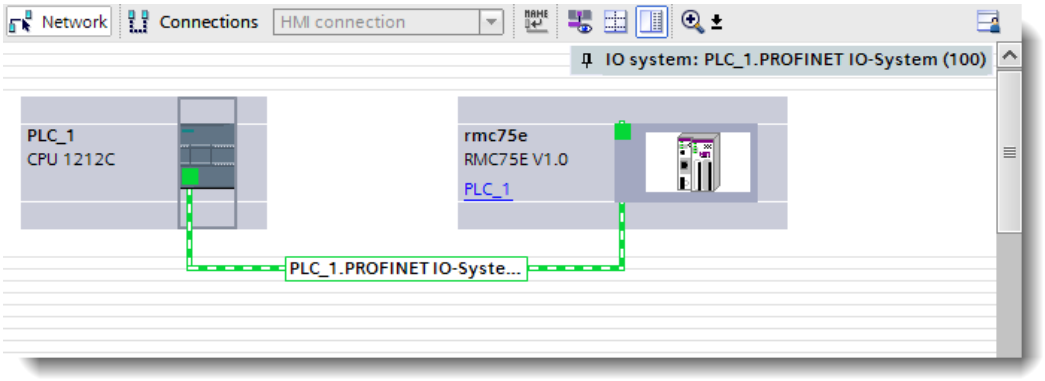
4. In the project tree, expand **Devices & networks** and open the **Network View**.
5. Use the Hardware Catalog to find the RMC controllers:



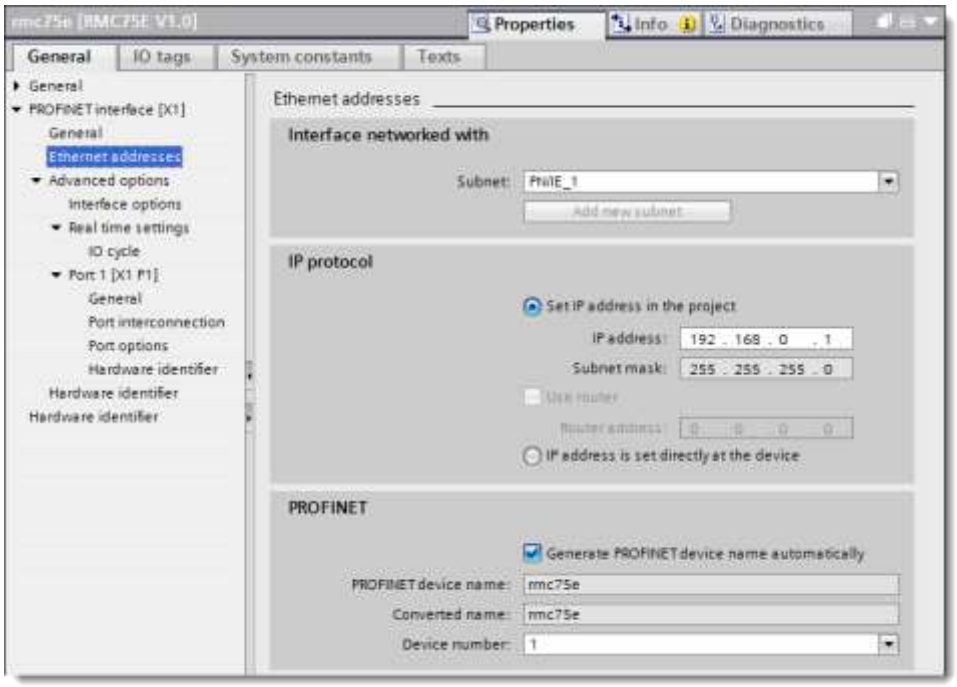
If the RMC device does not show that it is assigned to the PLC, then click the **Not assigned** link and select the PLC:



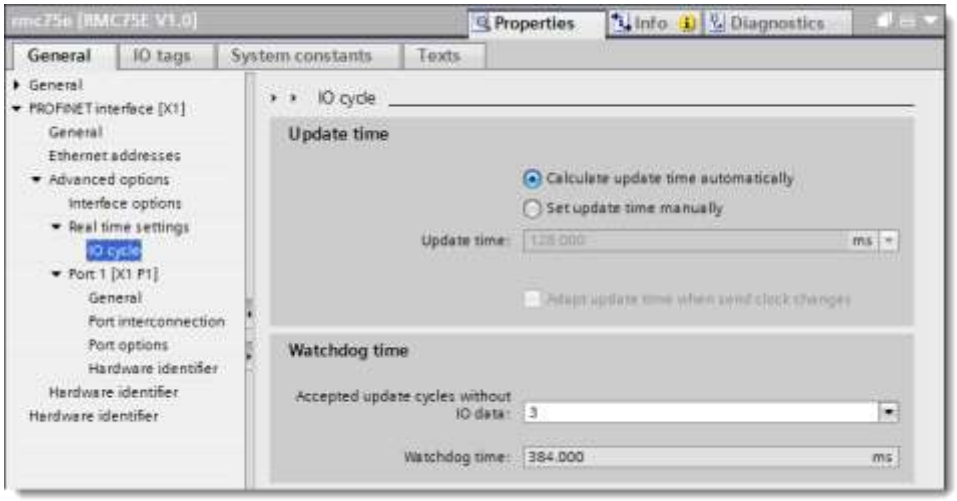
6. Drag the desired RMC device onto the PROFINET-IO-System network and connect the RMC to the PLC:



7. To set the PROFINET device name and IP address settings that the PLC expects for the RMC:
   a. Open the properties for the RMC device and browse to the **Ethernet Addresses** page as shown below:



   b. Under **PROFINET**, make sure that the **PROFINET Device Name** matches the device name set in the RMC through RMCTools. To change the name, you can either rename the device itself in TIA Portal and leave **Generate PROFINET device name automatically** checked, or uncheck **Generate PROFINET device name automatically** and enter the PROFINET device name.
   c. Under **IP protocol**, if you have set the RMC's IP address settings through RMCTools and want the S7 to use them as is, then select **IP address is set directly at the device.** Otherwise, select **Set IP address in the project**, and the IP address that the S7 will give to the RMC.
8. To view or change the PROFINET update time for the RMC:
   a. Open the properties for the RMC device and browse to the **IO Cycle** page as shown below:

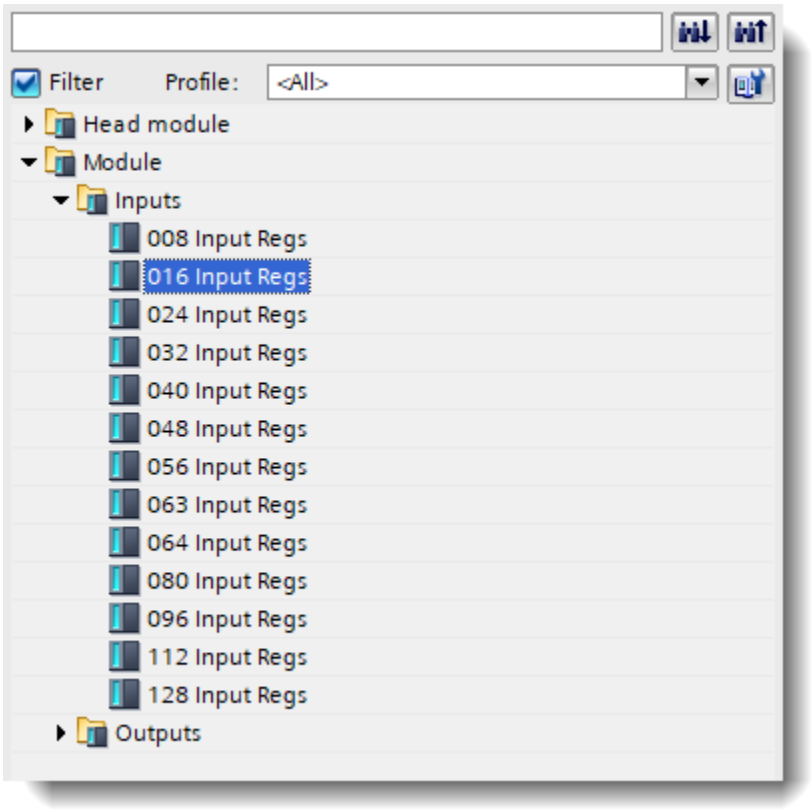


b. Under **Update Time**, select a value or accept the automatically-calculated value. The RMC75 and RMC150 support between 2.0 and 512.0 ms. The RMC200 supports between 1.0 and 512.0 ms. A commonly-used update time is 16.0 ms. To set the update time manually, first select **Set update time manually**.

c. Under **Watchdog Time**, the default number of update cycles should typically be used, which is 3 in this case.

9. Select the Input Data and Output Data lengths as follows. The Input Data corresponds to the Outgoing Cyclic I/O Data in the RMC and the Output Data corresponds to the Incoming Cyclic I/O Data in the RMC.

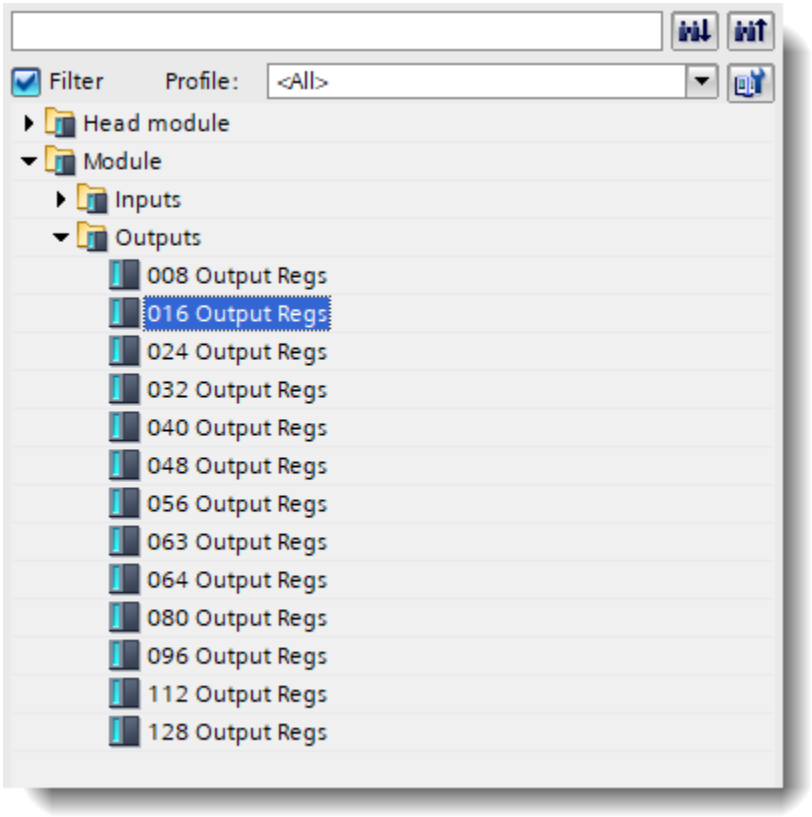
   a. In the Hardware Catalog under the device you added to the PROFINET network, expand the **Inputs** folder.



b. Determine which input module you need based on the number of registers to transfer.

c. Drag the desired input module onto slot 1 in the RMC's slot view.

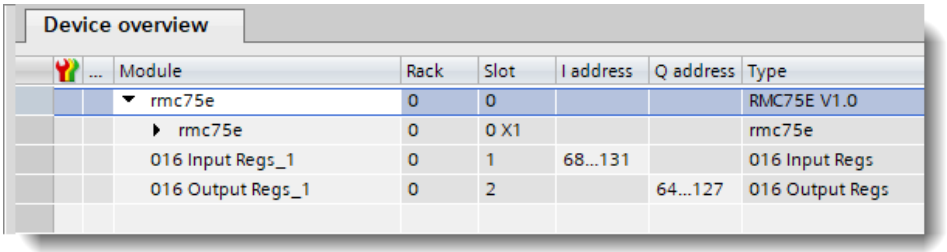
d. In the Hardware Catalog, expand the **Outputs** folder.



e. Determine which output module you need based on the number of registers to transfer.

f. Drag the desired output module onto slot 2 in the RMC's slot view.

g. The final module configuration will look similar to the following:

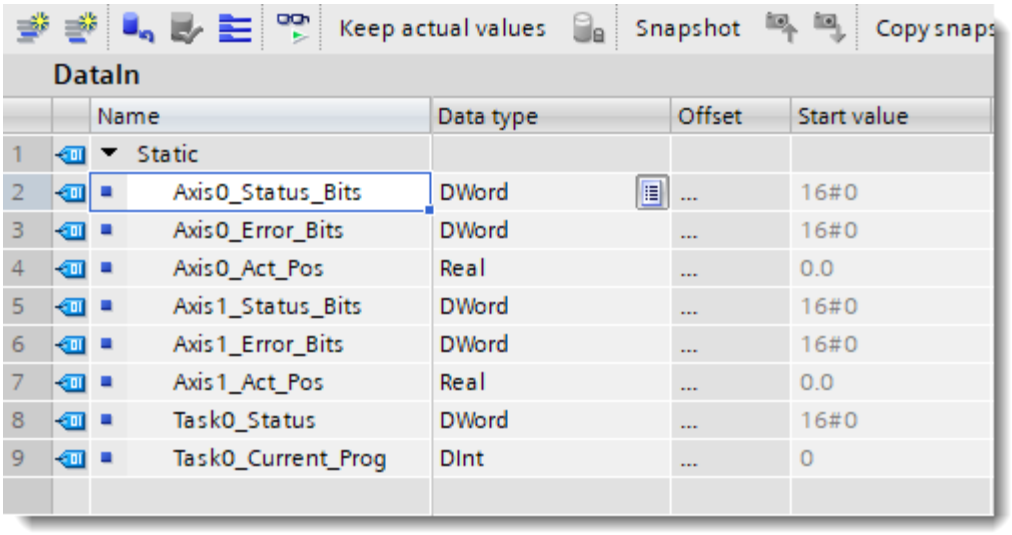


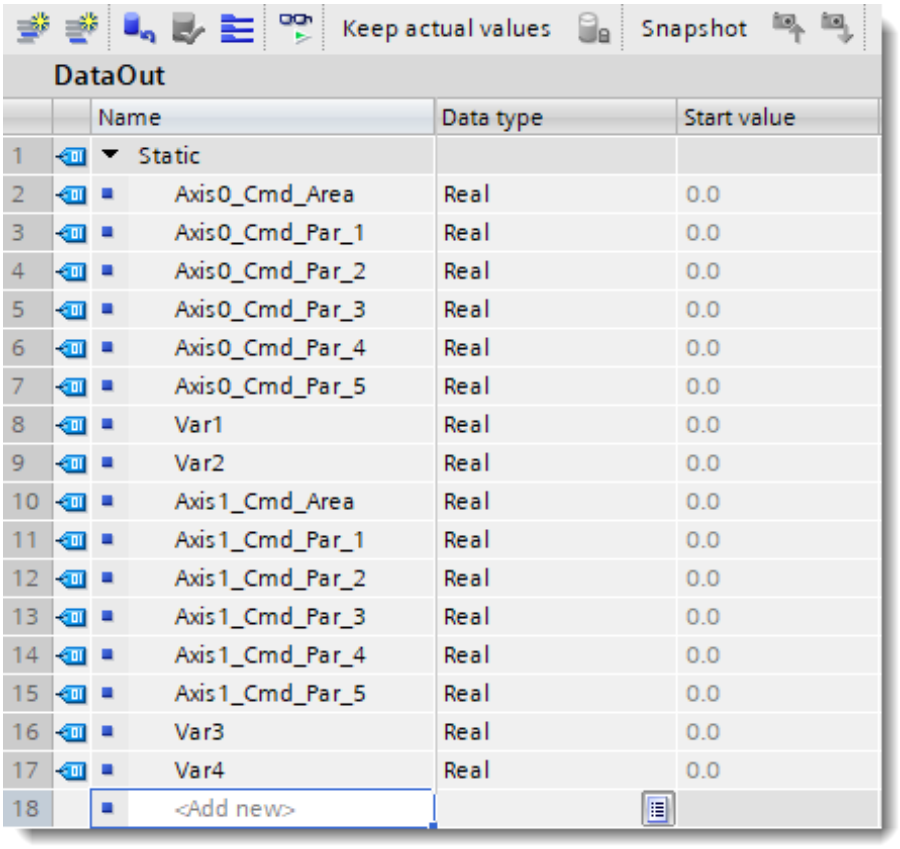
10. Save and download the configuration to your S7 controller.

Using the Cyclic I/O Data in the S7

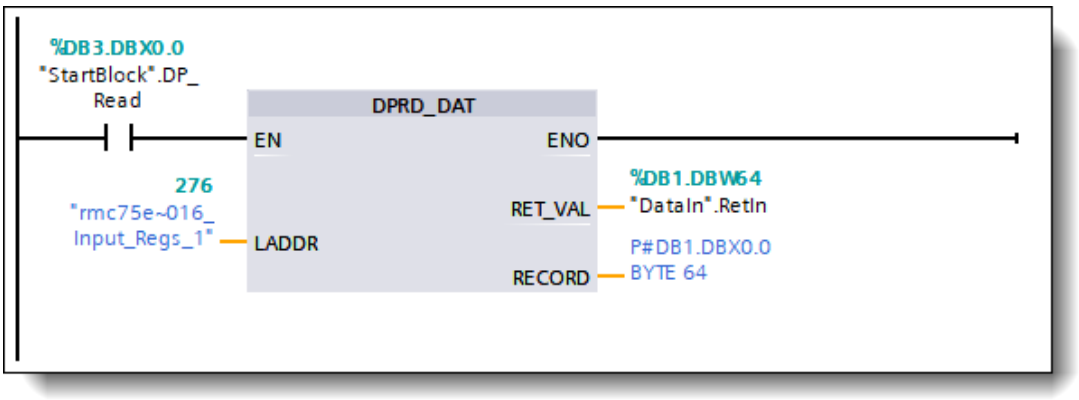
The first step in using cyclic I/O data in the S7 is to define a data block (DB) for each of the Input Data and Output Data. The structure of the Input and Output DBs should match the data defined in the RMC for the Outgoing and Incoming Cyclic I/O Data respectively.

This example matches the data entered in the Indirect Data Map above:

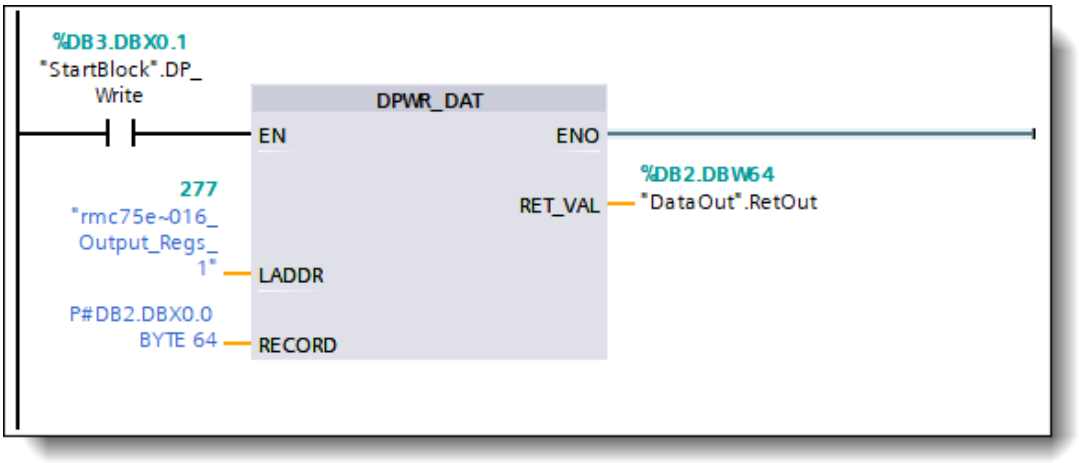




Within TIA Portal, the DPRD_DAT system function (SFC14) is used to get a consistent copy of the Input Data. The following ladder shows SFC14 taking a copy of the input data from the RMC and storing it into DB1.



After the input data has been loaded into DB1 using the DPRD_DAT system function, the S7 program can use this data and should set up the outgoing data in the output DB (DB2 in this example). After the output data has been fully set up, the DPWR_DAT system function (SFC15) is used to send a consistent copy of the data out over PROFINET to the RMC.



Notice that the RMC's I and Q data should generally not be accessed directly but should instead go through the DPRD_DAT and DPWR_DAT SFCs in order to ensure that the data is always handled as a consistent block.

Reading and Writing Record Data from the S7

Use the Data Records to read from or write to any location in the RMC. Record Data reads and writes can be performed while the cyclic data exchange is occurring. The maximum Record Data read or write length is 2048 32-bit registers (8192 bytes) for the RMC75 and RMC150, and 4096 32-bit registers (16,384 bytes) for the RMC200.

- **RMC75 and RMC150:**

  Fixed Data Records: Data Records 7-255 correspond to the register files 7 -255 in the RMC, as listed in the RMC150 Register Map and RMC75 Register Map. Reads and writes of Data Records 7-255 will begin at element 0 of the file. Fixed Data Records are useful for accessing plot data, the Variable Table, and even the command area.

  Custom Data Records: The locations of Data Records 1000-1003 can be specified in the PROFINET Settings Page in RMCTools. This overcomes the Fixed Data Records limitations of starting at element 0. Custom Data Records are useful for accessing any location in the RMC.

- **RMC200:**

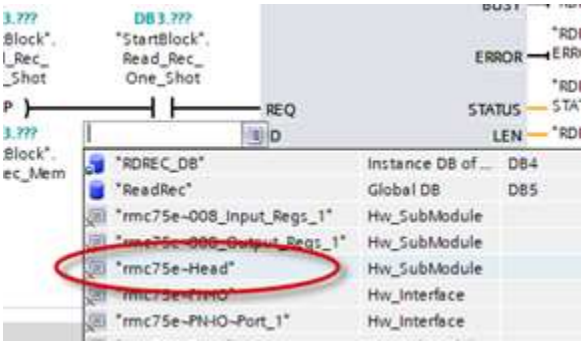
Fixed Data Records
Data Records 0-3 are assigned as listed in the PROFINET Data Records Address Map. Reads and writes of Data Records always begins at the first element in the record.

Configurable Data Records
The user can configure 32 more Data Records in the PROFINET Data Records Address Map. This is useful for accessing items such as plot data.

Reading and Writing Record Data

To read and write Record Data, use the RDREC (SFB52) and WRREC (SFB53) system function blocks.

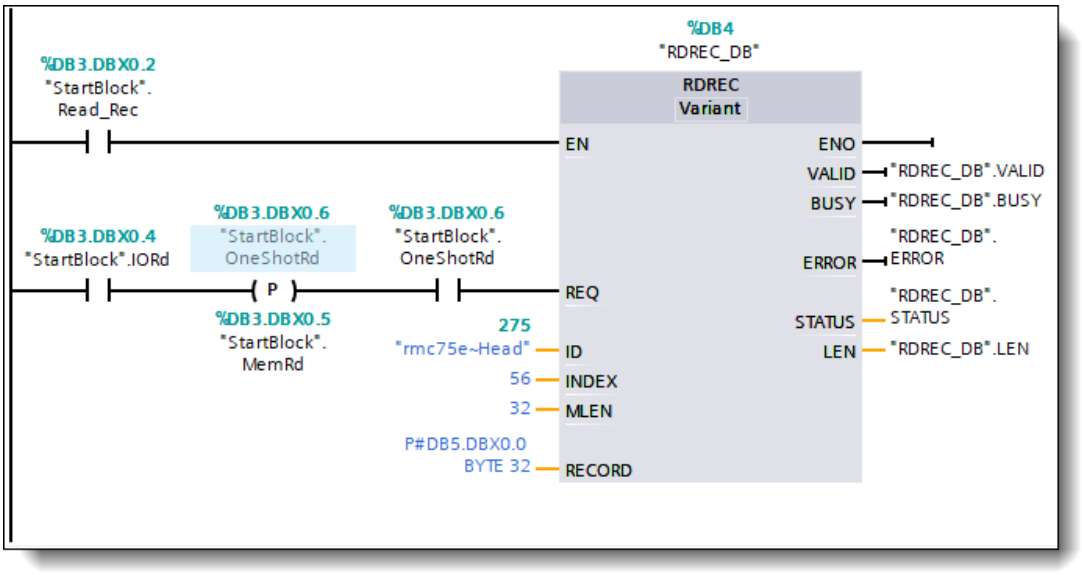
1. Insert the RDREC (SFB52) or WRREC (SFB53).
2. At the top of the SFB, type a name of a new Data Block (DB). If it does not exist, after pressing Enter, you will be prompted to create a new one. Choose **Yes**.
3. Wire the **EN** input to turn on as required by your application.
4. Wire the **REQ** input so that it will turn on for one scan to trigger the read or write. This can be done with a one-shot.
5. For the **ID**, choose the **~Head** of the RMC device:



6. **Index** is the Data Record number to read from or write to.
7. Set **MLen** or **Len** to four times the number of 32-bit registers to be sent or received. The maximum length is 8192 bytes (2048 32-bit registers) for the RMC75 and RMC150, and 16,384 bytes (4096 32-bit registers) for the RMC200.
8. Set **Record** to the Data Block that contains the source or destination data. You must define this Data Block. The contents of the Data Block should match the data area of the RMC.
9. Wire the **Valid**, **Busy**, **Error**, and **Status** outputs as required by your application.

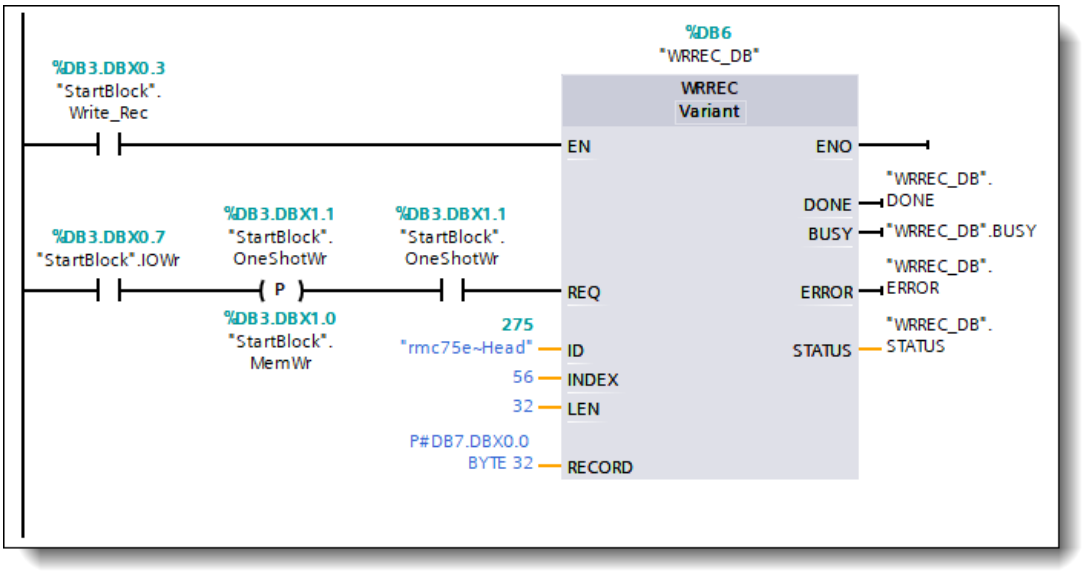
RDREC Example

This example reads 8 registers (32 bytes) starting at %MD56.0 for the RMC75E.



WRREC Example

This example writes 8 registers (32 bytes) starting at %MD56.0 for the RMC75E.



See Also

PROFINET Overview | Using Siemens S7 PLCs via PROFIBUS | Using a PROFINET I/O Connection | Setting up a PROFINET I/O Connection | Using PROFINET Record Data | Handling Broken PROFINET Connections | Troubleshooting PROFINET

6.10.14. Using Siemens S7 PLCs via PROFIBUS

The Siemens S7 PLC can communicate with the RMC75P or the RMC150E PROFIBUS module via PROFIBUS-DP. The PROFIBUS Overview topic describes the PROFIBUS communications.

See Also

Communications Overview | Using Siemens S7 PLCs via PROFINET

6.10.15. Using Wonderware with the RMC

The RMC family of controllers can communicate with Wonderware InTouch via Ethernet.

Example Programs

Delta provides example PLC programs to help you quickly set up the communications between your PLC and the RMC. See the downloads section of Delta's website at https://deltamotion.com.

Communicating with the RMC

RMC75E, RMC150E, or RMC200

Either the DASABCIP server (CIP) or DASMBTCP server (Modbus/TCP) can be used to communicate with the RMC75 via Ethernet. This topic describes both methods. The DASABTCP server can also used, but it does not easily read individual bits in a DWORD.

RMC75S

The DASMBSerial server (Modbus/RTU) can be used. This topic does not describe this method, although it is similar to using the DASMBTCP server.

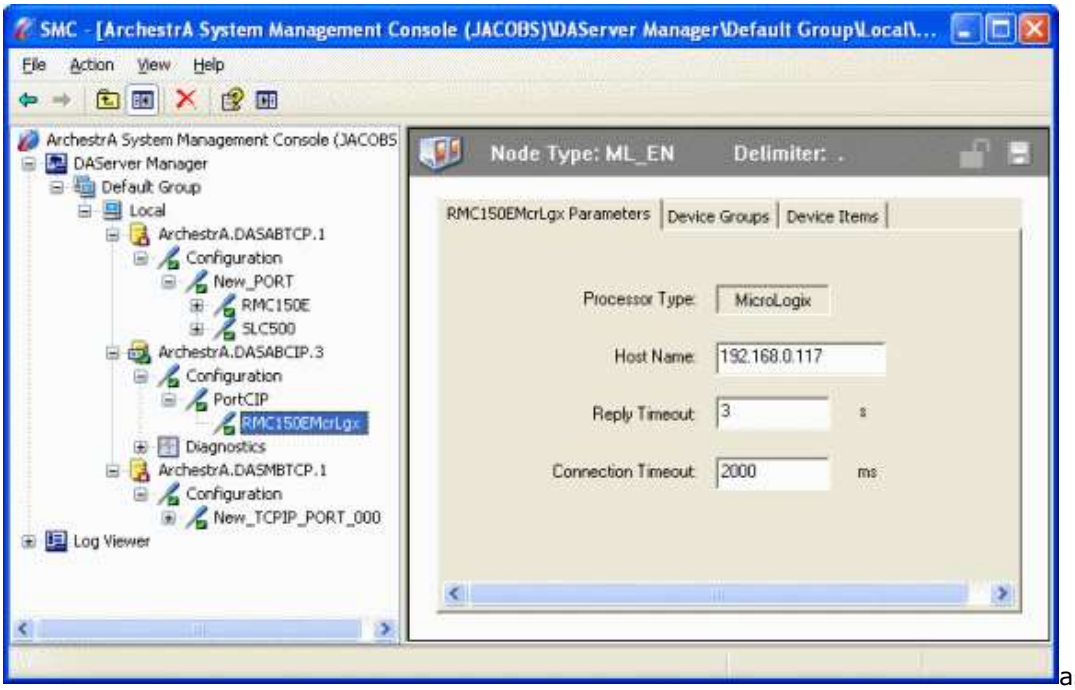
Using the DASABCIP Server

The DASABCIP Server uses EtherNet/IP communications.

Setting up the DASABCIP Server

Note: These instructions are for DASServer manager version 0750.0065.

1. Install the DASABCIP server.
2. In the ArchestrA System Management Console, under the ArchestrA.DASABCIP folder, add a new port.
3. Right-click the new port and choose **Add an ML_EN Object**. Set up the ML_EN Object as follows:
    a. In the **Host Name** box, enter the IP address of the RMC.
4. An example configuration is shown below.



asdf

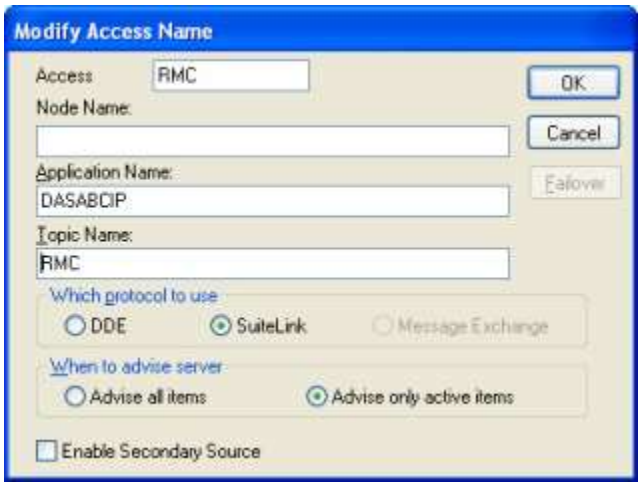
5. On the **Device Groups** tab, add a device called RMC.
6. Make sure to activate the DASABCIP server.

Adding an RMC Analog Display to an InTouch Window

Follow these steps to add the Axis 0 Actual Position to an InTouch window.

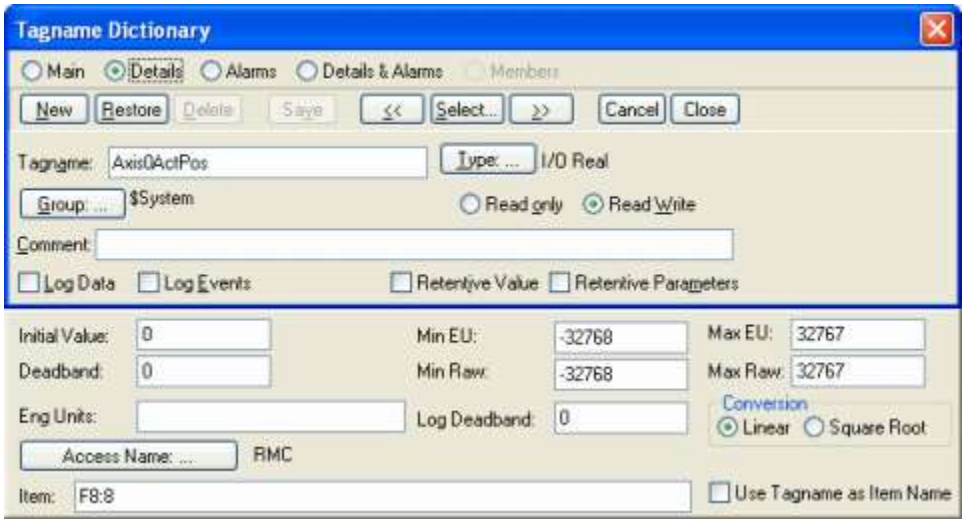
Note: These instructions are for InTouch 9.5.

1. On the **Special** menu, click **Access Names**, then click **Add**, and do the following:
   a. In the **Access** box, type an name like "RMC" (without the parentheses).
   b. In the **Application Name** box, type "DASABCIP" (without the parentheses).
   c. In the **Topic Name** box, type the name you entered in the Device Group tab in the System Management Console. This is "RMC" (without the parentheses).
   d. Click **OK.**



2. On the **Special** menu, click **Tagname Dictionary**, then click **New**, and do the following:
   a. In the **Tagname** box, type an name like "Axis0ActPos" (without the parentheses).

b. Click **Type** and choose **I/O REAL**, and click **OK**.
c. Choose **Read Only**.
d. Click **Access name**, choose **RMC** and click **Close**.
e. In the **Item** box, type "F8:8" (without the parentheses). This means it is looking at a register F8:8 in the RMC, which is the Axis 0 Actual Position.
f. Click **Save**, then click **Close**.



aa

3. In an InTouch window, go to the Wizard Selection dialog, choose **Value Displays**, choose **Analog Tagname Display**, and click **OK**.
4. Place the display on the window, double-click the display, and do the following:
   a. In the **Tagname** box, type "Axis0ActPos" (without the parentheses).
   b. In the **Number Format** box, choose ###,###.
   c. Click **OK**.
5. Click **Runtime!**
6. The runtime window will display the actual position of the RMC.

Item Address Details using DASABCIP

RMC data registers can be addressed as F or L registers. If the RMC register is a floating-point number, use F. If it is a DINT or DWORD use L.

To address a bit in an L word, use */b*, where bit is the bit number, beginning with 1. For example, L8:0/1 looks at the first bit in the Axis 0 Status bits.

Using the DASMBTCP Server

The DASMBTCP Server uses Modbus/TCP communications.

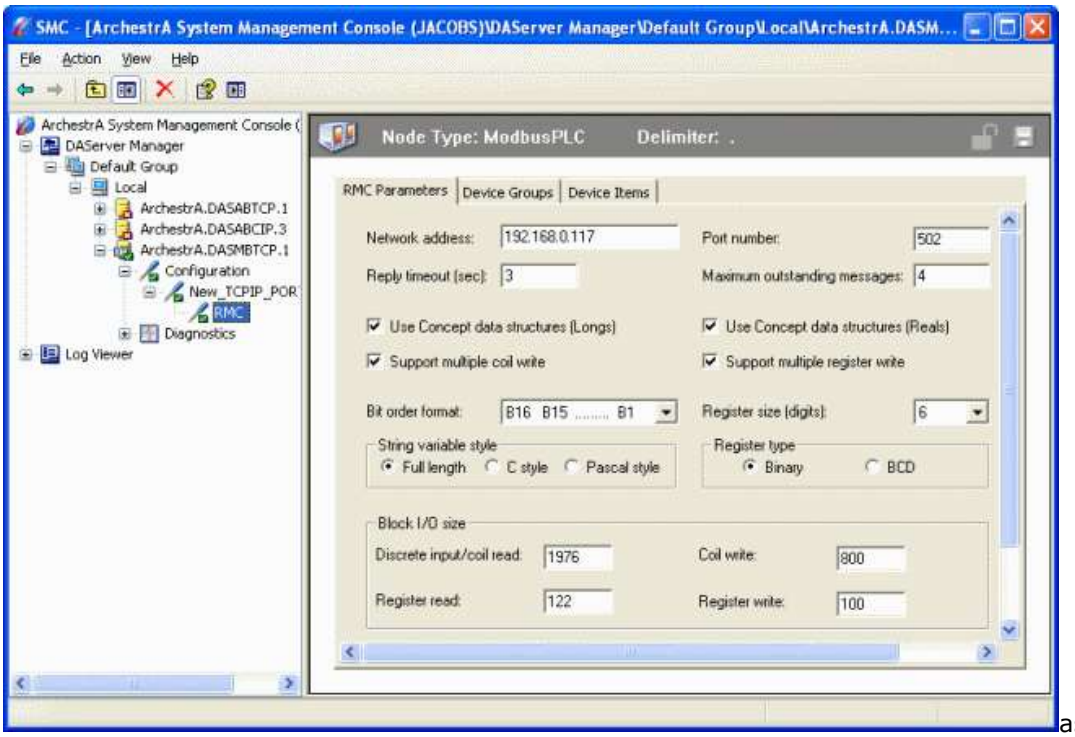
Setting up the DASMBTCP Server

Note: These instructions are for DASServer manager version 0750.0065.

1. Install the DASMBTCP server.
2. In the ArchestrA System Management Console, under the ArchestrA.DASMBTCP folder, add a new port. The port should be set to 502.
3. Right-click the new port and choose **Add a Modbus PLC Object**. Set up the ModbusPLC Object as follows:
   a. In the **Network address** box, enter the IP address of the RMC.
   b. In the **Bit order format** box, choose **B16 B16 … B1**.

c. In the **Register size** box, choose 6.
d. Select **Binary** Register Type.

4. An example configuration is shown below. Notice the IP Address is normally something like 192.168.0.34, but this example uses the URL for the online RMC75E.



asdf

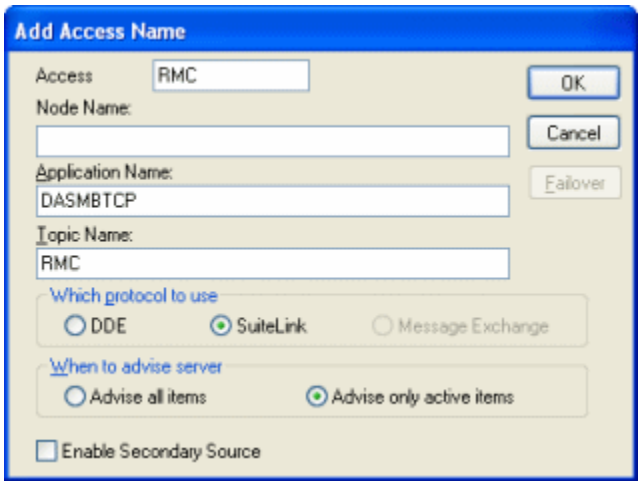
5. On the Device Groups tab, add a device called RMC.
6. Make sure to activate the DASMBTCP server.

Adding an RMC Analog Display to an InTouch Window

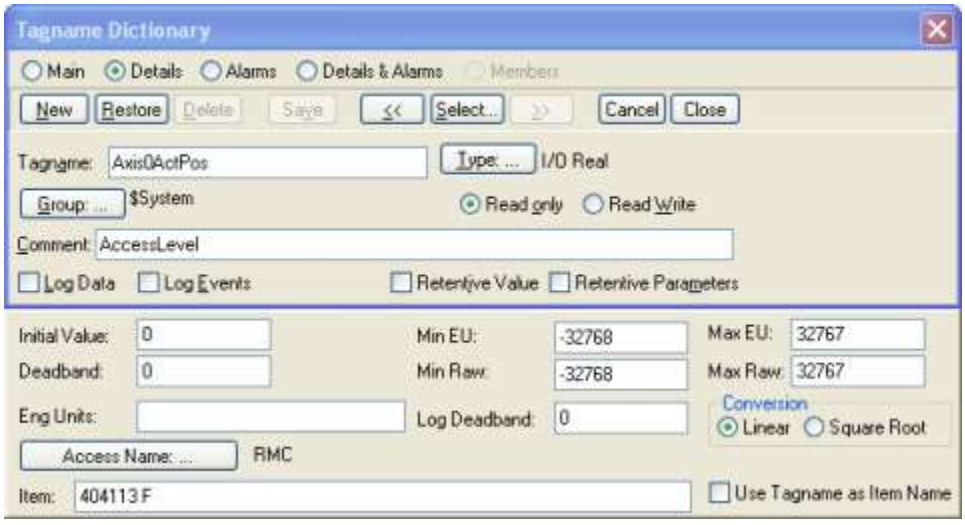
Follow these steps to add the Axis 0 Actual Position to an InTouch window.

Note: These instructions are for InTouch 9.5.

1. On the **Special** menu, click **Access Names**, then click **Add**, and do the following:
   a. In the **Access** box, type an name like "RMC" (without the parentheses).
   b. In the **Application Name** box, type "DASMBTCP" (without the parentheses).
   c. In the **Topic Name** box, type the name you entered in the Device Group tab in the System Management Console. This is "RMC" (without the parentheses).
   d. Click **OK.**



2. On the **Special** menu, click **Tagname Dictionary**, then click **New**, and do the following:
    a. In the **Tagname** box, type an name like "Axis0ActPos" (without the parentheses).
    b. Click **Type** and choose **I/O REAL**, and click **OK**.
    c. Choose **Read Only**.
    d. Click **Access name**, choose **RMC** and click **Close**.
    e. In the **Item** box, type "404113 F" (without the parentheses). This means it is looking at a "4" type register at address 4113 (the Modbus address for Axis 0 Actual Position) and it is a Floating-point value.
    f. Click **Save**, then click **Close**.



aa

3. In an InTouch window, go to the Wizard Selection dialog, choose **Value Displays**, choose **Analog Tagname Display**, and click **OK**.
4. Place the display on the window, double-click the display, and do the following:
    a. In the **Tagname** box, type "Axis0ActPos" (without the parentheses).
    b. In the **Number Format** box, choose 0.000.
    c. Click **OK**.
5. Click **Runtime!**
6. The runtime window will display the actual position of the RMC.

Item Address Details using DASMBTCP

All RMC data registers are of Modbus memory type "4".

In the **Item** box in the Tagname dictionary, adding " F" to the end of the Modbus address indicates it is a floating-point number. Adding " L" to the end of the Modbus address indicates it is a 32-bit integer. Adding ":" and then the number, such as "404097:1" treats it as a discrete, with number after the colon indicating the bit number. Wonderware counts bits starting at 1.

Calculating RMC Register Addresses

Use the Register Maps to find the Modbus/TCP addresses of the desired registers in the RMC75E. You can also use the Modbus Addressing topic to easily convert the two-level addressing to Modbus addresses.

See Also

Communications Overview | Register Maps

6.10.16. RMCLink .NET Assembly and ActiveX Control

For communication from a PC to the RMC

Tip:
RMCLink has it's own help. After installing RMCLink, you will find it on the Windows Start menu > All Programs > RMCLink Component > RMCLink Documentation. The **How to...** section is very helpful.

Communicate with any RMC from a Custom Application

The RMCLink component enables direct communication with any of Delta Computer System's RMC family of motion controllers from numerous programming languages and applications. Supporting serial RS-232 and Ethernet communications, RMCLink provides full functionality to read and write registers, read bits, and issue commands to all RMC family controllers.

RMCLink comes with sample projects to help you get up and running quickly. The RMCLink help includes detailed walk-throughs and numerous code snippets.

RMCLink is available for free download on Delta's website: https://deltamotion.com/dloads/. The download includes a detailed help file and examples to get you started. After installing RMCLink, the help file will be accessible from the Windows Start button>>All Programs menu.

Supported Programming Languages and Applications

RMCLink can be used from numerous programming languages and applications. It has three interfaces to make it intuitive and easy to use from any language.

The table below lists supported programming languages and applications and the respective RMCLink interface that should be used for that language. All the interfaces are included in the RMCLink download.

Programming Languages and Applications	RMCLink Interface
Visual Basic 5.0/6.0 VBA (Microsoft Excel, Microsoft Word, etc.) VBScript JScript PHP Python MATLAB	RMCLink COM Component
Visual Basic .NET Visual C# Visual C++/CLI	RMCLink.Interop .NET Assembly
Visual C++	RMCLink C++ Wrapper Class

Note: Drivers for use with National Instruments LabVIEW are available separately.

Supported RMC Communication Ports

RMCLink can communicate via Ethernet or serial RS-232. The table below lists the ports on the RMCs that it can communicate with.

RMC Module	RMCLink Supported Ports
RMC75E	10/100 Ethernet port
RMC75S	RS-232 Monitor port
	Note: If the serial settings on the second RMC75S RS-232 port are identical to the fixed RS-232 Monitor port settings, RMCLink can communicate with that port.
RMC75P	RS-232 Monitor port
RMC150E	10/100 Enet port
RMC200	Enet1 and Enet2 ports
RMC100	ENET port on the RMC100-ENET module
	RS-232 Monitor port on the RMC100 CPU module

Note:
The RMC100 is not supported by RMCTools and is not documented in this help file. It appears here only to fully explain RMCLink. Notice that RMCLink also is not a part of RMCTools, nor is fully documented in this help file. RMCLink contains its own very detailed help.

Using RMCLink

To use RMCLink, download it from Delta's website and install it. Open the RMCLink Documentation. You will find it on Windows Start menu > All Programs>RMCLink Component. Determine which Interface you need to use, based on your programming language. The **How to...** section is very helpful.

RMC Addresses in RMCLink

The RMCLink documentation includes the address maps you will need to use with RMCLink to access registers in the RMCs. Notice the addresses used in RMCLink may be different from any given in the RMC software.

See Also

Communications Overview | Using Sockets to Access the RMC over Ethernet | Communicating Directly over UDP

6.10.17. Using Other Master Controllers with the RMC

The RMC motion controllers are compatible with a large number of devices, and therefore it is not possible to include specific instructions for all compatible master controllers. This topic provides general information on how to use other master controllers with the RMC. The RMC is always a slave.

If your PLC, HMI or other device does not support any of the methods listed here, please contact a Delta Computer Systems sales engineer to discuss your device. Delta strives to support all major devices, and is interested in knowing about devices that the RMC does not support.

Ethernet

Communicating with the RMC via Ethernet from other devices will use one of the following four methods. Review each to determine which is appropriate for your device:

- **PLC Ethernet Emulation**
  The RMC responds to several common industrial Ethernet protocols and can *emulate,* or act like, several common PLCs. If your device supports reading and writing to registers in any of these PLCs, then your device should be able to communicate with the RMC. See Using the RMC's PLC Ethernet Emulation for details.
- **Standard Industrial Ethernet Protocols**
  The RMC75E, RMC150E, and RMC200 controllers support several common industrial Ethernet protocols, including Modbus/TCP, PROFINET and EtherNet/IP. If your device can make requests in any of the RMC's supported industrial Ethernet protocols, then it can likely communicate with the RMC. See the Supported Ethernet Protocols section in the Ethernet Overview topic for details.
- **.NET Assembly and ActiveX Control**
  Applications that are running on a Windows PC and that support ActiveX Controls, .NET Assemblies, or Microsoft Component Object Model (COM) can use Delta's RMCLink .NET Assembly and ActiveX Control. See the RMCLink topic for details.
- **Direct over TCP or UDP**
  Third-party or custom controllers that do not support any of the above three methods but can send and receive either TCP or UDP packets directly can communicate with the RMC by manually building and parsing packets over TCP or UDP. See the Communicating Directly over TCP and Communicating Directly over UDP topics for details. Delta has examples for certain programming languages, such as C.

Serial RS-232 and RS-485

The RMC75S supports serial RS-232 and RS-485. The RMC150 does not support serial communication, but can communicate with serial devices via serial-to Ethernet converters sold by other manufacturers. Contact Delta for more information.

Communicating with the RMC from serial devices will use one of the following methods. Review each to determine which is appropriate for your device:

- **Standard Industrial Ethernet Protocols**
  The RMC75S supports the several industrial serial protocols, including Modbus RTU, DF1, and the Mitsubishi Bidirectional Protocol. For details, see each respective topic.
- **.NET Assembly or ActiveX Control**
  Applications that are running on a Windows PC and that support ActiveX Controls, .NET Assemblies, or Microsoft Component Object Model (COM) can use Delta's RMCLink .NET Assembly and ActiveX Control. See the RMCLink topic for details.

PROFIBUS-DP

The RMC75P and RMC150 can communicate via PROFIBUS-DP. For details, see the PROFIBUS topic.

See Also

Communications Overview | Ethernet Overview | Serial Overview | PROFIBUS Overview

6.11. Address Maps

6.11.1. Address Maps

To open the Address Maps: Expand the desired controller in the Project pane and double-click **Address Maps.**

The Address Maps list all RMC register addresses and allow the user to map RMC registers to addresses for various communication protocols. Each individual register in the RMC can be addressed from external devices such as PLCs or HMIs via any of the protocol addresses listed in the Address Maps.

When to use the Address Maps

Use the Address Maps to:

1. **Set up the Indirect Data Map**
   Set up the Indirect Data Map to consolidate scattered registers to efficiently be communicated with an external device, such as a PLC. The Indirect Data Map is useful for configuring the individual items to be communicated. For large blocks of data, use the other address maps.
2. **Find RMC Register Addresses**
   Use the Address Maps to find the address of any register in the RMC for the communication protocol you are using. This applies to the Modbus, FINS, DF1, PROFINET and IEC maps. When setting up a PLC or HMI to communicate with the RMC, the Address Maps help find addresses to enter into the PLC or HMI.
   Registers can also be viewed, but not configured, in the register map help topics: RMC75 Register Map, RMC150 Register Map, and RMC200 Register Map.
3. **Configure Large Address Blocks**
   For the RMC200, the address maps for Modbus, FINS, DF1, and PROFINET provide some fixed addresses, with the remaining address areas configurable by the user. These maps are intended for addressing large blocks of data that can be accessed via a PLC or HMI. For smaller blocks of data, such as individual registers, use the Indirect Data Map instead. Each of the protocol address maps (Modbus, FINS, DF1, and PROFINET) include an address section for the Indirect Data Map.

Individual Address Maps

The individual Address Maps available are:

1. **Indirect Data Map**
   The Indirect Data Map allows scattered data items to be consolidated for efficient communications on all protocols, and is also used for setting up EtherNet/IP I/O, PROFINET, and PROFIBUS communication.

2. **Modbus Address Map**
   Lists all the Modbus addresses of RMC registers. The RMC75 and RMC150 have a fixed map that provides addresses for all registers. The RMC200 has an editable map, with some fixed addresses.
3. **FINS Address Map**
   Lists all the FINS addresses of RMC registers. The FINS protocol is used with Omron PLCs. The RMC75 and RMC150 have a fixed map that provides addresses for all registers. The RMC200 has an editable map, with some fixed addresses.
4. **DF1 Address Map**
   Lists all the DF1 addresses of RMC registers. The DF1 addressing is typically used for any device that can communicate with an Allen Bradley SLC 500, PLC-5, or MicroLogix PLC. The RMC75 and RMC150 have a fixed DF1 address map that provides addresses for all registers. The RMC200 has an editable map, with some preset addresses.
5. **PROFINET Data Records Address Map**
   For the RMC200, includes fixed and editable PROFINET data records. Data records are not required for PROFINET communication. See Using PROFINET Record Data for details.
6. **IEC Address Map**
   Lists the fixed internal addresses of the RMC registers. In addition, the following external communications use the same two-level numbering of the addresses as the IEC addresses:
   - DMCP
   - LabVIEW
   - Mitsubishi Procedure Exist Protocol
   - RMCLink

How to Use the Address Maps

Set up the Indirect Data Map

Use the Indirect Data Map to configure individual registers to be communicated. See Indirect Data Map for details.

Find RMC Register Addresses

To find the address of any register in the RMC for the communication protocol you are using:

1. In the Address Maps list, choose the desired protocol. Keep in mind that the IEC address map applies to several protocols as listed above.
2. Browse the list for the desired register and its address.
3. If you are using the Indirect Data Map, you can view the addresses in the Indirect Data Map for your protocol as follows:
   a. In the Indirect Data Map, right-click the **Reg #** column, point to **Address Format** and choose the desired protocol.
   b. The Reg # column will display the address of each item for your protocol.

Configure Large Address Blocks

For the RMC200, the Modbus, FINS, DF1, and PROFINET maps provide editable maps intended to be configured for large address blocks for the communication protocol you are using.

1. Choose the desired address map.
2. In the bottom row:
   a. In the **Start Address** column, enter the first address you wish to use. Make sure the address conforms to the syntax described in **Address Syntax** below.
   b. In the **Registers** column, enter the number of registers. This is the number of 32-bit registers in the RMC. Notice that this will map to twice as many 16-bit Modbus or FINS registers for these address maps.
   c. In the **Map To** column, enter the area in the RMC that this address range will apply to.

3. Repeat for additional address ranges. The rows will automatically re-order entries to keep them in numerical address order.
4. Click the Download button  to download the changes to the RMC.

Address Syntax

For user-configurable address maps, the addresses must follow the formats listed below. Address ranges as defined by the Start Address and End Address are not allowed to overlap.

Modbus

- Must be an odd-numbered 'holding register' address between 400001 and 465535.
- Holding registers start at 400001 and are 16-bit, so each 32-bit address starts at an odd address.

Example Addresses: 420001, 430201

FINS

- Must start with "D" or "E*n*_" where *n* is the number of the extended memory bank and must be in the range of 0-C (hexadecimal), and be followed by a 5-digit memory address.
- The memory address must be an even number between 0 and 32766 (FINS uses 16-bit addressing, so each 32-bit word starts at an even address number).

Example Addresses: D08400, E2_12000

DF1

- Must start with an 'F' (32-bit floating point registers) or an 'L' (32-bit integer), followed by *file* number, a colon (:), and the *element* number.
- The *file* number may be 0-4095, although most PLC's support only 7-255.
- The *element* number may be 0-4095, although most PLC's support only 0-255.

Example Addresses: F40:0, L112:100

PROFINET Data Records

- Data record index number, period (.), and the register offset number.
- Record index may be between 4 and 32767.
- Register offset must *always* be zero (0).

Example Addresses: 4.0 (index 4, offset 0), 100.0 (index 100, offset 0)

See Also

Indirect Data Map Editor | Indirect Data Map | Modbus Address Map | FINS Address Map | DF1 Address Map | PROFINET Data Records Address Map | IEC Address Map

6.11.2. Indirect Data Map

The Indirect Data Map maps data registers from anywhere in the RMC to a single area. This allows scattered data items to be consolidated for efficient communications. Instead of reading and writing to many locations, a single read and single write can be made.

Example:

Perhaps you wish to read 5 registers in the RMC75E via the Allen-Bradley CSP protocol (DF1 over Ethernet). These registers may have addresses F8:0, F8:8, F9:0, F9:8, and F56:0. Reading these would require several reads, and would not be very efficient. By using the Indirect Data Map, you can assign these registers to F18:0 to F18:4. Now only one read of 5 registers is required, which is much more efficient.

The Indirect Data Map is used for EtherNet/IP I/O, PROFINET, and PROFIBUS. See those topics for details on how the Indirect Data Map is used with those protocols.

When to Use the Indirect Data Map

The Indirect Data Map can be used for all protocol address types supported by the RMC. The Indirect Data Map is particularly useful for data that is frequently communicated, such as RMC status being read by a PLC. Here are particular details on when it must be used or isn't necessary.

PLCs

In general, anytime a PLC is communicating with the RMC, the Indirect Data Map should be used in order to achieve efficient communications. It is especially useful for data that is continuously transferred.

HMIs

When communicating with the RMC from an HMI, the Indirect Data Map is usually unnecessary because the HMI can address the RMC registers directly, and HMI communications are often not efficient to begin with.

Using the Indirect Data Map

The Indirect Data registers can be mapped to any registers in the RMC. Thereby, the values from the mapped registers in the RMC can be read from and written to by writing to and reading from the Indirect Data registers.

To set up the Indirect Data Map:

1. Open the Address Maps editor and select the Indirect Data Map.
2. In the **Map To** column of an Indirect Data Map row, click the ellipsis button to browse to the RMC register you want to map. You can map as many of the Indirect Data Map entries as you would like.
3. Click the Download button to download the changes to the RMC.
4. Now, with your PLC or other host controller, you can read or write your data directly from the Indirect Data registers. Notice that if a mapped register is read-only, you cannot write to it.

Note:
Do not write to registers in the Indirect Data Map that have not been mapped. This may unnecessarily fill up the Event Log with errors.

Example

The user would like to put some of the RMC75 registers in the Indirect Data Map so that the communications can be more efficient. The user decides to put the following registers in the Indirect Data Map:

- Axis 0 Status Bits
- Axis 0 Error Bits
- Axis 0 Actual Position
- Axis 0 Actual Pressure
- Axis 1 Status Bits
- Axis 1 Error Bits
- Axis 1 Actual Position

- Axis 1 Actual Pressure

The user then sets up the Indirect Data Map as shown below:

	Reg #	Map To	Description	Current
0	F18:0	%MD8.0	Axis0 Status Bits	N/A
1	F18:1	%MD8.1	Axis0 Error Bits	N/A
2	F18:2	%MD8.8	Axis0 Actual Position (pu)	N/A
3	F18:3	%MD8.23	Axis0 Actual Pressure (Pr)	N/A
4	F18:4	%MD9.0	Axis1 Status Bits	N/A
5	F18:5	%MD9.1	Axis1 Error Bits	N/A
6	F18:6	%MD9.8	Axis1 Actual Position (pu)	N/A
7	F18:7	%MD9.23 ...	Axis1 Actual Pressure (Pr)	N/A

To read the data from the mapped registers, read from F18:0 to F18:7. Now, instead of reading 8 registers in various locations, the user can read 8 registers in one block. Notice that if a mapped register is read-only, you cannot write to it.

Choosing the Address Format

The addresses of the registers in the Indirect Data Map are displayed in the Reg # column. To change the address format, right-click any cell in the Reg # column, choose **Address Formats**, and choose the desired format.

Advanced Details

Structure of Indirect Data Map Registers

The Indirect Data Map consists of two arrays of values. The first array is called the Indirect Data Map Definition, which holds register addresses that each entry in the second array (Indirect Data) represents. The Indirect Data Map Definition can be used for both readable and writable registers.

Note:
During normal communications with the RMC, it is the Indirect Data registers that should be written or read, *not* the Indirect Data Definition registers.

RMC75		RMC150		RMC200	
Indirect Data Map Definition	**Indirect Data Map**	**Indirect Data Definition**	**Indirect Data Map**	**Indirect Data**	**Indirect Data Map Definition**
%MD17.0	%MD18.0	%MD41.0	%MD42.0	%MD8.0	%MD12.0
%MD17.1	%MD18.1	%MD41.1	%MD42.1	%MD8.1	%MD12.1
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
%MD17.63	%MD18.63	%MD41.255	%MD42.255	%MD8.255	%MD12.255

Changing the Indirect Data from a PLC

Typically, the Indirect Data Map is set up in RMCTools. However, you can change the Indirect Data Map during runtime to map different registers. To do this, write to the Indirect Data Map Definition registers. The Indirect Data Map Definition registers contain the addresses of the

mapped registers. When writing addresses to the Indirect Data Map Definition registers, those addresses must be entered as an integer value as described here:

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

See Also

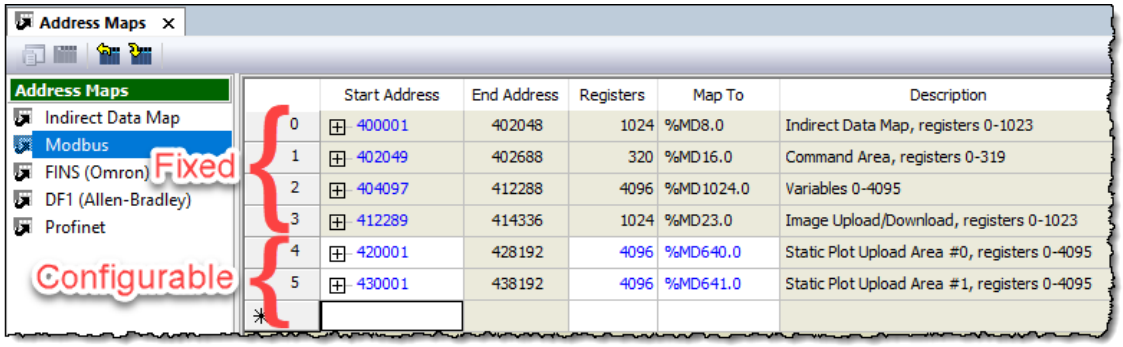
Communications Overview | Indirect Data Map Editor

6.11.3. Modbus Address Map

To access this map:
Expand the desired controller in the Project pane, double-click **Address Maps**, then choose **Modbus**.

The Modbus Address Map lists the Modbus addresses of all the externally-accessible registers in the RMC.

Browse the map to find the Modbus address of any externally-accessible RMC register. If you are using an RMC200, you can add configurable address ranges to the Modbus Address Map.



Using the Configurable Addresses

On the RMC200, use the configurable addresses if you need to access large blocks of RMC registers via Modbus, and those registers are not already included in the fixed addresses (see Modbus Addressing) and do not fit in the Indirect Data Map.

If you need access to a few individual registers, use the Indirect Data Map instead. You need not configure Modbus addresses if you use the Indirect Data Map, since the Indirect Data Map already has Modbus addresses.

Adding Modbus Address Ranges

For the RMC200, you can add up to 128 configurable address blocks to the fixed address blocks.

1. In the bottom row, in the **Start Address** column, enter the first Modbus address you wish to use. Make sure the address conforms to the syntax described in **Address Syntax** below.

2. In the **Registers** column, enter the number of registers. This is the number of 32-bit registers in the RMC. Notice that this will map to twice as many 16-bit Modbus registers.
3. In the **Map To** column, enter the area in the RMC that this address range will apply to.
4. Repeat for additional address ranges. The rows will automatically re-order entries to keep them in numerical address order.
5. Click the Download button to download the changes to the RMC.

Modbus Address Syntax

Modbus addresses must conform to the following formats listed below. Address ranges as defined by the Start Address and End Address are not allowed to overlap.

- Must be on odd-numbered 'holding register' address between 400001 and 465535.
- Holding registers start at 400001 and are 16-bit, so each 32-bit address starts at an odd address.

Example Addresses: 420001, 430201

See Also

Modbus/TCP | Modbus RTU | Modbus Addressing | Address Maps | Indirect Data Map

6.11.4. FINS Address Map

To access this map:
Expand the desired controller in the Project pane, double-click **Address Maps**, then choose **FINS**.

The FINS Address Map lists the FINS Addresses of all the externally-accessible registers in the RMC.

Browse the map to find the FINS address of any externally-accessible RMC register. If you are using an RMC200, you can add configurable address ranges to the FINS Address Map.

Using the Configurable Addresses

On the RMC200, use the configurable addresses if you need to access large blocks of RMC registers via FINS, and those registers are not already included in the fixed addresses (see FINS Addressing) and do not fit in the Indirect Data Map.

If you need access to a few individual registers, use the Indirect Data Map instead. You need not configure FINS addresses if you use the Indirect Data Map, since the Indirect Data Map already has FINS addresses.

Adding FINS Address Ranges

For the RMC200, you can add up to 128 configurable address blocks to the fixed address blocks.

1. In the bottom row, in the **Start Address** column, enter the first FINS address you wish to use. Make sure the address conforms to the syntax described in **Address Syntax** below.
2. In the **Registers** column, enter the number of registers. This is the number of 32-bit registers in the RMC. Notice that this will map to twice as many 16-bit FINS registers.
3. In the **Map To** column, enter the area in the RMC that this address range will apply to.
4. Repeat for additional address ranges. The rows will automatically re-order entries to keep them in numerical address order.
5. Click the Download button to download the changes to the RMC.

FINS Address Syntax

FINS addresses must conform to the following formats listed below. Address ranges as defined by the Start Address and End Address are not allowed to overlap.

- Must start with "D" or "E*n*_" where *n* is the number of the extended memory bank and must be in the range of 0-C (hexadecimal), and be followed by a 5-digit memory address.
- The memory address must be an even number between 0 and 32766 (FINS uses 16-bit addressing, so each 32-bit word starts at an even address number).

Example Addresses: D08400, E2_12000

See Also

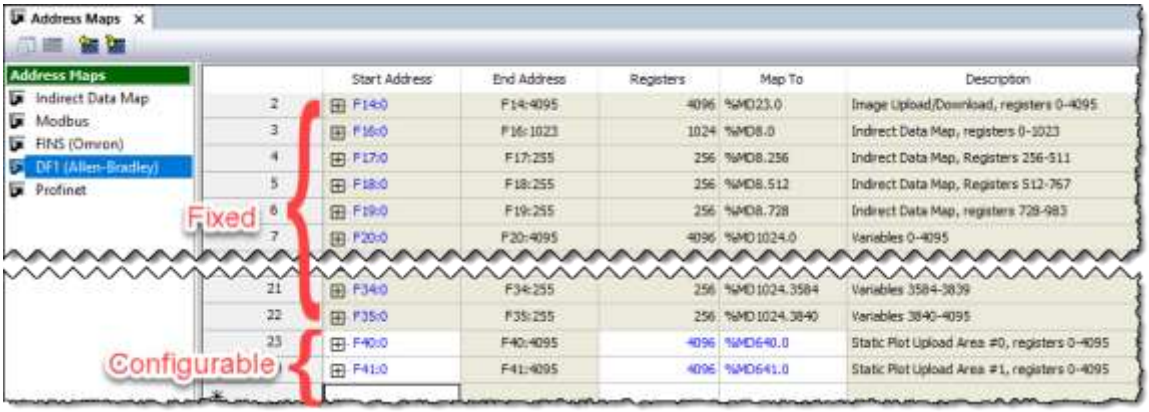
FINS/UDP | FINS Addressing | Address Maps | Indirect Data Map

6.11.5. DF1 Address Map

To access this map:
Expand the desired controller in the Project pane, double-click **Address Maps**, then choose **DF1**.

The DF1 Address Map lists the DF1 Addresses of all the externally-accessible registers in the RMC.

Browse the address map to find the DF1 address of any externally-accessible RMC register. If you are using an RMC200, you can add configurable address ranges to the DF1 Address Map. The DF1 addresses are all listed as F registers, but each register can also be accessed by external devices as L registers.



Using the Configurable Addresses

On the RMC200, use the configurable addresses if you need to access large blocks of RMC registers via DF1, and those registers are not already included in the fixed addresses (see DF1 Addressing) and do not fit in the Indirect Data Map.

If you need access to a few individual registers, use the Indirect Data Map instead. You need not configure DF1 addresses if you use the Indirect Data Map, since the Indirect Data Map already has DF1 addresses.

Adding DF1 Address Ranges

For the RMC200, you can add up to 128 configurable address blocks to the fixed address blocks.

1. In the bottom row, in the **Start Address** column, enter the first DF1 address you wish to use. Make sure the address conforms to the syntax described in **Address Syntax** below.
2. In the **Registers** column, enter the number of registers. This is the number of 32-bit registers in the RMC.
3. In the **Map To** column, enter the area in the RMC that this address range will apply to.
4. Repeat for additional address ranges. The rows will automatically re-order entries to keep them in numerical address order.
5. Click the Download button  to download the changes to the RMC.

DF1 Address Syntax

DF1 addresses must conform to the following formats listed below. Address ranges as defined by the Start Address and End Address are not allowed to overlap.

- Must start with an 'F' (32-bit floating point registers) or an 'L' (32-bit integer), followed by *file* number, a colon (:), and the *element* number.
- The *file* number may be 0-4095, although most PLC's support only 7-255.
- The *element* number may be 0-4095, although most PLC's support only 0-255.

Example Addresses: F40:0, L112:100

See Also

CSP | DF1 Addressing | Address Maps | Indirect Data Map

6.11.6. PROFINET Data Records Address Map

To access this map:
Expand the desired controller in the Project pane, double-click **Address Maps**, then choose **PROFINET Data Records**.

Note: The PROFINET Data Records Address Map is for the RMC200 only. The RMC75 and RMC150 Custom Data Records are configured in the PROFINET Settings Page.

The PROFINET Data Records Address Map lists the PROFINET Data Records addresses as they apply to registers in the RMC200. The RMC200 data records 0-3 are pre-defined and you can configure up to 128 additional data records. The records limit you to read and write starting at the *beginning* of each defined range.

Using the Configurable Addresses

On the RMC200, use the configurable addresses if you need to access large blocks of RMC registers via PROFINET Data Records, and those registers are not already included in the fixed addresses (see PROFINET Record Data Addressing), and do not fit in the Indirect Data Map.

If you need access to a few individual registers, use the Indirect Data Map instead. You need not configure PROFINET Data Records addresses if you use the Indirect Data Map, since the Indirect Data Map already has PROFINET Data Records addresses.

Adding Data Records

For the RMC200, you can add up to 128 data records in addition to the fixed records:

1. In the bottom row, in the **Start Address** column, enter the data record number you wish to use. Make sure the address conforms to the syntax described in **Address Syntax** below.
2. In the **Registers** column, enter the number of registers. This is the number of 32-bit registers in the RMC.
3. In the **Map To** column, enter the area in the RMC that this address range will apply to.
4. Repeat for additional address ranges. The rows will automatically re-order entries to keep them in numerical address order.
5. Click the Download button to download the changes to the RMC.

Data Record Address Syntax

Data Record addresses must conform to the following formats listed below. Address ranges as defined by the Start Address and End Address are not allowed to overlap.

- Data record index number, period (.), and the register offset number.
- Record index may be between 4 and 32767.
- Register offset must *always* be zero (0).

Example Addresses:

4.0 (index 4, offset 0)

100.0 (index 100, offset 0)

See Also

Using PROFINET Record Data | Address Maps

6.11.7. IEC Address Map

To access this map:
Expand the desired controller in the Project pane, double-click **Address Maps**, then choose **IEC**.

The IEC Address Map lists the IEC addresses of all the externally-accessible registers in the RMC. The two-level IEC address numbers are used for the following communications types:

- DMCP
- LabVIEW
- Mitsubishi Procedure Exist Protocol
- RMCLink

Browse the map to find the IEC address of any externally-accessible RMC register.

See Also

IEC Addressing | Address Maps

7. Hardware

7.1. RMC Hardware Overview

The RMC75, RMC150, and RMC200 motion controllers are compatible with the RMCTools software and share the same setup, programming, and tuning procedures.

	RMC75E	RMC75S	RMC75P	RMC150E	RMC200 Lite	RMC200 Standard
Max Physical Control Axes	2	2	2	8	18	50
Max Total Axes	4	4	4	16	48	128
Max User Tasks	4	4	4	10	32	64
Communications						
Ethernet	✓					
Serial RS232/485						
PROFIBUS						
Loop Times	250 µs to 4 ms	500 µs to 4 ms	500 µs to 4 ms	250 µs to 4 ms	125 µs to 4 ms	125 µs to 4 ms
RMCTools Connection	USB, Ethernet	RS-232	RS-232	USB, Ethernet	USB, Ethernet	USB, Ethernet
Feedback Options						
SSI		✓	✓	✓	✓	✓
Start/Stop, PWM	✓	✓	✓	✓	✓	✓
Quadrature (A,B,Z)	✓	✓	✓	✓	✓	✓
Analog (±10V, 4-20mA)	✓	✓	✓	✓	✓	✓
Resolver				✓		
Control Output						
±10 V	✓					
4-20 mA, ±20 mA						
Max RAM (for plots, curves)						
Plots	48 MB	128 KB	128 KB	48 MB	48 MB	192 MB

Curves	8 MB	256 KB	256 KB	8 MB	16 MB	64 MB
Flash Memory (for programming, curves)	1024 KB	96 KB	96 KB	1024 KB	6016 KB	6016 KB
Feature Key						
Display/Keypad						
Other						
Real-time clock						
Discrete I/O						
Class I, Div 2						

Accessories

VC2124

2-channel converter from the RMC's +/-10V output to current for current-driven valves. Each channel supplies up to +/-100 mA. Channels can be paralleled for more current.

RMC150 Quad Module Cable

Cable with a DB25 connector on one end and pigtail ends for the Quad module.

See Also

RMC75 Overview | RMC150 Overview | RMC200 Overview

7.2. RMC75

7.2.1. RMC75 Hardware Overview

The RMC70 motion controller series offers three RMC75 CPU's for one- and two-axis systems. A number of feedback options are available for a wide variety of hydraulic, electric, and pneumatic position and position–pressure or position-force applications.

See also Controller Features and RMC75 Part Numbering.

An RMC75 motion controller consists of the **CPU module**, **Axis module**, and optional **Expansion modules**. The CPU module and Axis module make up the **base module**. The base module is always shipped as one unit.

CPU Module **Axis Module** **Expansion Module**

CPU Modules

The CPU modules include the main motion control processing unit, the communication channel such as serial, PROFIBUS, Ethernet, etc. and a monitor port or USB port for communications to the RMCTools software.

CPU Module	Description	RMCTools Monitor Port
RMC75E	Controller with Ethernet Communications	USB
RMC75S	Controller with Serial Communications	RS-232
RMC75P	Controller with PROFIBUS-DP Communications	RS-232

Comparing the RMC75 CPU Modules

In addition to the communication port and RMCTools monitor port, the RMC75 CPUs differ in the following ways:

- **Monitor Port Speeds**
  The RMC75E USB port provides a much faster connection to RMCTools, with much faster plot upload speeds, than does the serial port on the RMC75S and RMC75P.
- **Retentive Variables**
  The RMC75E variables can be set to retentive, meaning that the Current Value will be retained between power cycles without requiring a Flash update.
- **Memory Size**
  The 75E has significantly more memory for plots, programs, curves. See Plot Overview, Program Capacity and Time Usage and Curve Storage Capacity for details.
- **Loop Times**
  75E supports faster loop times.
- **Internal Processor**
  The RMC75S and RMC75P internal processors are 32-bit. The RMC75E has some 64-bit processing capabilities. In practice, the only difference the user will experience due to this is that some mathematical calculations will produce slightly different results.

Axis Modules

The axis modules have one input per axis for interfacing to transducers and one Control Output per axis for interfacing to an actuator. Axis modules are either one or two axes. Each axis module interfaces to a different type of transducers.

Axis Module	Description
AA1	1-Axis Analog Voltage or Current Input, Analog Control Output
AA2	2-Axis Analog Voltage or Current Inputs, Analog Control Outputs
MA1	1-Axis MDT and SSI Input, Analog Control Output
MA2	2-Axis MDT and SSI Inputs, Analog Control Outputs
QA1	1-Axis Quadrature Encoder Input (5 V differential), Analog Control Output
QA2	2-Axis Quadrature Encoder Inputs, (5 V differential), Analog Control Outputs

Optional Expansion Modules

Up to four optional expansion modules can be added to enhance the capabilities of the RMC75. These modules can be field-added. Expansion modules can be used for reference axes, pressure control, or discrete I/O.

Exp Module	Description
D8	8 Individually Configurable Discrete I/O, 12-24 VDC
A2	2 Analog inputs (±10 V or 4-20 mA)
AP2	2 Analog inputs (±10 V or 4-20 mA)for Position-Pressure or Position-Force Control
Q1	1 Quadrature input (5 V differential) - max 2 Q1 modules per RMC75

Common Specifications

These are the general specifications for the RMC75 motion controllers. For specifications on individual modules, refer to each module.

See also RMC75 Mounting.

Motion Control	
Control loop time	User-selectable 0.5ms, 1ms, 2ms, or 4ms
Maximum speed	Unlimited
Power	
Voltage	+24 VDC +/-10%
Current – Base only	Typical 290 mA @ 24 VDC, max 375 mA
Current (4 Expansion modules)	Typical 385 mA @ 24 VDC, max 500 mA
DC-DC converter isolation	500 VAC, 700 VDC, input to controller
Mechanical	
Mounting	Symmetrical DIN 3 or panel-mount
Dimensions – Base units	3.25 x 5.0 x 2.75 in (8.3 x 12.7 x 7 cm) (WxHxD)
with 4 Expansion modules	Varies x 5.0 x 2.5 in (?? x 12.7 x 6.4 cm) (WxHxD)

Weight	12 ounces (0.37 kg) max
with 4 Expansion modules	Varies, 2.0 lb (0.9 kg) max
Environment	
Operating temperature	+32 to +140°F (0 to +60°C)
Storage temperature	-40 to +185°F (-40 to +85°C)
Humidity	95% non-condensing
Agency compliance	UL, CUL, CE: RMC75S, RMC75P, RMC75E

See Also

RMC75 Part Numbering | RMC150 Overview | Mounting and Wiring

7.2.2. RMC75 Part Numbering

Specify RMC75 part numbers when ordering and when contacting Delta customer support.

Part Numbering Schema

Base Unit		Optional Expansion Modules*
RMC7*NXXX*	**- *XXN***	**EXP70-*XXXN***
Choices	**Choices**	**Choices**
RMC75E RMC75S RMC75P	AA1 AA2 MA1 MA2 QA1 QA2	A2 AP2 D8 Q1

*Up to four optional expansion modules can be added to enhance the capabilities of the RMC75. These modules can be field-added. Expansion modules can be used for reference axes, pressure control, or discrete I/O.

Example Part Numbers

RMC75E-AA2

RMC75S-MA2

EXP70-D8

EXP70-Q1

Definitions

Module	Description
CPUs	
RMC75E	Controller with Ethernet Communications
RMC75S	Controller with Serial Communications

RMC75P	Controller with PROFIBUS-DP Communications
Axis Modules	
AA1	1-Axis Analog Voltage or Current Input, Analog Control Output
AA2	2-Axis Analog Voltage or Current Inputs, Analog Control Outputs
MA1	1-Axis MDT and SSI Input, Analog Control Output
MA2	2-Axis MDT and SSI Inputs, Analog Control Outputs
QA1	1-Axis Quadrature Encoder Input, Analog Control Output
QA2	2-Axis Quadrature Encoder Inputs, Analog Control Outputs
Expansion Modules	
D8	8 Individually Configurable Discrete I/O (Inputs and Outputs)
A2	2 Analog inputs
AP2	2 Analog inputs for Position-Pressure or Position-Force Control
Q1	1 Quadrature input (max 2 Q1 modules per RMC75)

See Also
RMC75 Overview

7.2.3. CPU Modules

7.2.3.1. RMC75 CPU Modules Overview

A CPU module contains the processing unit, the communication channel such as serial, PROFIBUS, Ethernet, etc. and a monitor port or USB port for communications to the RMCTools software. The CPU module, together with an axis module attached to the front of the CPU Module, make up a complete motion controller, called a base module. Optional expansion modules can be added to the right of the motion controller.

The front panel of the CPU contains status LEDs and connectors for power and communications.

CPU Module	Description	RMCTools Monitor Port
RMC75E	Controller with Ethernet Communications	USB
RMC75S	Controller with Serial Communications	RS-232
RMC75P	Controller with PROFIBUS-DP Communications	RS-232

Comparing the RMC75 CPU Modules

In addition to the communication port and RMCTools monitor port, the RMC75 CPUs differ in the following ways:

- **Monitor Port Speeds**
  The RMC75E USB port provides a much faster connection to RMCTools, with much faster plot upload speeds, than does the serial port on the RMC75S and RMC75P.
- **Retentive Variables**
  The RMC75E variables can be set to retentive, meaning that the Current Value will be retained between power cycles without requiring a Flash update.
- **Memory Size**
  The 75E has significantly more memory for plots, programs, curves. See Plot Overview, Program Capacity and Time Usage and Curve Storage Capacity for details.
- **Loop Times**
  75E supports faster loop times.
- **Internal Processor**
  The RMC75S and RMC75P internal processors are 32-bit. The RMC75E has some 64-bit processing capabilities. In practice, the only difference the user will experience due to this is that some mathematical calculations will produce slightly different results.

See Also

RMC75 Modules Overview | RMC75 Part Numbering | RMC150E/RMC151E CPU Modules

7.2.3.2. RMC75E CPU Module

The RMC75E CPU module provides the processing power of the RMC75 motion controllers with Ethernet communications. The RMC75E has a USB Monitor port for convenient connection to a PC running RMCTools (the RMC75S and RMC75P use an RS-232 serial port for the same purpose). The RMC75E supports auto-negotiation for 10/100Mbps and full/half duplex and also supports auto-crossover.

Features

- One 10/100 Mb/s Ethernet port
- One USB Monitor Port
- Supported Ethernet Protocols
  - EtherNet/IP
  - PROFINET
  - Modbus/TCP
  - CSP (also called DF1 over Ethernet)
  - FINS/UDP (Omron)
  - Procedure Exist (Mitsubishi)
  - Delta Motion Control Protocol

Part Number

The part number of the RMC75E is **RMC75E**. Like all RMC75 CPU modules, the RMC75E can only be ordered with an axis module.

For example, RMC75E-AA1 is an RMC75E with a one-axis AA1 module.

Specifications

General	
Weight	236 g + 16 g (connector)
Ethernet Interface	
Hardware Interface	IEEE 802.3 for 100BASE-T (twisted pair)
Data Rate	10/100 Mbs
Duplex	Full/Half Duplex
Features	Auto-negotiation, Auto-crossover (MDI/MDI-X)
Connector	RJ-45
Cable	CAT5, CAT5e or CAT6, UTP or STP
Configuration Parameters	IP address, subnet mask, gateway address, enable/disable autonegotiation
Configuration methods	BOOTP, DHCP, or static
Ethernet Protocols	
Application protocols	EtherNet/IP, PROFINET, Modbus/TCP, CSP (DF1 over Ethernet), FINS (Omron) Procedure Exist (Mitsubishi), DMCP (Delta Motion Control Protocol)
Framing Protocol	Ethernet II
Internet Protocol IP	IP (includes ICMP, ARP, and Address Collision Detection)
Transport Protocols	TCP, UDP
Other protocols	LLDP
USB Monitor Port Interface	
Connector	USB "B" receptacle
Data Rate	Full-speed (12 Mbps)

LEDs

Controller LED

This bi-color (red/green) LED has the following states:

State	Description
Steady Off	No power.
Steady Green	RUN Mode
Flashing Green (Slow)	PROGRAM Mode
Flashing Green (Fast)	Updating Flash or a controller restart is pending.
Flashing Red (Uniform on/off)	The RMC is in the loader. This should occur briefly when the controller is powered up. If this occurs during normal operation, an error has occurred in the controller and it must be reset. Cycle power to the RMC75 to reset it. **Causes:**

	○ **Power Interruption** In certain cases, a power interruption to the RMC may cause it to go into the loader. Even a very short interruption, such as 20 msec, can cause problems. This type of interruption cannot be measured with a standard multimeter. For accurate power supply verification, use an oscilloscope or similar high-speed monitoring device. ○ **Firmware Bug** Certain firmware bugs will cause the RMC to go into the loader. If this occurs, and power interruptions have been ruled out, contact Delta Technical support.
Flashing Red (Pattern)	The LED will flash 1, 2, or 3 times, pause and then repeat the pattern. All of these patterns indicate that the RMC failed a self-test of the RAM subsystem during startup. This is a non-recoverable major fault. The module will likely need to be shipped back to Delta for repair.
Steady Red	Non-Recoverable major fault. That is, the module cannot boot at all, not even into the loader. The module will likely need to be shipped back to Delta for repair. Some things to try to get it into the Loader (to be done at the direction of technical support) include removing all expansion and axis modules, and setting the Force to Loader jumper under the axis module.

Communication LEDs

These LEDS are located below the Controller LED on a gray background. These LEDs do not convey 10/100, FDX/HDX, or Collision information. The user will have to rely on their switch's LEDs or the RMCTools Communication Statistics to determine the state of those affairs.

Link/Act LED

The Link/Act LED reflects the status of the physical Ethernet connection between the RMC and the device on the other end of the Ethernet cable. If the Link LED is not on or is not blinking when you expect it to be, see the Ethernet Link/Act LED topic for troubleshooting information.

State	Description
Off	The Ethernet link is down
Flashing Green	The Ethernet link is up, with activity
Steady Green	The Ethernet link is up, no activity

Net LED

State	Description
Steady Off	No power, or no IP address configured.
Steady Red	Duplicate IP Address detected.
Flashing Red	IP address configured and in use, and a controlling EtherNet/IP I/O connection has timed out and not been re-established.
Flashing Green	IP address configured and in use, no EtherNet/IP or PROFINET connections are currently established, and there is no timed-out controlling EtherNet/IP I/O connection.
Steady Green	IP address configured and in use, at least one EtherNet/IP or PROFINET connection is

	currently established, and there is no timed-out controlling EtherNet/IP I/O connection.

Note:
The Monitor Port does not affect any LEDs on the RMC75.

Startup LED Test

When the RMC75E powers up, the LEDs listed below go through these states, with a 250 msec delay between each step:

CPU LED	Net LED
Green	Off
Red	Off
Green	Green
Green	Red
Green	Off

Prior to the LED test, the module will run through the loader, during which time the CPU LED will be solid red. After the LED test, the firmware will continue initialization, after which time the LEDs resume their normal behaviors.

See Also

RMC75E Wiring | Ethernet Overview | Communication Statistics | CPU Modules Overview (RMC75)

7.2.3.3. RMC75S CPU Module

The RMC75S combines multiple communication protocols with RS-232 and RS-485 transceiver options to form a versatile and industrial-hardened communication platform. The RS-232 option provides full-duplex point-to-point communications, while RS-485 allows half-duplex multi-drop networking with up to 128 RMC75s.

Due to limited throughput of serial communications, the RMC75S is best suited for applications where time-critical machine control functions related to motion are implemented in the RMC75S using the User Programs. The serial communications is fine for low-bandwidth monitoring or modifications to the RMC75S parameters or User Programs. In applications where higher throughput is necessary, consider the RMC75E Ethernet module.

Features

- One serial RS-232 or RS-485 Communications port
- One serial RS-232 Monitor Port
- Supported Protocols
  - Allen-Bradley DF1 (Full- and Half-duplex)
  - Modbus/RTU
  - Mitsubishi Bidirectional Protocol

Part Number

The part numbers of the RMC75S is **RMC75S**. Like all RMC75 CPU modules, the RMC75S can only be ordered with an axis module.

For example, RMC75S-QA2 is an RMC75S with a two-axis QA2 module.

Specifications

General	
Weight	245 g + 16 g (connector)
Serial Interface	
Transceivers	User-selectable: RS-232, RS-485
Baud Rates	9600, 19200, 38400, 57600, 115200 baud
Stop Bits	1 or 2
Data Length	8 bits
Parity	None, Odd, or Even
Isolation	500 VAC
Electrostatic Discharge (ESD) Protection	15 kV
RS-232 Interface	
Type	Single-Ended
Connector	DB-9 Male
Communication Distance	50 ft (12 m)
Network Type	Point-to-Point
RS-485 Interface	
Type	Differential
Connector	5-pin Terminal Block
Communication Distance	4000 ft (1200 m)
Network Type	RS-485: Point-to-Point or Multi-drop up to 128 nodes
RS-485 Input Impedance	48 kΩ (1/4 unit load)
Biasing	User selectable
Termination	120 Ω user selectable
RS-232 Monitor Port	
Connector	DB-9 Male
Cable	Null modem
Protocol	Allen-Bradley DF1 Full-Duplex, with CRC error detection
Settings	38400 baud, 8 data bits, no parity, 1 stop bit, no handshaking

LEDs

Controller LED

This bi-color (red/green) LED has the following states:

State	Description
Steady Off	No power.
Steady Green	RUN Mode
Flashing Green (Slow)	PROGRAM Mode

Flashing Green (Fast)	Updating Flash or a controller restart is pending.
Flashing Red	The device is in the loader. This should occur briefly when the controller is powered up. If this occurs during normal operation, an error has occurred in the controller and it must be reset. Cycle power to the RMC75 to reset it. **Causes:** ○ **Power Interruption** In certain cases, a power interruption to the RMC may cause it to go into the loader. Even a very short interruption, such as 20 msec, can cause problems. This type of interruption cannot be measured with a standard multimeter. For accurate power supply verification, use an oscilloscope or similar high-speed monitoring device. ○ **Firmware Bug** Certain firmware bugs will cause the RMC to go into the loader. If this occurs, and power interruptions have been ruled out, contact Delta Technical support.
Steady Red	Non-Recoverable major fault. That is, the module cannot boot at all, not even into the loader. The module will likely need to be shipped back to Delta for repair. Some things to try to get it into the Loader (to be done at the direction of technical support) include removing all expansion and axis modules, and setting the Force to Loader jumper under the axis module.

Communication LEDs

These LEDS are located below the Controller LED on a gray background.

Transmit LED

This LED reflects when data is being transmitted on the second serial port (RS-232/485).

State	Description
Steady Off	No power or no data being transmitted.
Steady or Flickering Green	Data is being transmitted.

Receive LED

This LED reflects when data is being received on the second serial port (RS-232/485).

State	Description
Steady Off	No power or no data being received.
Steady or Flickering Green	Data is being received.

Note:
The Monitor Port does not affect any LEDs on the RMC75.

See Also

RMC75S Wiring | Serial Overview | CPU Modules Overview (RMC75)

Show All

7.2.3.4. RMC75P CPU Module

PROFIBUS is a vendor-independent, open fieldbus standard for a wide range of applications in manufacturing and factory automation. This high-speed fieldbus was designed especially for communicating between programmable controllers and distributed I/O such as the RMC75 motion controller.

Up to 126 nodes can be connected to a single network spanning up to 14km. The RMC75P module supports data rates up to 12Mbaud, permitting high-speed on-the-fly downloads of positions and parameters to the RMC module and high-speed uploads of motion profile and status information to the host controller. The RMC75P's PROFIBUS interface gives the flexibility of several operating modes. Select the mode that best fits your application and PROFIBUS master's capabilities.

Features

- One PROFIBUS port
- One serial RS-232 Monitor Port
- 2 rotary switches to set the PROFIBUS station address.

Part Number

The part numbers of the RMC75P is **RMC75P**. Like all RMC7 CPU modules, the RMC75P can only be ordered with an axis module.

For example, RMC75P-MA2 is an RMC75P with a two-axis MA2 module.

Specifications

General	
Weight	239 g + 7 g (connector)
PROFIBUS-DP Interface	
Data Rate	9.6 kbaud up to 12 Mbaud
Isolation	2500 VAC
Product Identifier Number	0x07E1
Features Supported	Sync Mode, Freeze Mode, Auto-baud rate detect
Valid Station Addresses	0-99 (set by rotary switches on faceplate)
Connector	Standard PROFIBUS-DP DB-9 (use termination in cable connectors as per PROFIBUS specification)
RS-232 Monitor Port	
Connector	DB-9 Male
Cable	Null modem
Protocol	Allen-Bradley DF1 Full-Duplex, with CRC error detection
Settings	38400 baud, 8 data bits, no parity, 1 stop bit, no handshaking

LEDs

Controller LED

This bi-color (red/green) LED has the following states:

State	Description
Steady Off	No power.

Steady Green	RUN Mode
Flashing Green (Slow)	PROGRAM Mode
Flashing Green (Fast)	Updating Flash or a controller restart is pending.
Flashing Red	The device is in the loader. This should occur briefly when the controller is powered up. If this occurs during normal operation, an error has occurred in the controller and it must be reset. Cycle power to the RMC75 to reset it. **Causes:** ○ **Power Interruption** In certain cases, a power interruption to the RMC may cause it to go into the loader. Even a very short interruption, such as 20 msec, can cause problems. This type of interruption cannot be measured with a standard multimeter. For accurate power supply verification, use an oscilloscope or similar high-speed monitoring device. ○ **Firmware Bug** Certain firmware bugs will cause the RMC to go into the loader. If this occurs, and power interruptions have been ruled out, contact Delta Technical support.
Steady Red	Non-Recoverable major fault. That is, the module cannot boot at all, not even into the loader. The module will likely need to be shipped back to Delta for repair. Some things to try to get it into the Loader (to be done at the direction of technical support) include removing all expansion and axis modules, and setting the Force to Loader jumper under the axis module.

Net LED

The NET LED is located below the Controller LED on a gray background.

State	Description
Steady Off	The RMC's PROFIBUS channel is not communicating control data with the PROFIBUS master. See the Troubleshooting PROFIBUS topic for possible reasons.
Steady Green	The RMC's PROFIBUS channel is properly communicating control data with the PROFIBUS master. During normal operation, this LED should be steady with no flickering.

Note:
These are the only two LED states of the NET LED, but it is possible to have the Net LED flashing or flickering green, which indicates that the RMC75P is going on- and off-line on the PROFIBUS channel and generally indicates a network or configuration problem.

Note:
The Monitor Port does not affect any LEDs on the RMC75.

See Also

RMC75P Wiring | PROFIBUS Overview | CPU Modules Overview (RMC75)

7.2.4. Axis Modules

7.2.4.1. Axis Modules Overview

An axis module is the part of the RMC75 motion controller that interfaces to the transducers and drives. The axis module is mounted on the front of the RMC75 Control Module (see the RMC75 Modules Overview topic for an image). It has status LEDs and connectors for the axis wiring.

An axis module can have one or two axes. Each axis has one transducer interface, one ±10 Control Output, one Fault Input, and one Enable Output.

For other module types without a Control Output, such as analog input, digital I/O, and quadrature input, see the Expansion Modules. For a description of the various module types that make up the RMC75, see the RMC75 Modules Overview topic.

Available Axis Modules

The following axis modules are available for the RMC75:

(the number indicates the number of axes)

Module	Transducer Input Type	Control Output
MA1	MDT or SSI	±10V
MA2	MDT or SSI	±10V
AA1	Analog (±10V or 4-20 mA)	±10V
AA2	Analog (±10V or 4-20 mA)	±10V
QA1	Quadrature (5V RS-422)	±10V
QA2	Quadrature (5V RS-422)	±10V

For details on part numbers, see the RMC75 Part Numbering topic.

Axis Module Specifications

For specifications, see each individual module. In addition, see the Fault Input, Enable Output and Control Output topics.

See Also

RMC75 Modules Overview

7.2.4.2. AA Module

The AA module is one of the axis modules available for the RMC75. It interfaces to analog voltage or current transducers and has 1 analog servo output per axis. The AA module is available with either 1 or 2 axes, called the AA1 and the AA2.

Features

- One 16-bit voltage or current input per axis - each axis individually selectable
- ±10 V and 4-20 mA input ranges
- 8 times oversampling
- One +10 V exciter output per axis
- One ±10 V, 16-bit control output per axis
- Broken wire detection

- Current output up to ±200 mA with VC2124 converter option
- Can be used for controlling position, velocity, pressure, force. Can also be used for dual-loop control such as position-pressure when used in conjunction with the AP2 module.

Part Number

The part numbers of the Analog AA modules are **AA1** and **AA2** for the 1-axis and 2-axis versions, respectively.

For example, RMC75E-AA1 is an RMC75E with a one-axis AA1 module.

Setting Up the AA Module

To set up the AA module, read the following topics:

AA Wiring

Analog Position Scaling

Analog Velocity Scaling

Analog Acceleration Scaling

Analog Pressure/Force Scaling

Specifications (per axis)

General	
Weight	74 g (AA1), 76 g (AA2) + 22 g (per connector)
Analog Input Interface	
Inputs	One differential input per axis
Overvoltage Protection	±40 V
Input Ranges	-10 V to +10 V and 4-20 mA (each axis independently configured)
Broken wire detection	When the input is not connected, internal biasing pulls the input voltage down to its full negative value.
Input Impedance	Voltage Input: 5 MΩ Current Input: 250 Ω
Input Filter Slew Rate	25 V/ms (100 mA/ms)
Oversampling	8 times per control loop
Effective Resolution	19-bit over full ±10 V range (18-bit for 0-10 V and ±5 V, 17 bit for 0-5V, 16 bit for 4-20mA)
Offset Drift with Temperature	0.2 LSB/ºC (1 µA/ºC) typical*
Gain Drift with Temperature	20 ppm/ºC (0.003 %/ºC) typical*
Non-linearity	12 LSB (counts) typical
Exciter Output	10 VDC **±**2%, 8 mA maximum
Control Output	
Range	±10 V @ 5 mA (2 kΩ or greater load)(For current drive, use the VC2124 accessory:±10 mA to ±200 mA in 10 mA steps)
Tolerance At 10 V:	Currently unavailable
Resolution	16 bits
Output Isolation	Not isolated

Overload protection	One-second short-circuit duration
Overvoltage protection	Outputs are protected by clamp diodes
Fault Input	
Input Characteristics	12-24 VDC, sinking or sourcing
Logic Polarity	True High
Input "High" Range	7 to 26.4 VDC (polarity independent) 3 mA maximum
Input "Low" Range	0 to 3.5VDC (polarity independent) <1mA
Maximum Propagation Delay	100µs
Enable Output	
Output Type	Solid State Relay
Logic Polarity	User Selectable to Active Open or Active Closed
Isolation	500 VAC
Rated Voltage	12-24 V, max 30 V (DC or peak AC voltage)
Maximum Current	75 mA
Maximum Propagation Delay	2.0 ms
Closed	Low Impedance (50 Ω maximum)
Open	High Impedance (<1 µA leakage current at 250 V)

See the Fault Input and Enable Output topics for usage details.

*Max Offset Drift 4.8 µA/ºC, Max Gain Drift 0.009 %/ºC under maximum possible circuit component error amount, which is very unlikely to ever occur in practice.

LEDs

Axis LED

This LED represents the input and control status for the axis. This bi-color (red/green) LED has the following states:

State	Description
Steady Off	No power or this input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The input is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by halt command.

En/F LED

This LED represents the status of the Enable Output and Fault Input for the axis. This bi-color (red/green) LED will have the following states:

State	Description

Steady Off	No power or the Control Output is not assigned to an axis.
Steady Green	The Fault input is inactive, and the Enable output is active.
Flashing Green	The Fault input is inactive, and the Enable output is inactive.
Steady Red	The Fault input is active. Notice that it is not possible to tell if the Enable output is active or inactive.

See Also

AA Wiring | Analog Position Scaling | Analog Velocity Scaling | Analog Acceleration Scaling | Analog Pressure/Force Scaling | Analog Fundamentals

7.2.4.3. MA Axis Module

The MA module is one of the axis modules available for the RMC75. It interfaces to Magnetostrictive Start/Stop and PWM and SSI transducers and has 1 analog servo output per axis. The MA module is available with either 1 or 2 axes, called the MA1 and the MA2.

For details on Magnetostrictive and SSI, see the Magnetostrictive Displacement Transducers and Synchronous Serial Interface topics.

Note:
Linear magnetostrictive transducers with SSI output must be of the synchronized type. See the SSI Fundamentals topic for details.

Important!
When using SSI, the MA module requires RS-422 (5 V differential) SSI inputs. It does not support single-ended SSI inputs or higher voltage inputs.

Features

- One MDT or SSI input per axis - each axis individually selectable
- One ±10 V, 16-bit control output per axis
- Current output up to ±200mA with VC2124 converter option
- Can be used for controlling position. Can also be used for dual-loop control such as position-pressure when used in conjunction with the AP2 module.
- SSI can be used for rotary axes.

Part Number

The part numbers of the MA modules are **MA1** and **MA2** for the 1-axis and 2-axis versions, respectively.

For example, RMC75E-MA1 is an RMC75E with a MA1 module.

Supported Transducers

Many different versions of MDT and SSI transducers exist, and the RMC75 MA axis module supports a very wide variety of them. The following tables summarize the major transducer options supported by the MA module:

MDT Options	Value
Types	Start/Stop (S/S)

SSI Options	Value
Data Bits	8 to 32

	Pulse-Width-Modulated (PWM)
Count Range	32 bits
Wiring	Differential, 5 V, RS-422
Interrogation Modes	External. Notice that internal interrogation is not supported.
MDT Count Rate	240 MHz
Output Resolution	16 bits

SSI Format	Binary or Gray Code
SSI Errors	None, all zeros, all ones, or bit 21
Clock Rates	150, 250, and 375 kHz

Note:
Linear magnetostrictive SSI transducers must be of the synchronized type. This ensures that the time between position samples matches the control loop time of the RMC. If the transducer is not synchronized, the sample time may not match and make precise speed control difficult.

Blanking Period (For Neuter Outputs)

The RMC75 blanking period is set to 5µsec. Some older transducers, such as Temposonics I and II with neuter outputs, are not compatible with the RMC75 due to the short blanking period. They are compatible with the RMC150. For more details, see the MDT Fundamentals and MDT Blanking Period topics.

Setting Up the MA Module

To set up the MA module, read the following topics:

MA Wiring

MDT Scaling

SSI Scaling

Specifications (per axis)

Each axis is individually selectable for MDT or SSI.

General	
Weight	72 g (MA1), 71 g (MA2) + 22 g (per connector)
MDT Interface	
Inputs	Two RS422 differential Note: single-ended is not supported
Outputs	Two RS422 differential Note: single-ended is not supported
ESD protection	15 kV
Resolution	0.0005" (12.7 µm) Start/Stop; 0.00005" (1.27 µm) PWM
Count Rate	240 MHz
SSI Interface	
Inputs	Two RS422 differential
Clock Outputs	Two RS422 differential
Clock Frequency	Software selectable 150, 250, or 375 kHz.
Input Impedance	Data + Input: 148 Ω Data - Input: 185 Ω
Cable Type	Twisted pair, shielded

Cable Length Maximum	Transducer Dependent, approx. 300-600 ft. See the SSI Clock Rate topic for details.
ESD protection	15 kV
Resolution	Transducer dependent. Down to 1 µm (approx. 0.00004") for MDTs
Count Encoding	Binary or Gray Code
Count Data Length	8 to 32 bits
Control Output	
Range	±10 V @ 5 mA (2 kΩ or greater load)(For current drive, use the VC2124 accessory:±10 mA to ±200 mA in 10 mA steps)
Tolerance At 10 V:	Currently unavailable
Resolution	16 bits
Output Isolation	Not isolated
Overload protection	One-second short-circuit duration
Overvoltage protection	Outputs are protected by clamp diodes
Fault Input	
Input Characteristics	12-24 VDC, sinking or sourcing
Logic Polarity	True High
Input "High" Range	7 to 26.4 VDC (polarity independent) 3mA maximum
Input "Low" Range	0 to 3.5 VDC (polarity independent) <1 mA
Maximum Propagation Delay	100 µs
Enable Output	
Output Type	Solid State Relay
Logic Polarity	User Selectable to Active Open or Active Closed
Isolation	500 VAC
Rated Voltage	12-24 V, max 30 V (DC or peak AC voltage)
Maximum Current	75 mA
Maximum Propagation Delay	2.0 ms
Closed	Low Impedance (50 Ω maximum)
Open	High Impedance (<1 µA leakage current at 250 V)

See the Fault Input and Enable Output topics for usage details.

LEDs

Axis LED

This LED represents the input and control status for the axis. This bi-color (red/green) LED has the following states:

State	Description
Steady Off	No power or this input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The input is in open loop and the axis is not halted (not possible for reference axes).

Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by halt command.

En/F LED
This LED represents the status of the Enable Output and Fault Input for the axis. This bi-color (red/green) LED will have the following states:

State	Description
Steady Off	No power or the Control Output is not assigned to an axis.
Steady Green	The Fault input is inactive, and the Enable output is active.
Flashing Green	The Fault input is inactive, and the Enable output is inactive.
Steady Red	The Fault input is active. Notice that it is not possible to tell if the Enable output is active or inactive.

See Also
MA Wiring | MDT Fundamentals | SSI Fundamentals

7.2.4.4. QA Axis Module

The QA module is one of the axis modules available for the RMC75. It interfaces to Quadrature encoders (5 V differential only) and has 1 analog servo output per axis. The QA module is available with either 1 or 2 axes, called the QA1 and the QA2.

Features

- One 5 V differential (RS-422) quadrature encoder input per axis
- One ±10 V, 16-bit control output per axis
- Current output up to ±200 mA with VC2124 converter option
- High-speed inputs: one Homing and two Registration inputs per axis.
- Can be used for controlling position. Can also be used for dual-loop control such as position-pressure when used in conjunction with the AP2 module.
- The QA module can be used for rotary axes.
- Software-selectable input termination on ±A and ±B

Part Number

The part numbers of the QA modules are **QA1** and **QA2** for the 1-axis and 2-axis versions, respectively.

For example, RMC75E-QA1 is an RMC75E with a QA1 module.

Setting Up the QA Module

To set up the QA module, read the following topics:

QA Wiring

Specifications (per axis)

General	
Weight	83 g (QA1), 96 g (QA2)
Quadrature Interface	
Encoder Inputs	5 V differential (RS-422) receiver Quadrature A, B, Z (Open collector outputs not supported due to poor noise immunity)
Input Impedance	90 kΩ unterminated 120 Ω terminated A and B termination is software-selectable via the Input Termination parameter. The Z input is always terminated with 120 Ω.
Max. Encoder Frequency	8,000,000 quadrature counts/second
Common Mode Input Range	-10V to +13.2 V
Absolute Max Voltage	±25 VDC. Applying greater than ±25 V will damage the receiver chip and will require repair by Delta. Notice that the quadrature input is *not* compatible with 24V signals.
RegX/PosLim, RegY/NegLim, and Home Inputs	
Input Characteristics	High-Speed 12-24 VDC, Sinking (sourcing driver)
Logic Polarity	True High
Input "High" Range	7 to 26.4 VDC (polarity independent) 3 mA maximum
Input "Low" Range	0 to 3.5 VDC (polarity independent) <1 mA
Response Time	40 µs
Control Output	
Range	±10 V @ 5 mA (2 kΩ or greater load)(For current drive, use the VC2124 accessory:±10 mA to ±200 mA in 10 mA steps)
Tolerance At 10 V:	Currently unavailable
Resolution	16 bits
Output Isolation	Not isolated
Overload protection	One-second short-circuit duration
Overvoltage protection	Outputs are protected by clamp diodes
Fault Input	
Input Characteristics	12-24 VDC, sinking or sourcing
Logic Polarity	True High
Input "High" Range	7 to 26.4 VDC (polarity independent) 3 mA maximum
Input "Low" Range	0 to 3.5 VDC (polarity independent) <1 mA
Maximum Propagation Delay	100 µs
Enable Output	
Output Type	Solid State Relay

Logic Polarity	User Selectable to Active Open or Active Closed
Isolation	500 VAC
Rated Voltage	12-24 V, max 30 V (DC or peak AC voltage)
Maximum Current	75 mA
Maximum Propagation Delay	2.0 ms
Closed	Low Impedance (50 Ω maximum)
Open	High Impedance (<1 µA leakage current at 250 V)

See the Fault Input and Enable Output topics for usage details.

LEDs

Axis LED
This LED represents the input and control status for the axis. This bi-color (red/green) LED has the following states:

State	Description
Steady Off	No power or this input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The input is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by halt command.

En/F LED
This LED represents the status of the Enable Output and Fault Input for the axis. This bi-color (red/green) LED will have the following states:

State	Description
Steady Off	No power or the Control Output is not assigned to an axis.
Steady Green	The Fault input is inactive, and the Enable output is active.
Flashing Green	The Fault input is inactive, and the Enable output is inactive.
Steady Red	The Fault input is active. Notice that it is not possible to tell if the Enable output is active or inactive.

See Also
QA Wiring | Quadrature Scaling | Quadrature Fundamentals

7.2.5. Expansion Modules

7.2.5.1. Expansion Modules Overview

Up to four expansion modules can be added to the RMC75 motion controller for additional functionality. These modules can be installed in the field.

Module	Description	Usage
D8	Discrete I/O 8 individually configurable I/O	Discrete Inputs and Outputs
A2	Analog Reference 2 Analog Voltage or Current Inputs	Analog Reference Inputs
AP2	Analog Pressure 2 Analog Voltage or Current Inputs	Dual-loop control (position/velocity-pressure/force)
Q1	Quadrature Encoder Input	Quadrature Position Reference

Note: It is possible to add more analog inputs than can be assigned to axes. However, it is still possible to view the voltage of the extra analog inputs using the Analog Input Registers. Inputs that are unassigned to axes can have no status bits, error bits, scaling, filtering, etc.

See Also

RMC75 Modules Overview | RMC75 Part Numbering | Adding an Expansion Module

7.2.5.2. Adding an Expansion Module to a Controller

To add an expansion module to an RMC75 controller:

1. Disconnect power to the RMC75.
2. Remove the 4 screws on the far right side of the top and bottom surfaces of the controller.
3. Plug the expansion module into the right side of the controller.
4. Install the 4 screws into the same holes they were removed from in step 2.
5. Apply power to the RMC75.
6. Open RMCTools and open your project. If you do not have a previous project, start a new project and do not continue this procedure.
7. Go online with the controller. If the Controller Hardware/Firmware Differences dialog opens, click **Go Online**. Save the project. On the controller menu, click **Update Flash** to save the RMC75 settings to Flash memory.

   If you are not able to go online with the controller, you can configure your project in RMCTools to include the expansion module as follows:

   - Right-click the controller and click **Properties**.
   - Click **Add Module**, select the module you just added, and click **OK**.
   - Click **OK**, download the settings to the controller, and go online with the controller.

See Also

Expansion Modules Overview

7.2.5.3. A2 Expansion Module

The 2-input Analog expansion module (A2) is one of the optional expansion modules available for the RMC75 motion controller. This module provides two analog inputs with 16-bit resolution.

The A2 should not be confused with the AP*2*, which provides analog input with support for pressure control. If you do not require position-pressure/force or control, you do not need the AP2.

Note: Using the A2 module, it is possible to add more analog inputs on the RMC75 than can be assigned to axes. However, it is still possible to view the voltage of the extra analog inputs using the Analog Input Registers. Inputs that are unassigned to axes can have no status bits, error bits, scaling, filtering, etc.

Features

- Two 16-bit voltage or current inputs - each individually selectable
- ±10 V and 4-20 mA input ranges
- 8 times oversampling
- One +10 V exciter output
- Broken wire detection

Uses

- **Analog Reference Input**
  Reference inputs are often used as gearing or camming masters. Reference inputs cannot be used for direct control of the input axis.
- **Input for a Control Axis**
  The A2 inputs can be used for the input of a control axis. The Control Output must be from an axis module.

Part Number

The part number of the A2 module is EXP70-A2.

Specifications

General	
Weight	102 g + 19 g (connector)
Analog Interface	
Inputs	Two 16-bit differential
Isolation	500 VAC
Overvoltage protection	±40 V
Input ranges	±10 V and 4 20 mA (each channel independently configured)
Broken wire detection	When the input is not connected, internal biasing pulls the input voltage down to its full negative value.
Input impedance	Voltage Input: 5 MΩ Current Input: 250 Ω
Input filter slew rate	25 V/ms
Oversampling	8 times per control loop
Offset drift with temperature +10 V (4-20 mA)	0.2 LSB/ºC (1 µA/ºC) typical*

Gain drift with temperature +10V (4-20 mA)	20 ppm/ºC (0.003 %/ºC) typical*
Non-linearity	12 LSB (counts) typical (+10 V range)
Exciter Output	10 VDC ± 2%, 8 mA

*Max Offset Drift 4.8 µA/ºC, Max Gain Drift 0.009 %/ºC under maximum possible circuit component error amount.

LEDs

Input LED
This LED represents the input and control status for the axis. This bi-color (red/green) LED has the following states:

State	Description
Steady Off	No power or this input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The input is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by halt command.

See Also

A2 Wiring | Analog Position Scaling | Analog Velocity Scaling | Analog Acceleration Scaling | Analog Pressure/Force Scaling | Analog Fundamentals

7.2.5.4. AP2 Expansion Module

The 2-axis Analog Pressure expansion module (AP2) is one of the optional expansion modules available for the RMC75 motion controller. This module provides two analog inputs with 16-bit resolution and support for position-pressure and position-force control.

The AP2 should not be confused with the A*2*, which provides analog input and no support for secondary pressure control. The RMC75 always requires an AP2 module for position-pressure or position-force control.

Note: Using the AP2 module, it is possible to add more analog inputs on the RMC75 than can be assigned to axes. However, it is still possible to view the voltage of the extra analog inputs using the Analog Input Registers. Inputs that are unassigned to axes can have no status bits, error bits, scaling, filtering, etc.

Features

- Two 16-bit voltage or current inputs - each individually selectable
- ±10 V and 4-20 mA input ranges
- 8 times oversampling
- Broken wire detection

Part Number

The part number of the AP2 module is EXP70-AP2.

Uses

The AP2 is useful for the following purposes:

- **Dual-loop Control**

  The AP2 pressure or force inputs can be used in conjunction with the position or velocity control from an RMC7*x* axis module to provide the following combinations of dual-loop control:

  - **Position-Pressure or Position-Force**
  - **Position-Acceleration (advanced)**
  - **Velocity-Pressure or Velocity-Force**
  - **Velocity-Acceleration (advanced)**

- **Pressure/Force Control**

  The AP2 inputs, configured for pressure or force, can be used in conjunction with an output on an RMC75 axis module for solely pressure or force control. The RMC75 always requires an AP2 module for dual-loop. See the Position-Pressure and Position-Force Control and Position-Acceleration topics for more details.

- **Analog Reference Input**

  Reference inputs are often used as gearing or camming masters. Reference inputs cannot be used for direct control of the input axis. If the application requires a position input and not a pressure or force input, the A2 Expansion module is a better choice.

Specifications

General	
Weight	101 g + 16 g (connector)
Analog Interface	
Inputs	Two 16-bit differential
Isolation	500 VAC
Overvoltage protection	±40 V
Input ranges	±10 V and 4 20 mA (each channel independently configured)
Broken wire detection	When the input is not connected, internal biasing pulls the input voltage down to its full negative value.
Input impedance	Voltage Input: 5 MΩ Current Input: 250 Ω
Input filter slew rate	25 V/ms
Oversampling	8 times per control loop
Offset drift with temperature +10 V (4-20 mA)	0.2 LSB/ºC (1 µA/ºC) typical*
Gain drift with temperature +10V (4-20mA)	20 ppm/ºC (0.003 %/ºC) typical*
Non-linearity	12 LSB (counts) typical (+10 V range)

*Max Offset Drift 4.8 µA/ºC, Max Gain Drift 0.009 %/ºC under maximum possible circuit component error amount.

LEDs

Input LED
This LED represents the input and control status for the axis. This bi-color (red/green) LED has the following states:

State	Description
Steady Off	No power or this input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The input is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by halt command.

See Also
AP2 Wiring | Analog Pressure/Force Scaling | Analog Fundamentals

7.2.5.5. D8 Expansion Module

The D8 is one of the expansion modules available for the RMC75. The D8 adds eight discrete inputs or outputs to the RMC75 motion controller. Up to 32 I/O may be added to the RMC75 if the allowed maximum of four expansion modules are all D8s.

Each I/O is individually configurable in software as an input or output.

Features

- Eight discrete I/O points. Each is individually selectable as an input or output.

Uses

The discrete I/O on the D8 can be used for the following purposes:

- Turn outputs on and off with commands
- Use discrete I/O in User Programs
- Use discrete inputs to start User Programs
- SSI homing
- Physical Limit Inputs

Part Number

The part number of the D8 module is EXP70-D8.

Specifications

General	
Weight	113 g + 19 g (connector)
DI/O points	8; each is individually configurable as inputs or outputs.
Inputs	
Input Characteristics	12-24 VDC, sinking or sourcing

Logic Polarity	True High
Isolation	2500 VAC
Input "High" Range	7 to 26.4 VDC (polarity independent) 3 mA maximum
Input "Low" Range	0 to 3.5 VDC (polarity independent) <1 mA
Maximum Propagation Delay	100 µsec
Outputs	
Outputs	Solid State Relay
Isolation	500 VAC
Maximum voltage	± 30 V (DC or peak AC voltage rating of SSR)
Maximum current	75 mA
Maximum propagation delay	2.0 ms
Logic 1 (True, On)	Low impedance (50 Ω maximum)
Logic 0 (False, Off)	High impedance (<1 µA leakage current at 250 V)

LEDs

The D8 expansion module has 1 LED per I/O. The LEDs reflect the state of the input.

State	Description
No color	The input or output is off.
Green	The output is on.
Orange	The input is on.

Note:
Forcing an input *will not* affect the state of the LED. Only a physical current that turns on the input will make the LED red.
Forcing an output *will* make the LED green, because the output will physically be on (conducting).

See Also

D8 Wiring | Discrete IO Overview

7.2.5.6. Q1 Expansion Module

The Q1 expansion module is one of the expansion modules available for the RMC75. It has one quadrature input for 5 V differential A and B signals. The Q1 also has a high-speed **Reg** input that can be used for homing or registration of the quadrature input. Each RMC75 can support up to 2 Q1 modules.

The Q1 is typically used for a reference input from a quadrature encoder. Example applications include providing position a board on a flying cut-off application or the position of a belt on material handling applications.

One quadrature encoder can typically output its A and B signals to thirty-two (32) RMC75 Q1 modules. See the Q1 Wiring topic for details.

Maximum Number of Q1 Modules

The RMC75 can support at most two Q1 modules.

Features

- One 5V RS-422 Quadrature input
- One high-speed registration input for registration or homing

Uses

- **Encoder Reference Input**
  Reference inputs are often used as gearing or camming masters. Reference inputs cannot be used for direct control of the input axis.
- **Input for a Control Axis**
  The Q1 input can be used for the input of a control axis. The Control Output must be from an axis module.

Specifications

General	
Weight	99 g
Quadrature Interface	
Inputs	5V differential (RS422) receiver, Quadrature A, B
Input Impedance	16 kΩ unterminated 249 Ω terminated (selectable by jumpers)
Max. Encoder Frequency	8,000,000 quadrature counts/second
Common Mode Input Range	-10 V to +13.2 V
Absolute Max Voltage	±25 VDC. Applying greater than ±25 V will damage the receiver chip and will require repair by Delta. Notice that the quadrature input is *not* compatible with 24 V signals.
Daisy-Chaining	Daisy-chain one encoder to a maximum of 32 Q1 modules
Reg Input	
Input Characteristics	12-24 VDC, sinking or sourcing
Logic Polarity	True High
Min Turn-On Voltage	5.75 V
Min Turn-On Current	1.25 mA
Max Turn-On Voltage	6.5 V
Max Turn-On Current	2.0 mA
Max Input Voltage	26.4 VDC
Max Input Current	2.6 mA at 24 V
Reg Input Response Time	40 µs

LEDs

Quad LED
This LED represents the input and control status for the axis. This bi-color (red/green) LED has the following states:

State	Description
Steady Off	No power or this input is not assigned to an axis.

Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The input is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by halt command.

Reg LED
This LED represents the physical state of the registration (Reg) input:

State	Description
Off	The RMC75 has no power, or the "Reg" input is off.
Steady Amber	The "Reg" input is on (current is flowing).

See Also
Q1 Wiring | Quadrature Fundamentals

7.3. RMC150

7.3.1. RMC150 Hardware Overview

The RMC150 motion controller supports 2 to 8-axis systems. A number of feedback options are available for controlling a wide variety of hydraulic, electric, and pneumatic position and position–pressure or position-force applications. The built-in 10/100 Ethernet port provides connectivity to many PLCs and HMIs.

See also Controller Features and RMC150 Part Numbering.

Modular Design

The RMC150 is a modular controller. The RMC150E CPU fits in slot 1 (slots numbered 0-5) of the RMC100 backplane and supports most of the modules supported by the RMC100, as listed below.

Backplanes

Backplane widths are available with 3, 4, 5, or 6 slots. Slots are numbered starting with 0 at the left. The example below shows a 6-slot backplane.

Slot #	0	1	2	3	4	5

Slot Descriptions

Slot #	Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5
Alternate Name	Comm Slot	CPU Slot	Sensor Slot 1	Sensor Slot 2	Sensor Slot 3	Sensor Slot 4
Possible Modules	DI/O Universal I/O PROFIBUS	RMC150E RMC151E	Analog (A) Analog (G) Analog (H) MDT (M) SSI (S) Quad (Q) Resolver(R) DI/O Universal I/O SO Module	Analog (A) Analog (G) Analog (H) MDT (M) SSI (S) Quad (Q) Resolver(R) DI/O Universal I/O SO Module	Analog (A) Analog (G) Analog (H) MDT (M) SSI (S) Quad (Q) Resolver(R) DI/O Universal I/O SO Module	Analog (A) Analog (G) Analog (H) MDT (M) SSI (S) Quad (Q) Resolver(R) DI/O Universal I/O SO Module

Module Descriptions

Module	Slots	Description
CPUs		
RMC150E	1	CPU module with 10/100 Ethernet, USB Monitor Port, 2 discrete outputs, 2 discrete inputs.
RMC151E	1	CPU module with 10/100 Ethernet, USB Monitor Port, 2 discrete outputs, 2 discrete inputs, and Pressure/Force dual-loop enabled.
Sensor Modules		
Analog (A)	2-5	4 analog inputs (voltage or current), no analog outputs
Analog (G)	2-5	2 analog inputs (voltage only), 2 analog outputs
Analog (H)	2-5	4 analog inputs (voltage or current), 2 analog outputs
MDT (M)	2-5	2 MDT (Magnetostrictive Displacement Transducer) inputs, 2 analog outputs

SSI (S)	2-5	2 SSI (Synchronous Serial Interface) inputs, 2 analog outputs
Quad (Q)	2-5	2 Quadrature encoder inputs (5V differential A, B, Z), 2 analog outputs, home inputs, travel limit inputs, enable outputs.
Resolver (R)	0, 2-5	2 resolver inputs, 2 analog outputs
SO Module	2-5	1 SSI input, 1 SSI Output which retransmits the SSI input data, 2 analog outputs
Other		
DI/O	0, 2-5	8 discrete outputs, 18 discrete inputs
Universal I/O	0, 2-5	2 analog inputs, 6 discrete I/O, 2 high-speed Quadrature/SSI channels
PROFIBUS	0	PROFIBUS communication

See Also

RMC150 Part Numbering | RMC75 Overview | Mounting and Wiring

7.3.2. RMC150 Part Numbering

Specify RMC150 part numbers when ordering and when contacting Delta customer support.

Part Numbering Schema

The part number of a complete RMC150 is:

RMC15*p*E-*Xn*-*Xn*...-*Y*-*Z*

where

p is 0 for standard controller or 1 for Pressure/Force dual loop enabled

X is the part number of the modules in the sensor slots 1-4 (backplane slots 2-5)

n is the number of modules

Y is the part number of the module in the Comm slot (backplane slot 0)

Z is other options

Part Numbers

Module	Part Number
CPU Modules	
RMC150E	RMC150E
RMC151E	RMC151E
Sensor Slot Options	
Analog (A)	A
Analog (G)	G
Analog (H)	H
MDT (M)	M
SSI (S)	S
Quad (Q)	Q

Resolver (R)	R
DI/O	D
Universal I/O (UI/O)	U
SSI Input/Output (SO)	SO
Blank	BL
Comm Slot Options	
DI/O	DI/O
Universal I/O (UI/O)	UI/O
PROFIBUS	PROFI
Other Options	
Hazardous Locations Class I, Division 2 designation	HZ

Example Part Numbers

Part Number	Description
RMC150E-M1	An RMC150E with 1 MDT module.
RMC151E-M2-H1-DI/O	An RMC151E (pressure/force enabled) with 2 MDT modules, 1 Analog (H) module, and 1 Discrete I/O module in slot 0.
RMC150E-S4	An RMC150E with 4 SSI modules.
RMC150E-M2-D1-PROFI	An RMC150E with 2 MDT modules, one Discrete I/O module in a sensor slot, and a PROFIBUS module in the comm slot.
RMC150E-Q2-D2-DI/O	An RMC150E with 2 Quadrature (Q) modules, 2 Discrete I/O modules, and 1 Discrete I/O module in slot 0.
RMC150E-S4-DI/O-HZ	An RMC150E with 4 SSI modules, and 1 Discrete I/O module in slot 0, and Class I, Division 2 designation.

See Also

RMC150 Overview

7.3.3. RMC150E/RMC151E CPU Module

The **RMC150E** CPU module fits into the RMC100 backplane. The RMC150E CPU, with 10/100 Ethernet communications, supports up to 16 axes, with a maximum of 8 control axes and any number of reference axes up to a total of 16. The RMC150E requires RMCTools for setup, tuning, programming and diagnostics. The RMC150E is *not* supported by Delta's RMCWin software.

The **RMC151E** is identical to the RMC150E, but supports dual-loop Pressure/Force control. Dual-loop Pressure/Force control allows the following control combinations in a single axis:

- Position-Pressure Control
- Position-Force Control
- Position-Acceleration Control

- Velocity-Pressure Control
- Velocity-Force Control
- Velocity-Acceleration Control

Notice that single-loop pressure or force control does not require the RMC151E; the RMC150E supports single-loop pressure or force control applications.

Features

- One 10/100 Mb/s Ethernet port
- One USB Monitor Port
- Supported Ethernet Protocols
  - EtherNet/IP
  - PROFINET
  - Modbus/TCP
  - CSP (also called DF1 over Ethernet)
  - FINS/UDP (Omron)
  - Procedure Exist (Mitsubishi)
  - Delta Motion Control Protocol
- Discrete I/O
  - Two 12-24VDC Discrete Inputs
  - Two 12-24VDC Discrete Output

Part Number

The part numbers of the RM150 are **RMC150E**, or **RMC151E** with Pressure/Force enabled. The RMC150E or RMC151E can only be ordered as part of an entire controller.

Specifications

See also RMC150 Mounting Instructions.

General	
Weight	97 g (CPU module only) + 20 g (connector)
Motion Control	
Supported Number of Axes	Up to 8 control axes (dependent on backplane size) and any number of reference axes up to a total of 16.
Control Loop Time	User-selectable 250 µs, 500 µs, 1 ms, 2 ms, 4ms
Parameter Range	32-bit floating-point
Fail-safe Timers	Control outputs disabled after 16 ms (10 ms for revs older than 1.2C) with no firmware activity.
Ethernet Interface	
Hardware Interface	IEEE 802.3 for 100BASE-T (twisted pair)
Data Rate	10/100 Mbps
Duplex	Full/Half Duplex
Features	Auto-negotiation, Auto-crossover (MDI/MDI-X)
Connector	RJ-45
Cable	CAT5, CAT5e or CAT6, UTP or STP
Configuration Parameters	IP address, subnet mask, gateway address, enable/disable autonegotiation

Configuration methods	BOOTP, DHCP, or static
Ethernet Protocols	
Application protocols	EtherNet/IP, PROFINET, Modbus/TCP, CSP (DF1 over Ethernet), FINS (Omron) Procedure Exist (Mitsubishi), DMCP (Delta Motion Control Protocol)
Framing Protocol	Ethernet II
Internet Protocol IP	IP (includes ICMP, ARP, and Address Collision Detection)
Transport Protocols	TCP, UDP
Other protocols	LLDP
USB Monitor Port Interface	
Connector	USB "B" receptacle
Data Rate	Full-speed (12 Mbps)
Discrete Inputs	
Input Characteristics	2 Discrete Inputs, 12-24 VDC, sinking or sourcing
Logic Polarity	True High
Isolation	500 VAC
Input "High" Range	7 to 26.4 VDC (polarity independent) 3mA maximum
Input "Low" Range	0 to 3.5 VDC (polarity independent) <1 mA
Maximum Propagation Delay	160 µsec
Discrete Outputs	
Outputs	2 Discrete Outputs, Solid State Relay
Isolation	500 VAC
Maximum voltage	±30 V (DC or peak AC voltage rating of SSR)
Maximum current	±75 mA (±50 mA for Class I, Div 2)
Maximum propagation delay	2.0 ms
Logic 1 (True, On)	Low impedance (50 Ω maximum)
Logic 0 (False, Off)	High impedance (<1 µA leakage current at 250 V)
Power	
Voltage	+24 VDC ± 15%
Inrush Current	5A for 150 µs typical
Current - 3 slots	Typical 290 mA @ 24 VDC, max 375 mA
4 slots	Typical 385 mA @ 24 VDC, max 500 mA
5 slots	Typical 485 mA @ 24 VDC, max 625 mA
6 slots	Typical 585 mA @ 24 VDC, max 750 mA

DC-DC converter isolation	500 VAC
Environment	
Operating temperature	+32 to +140°F (0 to +60°C)
Storage temperature	-40 to +185°F (-40 to +85°C)
Humidity	95% non-condensing
Agency compliance	CE, UL and CUL. Class I, Division 2 optional.

LEDs

CPU LED

State	Description
Steady Off	No power.
Steady Green	Power is on.
Flashing Green	Updating Flash or a controller restart is pending.
Flashing Red (Uniform on/off)	The RMC is in the loader. This should occur briefly when the controller is powered up. If this occurs during normal operation, an error has occurred in the controller and it must be reset. Cycle power to the RMC75 to reset it. **Causes:** ○ **Power Interruption** In certain cases, a power interruption to the RMC may cause it to go into the loader. Even a very short interruption, such as 20 msec, can cause problems. This type of interruption cannot be measured with a standard multimeter. For accurate power supply verification, use an oscilloscope or similar high-speed monitoring device. ○ **Firmware Bug** Certain firmware bugs will cause the RMC to go into the loader. If this occurs, and power interruptions have been ruled out, contact Delta Technical support.
Flashing Red (Pattern)	The LED will flash 1, 2, or 3 times, pause and then repeat the pattern. All of these patterns indicate that the RMC failed a self-test of the RAM subsystem during startup. This is a non-recoverable major fault. The module will likely need to be shipped back to Delta for repair.
Steady Red	Non-Recoverable major fault. That is, the module cannot boot at all, not even into the loader. The module will likely need to be shipped back to Delta for repair.

RUN LED

State	Description
Steady Off	PROGRAM Mode or no power
Steady Green	RUN Mode

Communication LEDs

These LEDS are located below the Controller LED on a white background. These LEDs do not convey 10/100, FDX/HDX, or Collision information. The user will have to rely on their switch's LEDs or the RMCTools Communication Statistics to view those Ethernet status items.

Link/Act LED

The Link/Act LED reflects the status of the physical Ethernet connection between the RMC and the device on the other end of the Ethernet cable. If this LED is not on or is not blinking when you expect it to be, see the Ethernet Link/Act LED topic for troubleshooting information.

State	Description
Off	The Ethernet link is down
Flashing Green	The Ethernet link is up, with activity
Steady Green	The Ethernet link is up, no activity

Net LED

State	Description
Steady Off	No power, or no IP address configured.
Steady Red	Duplicate IP Address detected.
Flashing Red	IP address configured and in use, and a controlling EtherNet/IP I/O connection has timed out and not been re-established.
Flashing Green	IP address configured and in use, no EtherNet/IP or PROFINET connections are currently established, and there is no timed-out controlling EtherNet/IP I/O connection.
Steady Green	IP address configured and in use, at least one EtherNet/IP or PROFINET connection is currently established, and there is no timed-out controlling EtherNet/IP I/O connection.

Discrete I/O LEDs

State	Description
Off	No power or input/output is off.
On	Input/output is on.

Startup LED Test

When the RMC150 powers up, the LEDs listed below goes through these states, with a 250 msec delay between each step:

CPU LED	Run LED	Net LED
Green	Off	Off
Red	Off	Off
Green	Green	Off
Green	Red	Off
Green	Off	Green
Green	Off	Red
Green	Off	Off

Prior to the LED test, the module will run through the loader, during which time it will have a red CPU LED. After the LED test, the firmware will continue initialization, after which time the LEDs resume their regular behaviors.

See Also

RMC150E Wiring

7.3.4. Analog Modules

7.3.4.1. Analog (H) Module (RMC150)

Two Control Outputs and Four 16-bit Analog Inputs

The Analog (H) module for the RMC150 interfaces to analog voltage or current transducers. Providing four 16-bit resolution inputs, this module includes control outputs and is capable of controlling pressure, force, and position applications. The Analog (H) module front bezel is labeled **ANLG**.

The Analog (H) module can be used for position control. However, higher precision of position can typically be obtained with MDT or SSI feedback. Analog systems are often more susceptible to noise.

In addition to the Analog (H) module, the RMC150 supports the following analog modules:

- Analog (G) Module
  Two control outputs and two 16-bit analog voltage inputs for voltage-feedback position, velocity, pressure, or force control applications.
- Analog (A) Module
  A four-input 12-bit analog module with no control outputs. Used for reference inputs, or the secondary inputs of position-pressure or position-force axes.

Features

- Four isolated 16-bit inputs
- 8 times oversampling
- ±10 V, ±5 V, and 4-20 mA input ranges
- +10 V exciter output
- Two isolated, ±10 V, 12-bit control outputs per module
- Current output up to ±200 mA with VC2124 converter option
- Broken wire detection

Part Number

The part number of the Analog H module is **H*n***, where ***n*** is the number of Analog (H) modules.

For example, RMC150E-H1 is an RMC150E with one Analog (H) module. RMC150E-H3-Q1 is an RMC150E with three Analog (H) modules and one Quadrature module.

Setting Up the H Module

To set up the H module, read the following topics:

Analog Wiring

Analog Position Scaling

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

General	
Weight	115 g + 23 g (stock connectors)
Analog Input Interface	
Inputs	Four 16-bit differential
Isolation	500 VAC
Overvoltage Protection	±40 V

Input Ranges	±10 V, ±5 V and 4-20 mA (each channel independently configured via jumper)
Broken wire detection	When the input is not connected, internal biasing pulls the input voltage down to its full negative value.
Input Impedance	Voltage Input: 1 MΩ Current Input: 250 Ω
Input Filter Slew Rate	25 V/ms
Oversampling	8 times per control loop
Offset drift with temperature	0.2 LSB/°C typical (+10 V range)
Gain drift with temperature	20 ppm/°C typical (+10 V range)
Non-linearity	12 LSB (counts) typical (+10 V range)
Exciter Output	10 VDC ± 2%, 8 mA
Control Outputs	
Range	±10 V @ 5 mA (2 kΩ or greater load) (For current output, use the VC2124 accessory:±10 mA to ±200 mA in 10 mA steps)
Tolerance	At 10 V: +200 mV, -100 mV At 0 V: ±50 mV At -10 V: +100 mV, -200 mV
Resolution	12 bits
Output Isolation	500 VAC
Overload Protection	One-second short-circuit duration
Overvoltage Protection	Outputs are protected by clamp diodes

LEDS

Axis 0 LED

Axis LED applies to the Drive 0 output *or* feedback input 0 as described below:

1. If Drive 0 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 0 LED.
2. Otherwise, if a reference input axis (no output) uses input 0, then that axis will own the Axis 0 LED.
3. Otherwise, if Drive 0 is assigned to an output-only axis, then that axis will own Axis 0 LED.

Axis 1 LED

Axis 1 LED applies to the Drive 1 output *or* feedback input 1 or 2 as described below:

1. If Drive 1 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 1 LED.
2. Otherwise, if a reference input axis (no output) uses input 2, then that axis will own the Axis 1 LED.
3. Otherwise, if a reference input axis (no output) uses input 1, then that axis will own the Axis 1 LED.
4. Otherwise, if Drive 1 is assigned to an output-only axis, then that axis will own the Axis 1 LED.

LED Colors

State	Description
Steady Off	No power or the output/input is not assigned to an axis or the inputs are assigned to an axis as secondary inputs.

Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The output is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by a halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by a halt command.

See Also

Analog Inputs (A) module (RMC150) | Analog (G) module (RMC150) | RMC150 Overview | Analog Wiring

7.3.4.2. Analog (A) Module (RMC150)

Four 12-bit Analog Inputs

The Analog (A) module for the RMC150 provides four 12-bit resolution analog voltage or current inputs. This module does not include control outputs. The Analog (A) module is typically used for reference inputs, such as from a joystick, or for the secondary input of a position-pressure or position-force axis. The Analog (H) module front bezel is labeled **Analog Inputs**.

In addition to the Analog (A) module, the RMC150 supports the following analog modules:

- Analog (H) Module
  Two control outputs and four 16-bit inputs for controlling position, velocity, pressure, or force.
- Analog (G) Module
  Two control outputs and two 16-bit analog voltage inputs for voltage-feedback position, velocity, pressure, or force control applications.

Features

- Four isolated 12-bit inputs
- 8 times oversampling
- ±10 V, ±5 V, and 4-20 mA input ranges
- +10 V exciter output
- Broken wire detection

Part Number

The part number of the Analog (A) module is **A*n***, where ***n*** is the number of Analog (A) modules.

For example, RMC150E-M1-A1 is an RMC150E with one MDT and one Analog (A) module. RMC150E-S2-A2 is an RMC150E with two SSI and two Analog (A) modules.

Setting Up the Analog (A) Module

To set up the Analog (A) module, read the following topics:

Analog Wiring

Analog Position Scaling

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

General	
Weight	105 g + 16 g (stock connectors)
Analog Input Interface	
Inputs	Four 12-bit differential
Isolation	500 VAC
Overvoltage Protection	±40 V
Input Ranges	±10 V, ±5 V, and 4-20 mA (each channel independently configured via jumper)
Broken wire detection	When the input is not connected, internal biasing pulls the input voltage down to its full negative value.
Input Impedance	Voltage Input: 1 MΩ Current Input: 250 Ω
Input Filter Slew Rate	25 V/ms
Oversampling	8 times per control loop
Offset drift with temperature	0.01 LSB/°C typical (+10 V range)
Gain drift with temperature	20 ppm/°C typical (+10 V range)
Non-linearity	1 LSB (count) typical (+10 V range)
Exciter Output	10 VDC ± 2%, 8 mA

See Also

Analog (H) module (RMC150) | Analog (G) module (RMC150) | RMC150 Overview | Analog Wiring

7.3.4.3. Analog (G) Module (RMC150)

Two Control Outputs and Two 16-bit Voltage Inputs

Note: Delta recommends using the H module instead of the G module. The similarly-priced H module has four analog inputs instead of two, and supports 4-20 mA feedback.

The Analog (G) module for the RMC150 interfaces to analog voltage transducers (current is not supported). Providing two 16-bit resolution inputs, this module includes control outputs and is capable of controlling pressure, force, and position applications. The Analog (G) module front bezel is labeled **ANLG2**.

The Analog (G) module can be used for position control. However, higher precision of position can typically be obtained with MDT or SSI feedback. Analog systems are often very susceptible to noise.

In addition to the Analog (G) module, the RMC150 supports the following analog modules:

- Analog (H) Module
  Two control outputs and four 16-bit inputs for controlling position, velocity, pressure, or force.
- Analog Inputs (A) Module
  A four-input 12-bit analog module with no control outputs. Used for reference inputs, or the secondary inputs of position-pressure or position-force axes.

Features

- Two isolated 16-bit inputs

- 16 times oversampling
- ±10 V input range
- +10 V exciter output
- Two isolated, ±10 V, 12-bit control outputs per module
- Broken wire detection
- Current output up to ±200 mA with VC2124 converter option

Part Number

The part number of the Analog (G) module is **G*n***, where ***n*** is the number of Analog (G) modules.

For example, RMC150E-G1 is an RMC150E with one Analog (G) module. RMC150E-S1-G2 is an RMC150E with one SSI module and two Analog (G) modules.

Setting Up the Analog (G) Module

To set up the Analog (G) module, read the following topics:

Analog Wiring

Analog Position Scaling

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

General	
Weight	84 g + 12 g (stock connectors)
Analog Input Interface	
Inputs	Two 16-bit differential
Isolation	500 VAC
Overvoltage Protection	±40 V
Input Ranges	±10 V
Broken wire detection	When the input is not connected, internal biasing pulls the input voltage down to its full negative value.
Input Impedance	1 MΩ
Input Filter Slew Rate	25 V/ms
Oversampling	16 times per control loop
Offset drift with temperature	0.2 LSB/°C typical
Gain drift with temperature	20 ppm/°C typical
Non-linearity	25 LSB (counts) typical
Exciter Output	10 VDC ± 2%, 8 mA
Control Outputs	
Range	±10 V @ 5 mA (2 kΩ or greater load) (For current output, use the VC2124 accessory:±10 mA to ±200 mA in 10 mA steps)
Tolerance	At 10 V: +200 mV, -100 mV At 0 V: ±50 mV At -10 V: +100 mV, -200 mV
Resolution	12 bits
Output Isolation	500 VAC

Overload Protection	One-second short-circuit duration
Overvoltage Protection	Outputs are protected by clamp diodes

LEDS

Axis 0 LED

Axis LED applies to the Drive 0 output *or* feedback input 0 as described below:

1. If Drive 0 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 0 LED.
2. Otherwise, if a reference input axis (no output) uses input 0, then that axis will own the Axis 0 LED.
3. Otherwise, if Drive 0 is assigned to an output-only axis, then that axis will own Axis 0 LED.

Axis 1 LED

Axis 1 LED applies to the Drive 1 output *or* feedback input 1 as described below:

1. If Drive 1 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 1 LED.
2. Otherwise, if a reference input axis (no output) uses input 1, then that axis will own the Axis 1 LED.
3. Otherwise, if Drive 1 is assigned to an output-only axis, then that axis will own the Axis 1 LED.

LED Colors

State	Description
Steady Off	No power or the output/input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The output is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by a halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by a halt command.

See Also

Analog (H) module (RMC150) | Analog (A) module (RMC150) | RMC150 Overview | Analog Wiring

7.3.5. MDT Module

7.3.5.1. MDT (M) Module (RMC150)

The MDT module for the RMC150 interfaces to Magnetostrictive Displacement Transducers (MDTs). These absolute-position transducers are especially well suited for hydraulic applications because of their non-contact design, robustness, modularity, and resistance to contaminants. The RMC150 MDT module supports transducers with Start/Stop and Pulse Width Modulated (PWM) outputs. The RMC150 can be configured to trigger from the rising or falling edges on Start/Stop transducers. MDTs with SSI output are supported by the RMC150 SSI Module.

See the MDT Fundamentals topic for details on how MDT feedback works.

Features

- Two axes of MDT feedback per module
- Up to 0.001” resolution using Start/Stop
- Up to 0.0001” resolution using PWM with multiple recirculations
- Supports internal transducer recirculations
- Transducer length up to 398 inches (10.1 m) (4 ms loop time)
- Differential (recommended) or single-ended interface
- Two isolated, ±10 V, 12-bit control outputs per module
- Current output up to ±200 mA with VC2124 converter option

Part Number

The part number of the MDT module is **M*n***, where ***n*** is the number of MDT modules.

For example, RMC150E-M1 is an RMC150E with one MDT module. RMC150E-M3-Q1 is an RMC150E with three MDT modules and one Quadrature module.

Supported Transducers

The following tables summarize the major transducer options supported by the MDT module:

MDT Options	Value
Types	Start/Stop (S/S) Pulse-Width-Modulated (PWM)
Count Range	32 bits
Wiring	Differential, 5 V, RS-422, Single-ended 5 V
Interrogation Modes	External. Notice that internal interrogation is not supported.
MDT Count Rate	120 MHz

Blanking Period

The RMC150 has a blanking period options of 5 µsec (default) and 21 µsec. Some older transducers, such as Temposonics I and II with neuter outputs, require the longer blanking period. For more details, see the MDT Fundamentals and MDT Blanking Period topics.

Setting Up the MDT Module

To set up the MDT module, read the following topics:

MDT Wiring

MDT Scaling

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

General	
Weight	74 g + 16 g (stock connectors)
MDT Interface	
Axes	Two per module
Inputs	Two RS-422 differential (can also be used single-ended)
Outputs	Two RS-422 differential (can also be used single-ended)

ESD Protection	15 kV
Resolution	0.001 (25.4 µm) in Start/Stop; 0.0001 in (2.54 µm) PWM
Count Rate	120 MHz
Control Outputs	
Range	±10 V @ 5 mA (2 kΩ or greater load) (For current output, use the VC2124 accessory:±10 mA to ±200 mA in 10 mA steps)
Tolerance	At 10 V: +200mV, -100 mV At 0 V: ±50mV At -10 V: +100 mV, -200 mV
Resolution	12 bits
Output Isolation	500 VAC
Overload Protection	One-second short-circuit duration
Overvoltage Protection	Outputs are protected by clamp diodes

LEDS

Axis 0 LED

Axis LED applies to the Drive 0 output *or* feedback input 0 as described below:

1. If Drive 0 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 0 LED.
2. Otherwise, if a reference input axis (no output) uses input 0, then that axis will own the Axis 0 LED.
3. Otherwise, if Drive 0 is assigned to an output-only axis, then that axis will own Axis 0 LED.

Axis 1 LED

Axis 1 LED applies to the Drive 1 output *or* feedback input 1 as described below:

1. If Drive 1 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 1 LED.
2. Otherwise, if a reference input axis (no output) uses input 1, then that axis will own the Axis 1 LED.
3. Otherwise, if Drive 1 is assigned to an output-only axis, then that axis will own the Axis 1 LED.

LED Colors

State	Description
Steady Off	No power or the output/input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The output is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by a halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by a halt command.

See Also

RMC150 Overview | MDT Wiring

7.3.6. SSI Module

7.3.6.1. SSI (S) Module (RMC150)

Important!
The SSI module requires RS-422 (5V differential) SSI inputs. It does not support single-ended SSI inputs or higher voltage inputs.

The SSI module for the RMC150 interfaces to feedback devices using the Synchronous Serial Interface (SSI) standard. Many types of transducers are available with SSI, including magnetostrictive displacement transducers (MDTs), absolute encoders, and laser measuring devices. SSI has a number of advantages over other transducer interfaces. First, it offers higher noise immunity. Second, it gives absolute positions. Third, it supports a wide variety of transducers. Finally, many SSI devices offer higher precision; for example, MDTs with SSI output are available with resolutions to 0.5µm.

Note: Linear MDTs with SSI output should be of the synchronized type. Non-synchronized is not well-suited for motion control.

See the SSI Fundamentals topic for details on the SSI standard.

Features

- Two axes of SSI feedback per module
- Supports SSI devices with Binary or Gray Code data from 8 to 31 bits in length
- Differential RS-422 SSI interface
- Two isolated, ±10 V, 12-bit control outputs per module
- Current output up to ±200 mA with VC2124 converter option

Part Number

The part number of the SSI module is ***Sn***, where ***n*** is the number of SSI modules.

For example, RMC150E-S1 is an RMC150E with one SSI module. RMC150E-S3-Q1 is an RMC150E with three SSI modules and one Quadrature module.

Setting Up the SSI Module

To set up axes with SSI feedback, read the following topics:

SSI Wiring

SSI Scaling

The following parameters must be also set for axes with SSI feedback:

- SSI Data Bits
- SSI Format

The following advanced parameters can be optionally set for axes with SSI feedback:

- SSI Clock Rate
- Wire Break Detection
- SSI Overflow Mode
- SSI Home Source

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

General	
Weight	74 g + 16 g (stock connectors)

SSI Interface	
Inputs	Two RS-422 differential (does not support single-ended)
Clock Outputs	Two RS-422 differential (does not support single-ended)
Clock Frequency	user-selectable 230 kHz, 921 kHz
Input Impedance	Data + Input: 140 Ω Data - Input: 150 Ω
Required Cable Type	Twisted pair, shielded
Maximum Cable Length	Transducer Dependent, approx. 300-600 ft. See the SSI Clock Rate topic for details.
ESD Protection	15 kV
Resolution	Transducer dependent
Count Encoding	Binary or Gray Code
Count Data Length	8 to 31 bits
Control Outputs	
Range	±10 V @ 5 mA (2 kΩ or greater load) (For current output, use the VC2124 accessory:±10 mA to ±200 mA in 10 mA steps)
Tolerance	At 10 V: +200 mV, -100 mV At 0 V: ±50 mV At -10 V: +100 mV, -200 mV
Resolution	12 bits
Output Isolation	500 VAC, optically isolated
Overload Protection	One-second short-circuit duration
Overvoltage Protection	Outputs are protected by clamp diodes

LEDS

Axis 0 LED

Axis LED applies to the Drive 0 output *or* feedback input 0 as described below:

1. If Drive 0 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 0 LED.
2. Otherwise, if a reference input axis (no output) uses input 0, then that axis will own the Axis 0 LED.
3. Otherwise, if Drive 0 is assigned to an output-only axis, then that axis will own Axis 0 LED.

Axis 1 LED

Axis 1 LED applies to the Drive 1 output *or* feedback input 1 as described below:

1. If Drive 1 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 1 LED.
2. Otherwise, if a reference input axis (no output) uses input 1, then that axis will own the Axis 1 LED.
3. Otherwise, if Drive 1 is assigned to an output-only axis, then that axis will own the Axis 1 LED.

LED Colors

State	Description
Steady Off	No power or the output/input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.

Flashing Green	The output is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by a halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by a halt command.

See Also

RMC150 Overview | SSI Wiring

7.3.7. Quadrature Module

7.3.7.1. Quadrature (Q) Module (RMC150)

The Quadrature module for the RMC150 interfaces to RS-422 quadrature feedback devices, both rotary encoders and linear transducers. The Quad module allows cost effective control of a wide variety of electric drives as well as electric and hydraulic servo motors.

The 2-axis Quad module generates analog control outputs and interfaces to quadrature encoders with 5V RS-422 differential signals. Additional high-speed inputs allow for homing, registration, or positive and negative limits on a per axis basis.

See the Quadrature Fundamentals topic for details on how quadrature feedback works.

Features

- Two axes of 5 V RS-422 differential quadrature feedback per module
- Phase A and B (A, A, B, B)
- Index Z (Z, Z)
- 4,000,000 counts/second maximum
- High-speed dedicated Home position input
- 50 µs response to home input
- Positive and negative travel limit inputs
- Digital noise filters on all inputs
- Status LEDs
- Two isolated, ±10 V, 14-bit control outputs per module
- Current output up to ±200 mA with VC2124 converter option
- 1 Fault input per axis
- 1 Enable output per axis
- 2 Inputs per axis for Travel Limits or Registration Inputs
- All discrete inputs are isolated

Part Number

The part number of the MDT module is **Q*n***, where ***n*** is the number of Quad modules.

For example, RMC150E-Q1 is an RMC150E with one Quad module. RMC150E-Q3-H1 is an RMC150E with three Quad modules and one Analog H module.

Setting Up the Quad Module

To set up the Quad module, read the following topics:

Quad Wiring

Quadrature Scaling

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

General	
Weight	135 g
Quadrature Interface	
Encoder Inputs	5 V RS-422 differential receiver Quadrature A, B, Z
Input Impedance	215 Ω termination
Max. Encoder Frequency	4,000,000 quadrature counts/second
ESD Protection	15 kV
RegX/PosLim, RegY/NegLim, Home and Fault Inputs	
Input Characteristics	5-24 VDC, sinking (sourcing driver)
Logic Polarity	True High
Isolation	500 VAC
Input "High" Range	3.2 to 26.4 VDC 3.5 mA minumum, 10 mA maximum
Input "Low" Range	0 to 2 VDC <1 mA
Max Propagation Delay	50 µs
Enable Output	
Output Type	Solid State Relay
Logic Polarity	User selectable to Active Open or Active Closed
Rated Voltage	12-24 V, max ±30 V (DC or peak AC voltage)
Maximum Current	±75 mA (±50 mA for Class I, Div 2)
Maximum Propagation Delay	1.5 ms
Closed	Low impedance (50 Ω maximum)
Open	High impedance (<1 µA leakage current at 250 V)
Fault Input	
Input Characteristics	5-24 VDC, Sinking (sourcing driver)
Logic Polarity	True High
Input "High" Range	3.2 to 26.4 VDC (polarity independent) 10 mA maximum
Input "Low" Range	0 to 2 VDC (polarity independent) <1 mA
Maximum Propagation Delay	50 µs
Control Outputs	

Range	±10 V @ 5 mA (2 kΩ or greater load) (For current VC2124, use the VC2100 accessory:±10 mA to ±200 mA in 10 mA steps)
Tolerance	At 10 V: +200 mV, -100 mV At 0 V: ±50 mV At -10 V: +100 mV, -200 mV
Resolution	14 bits
Output Isolation	500 VAC
Overload Protection	One-second short-circuit duration
Overvoltage Protection	Outputs are protected by clamp diodes

Using the Quadrature Module Features

See the following topics for details on the various features of the Quadrature module:

Fault Input

Enable Output

Registration

Homing

LEDS

Axis 0 LED

Axis LED applies to the Drive 0 output *or* feedback input 0 as described below:

1. If Drive 0 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 0 LED.
2. Otherwise, if a reference input axis (no output) uses input 0, then that axis will own the Axis 0 LED.
3. Otherwise, if Drive 0 is assigned to an output-only axis, then that axis will own Axis 0 LED.

Axis 1 LED

Axis 1 LED applies to the Drive 1 output *or* feedback input 1 as described below:

1. If Drive 1 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 1 LED.
2. Otherwise, if a reference input axis (no output) uses input 1, then that axis will own the Axis 1 LED.
3. Otherwise, if Drive 1 is assigned to an output-only axis, then that axis will own the Axis 1 LED.

LED Colors

State	Description
Steady Off	No power or the output/input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The output is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by a halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by a halt command.

See Also

RMC150 Overview | Quadrature Wiring | Quadrature Scaling

7.3.8. Resolver Module

7.3.8.1. Resolver (R) Module (RMC150)

Two Control Outputs and Two Resolver Inputs

The Resolver (R) module for the RMC150 interfaces to resolvers. The Resolver module front bezel is labeled **RES**.

Features

- Two axes of resolver feedback per module
- 14 or 16 bit resolution
- Reference frequency from 800 Hz to 5 kHz
- Resolver Transformation Ratios from 0.42 to 1.41
- Two isolated, ±10 V, 14-bit drive outputs per module
- Current output up to ±200 mA with VC2124 or VC2100 converter options
- Custom options available – contact Delta

Part Number

The part number of the Resolver module is **R*n***, where ***n*** is the number of Resolver modules.

For example, RMC150E-R1 is an RMC150E with one Resolver module. RMC150E-R3-Q1 is an RMC150E with three Resolver modules and one Quadrature module.

Setting Up the Resolver Module

1. Wire the resolver according to the RMC150 Resolver Wiring topic.
2. Set the following parameters:
    - Resolver Resolution
    - Reference Amplitude
    - Reference Frequency
3. Scale the axis. Typically, resolvers will use Rotary Scaling. Notice that one revolution of the resolver will always consist of 65,536 counts on the RMC, regardless of the resolver resolution parameter.

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

General	
Weight	78 g
Resolver Interface	
Axes	Two per module
Reference Frequency	800 Hz to 5 kHz
Reference Output Voltage	1.42 to 4.8V RMS
Reference Output Current	28 mA max

Resolver Transformation Ratio (SIN_{MAX}/Reference)	0.42 to 1.41
Resolver Input Voltage	2V RMS ±15%
Resolution	14 or 16 bits
Maximum Speed	14 bits: 3000 RPM 16 bits: 600 RPM
Maximum Acceleration	14 bits: 1200 RPS per second 16 bits: 60 RPS per second
Accuracy	4 Minutes + 1 LSB
Control Outputs	
Range	±10 V @ 5 mA (2 kΩ or greater load) (For current output, use the VC2124 accessory:±10 mA to ±200 mA in 10mA steps)
Tolerance	At 10 V: +200 mV, -100 mV At 0 V: ±50 mV At -10 V: +100 mV, -200 mV
Resolution	14 bits
Output Isolation	500VAC

LEDS

Axis 0 LED

Axis LED applies to the Drive 0 output *or* feedback input 0 as described below:

1. If Drive 0 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 0 LED.
2. Otherwise, if a reference input axis (no output) uses input 0, then that axis will own the Axis 0 LED.
3. Otherwise, if Drive 0 is assigned to an output-only axis, then that axis will own Axis 0 LED.

Axis 1 LED

Axis 1 LED applies to the Drive 1 output *or* feedback input 1 as described below:

1. If Drive 1 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 1 LED.
2. Otherwise, if a reference input axis (no output) uses input 1, then that axis will own the Axis 1 LED.
3. Otherwise, if Drive 1 is assigned to an output-only axis, then that axis will own the Axis 1 LED.

LED Colors

State	Description
Steady Off	No power or the output/input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The output is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by a halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by a halt command.

See Also

RMC150 Overview | RMC150 Resolver Wiring | Resolver Fundamentals

7.3.8.2. Resolver (RW) Module (RMC150)

See the Resolver (R) module for standard resolver interface specifications. The Resolver (RW) module described in this topic is for a specific resolver configuration.

Two Control Outputs and Two Resolver Inputs

The Resolver (RW) module for the RMC150 interfaces to resolvers that operate at 400 Hz and 26 V RMS. The Resolver module front bezel is labeled **RES**.

Features

- Two axes of resolver feedback per module
- 14 or 16 bit resolution
- Reference frequency 400 Hz
- Resolver Transformation Ratio: 0.45
- Two isolated, ±10 V, 14-bit drive outputs per module
- Current output up to ±200 mA with VC2124 or VC2100 converter options
- Custom options available – contact Delta

Part Number

The part number of this Resolver module is **RW*n***, where ***n*** is the number of Resolver modules.

For example, RMC150E-RW1 is an RMC150E with one Resolver (RW) module. RMC150E-RW3-Q1 is an RMC150E with three Resolver (RW) modules and one Quadrature module.

Setting Up the Resolver Module

1. Wire the resolver according to the RMC150 Resolver Wiring topic.
2. Set the Resolver Resolution.
3. Scale the axis. Typically, resolvers will use Rotary Scaling. Notice that one revolution of the resolver will always consist of 65,536 counts on the RMC, regardless of the resolver resolution parameter.

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

Resolver (RW) Interface	
Axes	Two per module
Reference Frequency	400 Hz
Reference Input Voltage	26 V RMS
Resolver Transformation Ratio (SIN_{MAX}/Reference)	0.45
Resolver Input Voltage	11.8 V RMS ±15%
Resolution	14 or 16 bits
Maximum Speed	14 bits: 360 RPM

	16 bits: 90 RPM
Maximum Acceleration	14 bits: 30 RPS per second 16 bits: 4 RPS per second
Accuracy	4 Minutes + 1 LSB
Control Outputs	
Range	±10 V @ 5 mA (2 kΩ or greater load) (For current output, use the VC2124 accessory:±10 mA to ±200 mA in 10mA steps)
Tolerance	At 10 V: +200 mV, -100 mV At 0 V: ±50 mV At -10 V: +100 mV, -200 mV
Resolution	14 bits
Output Isolation	500VAC

LEDS

Axis 0 LED

Axis LED applies to the Drive 0 output *or* feedback input 0 as described below:

1. If Drive 0 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 0 LED.
2. Otherwise, if a reference input axis (no output) uses input 0, then that axis will own the Axis 0 LED.
3. Otherwise, if Drive 0 is assigned to an output-only axis, then that axis will own Axis 0 LED.

Axis 1 LED

Axis 1 LED applies to the Drive 1 output *or* feedback input 1 as described below:

1. If Drive 1 is assigned to an axis that has an input (i.e. isn't output-only), then that axis will own the Axis 1 LED.
2. Otherwise, if a reference input axis (no output) uses input 1, then that axis will own the Axis 1 LED.
3. Otherwise, if Drive 1 is assigned to an output-only axis, then that axis will own the Axis 1 LED.

LED Colors

State	Description
Steady Off	No power or the output/input is not assigned to an axis.
Steady Green	The axis is either in closed loop control or is a reference axis. The axis is not halted.
Flashing Green	The output is in open loop and the axis is not halted (not possible for reference axes).
Flashing Red	A Closed Loop Halt or External Halt has occurred because an error bit is set (not possible for reference axes). The LED will not turn red due to a halt caused by a halt command.
Steady Red	An Open Loop Halt or Direct Output Halt has occurred because an error bit is set (for reference axes, any halt has occurred). The LED will not turn red due to a halt caused by a halt command.

See Also

RMC150 Overview | RMC150 Resolver Wiring | Resolver Fundamentals | Resolver (R) Module

7.3.9. DI/O Module

7.3.9.1. DI/O Module (RMC150)

The Discrete I/O module for the RMC150 contains eight discrete outputs and eighteen discrete inputs. The discrete I/O are compatible with signals ranging from 5 to 24 VDC.

Discrete I/O can be used for the following purposes:

- Turn outputs on and off with commands
- Use discrete I/O in User Programs
- Use discrete inputs to start User Programs
- Physical Limit Inputs

Part Number

The DI/O Module can be placed in slot 0, 3, 4, or 5 of the RMC150 backplane. However, because the backplane connector of slot 0 (the leftmost slot) is different from that of slots 3-5, the part number of a DI/O module for slot 0 differs from a DI/O module for slots 3-5.

If placed in slot 0, the part number is **DI/O**. If placed in slots 3, 4, or 5, the part number is **D*n***, where ***n*** is the number of D modules. Multiple Discrete I/O modules can be placed in the same rack.

Slot	Part Number
0	**DI/O** This must be added to the end of the part number, for example RMC150E-M1-H2-DI/O.
3-5	**D*n*** where ***x*** is the number of DI/O modules, for example RMC150E-M1-D2.

Setting Up the DI/O Module

To set up the DI/O module, read the following topics:

Discrete I/O Configuration

DI/O Wiring

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

Note:
The characteristics of the I/O on the DI/O module differ from the I/O on the RMC150E CPU module.

General	
Weight	116 g + 28 g (stock connectors)
Inputs	
Inputs	18, compatible with signal levels ranging from 5 V to 24 V.
Input Characteristics	5-24 VDC, Sinking (sourcing driver)
Logic Polarity	True High
Isolation	500 VAC
Input "High" Range	3 to 26.4 VDC 3.2 mA minumum, 10 mA maximum

Input "Low" Range	0 to 2 VDC <1 mA
Filtering	Inputs 0-15: 500 µsec Inputs 16-17: 250 µsec
Maximum Propagation Delay	100 µsec + filtering
Outputs	
Outputs	8, Solid State Relay
Logic polarity	True On
Isolation	500 VAC RMS optically isolated
Maximum voltage	± 30 V (DC or peak AC voltage rating of SSR)
Maximum current	±75 mA (±50 mA for Class I, Div 2)
Maximum propagation delay	1.5 ms
Logic 1 (True, On)	Low impedance (50 Ω maximum)
Logic 0 (False, Off)	High impedance (<1 µA leakage current at 250 V)

See Also

RMC150 Overview | DI/O Wiring (RMC150)

7.3.10. UI/O Module

7.3.10.1. UI/O Module (RMC150)

Two Analog Inputs, 6 Discrete I/O, 2 Quadrature/SSI Channels with Inter-Controller Communication

The Universal I/O module for the RMC150 provides two analog inputs, six discrete I/O, and two high-speed channels that can be configured for quadrature inputs, SSI inputs, or even inter-controller communications for synchronizing axes between RMCs.

Analog Input Features

- Two 16-bit analog inputs, ±10 V or 4-20 mA

Discrete I/O Features

- 6 I/O, individually configurable as input or output
- Inputs: 12 to 24 VDC, sinking (require sourcing driver)
- Outputs: Solid state relay, 50 mA continuous
- Inputs 0 and 1 can be used as high-speed registration inputs in conjunction with the quadrature inputs.

High-Speed Channels

Each of the two RS-422 channels are independently configurable as Quadrature or SSI.

Quadrature Channels

- A and B quadrature inputs

- Requires 5 V differential (RS-422) signals
- Discrete inputs 0 and 1 can be used as high-speed registration or homing inputs in conjunction with quadrature inputs 0 and 1, respectively.

SSI Channels

The SSI channels can be configured to do the following tasks:

- **Receive SSI Input from a Transducer**
  This is a standard SSI input for obtaining data from an SSI transducer or encoder.
- **Monitor SSI Communication on Another Device**
  The SSI channel can monitor the data that another RMC is receiving from an SSI device. This makes it possible to synchronize multiple RMCs to one SSI transducer.
- **Send Data Out Via SSI (Slave)**
  The SSI channel behaves as a transducer or encoder and will return data to the requesting master.
- **Send Data between RMCs**
  The SSI channels can continuously send any RMC register to an SSI input on another RMC. This provides communication between RMCs. Notice that only one register can be transferred between RMCs.

Part Number

The UI/O Module can be placed in slot 0, 2, 3, 4, or 5 of the RMC150 backplane. However, because the backplane connector of slot 0 (the leftmost slot) is different from that of slots 2-5, the part number of a UI/O module for slot 0 differs from a UI/O module for slots 2-5.

If placed in slot 0, the part number is **UI/O**. If placed in slots 2, 3, 4, or 5, the part number is **U*n***, where ***n*** is the number of **U** modules. Multiple Universal I/O modules can be placed in the same rack.

Slot	Part Number
0	**UI/O** This must be added to the end of the part number, for example RMC150E-M1-H2-UI/O.
2-5	**U*n*** where ***n*** is the number of UI/O modules, for example RMC150E-M1-U2.

Setting Up UI/O Analog Inputs

1. Wire the analog input as described in the Analog Wiring topic.
2. Assign the input to an axis as described in the Defining Axes topic.
3. In the Axis Parameters Pane, for the axis to which the input is assigned, set the **Input Type** parameter to Voltage or Current.
4. Scale the axis as described in the Analog Position Scaling topic.

Setting Up UI/O Discrete I/O

1. Use the Discrete I/O Configuration dialog to configure each I/O point as an input or output.
2. Wire the I/O points as described in the RMC150 UI/O Wiring topic.

Setting Up UI/O Quad/SSI Channels

These channels must be configured before being used for such tasks as assigning to axis inputs. See Configuring UI/O Channels for details.

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

General

Weight	86 g + 28 g (stock connectors)
Analog Input Interface	
Inputs	Two 16-bit differential
Isolation	500 VDC
Overvoltage Protection	±40 V
Input Ranges	±10 V and 4-20 mA (each input individually configurable)
Input Impedance	5 MΩ
Input Frequency	1.2 kHz
Input Filter Slew Rate	25 V/ms
Sampling Rate	60 kHz
Offset Drift with Temperature	0.2 LSB/°C typical
Gain Drift with Temperature	20 ppm/°C typical
Non-linearity	12 LSB (counts) typical over full 16-bit range
Discrete I/O - General	
Discrete I/O Points	6; each is individually configurable as input or output
Isolation	500 VAC
Discrete I/O - Inputs	
Input Characteristics	12-24 VDC, sinking or sourcing
Logic Polarity	True High
Input "High" Range	7 to 26.4 VDC (polarity independent), 3 mA maximum
Input "Low" Range	0 to 3.5 VDC (polarity independent), <1 mA
Maximum Propagation Delay	160µsec + filtering
Filtering	50 µsec (value stable for 7 samples @ 8 µsec interval)
Registration	Inputs 0 and 1 can be used as high-speed registration inputs in conjunction with the quadrature inputs. For this use, filtering can be set to 100 ns (max propagation delay is still 100 µsec).
Discrete I/O - Outputs	
Output Characteristics	Solid State Relay
Logic Polarity	True On
Maximum Voltage	± 30 V (DC or peak AC voltage rating of SSR)
Maximum Current	±75 mA
Maximum Propagation Delay	2.0 ms
Logic 1 (True, On)	Low impedance (50 Ω maximum, 25 Ω typical)
Logic 0 (False, Off)	High impedance (<1 µA leakage current at 250 V)
High-Speed Channels	
Channels	2, independently configurable
Transducer types	Magnetostrictive (with SSI output), single- or multi-turn absolute encoders, quadrature encoders.

	Note: Linear magnetostrictive transducers with SSI output should be of the synchronized type. Non-synchronized is not well-suited for motion control.
Modes	Quadrature input SSI Standard input – for interfacing to transducers and encoders SSI Monitor input – for monitoring SSI communication on another device SSI Slave output – emulates a transducer SSI Master output – for sending data to another controller
Input Type (Data/Clock/Quad)	RS-422 (5 V differential) (Single-ended encoders not supported due to low noise immunity)
Output Type (Clock/Data)	RS-422 (5 V differential)
Clock frequency	250 kHz, 500 kHz, or 971 kHz, user-selectable
Resolution	Transducer dependent (typically down to 2 µm or approximately 0.00008” for MDTs)
SSI Count encoding	Binary or Gray Code
SSI Count data length	8 to 32 bits
Input Impedance	16 kΩ unterminated 120 Ω terminated A and B termination is software-selectable via the Input Termination parameter.
Registration	Discrete inputs 0 and 1 can be used as high-speed registration for channels 0 and 1 respectively.
Registration Response Time	160 µs
Max Encoder Frequency	8,000,000 quadrature counts per second
Maximum Cable Length	Dependent on transducer and cable quality (Low capacitance, shielded, twisted pair computer communication cable). SSI Wire Delay compensation can allow wire lengths that exceed the transducer manufacturer's specifications.
Electrostatic Discharge (ESD)protection	15 kV (human body model)
Isolation	500 VAC

See Also

RMC150 Overview | UI/O Wiring

7.3.10.2. Configuring UI/O High-Speed Channels

Each of the Universal I/O module's two RS-422 channels are independently configurable as Quadrature or SSI. These channels must be configured before being used for such tasks as assigning to axis inputs. To configure the channels, open the UI/O properties:

1. In the Project Pane, expand the **Modules** folder.
2. Double-click the desired UI/O module, then choose the **Quad/SSI** page.

3. Each channel can be configured to operate in one of the modes listed in the table below. See the sections below for instructions on configuring the channels for specific uses.

Quadrature Input	Is used by assigning to a control or reference axis. • +/-A and +/-B differential inputs with wire break detection. • One I/O point per channel (R0 for channel 0 or R1 for channel 1) can be used as a reference or home input for high-speed count latching. • Filtering on the R0/R1 input for use in high-speed latching is configurable via an axis parameter. • Termination on the A and B inputs can be enabled/disabled via an axis parameter.
SSI Axis Input	Is used by assigning to a control or reference axis. The following items can be configured via axis parameters: • **SSI Clock Mode:** Standard (Clock is output) or Monitor (Clock is input) • **Wire Delay Compensation:** Selects the amount to delay sampling Data with respect to Clock. A wizard is provided to calculate the proper value. • **SSI Input Termination:** Enable/disable termination on the Data input (and Clock input for Monitor mode) • **SSI Clock Rate:** 250 kHz, 500 kHz, or 971 kHz • **SSI Data Format:** Gray code or binary • **SSI Data Bits:** 8-32
SSI Register Input	Result is placed in any selected variable. This type of input cannot be assigned to an axis. The same configuration options as **SSI Axis Input** are provided for this mode, but are located in the Quad/SSI setup page for the module instead of in axis parameters.
SSI Output	Outputs the contents of a selectable register over the SSI +/-Data lines. Also, channel 1 has the option of automatically echoing the data coming in on channel 0 (if configured as an SSI Axis/Register Input) in which case a register is not selected. This is useful in cases where the user wants to share the SSI feedback with other devices. The following items can be configured through the Quad/SSI setup page: • **SSI Output Mode:** Master (Clock is output) or Slave (Clock is input) • **SSI Termination (Slave mode only):** Enable/disable termination on the Clock input • **SSI Clock Rate:** 250 kHz, 500 kHz, or 971 kHz • **SSI Data Format:** Gray code or binary • **SSI Data Bits:** 8-32

Configure Channel as a Quadrature Axis Input

For each channel that will be used as an input to an axis, do the following:

1. Choose **Quadrature Input**, then click **OK**.
2. Wire the quadrature input as described in the RMC150 UI/O Wiring topic.
3. Assign the input to an axis as described in the Defining Axes topic.
4. In the Axis Parameters Pane, set the following axis parameters:

- Input Termination to apply termination. Termination should always be used. If the high-speed channel inputs are daisy-chained, apply termination only to the last input in the chain. See RMC150 UI/O Wiring topic for details.
- If the axis will use the Reg input for registration or homing, uncheck Filter Reg Input if you need a faster response and are not concerned about electrical noise.

5. Scale the axis feedback as described in the Quadrature Scaling topic.

Configure Channel as an SSI Axis Input

For each channel that will be used as an input to an axis, do the following:

1. Choose **SSI Axis Input**, then click **OK**.
2. Wire the SSI input as described in the RMC150 UI/O Wiring topic.
3. Assign the input to an axis as described in the Defining Axes topic.
4. In the Axis Parameters Pane, set the following axis parameters:
   - SSI Data Bits
   - SSI Format
   - SSI Termination to apply termination. Termination should always be used. If the high-speed channel inputs are daisy-chained, apply termination only to the last input in the chain. See RMC150 UI/O Wiring topic for details.
   - Set the SSI Clock Mode to **Standard**
   - SSI Clock Rate, if necessary. The default value is usually fine.
   - SSI Wire Delay, if necessary, as described in the SSI Wire Delay topic.
   - SSI Overflow Mode, if necessary. The default value is usually fine.
5. Scale the axis feedback as described in the SSI Scaling topic.

Configure Daisy-Chained SSI Devices to Multiple UI/O Modules

Daisy-chaining refers to wiring an SSI device to multiple UI/O modules. When wiring a daisy-chained SSI system, the SSI master (the UI/O) should be on one end of the daisy chain with the SSI device on the other end, and any monitoring UI/O modules in the middle of the daisy chain. The wiring must be done in a sequential fashion, that is, the wiring goes from the SSI device to the first UI/O, then from that UI/O to the next UI/O, etc. Apply termination only to the SSI master. See RMC150 UI/O Wiring for more details.

1. Configure a channel on the master UI/O as an **SSI Axis Input** as described in the **Axis Input** section above. Use the SSI Termination parameter to apply termination.
2. For the remaining monitoring UI/O modules:
   a. Configure a channel as **SSI Axis Input** as described in the **Axis Input** section above, but set the SSI Clock Mode parameter to **Monitor** and do not apply termination.

Using SSI Output Mode

The SSI Output mode can be used to configure the SSI channel for the following:

- **SSI Device**
  The SSI channel behaves an encoder. This can be used to transfer data to a controller that has a standard SSI input. The remote controller sends a clock signal to the SSI channel, and the SSI channel returns the data.
  1. In the **Mode** box, choose **SSI Output**.
  2. Choose **Output the Contents of the following Register.**
  3. Set the **SSI Output Mode** to **Slave**.
  4. Set the remaining parameters as required.
- **SSI with Clock Output and Data Output**
  This is designed for connection of this channel to an SSI Monitor channel (with both clock and slave as inputs).
  1. In the **Mode** box, choose **SSI Output**.

2. Choose **Output the Contents of the following Register.**
3. Set the **SSI Output Mode** to **Master**.
4. Set the remaining parameters as required.

- **SSI Echo**
  Channel 1 is an SSI device and outputs the same data that Channel 0 received.
  1. In the **Mode** box, choose **SSI Output**.
  2. Choose **Automatically retransmit SSI Input 0 to SSI Output 1.**
  3. Typically, channel 1 will act as an SSI device, in which case **SSI Output Mode** should be set to **Slave**.
  4. Set the remaining parameters as required.
- **Bidirectional Communication Between Two RMCs**
  See below.

Configure Bidirectional Communication Between Two RMCs

The UI/O module can be used to communicate between RMCs.

1. Configure the UI/O module on the first RMC as follows:
   a. In Channel 0:
      - Choose **SSI Output** mode.
      - Set the **SSI Output Mode** to **Slave**.
      - In the **Source of Output Data** section, choose the register that contains the data to be sent. Typically this is a variable.
      - Set the **SSI Data Bits** to 32.

   b. In Channel 1:
      - Choose **SSI Register Input** mode
      - Set the **SSI Input Mode** to **Standard**
      - Set the **SSI Termination** to **±Data**.
      - In the **Location to store SSI Data** section, choose the register where the received data will be stored. This must be a variable.
      - Set the **SSI Data Bits** to 32.
2. Configure the UI/O on the second RMC as follows:
   a. In Channel 0:
      - Choose **SSI Register Input** mode
      - Set the **SSI Input Mode** to **Standard**
      - Set the **SSI Termination** to **±Data**.
      - In the **Location to store SSI Data** section, choose the register where the received data will be stored. This must be a variable.
      - Set the **SSI Data Bits** to 32.

   b. In Channel 1:
      - Choose **SSI Output** mode
      - Set the **SSI Output Mode** to **Slave**.
      - In the **Source of Output Data** section, choose the register that contains the data to be sent. Typically this is a variable.
      - Set the **SSI Data Bits** to 32.
3. Wire Channel 0 on the first RMC to Channel 0 on the second RMC as shown in RMC150 UI/O Wiring. Wire Channel 1 on the first RMC to Channel 1 on the second RMC.
4. Click **OK**.
5. The RMCs will automatically send data back and forth to each other.

See Also

UI/O Module | UI/O Wiring

7.3.11. PROFIBUS Module

7.3.11.1. PROFIBUS Module (RMC150)

The PROFIBUS module for the RMC150 provides PROFIBUS communications. Notice that since the RMC150E CPU has built-in Ethernet, it is possible to communicate via Ethernet and PROFIBUS simultaneously.

For details on setting up and using PROFIBUS, see the PROFIBUS Overview topic.

Features

- PROFIBUS-DP
- 1 Comm Status LED
- Standard PROFIBUS-DP DB-9 Connector

Part Number

The part number of the PROFIBUS module is **-PROFI**. The PROFIBUS module fits only in the left-most slot of the RMC backplane.

For example, RMC150E-S1-PROFI is an RMC150E with an SSI module and a PROFIBUS module. RMC150E-S3-Q1-PROFI is an RMC150E with three SSI modules, one Quadrature module, and a PROFIBUS module.

Specifications

For general specifications on the RMC150, see the RMC150 Overview topic.

General	
Weight	88 g
PROFIBUS-DP Interface	
Data Rate	9.6 kbaud up to 12 Mbaud
Isolation	2500 VAC
Product Identifier Number	0x0AC6
Features Supported	Sync Mode, Freeze Mode, Auto-baud rate detect
Valid Station Addresses	0-126 (set by software or Set Slave Address function)
Connector	Standard PROFIBUS-DP DB-9 (use termination in cable connectors as per PROFIBUS specification)

LEDS

Comm Status

State	Description
Steady Off	No PROFIBUS connection is established. See the Troubleshooting PROFIBUS topic for possible reasons.

Steady Green	A PROFIBUS connection is established.

Note:
These are the only two LED states of the NET LED, but it is possible to have the Net LED flashing or flickering green, which indicates that the RMC75P is going on- and off-line on the PROFIBUS channel and generally indicates a network or configuration problem.

See Also

RMC150 Overview | PROFIBUS Overview

7.4. RMC200

7.4.1. RMC200 Hardware Overview

The RMC200 offers many controller features for controlling a wide variety of hydraulic, electric, and pneumatic position and position–pressure or position-force applications. The built-in 10/100 Ethernet ports provide connectivity to many PLCs and HMIs.

The RMC200 offers Lite and Standard options.

	CPU	Bases	Max Physical Control Axes	Max Control Loops	Max Total Axes, including Virtual, Reference, and Outer Loop
Lite	CPU20L, with integrated power supply	B5L, B7L	18	36	48*
Standard	CPU40	B5, B7, B11, B15	50	100	128*

*The number of reference axes is limited by the hardware and will never reach this limit

See Comparing RMC200 Lite and Standard for more details.

Feature Key

The Feature Key specifies the control features of the RMC200. Currently, it specifies the number of control loops.

The key is a removable token installed in the CPU module. This allows for easy transfer from one RMC200 CPU to another when replacing one RMC200 with a spare RMC200. When initially purchased, the Feature Key comes loaded with the customer-ordered features. Features may also be added to the Feature Key later.

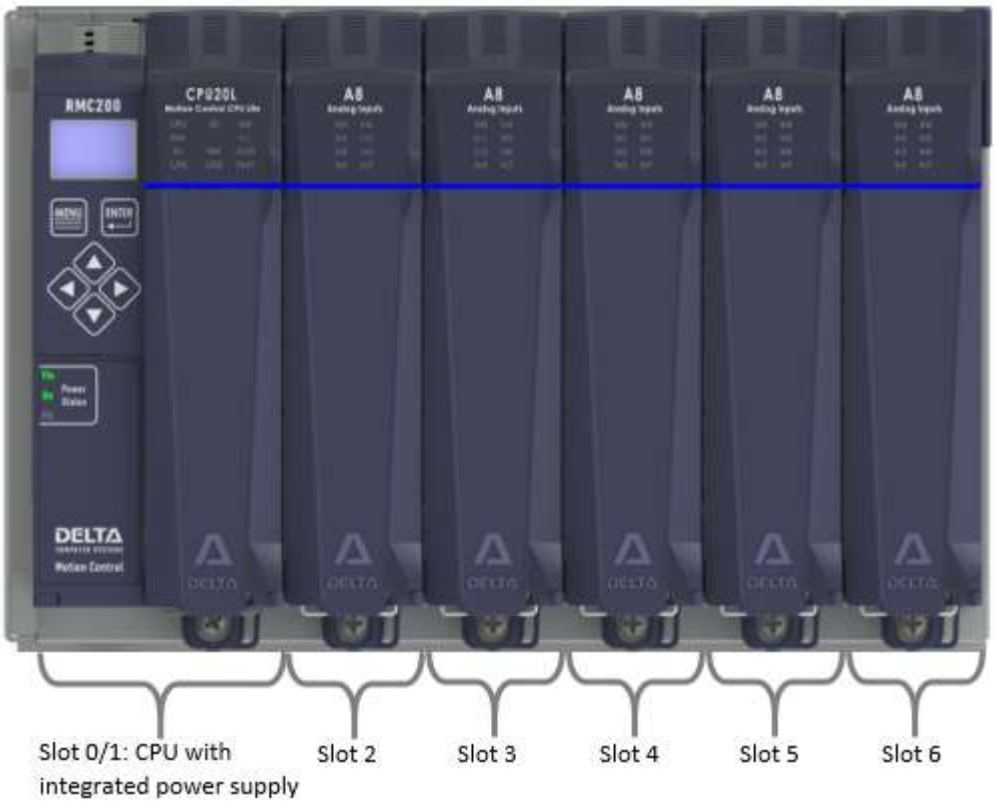
Modules

The RMC200 consists of a backplane with user-swappable power, CPU, and I/O modules.

RMC200 Lite:

Slot 0/1: CPU module with integrated power supply

Remaining slots: I/O modules



RMC200 Standard:

Slot 0: Power supply module

Slot 1: CPU module

Remaining slots: I/O modules

Module	Description
Backplanes	
B5L	5-slot Lite base
B7L	7-slot Lite base
B5	5-slot Standard base
B7	7-slot Standard base
B11	11-slot Standard base
B15	15-slot Standard base
Power Supplies	
PS4D	35 W, 24 VDC input power supply, for the B5, B7 and B11 bases
PS6D	50 W, 24 VDC input power supply, for the B15 base
CPUs	
CPU20L	Up to 18 physical control axes (number of control loops purchased separately, specified by the Feature Key) 2 Ethernet ports, 1 USB port 2 Discrete inputs (24 VDC), 2 Discrete outputs Integrated 24 VDC power supply
CPU40	Up to 50 physical control axes (number of control loops purchased separately, specified by the Feature Key) 2 Ethernet ports, 1 USB port 2 Discrete inputs (24 VDC), 2 Discrete outputs
I/O Modules	
CA4	4 Control Outputs (±10 V, 4-20 mA, ±20 mA)
CV8	8 Control Outputs (±10 V only)
S8	8 SSI, Start/Stop, or PWM Inputs, supports one quadrature input or one SSI monitor input
A8	8 Analog Inputs (±10 V, 4-20 mA)
LC8	8 Load Cell Inputs, ±5 mV/V, with Sense Input
Q4	4 Quadrature Encoder Inputs, supporting RS-422, HTL, or TTL, with home and registration inputs
U14	4 Analog Inputs, 2 Analog Outputs, 4 Discrete I/O, and 2 High-Speed channels for SSI, MDT, or quadrature encoder inputs, each with an extra high-speed discrete input
D24	20 configurable I/O, 4 fixed high-speed inputs for general purpose inputs, or 2 quadrature encoder inputs, or 2 pulse counter inputs

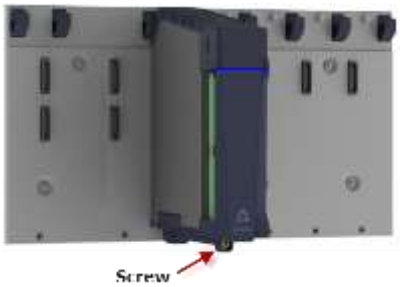
Installing and Removing Modules

To install a module on the backplane:

1. Tilt the module backwards.

2. Insert the module's hinge pins into the backplane hooks.

3. Tilt the module down.



4. Tighten the screw to 5 in-lbs (0.6 Nm).

Removal is the reverse of installation.

Power Dissipation

The maximum power dissipation of each RMC module is listed below.

Module	Power
B5L	2.0 W
B7L	2.7 W
B5	2.5 W
B7	2.7 W
B11	3.0 W
B15	3.2 W
PS4D	7.0 W
PS6D	10 W
CPU20L	9.0 W, includes power dissipation of integrated power supply
CPU40	8.0 W
CA4	1.4 W, all analog outputs in voltage mode 2.8 W, all analog outputs in current mode
CV8	2.0 W
S8	1.8 W
A8	1.4 - 2.4 W, depending on use of Exciter Output
LC8	1.4 - 2.4 W, depending on use of Exciter Output **Note:** Max power consumption is 1.3 W with no Exciter Output, 3.2 W with Exciter Output
Q4	1.4 W

U14	2.6 W, both analog outputs in voltage mode 3.2 W, both analog outputs in current output
D24	1.0 W, plus 50 mW per I/O point used

General Specifications

See also RMC200 Mounting.

Mechanical	
Mounting	Panel-mount
Dimensions B5L	7.0x 7.9 x 5.8 in. (WxHxD) (177 x 200.0 x 146 mm), with mounting lugs
Dimensions B7L	9.7 x 7.9 x 5.8 in. (WxHxD) (246 x 200.0 x 146 mm), with mounting lugs
Dimensions B5	7.9 x 7.9 x 5.8 in. (WxHxD) (200 x 200.0 x 146 mm), with mounting lugs
Dimensions B7	10.7 x 7.9 x 5.8 in. (WxHxD) (270 x 200.0 x 146 mm), with mounting lugs
Dimensions B11	16.2 x 7.9 x 5.8 in. (WxHxD) (410 x 200 x 146 mm), with mounting lugs
Dimensions B15	21.9 x 7.9 x 5.8 in. (WxHxD) (555 x 200 x 146 mm), with mounting lugs
Environment	
Operating temperature	-4 to +140°F (-20 to +60°C)
Storage temperature	-40 to +185°F (-40 to +85°C)
Humidity	Non-condensing
Agency compliance	CE, UL, CUL

See Also

RMC200 Part Numbering | RMC200 Mounting Instructions | Comparing RMC200 Lite and Standard

7.4.2. Comparing RMC200 Lite and Standard

The RMC200 Lite CPU20L and Standard CPU40 are very similar. The differences are listed below. In general, the CPU40 has more processing power and supports more axes. It also has more time available for running user programs, which can become important at small loop times.

	CPU20L	**CPU40**
Axes and Control Loops		
Max Physical Control Axes	18	50
Max Control Loops	36	100
Max Total Axes, including Virtual and Reference	48	128

Max User Tasks	32	64
Modules		
Power Supply	Integrated	Requires separate PS4D or PS6D power supply module
Supported bases	B5L, B7L	B5, B7, B11, B15
Supported Loop Times		
Supported single-loop control axes vs Loop Time[1]	125 µs: 0-3[2] 250 µs: 0-7[2] 500 µs: 18[4] 1 ms: 18 2 ms: 18 4 ms: 18	125 µs: 0-6[2] 250 µs: 0-11[2] 500 µs: 32[3] 1 ms: 50 2 ms: 50 4 ms: 50
Plots		
Max total plot data items	512	1024
Max number of plot templates	48	64
Max number of data items per plot	48	128
Plot storage	48 MB (12M samples)	192 MB (48M samples)
Curves		
Curve Storage	16 MB	64 MB
Communication		
Max EtherNet/IP processing	6000 packets/sec	8000 packets/sec

[1] More axes may be used for a given loop time by selecting High Control Loop Utilization. See the Loop Time topic for details.

[2] Requires selecting High Control Loop Utilization. See the Loop Time topic for details.

[3] Above 14 axes requires selecting High Control Loop Utilization. See the Loop Time topic for details.

[4] Above 11 axes requires selecting High Control Loop Utilization. See the Loop Time topic for details.

See Also

RMC200 Overview | CPU20L | CPU40

7.4.3. RMC200 Part Numbering

Specify RMC200 part numbers when ordering and when contacting Delta customer support.

A complete RMC200 motion controller requires:

- Backplane
- Power Supply
- CPU
- I/O modules
- Feature Key (for specifying control features, such as number of control axes and number of dual-loop axes)

Feature Key Part Numbering

The Feature Key specifies the control features of the RMC200. The key is a removable token installed in the CPU module. This allows for easy transfer from one RMC200 CPU to another when replacing one RMC200 with a spare RMC200. When initially purchased, the Feature Key comes loaded with the customer-ordered features. Features may be added to the Feature Key later.

Feature	Ordering Part Number
Feature Key	**R2-KL*nnn*, where *nnn* is the number of control loops**

Module Part Numbers

Lite bases support only the CPU20L. Standard bases support only the CPU40. All bases support all I/O modules.

Module	Ordering Part Number
Bases	
5-Slot Lite	**R200-B5L**
7-Slot Lite	**R200-B7L**
5-Slot Standard	**R200-B5**
7-Slot Standard	**R200-B7**
11-Slot Standard	**R200-B11**
15-Slot Standard	**R200-B15**
Power Supplies	
35 W, 24V input, for B5, B7 and B11 bases	**R200-PS4D**
50 W, 24V input, for B15 base	**R200-PS6D**
No power supply required for B5L or B7L	
CPU Modules	
CPU20L Lite, includes integrated power supply	**R200-CPU20L**
CPU40 Standard	**R200-CPU40**
I/O Modules	
4 Control Outputs (±10 V, 4-20 mA, ±20 mA)	**R200-CA4**
8 Control Outputs (±10 V only)	**R200-CV8**
8 Analog Inputs (±10 V, 4-20 mA)	**R200-A8**
8 SSI, Start/Stop, or PWM Inputs, supports one quadrature input	**R200-S8**
8 Load Cell Inputs, ±5 mV/V, with Sense Input	**R200-LC8**
4 Quadrature Inputs with home and registration	**R200-Q4**

4 Analog Inputs, 2 Analog Outputs, 4 Discrete I/O, 2 High-Speed Channels for SSI, MDT, or quadrature encoder inputs, each with an extra high-speed discrete input	**R200-U14**
20 Discrete I/O, 4 high-speed inputs for general-purpose, quadrature or pulse counter	**R200-D24**
Accessories	
Slot Cover	**R2-SC**

See Also

RMC200 Overview

7.4.4. Feature Key

The Feature Key specifies the control features of the RMC200. The key is a removable token installed in the CPU module. This allows for easy transfer of control features from one RMC200 CPU to another when replacing one RMC200 with a spare RMC200. When initially purchased, the Feature Key comes loaded with the customer-ordered features. Features may also be added to the Feature Key later.

Initial Ordering

When initially ordering an RMC200 CPU module, specify the desired control features. These features will be loaded on the removable Feature Key installed in the CPU.

Available Features

The following features are offered for the feature key:

Control Loops
The control loops on the feature key define the number of control axes. The maximum number of control loops supported by each control is:

	Max Control Loops
CPU20L Lite	36
CPU40 Standard	100

When defining axes, you may define more axes than the control loops allow, but the additional axes will be non-functional. The number of control loops required per axis are listed below.

- **Single-loop axis: 1 control loop**
  An axis is a single-loop axis if it controls one quantity, such as only position, or only force.
- **Dual-loop axis: 2 control loops**
  An axis that controls two quantities such as position and force.
- **Reference axis: 0 control loops**
  An axis with only an input, and no control output. Also called a half-axis.
- **Virtual axis: 0 control loops**
  A type of reference axis with no input, and no control, but provides virtual motion.
- **Output only axis: 0 control loops**
  An axis with only an analog output.

Removing and Installing the Feature Key

The Feature Key can be transferred from one RMC to another. When replacing a CPU module with a spare, the Feature Key should be transferred to the new CPU to ensure that all the control features are transferred.

1. Turn off power to the RMC200.
2. Remove the hold-down screw on the CPU module.
3. Remove the CPU module by rocking the bottom of the CPU module forward (toward you) and then lifting the module out.
4. Locate the yellow Feature Key on the back of the CPU.
5. Pull the Feature Key out.
6. Installation is the reverse of removal.

Adding Features to an Existing Feature Key

Adding features to the feature key involves generating a file to send to Delta, and applying a response file to the RMC.

Important: Once you generate a feature request file, do not generate another one until applying the response file. Generating a new Feature Key request file will invalidated the response file.

1. **Purchase Features**

   Determine which features you wish to add (see **Available Features** above), then contact your Delta distributor or Regional Sales Manager for payment details.

   Ordering Part Number: R2-Loop

   The part number **R2-Loop** specifies one loop. Order the quantity of loops desired.

2. **Generate the Feature Key Request File**
   a. In RMCTools, go online with the RMC200.
   b. In the Project Pane, expand the **Modules** folder and double-click the RMC200 CPU module.
   c. On the Feature Key page, click **Add Features**.
   d. Choose **Select features to add and generate a feature key response file** and click **Next**.
   e. Choose the desired features to add and click **Next**.
   f. Browse to a location to save the Feature Key request file.
   g. Click **Finish** to close the wizard. Do not generate another Feature Key request file until applying the response file you receive from Delta, otherwise the response file will be invalidated.
3. **Send the Feature Key Request File to Delta**
   a. Email the Feature Key response file to featurekey@deltamotion.com. In the email, include your contact information so Delta can verify that the features have been purchased.

b. Once Delta has verified payment, Delta will respond to the email with an attached Feature Key response file.

c. Save the Feature Key response file to your PC.

4. **Apply the Feature Key Response File**

   After receiving the Feature Key response file from Delta Computer Systems:

   a. Go online with the RMC200 and open the **RMC Feature Key** wizard as described in step 2 above.

   b. Choose **Apply the response that I have already received** and click **Next**.

   c. Browse to the location of the saved Feature Key Response file and click **Next**.

   d. The wizard will report the added features.

Specifications

Feature Key - Apply only to the Feature Key, not the CPU or other modules	
Contact Life	10,000 cycles min.
ESD Protection	15 kV
Read Cycles	Unlimited
Write Cycles	100,000 minimum (writes are performed only when applying new features)
Data Life (Storage) at 35°C	30 years minimum, 50 years typical
Operational Temperature	-40°C to +85°C (-40°F to +185°F)
Storage Temperature	-40°C to +100°C (-40°F to +212°F)

See Also

RMC200 Overview | CPU40 Module | CPU20L Module

7.4.5. Using the RMC200 SD Card

The SD card on the RMC200 provides the following functionality:

1. Save and restore the controller image
2. Store general files (e.g. machine design documents, data sheets, etc.)

You can use RMCTools to view the SD card contents and to copy/move files from and to the SD card. The RMC200 supports only SD cards with the FAT32 or FAT16 file system. Any SD card that is compatible with the RMC200 can also be inserted into and used in a PC.

Compatible Cards

The RMC200 SD card slot is compatible with the **R2-MEM-SD-1G** SD card sold by Delta.

You may also use any SD card that conforms to the specifications below. Delta recommends using industrial-grade cards with the RMC200. Industrial grade cards provide many benefits over commercial grade, including a wide temperature range and long data retention life.

Supported SD Cards	
Form factor	Standard size (32 mm × 24 mm × 2.1 mm)
Families	SD (SDSC) (standard capacity) and SDHC (high capacity)
File system	FAT32 and FAT16
Capacities	Up to 32 GB

SD LED

The SD LED indicates the state of the SD card. Do not remove the SD card when the SD LED is flashing green, or it may corrupt the card.

SD LED State	Description
Off	**No SD card** No SD card is present.
Steady Green	**SD Card Present** A compatible card (SDSC or SDHC) is present, and the card is inactive and safe to remove.
Flashing Green	**Read or Write Active - Do Not Remove** The controller is reading from or writing to the SD card. Do not remove the SD card while the LED is flashing.
Steady Red	**Incompatible SD Card** A card is present, but is not compatible. Make sure the card is SD (SDSC) or SDHC and has been formatted with a valid file system (FAT32 or FAT16).

Browsing the SD Card from RMCTools

To view the SD card contents and to copy/move files from and to the SD card:

1. In the Project View, expand **Modules** and double-click the **CPU**.
2. On the **SD Card** page, click **Browse Contents**.
3. In the Browse SD Card Contents dialog, you may perform the following operations:
    - Extract a file from the SD card to your PC
    - Delete a file
    - Add a file to the SD card from your PC

To perform other tasks, such as moving multiple files at once, creating folders, renaming files, etc., remove the SD card from the RMC200 and connect it to a PC.

Save Controller Image

The SD card can be used to save the controller image. It will be saved as a file named **ControllerImage.bin** in the root folder.

Before saving the controller image to the SD card, make sure the RMC has the desired project in it by downloading the project to the RMC from RMCTools. The controller image is built from the contents of the RMC itself, not from the RMCTools project.

Via RMCTools:

1. In the Project View, expand **Modules** and double-click the **CPU**.
2. On the **SD Card** page, in the **Save/Restore Controller Image** section, click **Save Image to SD Card**.
3. A message box will open. Click **Yes**.
4. When the image has successfully been stored on the SD card, a message box will indicate it is complete. Click **OK**.

   The Event Log will indicate that the controller image was saved to the SD card, or will provide an error message if it failed.

Via the Display Screen:

1. On the Display Screen, browse to **SD Card > Save Image** and press the **Enter** button.
2. Press the **Enter** button again to save the image.
3. The Display Screen will indicate when the save is complete.

   The Event Log will indicate that the controller image was saved to the SD card, or will provide an error message if it failed.

The ability to save or restore the controller image via the display screen can be disabled in the CPU Properties, in the Display Screen Page.

Via Commands:

Use the Save Controller Image (120) command to save the image. The Event Log will indicate that the controller image was saved to the SD card, or will provide an error message if it failed.

When using this command in user programs, you may wish to also access the Controller Image Command State register. See the Save Controller Image (120) command for more details.

Restore Controller Image

If the SD card contains a controller image, that image can be applied to the RMC200. The image must be in the root folder on the SD card as a file named **ControllerImage.bin**. The controller will restart once the image is restored.

Via RMCTools:

1. In the Project View, expand **Modules** and double-click the **CPU**.
2. On the **SD Card** page, in the **Save/Restore Controller Image** section, click **Restore Image from SD Card**.
3. A message box will open warning that all settings will be overwritten and the RMC will restart. Click **Yes.**
4. The image will be applied and the RMC will restart.

   The Event Log will provide an error message if it failed.

Via the CPU Display Screen:

1. On the Display Screen, browse to **SD Card > Restore Image** and press the **Enter** button.
2. Press the **Enter** button again to restore the image and restart the controller.
3. The Display Screen will indicate the image is being restored, then the RMC will restart.
   The Event Log will provide an error message if it failed.

The ability to save or restore the controller image via the display screen can be disabled in the CPU Properties, in the Display Screen Page.

Via Commands:

Use the Restore Controller Image (121) command to restore the image. The controller will restart once the image is restored. The Event Log will provide an error message if it failed.

When using this command in user programs, you may wish to also access the Controller Image Command State register. See the Restore Controller Image (121) command for more details.

See Also

SD Card | CPU40 | CPU20L | Save Controller Image (120) | Restore Controller Image (121) | Controller Image Command State Register

7.4.6. Bases

7.4.6.1. B5L Lite Base (RMC200)

5-Slot Lite Base

The B5L base for the RMC200 Lite supports a CPU20L and 3 I/O modules.

Note: This B5L Lite base that supports the CPU20L should not be confused with the B5 Standard base that supports the CPU40.

Features

- 3 I/O module slots
- First I/O slot (slot 2) includes a high-speed communications bus for specialty modules.
- Mounting lugs

Part Number

The part number of the B5L base is **R200-B5L**.

Specifications

General	
Dimensions	7.0 x 7.9 x 5.8 in. (WxHxD) (177 x 200.0 x 146 mm), with mounting lugs See also RMC200 Mounting Instructions.

Max Power Dissipation	2.0 W
Weight	356 g

Slot Descriptions

	Combined Slot 0/1	Slots 2-4
Possible Modules	CPU20L	A8 CA4 CV8 S8 Q4 U14 D24

See Also

RMC200 Overview | RMC200 Mounting Instructions | B5 Standard Base

7.4.6.2. B7L Lite Base (RMC200)

7-Slot Base

The B7L base for the RMC200 Lite supports a CPU20L and 5 I/O modules.

Note: This B7L Lite base that supports the CPU20L should not be confused with the B7 Standard base that supports the CPU40.

Features

- 5 I/O module slots
- First I/O slot (slot 2) includes a high-speed communications bus for specialty modules.
- Mounting lugs

Part Number

The part number of the B7L base is **R200-B7L**.

Specifications

General	
Dimensions	9.7 x 7.9 x 5.8 in. (WxHxD) (246 x 200.0 x 146 mm), with mounting lugs See also RMC200 Mounting Instructions.
Max Power Dissipation	2.7 W
Weight	485 g

Slot Descriptions

	Combined Slot 0/1	Slots 2-6

Possible Modules	CPU20L	A8 CA4 CV8 S8 Q4 U14 D24

See Also

RMC200 Overview | RMC200 Mounting Instructions | B7 Standard Base

7.4.6.3. B5 Standard Base (RMC200)

5-Slot Base

The B5 base for the RMC200 supports a power supply module, a CPU40, and 3 I/O modules.

Note: This B5 Standard base that supports the CPU40 should not be confused with the B5L Lite base that supports the CPU20L.

Features

- 5 slots
- First I/O slot (slot 2) includes a high-speed communications bus for specialty modules.
- Mounting lugs

Part Number

The part number of the B5 base is **R200-B5**.

Specifications

General	
Dimensions	7.9 x 7.9 x 5.8 in. (WxHxD) (200 x 200.0 x 146 mm), with mounting lugs See also RMC200 Mounting Instructions.
Max Power Dissipation	2.5 W
Weight	367 g

Slot Descriptions

	Slot 0 (Power Supply)	**Slot 1 (CPU)**	**Slots 2-4**
Possible Modules	PS4D	CPU40	A8 CA4 CV8 S8 Q4

			U14 D24

See Also

RMC200 Overview | RMC200 Mounting Instructions | B5L Lite Base

7.4.6.4. B7 Standard Base (RMC200)

7-Slot Base

The B7 base for the RMC200 supports a power supply module, a CPU40, and 5 I/O modules.

Note: This B7 Standard base that supports the CPU40 should not be confused with the B7L Lite base that supports the CPU20L.

Features

- 7 slots
- First I/O slot (slot 2) includes a high-speed communications bus for specialty modules.
- Mounting lugs

Part Number

The part number of the B7 base is **R200-B7**.

Specifications

General	
Dimensions	10.7 x 7.9 x 5.8 in. (WxHxD) (270 x 200.0 x 146 mm), with mounting lugs See also RMC200 Mounting Instructions.
Max Power Dissipation	2.7 W
Weight	512 g

Slot Descriptions

	Slot 0 (Power Supply)	Slot 1 (CPU)	Slots 2-6
Possible Modules	PS4D	CPU40	A8 CA4 CV8 S8 Q4 U14 D24

See Also

RMC200 Overview | RMC200 Mounting Instructions | B7L Lite Base

7.4.6.5. B11 Standard Base (RMC200)

11-Slot Base

The B11 base for the RMC200 supports a power supply module, a CPU40, and 9 I/O modules.

Features

- 11 slots
- First I/O slot (slot 2) includes a high-speed communications bus for specialty modules.
- Mounting lugs

Part Number

The part number of the B11 base is **R200-B11**.

Specifications

General	
Dimensions	16.2 x 7.9 x 5.8 in. (WxHxD) (410 x 200 x 146 mm), with mounting lugs See also RMC200 Mounting Instructions.
Max Power Dissipation	3.0 W
Weight	816 g

Slot Descriptions

	Slot 0 (Power Supply)	Slot 1 (CPU)	Slots 2-10
Possible Modules	PS4D	CPU40	A8 CA4 CV8 S8 Q4 U14 D24

See Also

RMC200 Overview | RMC200 Mounting Instructions

7.4.6.6. B15 Standard Base (RMC200)

15-Slot Base

The B15 base for the RMC200 supports a power supply module, a CPU40, and 13 I/O modules.

Features

- 15 slots
- First I/O slot (slot 2) includes a high-speed communications bus for specialty modules.
- Mounting lugs

Part Number

The part number of the B15 base is **R200-B15**.

Specifications

General	
Dimensions	21.9 x 7.9 x 5.8 in. (WxHxD) (555 x 200 x 146 mm), with mounting lugs See also RMC200 Mounting Instructions.
Max Power Dissipation	3.2 W
Weight	1016 g

Slot Descriptions

	Slot 0 (Power Supply)	Slot 1 (CPU)	Slots 2-14
Possible Modules	PS6D	CPU40	A8 CA4 CV8 S8 Q4 U14 D24

See Also

RMC200 Overview | RMC200 Mounting Instructions

7.4.7. Power Supplies

7.4.7.1. PS4D Power Supply (RMC200)

24 VDC, 35-Watt Power Supply

The PS4D power supply accepts 24 VDC power and provides power to the RMC200 Standard bases. The PS4D supports the B5, B7, and B11 Standard bases. Notice that the B15 base requires the PS6D power supply. The RMC200 Lite bases do not require a power supply module, since the CPU20L Lite has an integrated power supply.

Features

- Handles the B5, B7 or B11 bases
- Voltage over- and under-range LEDs.
- Unpluggable terminal block

Part Number

The part number of the PS4D module is **R200-PS4D.**

Ferrite for Conducted Emissions Compliance

A ferrite ring is required on the power wires for Conducted Emissions compliance. Recommended ferrite is Fair-Rite 0431167281.

Fusing

Fusing is not required for protecting the PS4D. The PS4D has an internal, non-user-accessible, fast-acting 5 A fuse. If an external fuse is installed, it should be a fast-acting fuse less than 5 A, such as 4 A. Fusing may also optionally be installed to protect the wiring.

Specifications

General	
Weight	463 g + 6 g (power connector)
Power	
Input Power	42 W max (1.8A at 24Vdc)
Input Voltage	Recommended 24Vdc ± 15% (20.4 – 27.6 V), 30 V max. Overvoltage shutdown at 36 V.
Output Power	35 W
Functional Isolation	500 VAC
LEDs	
Vin	**Input voltage level indicator:** **Green**: Normal range (20.4 – 27.6 Vdc) **Orange**: Voltage high or low, still operating **Steady Red**: Under- or over-voltage (outside of 18 – 36 V), or reverse voltage, not operating
On	**Output power indicator:** **Off**: Not providing power to base **Green**: Providing power to base
Flt	**Faults indicator:** **Orange**: Temperature high or power draw high, still operating **Flashing Red**: Under- or over-voltage, over power, or over temperature, output shut down **Flashing Green/Red**: Module not plugged into base, output shut down

See Also

RMC200 Overview | PS6D Module | PS4D and PS6D Wiring

7.4.7.2. PS6D Power Supply (RMC200)

24 VDC, 50-Watt Power Supply

The PS6D power supply accepts 24 VDC power and provides power to an RMC200 Standard base. The PS6D is required for the B15 base. Notice that the B5, B7, and B11 bases use the PS4D power supply. The RMC200 Lite bases do not require a power supply module, since the CPU20L Lite has an integrated power supply.

Features

- Handles the B15 base
- Voltage over- and under-range LEDs
- Unpluggable terminal block

Part Number

The part number of the PS6D module is **R200-PS6D.**

Ferrite for Conducted Emissions Compliance

A ferrite ring is required on the power wires for Conducted Emissions compliance. Recommended ferrite is Fair-Rite 0431167281.

Fusing

Fusing is not required for protecting the PS6D. The PS6D has an internal, non-user-accessible, fast-acting 5 A fuse. If an external fuse is installed, it should be a fast-acting fuse less than 5 A, such as 4 A. Fusing may also optionally be installed to protect the wiring.

Specifications

General	
Weight	541 g + 6 g (power connector)
Power	
Input Power	60 W max (2.5A at 24Vdc)
Input Voltage	Recommended 24Vdc ± 15% (20.4 – 27.6 V), 30 V max. Overvoltage shutdown at 36 V.
Output Power	50 W
Functional Isolation	500 VAC
LEDs	
Vin	**Input voltage level indicator:** **Green**: Normal range (20.4 – 27.6 Vdc) **Orange**: Voltage high or low, still operating **Steady Red**: Under- or over-voltage (outside of 18 – 36 V), or reverse voltage, not operating
On	**Output power indicator:** **Off**: Not providing power to base **Green**: Providing power to base
Flt	**Faults indicator:** **Orange**: Temperature high or power draw high, still operating **Flashing Red**: Under- or over-voltage, over power, or over temperature, output shut down **Flashing Green/Red**: Module not plugged into base, output shut down

See Also

RMC200 Overview | PS4D Module | PS4D and PS6D Wiring

7.4.8. CPU Modules

7.4.8.1. CPU20L (RMC200)

The CPU20L is the central processing module for RMC200 Lite backplanes. See also CPU40 Module (RMC200) for the central processing unit for RMC200 Standard backplanes.

Features

- Supports up to 18 control axes and 48 total axes (number of control loops purchased separately and specified by Feature Key)
- Integrated 24 VDC power supply
- 2 Ethernet ports (single IP address) supporting star, linear, and ring topologies.
- High-speed USB 2.0 port (480 Mbps) for communications with RMCTools and RMCLink
- Supported Ethernet Protocols (slave only)
  - EtherNet/IP
  - PROFINET
  - Modbus/TCP
  - CSP (also called DF1 over Ethernet)
  - FINS/UDP (Omron)
  - Procedure Exist (Mitsubishi)
  - Delta Motion Control Protocol
- Display screen with navigation buttons
- Feature Key slot, accessible from the back of the module
- SD Card Slot
- Discrete I/O:
  - 2 Outputs: solid-state relay (SSR) discrete outputs, individually isolated
  - 2 Inputs: 12-24VDC discrete inputs, individually isolated
  - Unpluggable terminal block for discrete I/O

See also Comparing RMC200 Lite and Standard.

Part Number

The part number of the CPU20L is **R200-CPU20L**.

Specifications

General	
Weight	908 g + 2 g (power connector) + 5 g (discrete I/O connector)
Motion Control	
Number of Axes	Up to 18 physical control axes (dependent on backplane size).

Total Axes	Up to 48, including virtual and reference axes.
Control Loop Time	User-selectable 125 µs, 250 µs, 500 µs, 1 ms, 2 ms, 4ms
USB Monitor Port Interface	
Connector	USB "B" receptacle
Data Rate	High-speed (480 Mbps)
Ethernet Interface	
Ports	2 ports (single IP address)
Supported Topologies	Star, linear, or ring
Hardware Interface	IEEE 802.3 for 100BASE-T (twisted pair)
Data Rate	100 Mbps
Duplex	Full Duplex
Features	Auto-negotiation, Auto-crossover (MDI/MDI-X)
Connectors	RJ-45 (2)
Cable	CAT5, CAT5e or CAT6, UTP or STP
Ethernet Configuration	
Configuration parameters	IP address, subnet mask, gateway address, enable/disable ports, auto-negotiation
Configuration methods	BOOTP, DHCP, or static
Ethernet Protocols	
Application protocols (slave only for all)	EtherNet/IP I/O and messaging PROFINET RT (I/O) and data records Modbus/TCP CSP (DF1 over Ethernet) FINS (Omron) Procedure Exist (Mitsubishi) DMCP (Delta Motion Control Protocol)
Framing protocol	Ethernet II
Internet protocol	IP (includes ICMP, ARP, and Address Collision Detection)
Transport protocols	TCP, UDP
Network management protocols	SNMPv1, SNMPv2c, LLDP
Ring management protocols	Device Level Ring (DLR), Media Redundancy Protocol (MRP)
Discrete Inputs (2)	
Input type	12-24 VDC inputs, polarity independent
Logic polarity	True "High"
Functional isolation	500 VAC, individually isolated
Input "High" range	9 to 26.4 VDC, 3 mA maximum
Input "Low" range	0 to 5 VDC, <1 mA
Maximum propagation delay	100 µs Off to On 750 µs On to Off (open collector drive)
Discrete Outputs (2)	
Output type	Solid State Relays (SSR)

Functional isolation	500 VAC, individually isolated
Rated voltage	max ±30 V (DC or peak AC voltage)
Maximum current	±75 mA
Maximum propagation delay	1.5 ms
Logic 1 (True, On)	Low impedance (15 Ω maximum)
Logic 0 (False, Off)	High impedance (<1 nA leakage current at 30V)
SD Card Slot	
Functionality	Save and restore the entire RMC controller image Store general files (e.g machine design documents, data sheets, etc.)
Supported form factor	Standard size (32 mm × 24 mm × 2.1 mm)
Supported families	SD (SDSC) (standard capacity) and SDHC (high capacity)
Supported file system	FAT32 and FAT16
Supported capacities	Up to 32 GB

Power Specifications

Power Input	
Input Power	28 W max (1.2 A at 24 VDC)
Input Voltage	Recommended 24Vdc ± 15% (20.4 – 27.6 V), 30 V max. Overvoltage shutdown at 36 V.
Output Power	19 W max
Power Dissipation	
Max Power Dissipation	9.0 W, including dissipation of power supply

LEDs

The CPU20L LEDs are arranged in the following order:

CPU	**SD**	**In0**
Run		**In1**
En	**Net**	**Out0**
L/A1	**L/A2**	**Out1**

All LEDs Flashing In Unison

If all the CPU20L LEDs are flashing in unison, a PROFINET Flash LED operation is in progress to help identify which physical device corresponds to a certain node.

Light Bar

State	Description
Off	No power to the RMC200 or the module was not inserted when the RMC powered up.
Blue	The module was recognized at power-up.

CPU LED

State	Description
Off	**No power**

Steady Green	**Device operational**
Flashing Red(1 sec period)	**Minor fault** The device has detected a recoverable minor fault. The CPU screen display can be used to view and clear or acknowledge the faults. Examples of faults: • Controller was restarted due to a watchdog timeout of unexpected exception. • Unsupported hardware detected. • Hardware mismatch with project file. • Duplicate IP address detected.
Flashing Red (Slower than 1 sec period)	**Continuous Reset Due to Watchdog Timeout** The controller is resetting due to a watchdog timeout. This is likely a fatal error which requires the CPU20L be sent to Delta. However, it is possible that it is a firmware fault that can be fixed in the field. In this condition the LED will flash slower than the normal 1 second period, and the OLED display will not turn on when the keypad buttons are pressed. The other LEDs will likely be off. Possible recovery method: press and hold both the up and down arrows on the keypad for about 10 seconds. If the cause was a firmware fault, the OLED display will turn on and display a message indicating the controller is running in recovery mode. From the keypad, the controller can be reset to defaults, or from RMCTools, new firmware can be downloaded (even though RMCTools will not fully go online).
Steady Red	**Major fault** The device has detected a non-recoverable fault. This occurs if the controller cannot load firmware or the loader.

RUN LED

State	Description
Off	PROGRAM Mode
Green	RUN Mode

En LED

State	Description
Off	Disabled Mode
Green	Program or Run Mode

SD LED

State	Description
Off	**No SD card** No SD card is present.
Steady Green	**SD Card Present** A compatible card (SDSC or SDHC) is present, and the card is inactive and safe to remove.
Flashing Green	**Read or Write Active - Do Not Remove** The controller is reading from or writing to the SD card.

	Do not remove the SD card while the LED is flashing.
Steady Red	**Incompatible SD Card** A card is present, but is not compatible. Make sure the card is SD (SDSC) or SDHC and has been formatted with a valid file system (FAT32 or FAT16).

Net LED

State	Description
Off	**No power, or no IP address is configured** Either the CPU20L is not powered or has not been configured with an IP address.
Flashing Green	**No Connection Established** **EtherNet/IP mode:** The device has a valid IP address, and has no CIP connection established (I/O or messaging) and no currently timed-out I/O connections. **PROFINET mode:** The device has a valid IP address, but has no PROFINET I/O connection established nor was there a fault that closed the last I/O connection.
Steady Green	**Connection established** **EtherNet/IP mode:** The device has a valid IP address, and has at least one CIP connection established (I/O or messaging) and no currently timed-out I/O connections. **PROFINET mode:** The device has a valid IP address and has a PROFINET I/O connection established.
Flashing Red	**I/O Connection faulted** **EtherNet/IP mode:** An Exclusive Owner I/O connection has timed out. **PROFINET mode:** An I/O connection was closed abnormally, including a timeout or alarm received from the IO controller.
Steady Red	**Duplicate IP address** The device has detected a duplicate IP address on this network.

L/A1, L/A2 (Link/Activity) LEDs

The Link/Act LED reflects the status of the physical Ethernet connection between the RMC and the device on the other end of the Ethernet cable. If this LED is not on or is not blinking when you expect it to be, see the Ethernet Link/Act LED topic for troubleshooting information.

State	Description
Off	**Link down** The Ethernet link is down.
Flashing Green	**Link up, activity** The Ethernet link is up, and activity is detected.
Steady Green	**Link up, idle** The Ethernet link is up, but no activity is detected.
Flashing Amber	**Link up, collision** The Ethernet link is up, but there was a collision.
Steady Amber	**Port disabled** The user has disabled this port.
Steady Red	**Major port fault** There was a major fault initializing the port.

In0 and In1 LEDs

State	Description
Off	**Off** The input is inactive (low).
Orange	**On** The input is active (high).

Out0 and Out1 LEDs

State	Description
Off	**Off** The output is inactive (low).
Yellow	**On** The output is active (high).

Power LEDs

LED	Function
Vin	**Input voltage level indicator:** **Green**: Normal range (20.4 – 27.6 Vdc) **Orange**: Voltage high or low, still operating **Steady Red**: Under- or over-voltage (outside of 18 – 36 V), or reverse voltage, not operating
On	**Output power indicator:** **Off**: Not providing power to base **Green**: Providing power to base
Flt	**Faults indicator:** **Orange**: Temperature high or power draw high, still operating **Flashing Red**: Under- or over-voltage, over power, or over temperature, output shut down **Flashing Green/Red**: Module not plugged into base, output shut down

See Also

RMC200 Overview | CPU40 | Comparing RMC200 Lite and Standard | Display Screen | Using the SD Card

7.4.8.2. CPU40 (RMC200)

The CPU40 is the central processing module for RMC200 Standard backplanes. See also CPU20L Module (RMC200) for the central processing unit for RMC200 Lite backplanes.

Features

- Supports up to 50 control axes and 128 total axes (number of control loops purchased separately and specified by Feature Key)
- 2 Ethernet ports (single IP address) supporting star, linear, and ring topologies.
- High-speed USB 2.0 port (480 Mbps) for communications with RMCTools and RMCLink
- Supported Ethernet Protocols

  - EtherNet/IP
  - PROFINET
  - Modbus/TCP
  - CSP (also called DF1 over Ethernet)
  - FINS/UDP (Omron)
  - Procedure Exist (Mitsubishi)
  - Delta Motion Control Protocol
- Display screen with navigation buttons
- Feature Key slot, accessible from the back of the module
- SD Card Slot
- Discrete I/O:
  - 2 Outputs: solid-state relay (SSR) discrete outputs, individually isolated
  - 2 Inputs: 12-24VDC discrete inputs, individually isolated
  - Unpluggable terminal block for discrete I/O

See also Comparing RMC200 Lite and Standard.

Part Number

The part number of the CPU40 is **R200-CPU40**.

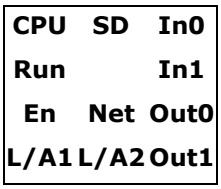
Specifications

General	
Weight	831 g + 5 g (discrete I/O connector)
Motion Control	
Control Axes	Up to 50 physical control axes (dependent on backplane size).
Total Axes	Up to 128, including virtual and reference axes.
Control Loop Time	User-selectable 125 µs, 250 µs, 500 µs, 1 ms, 2 ms, 4ms
USB Monitor Port Interface	
Connector	USB "B" receptacle
Data Rate	High-speed (480 Mbps)
Ethernet Interface	
Ports	2 ports (single IP address)
Supported Topologies	Star, linear, or ring
Hardware Interface	IEEE 802.3 for 100BASE-T (twisted pair)
Data Rate	100 Mbps
Duplex	Full Duplex
Features	Auto-negotiation, Auto-crossover (MDI/MDI-X)
Connectors	RJ-45 (2)
Cable	CAT5, CAT5e or CAT6, UTP or STP
Ethernet Configuration	
Configuration parameters	IP address, subnet mask, gateway address, enable/disable ports, auto-negotiation
Configuration methods	BOOTP, DHCP, or static
Ethernet Protocols	

Application protocols	EtherNet/IP, PROFINET, Modbus/TCP, CSP (DF1 over Ethernet), FINS (Omron) Procedure Exist (Mitsubishi), DMCP (Delta Motion Control Protocol)
Framing protocol	Ethernet II
Internet protocol	IP (includes ICMP, ARP, and Address Collision Detection)
Transport protocols	TCP, UDP
Network management protocols	SNMPv1, SNMPv2c, LLDP
Ring management protocols	Device Level Ring (DLR), Media Redundancy Protocol (MRP)
Discrete Inputs (2)	
Input type	12-24 VDC inputs, polarity independent
Logic polarity	True "High"
Functional isolation	500 VAC, individually isolated
Input "High" range	9 to 26.4 VDC, 3 mA maximum
Input "Low" range	0 to 5 VDC, <1 mA
Maximum propagation delay	100 µs Off to On 750 µs On to Off (open collector drive)
Discrete Outputs (2)	
Output type	Solid State Relays (SSR)
Functional isolation	500 VAC, individually isolated
Rated voltage	max ±30 V (DC or peak AC voltage)
Maximum current	±75 mA
Maximum propagation delay	1.5 ms
Logic 1 (True, On)	Low impedance (15 Ω maximum)
Logic 0 (False, Off)	High impedance (<1 nA leakage current at 30V)
SD Card Slot	
Functionality	Save and restore the entire RMC controller image Store general files (e.g machine design documents, data sheets, etc.)
Supported form factor	Standard size (32 mm × 24 mm × 2.1 mm)
Supported families	SD (SDSC) (standard capacity) and SDHC (high capacity)
Supported file system	FAT32 and FAT16
Supported capacities	Up to 32 GB
Power	
Max Power Dissipation	8.0 W

LEDs

The CPU40 LEDs are arranged in the following order:



All LEDs Flashing In Unison

If all the CPU40 LEDs are flashing in unison, a PROFINET Flash LED operation is in progress to help identify which physical device corresponds to a certain node.

Light Bar

State	Description
Off	No power to the RMC200 or the module was not inserted when the RMC powered up.
Blue	The module was recognized at power-up.

CPU LED

State	Description
Off	**No power**
Steady Green	**Device operational**
Flashing Red(1 sec period)	**Minor fault** The device has detected a recoverable minor fault. The CPU screen display can be used to view and clear or acknowledge the faults. Examples of faults: • Controller was restarted due to a watchdog timeout of unexpected exception. • Unsupported hardware detected. • Hardware mismatch with project file. • Duplicate IP address detected.
Flashing Red (Slower than 1 sec period)	**Continuous Reset Due to Watchdog Timeout** The controller is resetting due to a watchdog timeout. This is likely a fatal error which requires the CPU40 be sent to Delta. However, it is possible that it is a firmware fault that can be fixed in the field. In this condition the LED will flash slower than the normal 1 second period, and the OLED display will not turn on when the keypad buttons are pressed. The other LEDs will likely be off. Possible recovery method: press and hold both the up and down arrows on the keypad for about 10 seconds. If the cause was a firmware fault, the OLED display will turn on and display a message indicating the controller is running in recovery mode. From the keypad, the controller can be reset to defaults, or from RMCTools, new firmware can be downloaded (even though RMCTools will not fully go online).
Steady Red	**Major fault** The device has detected a non-recoverable fault. This occurs if the controller cannot load firmware or the loader.

RUN LED

State	Description

Off	PROGRAM Mode
Green	RUN Mode

En LED

State	Description
Off	Disabled Mode
Green	Program or Run Mode

SD LED

State	Description
Off	**No SD card** No SD card is present.
Steady Green	**SD Card Present** A compatible card (SDSC or SDHC) is present, and the card is inactive and safe to remove.
Flashing Green	**Read or Write Active - Do Not Remove** The controller is reading from or writing to the SD card. Do not remove the SD card while the LED is flashing.
Steady Red	**Incompatible SD Card** A card is present, but is not compatible. Make sure the card is SD (SDSC) or SDHC and has been formatted with a valid file system (FAT32 or FAT16).

Net LED

State	Description
Off	**No power, or no IP address is configured** Either the CPU40 is not powered or the CPU40 has not been configured with an IP address.
Flashing Green	**No Connection Established** **EtherNet/IP mode:** The device has a valid IP address, and has no CIP connection established (I/O or messaging) and no currently timed-out I/O connections. **PROFINET mode:** The device has a valid IP address, but has no PROFINET I/O connection established nor was there a fault that closed the last I/O connection.
Steady Green	**Connection established** **EtherNet/IP mode:** The device has a valid IP address, and has at least one CIP connection established (I/O or messaging) and no currently timed-out I/O connections. **PROFINET mode:** The device has a valid IP address and has a PROFINET I/O connection established.
Flashing Red	**I/O Connection faulted** **EtherNet/IP mode:** An Exclusive Owner I/O connection has timed out. **PROFINET mode:** An I/O connection was closed abnormally, including a timeout or alarm received from the IO controller.
Steady Red	**Duplicate IP address** The device has detected a duplicate IP address on this network.

L/A1, L/A2 (Link/Activity) LEDs

The Link/Act LED reflects the status of the physical Ethernet connection between the RMC and the device on the other end of the Ethernet cable. If this LED is not on or is not blinking when you expect it to be, see the Ethernet Link/Act LED topic for troubleshooting information.

State	Description
Off	**Link down** The Ethernet link is down.
Flashing Green	**Link up, activity** The Ethernet link is up, and activity is detected.
Steady Green	**Link up, idle** The Ethernet link is up, but no activity is detected.
Flashing Amber	**Link up, collision** The Ethernet link is up, but there was a collision.
Steady Amber	**Port disabled** The user has disabled this port.
Steady Red	**Major port fault** There was a major fault initializing the port.

In0 and In1 LEDs

State	Description
Off	**Off** The input is inactive (low).
Orange	**On** The input is active (high).

Out0 and Out1 LEDs

State	Description
Off	**Off** The output is inactive (low).
Yellow	**On** The output is active (high).

See Also

RMC200 Overview | RMC200 CPU20L | Comparing RMC200 Lite and Standard | Display Screen | Using the SD Card

7.4.8.3. CPU20L and CPU40 Display Screen

The display screen on the RMC200 CPU20L and CPU40 modules can be used to access the information and features listed below.

- **Faults**
  Lists any current hardware-related faults.
- **Ethernet IP Address**
  View and change the IP address settings and Ethernet_Protocol_Mode. The IP settings include subnet mask and default gateway, and also list the MAC address. To prevent changing the IP settings from the Display Screen, see the Ethernet Settings Page in the CPU Properties dialog.

- **Controller Name**
  Viewing the controller name on the Display Screen can be used to identify an RMC on a network containing many RMCs. The Controller Name is also visible from within RMCTools when browsing for RMCs.
- **Date/Time**
  Displays the RMC's date and time. The RMC has a real-time clock.
- **Feature Key**
  List the features on the Feature Key.
- **SD Card**
  Save or restore the controller image. See Using the SD Card for details.
- **Module Info**
  Lists the module type, module version, and module firmware version for each module in the backplane.

Navigating the Display Screen

Use the buttons to navigate the Display Screen menu:

	Action
	Move up, down, left or right. The right arrow will also move forward a level. The left arrow will also move back a level.
	Go forward a level or accept an edit.
	Go back a level.

Disabling Display Screen Features

The following Display Screen features may be disabled to prevent unauthorized users from making changes to the controller or making a copy of the controller contents:

- Changing the IP Address
- Saving the Controller Image to the SD card
- Restoring the Controller Image from the SD card

To disable features:

1. In the **Project** pane, expand the **Modules** folder, double-click the **CPU** module and click **Display Screen**.
2. Uncheck the features you wish to disable.
3. Click **OK**. The changes will be applied to the controller if RMCTools is online with the controller.
4. After making changes to these settings, make sure to Update Flash.

Screen Saver

The Display Screen has a screen saver that will automatically turn off after 60 seconds. Pressing any button will turn on the display again.

Disabling the Screen Saver:

- Press and hold the **Menu** button for 3 seconds. The display will show "Screen saver OFF."

Re-enabling the Screen Saver:

- Press and hold the **Menu** button for 3 seconds. The display will show "Screen saver ON."

The screen saver on/off setting is not saved in the controller through a power cycle.

See Also

CPU20L Module | CPU40 Module | Feature Key | Using the SD Card

7.4.8.4. RMC200 Real-Time Clock

The RMC200 CPU20L and CPU40 include a real-time clock. A real-time clock is a part of the RMC's circuitry that keeps track of time of day and date while taking into account time changes for the specified time zone region, such as daylight savings time. The real-time clock runs off a battery when the RMC is not powered, which allows the RMC to keep track of real time through power cycles.

When the RMC powers up, the RMC's real-time data registers take on the value of the real-time clock. While the RMC is powered, the real-time data registers increment the time according to the RMC's internal clock frequency–not according to the real-time clock.

Notice that the RMC's real-time clock is not synchronized in any way to network time or GPS time and will drift in the same way that a watch or wall clock will drift. Therefore, to keep accurate time, the real-time clock time must occasionally be adjusted using RMCTools.

In addition to the real-time registers that are set by the real-time clock at power-up, the RMC200 also has System Time registers that only record the time since the RMC powered up. While the RMC is powered, both the real time registers and system time registers increment according to the RMC's internal clock frequency. Most applications in the RMC that need to keep track of elapsed time should use the system time registers, not the real-time registers. This is because the time in the real-time registers may jump due to events such as daylight savings time or the user adjusting the real time. See the System Time topic for a list of the real-time registers and system time registers.

Viewing the Real-Time Clock

In RMCTools:

1. In the Project Pane, expand **Modules** and double-click the CPU module.
2. Click the **Date and Time** page.

On the Display Screen:

1. On the display screen, browse to **Date/Time**.

In Registers:

1. See the System Time Registers for the tag names and addresses of the registers that contain the real time.

Setting the Real-Time Clock from RMCTools

The Real-time clock can be set manually or set according to the time on the PC that is running RMCTools. For advanced users, it can be set via a CIP Service as described at the end of this topic.

1. In the Project Pane, expand **Modules** and double-click the CPU module.
2. Click the **Date and Time** page.

Set from PC:

a. In the **Set From PC** section, click **Set Controller Data and Time**.

Set Manually:

a. In the Set Manually section, select the time and time zone.
b. Click **Set Controller Data and Time**.

3. Click **OK** and save your project file to store the time zone in the project file.

Make sure to also set the time zone, and if necessary, choose the proper region within the time zone for the machine location. Some regions use daylight savings time and others do not, even within the same general time zone.

Updating Time Zones

As part of the real-time clock, the RMC stores the selected time zone definition, which includes the offset from the UTC and the daylight savings time definition. These definitions can change over time, such as countries removing or adjusting daylight savings time. Delta updates the RMCTools software annually with the latest UTC time zone definitions. RMCTools can then download the updated time zone definition to the RMC.

To update the RMC with the latest time zone information:

1. Download the most recent version of RMCTools from Delta's website.
2. Open RMCTools and go online with the RMC.
3. In the Project view, expand **Modules,** and double-clock the RMC200 CPU.
4. On the **Date and Time** page, in either the **Set From PC** or **Set Manually** section, verify that the correct time zone is shown, and click **Set Controller Date and Time**.
5. Click **OK** and save your project file to store the time zone in the project file.

Real-Time Clock Battery

A battery supplies power to the Real-Time Clock when the RMC is not powered. This battery is designed to last at least 10 years when the RMC is not externally powered. The battery is not field-replaceable. To replace the battery, the CPU module must be sent to Delta.

Real Time Clock Battery Low Warning

If the battery voltage drops below 2.5V, the warning message "Real time clock battery is low" will appear in the Event Log and on the Display Screen. If this occurs, the real-time clock values may be inaccurate.

If the real-time clock functionality is not used by the application, the battery low warning can be safely ignored. At the time of this writing (2023), very few applications use the time value derived from the real-time clock. Of the applications that use any of the time registers in the RMC, most use the system time (time since the RMC powered up), not the real-time registers.

Real-Time Clock Internal Fault

Errors related to the real-time clock are reported on the display screen and in the Event Log. The "Real time clock has an internal fault" fault condition occurs when any of the following occur:

- Communication error reading or writing registers in the RTC chip.
- The RTC chip did not accept the configuration written to it.
- The RTC oscillator has stopped unexpectedly.

If these errors occur, contact Delta technical support. However, if the real-time clock is not used in the application, this error can be safely ignored.

Setting the Real-Time Clock via CIP Service

Devices that support CIP communications can set the RMC's real-time clock. Allen-Bradley PLCs support CIP via the MSG instruction.

The following CIP service sets the real time clock in the RMC:

Service: Set_Attribute_Single (10 hex)

Object: C7 hex

Instance: 1

Attribute: 6

Source Element: SystemTime (type = DINT[2])

SystemTime[0] – Real Time UTC, seconds

SystemTime[1] – Real Time, nanoseconds

Seconds and nanoseconds are relative to 1970-01-01 00:00:00Z (midnight, January 1, 1970) and are in UTC (GMT). Nanoseconds are in the range of 0..999,999,999, wrapping every second. Seconds must be at least 1262304000 (2010-01-01T00:00:00Z). The **Real Time Local**, **seconds** value will automatically be updated for the currently-selected time zone.

Source Length: 8 bytes

Notice that Allen-Bradley PLCs use a microsecond time base and combines two 32-bit values into a 64-bit value, whereas the RMC uses a nanosecond time base and rolls over the nanoseconds 32-bit word at 1,000,000,000.

See Also

CPU20L Module | CPU40 Module | System Time Registers | System Time

7.4.9. I/O Modules

7.4.9.1. CA4 Module (RMC200)

4 Voltage or Current Outputs, with Fault Inputs and Enable Outputs

The CA4 module for the RMC200 provides 4 analog outputs intended for controlling an actuator. With individually configurable ±10 VDC, ±20 mA, or 4-20 mA, the CA4 can connect directly to a wide range of actuators. Each analog output has an associated Fault Input and Enable Output.

Features

- 4 analog outputs: ±10 VDC, ±20 mA, or 4-20 mA, individually configurable
- Supports custom ranges within the ±10 VDC and ±20 mA ranges, such as 0-10 V, 0-5 V, 1-5 V, etc.
- 18 bit resolution
- (4) 24 VDC Enable outputs
- (4) 24 VDC Fault Inputs
- Enable inputs and Fault outputs may be used as general-purpose discrete I/O
- Unpluggable terminal blocks

Part Number

The part number of the CA4 module is **R200-CA4.**

Setting Up the CA4 Module

Wiring

See CA4 Wiring.

Axis Definitions

In order to use a CA4 analog output, it must be assigned to an RMC internal software axis, via the Axis Definitions Dialog.

Output Parameters

After a CA4 analog output has been assigned to an axis, the following axis parameters must be configured:

- Output Type

Specifications

General	
Weight	393 g + 20 g (2 connectors)
Control Output (4 per module)	
Range	Voltage mode: ± 10 V @ 15 mA (670 Ω or greater load) Current mode: ± 20 mA @ 10 V (500 Ω or lower load)
Tolerance at full output	Voltage mode: ± 5 mV at 10 V Current mode: ± 10 μA at 20 mA
Resolution	18 bits
Hardware Output Filter	First-order filter, time constant 50 μsec
Functional Isolation	500 VAC
Overload protection	Continuous short to common
Overvoltage protection	Outputs are protected by clamp diodes
Enable Output (4 per module)	
Output type	Solid State Relay
Logic polarity	User selectable to Active Open or Active Closed
Functional Isolation	500 VAC
Rated voltage	max ±30 V (DC or peak AC voltage)
Maximum current	±75 mA
Maximum propagation delay	2.5 ms
Closed	Low impedance (12 Ω maximum)
Open	High impedance (<25 nA leakage current at 30 V)
Fault Input (4 per module)	
Input characteristics	12-24 VDC; polarity independent
Logic polarity	User selectable to Active Input "High" or Active Input "Low" (Open when module not powered)
Functional Isolation	500 VAC
Input "High" range	9 to 26.4 VDC (polarity independent), 3 mA maximum
Input "Low" range	0 to 5 VDC (polarity independent), <1 mA
Maximum propagation delay	100 μs Off to On 750 μs On to Off (open collector drive)
Power	
Max Power Dissipation	1.4 W, all analog outputs in voltage mode 2.8 W, all analog outputs in current mode

LEDs

Light Bar

State	Description

Off	No power to the RMC200 or the module was not inserted when the RMC200 powered up.
Blue	The module was recognized at power-up.

Control Output LEDs: Out0 .. Out3

State	Description
Off	**Not assigned** The output is not assigned to an axis.
Green	**Operating normally** The axis is operating normally with no latched errors.
Red	**Error** The axis has at least one error bit set that has halted the axis.
Amber	**Simulate mode** The axis is in simulate mode.

Enable Output LEDs: En0 .. En3

State	Description
Off	**Enable Output inactive** The Enable Output is currently inactive. If configured as an axis Enable Output, the output may be Active Closed or Active Open, but when a general purpose output, it will always be Active Closed.
Green	**On - as Enable Output** The Enable Output is configured as an axis Enable Output and is currently active (may be configured as Active Closed or Open).
Yellow	**On - as general-purpose output** The Enable Output is configured as a general-purpose output, and the output is currently closed (active).

Fault Input LEDs: Flt0 .. Flt3

State	Description
Off	**Fault Input inactive** The Fault Input is inactive (logically off). If configured as an axis Fault Input, the input may be active high or active low, but when a general purpose input, it will always be active high.
Red	**On - as Fault Input** The Fault Input is configured as an axis Fault Input, and the input is currently active (may be configured as Active High or Active Low).
Orange	**On - as general-purpose input** The Fault Input is configured as a general-purpose input, and the input is currently high (active).

See Also

RMC200 Overview | CA4 Wiring

7.4.9.2. CV8 Module (RMC200)

8 Voltage Outputs, with 8 Discrete I/O

The CV8 module for the RMC200 provides 8 voltage outputs intended for controlling an actuator. The module includes 8 discrete I/O that can be individually configured as general-purpose inputs, general-purpose outputs, Fault Inputs, or Enable Outputs.

Features

- 8 voltage outputs: ±10 VDC
- Supports custom ranges within the ±10 VDC range, such as 0-10 V, 0-5 V, 1-5 V, etc.
- 18 bit resolution
- 8 individually-configurable discrete I/O, 12-24 VDC input or Solid State Relay output
- Discrete I/O may be used as Enable inputs or Fault outputs
- Unpluggable terminal blocks

Part Number

The part number of the CV8 module is **R200-CV8.**

Setting Up the CV8 Module

Wiring

See CV8 Wiring.

Axis Definitions

In order to use a CV8 analog output, it must be assigned to an RMC internal software axis, via the Axis Definitions Dialog.

Output Parameters

After a CV8 analog output has been assigned to an axis, the following axis parameters must be configured:

- Output Type

Specifications

General	
Weight	397 g + 26 g (2 connectors)
Control Output (8 per module)	
Range	± 10 V @ 5 mA (2000 Ω or greater load)
Tolerance at full output	± 5 mV at 10 V
Resolution	18 bits
Hardware Output Filter	First-order filter, time constant 75 µsec
Functional Isolation	500 VAC
Overload protection	Continuous short to common
Overvoltage protection	Outputs are protected by clamp diodes
Discrete Outputs (up to 8 per module)	
Output type	Solid State Relay, individually isolated
Logic polarity	User selectable to Active Open or Active Closed
Functional Isolation	500 VAC
Rated voltage	max ±30 V (DC or peak AC voltage)
Maximum current	±75 mA

Maximum propagation delay	2.5 ms
Closed	Low impedance (12 Ω maximum)
Open	High impedance (<25 nA leakage current at 30 V)
Discrete Inputs (up to 8 per module)	
Input characteristics	12-24 Vdc; polarity independent, individually isolated
Logic polarity	User selectable to Active Input "High" or Active Input "Low" (Open when module not powered)
Functional Isolation	500 VAC
Input "High" range	9 to 26.4 Vdc (polarity independent), 3 mA maximum
Input "Low" range	0 to 5 Vdc (polarity independent), <1 mA
Maximum propagation delay	100 μs Off to On 750 μs On to Off (open collector drive)
Power	
Max Power Dissipation	2.0 W

LEDs

Light Bar

State	Description
Off	No power to the RMC200 or the module was not inserted when the RMC200 powered up.
Blue	The module was recognized at power-up.

Control Output LEDs: Out0 .. Out7

State	Description
Off	**Not assigned** The output is not assigned to an axis.
Green	**Operating normally** The axis is operating normally with no latched errors.
Red	**Error** The axis has at least one error bit set that has halted the axis.
Amber	**Simulate mode** The axis is in simulate mode.

Discrete I/O LEDs: D0 .. D7

State	Description
Off	**I/O point is off** If this I/O point is configured as an input, then the input is currently low. If this I/O point is configured as an output, then the output is currently open.
Orange	**Input is on** This I/O point is configured as an input, and the input is currently high.
Yellow	**Output is on** This I/O point is configured as an output, and the output is currently closed.

See Also

RMC200 Overview | CV8 Wiring

7.4.9.3. LC8 Module (RMC200)

8 Load Cell Inputs, ±5 mV/V, with Sense Input

The LC8 module is an 8-input load cell module for the RMC200. Each input supports a full Wheatstone bridge and provides a 6.75V excitation voltage to the bridge.

The 6.75 V excitation is intended to work with 350 Ohm load cells. Load cells with lower resistance are supported, as long as the total excitation current per terminal block does not exceed 80 mA.

Quarter and half bridges are supported with a customer-supplied bridge completion circuit.

External customer-supplied excitation voltages may be used as long as the Max Differential Input (±34.25 mV) is not exceeded and the input voltage at In+ or In- relative to -Exc is within the Input Voltage Range (0.6 V to 6.15 V typical).

Features

- 8 load cell inputs
- 4 or 6-wire load cells
- Wire voltage drop compensation
- Error detection
- Unpluggable terminal blocks
- 6.75 V excitation voltage (also supports customer-supplied excitation)
- Selectable filter frequency (module-wide setting plus per-channel setting)
- Overvoltage protection

Part Number

The part number of the LC8 module is **R200-LC8**.

Setting Up the LC8 Module

Wiring

See LC8 Wiring.

Module-wide Settings

For some high-speed applications, the Minimum Input Filter Frequency may need to be adjusted.

Axis Definitions

In order to use an LC8 input, it must be assigned to an RMC internal software axis via the Axis Definitions Dialog.

Wire Sense

The LC8 offers two methods of compensating for the voltage drop in the Exciter+ and Exciter- wires:

- **Adaptive:**
  The LC8 periodically measures the voltage at the Sense pin and determines the voltage drop on the Exc- wire. It assumes the voltage drop of the Exc+ and Exc- wires is the same. Therefore, the wire length and gauge of the Exc+ and Exc- wires must be identical.

Available only for 6-wire load cells.

To choose this option, set the **Exciter Mode** axis parameter to **Adaptive**.

- **Fixed Value:**
  The user can use a voltmeter to measure the exciter voltage at the load cell and enter the value into the Fixed Exciter Voltage axis parameter.

  This option is available for 4-wire and 6-wire load cells.

  To choose this option, set the **Exciter Mode** axis parameter to **Fixed Value**.

See Exciter Mode for more details.

Feedback Parameters

In addition to the Exciter Mode, the following axis parameters may need to be set:

- Load Cell Overflow Limit
- Load Cell Underflow Limit

Scale/Offset

After configuring the axis feedback parameters, the feedback value must be scaled to practical measurement units:

Load Cell Scaling

Specifications

General	
Weight	379 g + 26 g (2 connectors)
Load Cell Inputs	
Inputs	Eight 24-bit load cell inputs
Overvoltage Protection	±24 V, momentary
Input range	±33.75 mV (5 mV/V with 6.75 V excitation)
Max differential input	±34.25 mV (5.075 mV/V with 6.75 V excitation)
Input voltage range	In+ or In- relative to -Exc: 0.6 V to 6.15 V typical
Input impedance	5 MΩ
Input step response	70% in 2 samples times, 100% in 3 samples times
Sampling frequency	8 kHz max
Sampling filter	150 Hz to 2.4 kHz, based on sampling frequency
Offset drift with temperature	±40 nV/V/ºC typical
Gain drift with temperature	-0.005%/ºC (-50 ppm/ºC) typical
Non-linearity	±15 ppm of Full Scale Range typical
Exciter output	6.75 Vdc ± 2 mV typical. 80 mA max total of all exciter outputs per terminal block.
Sense Input	100 mV max range (relative to In-), 50 µV resolution
Common	
Max Power Dissipation	1.4 - 2.4 W, depending on use of Exciter Output
Max Power Consumption	1.3 W with no Exciter Output, 3.2 W with Exciter Output

Millivolt/Volt Calibration

Each LC8 input reports a millivolt/volt value. This is calculated as follows:

Millivolts/Volt = Millivolt Input / Exciter Voltage

The LC8 factory calibration process precisely measures the Exciter Voltage which is in the range 6.750 ± 0.003 V. The input is calibrated using this measured Exciter Voltage value and a 2 mV/V load cell simulator. This measured Exciter Voltage is also used as the Exciter Voltage value when the Exciter Mode axis parameter is set to **Nominal**. Therefore, with the **Nominal** mode selected, the user will see the Effective Exciter Voltage axis status register report a value that is likely not exactly 6.750 V, but rather the value measured during factory calibration.

The LC8 is calibrated in a room temperature environment, with the module being powered up at least 30 minutes before calibration. The internal reported temperature of the module during calibration is typically between 28 to 37ºC.

LEDs

Light Bar

State	Description
Off	No power to the RMC200 or the module was not inserted when the RMC200 powered up.
Blue	The module was recognized at power-up.

Input LEDs: In0 .. In7

Each load cell input has an associated input LED:

LED State	Description
Off	The input is not assigned to an axis.
Green	**Operating normally** The input is assigned to an axis and is receiving a valid value from the load cell.
Red	**Error** The input is assigned to an axis and is not receiving a valid value from the load cell. This may be due to a load cell error, or due to incorrectly configured axis feedback parameters. Load cell errors that cause an invalid value on the LC8 include: • The Millivolts/Volt value is greater than the Load Cell Overflow Limit or less than the Load Cell Underflow Limit, or the Millivolt Input is greater than 34.5 mV or less than -34.5 mV. • The In- or In+ voltage is outside of the range of 0.6 to 6.15 V.
Amber	**Simulate mode** The axis is in simulate mode.

See Also

RMC200 Overview | LC8 Wiring | Load Cell Fundamentals | Load Cell Scaling

7.4.9.4. S8 Module (RMC200)

8 SSI or Magnetostrictive Start/Stop or PWM inputs, supports one RS-422 quadrature input

The S8 module for the RMC200 provides eight inputs, individually software selectable as SSI (Synchronous Serial Interface), or magnetostrictive Start/Stop or PWM inputs. One RS-422 quadrature input (A+, A-, B+, B-) may be configured in software, using inputs 6 and 7 on the S8 module. Inputs 6 and 7 may be configured as a single SSI Monitor input.

Note: Linear magnetostrictive transducers with SSI output should be of the synchronized type. Non-synchronized is not well-suited for motion control. This does not apply to rotary SSI encoders.

Features

- 8 inputs, individually configurable for SSI, Start/Stop, or PWM
- Binary or Gray Code
- Rotary encoders or linear transducers
- 8 to 32 bits of data
- Differential RS-422 SSI interface
- Unpluggable terminal blocks
- Supports one RS-422 quadrature input using channels 6 and 7, with A+, A-, B+, B- signals only. Does not support homing or registration.
- Supports one SSI Monitor input using channels 6 and 7.

Part Number

The part number of the S8 module is **R200-S8**.

Setting Up the S8 Module

Wiring

See S8 Wiring.

Configuring an RS-422 Quadrature Input or SSI Monitor

The S8 SSI/MDT inputs 6 and 7 can be configured as an RS-422 quadrature input or an SSI Monitor input. See Configuring S8 Channels for details.

Axis Definitions

In order to use an S8 input (SSI, MDT, or quadrature), use the Axis Definitions dialog to assign it to an RMC internal software axis.

Feedback Parameters

After an S8 input has been assigned to an axis, the following axis parameters must be configured in Axis Tools in order for the S8 to communicate with the sensor:

SSI Transducers - Required	MDT Transducers - Required	Quadrature Encoders - Required
SSI/MDT Feedback Type SSI Data Bits SSI Format	SSI/MDT Feedback Type MDT Type	Linear/Rotary
Optional	**Optional**	
Linear/Rotary SSI Clock Rate Wire Break Detection SSI Overflow Mode SSI Home Source	MDT Blanking Period	AB Termination

SSI Termination SSI_High_Bits_to_Ignore SSI_Low_Bits_to_Ignore		

Scale/Offset

After configuring the axis feedback parameters, the feedback value must be scaled to practical measurement units:

MDT Scaling

SSI Scaling

Quadrature Scaling

Specifications

General	
Weight	386 g + 26 g (2 connectors)
SSI Interface	
Data Inputs	RS-422 differential
Clock Outputs	RS-422 differential
Termination	Software selectable data input impedance: 110 Ω or >200 kΩ
Clock Frequency	User-selectable 100 kHz to 2500 kHz
Maximum Cable Length	Transducer Dependent, approx. 3-2100 ft. See the SSI Clock Rate topic for details.
Resolution	Transducer dependent
Count Encoding	Binary or Gray Code
Count Data Length	8 to 32 bits
Bit Masking	A selectable number of high or low bits may be masked
Additional Settings	Selectable overflow modes to conform to various SSI transducers Wire break detection
MDT Interface (Start/Stop or PWM)	
Transducer interface types	MDT with Start/Stop or PWM (Pulse Width Modulated) feedback
Interrogation Outputs	RS-422 differential (transducer must be configured for external interrogation)
Return Inputs	RS-422 differential
Resolution	0.0005 in. with one recirculation
Count Rate	240 MHz
Recirculations	Supports multiple recirculations only for PWM transducers with internal recirculations.
Maximum transducer length	440 in. at 4ms (loop-time dependent)
Quadrature Interface	
Input	5V differential (RS-422) receiver for A+, A-, B+, B-
Connection	Uses the following pins: A+: Input 6 Ret/Dat+ A- : Input 6 Ret/Dat- B+: Input 7 Ret/Dat+

	B- : Input 7 Ret/Dat-
Input Impedance	Software selectable data input impedance: 110 Ω or >200 kΩ
Max Encoder Frequency	8,000,000 quadrature counts/second
Min Edge Alignment	55 ns time between A edge and B edge
Common	
Functional Isolation	500 VAC
Power	
Max Power Dissipation	1.8 W

LEDs

Light Bar

State	Description
Off	No power to the RMC200 or the module was not inserted when the RMC200 powered up.
Blue	The module was recognized at power-up.

Input LEDs: In0 .. In7

Each transducer input has an associated input LED:

LED State	Description
Off	The input is not assigned to an axis.
Green	**Operating normally** The input is assigned to an axis and is receiving a valid value from the transducer.
Red	**Error** The input is assigned to an axis and is not receiving a valid value from the transducer. This may be due to a transducer error, or due to incorrectly configured axis feedback parameters. Transducer errors that cause an invalid value on the S8 include: • For SSI, there is a Wire Break condition and Wire Break Detection is enabled. • For SSI, an SSI Overflow Pattern was detected. • For Start/Stop or PWM, there was no start pulse detected or rising edge of the PWM response. • For Start/Stop or PWM, there was no stop pulse or falling edge of the PWM response in the required time.
Amber	**Simulate mode** The axis is in simulate mode. **Note:** The input LED may appear to be amber for certain transducer errors when the RMC is alternately seeing a valid and invalid input, and the LED is switching between red and green colors quickly enough that the LED appears to be amber.

See Also

RMC200 Overview | S8 Wiring | MDT Fundamentals | SSI Fundamentals | Quadrature Fundamentals

7.4.9.5. Configuring S8 Channels

The S8 module offers standard MDT or SSI for all its inputs. In addition, channels 6 and 7 can be configured as one of the following options:

- **One Quadrature Input**
  A single quadrature input that uses SSI inputs 6 and 7.
- **One SSI Monitor Input**
  A single SSI Monitor input that uses SSI inputs 6 and 7.

These options must be configured before being assigned to axis inputs. To configure the channels, open the S8 properties dialog:

1. In the Project Pane, expand the **Modules** folder
2. Double-click the desired S8 module and choose **Configuration**.
3. In the **Channel 6/7 Operating Mode** section, choose the desired mode and click **OK**. See the sections below for details one each mode.

Two SSI/MDT Inputs

SSI inputs 6 and 7 will be standard inputs that support MDT or SSI.

Quadrature Input

The **One Quadrature Input** option will result in one RS-422 quadrature input (A+, A-, B+, B-) that uses the S8 inputs 6 and 7.

The I/O pins on the S8 module for these channels are redefined as:

Pin Label	Function
Ret/Dat6+	A+
Ret/Dat6-	A-
Int/Clk6+	unused
Int/Clk6-	unused
Ret/Dat7+	B+
Ret/Dat7-	B-
Int/Clk7+	unused
Int/Clk7-	unused

Once inputs 6 and 7 are configured as a quadrature input, you can use that quadrature input when defining axes in the Axis Definitions dialog.

SSI Monitor Input

The **One SSI Input** option will result in one SSI Monitor input that uses the S8 inputs 6 and 7.

The I/O pins on the S8 module for these channels are redefined as:

Pin Label	Function
Ret/Dat6+	Data Input +
Ret/Dat6-	Data Input -
Int/Clk6+	unused

Int/Clk6-	unused
Ret/Dat7+	Clock Input +
Ret/Dat7-	Clock Input -
Int/Clk7+	unused
Int/Clk7-	unused

Once inputs 6 and 7 are configured as an SSI Monitor input, you can use that input when defining axes in the Axis Definitions dialog.

See Also

S8 Module (RMC200) | SSI Fundamentals | MDT Fundamentals | Quadrature Fundamentals

---

7.4.9.6. A8 Module (RMC200)

8 Analog Inputs

The A8 module is an 8-input analog voltage or current module for the RMC200. Each input supports ±10V, 4-20 mA, or ±20 mA.

Features

- 8 differential inputs
- Inputs individually configurable for ±10V, 4-20 mA, or ±20 mA
- 18 bits resolution ADCs with oversampling for increased effective resolution and noise reduction
- 200 kHz internal sampling rate
- Broken wire detection
- Can be used for position, velocity, pressure, and force inputs.
- Unpluggable terminal blocks
- +10 VDC exciter for potentiometers (not for powering transducers)

Part Number

The part number of the A8 module is **R200-A8**.

Setting Up the A8 Module

Wiring

See A8 Wiring.

Axis Definitions

In order to use an A8 input, it must be assigned to an RMC internal software axis, via the Axis Definitions Dialog.

Feedback Parameters

After an A8 input has been assigned to an axis, the following axis parameters must be configured:

- Analog Input Type
- Analog Overflow Limit
- Analog Underflow Limit

The Analog Input Filter also applies to the A8, but typically need not be changed.

Scale/Offset

After configuring the axis feedback parameters, the feedback value must be scaled to practical measurement units:

Analog Position Scaling

Analog Velocity Scaling

Analog Acceleration Scaling

Analog Pressure/Force Scaling

Recovering From Saturation

When an analog input on the A8 module is driven beyond the -10.5 to 10.5 V range, the input is said to be *saturated*. This also occurs when the input is unconnected, since internal biasing circuitry will drive the input fully negative so that the RMC can detect that the input is disconnected. After being saturated, the input may take up to 5 seconds to fully recover once a voltage in the valid range is applied. During this time, a small offset approximately on the order of 300 microvolts may persist.

Specifications

General	
Weight	399 g + 26 g (2 connectors)
Analog Input Interface	
Inputs	Eight 18-bit differential inputs (higher resolution obtained by oversampling)
Functional Isolation	500 VAC
Overvoltage protection	±24 V
Nominal Input Ranges	±10 V, 4-20 mA, ±20 mA (each input independently configurable)
Max Differential Ranges	Voltage: -10.5 V to +10.5 V Current: -20 mA to +20 mA (continuous), -25 mA to +25 mA (peak)
Max Input Voltage Range	In+ or In- relative to Cmn: -14 V to +14 V typical
Broken wire detection	When the input is not connected, internal biasing pulls the input voltage down to its full negative value.
Input impedance	Voltage input: 1 MΩ Current input: 250 Ω
Input filter slew rate	25 V/ms
Sampling frequency	200 kHz internal sampling. Provides one filtered sample per control loop (e.g. 1 msec) to CPU.
Sampling filter	250 Hz – 4 kHz, user-selectable internal low-pass sampling filter.
Offset drift with temperature	0.2 LSB/°C typical (±10 V range)
Gain drift with temperature	20 ppm/°C typical (±10 V range)
Non-linearity	12 LSB (counts) typical (±10 V range)
Exciter output	10 Vdc ± 2%, 40 mA max total of all exciter outputs per terminal block
Power	

Max Power Dissipation	1.4-2.4 W, depending on use of Exciter Output

LEDs

Light Bar

State	Description
Off	No power to the RMC200 or the module was not inserted when the RMC200 powered up.
Blue	The module was recognized at power-up.

Input LEDs: In0 .. In7

Each transducer input has an associated input LED:

State	Description
Off	The input is not assigned to an axis.
Green	**Operating normally** The input is assigned to an axis and is receiving a valid value from the transducer.
Red	**Error** The input is assigned to an axis and is not receiving a valid value from the transducer. This may be due to a transducer error, or due to incorrectly configured axis feedback parameters. Transducer errors that cause an invalid value include: • The differential analog input signal is outside the user-specified Overflow/Underflow Limit parameters. • One or both of the individual input signals (In+ and In-) are outside the acceptable range.
Amber	**Simulate mode** The axis is in simulate mode.

See Also

RMC200 Overview | Analog Input Type | Analog Overflow Limit | Analog Underflow Limit | Analog Input Filter | A8 Wiring

7.4.9.7. Q4 Module (RMC200)

Four (4) Quadrature Encoder Inputs with Home and Registration Inputs

The Q4 module for the RMC200 provides four incremental quadrature encoder (A, B, Z) inputs with one home input and one registration input per encoder input. LED's indicate the state of each quadrature input, each home input, and each registration input.

The Q4 has the following software configuration options:

General	
Weight	372 g + 26 g (2 connectors)
Feature	**Options**

A and B Input Type	Set this to match your encoder. Delta always recommends RS-422 encoders for noise immunity and speed performance. • RS-422 differential receiver • Single-ended TTL-level input • Differential HTL (High Threshold Logic), up to 24 Vdc signals • Single-ended HTL* (High Threshold Logic), up to 24 Vdc signals
Z Input Type	Set this to match the Z (Index) pulse of your encoder. Delta always recommends RS-422 encoders for noise immunity and speed performance. • RS-422 differential receiver • Single-ended TTL-level input • Differential HTL (High Threshold Logic), up to 24 Vdc signals • Single-ended HTL* (High Threshold Logic), up to 24 Vdc signals • DI (Discrete Input), up to 24 Vdc signals
Counter Mode	Choose from standard quadrature or a pulse counter. Pulse counters cannot detect direction. • A Quad B (standard incremental quadrature encoder) • A Rising Edge (pulse counter) • A Falling Edge (pulse counter) • A Both Edges (pulse counter)
A and B Termination	Selectable On or Off
Z Termination	Selectable On or Off
H Input Type	• TTL • DI (Discrete Input), up to 24 Vdc signals
R Input Type	5 V or 12-24 V
HTL Threshold	For setting the threshold level of the A, B and Z single-ended HTL option: • 7 V (suitable for 12 V signals) • 12 V (suitable for 24 V signals)
Homing and Registration sources	The Homing and Registration functions on the Q4 can come from any of the following inputs: • For Q4 channel 0 or 1: Home0, Home1, Reg0, Reg1 • For Q4 channel 2 or 3: Home2, Home3, Reg2, Reg3
Power	
Max Power Dissipation	1.4 W

*The HTL single-ended threshold of these items can be set with the HTL Threshold parameter.

Delta recommends an RS-422 line driver output for quadrature encoders, as it provides the highest speed and very good noise immunity. The TTL and HTL input types are intended for retrofit applications where an existing encoder cannot easily be changed to RS-422.

Part Number

The part number of the Q4 module is **R200-Q4**.

Setting Up the Q4 Module

Wiring

See Q4 Wiring.

Axis Definitions

In order to use a Q4 quadrature input, it must be assigned to an RMC internal software axis, via the Axis Definitions Dialog.

Feedback Parameters

After a Q4 input has been assigned to an axis, the following axis parameters may need to be configured in Axis Tools:

- Counter Mode
- AB Input Type
- AB Termination
- Z Input Type
- Z Termination
- H Input Type
- R Input Type
- HTL Threshold

Scale/Offset

After configuring the axis feedback parameters, the feedback value must be scaled to practical measurement units:

Quadrature Scaling

Homing and Registration

For Homing and registration events on the Q4, there are multiple options for the source input. The source input is specified by the command parameters in the in the Arm Home (50) and Arm Registration (52) commands. See the Homing and Registration topics for more details.

Specifications

Quadrature Inputs	
A and B Input Types, software selectable	RS-422 (5V differential receiver for A+, A-, B+, B-) HTL differential (A+, A-, B+, B-) HTL single-ended 12V (A, B) HTL single-ended 24V (A, B) TTL single-ended (A, B)
Z Input Types, software selectable	RS-422 (Z+, Z-) HTL differential (Z+, Z-) HTL single-ended 12V (Z) HTL single-ended 24V (Z) TTL single-ended (Z) DI (discrete input) (Z)
Termination	Software selectable in RS-422 and TTL modes for A and B and for Z. Input impedance: 115 Ω or >100 kΩ
Absolute Max Input Voltage	26.2 V
Absolute Min Input Voltage	-26.2 V
Fault Voltage	The associated Fault status bit will turn on in the following cases: TTL, RS-422: Input voltage < -16V or > 16V (typical)

	HTL, DI: Input voltage < -16V (typical)
Home Inputs	
Input Types, software selectable	TTL single-ended DI (discrete input)
RS-422 Input	
Max Count Rate	12,000,000 counts per second
Min Edge Alignment	45 ns time between A edge and B edge
Min Differential Input Voltage	+/-460 mV max/min
Input Hysteresis	230 mV typical
HTL Differential Input	
Max Count Rate	2,000,000 counts per second
Min Edge Alignment	60 ns time between A edge and B edge
Min Differential Input Voltage	+/-2 V max/min
Input Hysteresis	1 V typical
HTL Single-ended 12V Input	
Max Count Rate	1,000,000 counts per second
Min Edge Alignment	71 ns time between A edge and B edge
Input Threshold	6 V to 8 V
Input Hysteresis	270 mV typical
HTL Single-ended 24V Input	
Max Count Rate	1,000,000 counts per second
Min Edge Alignment	71 ns time between A edge and B edge
Input Threshold	11 V to 13 V
Input Hysteresis	270 mV typical
TTL Single-ended Input	
Max Count Rate	1,000,000 counts per second
Min Edge Alignment	95 ns time between A edge and B edge
Input Threshold	0.8 V to 2.0 V
Input Hysteresis	530 mV typical
DI Input	
Input Threshold	5.5 V to 8 V
Input Hysteresis	1.2 V typical
Max Input Current	3.3 mA
Registration Inputs	
Input Characteristics	5 or 12-24 Vdc (software selectable)
Input “High” range	5 Vdc input: 3.5 to 5.5 Vdc, 7.5 mA max 12-24 Vdc input: 9 to 26.4 Vdc, 7 mA max
Input “Low” range	5 Vdc input: 0 to 1.7 Vdc, <1 mA 12-24 Vdc input: 0 to 5 Vdc, <1 mA

Maximum propagation delay	Off to On: 5 Vdc input: 0.3 µs 12-24 Vdc input: 0.3 µs On to Off: 5 Vdc input: 0.3 µs, (1.2 µs, open collector drive, 5V) 12-24 Vdc input: 0.5 µs, (11 µs, open collector drive, 24V)
Common	
Functional Isolation	500 VAC

LEDs

Light Bar

State	Description
Off	No power to the RMC200 or the module was not inserted when the RMC200 powered up.
Blue	The module was recognized at power-up.

Input LEDs: In0 .. In3

Each encoder input has an associated input LED:

LED State	Description
Off	The input is not assigned to an axis.
Green	**Operating normally** The input is assigned to an axis and the encoder signals are in a valid range.
Red	**Error** The input is assigned to an axis and is not receiving valid encoder signals. This may be due to a wire break or short, or due to incorrectly configured axis feedback parameters. Transducer errors that cause an invalid value on the Q4 include: • Wire break is detected. See the Encoder Status register for the wire break status bits. • Input voltage fault.
Amber	**Simulate mode** The axis is in simulate mode. **Note:** The input LED may appear to be amber for certain transducer errors when the RMC is alternately seeing a valid and invalid input, and the LED is switching between red and green colors quickly enough that the LED appears to be amber.

Home LEDs Hm0 .. Hm3 and Reg LEDs Reg0 .. Reg3

State	Description
Off	**Input is off** The input is currently low (inactive).
Orange	**Input is on**

	The input is currently high (active).

See Also

RMC200 Overview | Q4 Wiring

7.4.9.8. U14 Module (RMC200)

Universal Input/Output Module with:

- **4 Analog Inputs**
  ±10V, 4-20 mA, or ±20 mA
- **2 Analog Outputs**
  ±10V, 4-20 mA, or ±20 mA
- **4 Discrete I/O**
  Individually configurable as input (12 to 24 Vdc) or output (solid state relay)
- **2 High-Speed Channels**
  SSI, MDT, or quadrature encoder inputs
- **2 High-Speed Discrete Inputs**
  For general-purpose inputs or for registration or index (Z) inputs associated with a high-speed channel in quadrature mode

Analog Input Features

- Four differential analog inputs
- Individually software selectable as ±10 V, 4-20 mA, or ±20 mA
- 18 bits resolution ADCs with oversampling for increased effective resolution and noise reduction
- 200 kHz internal sampling rate
- Broken wire detection
- Can be used for position, velocity, pressure, and force inputs
- One LED per channel indicates the state of the respective input

Analog Output Features

- Two analog outputs specifically designed for control outputs to valves, amplifiers, or drives
- 18 bit resolution
- Individually software selectable as ±10 V, 4-20 mA, or ±20 mA. Also supports custom ranges within the ±10 V and ±20 mA ranges, such as 0-10 V, 0-5 V, 1-5 V, etc.
- One LED per channel indicates the state of the respective analog output

Discrete I/O Features

- 4 discrete I/O, individually configurable as input or output, individually isolated
- Inputs: 12 to 24 VDC, polarity independent, sinking or sourcing driver
- Outputs: Solid state relay, 75 mA continuous
- One LED per I/O point indicates the status of the input or output
- D0 and D1 may be used as registration or homing inputs in conjunction with the high-speed channels in quadrature input mode, but are not high-speed inputs.

High-Speed Channels

Each of the two high-speed channels are independently configurable as SSI, magnetostrictive Start/Stop or PWM, or Quadrature. One LED per channel indicates the state of the respective high-speed channel, and an additional LED indicates the state of each Reg/Z input.

SSI Channels

- Binary or Gray Code
- Rotary encoders or linear transducers
- 8 to 32 bits of data
- Differential RS-422 SSI interface

  The SSI channels can be configured to do the following tasks:

  - **SSI Input**
    This is a standard SSI input for obtaining data from an SSI transducer or encoder.
  - **SSI Monitor**
    Each channel can be configured as to monitor the data that another SSI master is receiving from an SSI device. This makes it possible to synchronize multiple RMCs to one SSI transducer.
  - **SSI Echo**
    Channel 1 can be configured to output the data from channel 0, which must be configured as an SSI input. Channel 1 will then behave as an SSI device (like a transducer or encoder). This is useful when another SSI device needs to receive the same data that the RMC is receiving, and the RMC must be in control of sending the clock to the SSI device.

MDT (Magnetostrictive Displacement Transducer) Channels

- Start/Stop or PWM
- Requires RS-422 signals

Quadrature Channels

- Each high-speed channel is individually software-configurable as one of the following quadrature (A and B) input types:
  - RS-422 differential receiver (recommended)
  - Single-ended TTL-level input
  - Differential HTL (High Threshold Logic), up to 24 Vdc signals
  - Single-ended HTL (High Threshold Logic), 7 or 12 V threshold
- Choose from standard quadrature or a pulse counter. Pulse counters cannot detect direction.
  - A Quad B (standard incremental quadrature encoder)
  - A Rising Edge (pulse counter)
  - A Falling Edge (pulse counter)
  - A Both Edges (pulse counter)
- The **Reg/Z0** and **Reg/Z1** inputs can be used as high-speed registration in conjunction with quadrature inputs 0 and 1.
- The **Reg/Z0** and **Reg/Z1** inputs can be used as an Index (Z) or Home input in conjunction with quadrature inputs 0 and 1, respectively, for homing based on the encoder's Index (Z) signal or a home signal.
- Discrete I/O points **D0** and **D1** can be used in conjunction with quadrature inputs 0 and 1 for registration and homing, but are not high-speed inputs.
- Each **Reg/Z** input is individually software-configurable as one of the following input types to support an index input that may be external to the encoder.
  - RS-422 differential receiver (recommended)
  - Single-ended TTL-level input
  - Differential HTL (High Threshold Logic), up to 24 Vdc signals
  - Single-ended HTL (High Threshold Logic), 7 or 12 V threshold

  - DI (discrete input), for 12-24 VDC signals
- Termination is software-selectable for the A and B inputs, and separately for the Z inputs.

  Delta recommends an RS-422 line driver output for quadrature encoders, as it provides the highest speed and very good noise immunity. The TTL and HTL input types are intended for retrofit applications where an existing encoder cannot easily be changed to RS-422. For new machine designs, Delta recommends RS-422.

Registration/Quad Z Inputs

The Reg/Z inputs may be used together with the high-speed channels when configured as quadrature, as described above in the **High-Speed Channels** section. The Reg/Z inputs are also available as general-purpose discrete inputs. When the high-speed channel is in Quadrature mode, the Reg/Z will be configurable using the Quadrature axis parameters. In all other modes, the Reg/Z will be set as a single-ended HTL input with a 12V threshold.

Part Number

The part number of the U14 module is **R200-U14**.

Setting Up U14 Analog Outputs

1. Wire the analog output as described in the U14 Wiring topic.
2. Assign the output to an axis as described in the Defining Axes topic.
3. In the Axis Parameters Pane, for the axis to which the output is assigned, set the Output Type parameter to the desired range.

Setting Up U14 Analog Inputs

1. Wire the analog input as described in the U14 Wiring topic.
2. Assign the input to an axis as described in the Defining Axes topic.
3. In the Axis Parameters Pane, for the axis to which the input is assigned, set the following parameters:
   - Analog Input Type to Voltage or Current.
   - The Analog Overflow Limit and Analog Underflow Limit define the valid range of voltage or current, but typically need not be changed.
   - The Analog Input Filter also applies to the U14, but typically need not be changed.
4. Scale the axis as described in the analog scaling topics:
   - Analog Position Scaling
   - Analog Velocity Scaling
   - Analog Acceleration Scaling
   - Analog Pressure/Force Scaling

Setting Up U14 Discrete I/O

1. Use the Discrete I/O Configuration window to configure each I/O point as an input or output.
2. Wire the I/O points as described in the U14 Wiring topic.

Setting Up U14 High-Speed Channels

1. Configure each high-speed channel as described in Configuring U14 High-Speed Channels. These channels must be configured before being assigned to axis inputs.
2. Assign the input to an axis as described in the Defining Axes topic.
3. In the Axis Parameters Pane, for the axis to which the input is assigned, set the following parameters:

SSI Transducers	MDT Transducers	Quadrature Encoders
Required		

SSI/MDT Feedback Type SSI Data Bits SSI Format	SSI/MDT Feedback Type MDT Type	Linear/Rotary Counter Mode AB Input Type
Application-specific		
Linear/Rotary SSI Clock Rate Wire Break Detection SSI Overflow Mode SSI Termination SSI High Bits to Ignore SSI Low Bits to Ignore	MDT Blanking Period	AB Termination Z Input Type Z Termination HTL Threshold

4. Scale the axis as described in the following scaling topics:
   - SSI Scaling
   - MDT Scaling
   - Quadrature Scaling

Specifications

Analog Inputs

Analog Input (4 per module)	
Inputs	Four 18-bit differential (higher resolution obtained by oversampling)
Functional Isolation	500 VAC
Overvoltage protection	±24 V
Nominal Input Ranges	±10 V, 4-20 mA, ±20 mA (each input independently configurable)
Max Differential Ranges	Voltage: -10.2 V to +10.2 V Current: -20 mA to +20 mA (continuous), -25 mA to +25 mA (peak)
Max Input Voltage Range	In+ or In- relative to Cmn: -14 V to +14 V typical
Broken wire detection	When the input is not connected, internal biasing pulls the input voltage down to its full negative value.
Input impedance	Voltage input: 1 MΩ Current input: 165 Ω
Input filter slew rate	25 V/ms
Sampling frequency	200 kHz internal sampling. Provides one filtered sample per control loop (e.g. 1 msec) to CPU.
Sampling filter	250 Hz – 4 kHz, user-selectable internal low-pass sampling filter.
Offset drift with temperature	0.2 LSB°C typical (±10 V range)
Gain drift with temperature	20 ppm/°C typical (±10 V range)
Non-linearity	12 LSB (counts) typical (±10 V range)

Analog Outputs

Analog Output (2 per module)	
Range	Voltage mode: ± 10 V @ 15 mA (670 Ω or greater load) Current mode: ± 20 mA @ 10 V (500 Ω or lower load)
Tolerance at full output	Voltage mode: ± 5 mV at 10 V Current mode: ± 10 μA at 20 mA

Resolution	18 bits
Hardware Output Filter	First-order filter, time constant 50 µsec
Functional Isolation	500 VAC
Overload protection	Continuous short to common
Overvoltage protection	Outputs are protected by clamp diodes

Discrete I/O

Discrete I/O	
Discrete I/O points	4; each is individually configurable as inputs or outputs.
Inputs	
Input Characteristics	12-24 VDC, polarity independent
Functional Isolation	500 VAC
Input "High" Range	9 to 26.4 VDC (polarity independent) 3 mA maximum
Input "Low" Range	0 to 5 VDC (polarity independent) <1 mA
Logic Polarity	True High
Maximum Propagation Delay	100 µsec, (750 µsec, open collector "Off")
Outputs	
Outputs	Solid State Relay
Load Types	DC general use, DC resistance, DC Pilot Duty
Functional Isolation	500 VAC
Maximum voltage	±30 V (DC or peak AC voltage)
Maximum current	±75 mA
Maximum propagation delay	2 ms turn-on, 0.5 ms turn-off
Logic 1 (True, On)	Low impedance (15 Ω maximum)
Logic 0 (False, Off)	High impedance (<100 nA leakage current at 30 V)

SSI Interface

SSI Interface	
Data Inputs	RS-422 differential
Clock Outputs	RS-422 differential
Termination	Software selectable data input impedance: 110 Ω or >200 kΩ
Clock Frequency	User-selectable 100 kHz to 2500 kHz
Maximum Cable Length	Transducer Dependent, approx. 3-2100 ft. See the SSI Clock Rate topic for details.
Resolution	Transducer dependent
Count Encoding	Binary or Gray Code
Count Data Length	8 to 32 bits
Bit Masking	A selectable number of high or low bits may be masked
Additional Settings	Selectable overflow modes to conform to various SSI transducers Wire break detection

Functional Isolation	500 VAC

MDT Interface

MDT Interface (Start/Stop or PWM)	
Transducer interface types	MDT with Start/Stop or PWM (Pulse Width Modulated) feedback
Interrogation Outputs	RS-422 differential (transducer must be configured for external interrogation)
Return Inputs	RS-422 differential
Resolution	0.0005 in. with one recirculation
Count Rate	240 MHz
Recirculations	Supports multiple recirculations only for PWM transducers with internal recirculations.
Maximum transducer length	440 in. at 4ms (loop-time dependent)
Functional Isolation	500 VAC

Quadrature Inputs

Quadrature Inputs	
A and B Input Types, software selectable	RS-422 (5V differential receiver for A+, A-, B+, B-) HTL differential (A+, A-, B+, B-) HTL single-ended 12V (A, B) HTL single-ended 24V (A, B) TTL single-ended (A, B)
Reg/Z Input Types, software selectable	RS-422 (Reg/Z+, Reg/Z-) HTL differential (Reg/Z+, Reg/Z-) HTL single-ended 12V (Reg/Z) HTL single-ended 24V (Reg/Z) TTL single-ended (Reg/Z) DI (discrete input) (Reg/Z)
Termination	Software selectable in RS-422 and TTL modes for A and B and for Reg/Z. Input impedance: 115 Ω or >200 kΩ
Absolute Max Input Voltage	26.2 V
Absolute Min Input Voltage	-26.2 V
Fault Voltage	The associated Fault status bit will turn on in the following cases: TTL, RS-422: Input voltage < -16V or > 16V (typical) HTL, DI: Input voltage < -16V (typical)
Maximum Propagation Delay (A, B, Reg/Z inputs)	RS-422: 25 ns All others: 100 ns
RS-422 Input	
Max Count Rate	8,000,000 counts per second
Min Edge Alignment	55 ns time between A edge and B edge

Min Differential Input Voltage	±460 mV max/min
Input Hysteresis	230 mV typical
HTL Differential Input	
Max Count Rate	2,000,000 counts per second
Min Edge Alignment	70 ns time between A edge and B edge
Min Differential Input Voltage	±2 V max/min
Input Hysteresis	1 V typical
HTL Single-ended 12V Input	
Max Count Rate	1,000,000 counts per second
Min Edge Alignment	80 ns time between A edge and B edge
Input Threshold	6 V to 8 V
Input Hysteresis	270 mV typical
Max Input Current	460 µA
Max Propagation Delay	300 ns
HTL Single-ended 24V Input	
Max Count Rate	1,000,000 counts per second
Min Edge Alignment	80 ns time between A edge and B edge
Input Threshold	11 V to 13 V
Input Hysteresis	270 mV typical
Max Input Current	460 µA
Max Propagation Delay	300 ns
TTL Single-ended Input	
Max Count Rate	1,000,000 counts per second
Min Edge Alignment	105 ns time between A edge and B edge
Input Threshold	0.8 V to 2.0 V
Input Hysteresis	530 mV typical
DI Input	
Input Threshold	5.5 V to 8 V
Input Hysteresis	1.2 V typical
Max Input Current	3.3 mA
Max Propagation Delay	300 ns
Common	
Functional Isolation	500 VAC

General

Weight	
Weight	401 g + 26 g (2 connectors)
Power	
Max Power Dissipation	2.6 W, both analog outputs in voltage mode 3.2 W, both analog outputs in current output

LEDs

Light Bar

State	Description
Off	No power to the RMC200 or the module was not inserted when the RMC200 powered up.
Blue	The module was recognized at power-up.

Analog Input LEDs: A0, A1, A2, A3

Each analog input has an associated LED:

State	Description
Off	The input is not assigned to an axis.
Green	**Operating normally** The input is assigned to an axis and is receiving a valid value from the transducer.
Red	**Error** The input is assigned to an axis and is not receiving a valid value from the transducer. This may be due to a transducer or wiring error, or due to incorrectly configured axis feedback parameters. Input errors that cause an invalid value include: • The differential analog input signal is outside the user-specified Overflow/Underflow Limit parameters. • One or both of the individual input signals (In+ and In-) are outside the acceptable range.
Amber	**Simulate mode** The axis is in simulate mode.

Analog Output LEDs: Out0, Out1

Each analog output has an associated LED:

State	Description
Off	**Not assigned** The output is not assigned to an axis.
Green	**Operating normally** The axis is operating normally with no latched errors.
Red	**Error** The axis has at least one error bit set that has halted the axis.
Amber	**Simulate mode** The axis is in simulate mode.

High-Speed Channel LEDs: S/Q0, S/Q1

Each channel has an associated LED:

LED State	Description
Off	The input is not assigned to an axis.
Green	**Operating normally**

	The input is assigned to an axis and is receiving a valid value from the transducer.
Red	**Error** The input is assigned to an axis and is not receiving a valid value from the transducer. This may be due to a transducer error, or due to incorrectly configured axis feedback parameters. Transducer errors that cause an invalid value on a high-speed channel include: • SSI: A Wire Break condition exists and Wire Break Detection is enabled • SSI: SSI Overflow Pattern was detected • MDT Start/Stop or PWM: There was no start pulse detected or rising edge of the PWM response • MDT Start/Stop or PWM: There was no stop pulse or falling edge of the PWM response in the required time • Quadrature: A wire break condition exists • Quadrature: An input voltage exceeded the allowable range
Amber	**Simulate mode** The axis is in simulate mode. **Note:** The input LED may appear to be amber for certain transducer errors when the RMC is alternately seeing a valid and invalid input, and the LED is switching between red and green colors quickly enough that the LED appears to be amber.

Registration/Index Inputs LEDs: R/Z0, R/Z1

Each registration/index (Reg/Z) input has an associated LED.

State	Description
Off	**Input is off** The input is currently low (inactive).
Orange	**Input is on** The input is currently high (active).

Discrete I/O LEDs: D0, D1, D2, D3

Each discrete I/O point has an associated LED.

State	Description
Off	**I/O point is off** If this I/O point is configured as an input, then the input is currently low (inactive). If this I/O point is configured as an output, then the output is currently open (inactive).
Orange	**Input is on** This I/O point is configured as an input, and the input is currently high (active).
Yellow	**Output is on** This I/O point is configured as an output, and the output is currently closed (active).

Note:
Forcing an input *will not* affect the state of the LED. Only a physical current that turns on the input will turn the LED on.
Forcing an output *will* turn the LED on, because the output will physically be on (conducting).

See Also

RMC200 Overview | U14 Wiring

7.4.9.9. Configuring U14 High-Speed Channels

Each of the two high-speed channels on the U14 module are independently configurable as SSI, MDT, or Quadrature. These channels must be configured before being used for such tasks as assigning to axis inputs. To configure the channels, open the U14 properties:

1. In the Project Pane, expand the **Modules** folder.
2. Double-click the desired U14 module, then choose the **Configuration** page.
3. Each channel can be configured to operate in one of the modes listed in the table below. See the sections below for instructions on configuring the channels for specific uses.
    - **SSI/MDT Input**
      Allows the channel to be used as an SSI or MDT input to be assigned to a control or reference axis.
    - **Quadrature Input**
      Allows the channel to be used as a A-quad-B quadrature encoder or pulse input to be assigned to a control or reference axis. One I/O point per channel (Reg/Z0 for channel 0 or Reg/Z1 for channel 1) can be used as a reference or home input for high-speed count latching.
    - **SSI Monitor Input**
      Allows the channel to monitor the SSI data transfer between two other devices. The channel's clock and data are both set as inputs. This channel can then be assigned to a control or reference axis.
    - **SSI Echo Output (Channel 1 only)**
      Automatically outputs the data from channel 0 onto channel 1. Channel 0 must first be configured as an SSI input. Channel 1 behaves as an SSI device, with a clock input and a data output. The data format and number of data bits will be the same as channel 0. The external clock may be in the range 25 kHz to 8 MHz.

Configure Channel as a Quadrature Axis Input

For each channel that will be used as a quadrature input, do the following:

1. Choose **Quadrature Input**, then click **OK**.
2. Wire the quadrature input as described in the U14 Wiring topic.
3. Assign the input to an axis as described in the Defining Axes topic.
4. In the Axis Parameters Pane, set the following axis parameters:

Required
Linear/Rotary
Counter Mode
AB Input Type
Application-specific
AB Termination
Z Input Type

Z Termination
HTL Threshold

If the input is in the middle of a daisy-chained signal, consider especially setting the Quadrature AB Termination and Quadrature Z Termination to disable termination. See U14 Wiring for details.

5. Scale the axis feedback as described in the Quadrature Scaling topic.

Configure Channel as an SSI Axis Input

For each channel that will be used as an SSI input, do the following:

1. Choose **SSI/MDT Axis Input**, then click **OK**.
2. Wire the SSI input as described in the U14 Wiring topic.
3. Assign the input to an axis as described in the Defining Axes topic.
4. In the Axis Parameters Pane, set the following axis parameters:

Required
SSI/MDT Feedback Type
SSI Data Bits
SSI Format
Application-specific
Linear/Rotary
SSI Clock Rate
Wire Break Detection
SSI Overflow Mode
SSI Termination
SSI High Bits to Ignore
SSI Low Bits to Ignore

5. Scale the axis feedback as described in the SSI Scaling topic.

Configure Channel as an MDT Axis Input

For each channel that will be used as an MDT input, do the following:

1. Choose **SSI/MDT Axis Input**, then click **OK**.
2. Wire the channel as described in the U14 Wiring topic.
3. Assign the input to an axis as described in the Defining Axes topic.
4. In the Axis Parameters Pane, set the following axis parameters:
    - SSI/MDT Feedback Type
    - MDT Type
    - MDT Blanking Period (typically only necessary for very old sensors)
5. Scale the axis feedback as described in the MDT Scaling topic.

Configure Channel as an SSI Monitor Input

For each channel that will be used as an SSI Monitor, do the following:

1. Choose **SSI Monitor Input**, then click **OK**.
2. Wire the SSI input as described in the U14 Wiring topic.
3. Assign the input to an axis as described in the Defining Axes topic.
4. In the Axis Parameters Pane, set the following axis parameters:

Required
SSI Data Bits

SSI Format
Application-specific
Linear/Rotary
SSI Clock Rate
Wire Break Detection
SSI Overflow Mode
SSI Termination
SSI High Bits to Ignore
SSI Low Bits to Ignore

5. Scale the axis feedback as described in the SSI Scaling topic.

Daisy-Chaining an SSI Device to Multiple Controllers

Daisy-chaining refers to wiring an SSI device to multiple controllers. When wiring a daisy-chained SSI system, the SSI input should be on one end of the daisy chain with the SSI device on the other end, and any SSI monitor inputs in the middle of the daisy chain. The wiring must be done in a sequential fashion, that is, the wiring goes from the SSI device to the first controller, then from that controller to the next controller, etc. Apply termination only to the SSI master. See U14 Wiring for more details.

1. Configure a channel on the master U14 as an **SSI Axis Input** as described in the **Configure Channel as an SSI Axis Input** section above. Termination is applied by default (see the SSI Termination parameter).
2. For the remaining monitoring U14 modules, configure a channel as an **SSI Monitor Input** as described in the **Configure Channel as an SSI Monitor Input** section above. Verify that the SSI Termination parameter is set to disabled for each monitor input.

Configure SSI Echo Mode

SSI Echo Mode retransmits on channel 1 the SSI data that is received on channel 0. This is useful for situations where the RMC must be the SSI master so that it is guaranteed to get the SSI data every loop time, and some other device wants to query the RMC for the same data, but possibly at a different rate or not as consistently.

Channel 1 behaves as an SSI device, with a clock input and a data output. The data format and number of data bits will be the same as channel 0. The external clock may be in the range 25 kHz to 8 MHz. Channel 0 must first be configured as an SSI input.

To configure SSI Echo Mode:

1. Configure channel 0 as an SSI axis input as described above in the **Configure Channel as an SSI Axis Input** section.
2. For channel 1, choose **SSI Echo output**, then click **OK**.
3. Channel 1 will automatically output the same data that channel 0 receives. The data format and number of data bits will be the same as channel 0. The external clock may be in the range 25 kHz to 8 MHz.

See Also

U14 Module (RMC200) | U14 Wiring

7.4.9.10. D24 Module (RMC200)

24-Point Discrete I/O Module including 4 high-speed inputs

The D24 discrete I/O module for the RMC200 contains 20 discrete I/O points that are designed for 24 VDC signals and are individually user-configurable as inputs or outputs, and 4 high-speed discrete inputs for 5 or 12-24 VDC signals for quadrature encoder, pulse counter, or general-purpose use.

General-Purpose Discrete I/O

Discrete I/O points (0-19) are general-purpose inputs:

I/O Points	Features
0-19	• Individually selectable as an input or output • As inputs, they work with 12-24 VDC signals • As outputs, they are solid state relays rated at 24 V • Arranged in 3 isolated groups of 8, 8, and 4 I/O points

High-Speed Discrete Input Features

The quadrature functionality of the D24 high-speed inputs is designed for single-ended and differential signals and voltages levels of 5 or 12-24 V, supports up to two quadrature encoder inputs depending on the configuration, and is not necessarily intended for RS-422 signals. The pulse counter functionality of the D24 high-speed inputs is designed for single-ended pulse train signals of 5 or 12-24 V. The D24 supports up to two pulse counters. These pulse counters can be used for feedback in a control axis.

Note:
For quadrature encoder signals, Delta always recommends the RS-422 line driver, as it provides the highest speed and noise immunity. RS-422 quadrature signals are supported by the Q4, U14, and S8 modules. The D24 quadrature input is intended for existing non-RS-422 encoders that cannot be easily changed.

The D24 high-speed inputs 20-23 support the following:

Feature	Options
Quadrature Option	Selectable in the D24 properties dialog: • None • 1 quadrature input (A, B, Z), uses inputs 20-22. • 1 quadrature input with complements (A, A, B, B) (no Z) and wire break detection, uses inputs 20-23. • 2 quadrature inputs with A and B, uses inputs 20-23.
A, B, and Z Input Type	Hardware selectable: • 5V or 24V signal levels (each input has a separate +5V pin and +24V pin) • Differential (defined by wiring method) • Single-ended (defined by wiring method)
Counter Mode	Choose from standard quadrature or a pulse counter. Pulse counters cannot detect direction. • A/B Quad (standard quadrature encoder) • A Rising Edge (pulse counter) • A Falling Edge (pulse counter) • A Both Edges (pulse counter)
H Input Threshold	Hardware selectable:

<table>
<tr><td></td><td>• 5V or 24V signal levels (each input has a separate +5V pin and +24V pin)</td></tr>
<tr><td>R Input Threshold</td><td>Hardware selectable:
• 5V or 24V signal levels (each input has a separate +5V pin and +24V pin)</td></tr>
<tr><td>Homing, and Registration sources</td><td>The inputs that can be used as home inputs or registration inputs, based on the selected D24 quadrature option (as selected in the D24 Properties) are as follows.
<table>
<tr><th>Quadrature Option</th><th>Possible Home and Registration Inputs</th></tr>
<tr><td>One quadrature input (A, B, Z)</td><td>D22, D23, D18, D19</td></tr>
<tr><td>One quadrature input (A, A, B, B) with wire break detection</td><td>D18, D19</td></tr>
<tr><td>Two quadrature inputs, using inputs D20&D21</td><td>D18, D19</td></tr>
<tr><td>Two quadrature inputs, using inputs D22&D23</td><td>D16, D17, D18, D19</td></tr>
</table></td></tr>
<tr><td>Event Timers</td><td>The high-speed inputs can be used for Event Timing. See Event Timers for details.</td></tr>
</table>

General-Purpose Inputs

The high-speed inputs can be used a general-purpose inputs and appear in the Discrete I/O Monitor.

Quadrature Encoder Functionality

The high-speed inputs are software-configurable as one of the following options:

- 1 quadrature input (A, B, Z), uses inputs 20-22.
- 1 quadrature input with complements (A, A, B, B) and wire break detection, uses inputs 20-23.
- 2 quadrature inputs with A and B, uses inputs 20-23.

The quadrature inputs support the following A and B signal types:

- Differential from 5V to 24V
- Single-ended from 5V to 24V

Note:
The D24 does not necessarily support RS-422. The minimum RS-422 differential is 0.200 V, whereas the D24 high-speed inputs require 3.5V to turn on. However, some RS-422 encoders do provide enough differential voltage to be compatible with the D24 quadrature input. Alternatively, an RS-422 signal can be wired to the D24 inputs via only one side of the RS-422 signal and a separately applied 5V supply. See the D24 Wiring topic for details.

Consider using the Q4, S8, or U14 modules for RS-422 quadrature encoders instead.

Pulse-counter Functionality

The high-speed inputs are software-configurable as up to two pulse counter inputs (input 20 and/or 22), with the following options:

- Rising edge counter
- Falling edge counter
- Rising and falling edge counter

This functionality requires configuring one or two quadrature encoder inputs.

Part Number

The part number of the D24 module is **R200-D24**.

Setting Up the D24 Module

Wiring

See D24 Wiring.

Configuring General-Purpose Inputs and Outputs

Use the Discrete I/O Configuration to individually configure I/O points 0-19 as inputs or output, assign tag names, and define behavior when the RMC enters Program mode or Fault mode.

The high-speed inputs 20-23 also appear in the Discrete I/O Configuration and can be assigned a tag name, and will also appear in the Discrete I/O Monitor.

Configuring High-Speed Inputs

To configure quadrature encoder inputs:

1. In the Project Pane, expand the **Modules** folder.
2. Right-click the D24 module and choose **Properties**.
3. On the **Configuration** page, choose the desired quadrature configuration.
4. Click **OK**. RMCTools may prompt you that the controller will need to restart to apply the changes.
5. Define an axis using the D24 quadrature input.
6. The following axis parameters may need to be configured in Axis Tools:
    - Counter Mode
7. Scale the axis as described in the Quadrature Scaling topic.

Specifications

General	
Weight	397 g + 26 g (2 connectors)
General Purpose Discrete I/O	
Discrete I/O points	20; each is individually configurable as inputs or outputs.
General Purpose Inputs	
Input Characteristics	12-24 VDC, polarity independent
Functional Isolation	500 VAC
Input "High" Range	9 to 26.4 VDC (polarity independent) 3 mA maximum
Input "Low" Range	0 to 5 VDC (polarity independent) <1 mA
Logic Polarity	True High
Maximum Propagation Delay	100 µs, (750 µs, open collector "Off")
General Purpose Outputs	
Outputs	Solid State Relay
Functional Isolation	500 VAC
Maximum voltage	±30 V (DC or peak AC voltage)
Maximum current	±75 mA
Maximum propagation delay	2 ms turn-on, 0.5 ms turn-off

Logic 1 (True, On)	Low impedance (15 Ω maximum)
Logic 0 (False, Off)	High impedance (<100 nA leakage current at 30 V)
High-Speed Inputs	
Input Characteristics	5 or 12-24 VDC (separate pins for 5 or 24 VDC)
Functional Isolation	500 VAC
Input "High" Range	5 VDC input: 3.5 to 5.5 VDC, 7.5 mA maximum 12-24 VDC input: 9 to 26.4 VDC, 7 mA maximum
Input "Low" Range	5 VDC input: 0 to 1.7 VDC, <1 mA maximum 12-24 VDC input: 0 to 5 VDC, <1 mA maximum
Logic Polarity	True High
Maximum Propagation Delay for general-purpose usage	100 µs
Maximum Propagation Delay for quadrature encoder and event-timing usage	Off to On: 5 Vdc input: 0.8 µs 12-24 Vdc input: 0.8 µs On to Off: 5 Vdc input: 0.8 µs, (1.7 µs, open collector drive, 5V) 12-24 Vdc input: 1.0 µs, (12 µs, open collector drive, 24V)
Maximum Input Frequency	5 Vdc input: 1,000 kHz, (400 kHz, open collector drive, 5V) 12-24 Vdc input: 500 kHz, (25 kHz, open collector drive, 24V)
Power	
Max Power Dissipation	1.0 W, plus 50 mW per I/O point used

LEDs

Light Bar

State	Description
Off	No power to the RMC200 or the module was not inserted when the RMC200 powered up.
Blue	The module was recognized at power-up.

I/O LEDs

The D24 module has 1 LED per I/O point. The LEDs reflect the state of the input.

State	Description
Off	**I/O point is off** If this I/O point is configured as an input, then the input is currently low (inactive). If this I/O point is configured as an output, then the output is currently open (inactive).
Orange	**Input is on** This I/O point is configured as an input, and the input is currently high (active).
Yellow	**Output is on** This I/O point is configured as an output, and the output is currently closed (active).

Note:
Forcing an input *will not* affect the state of the LED. Only a physical current that turns on the input will turn the LED on.
Forcing an output *will* turn the LED on, because the output will physically be on (conducting).

See Also

RMC200 Overview | D24 Wiring

7.5. General

Show All

7.5.1. Control Output

The Control Output is the analog control signal voltage supplied by the RMC to drive the amplifier, valve, or drive for the system to be controlled.

Depending on the hardware, the Control Output may be voltage with a range of -10 V to +10 V, or current with a range of -20 mA to +20 mA. Voltage output may be converted to current (up to ±200 mA) via Delta's VC2124 voltage-to-current converter.

The modules listed below have a physical Control Output. The Control Output is labeled **Drive** or **Drv** on the RMC150 modules.

Module	±10 V	4-20 mA	±20 mA
RMC75			
AA1, AA2	✓		
MA1, MA2	✓		
QA1, QA2	✓		
RMC150			
Analog (H)	✓		
Analog (G)	✓		
MDT (M)	✓		
Quad (Q)	✓		
Resolver (R)	✓		
SSI (S)	✓		
RMC200			
CA4	✓	✓	✓
CV8	✓		
U14	✓	✓	✓

Configuring the Control Output

RMC75/150

The user can configure the polarity of this output with the Invert Output Polarity parameter. To invert the polarity of the Control Output, set this bit.

RMC200

The user has great flexibility in configuring the range of the Control Output. See the Output Type axis parameter topic for details.

Specifications

For specifications, refer to the topic of the specific module containing the Control Output.

See Also

RMC75 AA Wiring | RMC75 A Wiring | RMC75 QA Wiring | RMC150 Control Output (Drive) Wiring | RMC200 CA4 Wiring | RMC200 CV8 Wiring | RMC200 U14 Wiring

7.5.2. Enable Output

The Enable Output is a purpose-specific output that is intended to be connected to a drive or amplifier. Using the Enable Output is optional and does not affect the Control Output. With the Enable Output, a drive or amplifier can be turned on or off.

Behavior

RMC75/150

The Enable Output can be turned on and off with the Set Enable Output (67) command. The Enable Output will turn off when a Direct Output Halt occurs, unless the axis' Direct Output status bit is already on.

RMC200

The Enable Output can be turned on and off with the Set Enable Output (67) command. When an axis becomes enabled, the Enable Output will turn on. The Enable Output will turn off when a Direct Output Halt or Disable Axis Halt occurs.

Supporting Modules

RMC75/150

On the RMC75 and RMC150, only dedicated Enable Outputs can be used as Enable Outputs. The modules listed below have one Enable Output per Control Output.

RMC75: MA, AA, QA

RMC150: Quad (Q)

RMC200

Any discrete output on any RMC200 module may be used as an Enable Output, as defined by the Enable Output parameter.

The CA4 module has one Enable Output per Control Output intended for use as an Enable Output, but may be used a general-purpose discrete output as well.

The CV8 module has eight discrete I/O points that may be used general-purpose discrete outputs, or as Enable Outputs.

The U14 module has four discrete I/O points that may be used general-purpose discrete outputs, or as Enable Outputs.

The CA4, CV8, and U14 discrete outputs are not assigned to function as an Enable Output by default. To assign it to function as an Enable Output, see **Configuring an Enable Output** below.

Configuring an Enable Output

RMC75/150

1. In the Axis Parameters, on the **All** tab, expand the **Output** section.
2. Set the Enable Output Behavior parameter to define whether the solid state output relay should be open or closed when the Enable Output is active.

RMC200

1. In the Axis Parameters, on the **All** tab, expand the **Output** section.
2. For the desired axis, click the Enable Output parameter.
3. From the drop-down list, choose a discrete output.
    a. If you wish to use the dedicated Enable Output on the CA4, choose **Dedicated**.
    b. The other discrete outputs on the RMC are listed by the IEC address of **%QX*s*.*n***, where ***s*** is the slot number and ***n*** is the number of the IO point on the module. The slot numbering ***s*** begins with 0 for the power supply, 1 for the CPU, 2 for the first IO module slot, etc.
4. Set the Enable Output Behavior parameter to define whether the solid state output relay should be open or closed when the Enable Output is active.

Additional Uses

On the RMC75 and RMC150, the physical Enable Output can be used as a general-purpose input by setting all the axis' Autostops to something other than Direct Output Halt, which will prevent the Enable Output from turning off when an axis halts. It can be turned on and off with the Set Enable Output (67) command. However, the output will not appear in the Discrete I/O Editor or Discrete I/O Monitor, and cannot be assigned a user tag name.

On the RMC200, if the Enable Output axis parameter is set to none, the physical Enable Output can be used a general-purpose input. The input will be listed in the Discrete I/O Editor and can be assigned a user tag name.

Specifications

For specifications, refer to the topic of the specific module containing the Control Output.

See Also

Enable Output Source | Enable Output Configuration | Fault Input

7.5.3. Fault Input

The Fault Input is a purpose-specific input and is intended to be connected to a drive amplifier or other source for a safety interlock. Using the Fault Input is optional.

When the Fault input is active, the Fault Status bit will turn on and the Fault Error bit will latch on. This will cause a halt if the Auto Stops are configured to do so, and the axis is not in the Direct Output state.

Supporting Modules

RMC75/150

On the RMC75 and RMC150, only dedicated Fault Inputs can be used as Fault Inputs. The modules listed below have one Fault Input per Control Output.

RMC75: MA, AA, QA

RMC150: Quad (Q)

RMC200

Any discrete input on any RMC200 module may be used as a Fault Input.

The CA4 module has one Fault Input per Control Output. That Fault Input may be used as a Fault Input, or as a general-purpose discrete input.

The CV8 module has eight discrete I/O points that may be used general-purpose discrete inputs, or as Fault Inputs.

The U14 module has four discrete I/O points that may be used general-purpose discrete inputs, or as Fault Inputs.

The CA4, CV8, and U14 discrete inputs are not assigned to function as Fault Inputs by default. To assign an input to function as a Fault Input, see **Configuring a Fault Input** below.

Configuring a Fault Input

RMC75/150

1. In the Axis Parameters, on the **All** tab, expand the **Output** section.
2. Set the Fault Input Polarity parameter to define whether the input should be active when the voltage applied across the Fault Input is high or low.

RMC200

1. In the Axis Parameters, on the **All** tab, expand the **Output** section.
2. For the desired axis, click the Fault Input Source parameter.
3. From the drop-down list, choose a discrete input.
    a. If you wish to use the dedicated Fault Input on the CA4, choose **Dedicated**.
    b. The other discrete inputs on the RMC are listed by the IEC address of **%IX*s.n***, where ***s*** is the slot number and ***n*** is the number of the IO point on the module.
    The slot numbering ***s*** begins with 0 for the power supply, 1 for the CPU, 2 for the first IO module slot, etc.
4. Fault Input Polarity parameter to define whether the input should be active when the voltage applied across the Fault Input is high or low.

Additional Uses

On the RMC75 and RMC150, the physical Fault Input can be used for SSI homing (RMC75 Only), Physical Limit Inputs, and can be used as a general-purpose input by setting the Fault Input Autostop for the associated axis to Status Only. However, the input will not appear in the Discrete I/O Editor or Discrete I/O Monitor, and cannot be assigned a user tag name. It's state can be accessed via the Fault Input status bit.

On the RMC200, if the Fault Input axis parameter is set to none, the physical Fault Input can be used a general-purpose input. The input will be listed in the Discrete I/O Editor and can be assigned a user tag name.

Specifications

For specifications, refer to the topic of the specific module containing the Control Output.

See Also

Fault Input Source| Fault Input Polarity | Enable Output

7.6. Accessories

7.6.1. Quadrature Cable

Part Number: RMC-CB-QUAD-01-xx

A cable can be purchased that connects directly to the RMC150 Quad module or the RMC75E QA1 and QA2 modules' DB-25 connectors. The cable can be purchased in one of three lengths. It separates the wires into three groups: drive, encoder, and limits. Each group has its own braided shield and insulation.

Quadrature Input / Analog Output Cables	
Part Number	RMC-CB-QUAD-01-06 RMC-CB-QUAD-01-10 RMC-CB-QUAD-01-015
Length	6 feet (2 m), 10 ft (3 m), or 15 ft (4.5 m)
Connector	DB-25 male with shielded housing (mates to connector on RMC QUAD)
User Connections	Flying leads (pigtails)
Cable	Each DB-25 connectors has three cables coming from it. Each has 24-gauge twisted pairs with an overall braided shield: • **Drive** (3 pair): Amplifier connections • **Encoder** (4 pair): Quadrature encoder connections • **Limits** (3 pair): Home and limit switches

Wire Colors

Notice that the cable connects to the same pins for the QA1, QA2, and QUAD modules, because these modules have identical pinouts.

RMC-CB-QUAD-01-xx	Connect to QA1 Pin	Connect to QA2 Pin	Function	Pin # QA1,QA2,QUAD
Enc: white/blue	A-	A-	A- from encoder	1
Enc: blue/white	A+	A+	A+ from encoder	2
Enc: white/orange	B-	B-	B- from encoder	3
Enc: orange/white	B+	B+	B+ from encoder	4
	n/c	n/c	No connection	5
Lim: white/orange	Reg Y/NegLim-	RY/NL-	Registration Y or Negative Limit	6
Lim: orange/white	Reg Y/NegLim+	RY/NL+		7
Lim: white/blue	Reg X/PosLim-	RX/PL-	Registration X or Positive Limit	8
Lim: blue/white	Reg X/PosLim+	RX/PL+		9
	n/c	n/c	No connection	10
	n/c	n/c	No connection	11
Drv: blue/white	Control Out	CtrlOut	Control Output	12
Drv: white/blue	Cmn	Cmn	Common	13
Enc: white/green	Z-	Z-	Index pulse from encoder	14
Enc: green/white	Z+	Z+		15
Enc: white/brown Enc: brown/white	Cmn	Cmn	Common	16

	n/c	n/c	No connection	17
Lim: white/green	Home-	Home-	Home Input	18
Lim: green/white	Home+	Home+		19
Drv: white/green	FltIn-	FltIn-	Fault Input	20
Drv: green/white	FltIn+	FltIn+		21
	n/c	n/c	No connection	22
	n/c	n/c	No connection	23
Drv: white/orange	EnOut-	EnOut-	Enable Output	24
Drv: orange/white	EnOut+	Enout+		25

See Also

Quadrature (Q) Module (RMC150) | QA Module (RMC75)

7.6.2. SD Card R2-MEM-SD-1G (for RMC200)

The R2-MEM-SD-1G SD card is an industrial rated SD card for use with the RMC200. Industrial grade cards offer a wide temperature range and long data retention life.

The SD card supports the following functionality on the RMC200:

- Save the controller image
- Restore the controller image
- Store general files (e.g machine design documents, data sheets, etc.)

For more details, see Using the SD Card.

Part Number

Ordering Number: R2-MEM-SD-1G

Specifications

R2-MEM-SD-1G	
Grade	Industrial
Card family	SDSC
File system	FAT16
Form factor	Standard
Capacity	1 GB
Memory type	SLC
Temperature range	-40 to +85°C

See Also

Using the SD Card

7.6.3. VC2124 and VC2100 Voltage-to-Current Converters

This document will refer to the VC2124, but the same information applies to the VC2100.

To convert the RMC's ±10 V Control Output to current (such as for servo valves), use Delta's VC2124 or VC2100 voltage-to-current converter. Information on the VC2124 and VC2100 is available on Delta's website, https://deltamotion.com. This topic explains how to set up the RMC parameters to work with the VC2124 and VC2100:

Scaling ±10 V to Current Output (± mA)

To scale the ±10 V Control Output to a current range centered around 0, such as ±85mA, use the following procedure:

1. Set the VC2124 current range to the closest value equal to or greater than the current range required by the valve.
2. Calculate the Output Limit parameter using the following equation:

   OutputLimit [V] = (Desired Limit[mA] * 10) / VC2124 current setting

Example:

Consider a servo valve that requires ±85 mA:

1. Set the VC2124 to **±90 mA**.
2. Calculate the Output Limit:

   OutputLimit [V] = (85[mA] * 10) / 90[mA]

   Output Limit = 9.444 [V]

Scaling ±10 V to 4-20 mA (RMC75 and RMC150)

If you are using an RMC module that only supports a ±10 V output together with a valve or drive that requires a 4-20mA input, follow these steps to set up the RMC parameters to work together with the VC2124 to produce a 4-20 mA output.

1. Set the VC2124 current range to **±20 mA**.
2. Set the Output Scale to: **4** V/100%
3. Set the Output Bias to: **6** V**.**
4. Set the Output Limit to: **4** V**.** (prevents the output from going below 4mA):

Inverting the Output

If a positive voltage makes the system move the wrong direction, you will need to invert the output. To do so, do steps 1-4 above, but set the Invert Output Polarity bit and set the Output Bias to -**6** V**.**

The mathematical reasoning is as follows:

After setting the VC2124 to ±20 mA, the next step is to scale the ±100% PFID output to 2-10V (the VC2124 will then convert the 2-10 V to 4-20 mA). Use the equation y=mx+b, where

y = 2-10 V range

x = ±100% range

m = Output Scale [V/100%]

b = Output Bias [V]

To calculate the Output Scale and Output Bias:

Output Scale = m = ($y1$ - $y0$)/($x1$-$x0$) = (10-2)/100 - (-100)) = 8[V]/200[%]

Output Scale = 4 [V/100%]

Output Bias = y1 - m * $x1$ = 10[V] - 4[V/100%] * 100[%]

Output Bias = 6 [V]

For different output ranges, use the same method to calculate the parameters.

Scaling ±10 V to 4-20 mA (RMC200)

If you are using an RMC module that only supports a ±10 V output together with a valve or drive that requires a 4-20mA input, follow these steps to set up the RMC parameters to work together with the VC2124 to produce a 4-20 mA output.

1. Set the VC2124 current range to **±20 mA**.
2. Set the Voltage at 100% to: **10** V
3. Set the Voltage at 0% to: **6** V
4. Set the Voltage at -100% to: **2** V

Inverting the Output

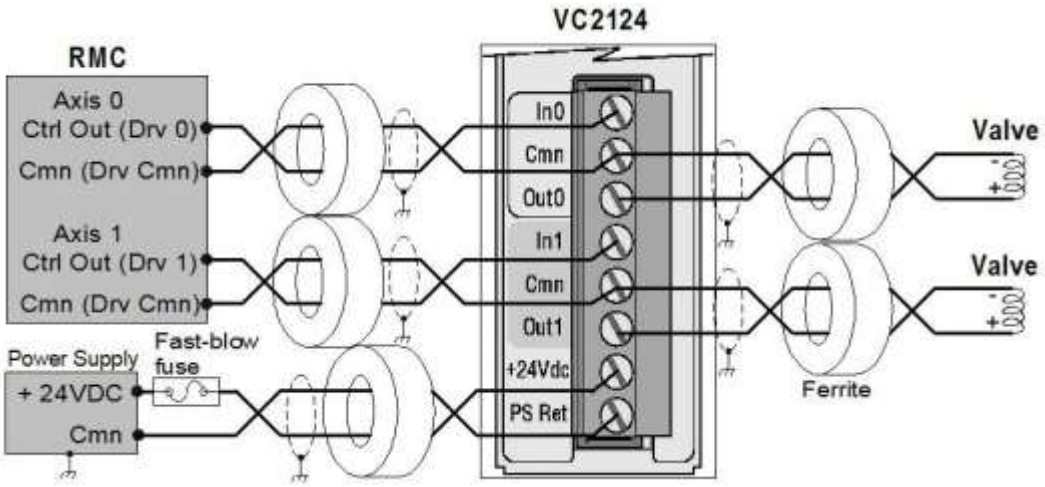
If a positive voltage makes the system move the wrong direction, you will need to invert the output. To do so, swap the values of Voltage at 100% and Voltage at -100%.

Wiring

VC2124

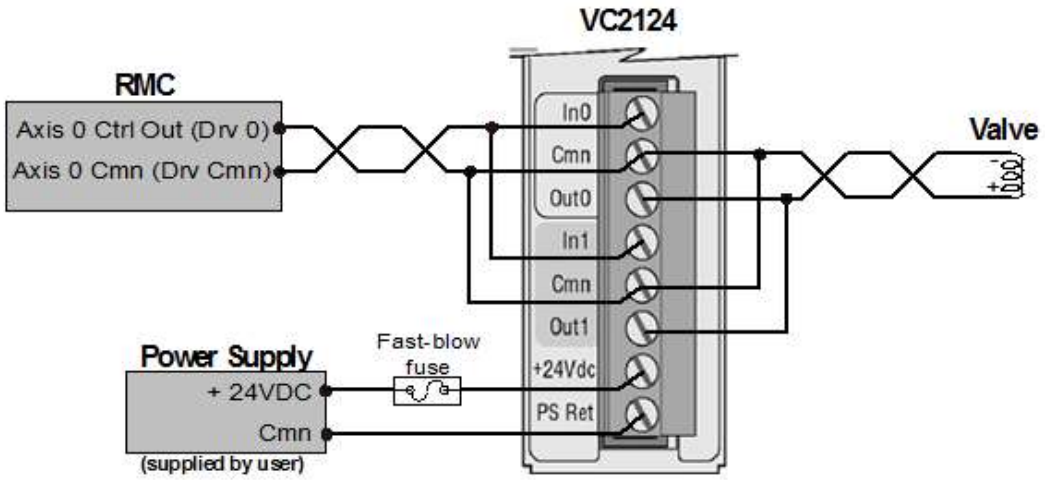
Fuse 24 VDC input with 5A maximum, UL-listed, fast-blow fuse. One fuse suffices for up to 10 VC2124s. For maximum protection, use one 500mA fuse per VC2124.

For noise immunity, use twisted, shielded pairs for all connections (twisted pair with overall shield is acceptable). For best noise immunity and CE compliance, keep wires from the RMC to the VC2124 as short as possible and less than 98 ft (30 m), and place ferrites on all cables as close to the VC2124 as possible. Sample ferrite part numbers from Steward: 28A2029-0A0 or 0A2, 28A5131-0A2, 28A0593-0A2, 28A0807-0A2, 28A3851-0A2, 28A2024-0A0 or 0A2.



VC2124 Parallel Outputs

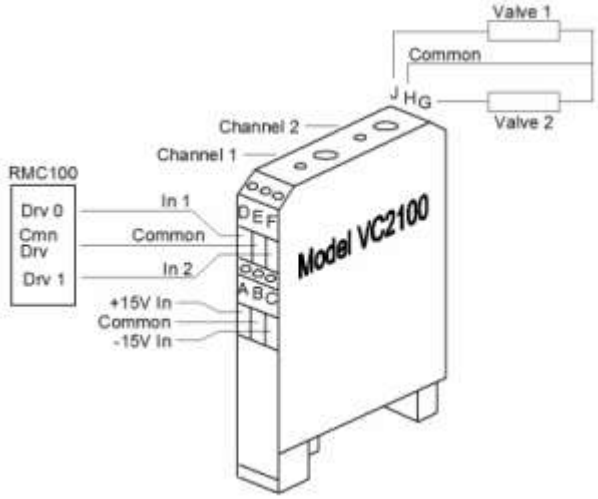
To achieve 200 mA output, wire the two channels in parallel as indicated here:



VC2100

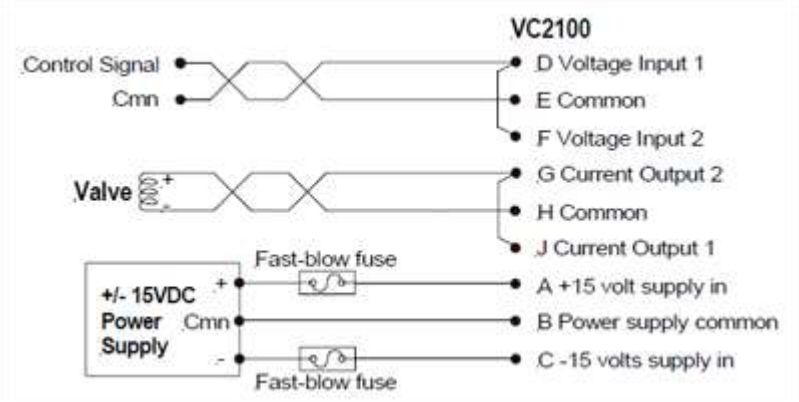
Fuse the ±15 VDC inputs with 5 A maximum, UL-listed, fast-blow fuses. For maximum protection, use two 500 mA fuses per VC2100.

For noise immunity, use twisted, shielded pairs for all connections (twisted pair with overall shield is acceptable). For best noise immunity, keep wires from the RMC to the VC2100 as short as possible and less than 98 ft (30 m), and place ferrites on all cables as close to the VC2100 as possible. Sample ferrite part numbers from Steward: 28A2029-0A0 or 0A2, 28A5131-0A2, 28A0593-0A2, 28A0807-0A2, 28A3851-0A2, 28A2024-0A0 or 0A2.



VC2100 Parallel Outputs

To achieve 200 mA output, wire the two channels in parallel as indicated here:



See Also

Output Bias | Output Scale | Control Output

7.7. Agency Compliance

This topic describes the requirements for compliance with various agencies.

Designations	
CE	RMC200: all modules RMC75: RMC75E CPU only - contact Delta for other modules
UL, CUL	RC75: all modules RMC150: all modules RMC200: all modules (except the LC8 module)
Class I, Division 2	RMC150E, RMC151E (when ordered with -HZ option). All modules except Analog (G), Stepper (QST), or Universal I/O (UI/O or U) modules. File E329627.

CE

For CE compliance and to minimize electrical interference:

- Use twisted pairs for all wiring where possible.
- Use shielded cables for all wiring.
- Keep RMC wiring separate from AC mains or conductors carrying high currents, especially high frequency switching power such as conductors between servo drives and motors or amplifiers and proportional valves.
- RMC200 only:
    - Install a ferrite on the power wires to the PS4D or PS6D power supply. Recommended ferrite is Fair-Rite 0431167281. The integrated power supply on the CPU20L does not require this ferrite.
    - Keep module closed during operation (for ESD susceptibility requirements)

UL and CUL

File number: E141684

RMC75, RMC150

The RMC75 and RMC150E are UL and CUL compliant. For UL and CUL compliance:

- Power supply must be Class 2.
- All RMC inputs and outputs must be connected to Class 2 circuits only.

RMC200

All RMC200 modules are UL and CUL compliant, except the LC8 which is pending.

Class I Division 2

File Number: E329627

Hazardous location designation Class I, Division 2, Groups A, B, C, D is available for the RMC150E. See the RMC150 Part Numbering topic for how to specify Class I, Division 2 when ordering.

- Products marked "Class I Division 2, Group A, B, C, D" are suitable for use in Class I Division 2, Groups A, B, C, and D hazardous locations and nonhazardous locations only.
- WARNING—EXPLOSION HAZARD—DO NOT DISCONNECT EQUIPMENT WHILE THE CIRCUIT IS LIVE OR UNLESS THE AREA IS KNOWN TO BE FREE OF IGNITABLE CONCENTRATIONS.
- WARNING—EXPLOSION HAZARD—SUBSTITUTION OF ANY COMPONENT MAY IMPAIR SUITABILITY FOR CLASS I, DIVISION 2.
- Maximum surrounding air temperature of 60° C.
- The RMC150E USB port is intended for configuration, programming, and troubleshooting purposes only. It should not be connected during normal operation.
- See the Wiring Guidelines for wire gauge, screw clamp torque and wire type requirements.

See Also

Wiring Guidelines | RMC150 Part Numbering

8. Command Reference

8.1. RMC Commands Overview

The RMC has a rich set of pre-programmed commands that perform anything from simple moves to complex motions to system control. Each command is represented by a name and a number.

For a list of all the RMC commands, see the Command List topic.

Issuing Commands

For details on how to send commands to the RMC, see the Issuing Commands topic.

Command Format

Each RMC command consists of a command number followed optionally by one to nine command parameters:

- **Command Number**
  Each RMC command has a number associated with it. You must use this number when you issue a command to the RMC. The number is typically included in parentheses whenever the command is mentioned. For example, the Move Absolute Command (20) has a number of 20. This number is not important when used in RMCTools.
- **Command Parameters**
  Some commands have command parameters. For example, the Move Absolute Command (20) has 5 command parameters: Position, Velocity, Accel rate, Decel Rate, and Direction. The RUN Mode (98) command has no parameters. When you issue a command, you must include any command parameters.

Command Area Registers

In the RMC register map, the command consists of one block of ten 32-bit command registers per axis. These registers are called the Command Area registers. You must write to these registers when issuing commands from a PLC or other host controller. When issuing commands from RMCTools, you need not be concerned with these addresses.

The command area registers are write-only. The Data Type for all commands and parameters is REAL.

RMC75 Command Registers

Address (IEC)	Register Name
%MD25.0-9	Axis 0 Command Registers
%MD25.0	Axis 0 Command
%MD25.1	Axis 0 Parameter 1
%MD25.2	Axis 0 Parameter 2
%MD25.3	Axis 0 Parameter 3
%MD25.4	Axis 0 Parameter 4
%MD25.5	Axis 0 Parameter 5
%MD25.6	Axis 0 Parameter 6
%MD25.7	Axis 0 Parameter 7
%MD25.8	Axis 0 Parameter 8

%MD25.9	Axis 0 Parameter 9
%MD25.10-19	Axis 1 Command Registers
%MD25.20-29	Axis 2 Command Registers
%MD25.30-39	Axis 3 Command Registers

RMC150 Command Registers

Address (IEC)	Register Name
%MD40.0-9	Axis 0 Command Registers
%MD40.0	Axis 0 Command
%MD40.1	Axis 0 Parameter 1
%MD40.2	Axis 0 Parameter 2
%MD40.3	Axis 0 Parameter 3
%MD40.4	Axis 0 Parameter 4
%MD40.5	Axis 0 Parameter 5
%MD40.6	Axis 0 Parameter 6
%MD40.7	Axis 0 Parameter 7
%MD40.8	Axis 0 Parameter 8
%MD40.9	Axis 0 Parameter 9
%MD40.10-19	Axis 1 Command Registers
%MD40.20-29	Axis 2 Command Registers
%MD40.30-39	Axis 3 Command Registers
%MD40.40-49	Axis 4 Command Registers
%MD40.50-59	Axis 5 Command Registers
%MD40.60-69	Axis 6 Command Registers
%MD40.70-79	Axis 7 Command Registers
%MD40.80-89	Axis 8 Command Registers
%MD40.90-99	Axis 9 Command Registers
%MD40.100-109	Axis 10 Command Registers
%MD40.110-119	Axis 11 Command Registers
%MD40.120-129	Axis 12 Command Registers
%MD40.130-139	Axis 13 Command Registers
%MD40.140-149	Axis 14 Command Registers
%MD40.150-159	Axis 15 Command Registers

RMC200 Command Registers

Address (IEC)	Description
%MD16.0	Axis 0 Command Registers
%MD16.1	Axis 0 Parameter 1
%MD16.2	Axis 0 Parameter 2
%MD16.3	Axis 0 Parameter 3
%MD16.4	Axis 0 Parameter 4
%MD16.5	Axis 0 Parameter 5
%MD16.6	Axis 0 Parameter 6

%MD16.7	Axis 0 Parameter 7
%MD16.8	Axis 0 Parameter 8
%MD16.9	Axis 0 Parameter 9
%MD16.10-19	Axis 1 Command Registers
%MD16.20-29	Axis 2 Command Registers
%MD16.30-39	Axis 3 Command Registers
%MD16.40-49	Axis 4 Command Registers
%MD16.50-59	Axis 5 Command Registers
%MD16.60-69	Axis 6 Command Registers
%MD16.70-79	Axis 7 Command Registers
%MD16.80-89	Axis 8 Command Registers
%MD16.90-99	Axis 9 Command Registers
%MD16.100-109	Axis 10 Command Registers
%MD16.110-119	Axis 11 Command Registers
%MD16.120-129	Axis 12 Command Registers
%MD16.130-139	Axis 13 Command Registers
%MD16.140-149	Axis 14 Command Registers
%MD16.150-159	Axis 15 Command Registers
%MD16.160-169	Axis 16 Command Registers
%MD16.170-179	Axis 17 Command Registers
%MD16.180-189	Axis 18 Command Registers
%MD16.190-199	Axis 19 Command Registers
%MD16.200-209	Axis 20 Command Registers
%MD16.210-219	Axis 21 Command Registers
%MD16.220-229	Axis 22 Command Registers
%MD16.230-239	Axis 23 Command Registers
%MD16.240-249	Axis 24 Command Registers
%MD16.250-259	Axis 25 Command Registers
%MD16.260-269	Axis 26 Command Registers
%MD16.270-279	Axis 27 Command Registers
%MD16.280-289	Axis 28 Command Registers
%MD16.290-299	Axis 29 Command Registers
%MD16.300-309	Axis 30 Command Registers
%MD16.310-319	Axis 31 Command Registers
...	...
%MD16.1270-1279	Axis 127 Command Registers

See Also

List of Commands

8.2. RMC Commands

This is a complete list of the commands currently available on the RMC, grouped by type. Each command is represented by an integer number in floating point format. This number is given in parentheses. For general command information, see the Command topic.

Note to technical writer: If you add any motion or open loop commands, make sure to update the Closed Loop Control and Open Loop Control topics.

Immediate Commands

Certain commands in the RMC are *immediate* commands. There is no limit to the number of immediate commands that can be issued to an axis per loop time, whereas a maximum of one non-immediate command per loop time can be issued to each axis. Immediate commands are usually not motion commands. The ability to issue multiple immediate commands in one loop time affects primarily the user programs, since it is difficult to issue many simultaneous commands via the external communications.

Command List

Command	Immediate	RMC75/ 150	RMC200	Closed Loop	Open Loop
General					
No-op (0)	✓	✓	✓		
Clear Faults (4)	✓	✓	✓		
Enable Controller (7)		✓	✓		
Fault Controller (8)		✓	✓		
Set Enable Output (67)	✓	✓	✓		
Enable/Disable Axis (97)		✓	✓		
RUN Mode (98)		✓	✓		
PROGRAM Mode (99)		✓	✓		
Motion - Stops					
Stop (Closed Loop) (6)		✓	✓	✓	
Stop (Open Loop) (22)		✓	✓		✓
Hold Current Position (5)		✓	✓	✓	
Closed Loop Halt (1)		✓	✓	✓	
Open Loop Halt (2)		✓	✓		✓
Direct Output Halt (3)		✓	✓		✓
Motion - Open Loop					
Direct Output (9)		✓	✓		✓
Open Loop Rate (10)		✓	✓		✓
Open Loop Absolute (11)		✓	✓		✓
Open Loop Relative (12)		✓	✓		✓
Motion - Synchronized					
Sync Move Absolute (13)		✓	✓	✓	
Sync Move Relative (14)		✓	✓	✓	
Sync Stop (17)		✓	✓	✓	
Motion - Point-to-Point					

Move Absolute (20)		✓	✓	✓	
Move Relative (21)		✓	✓	✓	
Quick Move Absolute (15)		✓	✓	✓	
Quick Move Relative (16)		✓	✓	✓	
Time Move Absolute (23)		✓	✓	✓	
Time Move Relative (24)		✓	✓	✓	
Advanced Time Move Absolute (26)		✓	✓	✓	
Advanced Time Move Relative (27)		✓	✓	✓	
Move Absolute (I-PD) (28)		✓	✓	✓	
Move Relative (I-PD) (29)		✓	✓	✓	
Motion - Gearing					
Gear Absolute (25)		✓	✓	✓	
Gear Pos (Clutch by Time) (30)		✓	✓	✓	
Gear Vel (Clutch by Time) (31)		✓	✓	✓	
Gear Pos (Clutch by Distance) (32)		✓	✓	✓	
Advanced Gear Move (33)		✓	✓	✓	
Phasing (34)		✓	✓	✓	
Geared Slave Offset (35)		✓	✓	✓	
Gear Pos (Clutch by Rate) (39)		✓	✓	✓	
Track Position (57)		✓	✓	✓	
Track Position (I-PD) (58)		✓	✓	✓	
Motion - Specialty					
Speed at Position (36)		✓	✓	✓	
Sine Start (72)		✓	✓	✓	
Sine Stop (73)		✓	✓	✓	
Change Master (79)	✓	✓	✓	✓	
Change Target Parameter (80)	✓	✓	✓	✓	
Curve Add (82)		✓	✓		
Curve Delete (83)		✓	✓		
Curve Delete Except (84)		✓	✓		
Curve Delete All (85)		✓	✓		
Curve Start (86)		✓	✓	✓	
Curve Start Advanced (88)		✓	✓	✓	
Motion - Velocity					
Move Velocity (37)		✓	✓	✓	
Move Velocity (I-PD) (38)		✓	✓	✓	
Motion - Transitions					
Transition Disable (55)	✓	✓	✓		
Transition Rate (56)	✓	✓	✓		

Command	**Immediate**	**RMC75/ 150**	**RMC200**	**Closed Loop**	**Open Loop**

Pressure/Force - Standard					
Ramp Pressure/Force (Rate) (18)		✓	✓	✓ *	
Hold Current Pressure/Force (19)		✓	✓	✓	
Ramp Pressure/Force (S-Curve) (41)		✓	✓	✓ *	
Ramp Pressure/Force (Linear) (42)		✓	✓	✓ *	
Stop Pressure/Force (43)		✓	✓	✓ *	
Enter Pressure/Force Control (Auto) (44)		✓	✓	✓	
Enter Pressure/Force Control (Time) (45)		✓	✓	✓	
Enter Pressure/Force Control (Rate) (46)		✓	✓	✓	
Pressure/Force - Limit					
Set Pressure/Force Limit Mode (40)	✓	✓	✓	✓	
Pressure/Force - Specialty					
Gear Absolute (Prs/Frc) (59)		✓	✓	✓ *	
Transition Disable (Prs/Frc) (63)	✓	✓	✓		
Transition Rate (Prs/Frc) (64)	✓	✓	✓		
Sine Start (Prs/Frc) (76)		✓	✓	✓ *	
Sine Stop (Prs/Frc) (77)		✓	✓	✓ *	
Change Target Parameter (Prs/Frc) (81)	✓	✓	✓	✓ *	
Curve Start (Prs/Frc) (87)		✓	✓	✓ *	
Curve Start Advanced (Prs/Frc) (89)		✓	✓	✓ *	
*These commands will not enter pressure/force control. The axis must first be in pressure/force control.					
Set Parameters					
Offset Position (47)		✓	✓		
Set Actual Position (49)		✓	✓		
Set Target Position (48)		✓	✓		
Set Actual Pressure/Force (65)		✓	✓		
Select Gain Set (75)	✓		✓		
Set Pos/Vel Ctrl Mode (68)	✓	✓	✓		
Feed Forward Adjust (69)	✓	✓	✓		
Integrator Adjust (70)	✓	✓	✓		
Set Integrator Mode (71)	✓	✓	✓		
Set Control Direction (96)		✓	✓		
Read Register (111)	✓	✓			
Write Register (112)	✓	✓			
Programming					
Set Discrete Output (60)	✓	✓	✓		
Clear Discrete Output (61)	✓	✓	✓		
Toggle Discrete Output (62)	✓	✓	✓		

Start Task (90)		✓	✓		
Stop Task (91)		✓	✓		
Plots					
Start Plot (100)	✓	✓	✓		
Stop Plot (101)	✓	✓	✓		
Trigger Plot (102)	✓	✓	✓		
Rearm Plot (103)	✓	✓	✓		
Enable/Disable Plot Trigger (104)	✓	✓	✓		
Step Editor Commands					
Expression (113)	✓	✓	✓		
System					
Arm Home (50)		✓	✓		
Disarm Home (51)		✓	✓		
Arm Registration (52)		✓	✓		
Disarm Registration (53)		✓	✓		
Learn Z Alignment (54)		✓	✓		
Pause/Resume Log (95)	✓	✓	✓		
Arm Event Timer (105)	✓		✓		
Disarm Event Timer (106)	✓		✓		
Update Flash (110)	✓	✓	✓		
Save Controller Image (120)	✓		✓		
Restore Controller Image (121)	✓		✓		

See Also
Commands Overview

8.3. General Commands

8.3.1. Command: No-op (0)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This is a command that does nothing. Writing a No-op (0) command to the RMC will do nothing and will not be recorded by the RMC. It is used in the following cases:

- When using **Send All** in the RMCTools Command Tool to send commands simultaneously to multiple axes, choose the No-op (0) command for the axes that you don't want to issue commands to. No commands will be sent to those axes.

See Also

List of Commands | Commands Overview

8.3.2. Command: Clear Faults (4)

Supported Axes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command clears all latched error bits, unless the underlying condition still exists. It also clears the Halted and External Halt status bits unless error bits remain set that would otherwise set these bits.

This command is often unnecessary because motion commands will also clear the above-mentioned status and error bits, unless the underlying condition still exists. However, this command may be useful to clear the faults just to see which conditions went away.

Except for its effect on the error bits and halt status bits, this command does not affect the state of the axis. That is, it will not stop a halting axis from continuing to ramp down its Control Output, and it will not switch an axis into or out of closed loop.

See Also

Error Bits | List of Commands | Commands Overview

8.3.3. Command: Enable Controller (7)

Supported Axes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command enables all the axes on the RMC. It is typically used after starting up the RMC. No motion commands, other than the Direct Output (9) command, are allowed on an axis if is not enabled.

This command does the following:

- Sets the Enabled bit in the Controller Status register.
- **RMC75/150:** Sets the Enabled status bit on all axes.
- **RMC200:** All the axes become enabled if the **Auto-Enable Axes** option is checked on the RUN/Disabled page of the Programming Properties. For each axis that was previously disabled, the Enable Output is turned on.

These steps enable most motion commands and Auto Stops on the RMC. This command does not clear any faults.

Note:
The RUN Mode (98) command also enables all axes, and puts the controller in RUN mode. See the Run/Program Mode topic for details.

Tip:
If the **Start Controller in RUN Mode** option in the Programming Properties is checked, the RMC will automatically start up with all axes enabled and in RUN mode. This option is useful if the RMC is used in a stand-alone application.

Note:
For the RMC75 and RMC150, this command does not have any effect on the Enable Output. It does not turn it on or off. Use the Set Enable Output (67) command to turn the Enable Output on or off.

See Also

Enable/Disable Axis (97) Command | RUN Mode (98) Command | List of Commands | Commands Overview

8.3.4. Command: Fault Controller (8)

Supported Axes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command stops all motion on the controller. To send this command from RMCTools, click the **Fault Controller** button on the toolbar.

This command does the following:

- **Halts all axes**
  Performs a Direct Output Halt on each axis.
- **RMC75/150: Puts the RMC in PROGRAM mode**
  This stops all Tasks and the Program Triggers. In order to start Tasks and the Program Triggers again, you must put the RMC into RUN mode.

- **RMC200: Puts the RMC in Disabled mode**
  This stops all Tasks and the Program Triggers, and disables all the axes. In order to send commands , you must put the RMC into Run or Program mode. In order to start Tasks and the Program Triggers again, you must put the RMC into RUN mode.

Why bother?

If you want to shut down the RMC because a major problem occurred, use this command, because you can be sure it will stop all motion on all axes, along with all user programs, and the Program Triggers. If the RMC is in RUN mode and you use a regular Halt instead of this command, the Program Triggers can still start a User Program, which may start motion again.

See Also

List of Commands | Commands Overview

8.3.5. Command: Set Enable Output (67)

Supported Axes: Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Enable/Disable:** Enable (1), Disable (0)	0 or 1

Description

This command enables or disables the Enable Output on axes that have an Enable Output.

To specify the behavior of the Enable output switch when the Enable Output is enabled, see the Enable Output Behavior topic.

Why Bother?

You can wire the Enable Output to the enable input on a motor drive or hydraulic valve. If an error occurs for which the corresponding Auto Stop is set to Direct Output Halt, the Enable Output will be turned off, and the drive or valve will shut off. This is a good safety precaution if your transducer fails, for example. Use this command to turn the Enable Output back on.

See Also

Enable Output | List of Commands | Commands Overview

8.3.6. Command: Enable/Disable Axis (97)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description
1	**Enable/Disable:** Enable (1), Disable (0)

Description

This command enables or disables a single axis. The Enabled Status bit indicates whether the axis is enabled or disabled. To enable all axes at once, use the Enable Controller (7) command. When this command is issued with the Disable parameter, it will turn off the Enabled Status bit and cause a Direct Output Halt on the axis.

RMC75/150:

- This command does not have any effect on the Enable Output. It does not turn it on or off.
- When the Enabled Status bit is off, the axis will not accept any motion commands other than the Direct Output (9) command.

RMC200:

- This command will affect the Enable Output. If the axis is currently disabled and this command is sent to enable the axis, the Enable Output will turn on. If the axis is currently enabled and this command is sent to disable the axis, the Enable Output will turn off.
- When the Enabled Status bit is off, the axis will not accept any motion commands.

Certain feedback parameters in the RMC require the axis' Enabled Status bit to be off before they can be changed. RMCTools automatically takes care of this. However, if you are changing parameters in the RMC from a PLC or other host controller, you may need to use this command.

See Also

Enable Controller (7) Command | List of Commands | Commands Overview

8.3.7. Command: RUN Mode (98)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command puts the RMC in RUN mode. If the RMC is already in RUN Mode, nothing happens when this command is issued. If the RMC is in PROGRAM mode when this command is issued, the following happens:

- The RMC is put into RUN mode.
- **RMC75/150:** All the axes become enabled if they were not previously enabled. This means that the Enabled status bit turns on for each axis, exactly as if the Enable Controller (7) command had been issued.

- **RMC200:** All the axes become enabled if the **Auto-Enable Axes** option is checked on the RUN/Disabled page of the Programming Properties.

Entering RUN mode will *not* affect the motion on the axes. The axes will continue doing what they were doing. Motion commands can be issued to an axis whether or not it is in RUN mode.

For details on RUN mode, see the RUN/PROGRAM/Disabled Mode topic.

See Also

PROGRAM Mode (99) | RUN/PROGRAM Mode | List of Commands | Commands Overview

8.3.8. Command: PROGRAM Mode (99)

Supported Axes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command puts the RMC in PROGRAM mode, which stops all running Tasks and the Program Triggers. Tasks cannot be restarted until the RMC is put back into RUN Mode.

Entering PROGRAM mode will not affect the motion on the axes. The axes will continue doing what they were doing, and motion commands can still be issued to the axis.

For more details, see the RUN/PROGRAM/Disabled Mode topic.

See Also

RUN Mode (98) | RUN/PROGRAM Mode | List of Commands | Commands Overview

8.4. Motion Commands

8.4.1. Motion Commands

The commands listed below are called *motion commands.* They are grouped by type. See the List of Commands topic for all RMC commands.

Command
Stops
Stop (Closed Loop) (6)
Stop (Open Loop) (22)
Hold Current Position (5)

Command
Specialty
Speed at Position (36)
Sine Start (72)
Sine Stop (73)

Closed Loop Halt (1)
Open Loop Halt (2)
Direct Output Halt (3)
Open Loop
Direct Output (9)
Open Loop Rate (10)
Open Loop Absolute (11)
Open Loop Relative (12)
Synchronized
Sync Move Absolute (13)
Sync Move Relative (14)
Sync Stop (17)
Point-to-Point
Move Absolute (20)
Move Relative (21)
Quick Move Absolute (15)
Quick Move Relative (16)
Time Move Absolute (23)
Time Move Relative (24)
Advanced Time Move Absolute (26)
Advanced Time Move Relative (27)
Move Absolute (I-PD) (28)
Move Relative (I-PD) (29)
Gearing
Gear Absolute (25)
Gear Pos (Clutch by Time) (30)
Gear Vel (Clutch by Time) (31)
Gear Pos (Clutch by Distance) (32)
Advanced Gear Move (33)
Phasing (34)
Geared Slave Offset (35)
Gear Pos (Clutch by Rate) (39)
Track Position (57)
Track Position (I-PD) (58)

Change Master (79)
Change Target Parameter (80)
Curve Start (86)
Curve Start Advanced (88)
Velocity
Move Velocity (37)
Move Velocity (I-PD) (38)
Transitions
Transition Disable (55)
Transition Rate (56)
Pressure/Force - Standard
Ramp Pressure/Force (Rate) (18)
Hold Current Pressure/Force (19)
Ramp Pressure/Force (S-Curve) (41)
Ramp Pressure/Force (Linear) (42)
Stop Pressure/Force (43)
Enter Pressure/Force Control (Auto) (44)
Enter Pressure/Force Control (Time) (45)
Enter Pressure/Force Control (Rate) (46)
Pressure/Force - Limit
Set Pressure/Force Limit Mode (40)
Pressure/Force - Specialty
Gear Absolute (Prs/Frc) (59)
Transition Disable (Prs/Frc) (63)
Transition Rate (Prs/Frc) (64)
Sine Start (Prs/Frc) (76)
Sine Stop (Prs/Frc) (77)
Change Target Parameter (Prs/Frc) (81)
Curve Start (Prs/Frc) (87)
Curve Start Advanced (Prs/Frc) (89)

See Also

List of Commands | Commands Overview

8.4.2. Stops

8.4.2.1. Command: Stop (Closed Loop) (6)

Supported Axes:	Position and Velocity Axes
Supported Control Modes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Deceleration Rate** (pos-units/s^2)	>0

Description

This command stops the axis in closed loop control, bringing the velocity to zero at the specified **Deceleration Rate**. The behavior of this command depends on the type of control of the axis at the moment this command is received:

- **Position PID**
  The Target Position stops by ramping the current Target Velocity to zero using the specified **Deceleration Rate**, while remaining in position control.
- **Position I-PD**
  If the last motion command was Move Absolute (I-PD) (28) or Move Relative (I-PD) (29), the Target Position will be moved toward the current position to ensure a quicker yet smooth stop. Otherwise, the Target Position will be stopped by ramping the current Target Velocity to zero using the specified **Deceleration Rate**, while remaining in position control.
- **Velocity PID**
  The Target Velocity will ramp down from the current velocity to zero using the specified **Deceleration Rate**, while remaining in velocity control.
- **Velocity I-PD**
  If the last motion command was Move Velocity (I-PD) (38), the Target Velocity will stop immediately. Otherwise, the velocity will ramp down from the current velocity to zero using the specified **Deceleration Rate**, while remaining in velocity control.
- **Open Loop**
  The axis transitions to closed-loop control and ramps the velocity to zero using the specified **Deceleration Rate** in V/sec.

If you wish to stop the axis, but not remain in closed loop control, consider using the Stop (Open Loop) (22).

Closed Loop Stopping of the Actual versus Target Position

The Stop (Closed Loop) command will stop the Target Position. Therefore, if the Actual is lagging significantly behind the target, the axis will actually continue moving until it reaches the stopped Target Position. If the Actual is lagging and you want the Actual Position to stop immediately, you can put the axis in open loop control (which sets the Target Position to the Actual Position) for one loop time, then issue the Stop (Closed Loop) command. This can be done in a user program. In the first step, issue the Open Loop Rate command. In the Output command parameter, put the Control Output. This will keep the Control Output from changing drastically. Then choose the Immediate link type and make the next step issue a Stop (Closed Loop) command. On most well-tuned systems, this issue is typically not a concern.

Status Bits

In Position Bit

The In Position status bit will not be set. Do not attempt to use the In Position bit to tell if the axis has stopped after a Stop (Closed Loop) (6) command.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has stopped.

Target Generator State A and B bits

B	A	Description
0	0	Done
0	1	Reserved
1	0	Reserved
1	1	Decelerating to zero velocity

See Also

Stop (Open Loop) (22) | Hold Current Position (5) Command | Stop Pressure/Force (43) | List of Commands | Commands Overview

8.4.2.2. Command: Stop (Open Loop) (22)

Supported Axes:	Control Axes
Supported Control Modes:	All
Firmware Limitations:	RMC75/150: 3.40 or newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Ramp Rate**	> 0

Description

This command puts the axis in open loop control and ramps the Control Output linearly to zero volts at the rate specified by the **Ramp Rate**. If the Output Bias is set, the Control output will ramp to the Output Bias value.

Notice that when the axis is in open loop, it may drift, even if the Control Output is zero. To stop the axis and hold position in closed-loop control, use the Stop (Closed Loop) (6) command.

This command is identical to the Open Loop Rate (10) command with a Requested Output of zero.

Default Ramp Rate

When choosing the Stop (Open Loop) command in the Command Tool, a User Program, or Shortcut Commands, the default value of the Ramp Rate is 100 V/sec. You should consider what value of ramp rate your application requires. 100 V/sec may be a very abrupt stop on some systems.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Control Output has reached zero (or the Output Bias value).

Target Generator State A and B bits

B	A	Description
0	0	The open loop stop is complete
0	1	reserved
1	0	reserved
1	1	Ramping Control Output toward 0 volts

See Also

Stop (Closed Loop) (6) | Open Loop Rate (10) | List of Commands | Commands Overview

8.4.2.3. Command: Hold Current Position (5)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to send commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Control Behavior** • Standard (0) • Reset (1) • Maintain (2)	a valid integer as described

Description

This command switches the axis to closed loop, unless it already is, and sets the Target and Command Positions equal to the Actual Position, and holds the current position. No ramping is done on the Target Position. This command should not be sent when the axis is moving at a significant speed, since this command will stop it suddenly. To slowly stop the axis, use the Stop (Closed Loop) (6) command instead.

It is not necessary to send this command after a closed loop command that will already hold the final position, such as the Move Absolute (20) command. Doing so will actually cause the axis to hold the current *actual* position, which is likely different from the initial commanded position.

The In Position Status bit is valid with this command.

Control Behavior

The **Control Behavior** parameter determines the way that the control mode is affected by this command. This parameter can have the following values:

- **Standard (0)**

  Not recommended for new designs.

The controller will try to maintain the current control output, setting the integrator or filtered output accordingly, and also taking into account the Velocity Feed Forward that results from the value of the Target Velocity at the time this command was sent. This means the Control Output will be affected by any noise on the velocity, which is typically undesirable since it may make the value of the Control Output unpredictable.

- **Reset (1)**

  The control output, integrator, and output filter are reset to zero, ignoring their states prior to this command being issued. This occurs in both Position PID and Position I-PD whether or not the axis is already in closed loop control. Notice that the Output Bias will still be applied.

  This option is recommended for situations where you want the Control Output to start at zero volts unconditionally.

  The **Reset** option is not available in RMC75/150 firmware versions prior to 3.53.3.

- **Maintain (2)**

  The controller will try to maintain the current control output, setting the integrator or filtered output accordingly. This option is recommended for situations where you want the Control Output to smoothly maintain it's value, such as transitioning from pressure/force to position control while a significant amount of control output is required to maintain position.

  This option is preferred over the Standard (0) option, since this option does not include the velocity in its Control Output calculation, and therefore will give a more predictable, smooth transition on the Control Output. Since this command assumes the axis is not moving much prior to the command being sent, and will not have any velocity afterward, the velocity typically need not be taken into account.

  The **Maintain** option is not available in RMC75/150 firmware versions prior to 3.65.0 or RMC200 firmware prior to 1.02.0.

Status Bits

In Position Bit

When the Actual Position is within the In Position Tolerance window from the Target Position, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit turns on immediately, indicating the target is complete.

Target Generator State A and B bits

These will both be off.

See Also

Stop (Closed Loop) (6) command | List of Commands | Commands Overview

8.4.2.4. Command: Closed Loop Halt (1)

Supported Axes:	All
Supported Control Modes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command initiates a Closed Loop Halt. See the Closed Loop Halt topic for details.

Status Bits

In Position Bit

The In Position status bit will not be set. Do not attempt to use the In Position bit to tell if the axis has stopped after a Closed Loop halt (1) command.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has stopped.

Target Generator State A and B bits

B	A	Description
0	0	Done
0	1	Reserved
1	0	Reserved
1	1	Decelerating to zero velocity

See Also

Halts Overview | Closed Loop Halt | List of Commands | Commands Overview

---

8.4.2.5. Command: Open Loop Halt (2)

Supported Axes:	All
Supported Control Modes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command initiates an Open Loop Halt. See the Open Loop Halt topic for details.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Control Output has reached zero.

Target Generator State A and B bits

B	A	Description

0	0	Done
0	1	Reserved
1	0	Reserved
1	1	Ramping Control Output toward 0 volts

See Also

Halts Overview | Open Loop Halt | List of Commands | Commands Overview

8.4.2.6. Command: Direct Output Halt (3)

Supported Axes:	All
Supported Control Modes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command initiates a Direct Output Halt. See the Direct Output Halt topic for details.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Control Output has reached zero.

Target Generator State A and B bits

B	A	Description
0	0	Done
0	1	Reserved
1	0	Reserved
1	1	Ramping Control Output toward 0 volts

See Also

Halts Overview | Direct Output Halt | List of Commands | Commands Overview

8.4.3. Open Loop

8.4.3.1. Command: Direct Output (9)

Supported Axes:	Control Axes
Supported Control Modes:	All

Caution: The Direct Output (9) command disables the autostop features on the axis! Use this command carefully!

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Output** (RMC75/150: V) (RMC200: %)	RMC75/150: -10≤V≤10 RMC200: -100≤%≤100
2	**Output Ramp Rate** (RMC75/150: V/sec) (RMC200: %/sec)	>0

"%" refers to the percent of Control Output.

Description

This command switches the axis to open loop control and ramps the Control Output linearly to the **Requested Output** at the rate specified in the **Output Ramp Rate**. This command will not be affected by any errors on the axis. If an error bit turns on, the Control Output will still be the value as commanded by this command.

This command turns on the Direct Output Status bit.

Note:
This command is identical to the Open Loop Rate (10) command, except that this command disables all auto stops until another motion command is issued. As such, this command is intended for testing the Control Output and *not* for actual motion control. **For motion control, the Open Loop Rate (10) command should be used instead.**

Default Ramp Rate

When choosing the open Loop rate command in the Command Tool, a User Program, or Shortcut Commands, the default value of the Ramp Rate is 100 V/sec or 1000 %/sec. You should consider what value of ramp rate your application requires. The default value may be a very abrupt stop on some systems.

Why Bother?

Use the Direct Output command to test the Control Output when you are setting up the system.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Control Output has reached the **Requested Output**.

Target Generator State A and B bits

B	A	Description
0	0	Constant Control Output at 0 volts
0	1	Ramping Control Output away from 0 volts
1	0	Constant Control Output at a non-zero voltage
1	1	Ramping Control Output toward 0 volts

See Also

Open Loop Rate (10) | List of Commands | Commands Overview

8.4.3.2. Command: Open Loop Rate (10)

Supported Axes:	Control Axes
Supported Control Modes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Output** (RMC75/150: V) (RMC200: %)	RMC75/150: -10≤V≤10 RMC200: -100≤%≤100
2	**Output Ramp Rate** (RMC75/150: V/sec) (RMC200: %/sec)	>0

"%" refers to the percent of Control Output.

Description

This command puts the axis in open loop control and ramps the Control Output linearly to the **Requested Output** at the rate specified by the **Output Ramp Rate**.

This command is identical to the Direct Output (9) command, except that this command does not disable the Auto Stops. Therefore, the Open Loop Rate (10) command should be used in all cases where open loop is required during the regular machine cycle, and the Direct Output (9) command should only be used to test the Control Output during setup.

When choosing the Open Loop Rate (10) command, the default value of the Ramp Rate is 100 V/sec or 1000 %/sec, which may be a very abrupt stop on some systems. You should consider what value of ramp rate your application requires, and not necessarily use the default value.

Why Bother?

Use this command when you want to give an Open Loop Control Output to the axis. Open Loop is good for making the axis move when you don't care about going exactly at a certain speed or reaching an exact position.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Control Output has reached the **Requested Output**.

Target Generator State A and B bits

B	A	Description
0	0	Constant Control Output at 0 volts
0	1	Ramping Control Output away from 0 volts
1	0	Constant Control Output at a non-zero voltage

1	1	Ramping Control Output toward 0 volts

See Also
Direct Output (9) | List of Commands | Commands Overview

8.4.3.3. Command: Open Loop Absolute (11)

Supported Axes:	Position Control Axes
Supported Control Modes:	All

Note:
This command is a specialized Open Loop command. Do not use before fully understanding how it works! To simply issue an Open Loop output, use the Open Loop Rate (10) command.

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Output** (RMC75/150: V) (RMC200: %)	RMC75/150: -10≤V≤10 RMC200: -100≤%≤100 See **Limitations** below
2	**Requested Position** (position-units)	any See **Limitations** below
3	**Direction** • Negative* (-1) • Nearest (0) • Positive* (1) • Current* (2) • Absolute* (3) * These options are intended for use with rotary axes. However, all options are available on linear axes, but have no effect. For more details, see the Using Rotary Motion topic.	a valid integer as described

"%" refers to the percent of Control Output.

Description

This command first puts the axis in open loop. Then, it maps the Control Output as a function of the Actual Position of the axis. As the Actual Position moves, the Control Output ramps from its value at the time the command was issued to the **Requested Output**. It reaches the **Requested Output** at the **Requested Position**. The Control Output is not ramped in a linear fashion; rather, it is ramped such that the position of the axis is ramped somewhat linearly.

This command sets up a mapping of the Control Output as a function of position. This mapping is valid until another command is issued. If the Actual Position reaches the Requested Position, the mapping is still valid. If the Actual Position moves backward, the Control Output will still follow the mapping.

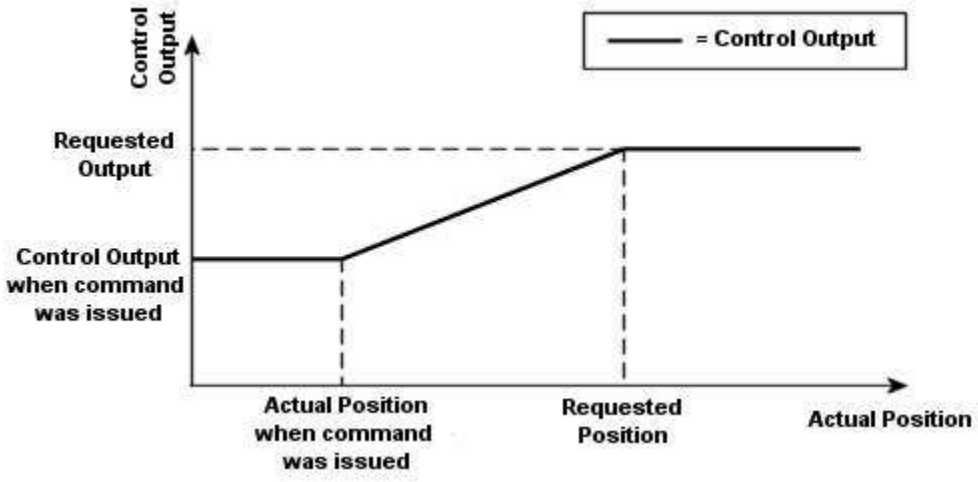
Limitations

- The **Requested Output** must be of the same sign as the Control Output at the time the command is issued. The Control Output cannot change sign during the ramping. An incorrect **Requested Output** will cause a command error.
- The **Requested Position** must be set such that the move direction in position units matches the sign of the Control Output. That is, if the **Requested Output** is positive, the **Requested Position** must be set such that the axis will move in the position direction.
- In Unidirectional Mode, both the Control Output at the time the command is issued and the **Requested Output** must be greater than or equal to zero.
- In Positive Unidirectional Mode, the change in position (**Requested Position** minus Actual Position when the command is issued) must be greater than zero.
- In Negative Unidirectional Mode, the change in position **(Requested Position** minus Actual Position when the command is issued) must be less than zero.

Note the following:

- This command does *not* ramp the Control Output based on time.
- If the axis does not move, the Control Output will not change.
- This command cannot be used for starting from a stop because no Control Output will be applied since the position does not move, and conversely, the position does not move because there is no Control Output.
- This command is for velocity drive axes only. This command should not be used for a torque drive.

The following diagram illustrates how this command maps the Control Output from the position:



If this command is used with a Requested Output of zero, notice after reaching position, the position will probably drift because it is in open loop. If you need to hold position, switch to closed loop control, or adjust the Output Bias to minimize drifting.

Why Bother?

This command is often used to stop the axis in Open Loop, instead of an Open Loop Rate. With this command you'll know approximately where the axis will stop, which you don't know with an Open Loop Rate. To do this, the Requested Output must be 0 (or close to zero).

Another use for this command is making an open loop profile based on position, which can be used in injection molding.

Another application is clamping. If the Requested Position is beyond the clamping point, some Control Output will always be applied when it is clamping. When moving into the clamp, the axis will be slowing down because of the ramp and won't hit so hard.

See Also

Open Loop Relative (12) | List of Commands | Commands Overview

8.4.3.4. Command: Open Loop Relative (12)

Supported Axes:	Position Control Axes
Supported Control Modes:	All

Note:
This command is a specialized Open Loop command. Do not use before fully understanding how it works! To simply issue an Open Loop output, use the Open Loop Rate (10) command.

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Output** (RMC75/150: V) (RMC200: %)	RMC75/150: -10≤V≤10 RMC200: -100≤%≤100 See **Limitations** in Open Loop Absolute (11).
2	**Requested Distance** (pos-units, signed)	Any. See **Limitations** in Open Loop Absolute (11).

"%" refers to the percent of Control Output.

Description

This command ramps the open loop Control Output from its current value to the **Requested Output** while the axis moves the distance specified by the **Requested Distance**.

This command is identical to the Open Loop Absolute (11) except that the Requested Distance is relative to the Actual Position when the command is issued, not an absolute position. See the Open Loop Absolute (11) topic for details.

Limitations

- The **Requested Output** must be of the same sign as the Control Output at the time the command is issued. The Control Output cannot change sign during the ramping. An incorrect **Requested Output** will cause a command error.
- The **Requested Distance** must be set such that the move direction in position units matches the sign of the Control Output. That is, if the **Requested Output** is positive, the **Requested Distance** must be set such that the axis will move in the position direction.
- In Unidirectional Mode, both the Control Output at the time the command is issued and the **Requested Output** must be greater than or equal to zero.

- In Positive Unidirectional Mode, the **Requested Distance** must be greater than zero.
- In Negative Unidirectional Mode, the **Requested Distance** must be less than zero.

See the Open Loop Absolute (11) topic for more details.

See Also
Open Loop Absolute (11) | List of Commands | Commands Overview

8.4.4. Synchronized

8.4.4.1. Command: Sync Move Absolute (13)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Position** (position-units)	any
2	**Requested Speed** (position-units/s)	>0
3	**Acceleration Rate** (position-units/s^2)	>0
4	**Deceleration Rate** (position-units/s^2)	>0
5	**Direction** • Negative* (-1) • Nearest (0) • Positive* (1) • Current* (2) • Absolute* (3) * These options are intended for use with rotary axes. However, all options are available on linear axes, but have no effect. For more details, see the Using Rotary Motion topic.	a valid integer as described
6	**Sync Group**	RMC75: 0 to 10 RMC150: 0 to 10 RMC200: 0 to 31

Description

This command initiates a ratioed synchronized move on the axis. In this type of synchronization, the motion of all the axes in the same **Sync Group** are synchronized such that all the axes start and stop moving simultaneously, and at any point during the move, each axis has completed the same percentage distance (or ratio) of its move. For example, if three axes start at 0 inches, and

are to move to 2, 4, and 6 inches, respectively, then at any point in the move, these axes' positions will be at a 1:2:3 ratio.

Synchronized moves are useful when several actuators are moving a rigid structure around a fulcrum. The axes do not need to start or stop at the same positions.

How to Start a Sync Move

To start a synchronized move, issue the Sync Move Absolute (13) or Sync Move Relative (14) commands simultaneously to each axis that you wish to include in the synchronized move. Refer to the **Issuing Sync Move Commands Simultaneously** section below. The **Sync Group** command parameter must be set to the same value for each axis to be synchronized together.

The Position, Speed, Acceleration, and Deceleration command parameters for all synchronized axes in the same group need not be the same. When the commands are issued, all the command parameters are evaluated and the RMC selects the maximum speeds and velocities for each axis such that (1) all axis positions are ratioed evenly throughout the move such that the axes arrive at the same time, (2) no individual axis exceeds its speed, acceleration, or deceleration parameters. Therefore, when issuing the Sync move, just set the velocities and accelerations to the maximum value you want on each axis, then the RMC will automatically ratio them so the axes remain in sync and arrive simultaneously.

For best results, all axes to be synchronized should be stopped when the Sync Move command is issued, although they are not required to be (this is required in firmware prior to 3.31.0). In closed loop control, "stopped" means the Target Velocity is zero; in open loop control, "stopped" means the Stopped Status Bit is be set. If the axes are moving (and not already synchronized) when a Sync Move is issued, this may cause discontinuities in the velocity and may jerk the axis. Changing directions while a sync move is in progress is not recommended.

Sync Group Details

The **Sync Group** command parameter is used to define which axes should be synchronized together. Valid group numbers are zero (0) to ten (10). It is possible to have multiple sync groups in progress at the same time. The sync group will be valid for the duration of the synchronized move. Once the Target Positions reach the Command Positions, the sync move will be complete, the axes no longer belong to that sync group, and the sync group number will be available for another synchronized move.

Changing a Sync Move In Progress

It is possible to speed up or slow down a sync move that is in progress while still maintaining the same synchronization ratio. To do so, issue sync move commands to all the synchronized axes such that the final position for each axis are the same as for the initial sync move command. It is not recommended that you change the final commanded positions of a sync move in progress. Doing so will likely cause a step jump in target speed on one or more axes, and is not allowed prior to 3.31.0 firmware.

Exiting a Sync Move In Progress

To remove an axis from a sync move in progress, issue any non-sync motion command to the axis. The motion of the remaining axes in the group will be re-evaluated based on their requested speeds and accelerations. This can cause the profile to speed up or use a different acceleration/deceleration value.

Synchronized Stop

If you need to stop a sync move in progress, use the Sync Stop (17) command to stop the axes while maintaining the synchronization.

Halting

If any axis in a sync group receives a Closed Loop Halt, then all axes in that sync group will halt while maintaining their sync ratios. When the halt is complete, the axes will no longer belong to the sync group. An Open Loop Halt or Direct Output Halt will immediately remove the axis from the sync group, and halt all the axis in that sync group with the same halt.

Notice that an axis can also be a member of a Halt Group as defined by the Halt Group Number axis parameter. Halts in any group will propagate recursively to other groups. If an axis halts, all

axes in that group will halt, and so will the axes in any other sync groups or halt groups that the other axes in the group are part of.

S-Curves vs. Trapezoidal

If the Requested Jerk Axis Parameter is non-zero for any of the axes in the sync group, then all the axes will use s-curves. If the Requested Jerk Axis Parameter is zero for all of the axes in the sync group, then all the axes will use a trapezoidal profile.

When using s-curves, if the sync move commands are re-issued when the axes are decelerating, it may cause the axes to overshoot the requested positions.

Issuing Sync Move Commands Simultaneously

This section describes various methods of issuing the Sync Move commands simultaneously.

- **From RMCTools**
  In the Command Tool, set up the sync commands for each desired axis. Make sure all other axis have a No-op(0) command, then click the **Send All** button. If the **Send All** button is not visible, right-click in the Command Tool and choose **Show Toolbar**.
- **From a User Program**
  Including the Sync Move commands in the same step of a user program will ensure that they are issued simultaneously.
- **From a Host Controller (PLC, HMI, etc.)**
  The Sync Move commands must be written to the RMC such that they are all included in the same communication packet. Some PLCs and many HMIs are not capable of doing this. If this is the case, you will need to create a user program to issue the commands simultaneously. Then use the PLC or HMI to start the user program.

Status Bits

In Position Bit

When the Target Positions of all axes in the sync group reach their final positions, and the Actual Position of this axis is within the In Position Tolerance window, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates that the Target Positions of all axes in the sync group have reached their final positions. Notice that this bit does not indicate whether the Actual Position has reached the Requested Position.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Acceleration
1	0	Constant Velocity
1	1	Deceleration

See Also

Sync Move Relative (14) | Sync Stop (17)

8.4.4.2. Command: Sync Move Relative (14)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Distance** (position-units)	any
2	**Requested Speed** (position-units/s)	>0
3	**Acceleration Rate** (position-units/s^2)	>0
4	**Deceleration Rate** (position-units/s^2)	>0
5	**Relative To:** • Command Position (0) • Actual Position (1) • Target Position (2)	a valid integer as described
6	**Sync Group**	RMC75: 0 to 10 RMC150: 0 to 10 RMC200: 0 to 31

Description

This command is identical to the Sync Move Absolute (13), except that the final Command Position is computed by adding the **Requested Distance** to the current value of the quantity specified in the **Relative To** parameter. The current value is the value at the time the RMC received the command.

Relative To Options:

- **Command Position (0)**
  The final Command Position is computed by adding the **Requested Distance** to the current Command Position.
- **Actual Position (1)**
  The final Command Position is computed by adding the **Requested Distance** to the current Actual Position.
- **Target Position (2)**
  The final Command Position is computed by adding the **Requested Distance** to the current Target Position.

For more details, see the Sync Move Absolute (13) command.

Status Bits

In Position Bit

When the Target Positions of all axes in the sync group reach their final positions, and the Actual Position is within the In Position Tolerance window, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates that the Target Positions of all axes in the sync group have reached their final positions. Notice that this bit does not indicate whether the Actual Position has reached the Requested Position.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Acceleration
1	0	Constant Velocity
1	1	Deceleration

See Also

Sync Move Absolute (13) | Sync Stop (17)

8.4.4.3. Command: Sync Stop (17)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD
Firmware Limitations:	RMC75/150: 3.00 or newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Deceleration Rate** (pos-units/s^2)	>0

Description

This command stops all synchronized axes in this axis's sync group. All axes remain synchronized together throughout the stop.

For the synchronized axes to which this command is issued, the current velocities and positions of those axes will be used to determine the deceleration rates such that none of those axes exceed the requested **Deceleration Rate**. Any other axes in the same group to which the Sync Stop (17) command was not issued will be ratioed according to their current ratios.

If this command is issued to an axis that is not synchronized, a command error will occur.

For more details on synchronizing axes, refer to the Sync Move Absolute (13) command.

Status Bits

In Position Bit

The Sync Stop (17) command will not turn on this bit.

Target Generator Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has stopped.

Target Generator State A and B bits

B	A	Description
0	0	Done

0	1	Reserved
1	0	Reserved
1	1	Decelerating to zero velocity

See Also

Sync Move Absolute (13) | Sync Move Relative (14)

8.4.5. Point-to-Point

8.4.5.1. Command: Move Absolute (20)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

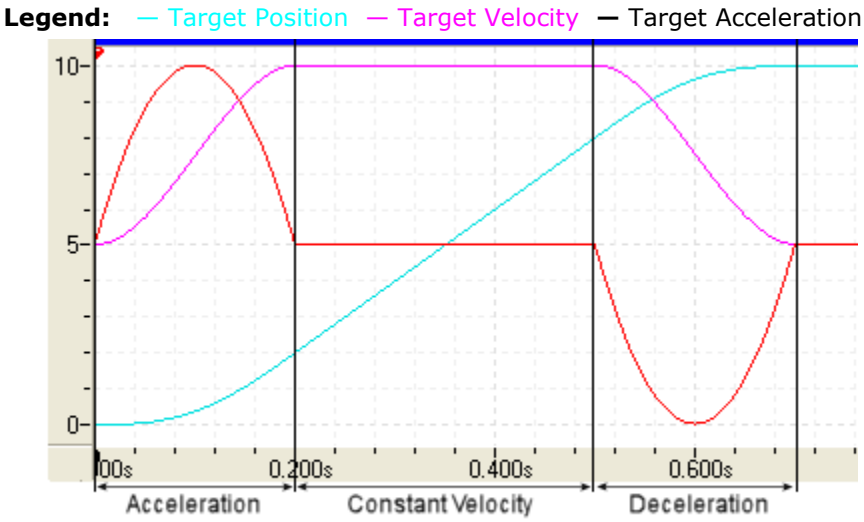
#	Parameter Description	Range
1	**Requested Position** (position-units)	any
2	**Requested Speed** (position-units/s)	≥0
3	**Acceleration Rate** (position-units/s^2)	>0
4	**Deceleration Rate** (position-units/s^2)	>0
5	**Direction** • Negative* (-1) • Nearest (0) • Positive* (1) • Current* (2) • Absolute* (3) * These options are intended for use with rotary axes. However, all options are available on linear axes, but have no effect. For more details, see the Using Rotary Motion topic.	a valid integer as described

Description

This command moves the axis in closed loop control from wherever the axis happens to be when the command is issued to the **Requested Position** using the **Requested Speed**, **Acceleration Rate**, and **Deceleration Rate**. The axis will stop at the **Requested Position** and hold that position in closed-loop control. If the axis is moving when this command is issued, it will start at that velocity and ramp to the requested velocity.

The **Direction** parameter is required only for rotary axes. For linear axes, it has no effect, and may be left at the default of Nearest (0).

The Move Absolute command generates a profile as shown in the plot below. The acceleration and deceleration are rounded, producing an "s-curve" velocity profile for smoother starts and stops.



Special Notes

Do Not Send Move Absolute Commands in Rapid Succession

The Target Acceleration is reset to zero at the beginning of each move. Therefore, if the Move Absolute command is sent in rapid succession such that it is received when the axis is accelerating or decelerating, the Target Acceleration in that part of the move will suddenly by reset to zero, causing jerky motion or overshoot, as described in the **S-Curves vs. Trapezoidal** section below.

It is possible to send the Move Absolute command in rapid succession if the Requested Jerk Axis Parameter is set to zero, or if the Target Type is set to Trapezoidal.

If your axis must follow a reference input, do not constantly issue this command to the axis to make it follow the position of the reference axis, unless you first set the Requested Jerk Axis Parameter is set to zero. Or, use gearing instead.

Issuing Another Move Command Before Reaching Position

If you issue a Move command and, before it is completed, issue another Move command to the same position and the new Deceleration Rate is equal to or smaller than the first one, the target profile may have to overshoot the requested position in order to not exceed the Deceleration rate. In cases where this may be a problem, it is better to use the Speed at Position command with a Requested Velocity of zero. Then the target profile will not overshoot the Requested position.

Moving in a Given Time

Sometimes, you may wish specify the amount of time the move must take. One option is to use the Time Move Absolute or Time Move Relative command. However, the maximum velocity cannot be specified with the Time Move command. You can use the equations below to determine Speeds and Accelerations of the Move Absolute or Move Relative command so that it will complete in a given amount of time:

Given

Distance = Distance to move (Command Position - current Target Position)

MoveTime = Desired total time for move

then,

Speed = 1.5 * Distance / MoveTime

Acceleration = Deceleration = (4.5 * Distance) / (MoveTime * MoveTime)

S-Curves vs. Trapezoidal

If the Requested Jerk Axis Parameter is non-zero, the axis will use s-curves. This is the default setting. If the Requested Jerk Axis Parameter is zero, then the axis will use a trapezoidal profile.

When using s-curves, if the move command is re-issued when the axis is decelerating, it may cause the axis to overshoot the requested position.

Starting Values When Command is Issued

When a closed-loop command is sent to an axis, the current target values are used as the starting point, such as Target Position and Target Velocity. Therefore, if the axis is in closed loop control when the command is received, it doesn't matter where the Actual Position and Actual Velocity are - the command will use the current Target Position and Target Velocity.

Note that in open loop control, there is no target and the Target Position and Target Velocity are set to be the same as the Actual Position and Actual Velocity.

Starting from Open Loop

If the axis is in open loop when this command is issued, it will use the Actual Position and Actual Velocity as the beginning Target Position and Target Velocity. Therefore, if the axis is stopped in open loop and the Actual Velocity is noisy (which is common), the initial Target Velocity may be large because the noise made it look large, even though it really isn't. If the requested acceleration is low, it can take a long time to change this initial Target Velocity to the requested velocity, which may cause the Target Position to move in the opposite direction before moving in the correct direction.

To avoid this problem, when starting from a stop in open loop, either use high acceleration and deceleration values, or issue a Hold Current Position (5) command before issuing the Move Absolute.

This also applies to other closed-loop position motion commands.

Status Bits

In Position Bit

When the Target Position reaches the **Requested Position** and the Actual Position is within the In Position Tolerance window, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates that the Target Position has reached the **Requested Position**. Notice that this bit does not indicate whether the Actual Position has reached the Requested Position.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Acceleration
1	0	Constant Velocity
1	1	Deceleration

See Also

Move Relative (21) | List of Commands | Commands Overview

8.4.5.2. Command: Move Relative (21)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Distance** (position-units)	any
2	**Requested Speed** (position-units/s)	≥0
3	**Acceleration Rate** (position-units/s^2)	>0
4	**Deceleration Rate** (position-units/s^2)	>0
5	**Relative to:** • Command Position (0) • Actual Position (1) • Target Position (2)	a valid integer as described

Description

This command moves the axis in closed loop control by the **Requested Distance**, using the **Requested Speed**, **Acceleration Rate**, and **Deceleration Rate**. The final Command Position is computed by adding the **Requested Distance** to the current value of the quantity specified in the **Relative To** parameter. The current value is the value at the time the RMC received the command.

Relative To Options:

- **Command Position (0)**
  The final Command Position is computed by adding the **Requested Distance** to the current Command Position.
- **Actual Position (1)**
  The final Command Position is computed by adding the **Requested Distance** to the current Actual Position.
- **Target Position (2)**
  The final Command Position is computed by adding the **Requested Distance** to the current Target Position.

This command is similar to the Move Absolute (20) command. However, this command does not have a Direction command parameter, and the Command Position is computed relatively instead of absolutely. See the Move Absolute (20) command for more details.

Status Bits

In Position Bit

When the Target Position has reached the Command Position and the Actual Position is within the In Position Tolerance window, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has reached the Command Position. Notice that this bit does not indicate whether the Actual Position has reached the Command Position.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Acceleration
1	0	Constant Velocity
1	1	Deceleration

See Also

Move Absolute (20) | List of Commands | Commands Overview

8.4.5.3. Command: Quick Move Absolute (15)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Position** (position-units)	any
2	**Requested Output** (RMC75/150: V) (RMC200: %)	RMC75/150: 0-10 V RMC200: 0-100 %
3	**Output Ramp Rate** (RMC75/150: V/sec) (RMC200: %/sec)	>0
4	**Deceleration Rate** (position-units/s^2)	>0
5	**Direction** • Negative* (-1) • Nearest (0) • Positive* (1) • Current* (2) • Absolute* (3) * These options are intended for use with rotary axes. However, all options are available on linear axes, but have no effect. For more details, see the Using Rotary Motion topic.	a valid integer as described

Description

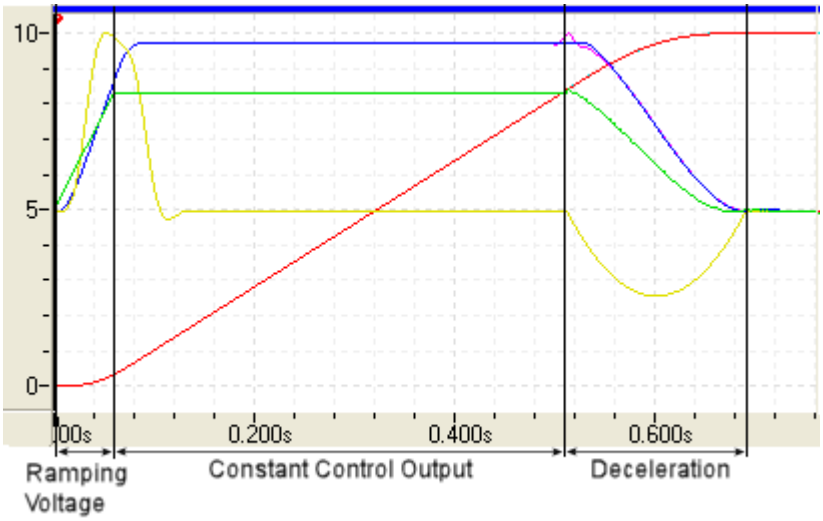
This command moves to the **Requested Position** with a combination of open loop and closed loop control. The first part of the move is made in open-loop control, then the axis switches to closed-loop control for the deceleration to the **Requested Position**.

This command is called "quick" because it makes it possible to give the maximum Control Output to move the system at its maximum speed. In closed loop control, reaching the maximum Control Output would cause an error.

When this command is issued, the axis ramps the Control Output in open loop from the current value to the **Requested Output** at the specified **Output Ramp Rate**. The Control Output stays at that value until the axis must begin decelerating in order to reach the **Requested Position** at the specified **Deceleration Rate**. Once the axis starts decelerating (which can happen while the Control Output is being ramped or while it is constant), the axis switches to closed loop, setting the Target Position and Velocity to the Actual Position and Velocity. The axis will hold the final position in closed loop control.

It is important that the closed-loop motion is tuned properly before using the Quick Move command! A Velocity Feed Forward that is incorrect may result in a discontinuity in the Control Output, causing the system to jerk.

The plot below shows a typical Quick Move. Notice how the Control Output ramps up linear, is flat, and when the axis switches to closed-loop control, the Control Output ramps down in a curved fashion.



Special Notes

Switching to closed-loop while Control Output is still ramping

The Quick Move command works best if the Control Output reaches its constant value before the axis switches to closed-loop control. If the switch from open-loop to closed-loop control occurs while the Control Output is ramping, the Actual Position may lag behind the Target Position momentarily. This is because the Target Acceleration is set to zero at this point, while the Actual Acceleration was likely non-zero. This discontinuity may cause some control issues.

Actual Velocity and/or Actual Acceleration filtering may be required

During the course of the Quick Move, the RMC calculates when it has to start decelerating in order to reach the requested position. At that point, it switches into closed loop control. When switching to closed loop control, the RMC uses the actual velocity and actual acceleration to determine the target profile to the requested position. If the velocity and/or acceleration feedback is noisy, the RMC may enter closed loop control to late, causing the Target Position to overshoot the requested position, or the RMC may calculate a very strange route to the requested position. If this occurs, you may need to filter the velocity.

For the RMC75 and RMC150, set the Actual Velocity Filter. For the RMC200, set the Velocity Display Filter.

Status Bits

In Position Bit

When the Target Position reaches the **Requested Position** and the Actual Position is within the In Position Tolerance window, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has reached the **Requested Position**. Notice that this bit does not indicate whether the Actual Position has reached the Command Position.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Ramping Control Output in Open Loop
1	0	Constant Control Output at Requested Output
1	1	Decelerating in Closed Loop

See Also

Quick Move Relative (16) | List of Commands | Commands Overview

8.4.5.4. Command: Quick Move Relative (16)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Distance** (position-units)	any
2	**Requested Output** (RMC75/150: V) (RMC200: %)	RMC75/150: 0-10 V RMC200: 0-100 %
3	**Output Ramp Rate** (V/sec)	>0
4	**Deceleration Rate** (position-units/s^2)	>0
5	**Relative To** • Command Position (0) • Actual Position (1) • Target Position (2)	a valid integer as described

Description

This command is identical to the Quick Move Absolute (15), except that the final Command Position is computed by adding the **Requested Distance** to the current value of the quantity

specified in the **Relative To** parameter. The current value is the value at the time the RMC received the command.

Relative To Options:

- **Command Position (0)**
  The final Command Position is computed by adding the **Requested Distance** to the current Command Position.
- **Actual Position (1)**
  The final Command Position is computed by adding the **Requested Distance** to the current Actual Position.
- **Target Position (2)**
  The final Command Position is computed by adding the **Requested Distance** to the current Target Position.

Status Bits

In Position Bit

When the Target Position reaches the Command Position and the Actual Position is within the In Position Tolerance window, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has reached the Command Position. Notice that this bit does not indicate whether the Actual Position has reached the Command Position.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Ramping Control Output in Open Loop
1	0	Constant Control Output at Requested Output
1	1	Decelerating in Closed Loop

See Also

Quick Move Absolute (15) | List of Commands | Commands Overview

8.4.5.5. Command: Time Move Absolute (23)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Position** (position-units)	any

2	**Time for Move** (sec)	≥0
3	**Direction** • Negative* (-1) • Nearest (0) • Positive* (1) • Current* (2) • Absolute* (3) * These options are intended for use with rotary axes. However, all options are available on linear axes, but have no effect. For more details, see the Using Rotary Motion topic.	a valid integer as described

Description

This commands moves the axis in closed loop control from the current position to the **Requested Position** in the **Time for Move** specified using a single fifth-order function. Acceleration and velocity are not limited. The axis will hold the final position in closed loop control.

If you need to complete a move in a given time, but need to limit the velocity and/or acceleration, see the Move Absolute (20) command topic.

Step Jumps

If the **Time for Move** parameter is set to zero, the Target Position will immediately jump to the requested position.

Re-sending While Moving

Re-sending this command while moving will cause the Target Acceleration to be set to zero, which will cause a velocity bobble. If this is undesired, use the Advanced Time Move Absolute (26) command, which will use the current Target Acceleration. Notice that the Final Velocity and Final Acceleration of the Advanced Time Move Absolute command must be set to zero to duplicate the Time Move Absolute behavior.

Status Bits

In Position Bit

When the Target Position reaches the Command Position and the Actual Position is within the In Position Tolerance window, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has reached the Command Position. Notice that this bit does not indicate whether the Actual Position has reached the Command Position.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Acceleration
1	0	Constant Velocity
1	1	Deceleration

See Also

Time Move Relative (24) | List of Commands | Commands Overview

8.4.5.6. Command: Time Move Relative (24)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Distance** (position-units)	any
2	**Time for Move** (sec)	≥0
3	**Relative To** • Command Position (0) • Actual Position (1) • Target Position (2)	a valid integer as described

Description

This command is identical to the Time Move Absolute (23), except that the final Command Position is computed by adding the **Requested Distance** to the current value of the quantity specified in the **Relative To** parameter. The current value is the value at the time the RMC received the command.

Relative To Options:

- **Command Position (0)**
  The final Command Position is computed by adding the **Requested Distance** to the current Command Position.
- **Actual Position (1)**
  The final Command Position is computed by adding the **Requested Distance** to the current Actual Position.
- **Target Position (2)**
  The final Command Position is computed by adding the **Requested Distance** to the current Target Position.

Steps Jumps

If the **Time for Move** parameter is set to zero, the Target Position will immediately to jump to the calculated Command Position.

Re-sending While Moving

Re-sending this command while moving will cause the Target Acceleration to be set to zero, which will cause a velocity bobble. If this is undesired, use the Advanced Time Move Relative (27) command, which will use the current Target Acceleration. Notice that the Final Velocity and Final Acceleration of the Advanced Time Move Relative command must be set to zero to duplicate the Time Move Relative behavior.

Status Bits

In Position Bit

When the Target Position reaches the Command Position and the Actual Position is within the In Position Tolerance window, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has reached the Command Position. Notice that this bit does not indicate whether the Actual Position has reached the Command Position.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Acceleration
1	0	Constant Velocity
1	1	Deceleration

See Also

Time Move Absolute (23) | List of Commands | Commands Overview

8.4.5.7. Command: Advanced Time Move Absolute (26)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD
Firmware Limitations:	RMC75/150: 1.45 or newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Position** (position-units)	any
2	**Final Velocity** (position-units/s)	any
3	**Final Acceleration** (position-units/s^2)	any
4	**Time for Move** (sec)	>0
5	**Direction** • Negative* (-1) • Nearest (0) • Positive* (1) • Current* (2) • Absolute* (3)	a valid integer as described

	* These options are intended for use with rotary axes. However, all options are available on linear axes, but have no effect. For more details, see the Using Rotary Motion topic.	

Description

This command calculates a target using a 5th-order polynomial. The closed-loop move will end with the requested conditions: position, velocity, and acceleration. The **Requested Position** is reached in the **Time for Move**. If the final velocity or acceleration is non-zero, this command should be followed by another motion command.

Once the **Requested Position** is reached, the Done status bit will be set. If either the **Final Velocity** or **Final Acceleration** are non-zero, the polynomial will continue for one loop time after the Done Status bit is set, and if a new command is not issued during this time, then a Closed Loop Halt will occur and the Runtime Error will be set (notice that if the Runtime Error AutoStop is set to a different type of halt, that halt will also occur). If the **Final Velocity** and the **Final Acceleration** are both zero, then the axis will hold position and no errors will be set.

This command is intended to be followed by another command, especially if either the **Final Velocity** or **Final Acceleration** are non-zero. Typically, this command is used only in a user program, where it is very easy to issue another command immediately when the Done bit is set. Use the Wait For or Conditional Jump Link Types to check for the Done bit.

This command creates a profile based on time. To make a profile based on a master position, see the Advanced Gear Move (33).

Why Bother?

This command is useful if you want to specify what the velocity and acceleration at a position and time should be. This command is typically used for only a segment of a move, and you should always issue another command after this one when the Done Status bit is set. This can easily be done in the User Programs. You can issue several of these moves in a row to get the motion profile you want, especially if you need to specify the velocity and acceleration at certain points. The Expression (113) command can be used to calculate the position, velocity, and acceleration values to be used in this command. Complex motion profiles may require many individual segments using the Advanced Time Move commands.

Using the Advanced Time Moves in User Programs

Indexing Arrays

The Advanced Time Move Absolute and Advanced Time Move Relative are intended for use in user programs. Typically, these commands are programmed in a loop and cycle through an array or arrays of positions, speeds, and/or accelerations. To do this, use the variable table to set up arrays. A variable for incrementing the array is also required. It must be of data type DINT.

Timing Considerations

When looping through long arrays with the Advanced Time Move commands, precise timing is often important. Each task can execute a maximum of one step per loop time. However, the commands in a step and the link condition in that same step are not executed in the same loop time.

When a step is executed, the commands in that step are issued in that loop time. Then, in the *next* loop time of the controller, the Link Type is evaluated. When the Link Type condition becomes true or tells the program to jump, the program will jump to the specified step and issue the commands in that step *in the same loop time* that the condition became true.

Once the **Requested Position** is reached, the Done status bit will be set. If either the **Final Velocity** or **Final Acceleration** are non-zero, the polynomial will continue for one loop time after the Done Status bit is set, and if a new command is not issued during this time, then a Closed Loop Halt will occur and the Runtime Error will be set (notice that if the Runtime Error AutoStop is set to a different type of halt, that halt will also occur). Make sure to create your program such that the next Advanced Time Move command will be issued within one loop time after the Done bit is set.

Motion Equations

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has reached the **Requested Position**.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Acceleration
1	0	Constant Velocity
1	1	Deceleration

See Also

Advanced Time Move Relative (27) | List of Commands | Commands Overview

8.4.5.8. Command: Advanced Time Move Relative (27)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD
Firmware Limitations:	RMC7/150: 1.45 or newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Distance** (position-units)	any
2	**Final Velocity** (position-units/s)	any
3	**Final Acceleration** (position-units/s^2)	any
4	**Time for Move** (sec)	>0
5	**Relative To** • Command Position (0) • Actual Position (1) • Target Position (2)	a valid integer as described

Description

This command is identical to the Advanced Time Move Absolute (26), except that the final Command Position is computed by adding the **Requested Distance** to the current value of the quantity specified in the **Relative To** parameter. The current value is the value at the time the RMC received the command.

Relative To Options:

- **Command Position (0)**
  The final Command Position is computed by adding the **Requested Distance** to the current Command Position.
- **Actual Position (1)**
  The final Command Position is computed by adding the **Requested Distance** to the current Actual Position.
- **Target Position (2)**
  The final Command Position is computed by adding the **Requested Distance** to the current Target Position.

See the Advanced Time Move Absolute (26) topic for more details.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has travelled the **Requested Distance**.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Acceleration
1	0	Constant Velocity
1	1	Deceleration

See Also

Advanced Time Move Absolute (26) | List of Commands | Commands Overview

8.4.5.9. Command: Move Absolute (I-PD) (28)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position I-PD
Firmware Limitations:	RMC75/150: 1.50 or newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Position** (position-units)	any
2	**Maximum Speed** (position-units/s)	≥0
3	**Direction** • Negative* (-1) • Nearest (0) • Positive* (1)	a valid integer as described

	• Current* (2) • Absolute* (3) * These options are intended for use with rotary axes. However, all options are available on linear axes, but have no effect. For more details, see the Using Rotary Motion topic.	

Description

This command is an advanced command. Do not use it unless you specifically intend to use the Position I-PD control mode. If you want to make a basic motion move, use the Move Absolute (20) command instead.

This command moves the axis in closed loop control from wherever the axis is when the command is issued to the **Requested Position** using the specified **Maximum Speed**. The axis will automatically be switched to the Position I-PD control mode during the motion, and will remain in the Position I-PD control mode until another command is issued to the axis. However, if a closed-loop halt occurs on the axis during the motion, the axis will remain in the Position I-PD control mode.

The **Direction** parameter is required only for rotary axes. For linear axes, it must be 0.

See Also

Move Relative (I-PD) (29) | Position I-PD | List of Commands | Commands Overview

8.4.5.10. Command: Move Relative (I-PD) (29)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position I-PD
Firmware Limitations:	RMC75/150: 1.50 or newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Distance** (position-units)	any
2	**Maximum Speed** (position-units/s)	≥0
3	**Relative to:** • Command Position (0) • Actual Position (1) • Target Position (2)	a valid integer as described

Description

This command is an advanced command. Do not use it unless you specifically intend to use the Position I-PD control mode. If you want to make a basic relative motion move, use the Move Relative (21) command instead.

This command is identical to the Move Absolute (I-PD) (28), except that the final Command Position is computed by adding the **Requested Distance** to the current value of the quantity

specified in the **Relative To** parameter. The current value is the value at the time the RMC received the command.

Relative To Options:

- **Command Position (0)**
  The final Command Position is computed by adding the **Requested Distance** to the current Command Position.
- **Actual Position (1)**
  The final Command Position is computed by adding the **Requested Distance** to the current Actual Position.
- **Target Position (2)**
  The final Command Position is computed by adding the **Requested Distance** to the current Target Position.

See the Move Absolute (I-PD) (28) topic for more details.

See Also

Move Absolute (I-PD) (28) | List of Commands | Commands Overview

8.4.6. Gearing

8.4.6.1. Command: Gear Absolute (25)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

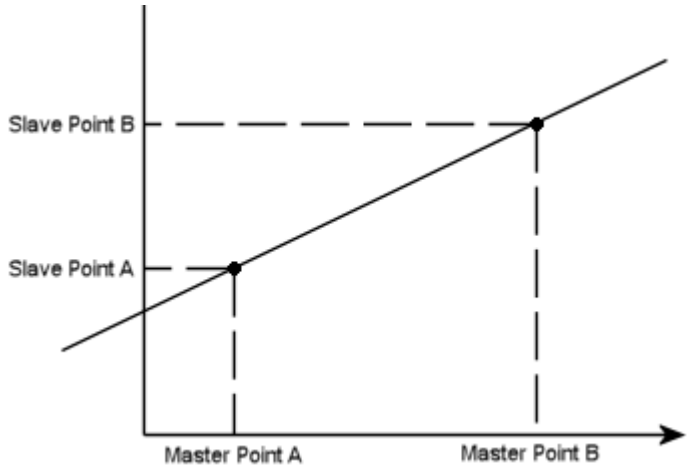
#	Parameter Description	Range
1	**Master Register** **Note:** See **Specifying a Register Address** below.	Valid RMC register
2	**Master Point A** (pu)	Any REAL number
3	**Master Point B** (pu)	Any REAL number
4	**Slave Point A** (pu)	Any REAL number
5	**Slave Point B** (pu)	Any REAL number
6	**Endpoint Behavior** • Fault (0) • Truncate (1) • Extrapolate (2)	A valid integer as described.

Description

This command sets up an absolute linear gearing relationship between the master register and the position target for the axis this command was issued to (the slave axis) and will make the slave axis follow that relationship. Typically, the master register is the Target or Actual Position of another axis. This command is very useful for making an axis follow a reference input (half-axis).

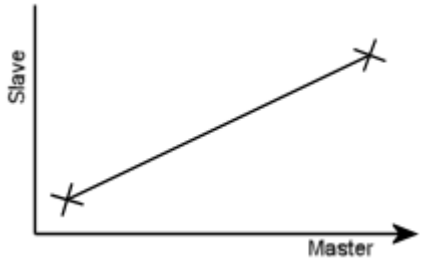
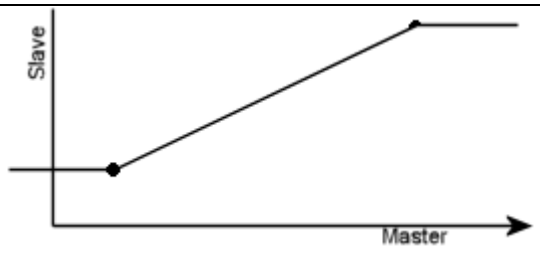
When this command is issued, the slave axis must either already be at the correct point specified by the relationship based on the current value of the master, or a Transition command must previously have been issued to specify how the axis should move to get to the line. Notice that this means most applications will need to use the Transition command!

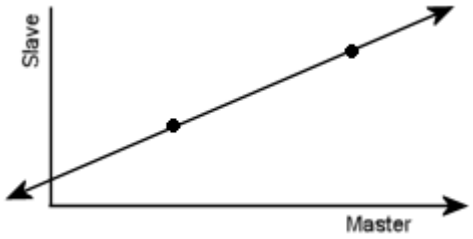
Master Point A, **Master Point B**, **Slave Point A**, and **Slave Point B** specify the linear relationship. The behavior beyond these points depends on the **Endpoint Behavior** parameter.



EndPoint Behavior

The Endpoint Behavior specifies what happens to the slave axis if the master moves outside of the range specified by Master Point A and Master Point B:

Option	Description	Image
Fault	If the master exceeds either endpoint, the slave axis' runtime error bit will be set which will halt the axis if the Auto Stops are configured to do so.	
Truncate	If the master moves past an endpoint, the slave axis' Target Position will stop at the endpoint. When the master moves back into the range, the gearing will resume. If the master is moving quickly when it exceeds the endpoint, it may cause the slave to stop abruptly. **Note:** If a superimposed transition is used, in	

	certain cases it can cause the slave to exceed the endpoints during the transition. If this causes problems, consider using a different type of transition.	
Extrapolate	The slave will always follow the master on the linear relationship.	

Related Commands

With the Gear Absolute command, the Target Position will follow the master value exactly. If the master is noisy, or exceeds the position, velocity, or acceleration limits of the slave axis, this can cause problems. In this case, consider using the Track Position (57) or Track Position (I-PD) (58) commands, which provide limits on the slave axis motion.

See the Gearing Overview topic for general information about gearing, including possible Gear Masters.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Target Generator State Bits

The Target Generator Done, State A and State B bits are all off during the gearing.

Pri. TG SI Busy (Primary Target Generator Superimposed Busy) Bit

This bit will be set during the transition. The transition begins when the motion command is issued, not necessarily when the Transition command is issued. When the transition completes, this bit will clear. At this point, the slave axis will be on the mapped relationship.

See Also

Gearing Overview | List of Commands | Commands Overview

8.4.6.2. Command: Gear Position (Clutch by Time) (30)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Numerator**	Any REAL number
2	**Denominator**	Any REAL number, not 0
3	**Master Register** **Note:** Specifying a Register Address	Valid RMC register
4	**Clutch Time (sec)**	>=0

Description

This command gears the axis to the **Master Register**, using the value in this register as the master position. Typically the master register is the Target or Actual Position of another axis.

When this command is issued, the RMC uses the current gear ratio or determines the current gear ratio by comparing the velocities of the master and slave. The gear ratio is then ramped from its current effective ratio to the requested ratio, in the time specified by the Clutch Time parameter. The ratio is determined by the **Numerator** and **Denominator**.

If the master register is not a Target or Actual Position, the Gear Position (Clutch by Time) (30) command will assume an initial gear ratio of 0:1. This may cause a jerk in the motion of the geared axis if it is already moving.

If the master is moving very slowly or nearly stopped when this command is issued, the initial gear ratio may be very large. There are two ways to avoid this problem:

- Use a different gearing command, such the Gear Pos (Clutch by Rate) (39) command.
- First issue this command with a 0:1 ratio and a time of zero, then immediately issue this command again with the desired gear ratio and clutch time.

If both axes are stopped or moving very slowly, this command can be issued with a **Clutch Time** of zero. If either of the axes are moving, a **Clutch Time** of zero may cause a sudden jerk.

See the Gearing Overview topic for general information about gearing, including Gear Ratio, Clutching and possible Gear Masters.

Gear Position (Clutch by Time) Example

In this example, Axis 1 (slave) gears to Axis 0 (master) at a 1:1 ratio. Both axes start at 0 pu. The master starts moving at time 0. At 0.2 seconds, the following Gear Pos (Clutch by Time) command is issued to the slave.

Numerator = 1

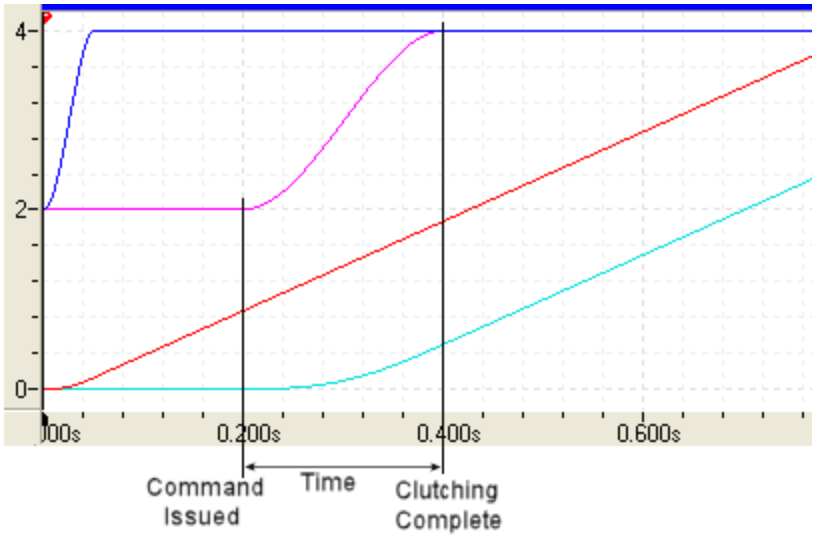
Denominator = 1

Master Register = _Axis[0].TarPos

Clutch Time = 0.2 sec

The plot below shows how the slave moves.

Legend: — Master Position — Master Velocity
— Slave Position — Slave Velocity



At 0.2 seconds, when the command was issued to the slave, it began ramping up the velocity such that slave reached the specified gear ratio at 0.4 seconds.

Applications Not Suitable for Clutch by Time

The Gear Position command clutches based on time, which means it will reach the requested gear ratio in the time specified. This command is not well-suited for the following types of applications:

- **If Position at which Axis Reaches Gear Ratio is Important**
  The Gear Pos (Clutch by Time) command does not directly specify the position at which the gear ratio will be reached. If the axis must reach the requested gear ratio at a certain position, such as in flying cut-off applications, use the Gear Pos (Clutch by Distance) (32) command instead.
- **If Maximum Velocity or Acceleration Must be Specified**
  The Gear Pos (Clutch by Time) command does not limit the velocity or acceleration. If you need to limit the velocity or acceleration, use the Gear Pos (Clutch by Rate) (39) instead.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the motion the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

Indicates the clutching is complete and the gear ratio is now locked, which occurs when the Time expires.

Target Generator State A and B bits

B	A	Description
0	0	Locked at a zero (0) ratio

0	1	Increasing the ratio (away from zero)
1	0	Locked at a non-zero ratio
1	1	Decreasing the ratio (toward zero)

See Also

Gearing Overview | List of Commands | Commands Overview

8.4.6.3. Command: Gear Velocity (31)

Supported Axes:	Position or Velocity Control Axes
Supported Control Modes:	Position PID, Position I-PD, Velocity PID, Velocity I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

[TODO] - add plot

Command Parameters

#	Parameter Description	Range
1	**Numerator**	Any REAL number
2	**Denominator**	Any REAL number, not 0
3	**Master Register** **Note: See Specifying a Register Address below.**	Valid RMC register
4	**Clutch Time (sec)**	Any REAL number >0

Description

This command gears the velocity of the axis to the Master Register, using the value in this register as the master velocity. Typically the master register is the Target or Actual Velocity of another axis.

When this command is issued, the RMC uses the current gear ratio or determines the current gear ratio by comparing the master to the velocities of the slave. The gear ratio is then ramped from its current effective ratio to the requested ratio in the time specified by the **Clutch Time** parameter. The gear ratio is determined by the **Numerator** and **Denominator**.

This command is particularly useful when gearing an axis to a velocity reference axis. Because a velocity reference axis does not have a position, the Gear Pos (Clutch by Time) (30) position command will not work, but the Gear Velocity command will. To gear a *position* to a master, use the Gear Pos (Clutch by Time) (30) command.

See the Gearing Overview topic for general information about gearing, including Gear Ratio, Clutching and possible Gear Masters.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

Indicates the clutching is complete and the gear ratio is now locked, which occurs when the Time expires.

Target Generator State A and B bits

B	A	Description
0	0	Locked at a zero (0) ratio
0	1	Increasing the ratio (away from zero)
1	0	Locked at a non-zero ratio
1	1	Decreasing the ratio (toward zero)

See Also

Gearing Overview | List of Commands | Commands Overview

8.4.6.4. Command: Gear Position (Clutch by Rate) (39)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Numerator**	Any REAL number
2	**Denominator**	Any REAL number
3	**Master Register** **Note:** See **Specifying a Register Address** below.	Valid RMC register
4	**Acceleration Rate** (position-units/s^2)	Any REAL number
5	**Jerk Rate** (position-units/s^3)	Any REAL number

Description

This command electronically gears the axis to the requested register, using this register as the master position. Typically, the master register is the Target or Actual Position of another axis. The slave ramps its target velocity using the **Acceleration Rate** and **Jerk Rate** (rate of change of acceleration) parameters until it reaches the synchronized gear ratio. The gear ratio is determined by the **Numerator** and **Denominator**.

This command is more robust than Gear Pos (Clutch by Time) (30) in that it avoids slave velocity discontinuities if the initial master velocity is low.

See the Gearing Overview topic for general information about gearing, including Gear Ratio, Clutching and possible Gear Masters.

Gear Position (Clutch by Rate) Example

In this example, Axis 1 (slave) gears to Axis 0 (master) at a 1:1 ratio. The master starts moving at time 0. When the master reaches 1 pu, the following Gear Pos (Clutch by Time) command is issued to the slave.

Numerator = 1

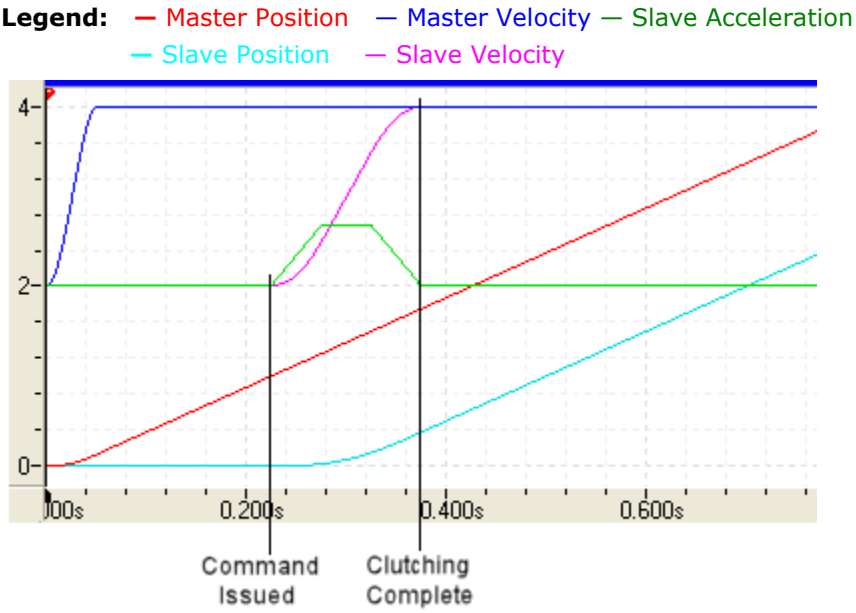
Denominator = 1

Master Register = _Axis[0].TarPos

Accel Rate = 50

Jerk Rate = 1000

The plot below shows how the slave moves.



When the command was issued to the slave, it began ramping up the velocity at the specified Acceleration and Jerk. As the slave approaches the master, the velocity is ramped down. The clutching is complete when the slave reaches the specified gear ratio.

Applications Not Suitable for Clutch by Rate:

The Gear Position (Clutch by Rate) command clutches based on a rate, which means it ramps its target velocity using the acceleration and jerk parameters until it reaches the synchronized gear ratio. This differs from clutching by distance, in which the axis would reach the requested gear ratio in a certain distance of travel.

Clutching by rate is not the best method of gearing for applications that require axes to be geared at a specified ratio by a certain distance, such as flying cut-offs, etc. Use the Gear Pos (Clutch by Distance) (32) command instead. If the axis must be geared within a certain amount of time, use the Gear Pos (Clutch by Time) (30) command.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

Indicates the clutching is complete and gear ratio is locked, which occurs when the Slave Velocity has reached the ratioed Master Velocity.

Target Generator State A and B bits

B	A	Description
0	0	Locked at a zero(0) ratio
0	1	Accelerating (away from zero velocity)
1	0	Locked at a non-zero ratio
1	1	Decelerating (toward zero velocity)

See Also

Gearing Overview | List of Commands | Commands Overview

8.4.6.5. Command: Gear Position (Clutch by Distance) (32)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Final Gear Ratio**	Any REAL number
2	**Master Register** **Note:** See **Specifying a Register Address** below.	Valid RMC register
3	**Master Sync Position** (position-units)	Any REAL number
4	**Slave Sync Position** (position-units)	Any REAL number
5	**Master Start Distance** (position-units)	Any REAL number
6	**Master Direction** If the master axis linear, this should be **Nearest (0)**. The other options will have no effect. If the master axis is rotary, see the **Rotary Motion** section below.	a valid integer as described

7	**Slave Direction** If the slave axis linear, this must be **Nearest (0)**. The other options will have no effect. If the slave axis is rotary, see the **Rotary Motion** section below.	a valid integer as described

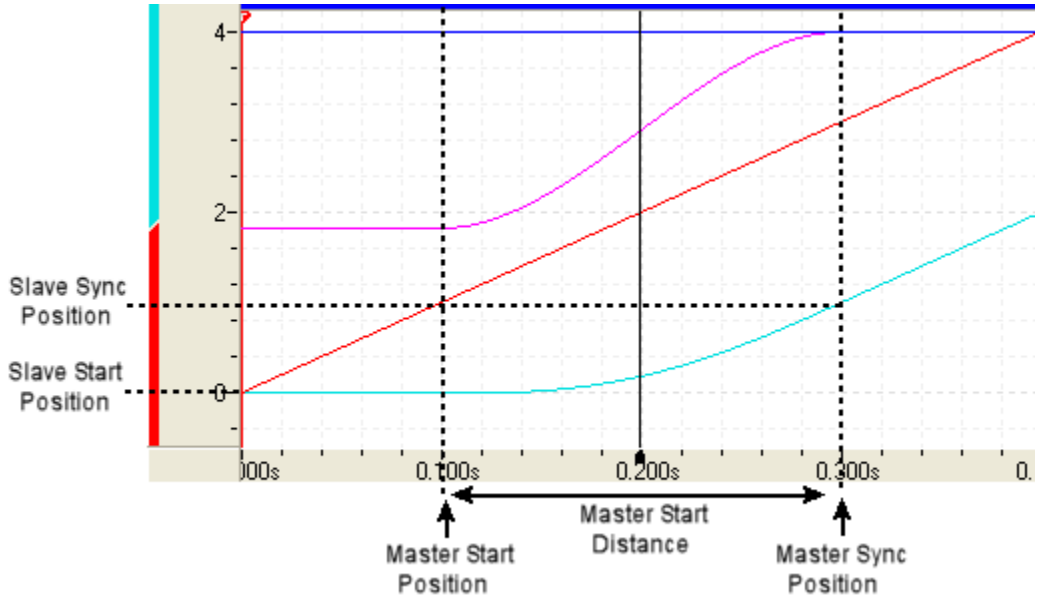
Description

This command electronically gears the axis to the requested register such that the two axes synchronize exactly at the requested **Master Sync Position** and **Slave Sync Position** and then remain geared at the specified **Final Gear Ratio**. Typically, the master register is the Target or Actual Position of another axis. This command is intended for use in flying-cutoff and flying-shear type applications. This command will work even if the speed of the master is changing.

See the Gearing Overview topic for general information about gearing, including Gear Ratio, Clutching and possible Gear Masters.

Definitions

The plot below is typical of a Gear Pos (Clutch by Distance) command. The table below uses the plot to define some important concepts used in the remaining discussion.



The master positions are indicated on the x-axis for clarity, although the positions are actually on the y-axis.

Term	Description
Master Sync Position	The position of the master at which the slave will be at its sync position and will be at locked in at the final gear ratio.
Master Start Distance	The distance from the **Master Sync Position** during which the clutching will take place. This area is also called the "clutch area".

	Note: This is the *distance*, not the absolute position!
Master Start Position	The master position at which the slave starts clutching. This position is **Master Sync Position - Master Start Distance.**
Slave Sync Position	The position of the slave at which the master will be at its sync position and the slave will be at locked in at the final gear ratio.
Slave Start Position	The slave position at the time the command is issued.
Slave Distance	The distance between the Slave Start Position and the Slave Sync Position.

Clutching

This command requires that the slave is either stopped (which is a zero gear ratio) or geared when the command is issued. This gear ratio will be used to create an initial constant gear ratio that holds until the master reaches the Master Start Position.

When the master moves beyond the Master Start Position, the gear ratio will ramped from the initial ratio to the requested ratio such that the two synchronize exactly at the requested **Master Sync Position** and **Slave Sync Position**. The **Master Start Distance** specifies how far from the **Master Sync Position** the clutching should start.

The **Master Start Distance** implies the direction with its sign. For example, with a **Master Sync Position** of 10, a **Master Start Distance** of 2 means that the clutching happens as the master moves from 8 up to 10. With a **Master Sync Position** of 10, a **Master Start Distance** of -2 means that the clutching happens from 12 down to 10.

Notice that the RMC will decrease the **Master Start Distance** in some cases. See the **Master Start Distance Details** section below for more details.

Reversing Direction

This command allows reversing the direction of the master. If the master reverses directions before the axes have reached the sync positions, the slave will follow the profile in reverse. If the master continues in reverse past the Master Start Distance, the slave will continue following the master in reverse using the initial gear ratio. The clutching will resume if the master returns to the **Master Start Position**.

Once the axes have reached the sync positions, the slave will lock in at the final gear ratio, and will gear to the master at that ratio whether the master goes forward or backward. It will not revert to the profile it followed before reaching the sync position.

Errors

If the axis is not stopped or already geared when this command is issued, a Command Error will be generated.

If the Master Position is already beyond the Master Sync Position when the command is issued (where "beyond" is defined by the direction implied by the Master Start Distance as described above), then the command will fail with a Command Error.

If the Master Position is already in the clutch area (between Master Start Position and Master Sync Position) when the command is issued, then the command will proceed but with a Command Modified error, since the effective Master Start Distance will be adjusted to make the move complete. Setting the Command Modified Auto Stop to Status Only will allow this move to proceed.

Uses for Gear Pos (Clutch by Distance) Command

The Gear Pos (Clutch by Distance) command is useful for the following purposes:

- **Flying-Cutoff**
  This command is designed for flying-cutoff or flying-shear type applications. See the example below for more details.
- **"Superimposed" Move**
  If an axis is already geared to a master, this command can be used to make a "superimposed"

move on the slave. To do this, issue this command with the same final gear ratio that the axis was already geared with. See the **Gear Ratio Details** section below for caveats. Notice that the RMC has other gearing commands intended for this purpose that may work better, such as Geared Slave Offset (35) and Phasing (34).

- **Stop Based on Master Position**
  If an axis is already geared, this command can be used to stop an axis based on the position of the master. Use a final gear ratio of zero. This guarantees that the slave will stop at the right position relative to the master.
- **Point-to-Point Move Based on Master Position**
  If an axis is stopped, this command can be used to do a point-to-point move on the axis based on the position of the master. Use a final gear ratio of zero. The axis will start moving as described in the **Master Start Distance Details Below** and will stop at the specified sync positions.

Flying-Cutoff Example

Consider this flying-cutoff example. A belt is moving a continuous pipe, which is to be cut to length while the belt is moving. The saw is mounted on a carriage that can speed up to the speed of the belt, and then it can make the cut. After cutting, the carriage returns to the home position.

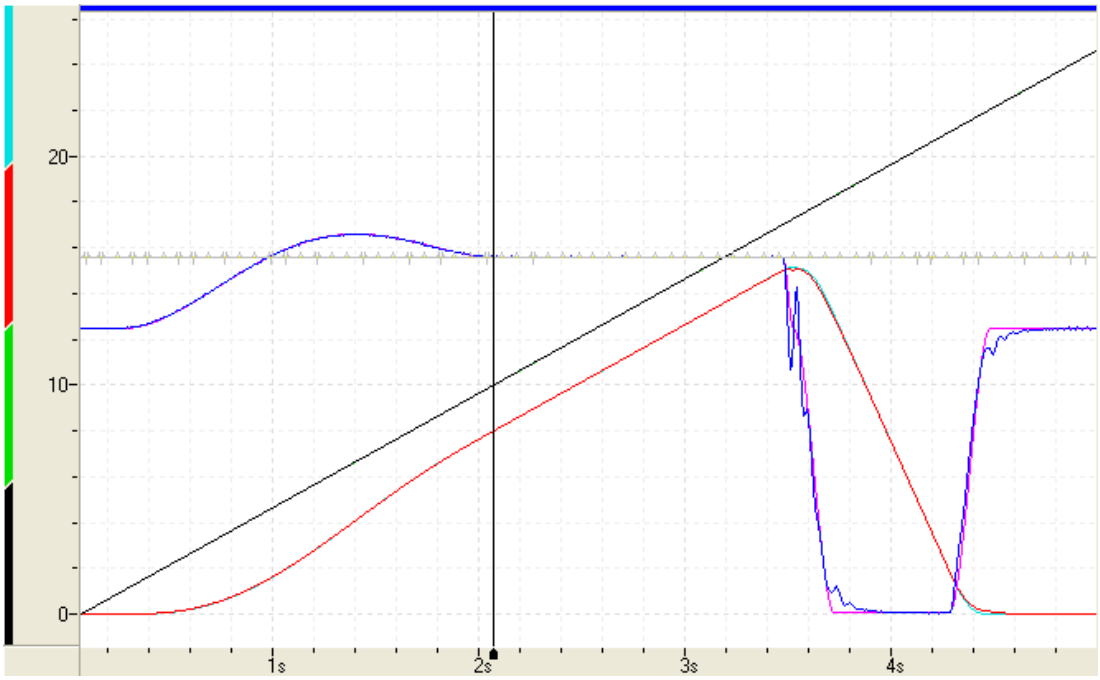
The belt (the master) is controlled by Axis 1 with a rotary encoder, moving at 5 inches/second. The carriage (the slave) is controlled by Axis 0. The carriage will start at and return to a home position of zero(0). The cut should take place when the master is at 12 in. and the slave is at 10 in and locked at a 1:1 ratio. The slave must be synchronized at 8 inches (2 inches before the cut). Therefore, the Slave Sync Position will be 8 and the Master Sync Position will be 10.

The Master Start Distance should be set to allow the slave time to get to speed. Because we will issue the command after the master crosses zero, we will set the Master Start Position greater than zero (0). We will use a Master Start Distance 9, resulting in a Master Start Position of 1.

The parameters of the Gear Position (Clutch by Distance) command are as listed below:

- **Final Gear Ratio = 1.0**
- **Master Register = F9:53 (Axis 1 Target Position)**
- **Master Sync Position = 10 inches**
- **Slave Sync Position = 8 inches**
- **Master Start Distance = 9 inches**

The plot of this motion is shown below. This plot also includes the motion of the carriage moving back (beginning at approximately 3.5 sec) after it has finished cutting.



The plot shows Axis 1 (the master) moving at a constant 5 in/sec. The carriage (Axis 0) is stopped, then accelerates to catch up to the master. Axis 0 and Axis 1 Target Positions are synchronized at 8 and 10, as indicated by the legend to the right. The legend indicates the values at the cursor in the plot.

In the plot, the axes remain synchronized (while the system is cutting) until approximately 3.5 seconds, at which point a command was issued to move the carriage back to its home position. This return move can be done with a Move Absolute, or even the Gear Pos (Clutch by Distance) command.

Plot Legend

Quantity	Value
Time (sec)	2.072
Axis0 Target Position (pu)	8.000
Axis0 Actual Position (pu)	7.999
Axis1 Target Position (pu)	10.000
Axis1 Actual Position (pu)	10.001
Axis0 Target Velocity (pu/s)	5.000
Axis0 Actual Velocity (pu/s)	5.002
Axis1 Target Velocity (pu/s)	5.000
Axis1 Actual Velocity (pu/s)	4.959

Master Start Distance Details

The **Master Start Distance** specifies how far from the **Master Sync Position** the clutching should start. This position is called the **Master Start Position**. When the master reaches this position, the slave axis begins moving. This section describes how the **Master Start Distance** affects the motion and cases in which the RMC modifies the Master Start Distance.

If the Slave is Initially Stopped or Geared at a Zero Ratio

The distance from the **Slave Start Position** (slave position at the time the command is issued) to the **Slave Sync Position** is called the **Slave Distance**. If the slave is stopped or geared at a ratio of zero when this command is issued, and the **Master Start Distance** is greater than (2.5 x Slave Distance / Final Gear Ratio), then the **Master Start Distance** will be reduced to (2.5 x Slave Distance / Final Ratio). This behavior is due to the mathematical formula used to calculate the profile. Without this behavior, the profile might move in the wrong direction first. Notice that after the command is issued, the slave will remain stopped or geared at 0:1 until the master reaches the modified **Master Start Position**.

Example

Consider a Gear Pos (Clutch by Distance) command issued with the parameters listed below. The command is issued when the slave axis is stopped at 0. Therefore, the **Slave Start Position** is 0.

Ratio = 1

Master Sync Pos = 5

Slave Sync Pos = 1

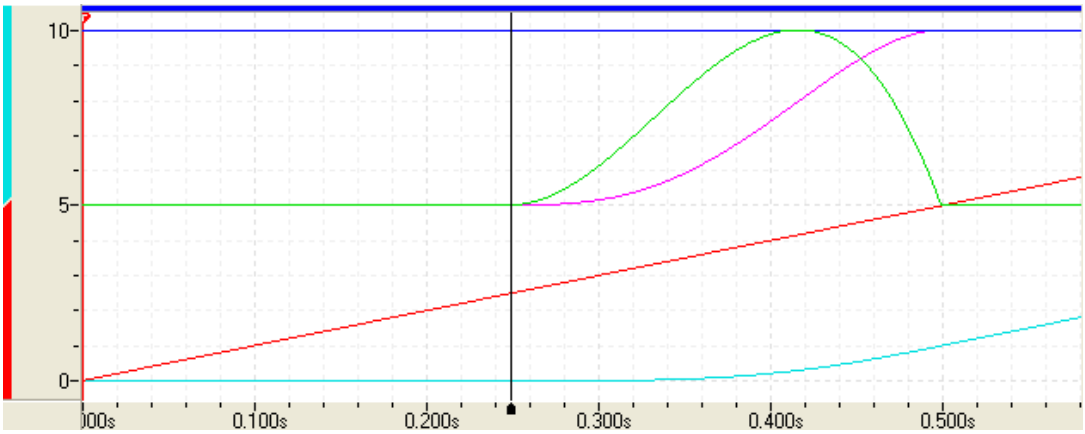
Master Start Distance = 4

The **Slave Distance** = Slave Sync Position - Slave Start Position = 1 - 0 = 1.

Also, (2.5 x Slave Distance / Final Gear Ratio) = (2.5 x 1 / 1) = 2.5.

Therefore, the specified **Master Start Distance** of 4 is greater than 2.5 as calculated above and will be limited to 2.5.

The plot of the motion looks like this:



The legend to the right indicates the values at the cursor in the plot. The Gear Pos (Clutch by Distance) command was issued at the beginning of the plot. Notice that the slave begins moving when the master is at 2.5 (the Master Sync Pos - Master Start Distance = 5 - 2.5 = 2.5). Before that point, the slave continues at the initial ratio, which was 0:1.

Plot Legend

Quantity	Value
Time (sec)	0.249
Axis0 Target Position (pu)	0.000
Axis1 Target Position (pu)	2.500
Axis0 Target Velocity (pu/s)	-0.000
Axis1 Target Velocity (pu/s)	10.000
Axis0 Target Accel (pu/s²)	0.0

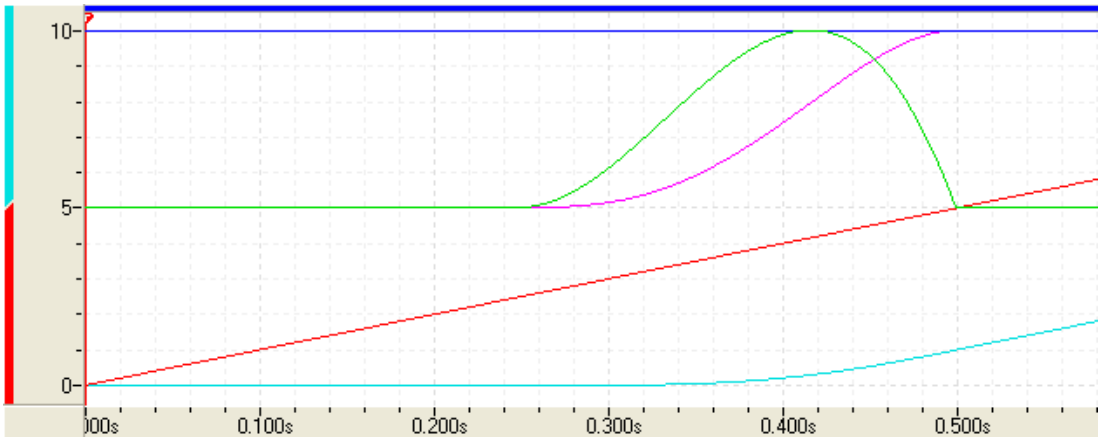
Important Master Start Distance Values

As discussed above, if the slave is initially stopped or geared at a ratio of zero, the **Master Start Distance** will be limited to (2.5 x Slave Distance / Final Gear Ratio). However, other values are also important. The ratio between the master and slave distances divides the behavior into four classes:

- ≥ 2.5 x Slave Distance / Final Gear Ratio
- = 2.0 x Slave Distance / Final Gear Ratio
- = 1.6666 x Slave Distance / Final Gear Ratio
- < 1.6666 x Slave Distance / Final Gear Ratio

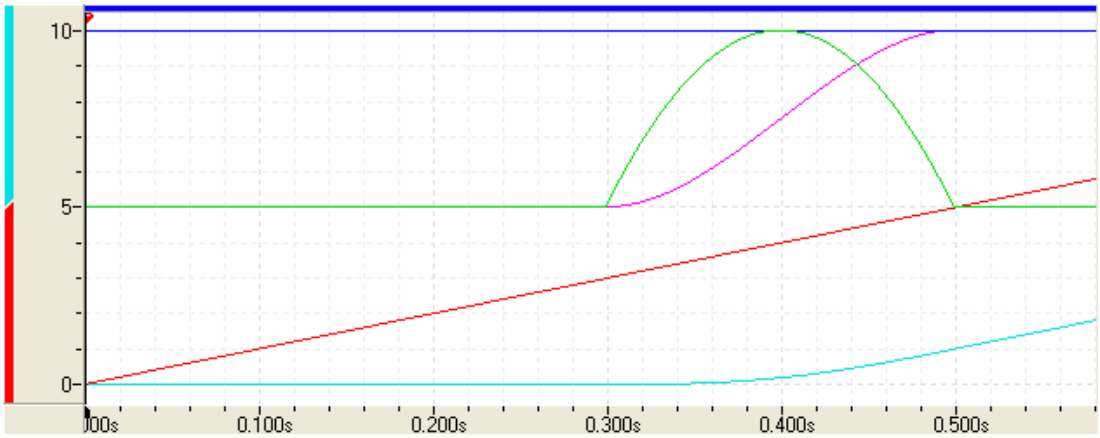
In the plots below, observe the slave velocity (magenta) and the slave acceleration (green) for each class listed above.

Master Start Distance ≥ 2.5 x Slave Distance / Final Gear Ratio



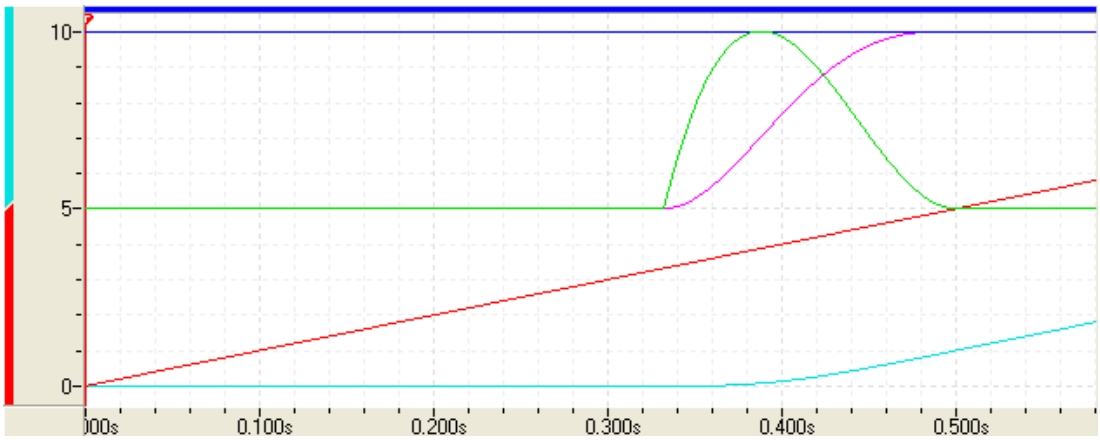
- Notice the smooth initial acceleration, but relatively high rate of acceleration at the sync position. This may cause minor difficulty tracking at the sync point.

Master Start Distance = 2.0 x Slave Distance / Final Gear Ratio



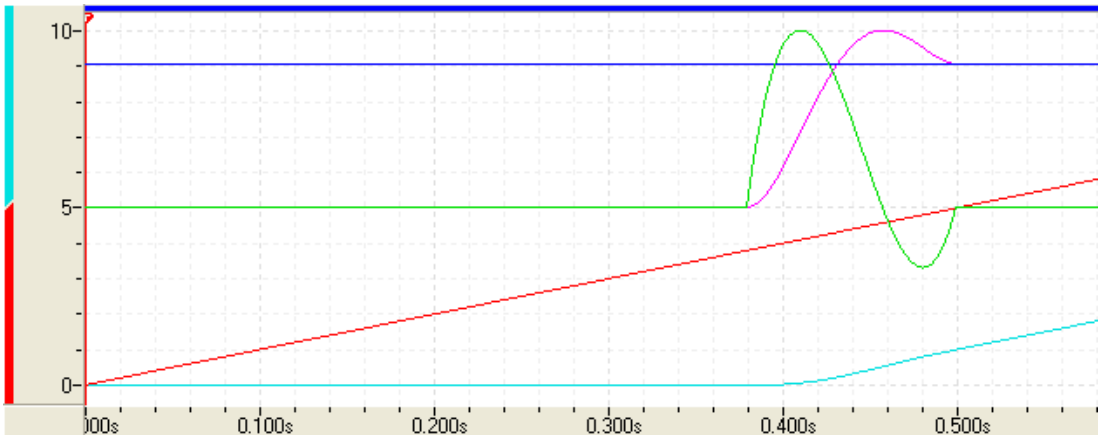
- Notice the symmetrical acceleration. There is still a relatively high rate of acceleration at the sync position, which may cause minor difficulty tracking at the sync point.

Master Start Distance = 1.6666 x Slave Distance / Final Gear Ratio



- Notice the sharp initial acceleration, but smooth rate of acceleration (due to zero jerk) at the sync position. This makes tracking easier at the sync point.

Master Start Distance < 1.6666 x Slave Distance / Final Gear Ratio



- Notice that the slave velocity increases above the master velocity in order to reach the sync position. This is acceptable or will be necessary in some applications.

If the Slave is not Initially Stopped

If the slave is not initially stopped when this command is issued, the **Master Start Distance** will not be truncated to 2.5 x Slave Distance / Final Gear Ratio. Therefore, the target profile may in some cases move in the other direction before moving in the direction of the master. Make sure to test the Gear Pos (Clutch by Distance) parameters you will be using to verify that the profiles are acceptable.

Gear Ratio Details

This command has only one register to specify the gear ratio. Therefore, it cannot exactly represent certain fractional numbers, such as 1/3. If you require such a ratio, first issue this command with the desired sync positions and an approximate final ratio, then issue the Gear Pos (Clutch by Time) (30) command after the axes have locked into the final gear ratio. Make sure to specify a time of zero (0) for the Gear Pos (Clutch by Time) command. To maintain accuracy, this should be done in a user program. Use the Target Generator Status bits to determine when the axes have locked in the final gear ratio. See the **Target Generator State Bits** section below.

The Gear Pos (Clutch by Distance) command will be accurate up to the sync positions. Therefore, you will not lose any accuracy by using the Gear Pos (Clutch by Distance) followed by a Gear Pos (Clutch by Time) command with a time of zero.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Rotary Motion

The **Master Direction** and **Slave Direction** parameters of this command are for use on rotary axes. For non-rotary axes, the direction parameters can only be "Nearest". The other options will have no effect. If the master register is not a Target Position or Actual Position register, the master directions cannot be rotary.

On rotary axes, the direction parameters specify the direction of the Slave Sync position from it's current position and the direction of the Master Start Distance from the Master Sync Position. See the Rotary Motion topic for details on the meaning of the direction options.

Direction Options

- Negative (-1)
- Nearest (0)
- Positive (1)
- Current (2)
- Absolute (3)

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

Set when the master position is at or beyond the **Master Sync Position**. The gear ratio is now locked and will no longer change even if the master moves prior to the **Master Sync Position**.

Target Generator State A and B bits

B	A	Description
0	0	The master is beyond the **Master Start Position**. Use the Target Generator Done bit to determine if it has reached the **Master Sync Position**.
0	1	Reserved
1	0	The master is prior to the **Master Start Position**.
1	1	Reserved

See Also

Gearing | List of Commands | Commands Overview

8.4.6.6. Command: Phasing (34)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Master Phase Offset** (position-units)	Any REAL number
2	**Master Distance** (position-units)	Any REAL number

Description

This command creates a phase shift in the master position of a slave axis. The master position is shifted in relation to the real physical position. This is analogous to opening a coupling on the master shaft for a moment, and is used to delay or advance an axis to its master. The phase shift

is seen from the slave. The master does not know that there is a phase shift experienced by the slave. The phase shift remains until another phasing command changes it again.

When this command is issued, the axis must be geared and the gear ratio must be locked. This command will not work if the axis is geared with a Gear Absolute (25) command.

As the master moves the **Master Distance** from its position at the time this command was issued, the **Master Distance Offset** will be added to the master position, as seen by the slave. The Master Distance implies the direction with its sign. For example, with an initial master position 5, a **Master Distance** of 2 means that the offset will happen as the master moves from 5 up to 7. A **Master Distance** of -2 means that the offset happens from 5 down to 3.

The Phasing command is related to the Geared Slave Offset (35). The Phasing command shifts the *master* position as seen by the slave. The Gear Slave Offset command shifts the *slave* position itself.

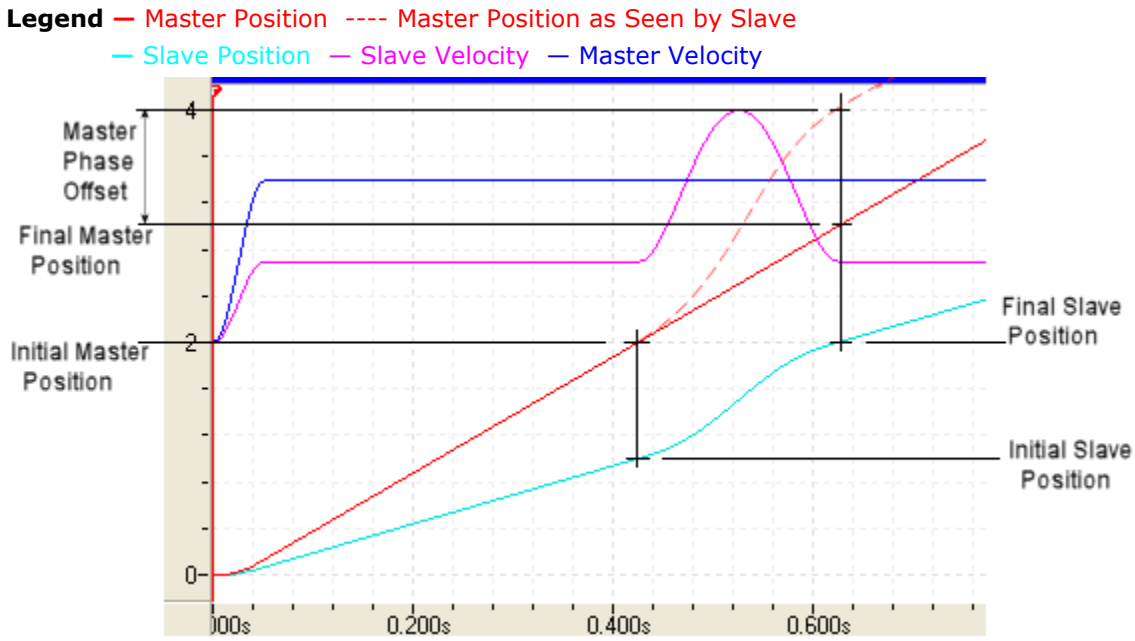
Phasing Example

In this example, Axis 1 (slave) is geared to Axis 0 (master) at a 1:2 ratio. Both axes start at 0 pu. The master is moving at 5 pu/sec to 4 pu. When the master reaches 2 pu, the following Phasing command is issued to the slave.

Master Phase Offset = 1

Master Distance = 1

The plot below shows how the slave moves.



The Initial Master and Slave Positions are the positions at the time the command was issued. The dashed red line is the the Master Position as seen by the slave. The Slave Final Position is 2.0 pu. If the Geared Slave Offset command had not been issued, it would have been at 1.5 pu at that master position. This difference is 0.5, which is a 1:2 to ratio of the Master Distance. Therefore, the Phasing command shifted the master by 1 pu, as seen by the slave.

Reversing Direction

This command allows reversing the direction of the master. If the master reverses direction before it has reached the Final Master Position, the slave will follow the profile in reverse. If the master continues in reverse past the Initial Master Position, the slave will continue following the master as directed from the previous gearing command. The offset will resume if the master returns to the Initial Master Position.

Once the master has reached the Final Master Position, the slave will continue gearing as directed from the previous gearing command whether the master goes forward or backward. It will not revert to the profile it followed before reaching the Final Slave Position.

Specifying an Absolute Position

The **Master Distance** parameter is relative to the master position at the time this command was issued. To specify an absolute position, use an Expression command that calculates the **Master Distance** by subtracting the current master position from the desired absolute position. For example, if the slave is gearing to the Axis 0 Target Position, and you wish to change the master phase by the time it reaches 5 pu, then use the following expression command:

```
MasterDist := 5 - _Axis[0].TarPos
```

where MasterDist is a variable defined in the variable table. Then use this variable in the **Master Distance** parameter of this command.

Rotary Axes

The **Master Distance** and **Master Phase Offset** specify distances on the master axis. These distances are complete distances whether the axis is rotary or not. If the master axis has a Position Unwind of 1, and the **Master Distance** is 2, the **Master Phase Offset** will be done over 2 pu, or 2 rotations of the master. The sign of the values indicates the direction.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

Set when the master position has reached the Final Master Position. The gear ratio is now locked and will no longer change even if the master moves prior to the Final Master Position.

Target Generator State A and B bits

B	A	Description
0	0	The master is beyond the Final Master Position. Use the Target Generator Done bit to determine if it has reached the Final Master Position.
0	1	Reserved
1	0	The master is prior to the Initial Master Position.
1	1	Reserved

See Also

Gearing Overview | Geared Slave Offset (35) | List of Commands | Commands Overview

8.4.6.7. Command: Geared Slave Offset (35)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Slave Offset** (position-units)	Any REAL number
2	**Master Distance** (position-units)	Any REAL number

Description

This command offsets the slave axis position by a specified amount. As the master moves the **Master Distance** from its position at the time this command was issued, the **Slave Offset** will be superimposed upon the slave position. When this command is issued, the axis must be geared and the gear ratio must be locked. This command will not work if the axis is geared with a Gear Absolute (25) command.

The Master Distance implies the direction with its sign. For example, with an initial master position 5, a **Master Distance** of 2 means that the offset will happen as the master moves from 5 up to 7. A **Master Distance** of -2 means that the offset happens from 5 down to 3.

The Geared Slave Offset command is related to the Phasing (34) command. The Phasing command shifts the *master* position as seen by the slave. The Gear Slave Offset command shifts the *slave* position itself.

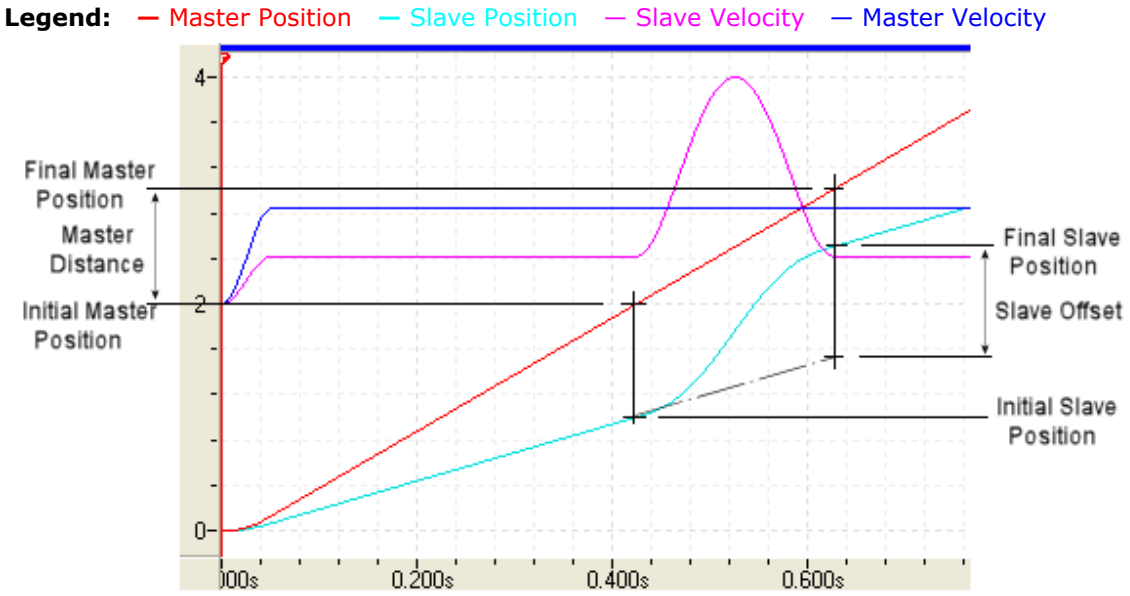
Geared Slave Offset Example

In this example, Axis 1 (slave) is geared to Axis 0 (master) at a 1:2 ratio. Both axes start at 0 pu. The master is moving at 5 pu/sec. When the master reaches 2 pu, the following Geared Slave Offset command is issued to the slave.

Slave Offset = 1

Master Distance = 1

The plot below shows how the slave moves.



The Initial Master and Slave positions are the positions at the time the command was issued. The Slave Final Position is 2.5 pu. If the Geared Slave Offset command had not been issued, it would have been at 1.5 pu at that master position. Therefore, the Geared Slave Offset moved the slave 1 pu, as specified by the command.

Reversing Direction

This command allows reversing the direction of the master. If the master reverses directions before the master has reached the Final Master Position, the slave will follow the profile in reverse. If the master continues in reverse past the Initial Master Position, no slave offset will be applied until the master returns to the Initial Master Position.

Once the master has reached the Final Master Position, the slave will continue gearing (as directed from the previous gearing command) whether the master goes forward or backward. It will not revert to the profile it followed before reaching the Final Slave Position.

Specifying an Absolute Position

The **Master Distance** parameter is relative to the master position at the time this command was issued. To specify an absolute position, use an Expression command that calculates the **Master**

Distance by subtracting the current master position from the desired Final Master Position. For example, if the slave is gearing to the Axis 0 Target Position, and you wish to do the Slave Offset by the time the master reaches 5 pu, then use the following expression command:

```
MasterDist := 5 - _Axis[0].TarPos
```

where MasterDist is a variable defined in the variable table. Then use this variable in the **Master Distance** parameter of this command.

Rotary Axes

The **Slave Offset** and **Master Distance** specify distances on the axes. These distances are complete distances whether the axes are rotary or not. If both axes have a Position Unwind of 1, and the **Slave Offset** is 2 and the **Master Distance** is 3, the slave will be offset 2 pu, or 2 revolutions, while master moves 3 pu, or 3 revolutions. The sign of the values indicates the direction.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

Set when the master position has reached the Final Master Position. The gear ratio is now locked and will no longer change even if the master moves prior to the Final Master Position.

Target Generator State A and B bits

B	A	Description
0	0	The master is at or beyond the Initial Master Position. Use the Target Generator Done bit to determine if it has reached the Final Master Position.
0	1	Reserved
1	0	The master is prior to the Initial Master Position.
1	1	Reserved

See Also

Gearing Overview | List of Commands | Commands Overview

8.4.6.8. Command: Advanced Gear Move (33)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Slave Sync Position** (position-units)	Any REAL number
2	**Final Gear Ratio**	Any REAL number
3	**Final Gear Ratio Rate**	Any REAL number
4	**Master Register**	Valid RMC register

	Note: See **Specifying a Register Address** below.	
5	**Master Sync Position** (position-units)	Any REAL number
6	**Master Direction** If the master axis linear, this must be **Nearest (0)**. The other options will have no effect. If the master axis is rotary, see the **Rotary Motion** section below.	a valid integer as described
7	**Slave Direction** If the slave axis linear, this must be **Nearest (0)**. The other options will have no effect. If the slave axis is rotary, see the **Rotary Motion** section below.	a valid integer as described

Description

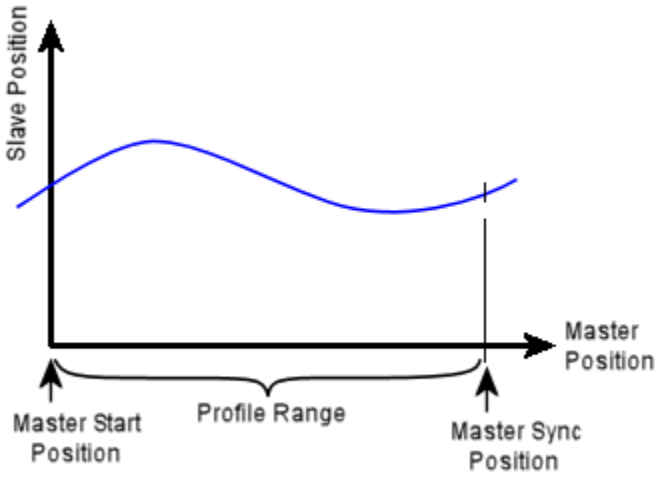
This command calculates a gear target profile (a cam segment) for the slave axis using a 5th-order polynomial based on the position of the master axis. The profile extends along the master from the Master Start Position (position at the time the command was issued) to the **Master Sync Position**.

The Slave Profile

As shown in the diagram below, the slave profile extends along the master from the Master Start Position (position at the time the command was issued) to the **Master Sync Position**. The slave profile between the Master Start Position and **Master Sync Position** endpoints is a 5th-order polynomial. The shape of the slave profile polynomial is defined by the conditions at these endpoints.

The endpoint conditions are:

- **Position**
  The position of the slave.
- **Gear Ratio**
  The gear ratio of the slave to the master. This is the rate of change of the slave position with respect to the master position.
- **Gear Ratio Rate**
  The gear ratio rate of the slave. This is the rate of change of the Gear Ratio with respect to the master position.



Because the profile is a 5th-order polynomial, it can have multiple points of inflection. This means the position can change direction several times. Care must be taken that the profile generates the expected profile. Make sure to test the command with the parameters you expect to use. Breaking the move into shorter sections can help minimize large swings.

Endpoint Conditions - Master Start Position

At the Master Start Position, the slave will be at the Slave Start Position, which is the position at the time the command was issued. The Gear Ratio and Gear Ratio Rate at this position are determined as described below.

If the slave was already geared when this command was issued, the gear ratio will be the slave's current gear ratio. The gear ratio rate will be the current gear ratio rate if the previous command was an Advanced Gear Move. Otherwise it will be zero (0).

If the slave was not geared when this command was issued, the gear ratio is set to the ratio of the current slave velocity to the current master velocity. The gear ratio rate is assumed to be zero (0). If the master is moving very slowly or nearly stopped when this command is issued, the computed gear ratio may be very large. There are two ways to avoid this problem:

- Use a different gearing command, such the Gear Pos (Clutch by Rate) (39) command.
- First issue the Gear Pos (Clutch by Time) (30) command with a 0:1 ratio and a time of zero, then immediately issue this command with the desired parameters.

Endpoint Conditions - Master Sync Position

At the **Master Sync Position**, the slave profile will be at the **Slave Sync Position** with the specified **Final Gear Ratio** and **Final Gear Ratio Rate**.

Following the Profile

While the master is in the Profile Range or on the Master Start Position and **Master Sync Position** endpoints, the slave will follow the profile regardless of the direction the master is moving.

If the master position moves to one of the endpoints and the gear ratio rate at that endpoint is zero, the profile will continue indefinitely at the gear ratio of the endpoint. If the master moves to the **Master Sync Position**, the slave will be locked at that gear ratio. If the master moves beyond the Master Start Position, the ratio will not be locked, meaning that if the master moves back inside the profile, the slave will follow the profile.

If the master position moves beyond one of the endpoints and the gear ratio rate at that endpoint is *non-zero*, the polynomial will continue for two loop times, and if a new command is not issued during this time, then a Closed Loop Halt will occur and the Runtime Error bit will be set. Notice that if the Runtime Error AutoStop is set to a different type of halt, that halt will also occur.

The Target Generator Status bits can be used to check whether the master has moved beyond an endpoint. See the **Target Generator State Bits** section below.

Intended Use

This command is intended to be followed by another command, especially if the Final Gear Ratio Rate is non-zero. Typically, this command is used only in a user program, where it is very easy to issue another command immediately when the Done bit is set, indicating the master has moved beyond the **Master Sync Position**. Use the Wait For or Conditional Jump Link Types to check for the Done bit.

A **Final Gear Ratio** of zero (0) means that the slave will stop when the master reaches the **Master Sync Position**.

Why Bother?

This command offers ultimate gearing flexibility and is useful if you need to specify an arbitrary motion profile as a function of the master position. For example, if you know the equation of the cam profile or cam profile segments you wish to create, you can use this command.

This command is typically used for only a segment of a move, and you should always issue another command after this one when the Done Status bit is set. This can easily be done in the User Programs. You can issue several of these moves in a row to get the cam profile you want, especially if you need to specify the gear ratio and gear ratio rate at certain points. The Expression (113) command can be used to calculate the position, gear ratio, and gear ratio rate at the specific master locations. Complex profiles may require many individual segments using the Advanced Gear Move commands.

Typically, using this command requires knowledge of calculus and a thorough understanding of using expressions in a user program. Notice that the Gear Ratio is the first derivative of the slave

position with respect to the master position and the Gear Ratio Rate is the second derivative of the slave position with respect to the master position.

This command is very similar to the Advanced Time Move Absolute (26), except that it creates a profile based on a master position rather than time. The Final Velocity and Final Acceleration of the Advanced Time Move are similar to the **Final Gear Ratio** and **Final Gear Ratio Rate** of the Advanced Gear Move.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Rotary Motion

The **Master Direction** and **Slave Direction** parameters of this command are for use on rotary axes. For non-rotary axes, the direction parameters should be "Nearest". The other options will have no effect. If the master register is not a Target Position or Actual Position register, the master directions cannot be rotary.

On rotary axes, the direction parameters specify the direction of the Slave Sync position from it's current position and the direction of the Master Start Distance from the Master Sync Position. See the Rotary Motion topic for details on the meaning of the direction options.

Direction Options

- Negative (-1)
- Nearest (0)
- Positive (1)
- Absolute (3)

Using the Advanced Gear Move in User Programs

Indexing Arrays

Typically, the Advanced Gear Move is programmed in a loop and cycles through an array or arrays of positions, speeds, and/or accelerations. To do this, use the variable table to set up arrays. A variable for indexing the array is also required. It must be of data type DINT.

Timing Considerations

When looping through long arrays with the Advanced Gear Move command, precise timing is often important. Each task can execute a maximum of one step per loop time. However, the commands in a step and the link condition in that same step are not executed in the same loop time.

When a step is executed, the commands in that step are issued in that loop time. Then, in the *next* loop time of the controller, the Link Type is evaluated. When the Link Type condition becomes true or tells the program to jump, the program will jump to the specified step and issue the commands in that step *in the same loop time* that the condition became true.

If the master position moves beyond one of the endpoints and the gear ratio rate at that endpoint is *non-zero*, the polynomial will continue for two loop times, and if a new command is not issued during this time, then a Closed Loop Halt will occur and the Runtime Error bit will be set. Make sure to create your program such that the next Advanced Gear Move command will be issued within two loop times after reaching one of the endpoints. The Target Generator Status bits can

be used to check whether the master has moved beyond an endpoint. See the **Target Generator State Bits** section below.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

Set when the master is at or beyond the **Master Sync Position**.

If the master is at or beyond the **Master Sync Position**, *and* the final Gear Ratio Rate is zero (0), this bit will be latched and the gear ratio will no longer change even if the master moves back within the Profile Range.

Target Generator State A and B bits

B	A	Description
0	0	The master is within the Profile Range. Or, the master has moved beyond the **Master Sync Position** and the final **Gear Ratio Rate** is zero. Use the Target Generator Done bit to determine which is the case.
0	1	The master is beyond (not at) the **Master Sync Position** and the final **Gear Ratio Rate** is non-zero. The axis will fault if a new command is not received in the next control loop.
1	0	The master is prior to the **Master Start Position**, and the initial gear ratio rate is zero.
1	1	The master is prior to (not at) the Master Start Position and the initial gear ratio rate is non-zero. The axis will fault if a new command is not received in the next control loop.

See Also

Gearing | List of Commands | Commands Overview

8.4.6.9. Command: Track Position (57)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Master Register** **Note:** See **Specifying a Register Address** below.	Valid RMC register
2	**Velocity Limit** (position-units/s)	>0
3	**Acceleration Limit** (position-units/s^2) This value should typically be at least 10 times the Velocity Limit.	>0
4	**Jerk Limit** (position-units/s^3)	>0

	This value should typically be at least 20 to 100 times the Acceleration Limit.	
5	**Deadband** (position-units)	≥0

Description

This command continuously tracks the specified **Master Register**. The axis Target Position is limited by the specified **Velocity Limit**, **Acceleration Limit**, and **Jerk Limit** and the Positive and Negative Travel Limits. This command is useful for smoothly tracking a signal containing noise or step-jumps, or for gearing to another position while not exceeding specified motion limits. The **Deadband** can be used to prevent the axis from following small amounts of noise while the master signal is for all practical purposes stopped.

Velocity, Acceleration, and Jerk Limits

It is very important that the **Velocity Limit**, **Acceleration Limit**, and **Jerk Limit** be set correctly. Acceleration is the rate at which the velocity increases. Jerk is the rate at which the acceleration increases. If the **Acceleration Limit** or **Jerk Limit** are set too low, the axis can significantly overshoot the master value. The larger the **Acceleration Limit** and **Jerk Limit** values are, the closer the axis will track. The **Jerk Limit** will especially affect how the axis tracks all the quick, small changes of the master.

Typically, the **Acceleration Limit** should be at least 10 times the **Velocity Limit**, and the **Jerk Limit** should be at least 20 to 100 times the **Acceleration Limit**.

For example, if Axis 0 should track the Axis 1 Actual Position and be limited to 10 in/sec, then the Track (57) command could be sent as follows:

Track Position (57)	
Master Register	_Axis[1].ActPos
Velocity Limit	10
Acceleration Limit	100
Jerk Limit	10000
Deadband	0

Setting the velocity, acceleration, and jerk limits to properly follow a highly dynamic signal can be challenging. To minimize overshoot, the acceleration and jerk limits should be significantly larger than the acceleration and jerk components of the master. Delta recommends that you test this command in the expected scenarios of your machine to make sure it will work as expected during normal operation.

Deadband

If you wish to use the deadband, enter a value that is large enough to reject the expected amount of noise on the **Master Register**. The deadband will begin to apply whenever the axis is stopped—as defined by the axis Target Velocity being less than the Stop Threshold parameter—for more than 10 loop times.

When the axis is stopped, the deadband will cause the axis to not follow the master until the master moves the specified deadband value from its (the master's) stopped location. When that occurs, the axis will begin to track the master. If the velocity, acceleration, and jerk limits are large, this initial motion can cause the axis to jerk suddenly. To minimize the initial sudden jerk, use the smallest Deadband value necessary.

Rotary Axes

When this command is sent to a rotary axis, the direction of motion will be in the Absolute Direction.

When the Master Register is a position from a rotary axis, the slave axis Target Position will continue in its current direction when the master position wraps. If you need the opposite behavior, that is, the slave axis attempts to always be at the same value as the rotary position,

and moves back when the master wraps, you can make a user program that continuously copies the rotary position to a variable, then use the variable as the Master Register.

Other Notes

Changes in the master value can effect the Target Position, even if the Target Position is far from the master value. Reducing the jerk and acceleration limits can help minimize this should it be a problem. Or, use the Track Position (I-PD) (58) command.

If you do not need any velocity, acceleration, or jerk limits, consider the Gear Absolute (25) command.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Status Bits

In Position Bit

Indicates the axis is in position. This bit is set when the Target Generator Done bit is set *and* the Actual Position is within the In Position Tolerance of the Target Position. This bit does not apply to virtual axes.

Target Generator Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done Bit

Indicates that the Target Position is stopped and stable. This bit is set if the slave axis Target Position is stopped—as defined by the slave axis Target Velocity being less than the Stop Threshold parameter—and the slave axis Target Position is within the In Position Tolerance of the master value. This bit is not valid for virtual axes.

Target Generator State A Bit

Indicates the Target Position is closely matched to the Master Register. This bit is set if the slave axis Target Position is within the In Position Tolerance of the master value, regardless of what the velocities are. This bit is not valid for virtual axes.

Target Generator State B Bit

Indicates the Target Position and Target Velocity are closely matched to the Master Register. This bit is set if the slave axis Target Position is within the In Position Tolerance of the master value *and* the slave axis Target Velocity is within the Stop Threshold of the master velocity. This bit is not valid for virtual axes.

See Also

Track Position (I-PD) (58) | Gear Absolute (25) | Gearing

8.4.6.10. Command: Track Position (I-PD) (58)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Master Register** **Note:** See **Specifying a Register Address** below.	Valid RMC register
2	**Velocity Limit** (position-units/s)	≥0

Description

This command will switch the axis to Position I-PD mode and will track the specified **Master Register**. The Target Position of the axis will always be identical to the value of the specified **Master Register**. The axis' Actual Position is limited by the **Velocity Limit** and by the positive and negative travel limits. This command is useful for smoothly tracking a signal containing noise or step-jumps, or for gearing to another position while not exceeding specified motion limits.

In I-PD mode, the axis will always have a following error. Compare this to the PID mode, where the feed forwards make it possible to follow a signal very precisely. Therefore, I-PD can be suitable for following targets with noise or step-jumps. Delta recommends familiarizing yourself with Position I-PD mode before using this command.

Velocity Limit

The *Actual* Velocity of the axis will be limited by the **Velocity Limit** command parameter. Notice that the *Target* Velocity of the axis will be identical to the rate of change of the **Master Register**.

Travel Limits

For linear axes, the axis motion will be limited by the Positive and Negative Travel Limits. The limit will be applied smoothly, such that the axis will not stop abruptly at the limit. The travel limits can be changed while this command is in progress, and the new limits will immediately be applied.

Axis Response

The response of the axis' Actual Position is defined by the **Velocity Limit** and by the tuning. In I-PD mode, the P, I, and D gains define how quickly the axis gets into position.

Other Notes

Since this command uses I-PD mode, it may cause the axis to lag the master value. If you need to more closely track the master value, consider the Track Position (57) command. If you do not need any velocity, acceleration, or jerk limits, consider the Gear Absolute (25) command.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Status Bits

In Position Bit

Indicates the axis is in position. This bit is set when the Target Generator Done bit is set *and* the Actual Position is within the In Position Tolerance of the Target Position.

Target Generator Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done Bit

Indicates that the Target Position is stopped. This bit is set if the Target Position is stopped—as defined by the Target Velocity being less than the Stop Threshold axis parameter.

Target Generator State A and B Bits

The Target Generator State A and State B bits do not apply.

See Also

Track Position (57) | Gear Absolute (25) | Gearing

8.4.7. Specialty

8.4.7.1. Command: Speed at Position (36)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Position** (position-units)	any
2	**Requested Velocity** (position-units/s)	any

Description

This command specifies a profile in terms of speed versus position. The RMC compares the current Target Position with the **Requested Position** and computes a target profile that reaches the **Requested Velocity** at the **Requested Position**. Once the **Requested Position** (and therefore **Requested Velocity**) is reached, the Done status bit will be set. If the **Requested Velocity** was non-zero, the target generator will continue at the final **Requested Velocity** indefinitely. This command is a closed loop control command.

Notice that it is difficult to determine exactly how long the move will take. This command will attempt to move the axis using the shortest path possible and will attempt to make the Target Acceleration non-zero during the entire move.

If the **Requested Position** is behind the Actual Position at the time this command was issued, a Command Error or Runtime error will be set. "Behind" refers here to being in the direction opposite to the current direction of travel. Similarly, if the axis is stopped, the Requested Velocity must be positive if the **Requested Position** is greater than the current position, and negative if the **Requested Position** is less than the current position.

Why Bother?

This command is good when you want to specify the ends, not the means. There are two applications where this really works well:

- When you suddenly want to stop at a certain position, but do not know what deceleration rates you need. Maybe you usually use a deceleration of 100 in/s^2, but if the axis is travelling fast close to the position where you want to stop, the decel rate of 100 might not be enough to stop you fast enough. If you issue a Speed at Position (36) command with a Requested Velocity of zero, the RMC will calculate the deceleration for you and will get you stopped at the requested position. Note that if you issue this command a long distance before you stop, the calculated decelerations may be so low that it takes a long time to stop.
- Injection molding applications often specify the speed relative to the position. You can set up a user program that issues this command for each trigger point, and it will go to the Requested Velocity at the position.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Position has reached the **Requested Position**.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Acceleration
1	0	Constant Velocity
1	1	Deceleration

See Also

List of Commands | Commands Overview

8.4.7.2. Command: Sine Start (72)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Offset** (position-units)	any
2	**Amplitude** (position-units)	≥ 0
3	**Frequency** (Hz)	0* to 1/4 of the loop frequency
4	**Cycles**	0 to 16 million in 0.25 steps (0 = continuous)

5	**Start Location** • Auto (0) • Mid-Pos (1) • Pos Peak (2) • Mid-Neg (3) • Neg Peak (4)	a valid integer as described
6	**Status Block** (address) **Note:** See **Specifying a Register Address** below.	Address or **none** (0)

*See the **Frequency** section below for details on the lower limit.

Description

This command starts a continuous sinusoidal move on a position axis. If the number of cycles is specified, the sine move will end after completing the cycles and will hold position. For pressure or force, use the Sine Start (Prs/Frc) (76) command.

Before issuing this command, the Target Position must be at the location specified by the **Start Location** command parameter, or, a Transition command must previously have been issued to specify how the axis should move to get to the sine curve. A transition command will allow you to start a sine curve even though the axis is not at the correct starting point.

If you want the axis to follow the sine curve exactly starting from the beginning, do not use a transition; instead, make sure the axis is at the starting point before issuing the Sine Start command.

Offset

The **Offset** specifies the position of the center of the sine wave.

Amplitude

The **Amplitude** specifies the distance from the center to the peak. The amplitude can be zero or a positive number. An amplitude of zero is typically only used when it will be followed by the Change Target Parameter (80) command to ramp the amplitude from zero to some value.

If the amplitude would cause the target position to exceed the Positive or Negative Travel Limits, a command error will occur, which will halt the axis if the Auto Stops are set to do so.

Cycles

The **Cycles** specifies the number of cycles, in increments of 0.25. Each cycle begins at the location specified by the **Start Location** parameter. If **Cycles** is zero, the sine wave will repeat endlessly. By specifying various increments of 0.25, you can control whether the sine wave stops at the bottom, top, or middle. If the **Cycles** is an integer number, the sine move will stop at the same location where it started.

A cycle count of 0.25 is not allowed if the Start Location is at one of the peaks, and the Frequency is zero.

Due to the limits of 32-bit floating point values, the Cycles parameter can be a maximum of 16 million only if it is a whole number. If it contains a half, such a 45.5, it can be a maximum of 7,999,999.5. If it contains a quarter, such a 12.25 or 97.75, it can be a maximum of 3,999,999.75.

Frequency

The frequency can be any value from zero up to the maximum frequency listed in the table below. The maximum frequency is a quarter of the loop frequency (the loop frequency is the inverse of the Loop Time). For example, if the loop time is 1000 µsec, the maximum frequency is 250 Hz.

Loop Time	Max Frequency
250 µsec	1000 Hz
500 µsec	500 Hz
1000 µsec	250 Hz
2000 µsec	125 Hz
4000 µsec	62.5 Hz

Fractional frequencies are valid, such as 0.345 or 5.84. Issuing this command with a frequency of zero is typically only done when it will be followed by the Change Target Parameter (80) to ramp the frequency from 0 to some value. A frequency of zero is not allowed if the Start Location is at one of the peaks, and the cycle count is 0.25.

Frequencies below 0.00001 on the RMC75 or RMC150E, and below 0.05 on the RMC75S and RMC75P, may cause an irregular Target Position when used with the Start Locations Mid-Pos or Mid-Neg. Test any very low frequencies before using them.

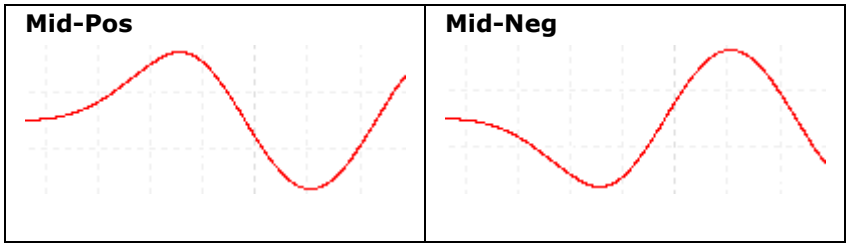
Start Location

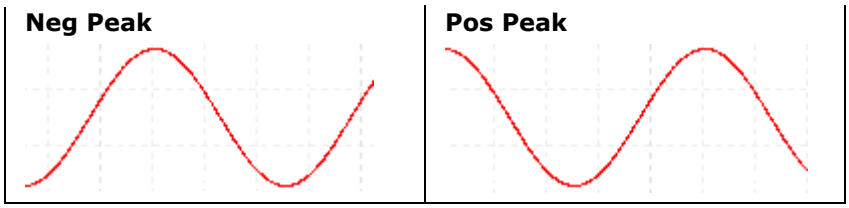
The **Start Location** specifies where on the sine wave the motion should start. When this command is issued, the Target Position must be at that position, unless a Transition command has previously been issued to specify how the axis should move to get to the sine curve. The **Start Location** options are given below. Notice that each cycle will start at the location specified by the **Start Location**.

Start Location	Description
Auto	If the Target Position is at the positive peak, negative peak, or the center, then the sine move will start there. If it starts at the center, it will move in the positive direction. See **Mid-Pos and Mid-Neg Details** below.
Mid-Pos	The sine wave will start at the center (zero degrees) and move in the positive direction. See **Mid-Pos and Mid-Neg Details** below.
Pos Peak	The sine wave will start at the positive peak (90 degrees).
Mid-Neg	The sine wave will start at the center (180 degrees) and move in the negative direction. See **Mid-Pos and Mid-Neg Details** below.
Neg Peak	The sine wave will start at the negative peak (270 degrees).

For example, if the **Start Location** is Auto, the **Offset** is 3, and the **Amplitude** is 1, then the Target Position must be at 2, 3, or 4 when this command is issued. If the **Start Location** is Pos Peak, the **Offset** is 10, and the **Amplitude** is 2, then the Target Position must be at 12 when this command is issued.

The diagrams below show the start of the sine wave for the **Start Location** options.





Mid-Pos and Mid-Neg Details

If the **Start Location** is **Mid-Pos** or **Mid-Neg**, in order to prevent a sudden jump in the velocity, the frequency will be logarithmically ramped from zero Hertz to the requested frequency during the first quarter cycle. Therefore, the first quarter cycle will not be a true sine function, and the first quarter cycle will take longer. See the **Mid-Pos** and **Mid-Neg** images above. If a sine move stops at the middle, the same type of ramping will occur in the last quarter cycle.

To achieve a different ramping behavior for **Mid-Pos** or **Mid-Neg** start locations, do the following:

- Send the **Sine Start** command with the frequency set to zero. The sine wave will start, but there will be no motion.
- Use the **Change Target Parameter** command to ramp the frequency to the desired value. The **Change Target Parameter** command provides great flexibility in defining the ramp.

Note: For Start Locations **Mid-Pos** or **Mid-Neg**, very low frequencies, such as 0.0001 or less, may cause an irregular Target Position. Test any very low frequencies before using them.

Note: For Start Locations **Mid-Pos** or **Mid-Neg** with a non-zero starting frequency, changing the frequency with the Change Target Parameter (80) command is not allowed until the sine wave has reached the first positive or negative peak.

Status Block

The optional **Status Block** specifies the location in the Variable Table of a block of six registers that provide read-only information on the sine move. This block will not be needed by most users and the Status Block parameter should then be set to **none**. For more details, see the **Sine Move Status** section below.

Stopping a Sine Move

If the number of cycles is non-zero, the sine move will automatically stop after reaching the number of cycles. By specifying the **Cycles** in various increments of 0.25, you can control whether the sine wave stops at the bottom, top, or middle. If the **Cycles** is a whole number, the sine move will stop at the same location where it started.

The Sine Stop (73) command is designed for stopping a sine move that is in progress. The stop location can be specified to be the positive peak, negative peak, next peak, middle, or after the current cycle completes. See the Sine Stop (73) command for more details.

If the stop location is the middle, the frequency will be ramped to zero hertz during the last quarter cycle. This prevents a sudden jump in the velocity, because the middle of a sine move otherwise has a non-zero velocity.

Of course, as with any motion on an axis, when a sine move is in progress, another motion command can be issued and the axis will immediately stop the sine move and start the new motion. For example, a Stop (Closed Loop) (6) command will stop the axis.

Ramping Sine Move Parameters

Use the Change Target Parameter (80) command to ramp the Offset, Frequency, or Amplitude, or change the Cycles of a sine move in progress. Each parameter can be ramped independently, that is, each parameter can be ramped whether or not other parameters are ramping. See the Change Target Parameter (80) command for details.

Example

A wave-generating application requires that the amplitude be ramped from 0 to 10 over 20 cycles. To do this, first issue the Sine Start (72) command with an Amplitude of 0. Then, issue the Change Target Parameter (80) command to ramp the Amplitude to 10 during 20 cycles.

Sine Move Status

Axis Status Registers for a Sine Move

The Cycles Axis Status register gives the whole number of cycles completed for the sine move in progress. It is listed in Axis Tools, in the Axis Status Registers pane, on the All tab, in the Target section. The Cycles Axis Status register is a DINT.

Status Block

Advanced users may wish to use the Sine Start command's **Status Block**, which provides read-only information on the sine move. This information is most useful when manipulating sine moves in user programs.

To use the Status Block, you must specify an address from the Variable Table in the **Status Block** parameter of the Start Sine command. The Status Block will require seven registers in the Variable Table, beginning with the specified address. As the sine move progresses, the selected registers in the Variable Table will be continuously updated. The selected variables will not be named automatically; you should give a descriptive name to each to help you keep track of them.

To prevent confusion, sine moves that are running simultaneously should not use the same Status Block address. Non-simultaneous sine moves can use the same Status Block address.

The Status Block provides the following information:

Status Block Offset	Name	Data Type	Description
0	Current Cycle Count	REAL	The number of whole cycles the sine move has completed. Each cycle begins at the location specified by the **Start Location** command parameter. The fractional part of the cycle is given by the Current Cycle Fraction below. For continuous sine moves (without a fixed number of cycles), this value will wrap to zero after it reaches 10,000,000 and then continue incrementing. For sine moves with a fixed number of cycles, this value will not go beyond the requested cycle count.
1	Current Cycle Fraction	REAL	The fractional part of the Current Cycle above. The fractional part is given in this separate register to retain accuracy as the Current Cycle reaches high values.
2	Current Angle	REAL	The current angle from 0 up to, but not including, 360 degrees. This angle is the mathematical angle as shown in the sine wave diagram above. An angle of zero does not necessarily coincide with the start of a cycle. This value can be used for such things as determining whether the sine move is at the positive peak, negative peak, or middle.
3	Current Amplitude	REAL	The current amplitude of the sine move in position-units.
4	Current Frequency	REAL	The current frequency of the sine move in hertz.
5	Current Offset	REAL	The current offset of the sine move in position-units.
6	Current Phase	REAL	The current phase of the sine move in degrees.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Status Block** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Status Bits

Target Generator Done bit

This bit indicates that the sine move has completed the specified number of cycles and has reached the ending point. Notice that this bit does not indicate whether the Actual Position has reached the Requested Position. If the Target Generator Done bit is set, and the Actual Position is within the In Position Tolerance window of the Target Position, the In Position Status bit will be set. The In Position bit indicates that the move is complete and the axis is at position.

Target Generator State A and B bits

B	A	Description
0	0	Reserved
0	1	Reserved
1	0	Reserved
1	1	Reserved

Pri. TG SI Busy (Primary Target Generator Superimposed Busy) Bit

This bit will be set during the transition. The transition begins when the motion command is issued, not necessarily when the Transition command is issued. When the transition completes, this bit will clear.

See Also

Sine Stop (73) | Sine Start (Prs/Frc) (76) | Sine Stop (Prs/Frc) (77)

8.4.7.3. Command: Sine Stop (73)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Stop Location** • Next Cycle (0) • Middle (1) • Pos Peak (2)	a valid integer as described

	• Neg Peak (3) • Next Peak (4)	

Description

This command stops a position sine move at a prescribed location. For stopping a pressure/force sine move, see the Sine Stop (Prs/Frc) (77) command.

The sine move will stop at the next occurrence of the specified **Stop Location**. The **Stop Location** can be any of the following:

- **Next Cycle**
  The sine move will stop at the end of the current cycle. The location of the end of each cycle is defined by the Sine Start (72)command. Stopping the sine move with the **Next Cycle** option will ensure that the sine move stops in the same location that it started. For example, if it started at the positive peak, this command will stop it at the next positive peak.

  If the stop location is the middle, and the axis is currently on the quadrant after a peak and before the midpoint, then the axis will not stop on that first middle point because it could lead to extreme accelerations. Instead, the axis will continue another full cycle and then stop at the middle point.

  If the stop location is the middle, the target generator will perform the stop by ramping the frequency down to zero over the last quarter-cycle. Due to this, any attempt to ramp the frequency with the Change Target Parameter (80) command during this portion of the move will be ignored.
- **Middle**
  The sine move will stop at the first middle point. However, if the axis is currently on the quadrant after a peak and before the midpoint, then the axis will not stop on that first middle point because it could lead to extreme accelerations. Instead, the axis will continue a half-cycle after the first middle point and stop at the next middle point.

  When the **Stop Location** is set to **Middle**, the target generator performs the stop by ramping the frequency down to zero over the last quarter-cycle. Due to this, any attempt to ramp the frequency with the Change Target Parameter (80) command during this portion of the move will be ignored.
- **Pos Peak**
  The sine move will stop at the next positive peak.
- **Neg Peak**
  The sine move will stop at the next positive peak.
- **Next Peak**
  The sine move will stop at the next peak, whether it is negative or positive.

Status Bits

In Position Bit

After the target position stops, and the Actual Position is within the In Position Tolerance window, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the target has stopped.

Target Generator State A and B bits

B	A	Description
0	0	Reserved
0	1	Reserved

1	0	Reserved
1	1	Reserved

See Also

Sine Start (72) | Sine Start (Prs/Frc) (72) | Sine Stop (Prs/Frc) (76)

8.4.7.4. Command: Change Master (79)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD
Firmware Limitations:	RMC75/150: 3.56.0 or newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Master Register** **Note:** See **Specifying a Register Address** below.	_Time or any other valid register of type REAL.

Description

This command immediately changes the master register of a curve that is currently in progress on the axis. A curve must currently be in progress on the axis when this command is sent, or a Command Error will occur.

If the velocity of the old master and the new master are different, the axis may experience a jerk when the master changes. Therefore, this command is intended to be used only when the speed of the old master and new master are approximately equal. An example application is the execution of a curve across the transition from one master conveyor belt to another master conveyor belt.

When the master is changed, the master endpoints are recalculated relative from the position of the new master at the time of the change.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

_Time:

The register address for the _Time tag in the RMC75 is %MD20.10. Therefore, the value is 20*4096 + 10 = 81930.

The register address for the _Time tag in the RMC150 is %MD44.10. Therefore, the value is 44*4096 + 10 = 180234.

The register address for the _Time tag in the RMC200 is %MD18.10. Therefore, the value is 18*4096 + 10 = 73738.

See Also

Curve Start (86) | Curve Start Advanced (88)

8.4.7.5. Command: Change Target Parameter (80)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Parameter** • Offset (0) • Amplitude (1) • Frequency (2) • Cycles (3) • Phase (4)	A valid integer as described
2	**New Value**	Depends on selected Parameter
3	**Ramp Type** • Time (0) • Cycles (1) • Rate (2) • Time (log) (3) • Cycles (log) (4) • Time - 5th (5)	A valid integer as described
4	**Ramp Value**	Depends on selected Ramp Type

Description

This command changes a target parameter for Sine Start (72) moves that are in progress. For example, this command can be used to perform frequency sweeps by ramping the frequency of a sine move in a certain amount of time. For pressure or force, see the Change Target Parameter (Prs/Frc) (81) command.

Target parameters can be ramped independently of each other. That is, each parameter can be ramped whether or not other parameters are ramping. Multiple Change Target Parameter commands can be issued from a single step in a user program, allowing them to be ramped together.

If ramping of frequency and phase is in progress when the sine wave begins a stop at the middle position, the frequency and phase ramping will cease when the sine wave begins its stop, which occurs at the last peak before the middle position.

If ramping of the offset is in progress when the sine wave stops, there may be a discontinuity in the Target Prs/Frc Rate.

Sine Start Parameters

When this command is used for changing target parameters of motion started with a Sine Start (72) command, the following parameters can be changed:

Offset

When ramping the offset, there will be a discontinuity in the velocity at the beginning and end of the ramp, except for the Time-5th ramp type. This can cause the axis to jump.

Valid Ramp Types	
Time	Ramp the Offset linearly in the number of seconds specified by the **Ramp Value**.
Cycles	Ramp the Offset linearly in the number of cycles specified by the **Ramp Value**.
Rate	Ramp the Offset linearly at the rate (units/sec) specified by the **Ramp Value**.
Time-5th	Ramp the Offset using a fifth-order polynomial in the number of seconds specified by the **Ramp Value**. This results in a smoother start and finish than can be obtained with the other ramp types. The starting rate of change is assumed to be zero.

Amplitude

The amplitude can be ramped to any positive value, or zero.

Valid Ramp Types	
Time	Ramp the Amplitude linearly in the number of seconds specified by the **Ramp Value**.
Cycles	Ramp the Amplitude linearly in the number of cycles specified by the **Ramp Value**.
Rate	Ramp the Amplitude linearly at the rate (units/sec) specified by the **Ramp Value**.
Time-5th	Ramp the Amplitude using a fifth-order polynomial in the number of seconds specified by the **Ramp Value**. This results in a smoother start and finish than can be obtained with the other ramp types. The starting rate of change is assumed to be zero.

Frequency

The frequency can be ramped to any positive value, or zero.

Valid Ramp Types	
Time	Ramp the Frequency linearly in the number of seconds specified by the **Ramp Value**.

Cycles	Ramp the Frequency linearly in the number of cycles specified by the **Ramp Value**.
Rate	Ramp the Frequency linearly at the rate (units/sec) specified by the **Ramp Value**.
Time (log)	Ramp the Frequency logarithmically in the number of seconds specified by the **Ramp Value**. Ramping frequency logarithmically typically gives a smoother frequency transition than linear ramping, especially over wide frequency ranges.
Cycles (log)	Ramp the Frequency logarithmically in the number of cycles specified by the **Ramp Value**. Ramping frequency logarithmically typically gives a smoother frequency transition than linear ramping, especially over wide frequency ranges.

Note: If the sine move is in the process of completing the first or last quarter-cycle of a sine move that respectively starts or ends at the midpoint, then any attempt by the user to ramp the frequency during this portion of the move will be ignored.

Cycles

The Cycles parameter is updated immediately when this command is issued. The **Ramp Type** and **Ramp Value** parameters are ignored. The Cycles specifies the total number of cycles from when the Sine Start command was issued.

Example

Consider a sine move that was started with a Cycles parameter of 1000, and when the cycle count reached 600 the Change Target Parameter command is issued to change the Cycles to 1300. The sine move will continue until it completes 1300 cycles from when the Sine Start command was issued. Therefore, it will complete 700 more cycles after the Change Target Parameter command is issued.

If the Cycles is less than the current cycles, the sine move will end. It will end at the next occurrence of the same ending location that is specified by the new Cycles parameter. This location will be the same as the starting phase if the requested Cycles is a whole number, but different otherwise.

Example

Consider a sine move that started with an Offset of 0, an Amplitude of 2, a Cycles parameter of 100, and a Start Location of Mid-Pos. After 30 cycles, the Change Target Parameter command is issued to change the Cycles to 21.25. Since 21.25 would result in an ending location at the positive peak, the sine move will end at the next occurrence of the positive peak, or a position of 2.

Phase

The phase can be ramped to any value from -360 to +360. The phase is relative to the phase when the Sine Start command was issued, which is always zero.

Valid Ramp Types	
Time	Ramp the Phase linearly in the number of seconds specified by the **Ramp Value**.
Cycles	Ramp the Phase linearly in the number of cycles specified by the **Ramp Value**. In cases when the frequency is also changing, the ramping of phase by cycles differs from that of the ramping of frequency, amplitude, and offset, since the ramping of phase by cycles does not take into account the additional progress through the cycle caused by the change in phase.
Rate	Ramp the Phase linearly at the rate (units/sec) specified by the **Ramp Value**.
Time-5th	Ramp the Phase using a fifth-order polynomial in the number of seconds specified by the **Ramp Value**. This results in a smoother

	start and finish than can be obtained with the other ramp types. The starting rate of change is assumed to be zero.

See Also

Sine Start (72) | Change Target Parameter (Prs/Frc) (81)

8.4.7.6. Command: Curve Add (82)

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Data Type	Range
1	**Curve ID**	Internal: DINT External: REAL	0-50000
2	**Curve Data** (address) **Note:** See **Specifying a Register Address** below.	REAL	any Variable Table address
3	**Interpolation Method** • Constant (0) • Linear (1) • Cubic (2) - default	REAL	a valid integer as described
4	**Life Cycle** • Standard (0) - default • Start-Once (1) • Complete-Once (2) • Permanent (3)	REAL	a valid integer as described

Description

This command is used for creating curves in the RMC from a PLC, PC, or even from user programs. For details on the procedure, see Creating Curves Using the Curve Add Command. This command is not necessary when creating curves in the Curve Tool. Once the curve data has been written to the Variable Table, sending this command will create the curve in the RMC.

Once a curve has been created, it can be viewed in the Curve Tool, and can be used with the Curve Start (86) or Curve Start (Prs/Frc) (87) command for a time-based profile or camming based on a master.

Creating a curve requires some processing time that typically exceeds one loop time. It is important to not start a curve before it has been processed and added to the Curve Store. The RMC will report the state of the Curve Add in the **Curve Status** register, which is part of the Curve Data.

For more details, refer to the Creating Curves Using the Curve Add Command topic.

Curve ID

This provides a way of identifying the curve. This can be any number from 0 to 50,000. The ID will be used in the Curve Start (86) and Curve Start (Prs/Frc) (87) commands to specify which

curve to start. A maximum of 128 curves can be stored in the Curve Store, although each ID can be from 0 to 50,000. For the Multiple Curves format, the curves will receive sequential IDs beginning with this Curve ID.

If a curve with the requested ID already exists, the existing curve will be deleted and the new curve with that ID added.

Curve Data

The **Curve Data** parameter specifies the starting address of the curve data in the Variable Table. For details on the curve data, see the Curve Data Formats topic.

For all the Curve Formats, the first item in the curve data is the **Curve Status**. Because it is the first item, the Curve Status is located at the address specified by the **Curve Data** parameter. The RMC reports the state of the curve processing in this Curve Status register. Adding a curve requires some processing time that typically exceeds one loop time. It is important to not try to use a curve until it has been processed and added to the Curve Store. When the Curve Status is 3, the curve has been added to the Curve Store and is ready to use.

The following Curve Status values are defined:

- (0) Processing Not Started

  By convention the user may want to set this register to this value before issuing the command. This is not strictly necessary, but otherwise the user must make sure not to check this register until after the command has been received.

- (1) Processing

  Once the command has been received, the Status will immediately be set to Processing. While in this state the command is currently using the Curve Data structure, and the curve is not ready for interpolation.

- (2) Part Complete

  Very long curves can be submitted in parts, as described in the Adding Large Curves topic. Once the command has completed processing this Curve Data structure (i.e. this part of the total curve), this status value will be used. The user can now write down the next part of the curve.

- (3) Curve Ready

  Once the curve data has been processed, and the entire curve is ready for interpolation, the status will be set to this value. Notice that this value will also be used for the final part for long curves submitted in parts. Therefore, for the last part of a curve, the **Part Complete (2)** state won't be used—the status will change directly from **Processing (1)** to **Curve Ready (3)**.

- (10+) Error

  Values of 10 and above will indicate various errors. When an error code is set in this register, it indicates that the command is done trying to process this Curve Data, but the curve was not submitted to the curve store. See the Curve Status Error Codes topic for a list of possible error values.

Interpolation Method

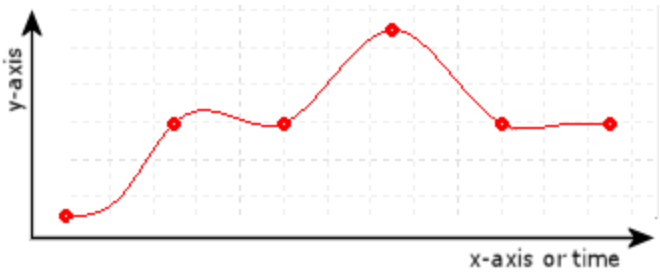
For the **Interpolation Method** parameter, choose from one of the methods below. The **Cubic (2)** method is the most common method and creates the smoothest motion.

- **Cubic (2)**

  The curve will smoothly go through all points. This is the most common interpolation method. This method will create smooth motion.

  On position axes, the Velocity Feed Forward and Acceleration Feed Forward will apply to cubic interpolated curves, but higher order gains should not be used, such as the Jerk Feed Forward, Double Differential Gain, and Triple Differential Gain.

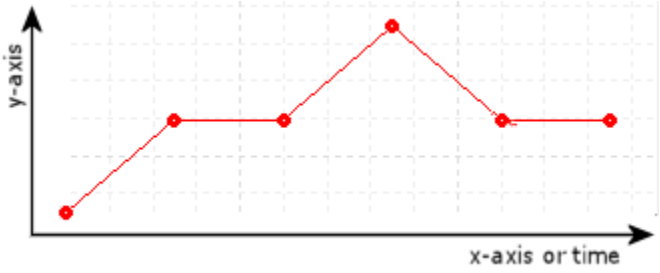
  On pressure or force axes, the Pressure/Force Rate Feed Forward will apply.



- **Linear (1)**

  The curve will consist of straight-line segments between each point. Because the velocity is not continuous, a position axis will tend to overshoot at each point. This type of curve is typically more suitable for pressure or force axes.

  On position axes, the Target Acceleration will always be zero. Therefore, the Acceleration Feed Forward will have no effect for linear interpolated curves.

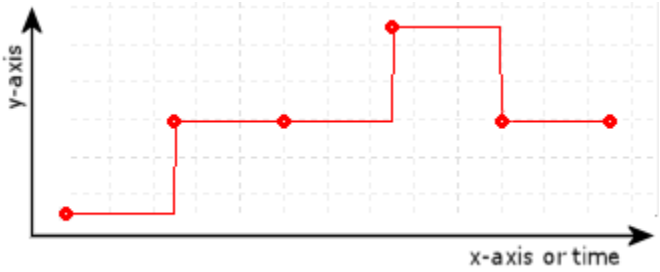


- **Constant (0)**

  The curve will consist of step jumps to each point. The curve will not be continuous. This method is seldom used, but may be useful in applications where step jumps are desired, such as some blow-molding systems. This method requires that the axis not be tuned very tightly, or the axis may oscillate and Output Saturated errors may occur. The Position I-PD control algorithm is recommended for following constant interpolated curves.

  On position axes, the Target Velocity and Target Acceleration will always be zero. Therefore, the Velocity Feed Forward and Acceleration Feed Forward will have no effect for constant interpolated curves.

  On pressure or force axes, the Target Rate will always be zero. Therefore, the Pressure/Force Rate Feed Forward will have no effect for constant interpolated curves.



Life Cycle

Determines how long the curve is stored:

- **Standard (0)**
  Added to curve store, but will not be saved to flash. Must be manually deleted. Standard curves will not be included in the upload or download in the Curve Tool, but can be viewed in the Curves in Controller window.
- **Start-Once (1)**
  Added to curve store, but will not be saved to flash. This curve will be deleted automatically as

soon as an interpolation of this curve has been started, but the curve will continue to be executed correctly. Start-Once curves will not be included in the upload or download in the Curve Tool, but can be viewed in the Curves In Controller window.

- **Complete-Once (2)**
  Added to curve store, but will not be saved to flash. This curve may be deleted manually, but will also be deleted automatically once an interpolation of this curve has been started and completed to the end of the curve (indicated by when the Target Generator Done bit turns on). Complete-Once curves will not be included in the upload or download in the Curve Tool, but can be viewed in the Curves in Controller window.
  Notice that if the Life Cycle is set to Complete-Once, but the curve is run with infinite cycles (Cycles = 0), the curve will never complete and therefore, will not automatically delete.
- **Permanent (3)**
  Added to curve store, and will be saved to Flash when Flash is updated. Must be manually deleted. Permanent curves *will* be included in the upload or download in the Curve Tool.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Curve Data** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

See Also

Curves Overview | Curve Start (86) | Curve Start (Prs/Frc) (87) | Creating Curves Using the Curve Add Command | Creating Large Curves using the Curve Add Command | Curve Status Error Codes | Curve Delete (83) | Curve Delete All (85) | Curve Delete Except (84)

8.4.7.7. Command: Curve Delete (83)

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Data Type	Range
1	**Curve ID**	Internal: DINT External: REAL	0-50000
2	**Curve Count**	Internal: DINT External: REAL	>0

Description

This command deletes curves with IDs in the range of **Curve ID** to **Curve ID** + **Curve Count -1** will de deleted. Only curves that exist are deleted. An entry will be logged in the Event Log for each curve that is deleted. Attempting to delete non-existent curves will not cause an error.

After a curve is deleted, it cannot be used anymore and the space it occupied in the curve store is freed up. If a curve is deleted while it is being followed, it will not be interrupted and will continue to work properly, but cannot be used again after the curve completes.

See Also

Curve Delete All (85) | Curve Delete Except (84) | Curve Add (82) | Curves Overview

8.4.7.8. Command: Curve Delete Except (84)

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Data Type	Range
1	**Curve ID**	Internal: DINT External: REAL	0-50000
2	**Curve Count**	Internal: DINT External: REAL	>0

Description

This command deletes all the curves except for curves with IDs in the range of **Curve ID** to **Curve ID** + **Curve Count -1**. An entry will be logged in the Event Log for each curve that is deleted.

After a curve is deleted, it cannot be used anymore and the space it occupied in the curve store is freed up. If a curve is deleted while it is being followed, it will not be interrupted and will continue to work properly, but cannot be used again after the curve completes.

See Also

Curve Delete (83) | Curve Delete All (85) | Curve Add (82) | Curves Overview

8.4.7.9. Command: Curve Delete All (85)

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command deletes all the curves in the controller. An entry will be logged in the Event Log for each curve that is deleted. If a curve is deleted while it is being followed, it will not be interrupted and will continue to work properly, but cannot be used again after the curve completes.

This command will also cancel any curve being downloaded in parts. If this is undesirable, use Curve Delete (83) or Curve Delete Except (84) commands in a manner that ensures that all curve IDs being used have been deleted.

See Also

Curve Delete (83) | Curve Delete Except (84) | Curve Add (82) | Curves Overview

8.4.7.10. Command: Curve Start (86)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Data Type	Range
1	**Curve ID**	Internal: DINT External: REAL	0-50000
2	**Master Register** **Note:** See **Specifying a Register Address** below.	REAL	_Time or any other valid register of type REAL.
3	**Cycles** 0 = continuous, 1 = 1 cycle *n* = *n* cycles (up to 16 million)	REAL	0 to 16 million (0 = continuous)

Description

This command starts following the position curve with the specified **Curve ID**. Curves can be used for splines and profiles based on time, or for camming based on a master. For pressure or force axes, use the Curve Start (Prs/Frc) (87) command.

For more advanced options, such as scaling or offsetting the curve, or absolute and relative options for the master or curve alignment, see the Curve Start Advanced (88) command.

Curve ID

Specifies which curve you wish to follow.

Master Register

This specifies whether you wish to follow the curve based on time, or based on some master register, such as the position of another axis.

- **Time-Based**
  To follow a curve based on time, choose **_Time** as the master (see **Specifying a Register Address** below).

  If you wish to follow a curve based on time, but sometimes need to slow down or speed up the interpolation rate, consider using a virtual axis as a master. See the **Using a Virtual Axis as a Curve Master** section below.
- **Master-Based**
  To follow a curve based on a master, choose the address of the master register, for example, the position of another axis. When this command is issued, the Point 0 X value is ignored. The RMC uses the position of the master at the time this command is issued as the starting X axis position (X_0). All the x-axis points are then computed relative to this point. Therefore, if you want the curve to follow the master axis at the actual x-axis values in the data, you should move the master to the starting location before issuing the Curve Start command.

  The curve will run based on the master position, even if the master changes directions.

Cycles

Specifies the number of times to repeat the curve. The value of 1 is most common and will run the curve once. The value 0 means the curve will repeat endlessly. Any other value (*n*) means to run the curve *n* times. For curves with cubic interpolation that will be repeated multiple times (**Cycles** is not 1), the **Natural-Velocity Endpoints (+1)** Interpolation Option is not allowed.

If the y-values of the first and last points are not equal, and the curve is run for more than 1 consecutive cycle, the curve will be automatically offset so that the first point of the next cycle matches the last point of the previous cycle.

When running a curve for multiple cycles, if you wish the curve to repeat at a certain interval, then it is important that the curve x-values be defined for that entire interval, including the last x-value. For example, in order to repeat the curve at an x-axis interval starting at 0 and going to 360, the first x-value of the curve must be zero, and the last x-value of the curve must be 360. When running multiple cycles, the first and last points of the curve will end up being the same point in time. This means that if the first and last y-values are not equal to each other, each cycle of the curve will be offset from the previous one.

Starting a Curve

For the axis that you issue the Curve Start command to, the Target (Position, Pressure, or Force) *must* be at the first point in the curve data (Y_0) before issuing the command, or, a Transition command must previously have been issued to specify how the axis should move to get to the curve. A transition command will allow you to start a curve even though the axis is not at the curve yet. If you want the axis to follow the curve exactly starting from the beginning, do not use a transition; instead, make sure the axis is at the starting point before issuing the Curve Start command.

If you are following the curve based on a master (not time), you may want to first move the *master* to it's starting location. This is because the curve master positions (the curve's x-data) will be relative to the position of the master at the time this command is issued. The curve does not use the actual X_0 value in the curve data. Therefore, if you want the curve to follow the master axis at the actual x-axis values in the data, you should move the master to the starting location before issuing the Curve Start command.

Using a Virtual Axis as a Curve Master

A virtual axis can be used as a curve master. It is sometimes desirable to gear to a virtual axis rather than executing the curve as a function of time. All the curves using the virtual axis as a master can be sped up or slowed down by speeding up or slowing down the virtual axis. The virtual axis can even be moved backwards causing the curve axes to back up. This cannot be done using the _Time master register.

Moving the master virtual axis at 1 unit/sec as the standard velocity makes gearing ratio calculations very easy. The acceleration and deceleration of the master virtual axis provide a smooth start and stop for the curve axis.

When using a virtual axis as a master, it is often useful to set up the virtual axis as rotary because it will never need to be manually reset. The wrapping automatically resets the master. If using a rotary master, the curve must be defined as describe in the **Using a Rotary Master** section below.

Using a Rotary Master

When using a rotary axis (whether virtual or real), if you wish the curve to repeat for each rotation, then it is important that the curve x-values be defined for the entire rotary range, including the last x-value. For example, if the Position Unwind is 360, and the curve is for an entire rotation starting at zero, then the first x-value of the curve must be zero, and the last x-value of the curve must be 360.

Cycles Status

The Cycles and Cycles (Pressure/Force) status registers hold the number of whole cycles that have been completed. These registers are valid for the Curve Start (86), Curve Start (Prs/Frc) (87), Sine Start (72) and Sine Start (Prs/Frc) (76) commands.

Completing a Curve

Time-Based Curves

Time-based curves will end after the specified number of cycles, and the Target Generator Done bit will be set when the axis reaches the final point. Infinite curves (Cycles = 0) will continue until another motion command is issued to the axis. Infinite curves will not cause the Target Generator Done bit to turn on.

If the curve ends with a non-zero velocity or acceleration, a new motion command must be issued within two loop times after the Target Generator Done bit has been set. If a new command is not issued during this time, then a Closed Loop Halt will occur and the Runtime Error bit will be set. Notice that if the Runtime Error AutoStop is set to a halt, that halt will also occur. Use a Link Condition in a user program to check whether the Target Generator Done bit has been set. See the **Target Generator State Bits** section below.

Master-Based Curves

Master-based curves with zero-velocity endpoints will follow the curve until another motion command is issued. When the master register is outside the range of the curve, the curve axis will remain stopped at the appropriate endpoint. The Target Generator Done and State Bits can be used to determine when the master has moved beyond either endpoint. See the **Target Generator State Bits** section below.

Master-based curves that end with *non-zero* velocity endpoints will continue for two loop times after the master moves beyond the starting or ending point. If another motion command is not issued within two loop times of the master moving beyond the starting or ending point, then a Closed Loop Halt will occur and the Runtime Error bit will be set. Notice that if the Runtime Error AutoStop is set to a halt, that halt will also occur. Use a Link Condition in a user program to check the Target Generator bits to determine whether the master has moved beyond the endpoints. See the **Target Generator State Bits** section below.

If you do not want the axis to halt after exceeding an endpoint, use the Curve Start Advanced (88) command.

For master-based cyclic curves, the endpoints are the beginning of cycle 0, and the end of the last cycle. Infinite cyclic curves will continue until another motion command is issued to the axis.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

_Time:

The register address for the _Time tag in the RMC75 is %MD20.10. Therefore, the value is 20*4096 + 10 = 81930.

The register address for the _Time tag in the RMC150 is %MD44.10. Therefore, the value is 44*4096 + 10 = 180234.

The register address for the _Time tag in the RMC200 is %MD18.10. Therefore, the value is 18*4096 + 10 = 73738.

Status Registers

The Cycles axis status register gives the whole number of cycles that the current curve has completed. After a curve has completed or been interrupted, this will remain at its current value. This value is reset each time the Curve Start command is sent.

Status Bits

In Position Bit

For curves based on _Time, the In Position bit will be set when the curve has completed. The In Position bit will not be set for curves that follow a master other than _Time.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful for determining whether the curve has reached the starting or ending points.

Target Generator Done bit

Set when the master position is at or beyond the ending point. Notice that for master-based curves with a zero-velocity ending point, the curve is still being followed. That is, if the master backs up into the curve, then the curve axis will resume following the curve backwards, and the Target Generator Done bit will be cleared.

Target Generator State A and B bits

B	A	Description
0	0	The curve is being followed and the master is beyond (not at) the ending point.
0	1	The curve is being followed and the master is prior to (not at) the starting point.
1	0	The curve is being followed and the master is between the starting and ending points or at the starting or ending point.
1	1	Reserved.

Pri. TG SI Busy (Primary Target Generator Superimposed Busy) Bit

This bit will be set during the transition. The transition begins when the motion command is issued, not necessarily when the Transition command is issued. When the transition completes, this bit will clear.

See Also

Curves Overview | Curve Add (82) | Curve Data Formats | Curve Start (Prs/Frc) (87) | Curve Start Advanced (88) | Change Master (79)

8.4.7.11. Command: Curve Start Advanced (88)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Data Type	Range
1	**Curve ID**	Internal: DINT External: REAL	0-50000
2	**Master Register** **Note:** See **Specifying a Register Address** below.	REAL	_Time or any other valid register of type REAL.
3	**Cycles** 0 = continuous, 1 = 1 cycle *n* = *n* cycles (up to 16 million)	REAL	0 to 16 million (0 = continuous)
4	**Options, sum of the following:** **Curve Alignment, one of:** Absolute Curve Alignment (+0) Relative Curve Alignment (+1) **Master Alignment, one of:** Relative Master Alignment (+0) Absolute Master Alignment (+2) **Endpoint Behavior** **For Absolute Master alignment, one of:** Fault (+0) Truncate (+4) Extrapolate (+8) **For Relative Master alignment, one of:** Standard (+0) Truncate (+4) Truncate and End (+8)	Internal: DINT External: REAL	≥ 0, whole numbers
5	**Curve Scale**	REAL	any
6	**Curve Offset**	REAL	any
7	**Master Scale**	REAL	≠ 0
8	**Master Offset**	REAL	any
9	**Status Block** (address)	REAL	Address or **none** (0)

	Note: See **Specifying a Register Address** below.		

Description

This command is identical to the Curve Start (86) command, but with additional advanced options such as offsetting or scaling the y-axis of the curve, offsetting or scaling the curve master, choosing absolute or relative curve alignment or master alignment, and the inclusion of a status block for advanced status information.

Notice that the command parameters do not change the curve itself in the RMC. It only defines the instance of the curve that is started with this command.

This topic describes only the advanced features of this command. For basic information on using this command, see the Curve Start (86) command.

Absolute/Relative Curve Alignment

Curve alignment refers to the Y-values of the curve, which are the positions of the curve axis:

Absolute

With *Absolute* curve alignment, the positions will be exactly as defined by the curve. To choose absolute curve alignment, add +0 to the **Options** command parameter.

Relative

With *Relative* curve alignment, the Y-values will be relative to the Target Position of the curve axis at the time the Curve Start Advanced command is sent. That is, the Y-values in the Curve Data will be adjusted such that the Point 0 Y-value is set to the current position of the axis. All the other Y-value points will be adjusted by the same amount. To choose relative curve alignment, add +1 to the **Options** command parameter.

Transitions

When using absolute curve alignment, the axis must be at the correct point in the curve when the Curve Start Advanced command is issued. The correct point can be very difficult to determine if the master is not at the starting point, and therefore, the Transition Rate (56) command must be used to specify how the axis should get to the correct point when this command is issued.

Absolute/Relative Master Alignment

Master alignment refers to the X-values of the curve, which are the positions of the master:

Absolute

With *Absolute* master alignment, the x-values will be exactly as defined by the curve. To choose absolute master alignment, add +2 to the **Options** command parameter.

Absolute master alignment is intended for applications where the curve must occur at an exact location based on the position of the master, such as in camming.

Relative

With *Relative* master alignment, the X-values will be relative to the master value at the time the Curve Start Advanced command is sent. That is, the X-values in the Curve Data will be adjusted such that the Point 0 X-value is set to the current value of the master. All the other X-value points will be adjusted by the same amount. To choose relative master alignment, add +0 to the **Options** command parameter.

Relative master alignment is intended for applications where the starting master point of the curve may vary, which may be the case when the master is a conveyor belt or a feed chain, such as in curve sawing. The Endpoint Behavior options for a relative master are specifically designed for this type of use case.

Using _Time as an Absolute Master

The _Time register wraps every second through the range [0.000000, 1.000000). It can be used as a master for curves that are to run based on time. When using _Time as an absolute master, the _Time is assumed to be zero when the Curve Start Advanced command is issued. Using a

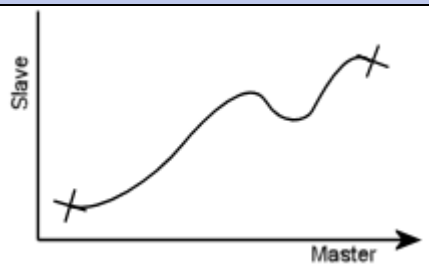
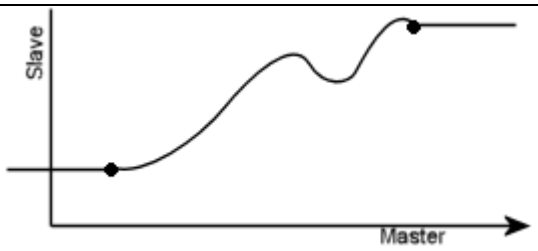
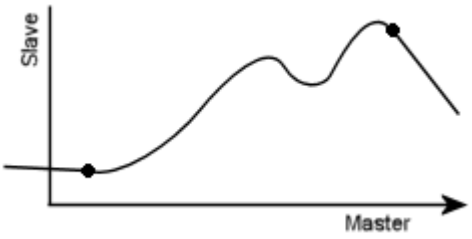
Master Offset will adjust the curve in time, effectively delaying the curve, or beginning at some point in the middle of the curve.

Endpoint Behavior

The Endpoint Behavior specifies what happens to the curve axis if the master moves outside of the X-value range defined by the curve data. For cyclic curves, the endpoints are the beginning of cycle 0, and the end of the last cycle. Infinite cyclic curves will continue until another motion command is issued to the axis.

Endpoint Behavior with Absolute Master Alignment

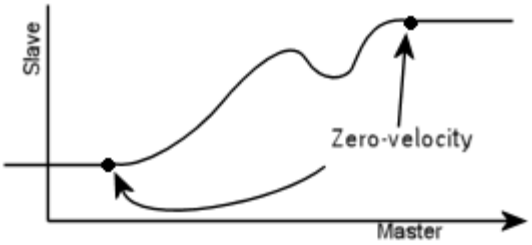
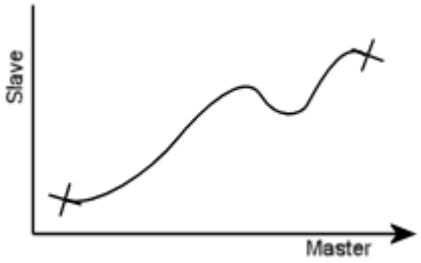
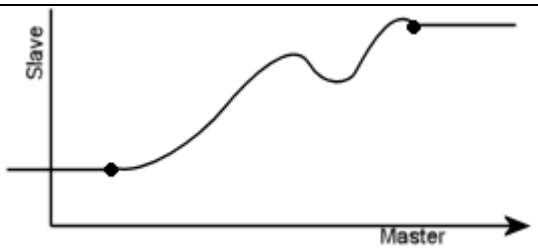
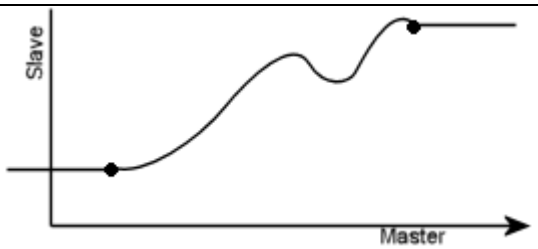
The Endpoint Behavior options listed below apply when Absolute Master Alignment (+2) is selected. The axis is assumed to be in the first cycle when this command is issued.

Option	Description	Image
Fault (+0)	If the master exceeds either endpoint, the curve axis' runtime error bit will be set which will halt the axis according to the Auto Stops setting for the Runtime error.	
Truncate (+4)	If the master moves past an endpoint, the curve axis' Target Position will stop at the endpoint. When the master moves back into the range, the curve will resume. If the master is moving quickly when it exceeds the endpoint, it may cause the curve axis to stop abruptly.	
Extrapolate (+8)	The curve will be extrapolated linearly, using the Target Velocity at the endpoint of the curve. When the master moves back into the range, the curve will resume.	

Endpoint Behavior with Relative Master Alignment

The Endpoint Behavior options listed below apply when Relative Master Alignment (+0) is selected.

Option	Description	Image
Standard (+0)	Curves with zero-velocity endpoints will follow the curve until the end is reached, or another motion command is issued. When the master register moves outside the range of the curve, the curve axis will remain stopped at the	Zero-velocity endpoints:

	appropriate endpoint, unless master re-enters the curve by reversing direction. Curves with Relative Master Alignment that end with *non-zero* velocity endpoints will continue for two loop times after the master moves beyond the starting or ending point. If another motion command is not issued within two loop times of the master moving beyond the starting or ending point, then a Closed Loop Halt will occur and the Runtime Error bit will be set. Notice that if the Runtime Error AutoStop is set to a halt, that halt will also occur. Use a Link Condition in a user program to check the Target Generator bits to determine whether the master has moved beyond the endpoints. See the **Target Generator State Bits** section below.	 Non-zero-velocity endpoints: 
Truncate (+4)	If the master moves past the beginning or end, the curve axis' Target Position will stop at that point. When the master moves back into the range, the curve will resume. If the master is moving quickly when it exceeds the endpoint, it may cause the curve axis to stop abruptly.	
Truncate and End (+8)	If the master moves past the end, the curve axis' Target Position will stop at the endpoint and the curve will stop executing. If the master moves back into the range, the curve will not resume. Notice that if the master moves past the beginning point, the curve will still continue executing when the master moves back in to the range. If the velocity at the endpoint is non-zero, the	

	velocity will abruptly go to zero.	

Curve Scale and Offset

The **Curve Scale** and **Curve Offset** parameters operate on the Y-values of the curve. These parameters can be used to shift the curve up or down in the y-axis, or compress or expand the curve in the y-axis. For each Y-value of the curve, the new Y-value is calculated as follows:

$Y_{new} = (Y_{original} \times \text{Curve Scale}) + \text{Curve Offset}$

The Curve Offset applies only to absolute curve alignment. The Curve Scale applies to both absolute and relative curve alignment.

Master Scale and Offset

The **Master Scale** and **Master Offset** parameters scale or shift the curve master. The X value used to index into the curve is found using the following formula:

$X = (\text{Master Value} + \text{Master Offset}) \times \text{Master Scale}$

This can also be viewed as shifting and expanding or contracting the curve itself along the X axis. From that perspective, the X value for each point is calculated as follows:

$X_{new} = (X_{original} \div \text{Master Scale}) - \text{Master Offset}$

The Master Offset applies only to absolute master alignment. The Master Scale applies to both absolute and relative master alignment.

Status Block

Advanced users may wish to use the Curve Start Advanced command's **Status Block**, which provides read-only information on the curve motion. This information is most useful when manipulating the curve in user programs.

To use the Status Block, you must specify an address from the Variable Table in the **Status Block** parameter of the Curve Start Advanced command. The Status Block will require six registers in the Variable Table, beginning with the specified address. As the curve motion progresses, the selected registers in the Variable Table will be continuously updated. The selected variables will not be named automatically; you should give a descriptive name to each to help you keep track of them.

To prevent confusion, curves that are running simultaneously should not use the same Status Block address. Non-simultaneous curves can use the same Status Block address.

The Status Block provides the following information:

Status Block Offset	Name	Data Type	Description
0	Current Cycle Count	REAL	The whole number of cycles of the curve that have been completed. For continuous curves (without a fixed number of cycles), this value will wrap to zero after it reaches 10,000,000 and then continue incrementing. For curves with a fixed number of cycles, this value will not go beyond the requested cycle count.
1	Current Index	REAL	Indicates the current X value being used for interpolating through the curve. This value is in the original X scale and offset with which the curve was defined. This value will wrap for each cycle.
2	Current Curve Scale	REAL	The current Curve Scale of the curve.

3	Current Curve Offset	REAL	The current Curve Offset of the curve. This value is calculated automatically for the Relative Curve Alignment option.
4	Current Master Scale	REAL	The current Master Scale of the curve.
5	Current Master Offset	REAL	The current Master Offset of the curve. This value is calculated automatically for the Relative Master Alignment option.

Running a Curve Backwards

To run a curve backwards with a _Time master, use Absolute Master Alignment, a Master Scale of -1, and a Master Offset value that is equal to the X value of the last point of the curve.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** and **Status Block** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

_Time:

The register address for the _Time tag in the RMC75 is %MD20.10. Therefore, the value is 20*4096 + 10 = 81930.

The register address for the _Time tag in the RMC150 is %MD44.10. Therefore, the value is 44*4096 + 10 = 180234.

The register address for the _Time tag in the RMC200 is %MD18.10. Therefore, the value is 18*4096 + 10 = 73738.

Status Registers

The Cycles axis status register gives the whole number of cycles that the current curve has completed. After a curve has completed or been interrupted, this will remain at its current value. This value is reset each time the Curve Start Advanced command is sent.

Status Bits

In Position Bit

The In Position bit will be set only in the following cases:

- For curves using _Time as the master register, the In Position bit will be set when the curve has completed.
- For curves with a relative master alignment, and the **Truncate and End (+8)** option, the In Position bit will be set when the master has moved beyond the ending point of the curve.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful for determining whether the curve has reached the starting or ending points.

Target Generator Done bit

Set when the master position is at or beyond the ending point. Notice that for master-based curves with a zero-velocity ending point, the curve is still being followed (unless the **Truncate and End** option has been selected and the master is beyond the ending point). That is, if the master backs up into the curve, then the curve axis will resume following the curve backwards, and the Target Generator Done bit will be cleared.

Target Generator State A and B bits

B	A	Description
0	0	The master is beyond (not at) the ending point.
0	1	The master is prior to (not at) the starting point.
1	0	The master is between the starting and ending points or at the starting or ending point.
1	1	Reserved.

Pri. TG SI Busy (Primary Target Generator Superimposed Busy) Bit

This bit will be set during the transition. The transition begins when the motion command is issued, not necessarily when the Transition command is issued. When the transition completes, this bit will clear.

See Also

Curves Overview | Curve Start (86) | Curve Start Advanced (Prs/Frc) (89) | Curve Data Formats | Change Master (79)

8.4.8. Velocity

8.4.8.1. Command: Move Velocity (37)

Supported Axes:	Position or Velocity Control Axes
Supported Control Modes:	Position PID, Position I-PD, Velocity PID, Velocity I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Speed** (position-units/s)	any
2	**Acceleration Rate** (position-units/s^2)	>0
3	**Deceleration Rate** (position-units/s^2)	>0
4	**Direction** • **Positive (1)**: Moves at a positive velocity. **Speed** must be positive. • **Negative (-1)**: Moves at a negative velocity. **Speed** must be positive. • **Current (2)**: Moves at the current Target velocity. If the Target Velocity is zero when the command is	a valid integer as described

	issued, it will move with a positive velocity. **Speed** must be positive. • **Signed (4)**: The direction of the move is determined by the sign of the **Speed** command parameter.	

Description

This command ramps the axis velocity to the **Requested Speed**. The velocity is ramped at the rates specified by the **Acceleration Rate** and **Deceleration Rate**. The **Acceleration Rate** is used when ramping the speed away from zero and **Deceleration Rate** is used when ramping the speed toward zero.

The behavior of the motion depends on the control mode:

- **Position PID**, **Position I-PD**

  In these modes, the Move Velocity performs **Continuous Position Control**. The Move Velocity command does not actually generate a Target Velocity. Instead, it generates a moving Target Position. The Target Position will move at the specified rate until commanded otherwise. Therefore, if the axis falls behind, it will not try to catch up to the Target Velocity; it will actually attempt to catch up to the Target Position. This may cause the axis to move considerably faster than the Target Velocity while it catches up.

  In these control modes, the Move Velocity command works well for applications where it is important that the Target Position is always moving at a certain rate. If it is very important that the velocity remain constant, and the position itself is not so important, you may wish to use the Velocity PID control mode instead.

- **Velocity PID**, **Velocity I-PD**

  In these modes, the Move Velocity command ramps the Target Velocity to the **Requested Speed**. The axis will move at that speed until commanded otherwise.

Special Notes

Move Velocity vs. Move Absolute on a Position Axis

On a position axis, the Move Velocity command is sometimes used unnecessarily. The user perhaps assumes the Move Velocity command is the best choice because it moves the axis at a certain speed. However, the Move Absolute (20) command will also move the axis at a specific speed, with the advantage that it stops at the requested position. The constant speed control portion of the move is identical for both of these commands.

An example is an injection molding press. When the press is injecting, the speed should be constant, until it reaches the stopping position. In this case, the Move Absolute (20) is a better choice than the Move Velocity command.

As rule of thumb, use the Move Velocity command when you want the axis to move at a certain velocity indefinitely. If you want the axis to eventually stop at some position, use the Move Absolute command.

Target Generator State Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Velocity has reached the **Requested Speed**.

Target Generator State A and B bits

B	A	Description
0	0	Stopped
0	1	Accelerating (away from zero velocity)

1	0	Constant Velocity
1	1	Decelerating (toward zero velocity)

See Also

List of Commands | Commands Overview

8.4.8.2. Command: Move Velocity (I-PD) (38)

Supported Axes:	Position or Velocity Control Axes
Supported Control Modes:	Velocity I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Speed** (position-units/s)	any
2	**Direction** • **Positive (1)**: Moves at a positive velocity. **Speed** must be positive. • **Negative (-1)**: Moves at a negative velocity. **Speed** must be positive. • **Current (2)**: Moves at the current Target velocity. If the Target Velocity is zero when the command is issued, it will move with a positive velocity. **Speed** must be positive. • **Signed (4)**: The direction of the move is determined by the sign of the **Speed** command parameter.	a valid integer as described

Description

This command is an advanced command. Do not use it unless you specifically intend to use the Velocity I-PD control mode. If you want to make a basic velocity move, use the Move Velocity (37) command instead.

This command moves the axis at the **Requested Speed**. When this command is issued, the Target Velocity is immediately set to the **Requested Speed**. The axis will automatically be switched to the Velocity I-PD control mode during the motion, and will remain in the Velocity I-PD control mode until another command is issued to the axis. However, if a closed-loop halt occurs on the axis during the motion, the axis will remain in the Velocity I-PD control mode.

See Also

Velocity Control | Velocity I-PD | Move Velocity (37) | List of Commands | Commands Overview

8.4.9. Transitions

8.4.9.1. Command: Transition Disable (55)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command disables transitions on a position axis. When transitions are disabled, issuing any command that requires a transition will cause a command error, which will halt the axis if the AutoStops are set to do so.

Transitions are useful for starting certain types motion even though the axis is not at the correct starting point. For example, the Curve Start (86), Sine Start (72), and Gear Absolute (25) commands normally require that the axis be at the correct starting point. However, if a transition has been enabled (by issuing the Transition Rate (56) command), then these commands can be issued even though the axis is not at the correct starting point. The axis will transition in the manner requested by the current transition mode. See the Transition Rate (56) command for details.

This command will not affect any transitions that are in progress. When the RMC powers up, transitions are disabled on all axes. To enable transitions, issue the Transition Rate (56) command.

See Also

Transition Rate (56)

8.4.9.2. Command: Transition Rate (56)

Supported Axes:	Position Control Axes
Supported Control Modes:	Position PID, Position I-PD

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Max Speed** (pu/s)	Any REAL number.
2	**Accel Rate** (pu/s^2)	Any REAL number.
3	**Transition Type** • Seek (0) • Reach (1) • Superimposed (2)	A valid integer as described.

4	**Direction** • Negative* (-1) • Nearest (0) • Positive* (1) • Current* (2) • Absolute* (3) * These options are intended for use with rotary axes. However, all options are available on linear axes, but have no effect. See the **Rotary Axes** section below.	a valid integer as described

Description

This command enables position axis transitions and defines the speed and acceleration of the transition. This command does not start any motion. Rather, it will apply when certain motion commands are issued to the axis. If you need transitions on a pressure/force axis, see the Transition Rate (Prs/Frc) (64) command.

Transitions are useful for starting certain types of motion even though the axis is not at the correct starting point. For example, the Sine Start (72), Curve Start (86), Curve Start Advanced (88) and Gear Absolute (25) commands normally require that the axis be at the correct starting point. However, if a transition has been enabled, then these commands can be issued even though the axis is not at the correct starting point. When the motion command is issued, the axis will move toward the requested profile (curve, sine wave, gearing relationship, etc.) as defined by the transition command.

When the RMC powers up, transitions are disabled on all axes. To enable position axis transitions, issue the Transition Rate (56) command. Once this command has been issued, it does not need to be issued again, unless you wish to specify a different transition, or if you need to re-enable transitions after disabling transitions. To disable position transitions, issue the Transition Disable (55) command.

This command will not affect any transitions that are in progress.

Transition Types

This command provides the following transition options:

- **Seek (0)**
  The axis will move toward the requested profile using the **Max Speed** and **Accel Rate**. When the position and velocity of the axis come close to the position and velocity of the profile, the axis will "lock" onto the profile. Use this option to get to the requested profile quickly and smoothly.
- **Reach (1)**
  The axis will move toward the requested profile using the **Max Speed** and **Accel Rate**. When the position reaches the position of the profile, the axis will "lock" onto the profile. Notice that this option does not require that the velocities be close when it locks on, and therefore may cause the axis to jerk. Use this option to get to the requested profile as quickly as possible.
- **Superimposed (2)**
  A trapezoidal or S-curve move using the **Max Speed** and **Accel Rate** will be superimposed onto the requested profile such that the axis will reach the profile. Notice that since the move is superimposed onto the profile, the axis will not necessarily move at the specified speed and acceleration, but rather at the sum of the speeds and accelerations from the requested profile and the superimposed portions of the move.

  This method will always guarantee that the axis will lock on to the requested profile, even if the **Max Speed** and **Accel Rate** are slower than that of the profile. The time it takes to lock on will be based on how far the axis position is from the requested profile and on the **Max Speed**. For example, if the axis is at 3, the profile is at 9, and the Max Speed is 6, it will take roughly one second to lock on. The "lock-on" of the Superimposed method will be as smooth or smoother than **Seek**.

This superimposed move will be trapezoidal if the Requested Jerk axis parameter is zero, or S-curve if the Requested Jerk is non-zero. See the Requested Jerk topic for details.

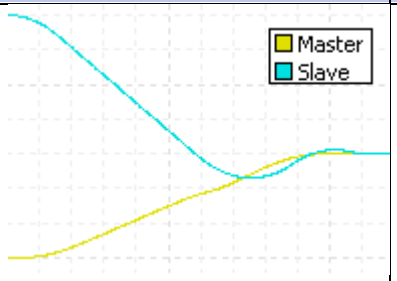
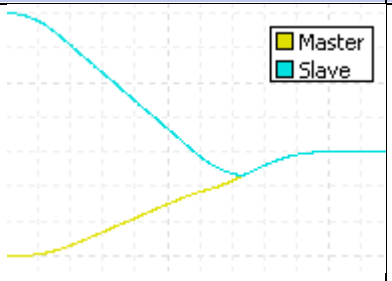
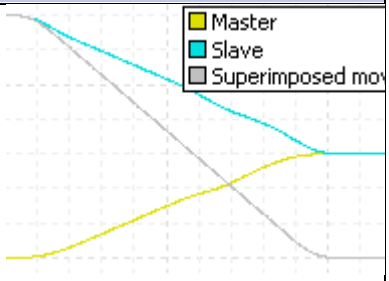
Choosing a Transition Type

In general, try the **Seek** method first. If it takes too long to lock on, switch to **Reach**. For either of these methods, make sure to set the **Max Speed** and **Accel Rate** to values higher than that of the profile it is trying to follow. Otherwise, the axis may never catch up to the profile. Notice that if the master register is stopped, then all 3 methods will perform similarly.

If the transition is used for a gearing application, the behavior of any of the methods will be the same if the master is not moving while the transition is taking place. If the master is moving, and is a well-behaved Target Position, then the **Seek** and **Reach** methods will both behave very similarly. If the master is moving, but is noisy, such as an analog reference signal, then the **Reach** method will "lock in" the quickest, but may cause jerk.

The **Superimposed** method is not as useful as the first two methods, but it can provide a more predictable "lock-on", as described above. Also, this method will always guarantee that the axis will lock on to the requested profile, even if the **Max Speed** and **Accel Rate** are slower than that of the profile.

Shown below is an example of how the various options work for one sample profile. Notice that the behavior will vary for other profiles.

Seek	Reach	Superimposed
		
The Seek method waits until the position and velocity are close before locking in. If the master was not accelerating or decelerating, it would lock in quicker.	The Reach method locks in as soon as the position reaches the master. This can cause a jerk in the Target Position of the slave axis.	The gray line shows the move that is superimposed onto the master, and results in the slave target.

Rotary Axes

Transitions fully support rotary axes. The Direction options listed below define the direction in which the axis should move to reach the desired starting position. The starting position is determined at the time a motion command requiring the transition is issued.

For the Negative, Nearest, Positive, and Current options, if the starting position value is outside the valid range of the rotary axis, the starting position will be set within the valid range using modulo arithmetic such that the position will be the same location within the range.

- **Negative (-1)**
  The axis will move in the negative direction to reach the starting position.
- **Nearest (0)**
  The axis will move in the direction which in which the starting position is nearest to the current Target Position.
- **Positive (1)**
  The axis will move in the positive direction to reach the starting position.
- **Current (2)**
  The axis will move in the direction of the current Target Velocity to reach the starting position. If the current Target Velocity is zero, it will behave as the Nearest option.

- **Absolute (3)**
  The starting position is treated as a position on a linear axis; the axis begins moving toward the starting position as if on a linear scale. If the position is outside of the valid position range, the axis rotates through the number of revolutions required to reach the position. Each time the Target Position wraps during the move, the Position Unwind value is subtracted from the Command Position until the Command Position is within the valid position range.

For more details and examples, see the Rotary Motion topic.

Status Bits

Pri. TG SI Busy (Primary Target Generator Superimposed Busy) Bit

This bit will be set when the transition begins. Notice that this is when the motion command is issued, not necessarily when the Transition rate command is issued. When the transition actually takes place, this bit will be set until the axis "locks on".

See Also

Transition Disable (55) | Curve Start (Prs/Frc) (87) | Sine Start (Prs/Frc) (76) | Gear Absolute (25)

8.5. Pressure/Force Control

8.5.1. Standard

8.5.1.1. Command: Hold Current Pressure/Force (19)

Supported Axes:	Pressure/Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Integrator Preload** (%)	-100% to 100%

Description

This command enters pressure or force control and holds the current Actual Pressure/Force. That is, the Command Pressure/Force and the Target Pressure/Force will be set to the Actual Pressure/Force, even if the axis is already in pressure/force control.

If the axis is in Pressure/Force Limit when this command is issued, the axis will switch to pressure/force control. Pressure/force limit is not active when the axis is in pressure/force control.

If the axis' pressure/force is changing quickly at the time you want to enter pressure/force control, you may want to use the Enter Pressure/Force Control (Auto) (44) or Enter Pressure/Force Control (Time) (45) commands instead.

The **Integrator Preload** specifies the value that the Integral Output Term should be set to when transitioning into pressure/force control. The Preload is specified as a percentage of the Output Scale (typically 10V or 100%), and is not affected by the pre-transition integrator state. Typically,

this can be left at the default value of 0. If the pressure/force exhibits a drop upon entering pressure/force, increase this value to help smooth the transition.

Compare to Stop Pressure/Force (43) Command

The Stop Pressure/Force (43) command will stop the current Target Pressure or Force immediately or with a ramp, and will not affect the state of pressure control on the axis. The Hold Current Pressure/Force command will immediately hold the current *Actual* Pressure or Force, and *will* put the axis in pressure or force control if it is not already.

Exiting Pressure/Force Control

To exit pressure/force control, send any open-loop command or closed-loop position command to the axis.

Status Bits

At Pressure/Force

If the Actual Pressure/Force is within the At Pressure/Force Tolerance window from the Target Pressure/Force, the At Pressure/Force Status bit will be set. This bit indicates that the axis is at the pressure or force.

Pressure/Force Control

The Pressure/Force Control Status bit indicates that the axis is in closed-loop pressure or force control.

Pressure/Force Target Generator Bits

The Pressure/Force Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Pressure/Force Target Generator Done bit

This bit indicates that the Target Pressure/Force has reached the **Requested Pressure/Force**. This bit will be set immediately when this command is issued.

Pressure/Force Target Generator State A and B bits

The Target Generator State A and B bits will always be zero during this command.

See Also

Stop Pressure/Force (43) | Enter Pressure/Force Control (Auto) (44) | Enter Pressure/Force Control (Time) (45) | Enter Pressure/Force Control (Rate) (46) | Ramp Pressure/Force (S-Curve) (41) | Ramp Pressure/Force (Linear) (42) | Pressure/Force Control Overview | List of Commands | Commands Overview

8.5.1.2. Command: Stop Pressure/Force (43)

Supported Axes:	Pressure or Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Pressure/Force Accel Rate** (Pr or Fr)	≥ 0

Description

This command stops the Pressure or Force target. The Target Pressure/Force will come to a stop at the rate specified by the **Pressure/Force Accel Rate** and the Command Pressure/Force will be set to the target. If the **Accel Rate** is zero, the Target Pressure/Force will stop immediately.

This command will not affect the state of pressure or force control. That is, if the axis is in pressure or force control, or pressure or force limit, this command will not take it out of that state, nor will it put it into pressure or force control if it was not in pressure or force control.

Compare to Hold Current Pressure/Force (19) Command

The Hold Current Pressure/Force (19) command will immediately hold the current Actual Pressure or Force, and will put the axis in pressure or force control if is not already. The Stop Pressure/Force command will stop the current *Target* Pressure or Force immediately or with a ramp, and will *not* affect the state of pressure control on the axis.

Status Bits

At Pressure/Force

This bit will not be set.

Pressure/Force Control

The Pressure/Force Control Status bit indicates that the axis is in closed-loop pressure or force control.

Pressure/Force Limit Enabled

This bit indicates that pressure/force limit is enabled. If an axis is in pressure or force control, pressure/force limit will not be active. If the axis exits pressure/force control to open loop or closed-loop position or velocity control, it will enter pressure/force limit if the Pressure/Force Limit Enabled bit is on.

Pressure/Force Limited

This bit indicates that pressure/force limit is enabled and the axis is limiting the pressure/force. If an axis is in pressure or force control, pressure/force limit will not be active and this bit will not be set.

Pressure/Force Target Generator Bits

The Pressure/Force Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Pressure/Force Target Generator Done bit

This bit indicates the move is complete, which occurs when the Target Pressure/Force has stopped.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	Done
0	1	Reserved
1	0	Reserved
1	1	Decelerating to zero velocity

See Also

Hold Current Pressure/Force (19) | Enter Pressure/Force Control (Auto) (44) | Enter Pressure/Force Control (Time) (45) | Enter Pressure/Force Control (Rate) (46) | Ramp Pressure/Force (S-Curve) (41) | Ramp Pressure/Force (Linear) (42) | Pressure/Force Control Overview | List of Commands | Commands Overview

8.5.1.3. Command: Ramp Pressure/Force (Rate) (18)

Supported Axes:	Pressure/Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Pressure/Force** (Pr or Fr)	any
2	**Pressure/Force Rate** (Pr/s or Fr/s)	≥ 0
3	**Pressure/Force Accel Rate** (Pr/s^2 or Fr/s^2)	≥ 0

Description

This command ramps the **Target Pressure** or **Force** to the requested **Pressure** or **Force** at the rate specified by the **Pressure/Force Rate** parameter. It will accelerate to that rate as specified by the **Pressure/Force Accel Rate**. This command will not put the axis in pressure/force control if it is not already in pressure/force control.

If the **Pressure/Force Accel Rate** is zero, it will mean infinite acceleration and the Pressure/Force will simply ramp linearly using the specified **Pressure/Force Rate**.

The Pressure/Force can also be ramped with other commands:

- Ramp Pressure/Force (S-Curve) (41)
- Ramp Pressure/Force (Linear) (42)

For more details, see the pressure/force control or pressure/force limit topics.

Status Bits

At Pressure/Force

When the axis is in Pressure/Force control or Pressure/Force Limit and the Target Pressure/Force reaches the **Requested Pressure/Force** and the Actual Pressure/Force is within the At Pressure/Force Tolerance window from the Target Pressure/Force, the At Pressure/Force Status bit will be set. This bit indicates that the ramp is complete and the axis is at the pressure or force.

Pressure/Force Control

The Pressure/Force Control Status bit indicates that the axis is in closed-loop pressure or force control.

Pressure/Force Limit Enabled

This bit indicates that pressure/force limit is enabled. If an axis is in pressure or force control, pressure/force limit will not be active. If the axis exits pressure/force control to open loop or closed-loop position or velocity control, it will enter pressure/force limit if the Pressure/Force Limit Enabled bit is on.

Pressure/Force Limited

This bit indicates that pressure/force limit is enabled and the axis is limiting the pressure/force. If an axis is in pressure or force control, pressure/force limit will not be active and this bit will not be set.

Pressure/Force Target Generator Bits

The Pressure/Force Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Pressure/Force Target Generator Done bit

This bit indicates that the Target Pressure or Target Force has reached the **Requested Pressure/Force**. If the ramp is interrupted, e.g. due to a halt, the done bit will not be set

because the commanded motion was not completed. Notice that this bit does not indicate whether the Actual Pressure or Actual Force has reached the **Requested Pressure/Force**.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	Pressure/Force is stopped (done)
0	1	Pressure/Force is accelerating
1	0	Pressure/Force is changing at a constant rate
1	1	Pressure/Force is decelerating

See Also

Ramp Pressure/Force (S-Curve) (41) | Ramp Pressure/Force (Linear) (42) | Stop Pressure/Force (43) | Enter Pressure/Force Control (Auto) (44) | Enter Pressure/Force Control (Time) (45) | Enter Pressure/Force Control (Rate) (46) | Pressure/Force Control Overview | List of Commands | Commands Overview

8.5.1.4. Command: Ramp Pressure/Force (S-Curve) (41)

Supported Axes: Pressure or Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Pressure/Force** (Pr or Fr)	any
2	**Ramp Time** (sec)	≥0

Description

This command ramps the Target Pressure/Force to the **Requested Pressure/Force** in the time specified by the **Ramp Time** parameter. The ramp is an s-curve generated by a 5th-order polynomial. This command will not put the axis in pressure/force control if it is not already in pressure/force control.

If the pressure or force is changing when this command is issued, and the **Requested Pressure/Force** requires that the pressure/ force must change direction, it is possible that the pressure/force may reach a large value before turning around, especially if the **Ramp Time** is large.

This command is not intended to be used if the Target Pressure is currently ramping. This will cause a discontinuity in the Target Force Rate. The Ramp Pressure/Force (Rate) (18) command is better suited for cases where the Target Pressure is currently ramping.

The Pressure/Force can also be ramped with other commands:

- Ramp Pressure/Force (Rate) (18)
- Ramp Pressure/Force (Linear) (42)

For more details, see the pressure/force control or pressure/force limit topics.

Status Bits

At Pressure/Force

When the axis is in Pressure/Force control or Pressure/Force Limit and the Target Pressure/Force reaches the **Requested Pressure/Force** and the Actual Pressure/Force is within the At Pressure/Force Tolerance window from the Target Pressure/Force, the At Pressure/Force Status bit will be set. This bit indicates that the ramp is complete and the axis is at the pressure or force.

Pressure/Force Control

The Pressure/Force Control Status bit indicates that the axis is in closed-loop pressure or force control.

Pressure/Force Limit Enabled

This bit indicates that pressure/force limit is enabled. If an axis is in pressure or force control, pressure/force limit will not be active. If the axis exits pressure/force control to open loop or closed-loop position or velocity control, it will enter pressure/force limit if the Pressure/Force Limit Enabled bit is on.

Pressure/Force Limited

This bit indicates that pressure/force limit is enabled and the axis is limiting the pressure/force. If an axis is in pressure or force control, pressure/force limit will not be active and this bit will not be set.

Pressure/Force Target Generator Bits

The Pressure/Force Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Pressure/Force Target Generator Done bit

This bit indicates that the Target Pressure or Target Force has reached the **Requested Pressure/Force**. If the ramp is interrupted, e.g. due to a halt, the done bit will not be set because the commanded motion was not completed. Notice that this bit does not indicate whether the Actual Pressure or Actual Force has reached the **Requested Pressure/Force**.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	Pressure/Force is stopped (done)
0	1	Pressure/Force is increasing
1	0	Reserved
1	1	Pressure/Force is decreasing

See Also

Ramp Pressure/Force (Linear) (42) | Ramp Pressure/Force (Rate) (18) | Stop Pressure/Force (43) | Enter Pressure/Force Control (Auto) (44) | Enter Pressure/Force Control (Time) (45) | Enter Pressure/Force Control (Rate) (46) | Pressure/Force Control Overview | List of Commands | Commands Overview

8.5.1.5. Command: Ramp Pressure/Force (Linear) (42)

Supported Axes:	Pressure or Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range

1	**Requested Pressure/Force** (Pr or Fr)	any
2	**Ramp Time** (sec)	≥ 0

Description

This command linearly ramps the **Target Pressure** or **Force** to the requested **Pressure** or **Force** in the time specified by the **Time for Move** parameter. This command will not put the axis in pressure/force control if it is not already in pressure/force control.

The Pressure/Force can also be ramped with other commands:

- Ramp Pressure/Force (Rate) (18)
- Ramp Pressure/Force (S-Curve) (41)

For more details, see the pressure/force control or pressure/force limit topics.

Status Bits

At Pressure/Force

When the axis is in Pressure/Force control or Pressure/Force Limit and the Target Pressure/Force reaches the **Requested Pressure/Force** and the Actual Pressure/Force is within the At Pressure/Force Tolerance window from the Target Pressure/Force, the At Pressure/Force Status bit will be set. This bit indicates that the ramp is complete and the axis is at the pressure or force.

Pressure/Force Control

The Pressure/Force Control Status bit indicates that the axis is in closed-loop pressure or force control.

Pressure/Force Limit Enabled

This bit indicates that pressure/force limit is enabled. If an axis is in pressure/force control, pressure/force limit will not be active. If the axis exits pressure/force control to open loop or closed-loop position or velocity control, it will enter pressure/force limit if the Pressure/Force Limit Enabled bit is on.

Pressure/Force Limited

This bit indicates that pressure/force limit is enabled and the axis is limiting the pressure/force. If an axis is in pressure or force control, pressure/force limit will not be active and this bit will not be set.

Pressure/Force Target Generator Bits

The Pressure/Force Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Pressure/Force Target Generator Done bit

This bit indicates that the Target Pressure/Force has reached the **Requested Pressure/Force**. If the ramp is interrupted, e.g. due to a halt, the done bit will not be set because the commanded motion was not completed. Notice that this bit does not indicate whether the Actual Pressure/Force has reached the **Requested Pressure/Force**.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	Pressure/Force is stopped (done)
0	1	Pressure/Force is increasing
1	0	Reserved
1	1	Pressure/Force is decreasing

See Also

Ramp Pressure/Force (S-Curve) (41) | Ramp Pressure/Force (Rate) (18) | Stop Pressure/Force (43) | Enter Pressure/Force Control (Auto) (44) | Enter Pressure/Force Control (Time) (45) | Enter

Pressure/Force Control (Rate) (46) | Pressure/Force Control Overview | List of Commands | Commands Overview

8.5.1.6. Command: Enter Pressure/Force Control (Auto) (44)

Supported Axes: Pressure/Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Pressure/Force** (Pr or Fr)	any
2	**Ramp Type** • Linear (0) • S-curve (1)	A valid integer as described
3	**Integrator Preload** (%)	-100% to 100%

Description

This command enters pressure or force control and then ramps the pressure or force to the **Requested Pressure/Force** in the manner specified by the **Ramp Type**. The rate of the ramp is automatically determined based on the Actual Pressure Rate or Actual Force Rate at the time this command is issued.

This command is especially designed for transitioning from position or velocity control to pressure or force control while the axis is moving.

A Linear **Ramp Type** will ramp the pressure or force linearly until it reaches the **Requested Pressure/Force**. An S-curve **Ramp Type** will provide a half s-curve ramp to the **Requested Pressure/Force**.

If the current Pressure/Force is stopped or is moving away from the **Requested Pressure/Force** when this command is issued, the Target Pressure/Force will immediately be set to the **Requested Pressure/Force**. This may cause the system to jerk.

Notice that if the Actual Pressure/Force Rate is very small when this command is issued, a very slow ramp may be calculated. If this is the case, and you do not want a slow ramp, consider using the Enter Pressure/Force Control (Time) (45) instead, which will make sure the axis reaches the **Requested Pressure/Force** within a specified time. Another option is to use the Hold Current Pressure/Force (19), which will enter pressure/force control and hold the current Actual Pressure/Force.

The **Integrator Preload** specifies the value that the Integral Output Term should be set to when transitioning into pressure/force control. The Preload is specified as a percentage of the Output Scale (typically 10V), and is not affected by the pre-transition integrator state. Typically, this can be left at the default value of 0. If the pressure/force exhibits a drop upon entering pressure/force, increase this value to help smooth the transition.

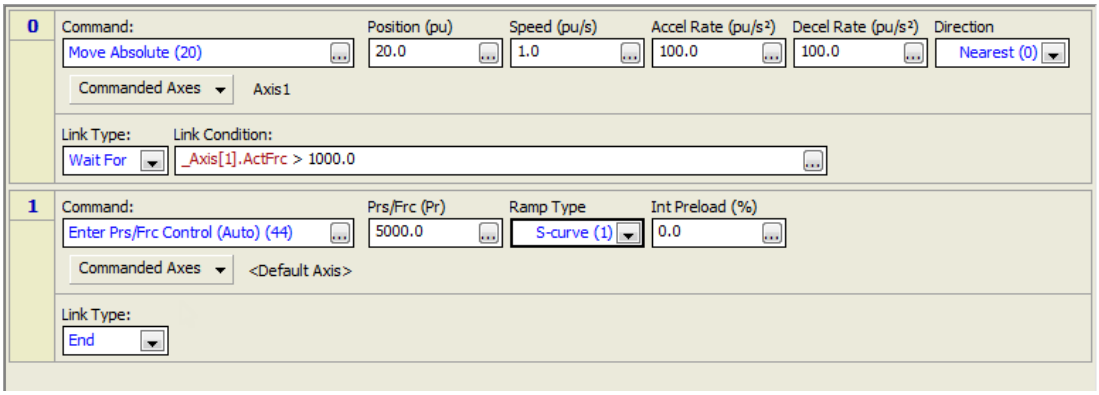
Example

Consider a hydraulic press that must move toward the material to be pressed, transition to force control and hold a certain force.

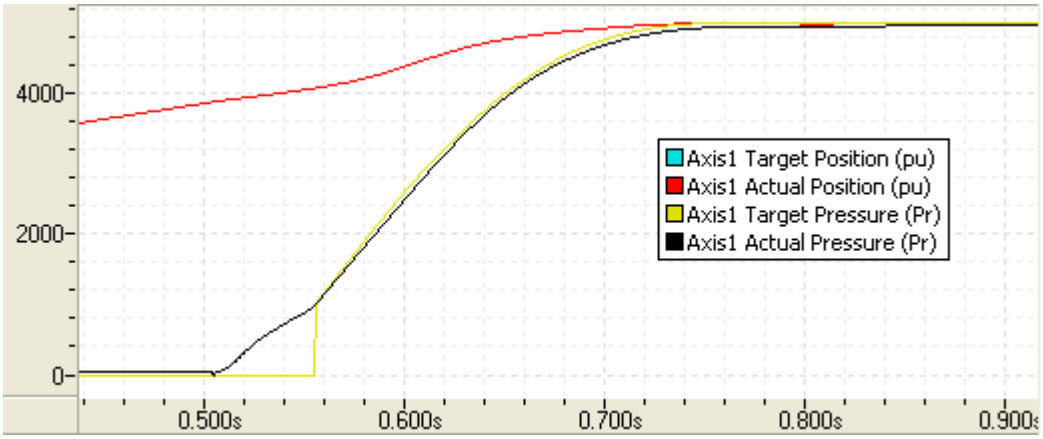
The user program below shows a possible sequence of steps.

Step 0 issues a Move Absolute command to move the axis to a position beyond where it expects to encounter the material. The Step 0 Link Type waits for the Actual Force to exceed 1000, then it

goes to Step 1 which issues an Enter Pressure/Force Control (Auto) command with an S-Curve Ramp Type. The axis will smoothly transition to force control and go to the Requested Pressure/Force of 5000.



The plot below shows the transition:



At the beginning of the plot, the position is increasing. At 0.5 seconds, the Actual Pressure starts increasing. When the Actual Pressure reaches 1000 at approximately 0.56 seconds, the Enter Pressure/Force Control command is issued and the axis smoothly transitions to pressure control, and the Target Pressure ramps to 5000.

Exiting Pressure/Force Control

To exit pressure/force control, send any open-loop command or closed-loop position command to the axis.

Status Bits

At Pressure/Force

When the axis is in Pressure or Force control and the Target Pressure or Force reaches the **Requested Pressure/Force** and the Actual Pressure or Actual Force is within the At Pressure/Force Tolerance window from the Target Pressure/Force, the At Pressure/Force Status bit will be set. This bit indicates that the ramp is complete and the axis is at the pressure or force.

Pressure/Force Control

The Pressure/Force Control Status bit indicates that the axis is in closed-loop pressure or force control.

Pressure/Force Target Generator Bits

The Pressure/Force Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Pressure/Force Target Generator Done bit

This bit indicates that the Target Pressure or Target Force has reached the **Requested Pressure/Force**. If the ramp is interrupted, e.g. due to a halt, the done bit will not be set because the commanded motion was not completed. Notice that this bit does not indicate whether the Actual Pressure or Actual Force has reached the **Requested Pressure/Force**.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	Pressure/Force is stopped (done)
0	1	Pressure/Force is increasing
1	0	Reserved
1	1	Pressure/Force is decreasing

See Also

Enter Pressure/Force Control (Time) (45) | Enter Pressure/Force Control (Rate) (46) | Ramp Pressure/Force (S-Curve) (41) | Ramp Pressure/Force (Linear) (42) | Stop Pressure/Force (43) | Hold Current Pressure/Force (19) | Pressure/Force Control Overview | List of Commands | Commands Overview

8.5.1.7. Command: Enter Pressure/Force Control (Time) (45)

Supported Axes: Pressure/Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Pressure/Force** (Pr or Fr)	any
2	**Ramp Type** • Linear (0) • S-curve (1)	A valid integer as described
3	**Time for Ramp** (sec)	≥ 0
4	**Integrator Preload** (%)	-100% to 100%

Description

This command enters pressure or force control and then ramps to the requested **Pressure** or **Force** in the specified **Time for Ramp**. A Linear **Ramp Type** will ramp the pressure or force linearly until it reaches the **Requested Pressure/Force**. An S-curve **Ramp Type** will provide a half s-curve ramp to the **Requested Pressure/Force**.

Upon entering pressure or force control, the Target Pressure or Target Force will be set to the Actual Pressure or Actual Force. The Target Pressure or Target Force will then be ramped to the **Requested Pressure/Force** in the specified **Time for Ramp**.

This command may cause the pressure/force rate to change suddenly at the transition, as shown in the plot in the example below. If this is a problem, try using a shorter **Time for Ramp**, or consider using the Enter Pressure/Force Control (Auto) (44) command, which will automatically ramp at the current Actual Pressure/Force Rate (depending on the tuning), eliminating the sudden change in rate. Another option is to create a user program that issues the Hold Current

Pressure/Force (19) command, then immediately issues the Ramp Pressure/Force (S-Curve) (41) command.

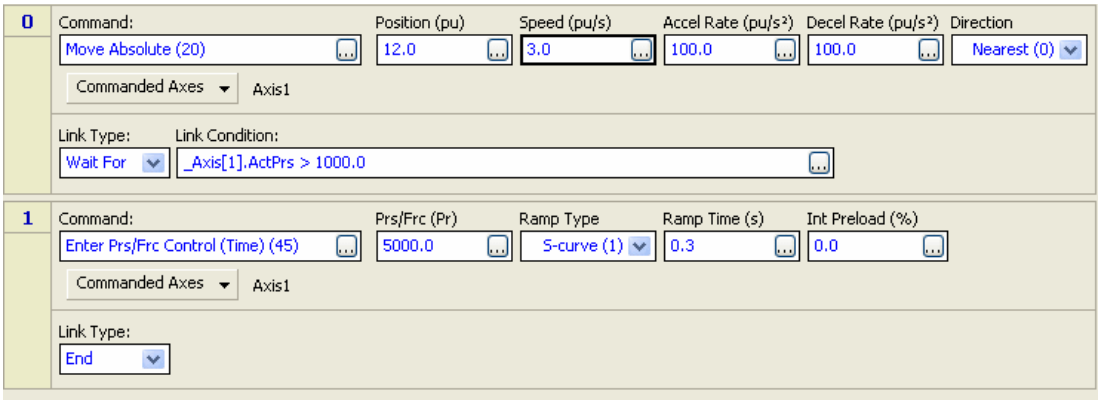
The **Integrator Preload** specifies the value that the Integral Output Term should be set to when transitioning into pressure/force control. The Preload is specified as a percentage of the Output Scale (typically 10V), and is not affected by the pre-transition integrator state. Typically, this can be left at the default value of 0. If the pressure/force exhibits a drop upon entering pressure/force, increase this value to help smooth the transition.

Example

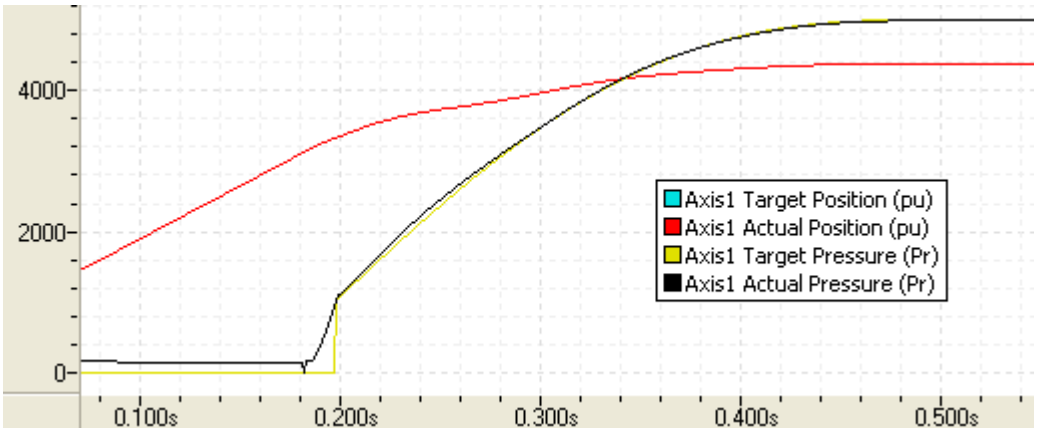
Consider a hydraulic press that must move toward the material to be pressed, transition to pressure control and hold a certain pressure.

The user program below shows a possible sequence of steps.

Step 0 issues a Move Absolute command to move the axis to a position beyond where it expects to encounter the material. The Step 0 Link Type waits for the Actual Pressure to exceed 1000, then it goes to Step 1 which issues an Enter Pressure/Force Control (Time) command with an S-Curve Ramp Type. The axis will transition to pressure control and go to the Requested Pressure/Force of 5000.



The plot below shows the transition:



At the beginning of the plot, the position is increasing. At 0.18 seconds, the Actual Pressure starts increasing. When the Actual Pressure reaches 1000 at approximately 0.2 seconds, the Enter Pressure/Force Control command is issued, the axis transitions to pressure control, and then the Target Pressure ramps to 5000.

Exiting Pressure/Force Control

To exit pressure/force control, send any open-loop command or closed-loop position command to the axis.

Status Bits

At Pressure/Force

When the axis is in Pressure or Force control and the Target Pressure or Force reaches the **Requested Pressure/Force** and the Actual Pressure or Actual Force is within the At Pressure/Force Tolerance window from the Target Pressure/Force, the At Pressure/Force Status bit will be set. This bit indicates that the ramp is complete and the axis is at the pressure or force.

Pressure/Force Control

The Pressure/Force Control Status bit indicates that the axis is in closed-loop pressure or force control.

Pressure/Force Target Generator Bits

The Pressure/Force Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Pressure/Force Target Generator Done bit

This bit indicates that the Target Pressure or Target Force has reached the **Requested Pressure/Force**. If the ramp is interrupted, e.g. due to a halt, the done bit will not be set because the commanded motion was not completed. Notice that this bit does not indicate whether the Actual Pressure or Actual Force has reached the **Requested Pressure/Force**.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	Pressure/Force is stopped (done)
0	1	Pressure/Force is increasing
1	0	Reserved
1	1	Pressure/Force is decreasing

See Also

Enter Pressure/Force Control (Auto) (44) | Enter Pressure/Force Control (Rate) (46) | Ramp Pressure/Force (S-Curve) (41) | Ramp Pressure/Force (Linear) (42) | Stop Pressure/Force (43) | Hold Current Pressure/Force (19) | Pressure/Force Control Overview | List of Commands | Commands Overview

8.5.1.8. Command: Enter Pressure/Force Control (Rate) (46)

Supported Axes: Pressure/Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Pressure/Force** (Pr or Fr)	any
2	**Pressure/Force Rate** (Pr/sec or Fr/sec)	> 0
3	**Pressure/Force Accel Rate** (Pr/sec^2 or Fr/sec^2)	≥ 0 (0=instantaneous) See **Caution** below.
4	**Integrator Preload** (%)	-100% to 100%

Description

This command enters pressure or force control and then ramps to the Target Pressure/Force to the **Requested Pressure/Force** at the rate specified by the **Pressure/Force Rate** parameter. It will accelerate to that rate as specified by the **Pressure/Force Accel Rate**.

Caution: Avoid overshoot!
This command starts with the Actual Pressure/Force Rate at the time the command is issued, and sbring it to the requested **Pressure/Force Rate** as specified by the **Accel Rate**. Therefore, if the pressure/force is rising rapidly when this command is issued, the Target Pressure/Force may overshoot the **Requested Pressure/Force** by a large amount. To prevent overshoot, set the **Requested Pressure/Force Accel Rate** to a very large value, or to zero for infinite acceleration. Or, use a different command for entering pressure/force control.

If the **Pressure/Force Accel Rate** is zero, the Pressure/Force will immediately begin changing at the specified **Pressure/Force Rate**.

Upon entering pressure or force control, the Target Pressure/Force will be set to the Actual Pressure/Force. The Target Pressure/Force will then be ramped to the **Requested Pressure/Force** at the specified **Pressure/Force Rate**.

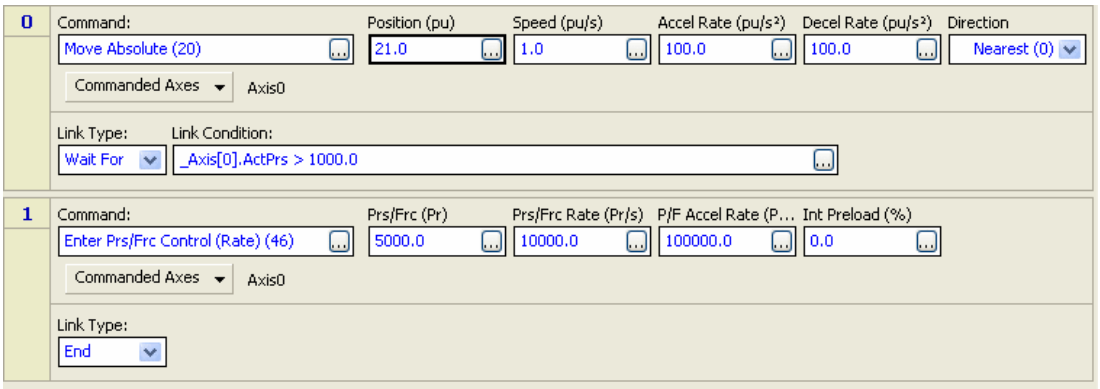
The **Integrator Preload** specifies the value that the Integral Output Term should be set to when transitioning into pressure/force control. The Preload is specified as a percentage of the Output Scale (typically 10V), and is not affected by the pre-transition integrator state. Typically, this can be left at the default value of 0. If the pressure/force exhibits a drop upon entering pressure/force, increase this value to help smooth the transition.

Example

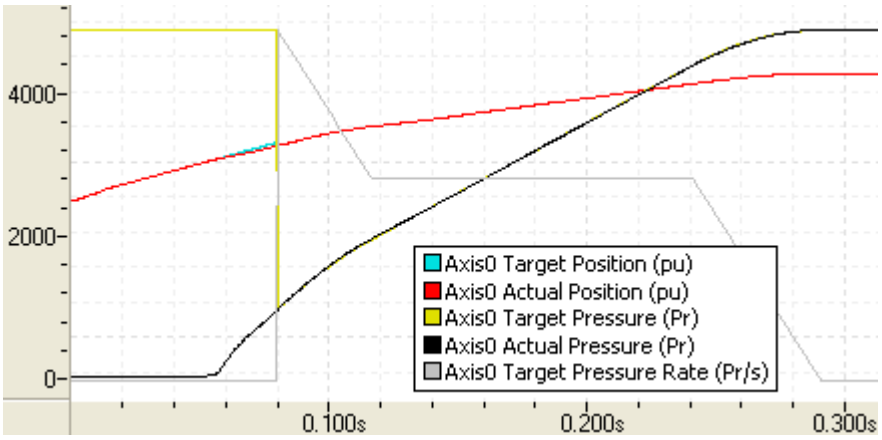
Consider a hydraulic press that must move toward the material to be pressed, transition to pressure control and hold a certain pressure.

The user program below shows a possible sequence of steps.

Step 0 issues a Move Absolute command to move the axis to a position beyond where it expects to encounter the material. The Step 0 Link Type waits for the Actual Pressure to exceed 1000, then it goes to Step 1 which issues an Enter Pressure/Force Control (Rate) command. The axis will transition to pressure control and go to the Requested Pressure/Force of 5000.



The plot below shows the transition:



At the beginning of the plot, the position is increasing. At approximately 0.55 seconds, the Actual Pressure starts increasing. When the Actual Pressure reaches 1000 at approximately 0.8 seconds, the Enter Pressure/Force Control command is issued, the axis transitions to pressure control, and then the Target Pressure ramps to 5000. Notice that the Target Pressure Rate decelerates to the specified rate and then decelerates to zero, stopping at the requested pressure.

Exiting Pressure/Force Control

To exit pressure/force control, send any open-loop command or closed-loop position command to the axis.

Status Bits

At Pressure/Force

When the axis is in Pressure or Force control and the Target Pressure or Force reaches the **Requested Pressure/Force** and the Actual Pressure or Actual Force is within the At Pressure/Force Tolerance window from the Target Pressure/Force, the At Pressure/Force Status bit will be set. This bit indicates that the ramp is complete and the axis is at the pressure or force.

Pressure/Force Control

The Pressure/Force Control Status bit indicates that the axis is in closed-loop pressure or force control.

Pressure/Force Target Generator Bits

The Pressure/Force Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Pressure/Force Target Generator Done bit

This bit indicates that the Target Pressure or Target Force has reached the **Requested Pressure/Force**. If the ramp is interrupted, e.g. due to a halt, the done bit will not be set because the commanded motion was not completed. Notice that this bit does not indicate whether the Actual Pressure or Actual Force has reached the **Requested Pressure/Force**.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	Pressure/Force is stopped (done)
0	1	Pressure/Force is accelerating
1	0	Pressure/Force is changing at a constant rate
1	1	Pressure/Force is decelerating

See Also

Enter Pressure/Force Control (Auto) (44) | Enter Pressure/Force Control (Time) (45) | Ramp Pressure/Force (S-Curve) (41) | Ramp Pressure/Force (Linear) (42) | Ramp Pressure/Force (Rate)

(18) | Stop Pressure/Force (43) | Hold Current Pressure/Force (19) | Pressure/Force Control Overview | List of Commands | Commands Overview

8.5.2. Pressure/Force Limit

8.5.2.1. Command: Set Pressure/Force Limit Mode (40)

Supported Axes: Pressure or Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Pressure/Force Limit** • Disabled (0) • Positive (1) • Negative (2)[1] • Bidirectional (3)[1] [1]**Note:** The Negative and Bidirectional options require firmware 3.44.0 or newer.	A valid integer as described

Description

This command enables or disables Pressure/Force Limit, and specifies the direction of the limiting. Pressure/Force Limit limits the Actual Pressure or Force during a position or velocity move. Pressure/Force Limit is useful in applications that require moving to a position while not exceeding a certain pressure or force during the move.

In Pressure/Force Limit, the Target Pressure/Force is the value at which the Actual Pressure or Force will be limited. The axis position or velocity can be controlled normally as long as the Actual Pressure or Force does not approach the limit. As the Actual Pressure or Force approaches the threshold, the RMC will limit the motion so that the pressure/force limit is not exceeded. For more details, see the Pressure/Force Limit Algorithm topic.

Direction

The direction of the force limiting is specified by choosing the **Positive**, **Negative**, or **Bidirectional** parameters.

- **Positive**
  The pressure/force will be limited to the Target Pressure/Force, in the direction of increasing pressure/force.
- **Negative**
  The pressure/force will be limited to the negative value of the Target Pressure/Force, in the direction of decreasing pressure/force. The Target Pressure/Force itself should be a positive value, but the limit will be applied to the negative value.
- **Bidirectional**
  The pressure/force will be limited in both directions of motion, to the positive Target Pressure/Force in the direction of increasing pressure/force, and to the negative Target Pressure/Force in the direction of decreasing pressure/force. The Target Pressure/Force itself

should be a positive value. If different limit values are required in each direction, then the user must keep track of the direction of motion and set the Target Pressure/Force appropriately.

Limitations

Pressure Limit cannot be used in Direct Output. To issue the Set Pressure/Force Limit Mode command, the Direct Output Status bit must be off. To turn off the Direct Output Status bit, put the axis in Open Loop or Closed Loop control.

If the axis is in pressure/force control, then pressure/force limit will be ignored. However, the **Set Pressure/Force Limit Mode (40)** command will still set and clear the Pressure/Force Limit Enabled status bit. If the axis exits pressure/force control, pressure/force limit will be enabled if the Pressure/Force Limit Enabled status bit is set.

Note:
Pressure/force limit may affect motion even when the Actual Pressure/Force is well below the limit. In order to achieve precise motion when pressure/force is not important, do not enable Pressure/Force Limit mode. Enable pressure/force limit only after the pressure/force has increased close to the point where the pressure/force is to be limited.
Another option is to use pressure/force control, which will not affect the position motion.

Usage Details

Note:
See the Position-Pressure and Position-Force Control topic for information on setting up a position-pressure or position-force axis.

Entering Pressure or Force Limit Mode

This procedure is can be used to enter Pressure/Force Limit mode.

1. Make sure the Target Pressure/Force has been set to the desired value. For example, the Ramp Pressure/Force (Linear) (42) command can be used to set the Target Pressure/Force to the desired value.
2. Send the Set Pressure/Force Limit Mode (40) command to enable Pressure/Force Limit.
3. Move the axis as desired. When the motion begins to affect the pressure, the RMC will limit the motion such that the Actual Pressure/Force does not exceed the Target Pressure/Force.

   Note:
   Pressure/Force Limit mode may affect normal closed-loop motion even when the pressure is very low. Therefore, if possible, do not enter Pressure/Force Limit until you need to.

Changing the Target Pressure/Force

Once the axis is in pressure or force control, you can issue any of the following pressure/force commands to change the pressure/force. For example, you can ramp the pressure up or down, or perform a sinusoidal profile, or follow a spline. For details on each command, see the respective help topics.

- Ramp Pressure/Force (Linear) (42)
- Ramp Pressure/Force (S-Curve) (41)
- Stop Pressure/Force (43)
- Sine Start (Prs/Frc) (76)
- Sine Stop (Prs/Frc) (77)
- Change Target Parameter (Prs/Frc) (81)
- Curve Start (Prs/Frc) (87)
- Curve Start Advanced (Prs/Frc) (89)

DO NOT issue the pressure/force commands that enter pressure/force control, because this will exit pressure/force limit:

- Hold Current Pressure/Force (19)
- Enter Pressure/Force Control (Auto) (44)
- Enter Pressure/Force Control (Time) (45)
- Enter Pressure/Force Control (Rate) (46)

Exiting Pressure/Force Limit

An axis will exit pressure/force limit mode if any of the following occurs:

- The **Set Pressure/Force Limit Mode (40)** command is sent with the **Pressure/Force Limit** command parameter set to 0 (Disabled).
- A Direct Output (9) command is sent to the axis.
- An Open Loop Halt or Direct Output Halt occurs on the axis.
- The Fault Controller (8) command is issued to the RMC.

For more details, see the Pressure/Force Limit topic.

Status Bits

Pressure/Force Limit Enabled

This bit indicates that pressure/force limit is enabled. If an axis is in pressure or force control, pressure/force limit will not be active. If the axis exits pressure/force control to open loop or closed-loop position or velocity control, it will enter pressure/force limit if the Pressure/Force Limit Enabled bit is on.

Pressure/Force Limited

This bit indicates that pressure/force limit is enabled and the axis is limiting the pressure/force. If an axis is in pressure or force control, pressure/force limit will not be active and this bit will not be set.

At Pressure/Force

When the axis is in Pressure/Force control or Pressure/Force Limit and the Target Pressure/Force reaches the **Requested Pressure/Force** and the Actual Pressure/Force is within the At Pressure/Force Tolerance window from the Target Pressure/Force, the At Pressure/Force Status bit will be set. This bit indicates that the ramp is complete and the axis is at the pressure or force.

See Also

Ramp Pressure/Force (S-Curve) (41) | Ramp Pressure/Force (Linear) (42) | Stop Pressure/Force (43) | Pressure/Force Control Overview | List of Commands | Commands Overview

8.5.3. Specialty

8.5.3.1. Command: Gear Absolute Prs/Frc (59)

Supported Axes:	Pressure or Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range

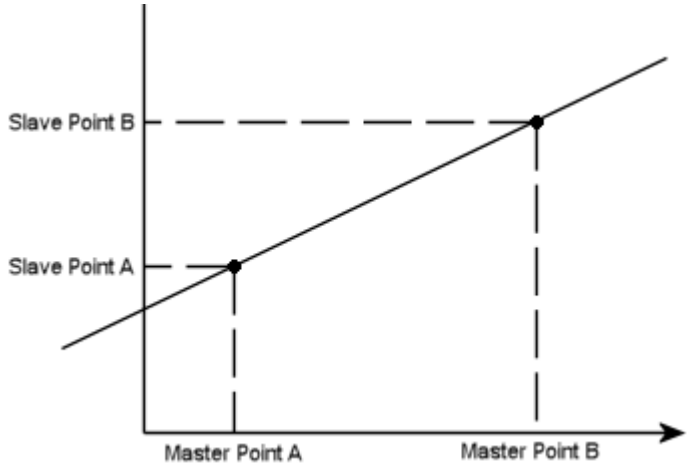
1	**Master Register** **Note:** See **Specifying a Register Address** below.	Valid RMC register
2	**Master Point A** (Pr or Fr)	Any REAL number
3	**Master Point B** (Pr or Fr)	Any REAL number
4	**Slave Point A** (Pr or Fr)	Any REAL number
5	**Slave Point B** (Pr or Fr)	Any REAL number
6	**Endpoint Behavior** • Fault (0) • Truncate (1) • Extrapolate (2)	A valid integer as described.

Description

This command sets up an absolute linear gearing relationship between the master register and the pressure/force target for the axis this command was issued to (the slave axis) and will make the slave axis follow that relationship. Typically, the master register is the Actual Pressure/Force of a reference axis. This command is very useful for making an axis follow a reference input (half-axis).

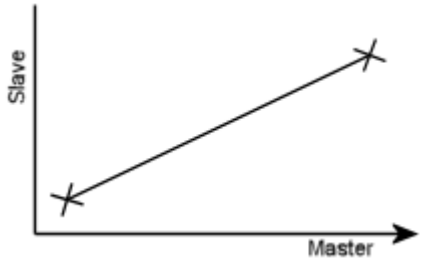
When this command is issued, the Target Pressure/Force for the slave axis must either already be at the correct point specified by the relationship based on the current value of the master, or a Transition command must previously have been issued to specify how the axis should move to get to the line.

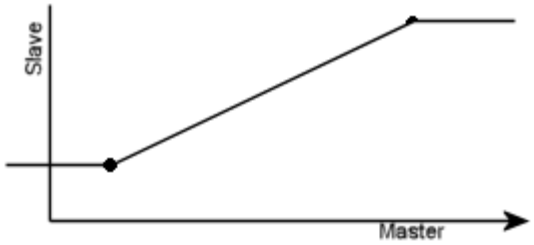
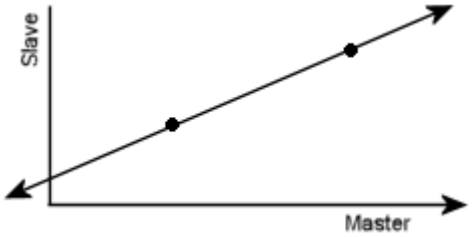
Master Point A, **Master Point B**, **Slave Point A**, and **Slave Point B** specify the linear relationship. The behavior beyond these points depends on the **Endpoint Behavior** parameter.



EndPoint Behavior

The Endpoint Behavior specifies what happens to the slave axis if the master moves outside of the range specified by Master Point A and Master Point B:

Option	Description	Image
Fault	If the master exceeds either endpoint, the slave axis' runtime error bit will be set which will halt the axis if the Auto Stops are configured to do so.	

Truncate	If the master moves past an endpoint, the slave axis' Target Pressure/Force will stop at the endpoint. When the master moves back into the range, the gearing will resume. Notice that if the master is moving quickly when it exceeds the endpoint, it may cause the slave's Target Pressure/Force to stop abruptly. **Note:** If a superimposed transition is used, in certain cases it can cause the slave to exceed the endpoints during the transition. If this causes problems, consider using a different type of transition.	
Extrapolate	The slave will always follow the master on the linear relationship.	

See the Gearing Overview topic for general information about gearing, including possible Gear Masters.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Master Register** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Target Generator State Bits

The Pressure/Force Target Generator Done, State A and State B bits are all off during the gearing.

Prs/Frc TG SI Busy (Pressure/Force Target Generator Superimposed Busy) Bit

This bit will be set during the transition. The transition begins when the motion command is issued, not necessarily when the Transition command is issued. When the transition completes, this bit will clear.

See Also

Gearing Overview | List of Commands | Commands Overview

8.5.3.2. Command: Transition Disable Prs/Frc (63)

Supported Axes: Pressure/Force Control Axes

Command Parameters

None.

Description

This command disables transitions on a pressure/force axis. When transitions are disabled, issuing any pressure/force command that requires a transition will cause a command error, which will halt the axis if the AutoStops are set to do so.

Transitions are useful for starting certain types motion even though the axis is not at the correct starting point. For example, the Curve Start (Prs/Frc) (87), Sine Start (Prs/Frc) (76), and Gear Absolute (Prs/Frc) (59) commands normally require that the Target Pressure/Force be at the correct starting point. However, if a pressure/force transition has been enabled (by issuing the Transition Rate (Prs/Frc) (64) command), then these commands can be issued even though the axis is not at the correct starting point. The axis will transition in the manner requested by the current transition mode. See the Transition Rate (Prs/Frc) (64) command for details.

This command will not affect any transitions that are in progress. When the RMC powers up, transitions are disabled on all axes. To enable pressure/force transitions, issue the Transition Rate (Prs/Frc) (64) command.

See Also

Transition Rate (Prs/Frc) (64)

8.5.3.3. Command: Transition Rate (Prs/Frc) (64)

Supported Axes: Pressure/Force Control Axes

Command Parameters

#	Parameter Description	Range
1	**Prs/Frc Rate** (Pr/s or Fr/s)	Any REAL number.
2	**Prs/Frc Accel Rate** (Pr/s^2 or Fr/s^2)	Any REAL number.
3	**Transition Type** • Seek (0) • Reach (1)	A valid integer as described.

	• Superimposed (2)	

Description

This command enables transitions and defines the pressure/force rate and pressure/force acceleration of the transition. This command does not start any motion. Rather, it will apply when certain pressure/force commands are issued to the axis.

Transitions are useful for starting certain types of motion even though the axis is not at the correct starting point. For example, the Gear Absolute (Prs/Frc) (59), Sine Start (Prs/Frc) (76), and Curve Start (Prs/Frc) (87) commands normally require that the Target Pressure/Force be at the correct starting point. However, if a transition has been enabled, then these commands can be issued even though the axis is not at the correct starting point. When the pressure/force command is issued, the axis will move toward the requested profile (curve, sine wave, gearing relationship, etc.) as defined by the transition command.

When the RMC powers up, transitions are disabled on all axes. To enable pressure/force transitions, issue the Transition Rate Prs/Frc (64) command. Once this command has been issued, it does not need to be issued again, unless you wish to specify a different transition, or if you need to re-enable pressure/force transitions after disabling them. To disable pressure/force transitions, issue the Transition Disable (Prs/Frc) (63) command.

This command will not affect any transitions that are in progress.

Transition Types

This command provides the following transition options:

- **Seek (0)**
  The axis will move toward the requested profile using the **Prs/Frc Rate** and **Prs/Frc Accel Rate**. When the pressure/force and pressure/force rate of the axis come close to the position and velocity of the profile, the axis will "lock" onto the profile. Use this option to get to the requested profile quickly and smoothly.
- **Reach (1)**
  The axis will move toward the requested profile using the **Prs/Frc Rate** and **Prs/Frc Accel Rate**. When the pressure/force reaches the pressure/force of the profile, the axis will "lock" onto the profile. Notice that this option does not require the pressure/force rates to be close when it locks on, and therefore may cause the axis to jerk. Use this option to get to the requested profile as quickly as possible.
- **Superimposed (2)**
  A trapezoidal move using the **Prs/Frc Rate** and **Prs/Frc Accel Rate** will be superimposed onto the requested profile such that the axis will reach the profile. Notice that since the move is superimposed onto the profile, the axis will not necessarily move at the specified pressure/force rate and acceleration rate, but rather at the sum of the rates and accel rates from the requested profile and superimposed portions of the move.

  This method will always guarantee that the axis will lock on to the requested profile, even if the **Prs/Frc Rate** and **Prs/Frc Accel Rate** are slower than that of the profile. The time it takes to lock on will be based on how far the current Target Pressure/Force is from the profile and on the **Prs/Frc Rate**. For example, if the Target Pressure/Force is 3, the requested profile is at 9, and the Prs/Frc Rate is 6, it will take roughly one second to lock on. The "lock-on" of the Superimposed method will usually be as smooth or smoother than **Seek**.

Choosing a Transition Type

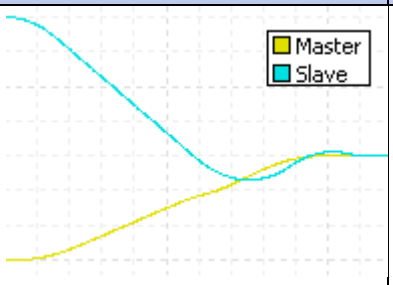
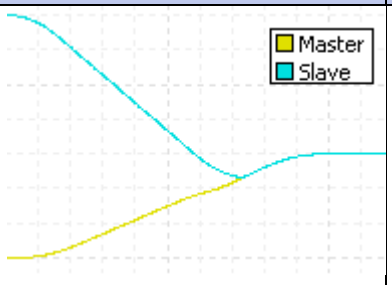
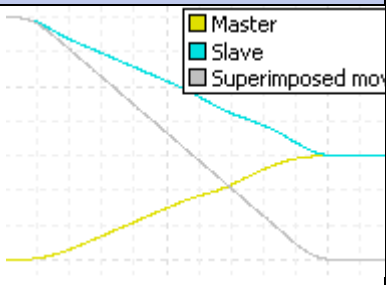
In general, try the **Seek** method first. If it takes too long to lock on, switch to **Reach**. For either of these methods, make sure to set the **Prs/Frc Rate** and **Prs/Frc Accel Rate** to values higher than that of the profile it is trying to follow. Otherwise, the axis may never catch up to the requested profile. Notice that if the master register is stopped, then all 3 methods will perform similarly.

If the transition is used for a gearing application, the behavior of any of the methods will be the same if the master is not moving while the transition is taking place. If the master is moving, and is a well-behaved Target Pressure/Force, then the **Seek** and **Reach** methods will both behave

very similarly. If the master is moving, but is noisy, such as an analog reference signal, then the **Reach** method will "lock in" the quickest, but may cause a jerk.

The **Superimposed** method is not as useful as the first two methods, but it can provide a more predictable "lock-on", as described above. Also, this method will always guarantee that the axis will lock on to the requested profile, even if the **Prs/Frc Rate** and **Prs/Frc Accel Rate** are slower than that of the curve.

Shown below is an example of how the various options work for one sample profile. Notice that the behavior will vary for other master signals.

Seek	Reach	Superimposed
		
The Seek method waits until the pressure/force and pressure/force rate are close before locking in. If the master was not accelerating or decelerating, it would lock in quicker.	The Reach method locks in as soon as the pressure/force reaches the master. This can cause a jerk in the Target Pressure/Force of the slave axis.	The gray line shows the move that is superimposed onto the master, and results in the slave target.

Status Bits

Prs/Frc TG SI Busy (Pressure/Force Target Generator Superimposed Busy) Bit

This bit will be set when the transition begins. Notice that this is when the motion command is issued, not necessarily when the Transition rate command is issued. When the transition actually takes place, this bit will be set until the axis "locks on".

See Also

Transition Disable (Prs/Frc) (63) | Transition Rate (56)

8.5.3.4. Command: Sine Start (Prs/Frc) (76)

Supported Axes: Pressure or Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Offset** (pressure- or force-units)	any
2	**Amplitude** (pressure- or force-units)	≥ 0
3	**Frequency** (Hz)	0* to 1/4 of the loop frequency

4	**Cycles**	0 to 16 million in 0.25 steps (0 = continuous)
5	**Start Location** • Auto (0) • Mid-Pos (1) • Pos Peak (2) • Mid-Neg (3) • Neg Peak (4)	a valid integer as described
6	**Status Block** (address) **Note:** See **Specifying a Register Address** below.	Address or **none** (0)

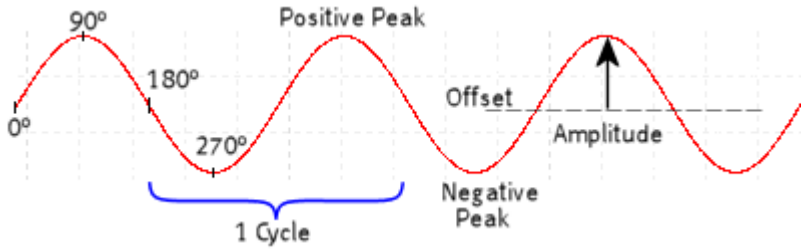
*See the **Frequency** section below for details on the lower limit.

Description

This command starts a continuous sinusoidal move on the pressure or force. If the number of cycles is specified, the sine move will end after completing the cycles and will remain at that pressure or force. For position axes, use the Sine Start (72) command.

Before issuing this command, the Target Pressure/Force must be at the location specified by the **Start Location** command parameter, or, a Transition command must previously have been issued to specify how the axis should move to get to the sine curve. A transition command will allow you to start a sine curve even though the axis is not at the correct starting point.

If you want the axis to follow the sine curve exactly starting from the beginning, do not use a transition; instead, make sure the axis is at the starting point before issuing the Sine Start command.



Offset

The **Offset** specifies the center of the sine wave.

Amplitude

The **Amplitude** specifies the distance from the center to the peak. The amplitude can be zero or a positive number. An amplitude of zero is typically only used when it will be followed by the Change Target Parameter (Prs/Frc) (81) command to ramp the amplitude from zero to some value.

Cycles

The **Cyles** specifies the number of cycles, in increments of 0.25. Each cycle begins at the location specified by the **Start Location** parameter. If **Cycles** is zero, the sine wave will repeat endlessly. By specifying various increments of 0.25, you can control whether the sine wave stops at the bottom, top, or middle.

A cycle count of 0.25 is not allowed if the Start Location is at one of the peaks, and the Frequency is zero.

Due to the limits of 32-bit floating point values, the Cycles parameter can be a maximum of 16 million only if it is a whole number. If it contains a half, such a 45.5, it can be a maximum of 7,999,999.5. If it contains a quarter, such a 12.25 or 97.75, it can be a maximum of 3,999,999.75.

Frequency

The frequency can be any value from zero up to the maximum frequency listed in the table below. The maximum frequency is a quarter of the loop frequency (the loop frequency is the inverse of the Loop Time). For example, if the loop time is 1000 µsec, the maximum frequency is 250 Hz.

Loop Time	Max Frequency
250 µsec	1000 Hz
500 µsec	500 Hz
1000 µsec	250 Hz
2000 µsec	125 Hz
4000 µsec	62.5 Hz

Fractional frequencies are valid, such as 0.345 or 5.84. Issuing this command with a frequency of zero is typically only done when it will be followed by the Change Target Parameter (Prs/Frc) (81) to ramp the frequency from 0 to some value. A frequency of zero is not allowed if the Start Location is at one of the peaks, and the cycle count is 0.25.

Frequencies below 0.00001 on the RMC75 or RMC150E, and below 0.05 on the RMC75S and RMC75P, may cause an irregular Target Force/Pressure when used with the Start Locations Mid-Pos or Mid-Neg. Test any very low frequencies before using them.

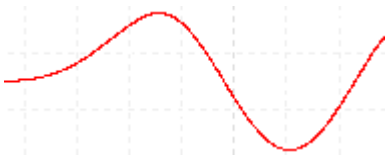
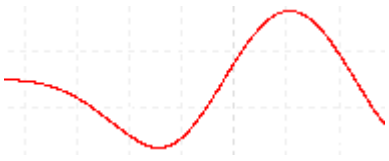
Start Location

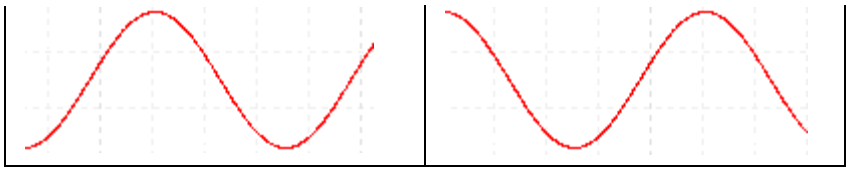
The **Start Location** specifies where on the sine wave the motion should start. When this command is issued, the Target Pressure/Force must be at that location, unless a Transition command has previously been issued to specify how the axis should move to get to the sine curve. The **Start Location** options are given below. Notice that each cycle will start at the location specified by the **Start Location**.

Start Location	Description
Auto	If the Target Pressure or Force is at the positive peak, negative peak, or the center, then the sine move will start there. If it starts at the center, it will move in the positive direction. See **Mid-Pos and Mid-Neg Details** below.
Mid-Pos	The sine wave will start at the center (zero degrees) and move in the positive direction. See **Mid-Pos and Mid-Neg Details** below.
Pos Peak	The sine wave will start at the positive peak (90 degrees).
Mid-Neg	The sine wave will start at the center (180 degrees) and move in the negative direction. See **Mid-Pos and Mid-Neg Details** below.
Neg Peak	The sine wave will start at the negative peak (270 degrees).

For example, if the **Start Location** is Auto, the **Offset** is 3, and the **Amplitude** is 1, then the Target Pressure/Force must be at 2, 3, or 4 when this command is issued. If the **Start Location** is Pos Peak, the **Offset** is 10, and the **Amplitude** is 2, then the Target Pressure/Force must be at 12 when this command is issued.

The diagrams below show what the start of the sine wave looks like for a the starting locations.

Mid-Pos	**Mid-Neg**
	
Neg Peak	**Pos Peak**



Mid-Pos and Mid-Neg Details

If the **Start Location** is **Mid-Pos** or **Mid-Neg**, in order to prevent a sudden jump in the velocity, the frequency will be logarithmically ramped from zero Hertz to the requested frequency during the first quarter cycle. Therefore, the first quarter cycle will not be a true sine function, and the first quarter cycle will take longer. See the **Mid-Pos** and **Mid-Neg** images above. If a sine move stops at the middle, the same type of ramping will occur in the last quarter cycle.

To achieve a different ramping behavior for **Mid-Pos** or **Mid-Neg** start locations, do the following:

- Send the **Sine Start** command with the frequency set to zero. The sine wave will start, but there will be no motion.
- Use the **Change Target Parameter** command to ramp the frequency to the desired value. The **Change Target Parameter** command provides great flexibility in defining the ramp.

Note: For Start Locations **Mid-Pos** or **Mid-Neg**, very low frequencies, such as 0.0001 or less, may cause an irregular Target Position. Test any very low frequencies before using them.

Note: For Start Locations **Mid-Pos** or **Mid-Neg** with a non-zero starting frequency, changing the frequency with the Change Target Parameter (Prs/Frc) (81) command is not allowed until the sine wave has reached the first positive or negative peak.

Status Block

The optional **Status Block** specifies the location in the Variable Table of a block of six registers that provide read-only information on the sine move. This block will not be needed by most users and the Status Block parameter should then be set to **none**. For more details, see the **Sine Move Status** section below.

Stopping a Sine Move

If the number of cycles is non-zero, the sine move will automatically stop after reaching the number of cycles. By specifying the Cycles in various increments of 0.25, you can control whether the sine wave stops at the bottom, top, or middle. If the Cycles is a whole number, the sine move will stop at the same location where it started.

The Sine Stop (Prs/Frc) (77) command is especially designed for stopping a pressure/force sine move. The specific stop location can be specified to be the positive peak, negative peak, next peak, middle, or after the current cycle completes. See the Sine Stop (Prs/Frc) (77) command for more details.

If the stop location is the middle, the frequency will be ramped to zero hertz during the last quarter cycle. This prevents a sudden jump in the pressure/force rate, because the middle of a sine move otherwise has a non-zero rate.

Of course, as with any motion on an axis, when a sine move is in progress, another pressure/force command can be issued and the axis will immediately stop the sine move and start the new motion. For example, a Stop Pressure/Force (43) command will stop the Pressure/Force target.

Ramping Sine Move Parameters

Use the Change Target Parameter (Prs/Frc) (81) command to ramp the Offset, Frequency, or Amplitude, or change the Cycles of a pressure/force sine move in progress. Each parameter can be ramped independently, that is, each parameter can be ramped whether or not other parameters are ramping. See the Change Target Parameter (Prs/Frc) (81) command for details.

Example

A testing application requires that the amplitude be ramped from 0 to 10 over 20 cycles. To do this, first issue the Sine Start (Prs/Frc) (76) command with an Amplitude of 0. Then, issue the Change Target Parameter (Prs/Frc) (81) command to ramp the Amplitude to 10 during 20 cycles.

Sine Move Status

Axis Status Registers for a Sine Move

The Cycles (Pressure/Force) Axis Status register gives the current number of cycles completed for the pressure or force sine move in progress. It is listed in Axis Tools, in the Axis Status Registers pane, on the All tab, in the Target section. The Cycles Axis Status register is a DINT.

Status Block

Advanced users may wish to use the Sine Start command's **Status Block**, which provides read-only information on the sine move. This information is most useful when manipulating sine moves in user programs.

To use the Status Block, you must specify an address from the Variable Table in the **Status Block** parameter of the Start Sine command. The Status Block will require six registers in the Variable Table, beginning with the specified address. As the sine move runs, the selected registers in the Variable Table will be continuously updated. The selected variables will not be named; you should give a descriptive name to help you keep track of them.

To prevent confusion, sine moves that are running simultaneously should not use the same Status Block addresses. Non-simultaneous sine moves can use the same Status Block addresses.

The Status Block provides the following information:

Status Block Offset	Name	Data Type	Description
0	Current Cycle Count	REAL	The number of cycles the sine move has completed. Each cycle begins at the location specified by the **Start Location** command parameter. The fractional part of the cycle is given by the Current Cycle Fraction below. For continuous sine moves (without a fixed number of cycles), this value will wrap to zero after it reaches 10,000,000 and then continue incrementing. For sine moves with a fixed number of cycles, this value will not go beyond the requested cycle count.
1	Current Cycle Fraction	REAL	The fractional part of the Current Cycle above. The fractional part is given in this separate register to retain accuracy as the Current Cycle reaches high values.
2	Current Angle	REAL	The current angle from 0 up to, but not including, 360 degrees. This angle is the mathematical angle as shown in the sine wave diagram above. An angle of zero does not necessarily coincide with the start of a cycle. This value can be used for such things as determining whether the sine move is at the positive peak, negative peak, or middle.
3	Current Amplitude	REAL	The current amplitude of the sine move in pressure or force units.
4	Current Frequency	REAL	The current frequency of the sine move in hertz.
5	Current Offset	REAL	The current offset of the sine move in pressure or force units.
6	Current Phase	REAL	The current phase of the sine move in degrees.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Status Block** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Status Bits

Pressure/Force Target Generator Done bit

This bit indicates that the sine move has completed the specified number of cycles. If the motion move is interrupted, e.g. due to a halt, the done bit will not be set because the commanded motion was not completed (it may be set after the interrupting motion is complete). Notice that this bit does not indicate whether the Actual Pressure or Force has reached the Requested Pressure or Force. If the Pressure/Force Target Generator Done bit is set, and the Actual Pressure or Force is within the At Pressure/Force Tolerance window, the At Pressure/Force Status bit will be set. This bit indicates that the move is complete and the axis is at pressure or force.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	Reserved
0	1	Reserved
1	0	Reserved
1	1	Reserved

Prs/Frc TG SI Busy (Pressure/Force Target Generator Superimposed Busy) Bit

This bit will be set during the transition. The transition begins when the motion command is issued, not necessarily when the Transition command is issued. When the transition completes, this bit will clear.

See Also

Sine Stop (Prs/Frc) (77) | Sine Start (72) | Sine Stop (73)

8.5.3.5. Command: Sine Stop (Prs/Frc) (77)

Supported Axes: Pressure or Force Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Stop Location** • Next Cycle (0)	a valid integer as described

	• Middle (1) • Pos Peak (2) • Neg Peak (3) • Next Peak (4)	

Description

This command stops a pressure or force sine move at a prescribed location. For stopping a position sine move, see the Sine Stop (73) command.

The sine move will stop at the next occurrence of the specified **Stop Location**. The **Stop Location** can be any of the following:

- **Next Cycle**
  The sine move will stop at the end of the current cycle. The location of the end of each cycle is defined by the Sine Start (Prs/Frc) (76) command. Stopping the sine move with the **Next Cycle** option will ensure that the sine move stops in the same location that it started. For example, if it started at a positive peak, this command will stop it at the next positive peak.

  If the stop location is the middle, and the axis is currently on the quadrant after a peak and before the midpoint, then the axis will not stop on that first middle point because it could lead to extreme pressure or force rates. Instead, the axis will continue another full cycle and then stop at the middle point.

  If the stop location is the middle, the target generator will perform the stop by ramping the frequency down to zero over the last quarter-cycle. Due to this, any attempt to ramp the frequency with the Change Target Parameter (Prs/Frc) (81) command during this portion of the move will be ignored.
- **Middle**
  The sine move will stop at the first middle point. However, if the axis is currently on the quadrant after a peak and before the midpoint, then the axis will not stop on that first middle point because it could lead to extreme pressure or force rates. Instead, the axis will continue a half-cycle after the first middle point and stop at the next middle point.

  When the **Stop Location** is set to **Middle**, the target generator performs the stop by ramping the frequency down to zero over the last quarter-cycle. Due to this, any attempt to ramp the frequency with the Change Target Parameter (Prs/Frc) (81) command during this portion of the move will be ignored.
- **Pos Peak**
  The sine move will stop at the next positive peak.
- **Neg Peak**
  The sine move will stop at the next positive peak.
- **Next Peak**
  The sine move will stop at the next peak, whether it is negative or positive.

Status Bits

At Pressure or Force Bit

After the target pressure or force stops, and the Actual Pressure or Force is within the At Pressure/Force Tolerance window, the At Pressure/Force Status bit will be set. This bit indicates that the move is complete and the axis is at pressure or force.

Pressure/Force Target Generator Done bit

This bit indicates the move is complete, which occurs when the target has stopped.

Pressure/Force Target Generator State A and B bits

B	A	Description
0	0	Reserved
0	1	Reserved

1	0	Reserved
1	1	Reserved

See Also

Sine Start (Prs/Frc) (76) | Sine Start (72) | Sine Stop (73)

8.5.3.6. Command: Change Target Parameter (Prs/Frc) (81)

Supported Axes:	Pressure or Force Axes
Supported Control Modes:	Pressure and Force

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Parameter** • Offset (0) • Amplitude (1) • Frequency (2) • Cycles (3) • Phase (4)	A valid integer as described
2	**New Value**	Depends on selected Parameter
3	**Ramp Type** • Time (0) • Cycles (1) • Rate (2) • Time (log) (3) • Cycles (log) (4) • Time - 5th (5)	A valid integer as described
4	**Ramp Value**	Depends on selected Ramp Type

Description

This command changes a target parameter for Sine Start (Prs/Frc) (76) moves that are in progress. For example, this command can be used to perform frequency sweeps by ramping the frequency of a sine move in a certain amount of time. For position control, see the Change Target Parameter (80) command.

Target parameters can be ramped independently of each other. That is, each parameter can be ramped whether or not other parameters are ramping. Multiple Change Target Parameter commands can be issued from a single step in a user program, allowing them to be ramped together.

If ramping of frequency and phase is in progress when the sine wave begins a stop at the middle position, the frequency and phase ramping will cease when the sine wave begins its stop, which occurs at the last peak before the middle position.

If ramping of the offset is in progress when the sine wave stops, there may be a discontinuity in the Target Velocity.

Sine Start Parameters

When this command is used for changing target parameters of motion started with a Sine Start (Prs/Frc) (76) command, the following parameters can be changed:

Offset

When ramping the offset, there will be a discontinuity in the pressure/force rate at the beginning and end of the ramp, except for the Time-5th ramp type. This can cause the axis to jump if the Pressure/Force Rate Feed Forward is being used.

Valid Ramp Types	
Time	Ramp the Offset linearly in the number of seconds specified by the **Ramp Value**.
Cycles	Ramp the Offset linearly in the number of cycles specified by the **Ramp Value**.
Rate	Ramp the Offset linearly at the rate (units/sec) specified by the **Ramp Value**.
Time-5th	Ramp the Offset using a fifth-order polynomial in the number of seconds specified by the **Ramp Value**. This results in a smoother start and finish than can be obtained with the other ramp types. The starting rate of change is assumed to be zero.

Amplitude

The amplitude can be ramped to any positive value, or zero.

Valid Ramp Types	
Time	Ramp the Amplitude linearly in the number of seconds specified by the **Ramp Value**.
Cycles	Ramp the Amplitude linearly in the number of cycles specified by the **Ramp Value**.
Rate	Ramp the Amplitude linearly at the rate (units/sec) specified by the **Ramp Value**.
Time-5th	Ramp the Phase using a fifth-order polynomial in the number of seconds specified by the **Ramp Value**. This results in a smoother start and finish than can be obtained with the other ramp types. The starting rate of change is assumed to be zero.

Frequency

The frequency can be ramped to any positive value, or zero.

Valid Ramp Types	
Time	Ramp the Frequency linearly in the number of seconds specified by the **Ramp Value**.
Cycles	Ramp the Frequency linearly in the number of cycles specified by the **Ramp Value**.
Rate	Ramp the Frequency linearly at the rate (units/sec) specified by the **Ramp Value**.

Time (log)	Ramp the Frequency logarithmically in the number of seconds specified by the **Ramp Value**. Ramping frequency logarithmically typically gives a smoother frequency transition than linear ramping, especially over wide frequency ranges.
Cycles (log)	Ramp the Frequency logarithmically in the number of cycles specified by the **Ramp Value**. Ramping frequency logarithmically typically gives a smoother frequency transition than linear ramping, especially over wide frequency ranges.

Note: If the sine move is in the process of completing the first or last quarter-cycle of a sine move that respectively starts or ends at the midpoint, then any attempt by the user to ramp the frequency during this portion of the move will be ignored.

Cycles

The Cycles parameter is updated immediately when this command is issued. The **Ramp Type** and **Ramp Value** parameters are ignored. The Cycles specifies the total number of cycles from when the Sine Start command was issued.

Example

Consider a sine move that was started with a Cycles parameter of 1000, and when the cycle count reached 600 the Change Target Parameter command is issued to change the Cycles to 1300. The sine move will continue until it completes 1300 cycles from when the Sine Start command was issued. Therefore, it will complete 700 more cycles after the Change Target Parameter command is issued.

If the Cycles is less than the current cycles, the sine move will end. It will end at the next occurrence of the same ending location that is specified by the new Cycles parameter. This location will be the same as the starting phase if the requested Cycles is a whole number, but different otherwise.

Example

Consider a sine move on that force axis that started with an Offset of 0, an Amplitude of 2, a Cycles parameter of 100, and a Start Location of Mid-Pos. After 30 cycles, the Change Target Parameter command is issued to change the Cycles to 21.25. Since 21.25 would result in an ending location at the positive peak, the sine move will end at the next occurrence of the positive peak, or a force of 2.

Phase

The phase can be ramped to any value from -360 to +360. The phase is relative to the phase when the Sine Start command was issued, which is always zero.

Valid Ramp Types	
Time	Ramp the Phase linearly in the number of seconds specified by the **Ramp Value**.
Cycles	Ramp the Phase linearly in the number of cycles specified by the **Ramp Value**. In cases when the frequency is also changing, the ramping of phase by cycles differs from that of the ramping of frequency, amplitude, and offset, since the ramping of phase by cycles does not take into account the additional progress through the cycle caused by the change in phase.
Rate	Ramp the Phase linearly at the rate (units/sec) specified by the **Ramp Value**.
Time-5th	Ramp the Phase using a fifth-order polynomial in the number of seconds specified by the **Ramp Value**. This results in a smoother start and finish than can be obtained with the other ramp types. The starting rate of change is assumed to be zero.

See Also

Sine Start (Prs/Frc) (76) | Change Target Parameter (80)

8.5.3.7. Command: Curve Start (Prs/Frc) (87)

Supported Axes: Pressure or Force Control Axes

Command Parameters

#	Parameter Description	Data Type	Range
1	**Curve ID**	Internal: DINT External: REAL	0-50000
2	**Master Register**	REAL	_Time or any other valid register of type REAL.
3	**Cycles** 0 = infinite, 1 = 1 cycle *n* = *n* cycles (up to 16 million)	REAL	0 to 16 million (0 = continuous)

Description

This command is identical to the Curve Start (86) command, but is used for pressure or force. For details, refer to the Curve Start (86) command.

For more advanced options, such as scaling or offsetting the curve, or absolute and relative options for the master or curve alignment, see the Curve Start Advanced (Prs/Frc) (89) command.

See Also

Curve Start (86) | Curves Overview

8.5.3.8. Command: Curve Start Advanced (Prs/Frc) (89)

Supported Axes: Pressure or Force Control Axes

Command Parameters

#	Parameter Description	Data Type	Range
1	**Curve ID**	Internal: DINT External: REAL	0-50000
2	**Master Register**	REAL	_Time or any other valid register of type REAL.

3	**Cycles** 0 = infinite, 1 = 1 cycle *n* = *n* cycles (up to 16 million)	REAL	0 to 16 million (0 = continuous)
4	**Options, sum of the following:** **Curve Alignment, one of:** Absolute Curve Alignment (+0) Relative Curve Alignment (+1) **Master Alignment, one of:** Relative Master Alignment (+0) Absolute Master Alignment (+2) **Endpoint Behavior** **For Absolute Master alignment, one of:** Fault (+0) Truncate (+4) Extrapolate (+8) **For Relative Master alignment, one of:** Standard (+0) Truncate (+4) Truncate and End (+8)	Internal: DINT External: REAL	≥ 0, whole numbers
5	**Curve Scale**	REAL	any
6	**Curve Offset**	REAL	any
7	**Master Scale**	REAL	≠ 0
8	**Master Offset**	REAL	any
9	**Status Block** (address)	REAL	Address or **none** (0)

Description

This command is identical to the Curve Start Advanced (88) command, but is used for pressure or force. For details, refer to the Curve Start Advanced (88) command.

See Also

Curve Start Advanced (88) | Curves Overview

8.6. Set Parameters

8.6.1. Command: Offset Position (47)

Supported Axes: All Position Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Data Type	Range
1	**Position Change** (position units)	REAL	Any

Description

This command offsets the Actual Position of the axis by the specified **Position Change**. This command can be sent anytime, including while in motion in closed-loop control. The method by which this offset is performed depends on the axis type:

- **Incremental axes:**
  The offset is done by adjusting the accumulated Actual Position and the Target and Command Positions.
- **Absolute linear axes:**
  The offset is done by adjusting the Position Offset parameter, which effectively adjusts the Actual Position, and by adjusting the Target and Command Positions by the same amount.
- **Absolute rotary axes:**
  The offset is done by adjusting the Count Offset parameter by an amount that will result in the Actual Position changing by the requested amount. The Target and Command Positions will be adjusted by the same amount.

The Set Actual Position (49) command is much more commonly used than the Offset Position command. If you need to simply change the current Actual Position, use the Set Actual Position (49) command instead. Notice also that the Position Offset or Count Offset parameter can be set directly as well by writing directly to that register. However, if they are written to directly, the axis must be in the Direct Output state.

See Also

Set Actual Position (49) | Set Target Position (48) | List of Commands | Commands Overview

8.6.2. Command: Set Target Position (48)

Supported Axes: Position Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Position** (pos-units)	any

Description

This command sets the Target Position to the value specified by the **Requested Position** parameter, and adjusts the Actual Position by the same amount. This can be done even while the axis is in motion, without any adverse effects.

The method of adjusting the Actual Position is axis dependent:

- **Absolute axes:** the Position Offset parameter is adjusted.
- **Incremental axes:** the Actual Position is adjusted directly.

On a rotary axis, if the Requested Position is outside the unwind limits, a command error will be generated.

See Also

Offset Position (47) | Set Actual Position (49) | List of Commands | Commands Overview

8.6.3. Command: Set Actual Position (49)

Supported Axes:	Position Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Requested Position** (pos-units)	any

Description

This command sets the Actual Position to the value specified by the **Requested Position** parameter, and adjusts the Target Position by the same amount. This can be done even while the axis is in motion, without any adverse effects.

The method of adjusting the Actual Position is axis dependent:

- **Absolute axes:** the Position Offset parameter is adjusted.
- **Incremental axes:** the Actual Position is just an accumulation, so it is simply adjusted directly.

See Also

Offset Position (47) | Set Target Position (48) | List of Commands | Commands Overview

8.6.4. Set Actual Pressure/Force (65)

Supported Axes:	Pressure, Force, Acceleration Axes
Firmware Limitations:	RMC75/150: 3.60 or newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Pressure/Force** (Pr or Fr)	any

Description

This command sets the Actual Pressure, Actual Force, or Actual Acceleration to the value specified by the **Requested Pressure/Force** parameter. If the axis is in pressure/force control or pressure/force limit is enabled, then the Target Pressure/Force and Command Pressure/Force will be adjusted by the same amount that the Target Pressure/Force changed. This command can be issued even while the axis is controlling pressure or force, without any adverse effects.

The method of adjusting the Actual Pressure/Force is axis dependent:

- **Single-Input Pressure/Force/Acceleration:** The Pressure/Force Offset parameter is adjusted.
- **Dual-Input Force/Acceleration:** The Channel A Force Offset or Channel A Acceleration Offset parameter is adjusted.

Advanced

In simulate mode, the behavior of this command is peculiar. This command will change the **Pressure/Force axis parameter**, but the **Actual Pressure/Force** remains unchanged unless the offset is large enough that the counts are truncated to a valid range (i.e. +/- 10.1 volts).

This is a result of how simulate mode behaves: it first calculates a simulator force based on position, then calculates the Counts required to generate the force (for differential force, Pressure B is assumed to be zero), then limits the counts to a valid range.

See Also

List of Commands | Commands Overview

8.6.5. Command: Set Pos/Vel Ctrl Mode (68)

Supported Axes:	Position Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description
1	**Control Mode** **Values** / **Control Mode** 0 – Position PID 1 – Position I-PD 4 – Velocity PID 5 – Velocity I-PD

Description

This command selects the closed loop control mode to be used when the next position or velocity motion command is issued. This command updates the Next Pos/Vel Control Mode register. The Current Control Mode register indicates the current control mode.

This command is an *immediate* command. Each step in a user program can have a maximum of one non-immediate command per axis. There is no limit to the number of immediate commands in a single step of a user program.

See Also

Next Pos/Vel Control Mode | Current Control Mode | Default Pos/Vel Control Mode | List of Commands | Commands Overview

8.6.6. Command: Feed Forward Adjust (69)

Supported Axes:	Position and Velocity Control Axes
Supported Control Modes:	Position PID, Velocity PID

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command is used to automatically set the Velocity Feed Forward values for position or velocity axes. After a closed-loop move is made where the axis reaches constant velocity with zero acceleration and jerk, issuing this command will set the Feed Forward for the direction last moved. The new Feed Forward value will be applied immediately. This command will have no effect if the Output Saturated bit is set or the axis is pressure or force limited.

This command can also be used when manually tuning an axis, but is not as accurate as autotuning. If the valve has deadband, this command may be more accurate than autotuning. This command can also be used during machine operation to adjust the Velocity Feed Forwards for changing system dynamics.

This command should be issued *after* a move has completed. Since the new Feed Forward value is applied immediately, issuing this command during motion may cause a sudden jerk, which may cause the output to saturate and the axis to halt.

This command should be used only after smooth motion. If the feedback is noisy, or the axis oscillates or does not reach steady state during the move, this command will give erroneous results. If the Output Bias is not adjusted correctly, the value for the Feed Forward may also be incorrect.

The result of this command will be logged in the Event Log.

Advanced Details

While the target profile is at constant velocity in Position PID or Velocity PID control, a copy of the current PFID Output and Target Velocity is saved each loop time. Notice that both data items are independent of the various output stages, including deadband, bias, ratio, and output polarity. If any of the following is true at the time of the capture, then the captured values are considered invalid:

- The Output Saturated bit is set.
- The axis is currently P/F limited.
- The sign of the PFID Output and Target Velocity are not the same.

When the Feed Forward Adjust command is issued, the appropriate Feed Forward gain will be set to the last-captured PFID Output divided by the last-captured Target Velocity.

See Also

Velocity Feed Forward

8.6.7. Command: Integrator Adjust (70)

Supported Axes:	All Axes
Supported Control Modes:	Position PID, Velocity PID, Pressure/Force

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Integral Output** (%)	-100% to 100%
2	**Integrator Select** • Active (0) • Pos/Vel (1) • Prs/Frc (2)	A valid integer as described.

Description

This command adjusts the Integral Output Term. The **Integral Output** value will be applied immediately to the Integral Output Term specified by the **Integrator Select** command parameter. Zeroing the Integral Output Term is useful if it is has undesirably wound up for some reason. This command is ignored if the Current Integrator Mode is set to "Always Zero".

The **Integral Output** value is given in percent of maximum Control Output, which is normally 10V. Therefore, 0% means clear the integrator, 100% means 10V, and -100% means -10V.

The **Integrator Select** command parameter selects which integrator to apply the new value to. Typically, this should be set to **Active (0)**, to apply to the current integrator. The other options are only necessary for Pressure/Force Limit in Position PID and Velocity PID modes.

- **Active (0)**
  Apply the **Integral Output** to the current integrator. If the axis is in PosPID with PFLimit or VelPID with PFLimit, then both integrators will be set to the specified value, which is only recommended when the value is zero.
- **Pos/Vel (1)**
  Apply the integrator value to the position or velocity integrator. This will work for Position PID and Velocity PID (with or without Pressure/Force Limit enabled). It will not affect Open Loop, P/F Control, nor the I-PD modes.
- **Prs/Frc (2)**
  Apply the integrator value to the Pressure/Force integrator. This will work for Position PID, Velocity PID, and Open Loop modes, if P/F Limit is enabled, plus in the Pressure/Force PID control mode. It has no effect if Pressure/Force Limit and Pressure/Force Control are disabled or if in one of the I-PD modes.

The Integral Output Term cannot be adjusted in any of the I-PD modes whether or not Pressure/Force Limit enabled.

The Integral Output Term can also be manipulated with the Set Integrator Mode (71) command and the Default Integrator Mode axis parameter.

Why Bother?

Sometimes, the Integral Output Term may wind up, which can cause problems. For example, one customer wanted to move a cylinder until it hit an object, and then back up 1 inch. The problem was that when the cylinder hit the object, the Integral Output Term wound up, causing the cylinder to delay after it was commanded to move backward. By clearing the integrator immediately before issuing the command to back up, the problem was solved.

See Also

Integral Gain | Integral Output Term

8.6.8. Command: Set Integrator Mode (71)

Supported Axes: Control Axes

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description
1	**Mode** • Held (0) • Active (1) • Zero (2)* • TGDone (3)* • Decel (4)* *Modes 2-4 are available in RMC75/150 firmware 3.66.0 and newer, and RMC200 firmware 1.05.0 and newer.

Description

This command sets the Current Integrator Mode, which defines when the Integral Gain is active. This command allows the Integrator Mode to be changed at any time. The Integrator Mode applies to both the primary and secondary control loops.

For details on the integrator modes, see the Default Integrator Mode topic.

Note:
For RMC75/150 firmware versions prior to 3.66.0, this command applied the integrator mode to the Default Integrator Mode axis parameter. This does not occur with later firmware versions or on the RMC200.

See Also

Integrator Mode | List of Commands | Commands Overview | Integrator Adjust (70) Command

8.6.9. Command: Select Gain Set (75)

Supported Axes:	Position and Velocity Control Axes
Supported Controllers:	RMC200

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Gain Set:** 0, 1	0,1

Description

If the Gain Sets axis parameter has been set to dual gain sets, this command selects the gain set to be used. The current gain set will change as soon as this command is received. The Current Gain Set axis status register displays the current gain set.

Dual gain sets may be used on systems where the response changes significantly at some point, such that different gains are required. For example, some presses may use a large valve for a portion of the stroke, then switch to a much smaller valve toward the end of the stroke. This requires different gains to be used. The Gain Sets feature is a convenient method of quickly switching gains when switching valves.

Status Bits

In Position Bit

When the Target Position reaches the **Requested Position** and the Actual Position is within the In Position Tolerance window, the In Position Status bit will be set. This bit indicates that the move is complete and the axis is at position.

Target Generator Bits

The Target Generator bits in the Status Bits register indicate which portion of the move the axis is currently in. These bits are useful when programming complex motion sequences.

Target Generator Done bit

This bit indicates that the Target Position has reached the **Requested Position**. Notice that this bit does not indicate whether the Actual Position has reached the Requested Position.

Target Generator State A and B bits

B	A	Description
0	0	The target generator is complete
0	1	Acceleration
1	0	Constant Velocity
1	1	Deceleration

See Also

Gain Sets Overview | Current Gain Set

8.6.10. Command: Set Control Direction (96)

Supported Axes:	Position, Velocity, Pressure, and Force Control Axes
Supported Control Modes:	Position PID, Position I-PD, Velocity PID, Velocity I-PD, Pressure/Force Control

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Direction** • Positive (1) • Negative (-1)	a valid integer as described

Description

This command, together with the Unidirectional Mode axis parameter, is specifically designed for systems that require a unipolar Control Output, but need to switch direction externally. This command is only valid when Unidirectional Mode is enabled—that is, set to Positive, Negative, or Automatic.

If the Unidirectional Mode axis parameter is set to Positive or Negative, issue the Set Control Direction (96) command at any time to set the desired direction.

If the Unidirectional Mode axis parameter is set to Automatic, the direction is determined based on the Target Velocity. The Set Control Direction (96) command may still be used while the axis is stopped to indicate a change in the system direction prior to a move command. Notice that issuing this command will have no effect in Automatic mode while the Target Velocity is not stopped.

For many systems, this command is unnecessary, because the Automatic option handles direction changes very well for most two-valve systems, and for a unidirectional belt, the direction is set by the Unidirectional Mode parameter on startup and never needs to be changed after that.

This command can be issued while the axis in closed loop control.

For more details, see the Unidirectional Mode topic.

See Also

Unidirectional Mode

8.6.11. Command: Read Register (111)

Supported Axes:	All
Supported Controllers:	RMC75/150

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Address** **Note:** See **Specifying a Register Address** below.	unsigned integer

Description

Note: Delta does not recommend using this command. PLCs can read any register directly without needing to send this command. In a user program, use the Expression (113) command to directly access registers.

This command copies the register pointed to by the **Address** into the axis' Read Response register.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Address** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

Why Bother?

This command is used with certain communication types. If the communications is set up to continuously return the Read Response register, then issuing this command will allow you to read whatever register you want. This is a good way of accessing more registers for communication types that are set up to only return certain registers.

See Also

Write Register (112) | List of Commands | Commands Overview

8.6.12. Command: Write Register (112)

Supported Axes:	All
Supported Controllers:	RMC75/150

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

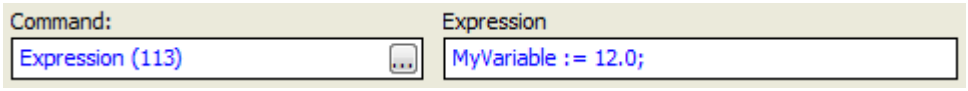
#	Parameter Description	Range
1	**Address** **Note:** See **Specifying a Register Address** below.	unsigned integer
2	**Value**	

Description

Note: Delta does not recommend using this command. From a PLC, you can write to any register directly without needing to send this command. In a user program, use the Expression (113) command to write directly to registers.

This command writes the value specified by the **Value** parameter into the address specified by the **Address** parameter.

This command is not intended to be used in user programs. Use the Expression (113) command instead. For example, the following expression will write a value of 12 to the variable **MyVariable**:



Notice that this command is not necessary for indirect addressing in expressions, as the REG_REAL(file, elem) and related functions are the preferred method of indirect addressing.

Specifying a Register Address

When issuing this command from anywhere other than RMCTools, the addresses in the **Address** command parameter must be entered as an integer value.

RMC addresses are represented in IEC format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example:

Register address %MD8.33 is 8*4096 + 33 = 32801.

See Also

Read Register (111) | List of Commands | Commands Overview

8.7. System

8.7.1. Command: Arm Home (50)

Supported Axes:	Quadrature, SSI, Resolver

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

<table>
<tr><th>#</th><th>Parameter Description</th><th>Range</th></tr>
<tr><td>1</td><td>Home Position (position-units)</td><td>any</td></tr>
<tr><td>2</td><td>Trigger Type
<table>
<tr><th rowspan="2">#</th><th rowspan="2">Name</th><th colspan="3">Available on</th></tr>
<tr><th>RMC75</th><th>RMC150</th><th>RMC200</th></tr>
<tr><td>0</td><td>H Rising</td><td>QAx, Q1, SSI[1]</td><td>Quad, UI/O</td><td>Q4, D24, U14</td></tr>
</table></td><td>Integers 0 to 7</td></tr>
</table>

	1	H Falling	QAx, Q1, SSI[1]	Quad, UI/O	Q4, D24, U14
	2	Z	QAx, SSI[1]	Quad, SSI[1], Resolver[1]	Q4, D24[2], U14
	3	Z And H	QAx, SSI[1]	Quad	Q4, D24[2], U14[3]
	4	Z And Not H	QAx, SSI[1]	Quad	Q4, D24[2], U14[3]
	5	Absolute Adjust (H Rising)	SSI[1]		
	6	Absolute Adjust (H Falling)	SSI[1]		
	7	Absolute Adjust (Immed)	SSI[1]	SSI[1], Resolver[1]	

[1]Valid only for axes configured as Incremental.

[2]Valid only for the D24 option One Quadrature Input (A,B,Z).

[3]The Trigger Types of **Z And H (3)** and **Z And Not H (4)** cannot be used when the Home Input command parameter is set to **Reg/Z0 (0)** or **Reg/Z1 (0)**.

3 — **Repeat Mode:** Single (0), Repeat(1) — 0 or 1

4 — **Home Input:**

RMC75:

#	QA1, QA2	MA1, MA2 SSI Homing
0	Home	Specified by the SSI Home Source parameter.

RMC150:

	Quad	UI/O
0	Home	Channel 0: R0 Channel 1: R1

RMC200 Q4:

#	Inputs
0	Hm
1	Reg

RMC200 U14:

#	Channel 0	Channel 1
0	Reg/Z0[1]	Reg/Z1[1]
1	D0[2]	D1[2]

[1] The Trigger Types of **Z And H (3)** and **Z And Not H (4)** cannot be used when the Home Input is set to **Reg/Z0 (0)** or **Reg/Z1 (0)**.

Integers 0 to 3

[2] Inputs D0 and D1 are not high-speed inputs, and will provide slower homing response than the Reg/Z inputs.

RMC200 D24:

#	One quadrature input (A,B,Z)	One quadrature input (A,A,B,B) with wire break detection	Two quadrature inputs, D20&D21	Two quadrature inputs, D22&D23
0	D22	D18	D18	D16
1	D23	D19	D19	D17
2	D18	n/a	n/a	D18
3	D19	n/a	n/a	D19

Note: Inputs D16-D19 are not high-speed inputs, and will provide slower homing response than inputs D22-23.

Description

This command enables the trigger for homing the axis. This command will clear the Home Latched status bit, and set the Home Armed status bit. The home will remain armed until the home trigger condition occurs or the Disarm Home (51) command is issued.

If a home trigger condition occurs when the home is armed, the following occurs:

- The Home Latched bit is set. Notice that in **Repeat** mode, the Home Latched bit may already have been set.
- The Actual Position is set to the Home Position, with the Target and Command Positions, if available on this axis type, being adjusted the same amount. The offset is done by adjusting the accumulated Actual Position and the Target and Command Positions.
- Unless this command was issued in Repeat mode, the Home Armed status bit is cleared.
- The Home event is logged in the Event Log.

The **Trigger Types** command parameter provides numerous options for triggering a home. The most common **Trigger Type** is **H Rising** (rising edge of the Home input) or **H Falling** (falling edge of the Home input). See the Homing for more details on what the trigger types mean.

Errors

If the Index (Z) Wire Break error bit is on at any time while the home is armed, it will cause an axis Runtime error.

See Also

List of Commands | Commands Overview | Homing | Disarm Home (51) Command

8.7.2. Command: Disarm Home (51)

Supported Axes: Quadrature, SSI, Resolver

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

This command has no command parameters.

Description

This command disarms the home, if it was armed. This command clears the Home Armed status bit.

See the Homing topic for details on homing.

See Also

List of Commands | Commands Overview | Homing | Arm Home (50) Command

8.7.3. Command: Arm Registration (52)

Supported Axes:	Quadrature

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

<table>
<tr><th>#</th><th>Parameter Description</th><th>Range</th></tr>
<tr><td>1</td><td>Registration Number</td><td>0 or 1</td></tr>
<tr><td>2</td><td>Registration Input:

RMC75:
<table>
<tr><th>#</th><th>Q1 Input</th><th>QAx Input</th></tr>
<tr><td>0</td><td>Reg</td><td>RegX/PosLim</td></tr>
<tr><td>1</td><td>n/a</td><td>RegY/NegLim</td></tr>
</table>
RMC150:
<table>
<tr><th></th><th>Quad Input</th><th>UI/O Quad Input</th></tr>
<tr><td>0</td><td>RegX/PosLim</td><td>R0</td></tr>
<tr><td>1</td><td>RegY/NegLim</td><td>R1</td></tr>
</table>
RMC200 Q4:
<table>
<tr><th>#</th><th>Channel 0</th><th>Channel 1</th><th>Channel 2</th><th>Channel 3</th></tr>
<tr><td>0</td><td>Reg0</td><td>Reg1</td><td>Reg2</td><td>Reg3</td></tr>
<tr><td>1</td><td>Hm0</td><td>Hm1</td><td>Hm2</td><td>Hm3</td></tr>
<tr><td>2</td><td>Reg1</td><td>Reg0</td><td>Reg3</td><td>Reg2</td></tr>
</table></td><td>0-3</td></tr>
</table>

3	Hm1	Hm0	Hm3	Hm2

RMC200 U14:

#	Channel 0	Channel 1
0	Reg/Z0	Reg/Z1
1	Reg/Z1	Reg/Z0
2	D0	D0
3	D1	D1

Note: Inputs D0 and D1 are not high-speed inputs, and will provide slower registration response than the Reg/Z inputs.

RMC200 D24:

#	One quadrature input (A,B,Z)	One quadrature input (A,A,B,B) with wire break detection	Two quadrature inputs, using inputs D20&D21	Two quadrature inputs, using input D22&D23
0	D22	D18	D18	D16
1	D23	D19	D19	D17
2	D18	n/a	n/a	D18
3	D19	n/a	n/a	D19

Note: Inputs D16-D19 are not high-speed inputs, and will provide slower registration response than inputs D22-23.

3	**Input Edge:** Falling (0), Rising(1)	0 or 1

Description

This command enables a position registration to occur on the specified input on the specified edge.

The **Registration Number** parameter specifies which registration this will apply to, either 0 or 1. The **Registration Input** parameter specifies which input will be used for the registration.

When this command is issued the following will occur:

- The armed status bit of the selected Registration Number (0 or 1) will be set (Registration 0 Armed or Registration 1 Armed).
- The latched status bit of the selected Registration Number (0 or 1) will be cleared (Registration 0 Latched or Registration 1 Latched).

If the registration is armed and a registration event occurs, the exact position at the registration event will be latched and recorded in the Registration 0 Position or Registration 1 Position status register.

For details on registration, see the Registration topic.

See Also

List of Commands | Commands Overview | Registration | Disarm Registration (53) Command

8.7.4. Command: Disarm Registration (53)

Supported Axes:	Quadrature

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Registration Number**	0 or 1

Description

This command disarms the specified registration, if it was armed. This command clears the corresponding Registration Armed status bit (Registration 0 Armed and Registration 1 Armed).

For details on registration, see the Registration topic.

See Also

List of Commands | Commands Overview | Registration | Arm Registration (52) Command

8.7.5. Command: Learn Z Alignment (54)

Supported Axes:	Quadrature on the RMC75 QA, RMC150 Q, RMC200 Q4, RMC200 U14, and RMC200 D24 modules

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command instructs the quadrature axis to monitor the encoder for an Indez (Z) pulse in order to determine its alignment with relationship to the A and B pulses.

After the Learn Z Alignment command has been issued, the Learning Z Alignment status bit will be set and will remain set until the axis has learned the Index (Z) alignment. The axis will learn the Z alignment as soon as the Index (Z) pulse is encountered. Once the axis has learned the Z alignment, it will clear the Learning Z Alignment status bit, and will update the Index (Z) Home Location parameter.

Determining the Z alignment is important for accurate homing with the Index (Z) pulse in either direction of motion on a quadrature encoder. See the Index (Z) Home Location topic for details.

Using this Command During Setup

Typically, you only need to issue this command when setting up the axis:

1. Issue the Learn Z Alignment (54) command to the axis. The Learning Z Alignment status bit will turn on.

2. Rotate the encoder at least one full revolution so that the Learning Z Alignment status bit turns off. This indicates that the Index (Z) Home Location parameter has been set.
3. Update Flash to store the settings in Flash memory.

See Also

List of Commands | Commands Overview | Homing | Index (Z) Home Location

8.7.6. Command: Pause/Resume Log (95)

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Pause/Resume** • Pause (0) • Resume (1)	A valid integer as described.

Description

This command pauses or resumes the Event Log. When the Event Log is paused, it will not log any entries. When it is resumed, it will continue logging entries from the time it was resumed.

This command is useful for troubleshooting. A user program can wait for some event to occur and then pause the Event Log. Then later, the log can be uploaded via RMCTools and analyzed.

The number of entries the Event Log can hold is limited. If that limit is exceeded, the oldest entries disappear. Therefore, during normal execution of the Event Log, an error event may disappear before the log can be uploaded. Pausing the Event Log solves this dilemma.

This command is not the same as the **Pause** and **Resume** buttons in the Event Log Monitor toolbar. The Pause button in the Event Log Monitor stops uploading new Event Log items, although the Event Log is still logging. When the Event Log Monitor is paused, it will also try to backfill any missing items as much as possible. Because the Event Log is finite, the Event Log Monitor may not be able to backfill all its missing data.

When the Event Log is paused due to this command, the Error Icon and Error Bubble in RMCTools will not function.

See Also

Event Log Monitor

8.7.7. Command: Update Flash (110)

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

This information is not important when updating the Flash directly from RMCTools without issuing a command.

#	Parameter Description	Data Type
1	**Section to Store** - must be 0	Internal: DINT External: REAL

Description

This command stores all of the RMC controller data to Flash memory for storage in case of power loss. The RMC will continue to control motion as usual during a Flash update. On the RMC75/150, the green CPU LED will flash while a Flash update is in progress. For RMC75/150's with firmware prior to 3.30.0, if power is lost during a Flash update, all the data in the RMC may be lost.

Note:
You must Update Flash in order to save controller data, or it will be lost when or power is removed from the RMC!

Updating Flash from RMCTools

You can update Flash directly from RMCTools without using this command. See the Updating Flash topic for details. You may also be prompted to save to Flash when the controller is warm restarted after changing certain controller data.

Special Note

The Flash memory chips on the RMC75S, RMC75P, and RMC200 are limited 100,000 update cycles. The RMC75E and RMC150E are limited to 1 million update cycles. Due to these limits, you should take care to program the RMC so that this limit will not be exceeded during the life of the controller.

See Also

Updating Flash | List of Commands | Commands Overview

8.7.8. Command: Arm Event Timer (105)

Supported Axes:	All
Firmware Limitations:	RMC75/150: n/a RMC200 CPU: 1.06.0 and newer RMC200 D24: 1.04A and newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Module Slot** The number of the slot that contains the module with the high-speed input to be used for event timing.	A valid integer slot number from 2 to the maximum

<table>
<tr><td></td><td>The first I/O module slot number on the RMC200 is slot 2.</td><td>number of slots in the base.</td></tr>
<tr><td>2</td><td>Event Timer
The event timer number on the module. This is the internal software resource for the timer. See Event Timers for details.</td><td>D24: 0-3</td></tr>
<tr><td>3</td><td>Event Source
The physical event which will cause the timer to latch.
Quad Count (0)
In0 Rising Edge (1)
In0 Falling Edge (2)
In1 Rising Edge (3)
In1 Falling Edge (4)
D24 Hardware Inputs:<table><tr><th>Event Timers</th><th>Event Source</th><th>Hardware Input</th></tr><tr><td rowspan="5">0 or 1</td><td>Quad Count (0)</td><td>Quad Channel 0</td></tr><tr><td>In0 Rising Edge (1)</td><td>D20</td></tr><tr><td>In0 Falling Edge (2)</td><td>D20</td></tr><tr><td>In1 Rising Edge (3)</td><td>D21</td></tr><tr><td>In1 Falling Edge (4)</td><td>D21</td></tr><tr><td rowspan="5">2 or 3</td><td>Quad Count (0)</td><td>Quad Channel 1</td></tr><tr><td>In0 Rising Edge (1)</td><td>D22</td></tr><tr><td>In0 Falling Edge (2)</td><td>D22</td></tr><tr><td>In1 Rising Edge (3)</td><td>D23</td></tr><tr><td>In1 Falling Edge (4)</td><td>D23</td></tr></table></td><td>A valid integer as described.</td></tr>
<tr><td>4</td><td>Timer Mode
One-Shot (0)
Continuous (1)
Alternating (2)</td><td>A valid integer as described.</td></tr>
</table>

Description

This command arms an Event Timer. Event Timers are used to capture the time of a high-speed input event. For details on using Event Timers, see the Event Timers topic.

Slot number
Specifies which slot number of the base contains the module with the high-speed input.

Base	Valid I/O Module Slot Numbers
B5L	2–4
B7L	2–6
B5	2–4
B7	2–6
B11	2–10
B15	2–14

Event Timer and Event Source

The Event Timer is the internal software resource for the timer, not the actual hardware input. The Event Source defines the hardware input to be used. Each Event Timer supports certain Event Sources as listed below:

Event Timers	Event Source Names	Hardware Input
0 or 1	Quad Count: Any change in the quadrature count, useful for calculating velocity from last two counts.	Quadrature Channel 0
	In0 Rising Edge, In0 Falling Edge	Discrete Input D20
	In1 Rising Edge, In1 Falling Edge	Discrete Input D21
2 or 3	Quad Count: Any change in the quadrature count, useful for calculating velocity from last two counts.	Quadrature Channel 1
	In0 Rising Edge, In0 Falling Edge	Discrete Input D22
	In1 Rising Edge, In1 Falling Edge	Discrete Input D23

Timer Modes

The following modes are available:

1. **One-Shot**
   The Event Timer will capture the time of the first occurrence of the event after the timer is armed.
2. **Continuous**
   The Event Timer will continue to capture the time of each occurrence of the event after the timer is armed. The time of the most recent event will be stored in the Event Timer registers.
3. **Alternating**
   The Event Timer will capture the time of every other occurrence of the event after the timer is armed.

See Also

Event Timers | Disarm Event Timer (106)

This never appears:

Event Timers	Event Source Names	Hardware Input
0, 1	Quad Count	Quadrature Channel 0
0, 1	In0 Rising Edge, In0 Falling Edge	Discrete Input D20
0, 1	In1 Rising Edge, In1 Falling Edge	Discrete Input D21
2, 3	Quad Count	Quadrature Channel 1
2, 3	In0 Rising Edge, In0 Falling Edge	Discrete Input D22
2, 3	In1 Rising Edge, In1 Falling Edge	Discrete Input D23

Event Source	Event Timers	Hardware Input
Quad Count	0, 1	Quadrature Channel 0
Quad Count	2, 3	Quadrature Channel 1
In0 Rising Edge, In0 Falling Edge	0, 1	D20
In1 Rising Edge, In1 Falling Edge	0, 1	D21
In0 Rising Edge, In0 Falling Edge	2, 3	D22
In1 Rising Edge, In1 Falling Edge	2, 3	D23

Name	Event Timers	Hardware Input
In0	0, 1	D20
In1	0, 1	D21
In0	2, 3	D22
In1	2, 3	D23

8.7.9. Command: Disarm Event Timer (106)

Supported Axes:	All
Firmware Limitations:	RMC75/150: n/a RMC200 CPU: 1.06.0 and newer RMC200 D24: 1.04A and newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range

1	**Module Slot** The number of the slot that contains the module with the high-speed input to be used for event timing. The first I/O module slot number on the RMC200 is slot 2.	A valid integer slot number from 2 to the maximum number of slots in the base.
2	**Event Timer** The event timer number on the module. This is the internal software resource for the timer. See Event Timers for details.	D24: 0-3

Description

This command disarms a previously armed Event Timer. See Event Timers for details.

See Also

Event Timers | Arm Event Timer (105)

8.7.10. Command: Save Controller Image (120)

Firmware Limitations:	RMC200: 1.10.0 or newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command saves the controller image to the RMC200 SD card. The controller image will be saved on the SD card as a file named **ControllerImage.bin** in the root folder.

The Event Log will indicate that the controller image was saved to the SD card, or will provide an error message if it failed.

Controller Image Command State Register

The Controller Image Command State Register (%MD18.13) provides useful information when this command is used.

The register indicates the status of the most recently issued Save/Restore Controller Image command:

Value (DINT)	Status
0	No action requested. This will be the value after startup, including after an image is successfully restored and the controller is restarted.
10	Building or saving controller image.
11	Controller image saved successfully.
20	Loading or applying controller image.
21	Image successfully restored to controller. Controller will hold this state for 500 ms, then automatically restart.

30	Unable to start command since an SD card exclusive access session was active.
31	Unable to start command since another SD card operation was in progress.
40	Save image failed due to an internal error.
41	Save image failed due to an SD card write error.
42	Save image failed because no SD card is installed.
43	Save image failed because the installed SD card is incompatible.
44	Save image failed because the controller is copy protected.
45	Save image failed because the installed SD card is not writable.
46	Save image failed because the controller image could not be built.
50	Restore image failed due to an internal error.
51	Restore image failed due to an SD card read error.
52	Restore image failed because no SD card is installed.
53	Restore image failed because the installed SD card is incompatible.
54	Restore image failed because the controller image file was invalid.
55	Restore image failed because the controller image could not be applied.
56	Restore image failed because no controller image file was found on the SD card.

See Also

Controller Image Upload/Download | Restore Controller Image (121)

8.7.11. Command: Restore Controller Image (121)

Firmware Limitations:	RMC200: 1.10.0 or newer

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

None.

Description

This command restores the controller image from the RMC200 SD card to the controller and then restarts the controller 500 msec after the image was restored. Restoring the image may take up to several seconds. The controller image must exist as a file on the SD card named **ControllerImage.bin** in the root folder. The Event Log will provide an error message if the restore failed.

Controller Image Command State Register

The Controller Image Command State Register (%MD18.13) provides useful information when this command is used.

The register indicates the status of the most recently issued Save/Restore Controller Image command:

Value (DINT)	Status
0	No action requested. This will be the value after startup, including after an image is successfully restored and the controller is restarted.
10	Building or saving controller image.
11	Controller image saved successfully.
20	Loading or applying controller image.
21	Image successfully restored to controller. Controller will hold this state for 500 ms, then automatically restart.
30	Unable to start command since an SD card exclusive access session was active.
31	Unable to start command since another SD card operation was in progress.
40	Save image failed due to an internal error.
41	Save image failed due to an SD card write error.
42	Save image failed because no SD card is installed.
43	Save image failed because the installed SD card is incompatible.
44	Save image failed because the controller is copy protected.
45	Save image failed because the installed SD card is not writable.
46	Save image failed because the controller image could not be built.
50	Restore image failed due to an internal error.
51	Restore image failed due to an SD card read error.
52	Restore image failed because no SD card is installed.
53	Restore image failed because the installed SD card is incompatible.
54	Restore image failed because the controller image file was invalid.
55	Restore image failed because the controller image could not be applied.
56	Restore image failed because no controller image file was found on the SD card.

See Also

Controller Image Upload/Download | Save Controller Image (120)

8.8. Programming

8.8.1. Command: Start Task (90)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range

1	**Task Number** - Specifies which task to run the User Program on.	any valid task number
2	**Program Number -** Specifies the User Program to start at. This must be the number of the User Program.	any valid User Program number

Description

This command starts running the specified User Program on the specified Task. If the specified Task is currently running a User Program, the Task will first be stopped, then it will start immediately at the specified User Program.

The RMC must be in Run Mode in order to run tasks.

Note:
To stop a Task, see the Stop Task (91) command.

Tip:
After starting a User Program on a task, use the Task Monitor to monitor the status of the task. It shows which step on which User Program is running on each task.

Task Numbers

The tasks are numbered beginning with 0. The RMC75 has a maximum of 4 tasks, the RMC150 has a maximum of 10 tasks, and the RMC200 has a maximum of 32 tasks.

The number of actual tasks is specified on the **General** page of the Programming Properties dialog. The **Task Number** parameter must be within the number of tasks allocated on the Programming Properties dialog.

Details

The following items are important when issuing a Start Task command:

- **Selecting which Task to run**

  When you issue the Start Task command, you must specify which task to start. What happens depends on what the task is doing when you issue the Start Task command:

  - **If the task is stopped when you issue the Start Task command:**
    The task will simply start running the User Program you specified.
  - **The task is already running when you issue the Start Task command:**
    The task will stop the User Program it is already running and immediately start at the User Program you specified.

- **Selecting the User Program**

  The second command parameter, **Program Number**, specifies which User Program the Task will start running.

- **Which axis to issue the Start Task command to**

  In the User Program that you will start, if you have specified for each step which axis the command will be issued to, then it does not matter which axis you issue the Start Task command to. If you have *not* specified which axis the command in that step will be issued to, then the command will be issued to the axis which the Start Task command was issued to.

Issuing Start Task from a PLC

When issuing the Start Task command from a PLC, the second Command Parameter, Program, requires the Program *number*. The number for each User Program is listed in the Project Pane in the User Programs node.

What is a Task?

Tasks are for running User Programs. Each task is an execution engine that can run one User Program at a time. The RMC75 has up to four tasks. Therefore, the RMC75 can run up to four User Programs simultaneously. The RMC150 has up to ten tasks and can run up to ten User

Programs simultaneously. The RMC200 CPU20L has 32 tasks and can run up to 32 User Programs simultaneously whereas the CPU40 has 64 tasks and can run up to 64 User Programs. See the Tasks topic for more details.

Valid Task Numbers

You can choose a task number up to the number of task that have been made available. The default number of tasks is smaller than the maximum number of tasks available. To increase the number of tasks, use the Programming Properties dialog.

See Also

Stop Task (91) Command | Tasks Overview | RUN/PROGRAM Mode | List of Commands | Commands Overview

8.8.2. Command: Stop Task (91)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Task Number** - Specifies which task to stop.	any valid task number

Description

This command stops the specified task. Any User Programs running on the task will be stopped.

To start a task, see the Start Task (90) command.

What is a Task?

Tasks are for running User Programs. Each task is an execution engine that can run one User Program at a time. The RMC75 has up to four tasks. Therefore, the RMC75 can run up to four User Programs simultaneously. The RMC150 has up to ten tasks. Therefore, the RMC150 can run up to ten User Programs simultaneously. See the Tasks topic for more details.

Valid Task Numbers

You can choose a task number up to the number of task that have been made available. The default number of tasks is smaller than the maximum number of tasks available. To increase the number of tasks, use the Programming Properties dialog.

See Also

Start Task (90) Command | Tasks Overview | List of Commands | Commands Overview

8.8.3. Command: Set Discrete Output (60)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**I/O Point**	any valid discrete output

Description

This command turns on the specified discrete output. To use this command, you must have defined a discrete I/O point to be an output using the Discrete I/O Configuration dialog.

For details on settings multiple outputs simultaneously, see the Using Discrete I/O topic.

This command is an *immediate* command. There is no limit to the number of immediate commands in a single step of a user program. Each step in a user program can have a maximum of one non-immediate command per axis.

I/O Point

When issuing this command from RMCTools, the **I/O Point** is the tag name of the discrete output, if it has a name, otherwise, the **I/O Point** will be the address of the discrete output.

Use the Discrete I/O Monitor to find the address or tag name of a particular I/O point, or to assign a tag name to an I/O point. The address number appears in the icon of each discrete I/O point in the I/O monitor.

I/O Point from a Host Controller

When issuing this command from a host controller, such as a PLC or HMI, the **I/O Point** must be specified in integer format. The table below describes how to determine the integer value of a discrete output.

RMC75	RMC150/RMC200
Integer Number of a Discrete Output Integer Number = address number The Discrete I/O Monitor also displays the integer number of each I/O point. **Examples** The integer number for output %QX2 is 2. The integer number for output %QX13 is 13.	**Integer Number of a Discrete Output** Integer Number = (32 x slot#) + I/O# The slot numbering starts with 0 for the left-most module in the RMC150 and RMC200. The Discrete I/O Monitor also displays the slot number and I/O number of each I/O point. For example, the I/O point 4 in slot 3 is displayed as 3.4. **Examples** The integer number for output 7 in slot 0 is: (32 x 0) + 7 = 7 The integer number for output 1 in slot 1 (the CPU slot) is: (32 x 1) + 1 = 33 The integer number for output 3 in slot 5 is: (32 x 5) + 3 = 163

See Also

Clear Discrete Output(61) | Toggle Discrete Output(62) | Discrete I/O Overview | List of Commands | Commands Overview

8.8.4. Command: Clear Discrete Output (61)

Supported Axes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**I/O Point**	any valid discrete output

Description

This command turns off the specified discrete output. To use this command, you must have defined a discrete I/O point to be an output using the Discrete I/O Configuration dialog.

For details on settings multiple outputs simultaneously, see the Using Discrete I/O topic.

This command is an *immediate* command. Each step in a user program can have a maximum of one non-immediate command per axis. There is no limit to the number of immediate commands in a single step of a user program.

I/O Point

When issuing this command from RMCTools, the **I/O Point** is the tag name of the discrete output, if it has a name, or the address of the discrete output.

Use the Discrete I/O Monitor to find the address or tag name of a particular I/O point, or to assign a tag name to an I/O point. The address number appears in the icon of each discrete I/O point in the I/O monitor.

I/O Point from a Host Controller

When issuing this command from a host controller, such as a PLC or HMI, the **I/O Point** must be specified in integer format. The table below describes how to determine the integer value of a discrete output.

RMC75	RMC150/RMC200
Integer Number of a Discrete Output Integer Number = address number The Discrete I/O Monitor also displays the integer number of each I/O point. **Examples**	**Integer Number of a Discrete Output** Integer Number = (32 x slot#) + I/O# The slot numbering starts with 0 for the left-most module in the RMC150 and RMC200. The Discrete I/O Monitor also displays the slot number and I/O number of each I/O point. For example, the I/O point 4 in slot 3 is displayed as 3.4.

The integer number for output %QX2 is 2. The integer number for output %QX13 is 13.	**Examples** The integer number for output 7 in slot 0 is: (32 x 0) + 7 = 7 The integer number for output 1 in slot 1 (the CPU slot) is: (32 x 1) + 7 = 39 The integer number for output 3 in slot 5 is: (32 x 5) + 3 = 163

See Also

Set Discrete Output (60) | Toggle Discrete Output(62) | Discrete I/O Overview |List of Commands | Commands Overview

8.8.5. Command: Toggle Discrete Output (62)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**I/O Point**	any valid discrete output number

Description

This command toggles the specified discrete output. If the output was on, this command will turn it off. If the output was off, this command will turn it on. To use this command, you must have defined a discrete I/O point to be an output using the Discrete I/O Configuration dialog.

For details on settings multiple outputs simultaneously, see the Using Discrete I/O topic.

This command is an *immediate* command. Each step in a user program can have a maximum of one non-immediate command per axis. There is no limit to the number of immediate commands in a single step of a user program.

I/O Point

When issuing this command from RMCTools, the **I/O Point** is the tag name of the discrete output, if it has a name, otherwise, the **I/O Point** will be the address of the discrete output.

Use the Discrete I/O Monitor to find the address or tag name of a particular I/O point, or to assign a tag name to an I/O point. The address number appears in the icon of each discrete I/O point in the I/O monitor.

I/O Point Number from a Host Controller

When issuing this command from a host controller, such as a PLC or HMI, the **I/O Point Number** must be specified in integer format. The table below describes how to determine the integer value of a discrete output.

RMC75	RMC150/RMC200

Integer Number of a Discrete Output	**Integer Number of a Discrete Output**
Integer Number = address number The Discrete I/O Monitor also displays the integer number of each I/O point. **Examples** The integer number for output %QX2 is 2. The integer number for output %QX13 is 13.	Integer Number = (32 x slot#) + I/O# The slot numbering starts with 0 for the left-most module in the RMC150 and RMC200. The Discrete I/O Monitor also displays the slot number and I/O number of each I/O point. For example, the I/O point 4 in slot 3 is displayed as 3.4. **Examples** The integer number for output 7 in slot 0 is: (32 x 0) + 7 = 7 The integer number for output 1 in slot 1 (the CPU slot) is: (32 x 1) + 7 = 39 The integer number for output 3 in slot 5 is: (32 x 5) + 3 = 163

See Also

Set Discrete Output (60) | Clear Discrete Output(61) | Discrete I/O Overview | List of Commands | Commands Overview

8.9. Plots

8.9.1. Command: Start Plot (100)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Plot Number**	any valid plot number

Description

This command starts a one-time plot. It erases the plot buffer and starts capturing a plot until the buffer is full. At that point, it stops capturing and waits until the plot is reset. Compare this with the Stop Plot (101) and Trigger Plot (102) commands.

To find out more about plotting, see the Using Plots topic.

See Also

Stop Plot (101) Command | Plot Overview | Triggering Plots | List of Commands | Commands Overview

8.9.2. Command: Stop Plot (101)

Supported Axes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Plot Number**	any valid plot number

Description

This command stops a continuous plot capture immediately. This allows RMCTools to read up the entire plot without any gaps. Plots are either continuously capturing data or done capturing data. Compare this with the Start Plot (100) and Trigger Plot (102) commands.

To find out more about plotting, see the Using Plots topic.

See Also

Start Plot (100) Command | Plot Overview | Triggering Plots | List of Commands | Commands Overview

8.9.3. Command: Trigger Plot (102)

Supported Axes:	All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Plot Number**	any valid plot number

Description

This command triggers a plot. Triggering a plot means to start a plot, and include plot data from *before* the moment at which plot was triggered. This is very useful for troubleshooting. For example, if a plot is triggered when an error occurs, the plot can show data from before the error occurred, giving the user insight as to what happened.

The amount of plot data included from before the trigger is defined by the *Pre-Trigger Percent* in the Plot Template Editor. The default Pre-Trigger Percent is 0%, which means no data from before the trigger will be included. In order to include data from before the trigger, the Pre-Trigger Percent must be greater than zero *and* the plot Rearm Mode in the Plot Template Editor must be set to Manual Rearm.

When the plot is set to Manual Rearm, the Rearm Plot (103) command must be issued before triggering the plot. Notice that rearming a plot will clear all previously captured data for that plot. If a plot is triggered immediately after rearming it, and the Pre-Trigger Percent is greater than 0%, the plot will not contain any data before the trigger. After rearming, there must be a delay long enough to record the data to fill the plot before the trigger time.

If the plot trigger has been disabled, this command will be ignored. The plot trigger is automatically disabled while in Trend Mode in the Plot Manager, or can be disabled with the Enable/Disable Plot Trigger (104) command.

This command differs from the Start Plot (100) commands in the following ways:

- The Start Plot (100) command will start a plot that only includes data from after the Start Plot command was issued.
- The Start Plot (100) command starts a new plot immediately, even if the specified Plot number is currently capturing a plot. The Trigger Plot (102) command will not trigger a plot if the specified Plot Number is currently capturing a plot.

Plots can also be triggered automatically when a motion command is issued to an axis. To find out more on triggering, see the Triggering Plots topic.

See Also

Plot Overview | Triggering Plots | List of Commands | Commands Overview

8.9.4. Command: Rearm Plot (103)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Plot Number**	any valid plot number

Description

This command rearms a plot so that it can be triggered. If a plot is set to automatically rearm, this command is unnecessary. If a plot is set to manual trigger, then this command must be issued before the plot is triggered. If the Plot Trigger Percentage is non-zero, then make sure to issue this command well before triggering the plot, or you may not capture enough pre-trigger data.

Rearming a plot will clear all previously captured data for that plot.

To find out more about rearming, see the Triggering Plots topic.

Note:
If a plot is triggered immediately after rearming it, and the trigger position is greater than 0%, the

plot will not contain any data before the trigger. After rearming, there must be a delay long enough to record the data to fill the plot before the trigger time.

Details

The Rearm Plot command can be issued from the **Online** menu in the Plot Manager.

See Also

Plot Overview | Triggering Plots | List of Commands | Commands Overview

8.9.5. Command: Enable/Disable Plot Trigger (104)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description	Range
1	**Plot Number**	any valid plot number
2	**Enable/Disable:** Disable(0), Enable(1)	0 or 1

Description

This command enables or disables the trigger for a plot. The automatic trigger and any triggers from the Trigger Plot (102) command will be disabled or enabled. The automatic trigger can only be disabled and enabled if it was already defined in the plot template.

To find out more on triggering, see the Triggering Plots topic.

See Also

Plot Overview | Triggering Plots | List of Commands | Commands Overview

8.10. Step Editor Commands

8.10.1. Command: Expression (113)

Supported Axes: All

See the Commands Overview topic for basic command information and how to issue commands from PLCs, HMIs, etc.

Command Parameters

#	Parameter Description

1	Expression

Description

This command can only be used in User Programs. It runs the Expression as entered by the user. Each expression command can contain multiple assignments, multiple lines, branching using IF statements, local variables, and can include comments.

This command is very versatile and can be used for such things as:

- Mathematical calculations.
- Assign values to registers, such as variables, parameters and discrete I/O.
- Turn discrete outputs on.
- Get the state of discrete inputs or outputs.
- Copy data from one location in the variable table to another location in the variable table. See the COPY function for details.

Step-Local Variables

The Expression (113) command can be used together with step-local variables. These are variables that are declared within a user program step, and can only be accessed from that step and are valid only for the execution of the step. For complex expressions, using local variables can significantly reduce the number of variables needed in the variable table.

Examples

Example 1

Adds 2 to the square root of Num2 and assigns the result to Num1.

```
// Calculate 2 plus the square root of Num2, and save it in Num1.
Num1 := 2.0 + SQRT(Num2);
```

Example 2

The first line adds 2.56 to the Actual Position and assigns the result to the Position Offset.

The second line divides the minimum value of Axis 0 or Axis 1 Actual Position by 2, adds 6, and assigns the result to the variable SampleVariable.

```
// Set the axis 0 Position Offset parameter to the current Actual
// Position plus 2.56.
_Axis[0].PosOffset:= _Axis[0].ActPos + 2.56;

// Calculate 6 plus half the minimum of the first two axes current Actual
// positions. Save this in SampleVariable.
SampleVariable := Min(_Axis[0].ActPos,_Axis[1].ActPos) / 2.0 + 6.0;
```

Example 3

An IF statement.

```
IF _Axis[0].StatusBits.InPos = True THEN
  MyREAL1 := 34.0;
  MyREAL2 := 70023.0;
```

```
ELSIF ABS(_Axis[0].ActPos) > 20.0 THEN
 %QX0.1 := True;
ELSE
 MyDINT := 2;
END_IF
```

Example 4

A COPY function. The variable i must be a DINT.

```
COPY(MyArray[0],MyOtherArray[i],20);
```

Example 5

Commenting a large section of code. You may wish to comment out a large section of code temporarily during development. The commented text will not be executed.

```
IF _Axis[0].StatusBits.InPos = True THEN
 MyREAL1 := 34.0;
 MyREAL2 := 70023.0;
(*
ELSIF ABS(_Axis[0].ActPos) > 20.0 THEN
 %QX0.1 := True;
 *)
ELSE
 MyDINT := 2;
END_IF
```

Entering an Expression

- Choose the Expression (113) command in a step in the User Program.
- Double-click the Expression parameter box. The Expression Builder will open.
- Use the Expression Editor to enter an expression in the Expression command. An expression consists of tags, operators and functions. You can type directly into the expression and also use the Expression Editor tabs to find the tags, operators and functions you need. To insert an item from the Expression Builder, choose the item and click **Insert**, or just double-click the item.
- If the expression consists of multiple lines, then each assignment statement must end with a semicolon, as shown in the examples above.

  For a full description of building expressions, see the Expressions Overview topic.

Data Types

Expressions follow a strict data type convention. If you mix data types in an expression, you must use the type conversion functions. Note that "3" can be either a DINT or REAL type, but "3.0" is only REAL type.

Order of Execution

Within a Step

Items in a step are always executed in order starting from the top. Therefore, the Expression command is executed at the point it appears in a step in a user program. If the Expression

command assigns a value to a register, that register will take on that value immediately. If that register is used in any following commands in the step, the register will use the value that it was assigned.

Within the Expression Command

Statements in an Expression are executed in order starting from the top.

Logging Results

The RMC logs the results of assignments and copies in the Expression (113) command to the Event Log for ease of troubleshooting. Logging does require significant processing time. If your user programs exceed the allotted execution time, one method of reducing the execution time is to disable expression logging. This will increase the amount of logic that fits in each user program step.

To disable expression logging, in the Programming Properties dialog, in the **Verify** tab, uncheck **Log Expression Statements**. With expression logging disabled, you can still use the LOG_EVENT function to log specific items.

For more details on reducing user program execution time, see Program Capacity and Time Usage.

See Also

Expressions Overview | List of Commands | Commands Overview

9. Register Reference

9.1. Registers

A *register* is a place in the RMC memory that stores data. The registers in the RMC are 32 bits and are any of the following data types:

- REAL: 32-bit floating point
- DINT: 32-bit integer
- DWORD: 32-bit string of bits

Register Addresses

Each register in the RMC can be addressed with several address types, such as Allen-Bradley DF1, Modbus, Omron, and IEC. Some register addresses are displayed in RMCTools, such as in the **Reg #** column in Axis Tools, Indirect Data Map, and the Variable Table. To view those addresses in various formats, in the **Reg#** column, right-click the address and choose Address Formats.

When referencing registers from within the RMC, such as in user programs, you do not need to use an address. You can use tag names instead. Tag names are given in the topic for each register.

Finding Register Addresses

When setting up communications with the RMC from a host controller such as a PLC, you will need to find the addresses of registers in the RMC. To do so,use the Address Maps in RMCTools, or the Register Map help topics.

Address Maps

1. In the Project tree, double-click **Address Maps**.
2. Choose the desired addressing protocol.
3. Browse the registers list for the register you need.

Register Maps

The Register Maps help topics list the addresses of all the registers in the RMC75, RMC150, and RMC200, and include links to detailed descriptions of each register.

- RMC75 Register Map
- RMC150 Register Map
- RMC200 Register Map

Address Formats

For each register, the register map provides addresses in the formats listed below. The address type you use will depend on the communication method you use. For more details on the addressing formats, see the respective topics.

- IEC-61131 Addressing
- Allen-Bradley DF1 Addressing
- Modbus Addressing
- FINS Addressing

See Also

RMC75 Register Map | RMC150 Register Map | RMC200 Register Map | Address Maps

9.2. Register Descriptions

9.2.1. Axis Definitions

9.2.1.1. Axis Definition Registers

The Axis Definition registers contain axis assignment information. These registers are not intended to be accessed directly by the user. In general, do not read or write to these registers. To define the axes on the RMC, use RMCTools. For details, see the Defining Axes topic.

In rare, advanced cases, the user may wish to change the axis definitions as part of the machine operation. This topic describes these registers and how to write to these registers from a host controller to change the axis definitions. Notice that changing axis definitions requires restarting the RMC, and therefore cannot de done while motion is in progress.

The Registers

The axis definition registers contain two areas: the Current Axis Definitions and the Requested Axis Definitions.

RMC75 Address	RMC150 Address	RMC200 Address	Register Name	Data Type
%MD19.0-15	%MD43.0-63	%MD24.0-511	Current Axis Definitions (Read-Only)	DWORD
%MD19.16-31	%MD43.64-127	%MD25.0-511	Requested Axis Definitions	DWORD

The Current Axis Definitions and the Requested Axis Definitions will generally be the same except in two cases:

(1) The user has written to the requested block and intends to do a warm restart or burn to flash and do a cold restart, or

(2) The requested axis definitions found on startup are invalid for the current hardware configuration; in this case the Current Axis Definitions will be the default for the current hardware configuration.

Changing the Axis Definitions from a Host Controller

Delta recommends changing axis definitions only in RMCTools. However, if your application absolutely requires it, you can change the axis definitions from your host controller (such as a PLC) by writing to the axis definition registers. This section describes how to do so.

The axis definition registers are fairly complex. Therefore, Delta recommends that you do not try to figure them out and write to the ones you need. Instead, Delta recommends that you use RMCTools to define the axes as you desire, then read all of the Current Axis Definition registers and save the data. Then, to apply that axis definition later, write that data to the Requested Axis Definition registers, update Flash, and restart the controller. The procedure is given below:

1. **Obtain the Axis Definitions**
   For each set of axis definitions that you will need, do the following:
   a. In RMCTools, set up the axis definitions as you desire, and apply them to the RMC.
   b. With your host controller, read the Current Axis Definitions registers, and store that data for later.
2. **Apply the Axis Definitions**
   Each time you wish to change the axis definitions, write your stored Axis Definitions to the Requested Axis Definitions registers.
3. **Restart the RMC**
   The RMC75, RMC150, and RMC200 can be restarted by writing to the Loader Command register. The following methods are available. See the Loader Command topic for details.
   a. **Cold Restart with Flash Update**
      This method first updates Flash, then does a cold restart of the RMC, which is the same as cycling power. Use this method if you want to automatically save the Axis Definitions to Flash and then restart.
   b. **Cold Restart without Flash Update**
      Does a cold restart of the RMC, which is the same as cycling power. You must manually save anything to Flash before using this method.
   c. **Warm Restart**
      This method does a warm restart, which retains all the data in the RMC, but doesn't save to Flash. Notice that the restart will apply the new axis definitions and will set all variables to initial values.
4. **Restart Communications**
   If communicating over USB or serial, wait 4 seconds for the controller to be ready to communicate again. If communicating over Ethernet, wait 8 seconds. After waiting the specified time, re-open the connection to the controller and resume communication.
5. If you changed axis definitions, some parameters may have changed and you may need to update them.

Axis Definition Registers Details

In each axis definition area (current and requested), there are four 32-bit DWORD registers per axis. Here is how these registers are defined:

n = axis number

Register	Bits	Description	Values
Register (*n* x 4)+0: Axis and Input Types			
(*n* x 4)+0	0-7	Axis Type	0 - None 1 - Servo Position Control 3 - Servo Velocity Control 5 - Pressure Control 6 - Force (single-input) Control 7 - Force (dual-input, diff.) Control 8 - Position-Pressure Control 9 - Position-Force (single-input) Control 10 - Position-Force (dual-input, diff.) Control 11 - Velocity-Pressure Control 12 - Velocity-Force (single-input) Control 13 - Velocity-Force (dual-input, diff.) Control 14 - Position Reference

			15 - Velocity Reference 16 - Pressure Reference 17 - Force (single-input) Reference 18 - Force (dual-input, diff.) Reference 19 - Virtual Axis 20 - Analog Output Only 22 - Position-Accel (single-input) Control 23 - Position-Accel (dual-input, diff.) Control 24 - Velocity-Accel (single-input) Control 25 - Velocity-Accel (dual-input, diff.) Control 26 - Accel (single-input) Reference 27 - Accel (dual-input, diff.) Reference
	8-15	Reserved	
	16-23	Input 0 (Primary) Type	0 - None 1 - Position 2 - Velocity 3 - Pressure 4 - Force (single-input) 5 - Force (dual-input, diff.) 6 - Accel (single-input) 7 - Accel (dual-input, diff.)
	24-31	Input 1 (Secondary) Type	Same as Input 0 (Primary) Type
Register (*n* x 4)+1: Input 0 Feedback			
(*n* x 4)+1	0-3	Feedback 0.0 Type	0 - None 1 - SSI/MDT (RMC75 and RMC200 only) 2 - Quadrature 3 - Custom 6 - Analog 7 - MDT (RMC150 only) 8 - SSI (RMC150 only) 9 - Resolver (RMC150 only) 10 - Load Cell (RMC200 only)
	4-7	Feedback 0.0 Module	**RMC75 Module IDs:** 0 - Axis Module - Axis 0 Connector 1 - Axis Module - Axis 1 Connector 2 - Expansion Slot #1 3 - Expansion Slot #2 4 - Expansion Slot #3 5 - Expansion Slot #4 **RMC150 Module IDs:** 0 - Comm Slot (not used... yet)

			1 - CPU Slot (not used) 2 - Sensor Slot #1 3 - Sensor Slot #2 4 - Sensor Slot #3 5 - Sensor Slot #4 **RMC200 Module IDs:** 0 - Power Supply (invalid) 1 - CPU (invalid) 2 - Slot #2 ... 14 - Slot #14
	8-11	Feedback 0.0 Channel	0 - First Input on the module (notice that each connector on the 2-axis RMC75 Axis Modules starts over at 0) 1 - Second Input on the module etc.
	12-15	Reserved	
	16-19	Feedback 0.1 Type (only for differential input types)	see Feedback 0.0 Type
	20-23	Feedback 0.1 Module	see Feedback 0.0 Module
	24-27	Feedback 0.1 Channel	see Feedback 0.0 Type
	28-31	Reserved	
Register (*n* x 4)+2: Input 1 Feedback - see Input 0 Feedback			
(*n* x 4)+2	see Input 0 Feedback		
Register (*n* x 4)+3: Output Definition			
(*n* x 4)+3	0-3	Output Type	0 - None 1 - Analog Output 3 - Virtual (Outer Loop)
	4-7	Output Module	**RMC75 Output Module IDs:** 0 - Axis Module - Axis 0 Connector 1 - Axis Module - Axis 1 Connector **RMC150 Module IDs:** 2 - Sensor Slot #1 3 - Sensor Slot #2 4 - Sensor Slot #3 5 - Sensor Slot #4 **RMC200 Module IDs:** 0 - Power Supply (invalid) 1 - CPU (invalid) 2 - Slot #2 ... 14 - Slot #14

	8-11	Output Channel	0 - First output on the module (notice that each connector on the 2-axis RMC75 Axis Modules starts over at 0) 1 - Second output on the module etc.
	12-31	Reserved	

See Also

Loader Command

9.2.2. Axis Status Registers

9.2.2.1. Axis Status Registers Overview

The Status Registers provide information on the status of each axis. These registers are not editable; they are read-only. Each Register is a 32-bit word. The Status Registers are axis dependant. Each group listed below contains Status Registers for the specified axes. For details on the addressing format for the registers, see the Register Address Format topic.

For a list of the Status Registers, see the Register Maps.

Tag Names

Tag names for axis status and parameter registers use the format _Axis[*x*].*reg*, where *x* specifies the axis number and *reg* is the tag name for that register. For example, _Axis[2].ActPos is the Actual Position of Axis 2.

If there is no number in the brackets, such as _Axis[].ActPos, the axis is the current axis for the current task. See the Tasks topic for more details.

See Also

Register Maps | Parameter Registers

9.2.2.2. Common

9.2.2.2.1. Status Bits Register

Note To Help Editor:

To add a Status Bit:

- **Add the text**
- **Add a bookmark next to it. Use the same format as the other bookmarks.**
- **In the TRUE CODE, change the id of the bookmark to "id=xstatBit*n*", where *n* is the number of the status bit. Look at the other id's to make sure it is right**

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.0, where *n* = 8 + the axis number **RMC150:** %MD*n*.0, where *n* = 8 + the axis number

	RMC200: %MD*n*.0, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].StatusBits, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	DWORD

Description

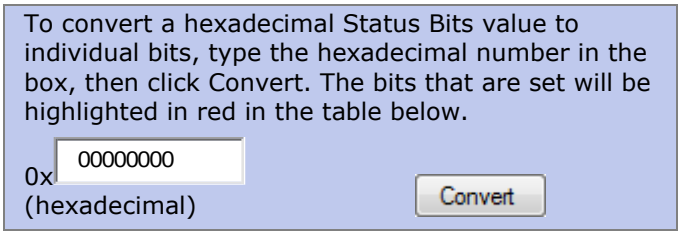
The Status Bits register is a collection of bits that provide a summary of the state of the axis. See the Registers Overview topic for details on how to address each specific bit in the various address formats.

Addressing Status Bits

In RMCTools, individual Status bits can be addressed by appending the bit tag name to "_Axis[*n*].StatusBits.", where *n* is the axis number. For example, "_Axis[1].Statusbits.InPos" is the Axis 1 In Position Status Bit.

Status Bits

This is a list of all the status bits. Not all axes will have all the status bits. The bit number of each bit is the same, whether or not all bits exist for a given axis.



#	Tag	Register Name
0	InPos	In Position
		This bit is set when a closed-loop position move is completed and the axis position is at the Command Position. This is defined as when the Target Position has reached the Command Position *and* the absolute difference between the Actual Position and the Command Position is less than the In Position Tolerance parameter. The In Position bit is not latched and will clear if the axis moves back outside the In Position window. This bit will not be set after stopping due to a Closed Loop Halt or a Stop (Closed Loop) (6) command. This bit is only used when controlling position in a mode that has a commanded position. Therefore, it will be clear in Open Loop, Velocity Control, or Position Control in modes such as gearing that have no final requested position.
1	AtVel	At Velocity
		This bit is set when a velocity move is completed and the axis velocity is at the Command Velocity. This is defined as when the Target Velocity has reached the Command Velocity and the absolute difference between the Actual Velocity and the Command Velocity is less than the At Velocity Tolerance parameter. The At Velocity bit is not latched and will clear if the axis speed moves back outside the At Velocity window. This bit is only used when controlling velocity in a mode that has a requested velocity. Therefore, it will be clear in Open Loop, Position Control, or Velocity Control in modes such as gearing that have no final requested velocity. For these cases, to determine whether an

axis has reached the constant velocity portion of the move, use the Target Generator State A and State B status bits.

2 OpenLoop — Open Loop

This bit is set when the axis is in Open Loop control or in the Direct Output state.
This bit will be off when Pressure/Force Limit is enabled, even if the axis was in open loop previously.

3 FaultIn — Fault Input

This bit is set when the Fault Input on the axis is active. This bit does not latch like the Fault Input error bit.

4 PosLimitIn — Positive Limit Input

This bit is set when the Positive Limit Input is active.

5 NegLimitIn — Negative Limit Input

This bit is set when the Negative Limit Input is active.

6 Stopped — Stopped

This bit is set when the axis is stopped.

For control axes, this is defined as when the Target Velocity is zero and the Actual Velocity is less than the Stop Threshold parameter. The axis must be in closed-loop control.

For reference axes, this is defined as when the Actual Velocity is less than the Stop Threshold parameter.

7 InputEst — Input Estimated

The Input Estimated bit indicates that the position or velocity input is currently being estimated due to any of these errors:

- No Transducer Error
- Transducer Overflow Error
- Noise Error

Not all the error bits listed above apply to all input types. See the individual error bits for details.

"Estimated" means that the feedback value is not accurate. It does not necessarily mean that it is being estimated to a high degree of certainty.

Position Inputs

In general (see below for actual details), when one of the errors listed above occurs, the Input Estimated bit will turn on to indicate an error condition exists and the feedback value is being estimated. If the error condition occurs for successive 3 loop times, the associated error bit will be set. Once the underlying error condition has gone away, the Input Estimated bit will remain on for two loop times before clearing.

The Input Estimated bit is set in the following cases:

- If a No Transducer or Transducer Overflow condition exists, the Input Estimated bit is always set, even after the No Transducer or Transducer Overflow error bit is set following the first 3 loop times. The RMC estimates the position by extrapolating the last two positions.
- For the RMC150 MDT module, if a No Transducer condition exists, the Input Estimated bit will be on for up to 6 loop times before the No Transducer Error bit will be set.

- For non-Quadrature feedback, the Input Estimated bit is set if a Noise Error condition exists for less than 3 loop times. After the 3rd loop time, the Input Estimated bit will turn off and the Noise Error bit will be set.
- For Quadrature, if a Noise Error condition occurs due to an illegal Quadrature transition (e.g. overspeed), then the Input Estimated bit is always set, even after the Noise Error bit is set following the first 3 loop times.

Velocity, Pressure and Force Inputs (Analog and mV/V):

The Input Estimated bit is not set if a Noise Error condition exists. A Noise Error condition will immediately set the Noise Error bit.

If a No Transducer or Transducer Overflow condition exists, the Input Estimated bit is always set, even after the error bit is set following the first 3 loop times.

8 EnableOut Enable Output

This bit is set when the Enable Output is on.

9 TGDone Primary Target Generator Done

This bit is set when the Primary Target Generator has completed its course, that is, when a motion command has been completed. If the motion move is interrupted, e.g. due to a halt, the done bit will not be set because the last commanded motion was not completed.

Specific details on the Target Generator status bit are included in the help topics for each motion command.

Example 1:
After a Move Absolute command has been issued, this bit will turn on when the Target Position reached the requested position.

Example 2:
After a Move Velocity command has been issued, this bit will turn on when the Target Velocity has reached the requested speed.

1
0 TGStateA Primary Target Generator State A

The Target Generator State A and State B bits indicate the current state of the Primary Target Generator. For each motion type, the following table describes in general the state indicated by the combinations of A and B. These bits are only valid for motion commands.

Specific details on the Target Generator status bit are included in the help topics for each motion command.

B	A	Open Loop Commands	Point-to-Point Commands	Velocity Moves	Quick Moves	Halts and Stops
0	0	Constant Control Output at 0 V	Done	Stopped	Done	Done
0	1	Ramping Control Output	Accelerating	Accelerating (away from 0 velocity)	Ramping Control Output in Open Loop	Reserved

		away from 0 V				
1	0	Constant Control Output at non-zero V	Constant Velocity	Constant Velocity	Constant Control Output at Requested Output	Reserved
1	1	Ramping Control Output toward 0 V	Deceleratin g	Deceleratin g (toward 0 velocity)	Deceleratin g in Closed Loop	Deceleratin g or Ramping Down the Control Output

11 TGStateB Primary Target Generator State B

See the Target Generator State A bit for a description of how these bits work. Specific details on the Target Generator status bit are also included in the help topics for each motion command.

12 DirectOut Direct Output

When this bit is set, the axis in a Direct Output state. The axis is in open loop control, ignores all error bits, and the Auto Stops have no effect.

The Direct Output (9) command, and the Direct Output Halt will turn on this bit. If the axis is disabled with the Enable/Disable Axis (97) command, this bit will turn on. If the axis is enabled, any motion command (except Direct Output (9)) will turn off this bit.

Certain parameters that affect motion require the Direct Output Status bit to be on when changing their values.

13 Enabled Enabled

RMC75/150:

Motion commands other than Direct Output (9) are not allowed on the axis if this bit is not set. This bit is set in the following cases:

- When the Enable Controller (7) or Enable/Disable Axis (97) command is issued to the axis
- When the axis enters RUN Mode.

The Enabled bit will be cleared when the axis is disabled via the Enable/Disable Axis (97) command.

RMC200:

No motion commands are allowed on the axis if this bit is not set. This bit is set in the following cases:

- When the Enable Controller (7) command is sent to the controller, if the Auto-Enable Axes setting in the Programming Properties dialog is checked.
- When the Enable/Disable Axis (97) command is sent to the axis to enable it.
- When the axis enters Program mode or Run Mode, if the Auto-Enable Axes setting in the Programming Properties dialog is checked.

The Enabled bit will be cleared when the axis is disabled via the Enable/Disable Axis (97) command, a Disable Axis AutoStop, or when the controller enters Disabled Mode.

14	ExtHalt	External Halt This bit is set when External Halt has occurred. An External Halt can only be triggered by an Auto Stop.
15	Halted	Halted This bit is set when one of the following halts occurs: Open Loop Halt, Open Loop Halt with Disable Drive, or Closed Loop Halt. These halts can then be triggered via Auto Stops or via explicit commands - see each halt topic for details.
16	PFControl	Pressure/Force Control This bit is set when the axis is in closed-loop pressure or force control. This bit will not be set for pressure/force limit; see the Pressure/Force Limit Enabled and Pressure/Force Limited bits below.
17	PFLimitEnabled	Pressure/Force Limit Enabled This bit is set when Pressure/Force Limit mode is enabled on the axis. When this bit is set, pressure or force will be limited to the Target Pressure/Force. Pressure/Force Limit mode can only be enabled or disabled with the Set Pressure/Force Limit Mode (40) command. This bit does not indicate whether pressure/force is being limited. To determine whether pressure or force is being limited, see the Pressure/Force Limited bit.
18	PFLimited	Pressure/Force Limited This bit is set when the pressure/force is currently being limited. That is, the position or velocity control is actually being affected in order to limit the pressure/force. This bit can only be on if the Pressure/Force Limit Enabled bit is on. See the Pressure Limit topic for details on pressure control. This bit will not be set in pressure/force control; see the Pressure/Force Control bit above.
19	AtPF	At Pressure/Force This bit indicates that the Actual Pressure/Force has reached the Command Pressure/Force. This bit is set when the Target Pressure/Force and the Actual Pressure/Force are at the Command Pressure/Force. This is defined as when the Pressure/Force Error is less than the At Pressure/Force Tolerance. The axis must be in closed-loop pressure/force control or pressure/force limit and the Pressure/Force Target Generator Done bit must be set.
20	PFInputEst	Pressure/Force Input Estimated If a Pressure/Force No Transducer or Pressure/Force Transducer Overflow condition exists, the Pressure/Force Input Estimated bit will turn on to indicate an error condition exists and the feedback value is being estimated. If the error condition occurs for successive 3 loop times, the associated Pressure/Force No Transducer or Pressure/Force Transducer Overflow error bit will be set. The Pressure/Force Input Estimated bit is set as long as the error condition exists, even after the error bit is set following the first 3 loop times.

"Estimated" means that the feedback value is not accurate. It does not necessarily mean that it is being estimated to a high degree of certainty.

The Pressure/Force Input Estimated bit is not set if a Pressure/Force Noise Error condition exists. A Pressure/Force Noise Error condition will immediately set the Pressure/Force Noise Error bit.

21 PFTGDone — Pressure/Force Target Generator Done

This bit is set when the Pressure/Force Target Generator has completed its course, that is, when a pressure or force ramp has been completed. If the ramp is interrupted, e.g. due to a halt, the done bit will not be set because the ramp was not completed.

22 PFTGStateA — Pressure/Force Target Generator State A

This bit is set to indicate the current state of the Pressure/Force Target Generator.

The Pressure/Force Target Generator State A and State B bits indicate the current state of the Pressure/Force Target Generator. For each motion type, the following table describes in general the state indicated by the combinations of A and B. These bits are only valid for motion commands.

Specific details on the Target Generator status bit are included in the help topics for each motion command.

B	A	Ramp: Linear, Time, or Auto	Ramp Rate	Stop
0	0	Pressure/Force is stopped (done)	Pressure/Force is stopped (done)	Pressure/Force is stopped (done)
0	1	Pressure/Force is increasing	Pressure/Force is accelerating	Reserved
1	0	Reserved	Pressure/Force is changing at a constant rate	Reserved
1	1	Pressure/Force is decreasing	Pressure/Force is decelerating	Decelerating toward zero rate.

23 PFTGStateB — Pressure/Force Target Generator State B

See the Pressure/Force Target Generator State A bit for a description of how these bits work. This bit is set to indicate the current state of the Pressure/Force Target Generator.

24 TGSIBusy — Primary Target Generator Superimposed Busy

This bit is set to indicate that a superimposed move is in progress. This can be caused by a Transition.

25 PFTGSIBusy — Pressure/Force Target Generator Superimposed Busy

This bit is set to indicate that a superimposed move is in progress. This can be caused by a pressure/force transition.

26 FeedbackOK — Primary Axis Feedback OK

This bit indicates that the feedback transducer on the primary input has no errors. This bit provides the instantaneous general status of

		the transducer, and is especially useful when implementing Custom Feedback. This bit is available in firmware 3.54.0 and newer. The Transducer OK bit will be off if the transducer is experiencing any of the following: • No Transducer • Transducer Overflow • Transducer Noise (only for position axes) Notice that the Transducer OK bit is instantaneous. The No Transducer, Transducer Overflow, and Transducer Noise axis error bits do not necessarily provide the instantaneous state of the feedback as the Transducer OK status bit does. These three error bits will turn on only after the error has existed for three loop times, during which time the Input Estimated bit is on. In addition, they will latch on and remain on even if the underlying error has gone away.
27	SecFeedbackOK	Secondary Axis Feedback OK This bit indicates that the feedback transducer on the secondary input has no errors. This bit provides the instantaneous general status of the transducer, and is especially useful when implementing Custom Feedback. This bit is available in firmware 3.54.0 and newer. The Secondary Transducer OK bit will be off if the transducer is experiencing any of the following: • Pressure/Force No Transducer • Pressure/Force Transducer Overflow Notice that the Secondary Transducer OK bit is instantaneous. The P/F No Transducer and P/F Transducer Overflow axis error bits do not necessarily provide the instantaneous state of the feedback as the Transducer OK status bit does. These three error bits will turn on only after the error has existed for three loop times, during which time the Input Estimated bit is on. In addition, they will latch on and remain on even if the underlying error has gone away.
31	-	Command Acknowledge Bit The RMC sets the Command Acknowledge bit to match the Command Request bit of the last command. This bit is intended to be used for synchronizing a PLC program with the RMC. See the Command Acknowledge Bit topic for details.

See Also

Status Registers | Error Bits

9.2.2.2.2. Error Bits Register

Note to Help Editor:

To add an Error Bit:

- **Add the text**
- **Add a bookmark next to it. Use the same format as the other bookmarks.**

- **In the TRUE CODE, change the id of the bookmark to "id=errBit0N", where N is the number of the error bit. Look at the other id's to make sure it is right.**

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.1, where *n* = 8 + the axis number **RMC150:** %MD*n*.1, where *n* = 8 + the axis number **RMC200:** %MD*n*.1, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ErrorBits, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	DWORD

Description

The Error Bits register is an Axis Status Register. It is a collection of bits that provide a summary of the errors on the axis. An error bit is set when certain conditions occur. Unless otherwise noted, these bits do not clear unless a Clear Faults command or any other motion command is issued *and* the underlying error condition has gone away.

When a bit is set, a halt will occur if the corresponding Auto Stop is configured to do so and the axis is not in the Direct Output state. Notice that the No Transducer and Transducer Overflow Error bits can not be set to Status Only.

When an error bit is set, it will be reported in the Event Log Monitor.

Addressing Error Bits

In RMCTools, individual Error bits can be addressed by appending the bit tag name to "_Axis[*n*].ErrorBits.", where *n* is the axis number. For example, "_Axis[1].ErrorBits.FollowErr" is the Axis 1 Following Error Bit.

Clearing Error Bits

When an error bit is set, it remains set although the error condition may have disappeared. For example, if the Control Output saturates, the Output Saturated error bit will be set and remain set even though the Control Output is no longer saturated.

To clear errors, issue the Clear Faults (4) command. It will clear all the error bits whose underlying error condition has gone away.

Issuing a motion command will also clear any error bits whose underlying error condition has gone away.

Error Bits

This is a list of all the status bits. Not all axes will have all the status bits. The bit number of each bit is the same, whether or not all bits exist for a given axis.

To convert a hexadecimal Error Bits value to individual bits, type the hexadecimal number in the box, then click Convert. The bits that are set will be highlighted in red in the table below.

0x 00000000 (hexadecimal) Convert

Bit	Tag	Register Name
0	FollowErr	Following Error If the axis is in closed loop *position* control, this bit is set when the Position Error exceeds the Position Error Tolerance parameter.

If the axis is in closed loop *velocity* control, this bit is set when the Velocity Error exceeds the Velocity Error Tolerance parameter.

This error is not used in I-PD control.

1 - Reserved

2 OutSat Output Saturated

If the Control Output exceeds the Output Limit parameter (positive or negative), then the Control Output is saturated and this bit is set. This bit is not set when the Control Output is limited at the DAC output stage. It is just silently truncated. Truncation at this stage results from applying the Output Bias and/or Output Polarity.

3 FaultIn Fault Input

This bit is set when the Fault Input for the axis goes active. This bit does latch unlike the Fault Input State status bit.

Some axes do not have a Fault Input.

4 PosLimitIn Positive Limit Input

This bit is set when the Positive Limit Input becomes active. Do not confuse this with the Positive Overtravel bit.

5 NegLimitIn Negative Limit Input

This bit is set when the Negative Limit Input becomes active. Do not confuse this with the Positive Overtravel bit.

6 NoTrans No Transducer

This bit is set when the RMC detects that no transducer is connected. The following list indicates when this occurs for each transducer type.

Analog Voltage

RMC75/150: This bit will not turn on for voltage feedback. The Transducer Overflow bit is used instead.

RMC200: A broken wire is detected, as indicated by:

- The voltage input is less than the minimum differential voltage (-10.5 V for the A8 and -10.2 V for the U14), *and*
- One of both of the individual input signals (In+ and In-) are out of range (see A8 or U14 specifications).

Analog Current

RMC75/150: The current is less than 3.6mA.

RMC200: A broken wire is detected, as indicated by the current being less than the Analog Underflow Limit (default is 3.6mA).

MDT

RMC75 or RMC200: The RMC does not detect a start pulse (for Start/Stop) or rising edge on the Return pulse (for PWM).

RMC150: The RMC has not detected a start pulse (for Start/Stop) or rising

edge on the Return pulse (for PWM) for the last six (6) control loops.

Load Cell	**LC8**: The RMC does not detect valid data. This may be due to an internal Out of Range Error, Data not Valid, or Analog to Digital Converter not Initialized error, which are all reported in the Transducer Status B register.
Quadrature	**RMC75:** A broken A or B wire is detected, as indicated by the A Wire Break and B Wire Break status bits. **RMC150:** This bit is not used by the RMC150. **RMC200:** A broken A or B wire is detected, as indicated by the A Wire Break and B Wire Break status bits.
Resolver	The RMC detects an over-speed, over-acceleration, or voltage error on the Resolver.
SSI	The RMC does not detect any valid data. The data line is the wrong polarity at either the beginning of the serial data stream or the end of the serial data stream, indicating the line is not being actively driven by the transducer.

In general (see the Input Estimated status bit for details and exceptions, including the RMC150 MDT module), the No Transducer error bit is not set until the error condition is present for three consecutive loop times. When a No Transducer error condition is first detected, the RMC estimates the position by extrapolating the last two positions. The Input Estimated Status bit be will be on while this occurs. After the condition is detected 3 times in a row, the No Transducer error bit is set. The Input Estimated bit is always set while the error condition is active, even after the No Transducer error bit is set following the first 3 loop times.

Once the No Transducer error bit has been set, it cannot be cleared until a transducer is detected three loop times in a row.

7 TransOverflow Transducer Overflow

This bit is set when the RMC detects an overflow on the input from the transducer. The following list indicates when this occurs for each transducer type.

Analog Voltage	**RMC75:** The voltage is greater than +10.1V or less than -10.1V. **RMC150 (±5V Range):** The voltage is greater than +5.04V or less than -5.04V.

	RMC150 (±10V Range: The voltage is greater than +10.08V or less than -10.08V. **RMC200:** An input signal is detected (that is, the No Transducer error is not set) but is not valid, as indicated by: • The voltage is above the Analog Overflow Limit (default is 10.1 V), *or* • The voltage is above the maximum differential voltage (10.5 V for the A8 and 10.2 V for the U14), *or* • The voltage is below the Analog Underflow Limit (default is -10.1 V), *or* • The voltage is below the minimum differential voltage (-10.5 V for the A8 and -10.2 V for the U14), *or* • One of both of the individual input signals (In+ and In-) are out of range (see A8 or U14 specifications).
Analog Current	**RMC75:** The current exceeds 21mA. **RMC150:** The current exceeds 20.1mA. **RMC200:** An input signal is detected (that is, the No Transducer error is not set) but is not valid, as indicated by: • The current exceeds the Analog Overflow Limit (default is 21.0 mA), *or* • One of both of the individual input signals (In+ and In-) are out of range (see A8 or U14 specifications).
MDT	**RMC75 or RMC200:** The RMC does not detect a stop pulse (for Start/Stop) or falling edge on the Return pulse (for PWM) by the end of the control loop. **RMC150:** The RMC has not detected a stop pulse (for Start/Stop) or falling edge on the Return pulse (for PWM) after three (3) control loops.
Load Cell	**LC8**: The Millivolts/Volt value is greater than the Load Cell Overflow Limit or less than the Load Cell Underflow Limit, or the Millivolt Input

	is greater than 34.5 mV or less than -34.5 mV.
Quadrature	Not used for Quadrature.
Resolver	Not used for resolvers.
SSI	As specified by the SSI Overflow Mode.

In general (see the Input Estimated status bit for details and exceptions), the Transducer Overflow error bit is not set until the error condition is present for three consecutive loop times. When a Transducer Overflow error condition is first detected, the RMC estimates the position by extrapolating the last two positions. The Input Estimated status bit be will be on while this occurs. After the condition is detected 3 times in a row, the Transducer Overflow error bit is set. The Input Estimated bit is always set while the error condition is active, even after the Transducer Overflow error bit is set following the first 3 loop times.

8 NoiseErr Noise Error

Note:
The Noise Error bit may turn on because the Noise Error Rate parameter is set too low. The Noise Error Rate should be set to a value much higher than the expected velocity of the axis. If your axis is experiencing Noise Errors, this is the first item you should check. Setting the Noise Error Auto Stop to Status Only to avoid halting due to an overly low Noise Error Rate parameter can cause significant control problems.

This bit is set when the RMC detects transducer noise on the feedback, before it is filtered. The following list indicates when this occurs for various transducer types.

Position:	
Analog, MDT, SSI, Resolver	If the input rate of change of the input exceeds the rate specified by the Noise Error Rate parameter for a period of 3 loop times, then this error bit will be set. During noise less than 3 loop times, the Actual Position is estimated to allow recovery from electrical noise, during which time the Input Estimated Status bit will be on. Make sure that the Noise Error Rate parameter is set much higher than the expected velocity of the system. If it is set too low, the Actual Position and Actual Velocity may be incorrectly calculated, adversely affecting control. A Noise Error can also be triggered on a Resolver axis if the speeds or accelerations exceed the maximums.

See the Resolver Module specifications for details.

Position: Quadrature	If the RMC detects an illegal transition (both A and B signals transition simultaneously), or an overspeed condition (pulse frequency exceeds maximum specifications), for a period of 3 loop times, then the Noise Error bit will be set. During noise less than 3 loop times, the Actual Position is estimated, during which time the Input Estimated Status bit will be on.
Velocity: Analog	If the input rate of change of the input exceeds the rate specified by the Noise Error Rate parameter, then this error bit will be set. There is no 3 loop delay.
Pressure, Force, Load Cell: Analog and mV/V	If the input rate of change of the input exceeds the rate specified by the Noise Error Rate parameter, then this error bit will be set. There is no 3 loop delay.

9 PosOvertravel Positive Overtravel

This bit is set when the Target Position travels beyond the Positive Travel Limit and is not in the Direct Output state. The Target Position can exceed the Positive Travel Limit by up to half a transducer count before this error is triggered.

Note:
This bit turns on only when exceeding the limit or attempting to go further past the limit. It will turn off if a command is issued that keeps the axis in the same place or moves it toward the limit.

10 NegOvertravel Negative Overtravel

This bit is set when the Target Position travels beyond the Negative Travel Limit and is not in the Direct Output state. The Target Position can exceed the Negative Travel Limit by up to half a transducer count before this error is triggered.

Note:
This bit turns on only when exceeding the limit or attempting to go further past the limit. It will turn off if a command is issued that keeps the axis in the same place or moves it toward the limit.

11 CmdErr Command Error

When an invalid command or command with invalid parameters is issued, this bit is set and an error number is stored in the Last Error Number status register. Depending on the Event Log filter settings, this error may also be logged to the Event Log. See the Error Codes topic for a list

		of errors that cause this bit to be set. This bit is cleared when any valid command is issued.
12	CmdMod	Command Modified This bit is set when a command has been modified. If a command contains an invalid Command Parameter, it may be modified. A command that has been modified will then be a valid command. For example, it the Requested Position is beyond the travel limits, the command will be automatically changed so that it will move to the limit. By default, an axis will halt when a command is modified. If the Command Modified Autostop is set to Status Only, then the axis will not halt and the modified command will be processed as usual. When this error occurs, an error number will be stored in the Last Error Number status register. Depending on the Event Log filter settings, this error may also be logged to the Event Log. See the Error Codes topic for a list of errors that cause this bit to be set. This bit is cleared when any valid command is issued.
13	CfgErr	Configuration Error When an attempt is made to write an invalid value to a parameter, this bit is set and an error number is stored in the Last Error Number status register. Depending on the Event Log filter settings, this error may also be logged to the Event Log. See the Error Codes topic for a list of errors that cause this bit to be set. This bit is cleared when any valid parameter write occurs.
14	RunErr	Runtime Error When an unexpected condition occurs that does not have its own error bit, this bit is set and an error number is stored in the Last Error Number status register. Depending on the Event Log filter settings, this error may also be logged to the Event Log. See the Error Codes topic for a list of errors that cause this bit to be set.
15	OutFault	Output Faulted Applies to axis with a Control Output assigned to the RMC200 CA4, CV8, or U14 modules. Indicates that physical fault occurred on the output such as an overcurrent or overvoltage. This may be caused by an open wire or shorted wire. Once the cause is remedied, the output will likely function properly.
16	-	Reserved
17	-	Reserved
18	PFNoTrans	Pressure/Force No Transducer This error bit is set in the same way that the No Transducer error bit is set, except this bit is set based on the secondary analog pressure/force input.
19	PFTransOverflow	Pressure/Force Transducer Overflow This error bit is set in the same way that the Transducer Overflow error bit is set, except this bit is set based on the secondary analog pressure/force input.
20	PFNoiseErr	Pressure/Force Noise Error

Note:
The Pressure/Force Noise Error bit may turn on because the Noise Error Rate parameter is set too low. The Pressure/Force Noise Error Rate should be set to a value much higher than the expected velocity of the axis. If your axis is experiencing Noise Errors, this is the first item you should check. Setting the Pressure/Force Noise Error Auto Stop to Status Only to avoid halting due to an overly low Noise Error Rate parameter can cause significant control problems.

This bit is set when the RMC detects transducer noise. If the input rate of change of the pressure or force input exceeds the rate specified by the Noise Error Rate parameter, then this error bit will be set.

This bit applies to the secondary analog pressure/force input. On pressure-only or force-only axes, the *primary* Noise Error bit #8 will apply to the pressure or force input.

21 PFFollowErr Pressure/Force Following Error

In pressure/force control, this error bit is set when the Pressure/Force Error exceeds the Pressure/Force Error Tolerance.

In pressure/force limit, this bit is set as follows:

- **Positive Mode:**
  When the Actual Pressure/Force exceeds the Target Pressure/Force by more than the Pressure/Force Error Tolerance
  (Actual > Target + Error Tolerance).
- **Negative Mode:**
  When the Actual Pressure/Force is less than the negated Target Pressure/Force by more than the Pressure/Force Error Tolerance
  (Actual < -Target - Error Tolerance).
- **Bidirectional Mode:**
  When either of the Positive or Negative cases are true.

See Also
Status Registers | Auto Stops | Status Bits

9.2.2.2.3. Last Error Number Register

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.2, where *n* = 8 + the axis number **RMC150:** %MD*n*.2, where *n* = 8 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].LastErrorNo, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Miscellaneous
Data Type:	DINT

Description

This status register stores the number of the last error that occurred. See the Error Codes topic for a description of the error numbers.

See Also

Status Registers

9.2.2.2.4. Read Response

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.4, where *n* = 8 + the axis number **RMC150:** %MD*n*.4, where *n* = 8 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].ReadResponse, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Miscellaneous
Data Type:	REAL

Description

This status register stores the value returned by the Read Register (111) command. It also stores the value returned by reading a single register via the Basic/Enhanced PROFIBUS Modes on the RMC75P.

See Also

Status Registers

9.2.2.3. Feedback

9.2.2.3.1. Actual Position

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.8, where *n* = 8 + the axis number **RMC150:** %MD*n*.8, where *n* = 8 + the axis number **RMC200:** %MD*n*.20, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ActPos, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	pu

Description

The Actual Position is the measured position of the axis at any moment, updated every control loop. This status register is valid only on position control axes. The Actual Position register may have filtering applied.

Filtering

The Actual Position can be filtered in several different ways. The default setting is that the Actual Position is not filtered, but filtering may be applied if the signal is excessively noisy. See the Filtering topic for details.

RMC75/150:

The Actual Position is the value after filtering has been applied. See the Filtering topic for details.

RMC200:

The Actual Position is the final filtered value after both the control filter and display filter have been applied. The unfiltered value is given by the Actual Position (Unfiltered) status register, and the first filtered value is given by the Actual Position (Control) value. The control algorithm uses the Actual Position (Control) value. If any further filtering is desired for display purposes, the display filter may be applied, and the final filtered value is given by the Actual Position register.

See the Filtering topic for details.

Calculation

The Actual Position is calculated from the transducer as follows:

Linear MDT, SSI or Resolver Input:

Actual Position = (Counts x Position Scale) + Position Offset

Rotary SSI or Resolver Input:

See the Rotary Scaling topic for details.

Quadrature Input:

Actual Position = (Counts x Position Scale)

Voltage Input:

Actual Position = (Voltage x Position Scale) + Position Offset

Current Input:

Actual Position = (Current x Position Scale) + Position Offset

To properly scale the counts to position-units, see the Position Scale topic.

If the Actual Position is noisy, it can be filtered. See the Actual Position Filter topic for details.

24-Bit Limit

The Actual Position is calculated from the Counts register. For SSI and quadrature inputs, this value may exceed 24 bits. See the **Exceeding 24 Bits** section of the Feedback Resolution topic for details on the 24-bit limitation of the Counts register.

See Also

Actual Position (Unfiltered) | Actual Position (Control)

9.2.2.3.2. Actual Velocity

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.9, where *n* = 8 + the axis number
	RMC150: %MD*n*.9, where *n* = 8 + the axis number
	RMC200: %MD*n*.21, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ActVel, where *n* is the axis number

How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	pu/sec

Description

This is the velocity of the axis. It is updated every control loop. The Actual Velocity register may have filtering applied.

Filtering

The Actual Velocity can be filtered in several different ways. By default, on position axes, the Actual Velocity is filtered to reduce noise caused by quantization errors. See the Filtering topic for details.

RMC75/150:

The Actual Velocity is the value after filtering has been applied.

RMC200:

The Actual Velocity is the final filtered value after both the control filter and display filter have been applied. The unfiltered value is given by the Actual Velocity (Unfiltered) status register, and the first filtered value is given by the Actual Velocity (Control) value. The control algorithm uses the Actual Velocity (Control) value. Further filtering is applied for display purposes, and the final filtered value is given by the Actual Velocity register.

Calculation

The Actual Velocity is obtained differently depending on the axis type:

- **Position Feedback Axes**
  The Actual Velocity is calculated from the change in the Counts, Voltage, or Current status register, depending on the feedback type. It is calculated as follows:

  Actual Velocity = (Counts(n) - Counts(n-1)) x Position Scale / Control Loop Time

  where n is the control loop.

  The velocity may be filtered as described in the **Filtering** section above.
- **Velocity Feedback Axes**
  The Actual Velocity is the velocity from the transducer. It is calculated as follows:

  Actual Velocity = Velocity Scale x [(Voltage or Current) + Velocity Offset ± Velocity Deadband)]

  The velocity may be filtered as described in the **Filtering** section above.

See Also

Actual Velocity (Control) | Actual Velocity (Unfiltered) | Filtering

9.2.2.3.3. Actual Acceleration

Type:	Axis Status Register
RMC75 Address:	**Primary Position/Velocity Input:** %MD*n*.10, where *n* = 8 + the axis number **Primary Acceleration Input:** %MD*n*.8, where *n* = 8 + the axis number **Secondary Input:** %MD*n*.23, where *n* = 8 + the axis number

RMC150 Address:	**Primary Position/Velocity Input:** %MD*n*.10, where *n* = 8 + the axis number **Primary Acceleration Input:** %MD*n*.8, where *n* = 8 + the axis number **Secondary Input:** %MD*n*.23, where *n* = 8 + the axis number
RMC200 Address:	**Primary Position/Velocity Input:** %MD*n*.22, where *n* = 256 + the axis number **Primary Acceleration Input:** %MD*n*.20, where *n* = 256 + the axis number **Secondary Input:** %MD*n*.80, where *n* = 256 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].ActAcc **Secondary Input:** _Axis[*n*].SecActAcc where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback
Data Type:	REAL
Units:	pu/sec^2

Description

This is the actual acceleration of the axis. It is updated every control loop. The Actual Acceleration may have filtering applied.

Filtering

The Actual Acceleration can be filtered in several different ways. By default, on position axes, the Actual Acceleration is filtered to reduce noise caused by quantization errors. See the Filtering topic for details.

RMC75/150:

The Actual Acceleration is the value after filtering has been applied.

RMC200:

The Actual Acceleration is the final filtered value after both the control filter and display filter have been applied. The unfiltered value is given by the Actual Acceleration (Unfiltered) status register, and the first filtered value is given by the Actual Acceleration (Control) value. The control algorithm uses the Actual Acceleration (Control) value. Further filtering is applied for display purposes, and the final filtered value is given by the Actual Acceleration register.

Calculation

The Actual Acceleration is obtained differently depending on the feedback type:

- **Position Feedback**
  The Actual Acceleration is the second derivative of the Actual Position. The acceleration may be filtered as described above.
- **Velocity Feedback**
  The Actual Acceleration is the change in the Actual Velocity. The acceleration may be filtered as described above.
- **Acceleration Feedback**
  The Actual Acceleration is calculated from the input (Voltage or Current). It is calculated as follows:

  Actual Acceleration = ((Voltage or Current) + Acceleration Offset) x Acceleration Scale

Algorithm Usage

The Actual Acceleration is used in both Active Damping and Acceleration Control. The following gains apply to the Actual Acceleration:

- Double Differential Gain
- Active Damping Proportional Gain

See Also
Actual Acceleration (Unfiltered) | Actual Acceleration (Control) | Filtering

9.2.2.3.4. Actual Jerk

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.9, where *n* = 8 + the axis number Secondary Input: %MD*n*.24, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.9, where *n* = 8 + the axis number Secondary Input: %MD*n*.24, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.21, where *n* = 256 + the axis number Secondary Input: %MD*n*.81, where *n* = 256 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].ActJerk, where *n* is the axis number **Secondary Input:** _Axis[*n*].SecActJerk, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback
Data Type:	REAL
Units:	pu/sec^2

Description

This is the actual jerk (rate of change of acceleration) of the axis. It is updated every control loop. This value is valid only on acceleration feedback axes.

The Actual Jerk is calculated as the first derivative of the Actual Acceleration.

Filtering the Jerk

The Actual Jerk is filtered by default. See the Actual Jerk Filter topic for details on filtering the Actual Jerk. Filtering makes the Actual Jerk less noisy.

Algorithm Usage

The Actual Jerk is used in both Active Damping and Acceleration Control. The following gains apply to the Actual Acceleration:

- Triple Differential Gain
- Active Damping Differential Gain

See Also
Status Registers

9.2.2.3.5. Actual Pressure/Force

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.8, where *n* = 8 + the axis number Secondary Input: %MD*n*.23, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.8, where *n* = 8 + the axis number Secondary Input: %MD*n*.23, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.20, where *n* = 256 + the axis number Secondary Input: %MD*n*.80, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].ActPrs, where *n* is the axis number **Force Input:** _Axis[*n*].ActFrc, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	Pr or Fr

Description

The Actual Pressure or Actual Force is the measured pressure or force of the axis at any moment. This value is updated every control loop. The Actual Pressure or Actual Force may have filtering applied.

Filtering

The Actual Pressure or Force can be filtered in several different ways. The default setting is that the Actual Pressure or Force is not filtered, but filtering may be applied if the signal is excessively noisy. See the Filtering topic for details.

RMC75/150:

The Actual Pressure or Actual Force is the value after filtering has been applied.

RMC200:

The Actual Pressure or Actual Force is the final filtered value after both the control filter and display filter have been applied. The unfiltered value is given by the Actual Pressure/Force (Unfiltered) status register, and the first filtered value is given by the Actual Pressure/Force (Control) value. The control algorithm uses the Actual Pressure/Force (Control) value. Further filtering may be applied for display purposes, and the final filtered value is given by the Actual Pressure/Force register.

Calculation

The pressure or force is calculated as follows:

Pressure Input:

Voltage Input:

Actual Pressure = Voltage * Pressure Scale + Pressure Offset

Current Input:

Actual Pressure = Current * Pressure Scale + Pressure Offset

Single-Input Force:

RMC75/150:

Voltage Input:

Actual Force = Voltage * Force Scale + Force Offset

Current Input:

Actual Force = Current * Force Scale + Force Offset

RMC200:
The RMC200 includes the Voltage or Current Offset and Negative Correction Factor parameters, which are intended to be used with load cells. The Actual Force calculation is as follows:

Voltage Input (±10V, 0-10V, ±5V):

If ((Voltage + Voltage Offset) < 0) Then
Actual Force = (Voltage + Voltage Offset) x Negative Correction Factor x Force Scale + Force Offset
Else
Actual Force = (Voltage + Voltage Offset) x Force Scale + Force Offset
EndIf

Current Input (4-20 mA, ±20 mA):

If ((Current + Current Offset) < 0) Then
Actual Force = (Current + Current Offset) x Negative Correction Factor x Force Scale + Force Offset
Else
Actual Force = (Current + Current Offset) x Force Scale + Force Offset
EndIf

Load Cell Input (mV/V):

If ((Millivolts/Volts + Millivolts/Volt Offset) < 0) Then
Actual Force = (Millivolts/Volts + Millivolts/Volt Offset) x Negative Correction Factor x Force Scale + Force Offset
Else
Actual Force = (Millivolts/Volts + Millivolts/Volt Offset) x Force Scale + Force Offset
EndIf

Dual-Input (Differential) Force Input:

RMC75/150:
Actual Force = Channel A Force - Channel B Force

RMC200:
Actual Force = Channel A Force - Channel B Force + Dual Channel Force Offset

See Also
Status Registers

9.2.2.3.6. Actual Pressure/Force Rate

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.9, where *n* = 8 + the axis number Secondary Input: %MD*n*.24, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.9, where *n* = 8 + the axis number Secondary Input: %MD*n*.24, where *n* = 8 + the axis number **RMC200:**

	Primary Input: %MD*n*.21, where *n* = 256 + the axis number Secondary Input: %MD*n*.81, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].ActPrsRate, where *n* is the axis number **Force Input:** _Axis[*n*].ActFrcRate, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	Pr/s or Fr/s

Description

The Actual Pressure Rate or Actual Force Rate register is the calculated rate of change in the Actual Pressure or Actual Force of the axis at any moment. This value is updated every control loop. The Actual Pressure Rate or Force Rate may have filtering applied.

Filtering

The Actual Pressure or Force can be filtered. The default setting is that the Actual Pressure Rate or Force Rate is filtered. See the Filtering topic for details.

RMC75/150:

The Actual Pressure Rate or Actual Force Rate is the value after filtering has been applied.

RMC200:

The Actual Pressure Rate or Actual Force Rate is the final filtered value after both the control filter and display filter have been applied. The unfiltered value is given by the Actual Pressure/Force Rate (Unfiltered) status register, and the first filtered value is given by the Actual Pressure/Force Rate (Control) value. The control algorithm uses the Actual Pressure/Force Rate (Control) value. Further filtering may be applied for display purposes, and the final filtered value is given by the Actual Pressure/Force Rate register.

See Also

Actual Pressure/Force Rate (Unfiltered) | Actual Pressure/Force Rate (Control) | Filtering

9.2.2.3.7. Counts

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.11, where *n* = 8 + the axis number Secondary Input: %MD*n*.26, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.11, where *n* = 8 + the axis number Secondary Input: %MD*n*.26, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.26, where *n* = 256 + the axis number Secondary Input: %MD*n*.86, where *n* = 256 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].Counts **Secondary Input:** _Axis[*n*].SecCounts where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback
Data Type:	REAL

Units:	counts

Description

Counts are the feedback from the transducer. The Counts register is used to calculate the Actual Position for axes with MDT, SSI, or Quadrature transducers. Axes with voltage or current input do not have a Counts register. They have a Voltage or Current register. All feedback types also have a Raw Counts register that is related to the Counts, Voltage or Current.

MDT Inputs:

The Counts are the Raw Counts.

SSI Inputs:

The Counts are the Raw Counts with the Count Offset applied. Also, for absolute rotary axes, the Counts are in the opposite direction if the Position Unwind is negative.

The Counts formula for positive Position Unwind is:

Counts = (RawCounts + CountOffset) MOD MaxCounts

For negative Position Unwind, the Counts formula is

Counts = ([MaxCounts-1] - RawCounts + CountOffset) MOD MaxCounts

Quadrature Inputs:

The Counts register accumulates encoder counts. The direction of the accumulation depends on the sign of the Position Scale or Position Unwind parameter. This value is adjusted when the axis is homed or when an Offset Position (47), Set Target Position (48), or Set Actual Position (49) command is issued. This value will wrap between -2,147,483,648 and +2,147,483,647. However, Delta recommends that the this value never exceed 24 bits (16,777,216). See the **Exceeding 24 Bits** section of the Feedback Resolution topic for details.

Resolver Inputs:

The Counts are the Raw Counts with the Count Offset applied. Also, for absolute rotary axes, the Counts are in the opposite direction if the Position Unwind is negative.

The Counts formula for positive Position Unwind is:

Counts = (RawCounts + CountOffset) MOD MaxCounts

For negative Position Unwind, the Counts formula is

Counts = ([MaxCounts-1] - RawCounts + CountOffset) MOD MaxCounts

Analog Inputs:

Axes with voltage or current input do not have a Counts register. They have a Voltage or Current register that derives from the Raw Counts.

Custom Inputs:

The Counts register takes on the value that is written to the Custom Counts register.

See the **Exceeding 24 Bits** section of the Feedback Resolution topic for details on the 24-bit limitation of the Counts register.

See Also

Raw Counts | Count Offset

9.2.2.3.8. Voltage

Type:	Axis Status Register

Address:	**RMC75:** Primary Input: %MD*n*.11, where *n* = 8 + the axis number Secondary Input: %MD*n*.26, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.11, where *n* = 8 + the axis number Secondary Input: %MD*n*.26, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.26, where *n* = 256 + the axis number Secondary Input: %MD*n*.86, where *n* = 256 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].Voltage **Secondary Input:** _Axis[*n*].SecVoltage where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback
Data Type:	REAL
Units:	Volts

Description

The Volts register is the voltage feedback from an analog transducer. It is derived from the raw counts register, which for the RMC75/150 is the value from the analog-to-digital converter, and for the RMC200 is the filtered value (see Analog Input Filter) from the analog-to-digital converter. The Voltage register is primarily for the user to set up and troubleshoot scaling.

The Voltage register is for axes with voltage feedback. The Current register is for axes with current feedback. Axes with other feedback types do not have a Current or Voltage register; they have a Counts register.

The RMC derives the Voltage from the Raw Counts, which is the value from the Analog-to-Digital Converter. The calculations are listed below. These are approximate, as the analog calibration values are also included in the calculation. The levels at which the value saturates and certain errors occur are also included.

Module	Range	Formula (approximate)	Saturates	No Transducer	Transducer Overflow
RMC75					
AA1, AA2, A2, AP2	±10 V	RawCounts x 10.125 V / 32,768 raw counts	at ±10.125 V	n/a	beyond ±10.1 V
RMC150					
A, G, H	±10 V	RawCounts x 10.0 V / 32,500 raw counts	at ±10.08 V	n/a	beyond ±10.08 V
A, H	±5 V	RawCounts x 5.0 V / 32,500 raw counts	at ±5.04 V	n/a	beyond ±5.04 V
UI/O	±10 V	RawCounts x 10.125 V / 32,768 counts	at ±10.125 V	n/a	beyond ±10.1 V
RMC200					
A8	±10 V	RawCounts x 10.5 V /	at ±10.5 V	Differential voltage < -10.5	Beyond Analog Underflow Limit or

		2,147,483,648 raw counts		V and either In+ or In- is out of range (see A8 specification)	Analog Overflow Limit, or saturated at maximum differential voltage, or either In+ or In- is out of range (see U14 or A8 specification)
U14	±10 V	RawCounts x 10.2 V / 2,147,483,648 raw counts	at ±10.2 V	Differential voltage < -10.2 V and either In+ or In- is out of range (see U14 specification)	

See Also

Current | Raw Counts | Analog Input Filter

9.2.2.3.9. Current

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.11, where *n* = 8 + the axis number Secondary Input: %MD*n*.26, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.11, where *n* = 8 + the axis number Secondary Input: %MD*n*.26, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.26, where *n* = 8 + the axis number Secondary Input: %MD*n*.86, where *n* = 8 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].Current **Secondary Input:** _Axis[*n*].SecCurrent where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback
Data Type:	REAL
Units:	mA

Description

The Current register is the current feedback from an analog transducer. It is derived from the raw counts register, which for the RMC75/150 is the value from the analog-to-digital converter, and for the RMC200 is the filtered value (see Analog Input Filter) from the analog-to-digital converter. The Current register is primarily for the user to set up and troubleshoot scaling.

The Current register is for axes with current feedback. The Voltage register is for axes with voltage feedback. Axes with other feedback types do not have a Current or Voltage register; they have a Counts register.

The RMC derives the Current from the Raw Counts, which is the value from the Analog-to-Digital Converter. The calculations are listed below. These are approximate, as the analog calibration values are also included in the calculation. The levels at which the value saturates and certain errors occur are also included.

Module	Range	Formula (approximate)	Saturates	No Transducer	Transducer Overflow
RMC75					
AA1, AA2, A2, AP2	4-20 mA	RawCounts x 20.0 mA / 16,384 counts	at ±21.0 mA	< 3.6 mA	> 21.0 mA
RMC150					
A, H	4-20 mA	RawCounts x 20.0 mA / 32,500 counts	at ±20.1 mA	< 3.6 mA	> 20.1 mA
UI/O	4-20 mA	RawCounts x 20.0 mA / 16,384 counts	at ±21.0 mA	< 3.6 mA	> 21.0 mA
RMC200					
A8	4-20 mA, ±20 mA	RawCounts x 42.0 mA / 2,147,483,648 counts	at ±42.0 mA*	Less than Analog Underflow Limit	Greater than Analog Overflow Limit or either In+ or In- is out of range (see A8 specification).
U14	4-20 mA, ±20 mA	RawCounts x 32.95 mA / 2,147,483,648 counts	at ±32.95 mA*	Less than Analog Underflow Limit	Greater than Analog Overflow Limit or either In+ or In- is out of range (see U14 specification).

*The current input is not intended for use beyond ±20 mA.

See Also

Voltage | Raw Counts | Analog Input Filter

9.2.2.3.10. Raw Counts

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.12, where *n* = 8 + the axis number Secondary Input: %MD*n*.27, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.12, where *n* = 8 + the axis number Secondary Input: %MD*n*.27, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.27, where *n* = 256 + the axis number

	Secondary Input: %MD*n*.87, where *n* = 256 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].RawCounts **Secondary Input:** _Axis[*n*].SecRawCounts where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback
Data Type:	DINT
Units:	raw counts

Description

The Raw Counts register is the value read directly from the transducer. They are used to derive the Counts, Volts, Current, or Millivolts, which in turn are used to calculate the Actual Position, Velocity, Acceleration, Pressure, or Force.

The Raw Counts are determined in the following manner for each transducer type:

- **MDT**
  The number of MDT clock counter cycles from the start pulse to the stop pulse (or from the beginning to the end of the PWM response). This is identical to the Counts.
- **SSI**
  The value clocked in from the transducer.
- **Analog Voltage or Current**
  The digital value read from the A-to-D converter. This value is then converted to Volts or Current as listed in the table below, using the factory-set calibration values of the module. The mapping is approximate, since the analog calibration values will adjust the Raw Counts slightly.
  Dual-input (differential) Force or Acceleration inputs have Channel A Raw Counts and Channel B Raw Counts.

Module	Mode	Raw Counts Range	Maps to approximately...	Formula (approximate)
RMC75				
AA1, AA2, A2, AP2	±10 V	-32,768 to +32,767	±10.13 V	RawCounts x 10.125 V / 32,768 counts
	4-20 mA		±40.00 mA*	RawCounts x 20.0 mA / 16,384 counts
RMC150				
A, G, H	±10 V	-32,768 to +32,767	±10.08 V	RawCounts x 10.0 V / 32,500 counts
A, H	±5 V		±5.04 V	RawCounts x 5.0 V / 32,500 counts
	4-20 mA		±20.17 mA	RawCounts x 20.0 mA / 32,500 counts
UI/O	±10 V		±10.13 V	RawCounts x 10.125 V / 32,768 counts
	4-20 mA		±40.00 mA*	RawCounts x 20.0 mA / 16,384 counts
RMC200				
A8	±10 V	-2,147,483,648 to +2,147,483,647	±10.55 V	RawCounts x 10.5 V / 2,147,483,648 counts
	4-20 mA		±42.18 mA*	RawCounts x 42.0 mA / 2,147,483,648 counts

U14	±10 V		±10.24 V	RawCounts x 10.2 V / 2,147,483,648 counts
	4-20 mA		±33.11 mA*	RawCounts x 32.95 mA / 2,147,483,648 counts

***The current input is not intended for use beyond ±20 mA.

- **Quadrature**
  Accumulated encoder counts since module power-up. Unlike the Counts status register, this counter is never reset, and is not affected by the sign Position Scale or Position Unwind parameter. The Raw Counts will wrap between -2,147,483,648 and +2,147,483,647.
- **Load Cell**
  The digital value read from the A-to-D converter. This value is then converted to Millivolt Input via the following formula, using the factory-set calibration values of the module. The calculation is approximate, since the analog calibration values will adjust the Raw Counts slightly.

  Millivolts = Raw Input * 34.25 mV (Nominal Range) / 8,388,608 (Half range counts)

  The Millivolts are then converted to Millivolts/Volt as described in the Millivolts/Volt topic.
- **Resolver**
  The Raw Counts are the absolute counts of the resolver. One revolution of the resolver will give 0 to 65535 raw counts, at which point it will wrap. This is regardless of the resolution. For 14-bit resolution, the counts will increment by 4. For 16-bit resolution, the counts will increment by 1.
- **Custom Inputs**
  The Raw Counts register is the Counts register rounded the nearest integer.

The Raw Counts register is a 32-bit unsigned register. However, when converted to Counts, Volts or Current, the value is converted to a floating-point value. See the **Exceeding 24 Bits** section of the Feedback Resolution topic for details on the 24-bit limitation of the Counts register.

See Also
Status Registers

9.2.2.3.11. Channel A, B Raw Counts

Type:	Axis Status Register
RMC75 Address:	**Primary Input:**
	Channel A: %MD*n*.12, where *n* = 8 + the axis number
	Channel B: %MD*n*.15, where *n* = 8 + the axis number
	Secondary Input:
	Channel A: %MD*n*.27, where *n* = 8 + the axis number
	Channel B: %MD*n*.30, where *n* = 8 + the axis number
RMC150 Address:	**Primary Input:**
	Channel A: %MD*n*.12, where *n* = 8 + the axis number
	Channel B: %MD*n*.15, where *n* = 8 + the axis number
	Secondary Input:
	Channel A: %MD*n*.27, where *n* = 8 + the axis number
	Channel B: %MD*n*.30, where *n* = 8 + the axis number

RMC200 Address:	**Primary Input:** Channel A: %MD*n*.27, where *n* = 256 + the axis number Channel B: %MD*n*.31, where *n* = 256 + the axis number **Secondary Input:** Channel A: %MD*n*.87, where *n* = 256 + the axis number Channel B: %MD*n*.91, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].CountsA, where *n* is the axis number _Axis[*n*].CountsB, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback
Data Type:	DINT
Units:	counts

Description

The Channel A Raw Counts is the value from the analog-to-digital converter on input 0 of a dual-input (differential) force or acceleration axis. This value is then multiplied by a factory-calibrated internal value to calculate the Channel A Voltage or Channel A Current.

The Channel B Raw Counts is the value from the analog-to-digital converter on input 1 of a dual-input (differential) force or acceleration axis. This value is then multiplied by a factory-calibrated internal value to calculate the Channel B Voltage or Channel B Current.

The conversion to volts or current is listed in the table below, using the factory-set calibration values of the module. The mapping is approximate, since the analog calibration values will adjust the Raw Counts slightly.

Module	Mode	Raw Counts Range	Maps to approximately...	Formula (approximate)
RMC75				
AA1, AA2, A2, AP2	±10 V	-32,768 to +32,767	±10.13 V	RawCounts x 10.125 V / 32,768 counts
	4-20 mA		±40.00 mA*	RawCounts x 20.0 mA / 16,384 counts
RMC150				
A, G, H	±10 V	-32,768 to +32,767	±10.08 V	RawCounts x 10.0 V / 32,500 counts
A, H	±5 V		±5.04 V	RawCounts x 5.0 V / 32,500 counts
	4-20 mA		±20.17 mA	RawCounts x 20.0 mA / 32,500 counts
UI/O	±10 V		±10.13 V	RawCounts x 10.125 V / 32,768 counts
	4-20 mA		±40.00 mA*	RawCounts x 20.0 mA / 16,384 counts
RMC200				
A8	±10 V	-2,147,483,648 to +2,147,483,647	±10.55 V	RawCounts x 10.5 V / 2,147,483,648 counts
	4-20 mA		±42.18 mA*	RawCounts x 42.0 mA / 2,147,483,648 counts
U14	±10 V		±10.24 V	RawCounts x 10.2 V / 2,147,483,648 counts

	4-20 mA		±33.11 mA*	RawCounts x 32.95 mA / 2,147,483,648 counts

*The current input is not intended for use beyond ±20 mA.

See Also
Status Registers

9.2.2.3.12. Channel A, B Current

Type:	Axis Status Register
RMC75 Address:	**Primary Input:** Channel A: %MD*n*.11, where *n* = 8 + the axis number Channel B: %MD*n*.14, where *n* = 8 + the axis number **Secondary Input:** Channel A: %MD*n*.26, where *n* = 8 + the axis number Channel B: %MD*n*.29, where *n* = 8 + the axis number
RMC150 Address:	**Primary Input:** Channel A: %MD*n*.11, where *n* = 8 + the axis number Channel B: %MD*n*.14, where *n* = 8 + the axis number **Secondary Input:** Channel A: %MD*n*.26, where *n* = 8 + the axis number Channel B: %MD*n*.29, where *n* = 8 + the axis number
RMC200 Address:	**Primary Input:** Channel A: %MD*n*.26, where *n* = 256 + the axis number Channel B: %MD*n*.30, where *n* = 256 + the axis number **Secondary Input:** Channel A: %MD*n*.86, where *n* = 256 + the axis number Channel B: %MD*n*.90, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].CurrentA, where *n* is the axis number _Axis[*n*].CurrentB, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback
Data Type:	REAL
Units:	counts

Description

These Current registers hold the current feedback from each channel's analog transducer on dual-input force or acceleration inputs. They are derived from the Channel A and B Raw Counts registers, which for the RMC75/150 are the values from the analog-to-digital converters, and for the RMC200 are the filtered values (see Analog Input Filter) from the analog-to-digital converters The Current registers are primarily provided for the user to set up and troubleshoot scaling.

The Channel A and B Current registers are used to calculate the Channel A and B Force or Acceleration.

The Current registers are for axes with current feedback. The Voltage registers are for axes with voltage feedback. Axes with other feedback types do not have a Current or Voltage register; they have a Counts register.

The RMC derives the Current from the Raw Counts, which is the value from the Analog-to-Digital Converter. The calculations are listed below. These are approximate, as the analog calibration values are also included in the calculation. The levels at which the value saturates and certain errors occur are also included.

Module	Range	Formula (approximate)	Saturates	No Transducer	Transducer Overflow
RMC75					
AA1, AA2, A2, AP2	4-20 mA	RawCounts x 20.0 mA / 16,384 counts	at ±21.0 mA	< 3.6 mA	> 21.0 mA
RMC150					
A, H	4-20 mA	RawCounts x 20.0 mA / 32,500 counts	at ±20.1 mA	< 3.6 mA	> 20.1 mA
UI/O	4-20 mA	RawCounts x 20.0 mA / 16,384 counts	at ±21.0 mA	< 3.6 mA	> 21.0 mA
RMC200					
A8	4-20 mA, ±20 mA	RawCounts x 42.0 mA / 2,147,483,648 counts	at ±42.0 mA*	Less than Analog Underflow Limit	Greater than Analog Overflow Limit or either In+ or In- is out of range (see A8 specification).
U14	4-20 mA, ±20 mA	RawCounts x 32.95 mA / 2,147,483,648 counts	at ±32.95 mA*	Less than Analog Underflow Limit	Greater than Analog Overflow Limit or either In+ or In- is out of range (see U14 specification).

***The current input is not intended for use beyond ±20 mA.

See Also

Channel A, B Voltage | Channel A, B Raw Counts | Analog Input Filter

9.2.2.3.13. Channel A, B Voltage

Type:	Axis Status Register
RMC75 Address:	**Primary Input:**
	Channel A: %MD*n*.11, where *n* = 8 + the axis number
	Channel B: %MD*n*.14, where *n* = 8 + the axis number
	Secondary Input:
	Channel A: %MD*n*.26, where *n* = 8 + the axis number

	Channel B: %MD*n*.29, where *n* = 8 + the axis number
RMC150 Address:	**Primary Input:**
	Channel A: %MD*n*.11, where *n* = 8 + the axis number
	Channel B: %MD*n*.14, where *n* = 8 + the axis number
	Secondary Input:
	Channel A: %MD*n*.26, where *n* = 8 + the axis number
	Channel B: %MD*n*.29, where *n* = 8 + the axis number
RMC200 Address:	**Primary Input:**
	Channel A: %MD*n*.26, where *n* = 256 + the axis number
	Channel B: %MD*n*.30, where *n* = 256 + the axis number
	Secondary Input:
	Channel A: %MD*n*.86, where *n* = 256 + the axis number
	Channel B: %MD*n*.90, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].VoltageA, where *n* is the axis number
	_Axis[*n*].VoltageB, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback
Data Type:	REAL
Units:	Volts

Description

These Voltage registers hold the voltage feedback from each channel's analog transducer on dual-input force or acceleration inputs. They are derived from the Channel A and B raw counts registers, which for the RMC75/150 are the values from the analog-to-digital converters, and for the RMC200 are the filtered values (see Analog Input Filter) from the analog-to-digital converters. The Voltage registers are primarily provided for the user to set up and troubleshoot scaling.

The Channel A and B Voltage registers are used to calculate the Channel A and B Force or Acceleration.

The Voltage registers are for axes with voltage feedback. The Current registers are for axes with current feedback. Axes with other feedback types do not have a Current or Voltage register; they have a Counts register.

The RMC derives the Voltage from the Raw Counts, which is the value from the Analog-to-Digital Converter. The calculations are listed below. These are approximate, as the analog calibration values are also included in the calculation. The levels at which the value saturates and certain errors occur are also included.

Module	Range	Formula (approximate)	Saturates	No Transducer	Transducer Overflow
RMC75					
AA1, AA2, A2, AP2	±10 V	RawCounts x 10.125 V / 32,768 raw counts	at ±10.125 V	n/a	beyond ±10.1 V
RMC150					
A, G, H	±10 V	RawCounts x 10.0 V / 32,500 raw counts	at ±10.08 V	n/a	beyond ±10.08 V
A, H	±5 V	RawCounts x 5.0 V / 32,500 raw counts	at ±5.04 V	n/a	beyond ±5.04 V

UI/O	±10 V	RawCounts x 10.125 V / 32,768 counts	at ±10.125 V	n/a	beyond ±10.1 V
RMC200					
A8	±10 V	RawCounts x 10.5 V / 2,147,483,648 raw counts	at ±10.5 V	Differential voltage < -10.5 V and either In+ or In- is out of range (see A8 specification)	Beyond Analog Underflow Limit or Analog Overflow Limit, or saturated at maximum differential voltage, or either In+ or In- is out of range (see U14 or A8 specification)
U14	±10 V	RawCounts x 10.2 V / 2,147,483,648 raw counts	at ±10.2 V	Differential voltage < -10.2 V and either In+ or In- is out of range (see U14 specification)	

See Also

Channel A, B Current | Channel A, B Raw Counts | Analog Input Filter

9.2.2.3.14. Channel A, B Force

Type:	Axis Status Register
RMC75 Address:	**Primary Input:** Channel A: %MD*n*.10, where *n* = 8 + the axis number Channel B: %MD*n*.13, where *n* = 8 + the axis number **Secondary Input:** Channel A: %MD*n*.25, where *n* = 8 + the axis number Channel B: %MD*n*.28, where *n* = 8 + the axis number
RMC150 Address:	**Primary Input:** Channel A: %MD*n*.10, where *n* = 8 + the axis number Channel B: %MD*n*.13, where *n* = 8 + the axis number **Secondary Input:** Channel A: %MD*n*.25, where *n* = 8 + the axis number Channel B: %MD*n*.28, where *n* = 8 + the axis number
RMC200 Address:	**Primary Input:** Channel A: %MD*n*.24, where *n* = 256 + the axis number Channel B: %MD*n*.28, where *n* = 256 + the axis number **Secondary Input:** Channel A: %MD*n*.84, where *n* = 256 + the axis number Channel B: %MD*n*.88, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ActFrcA, where *n* is the axis number _Axis[*n*].ActFrcB, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback

Data Type:	REAL
Units:	Frc

Description

The Channel A Force and Channel B Force are the calculated forces from input 0 and input 1, respectively, of a dual-input (differential) force input. These are then differenced to calculate the Actual Force. Channel A Force and Channel B Force are used in the Actual Force calculations as follows:

RMC75/150:

Channel A Force = Channel A Voltage (or Current) * Channel A Scale + Channel A Offset

Channel B Force = Channel B Voltage (or Current) * Channel B Scale + Channel B Offset

Actual Force = **Channel A Force** - **Channel B Force**

RMC200:

Channel A Pressure = Channel A Voltage (or Current) * Channel A Pressure Scale + Channel A Pressure Offset

Channel B Pressure = Channel B Voltage (or Current) * Channel B Pressure Scale + Channel B Pressure Offset

Channel A Force = Channel A Pressure * Channel A Force Scale

Channel B Force = Channel B Pressure * Channel B Force Scale

Actual Force = **Channel A Force** - **Channel B Force** + Dual Channel Force Offset

See Also

Status Registers | Input Type: Dual-Input Force

9.2.2.3.15. Channel A, B Pressure

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: Channel A: %MD*n*.25, where *n* = 256 + the axis number Channel B: %MD*n*.29, where *n* = 256 + the axis number Secondary Input: Channel A: %MD*n*.85, where *n* = 256 + the axis number Channel B: %MD*n*.89, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ActPrsA, where *n* is the axis number _Axis[*n*].ActPrsB, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback Axes Status Registers Pane, All tab: Presssure/Force/Accel Feedback
Data Type:	REAL
Units:	Frc

Description

The RMC200 Channel A Pressure and Channel B Pressure are the calculated pressures from input 0 and input 1, respectively, of a dual-input (differential) force input. These are then used to calculate the Channel A and B Actual Forces. Channel A Pressure and Channel B Pressure are used in the Actual Force calculations as shown below.

The RMC75/150 controllers use a different method of calculating the differential force, as described in Input Type: Dual-Input Force, and do not calculate the Channel A and B pressures.

RMC200 Differential Actual Force Calculation

Channel A Pressure = Channel A Voltage (or Current) * Channel A Pressure Scale + Channel A Pressure Offset

Channel B Pressure = Channel B Voltage (or Current) * Channel B Pressure Scale + Channel B Pressure Offset

Channel A Force = **Channel A Pressure** * Channel A Force Scale

Channel B Force = **Channel B Pressure** * Channel B Force Scale

Actual Force = Channel A Force - Channel B Force + Dual Channel Force Offset

See Also

Status Registers | Input Type: Dual-Input Force

9.2.2.3.16. Channel A, B Acceleration

Type:	Axis Status Register
RMC75 Address:	**Primary Input:**
	Channel A: %MD*n*.10, where *n* = 8 + the axis number
	Channel B: %MD*n*.13, where *n* = 8 + the axis number
	Secondary Input:
	Channel A: %MD*n*.25, where *n* = 8 + the axis number
	Channel B: %MD*n*.28, where *n* = 8 + the axis number
RMC150 Address:	**Primary Input:**
	Channel A: %MD*n*.10, where *n* = 8 + the axis number
	Channel B: %MD*n*.13, where *n* = 8 + the axis number
	Secondary Input:
	Channel A: %MD*n*.25, where *n* = 8 + the axis number
	Channel B: %MD*n*.28, where *n* = 8 + the axis number
RMC200 Address:	**Primary Input:**
	Channel A: %MD*n*.24, where *n* = 256 + the axis number
	Channel B: %MD*n*.28, where *n* = 156 + the axis number
	Secondary Input:
	Channel A: %MD*n*.84, where *n* = 256 + the axis number
	Channel B: %MD*n*.88, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ActAccA, where *n* is the axis number
	_Axis[*n*].ActAccB, where *n* is the axis number

How to Find:	Axes Status Registers Pane, All tab: Feedback
Data Type:	REAL
Units:	Fr

Description

The Channel A Acceleration is the calculated acceleration from input 0 of a dual-input (differential) Acceleration input.

The Channel B Acceleration is the calculated acceleration from input 1 of a dual-input (differential) Acceleration input.

Channel A Acceleration and Channel B Acceleration are calculated as follows:

Voltage Input:

Channel A Acceleration = (Channel A Voltage + Channel A Offset) * Channel A Scale

Channel B Acceleration = (Channel B Voltage + Channel B Offset) * Channel B Scale

Current Input:

Channel A Acceleration = (Channel A Current + Channel A Offset) * Channel A Scale

Channel B Acceleration = (Channel B Current + Channel B Offset) * Channel B Scale

The Channel A and Channel B acceleration are summed to produce the resultant Actual Acceleration.

See Also

Status Registers

9.2.2.3.17. Custom Counts

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.65, where *n* = 8+ the axis number Secondary Input: %MD*n*.66, where *n* = 8+ the axis number **RMC150:** Primary Input: %MD*n*.65, where *n* = 8+ the axis number Secondary Input: %MD*n*.66, where *n* = 8+ the axis number **RMC200:** Primary Input: %MD*n*.50, where *n* = 256+ the axis number Secondary Input: %MD*n*.110, where *n* = 256+ the axis number
System Tag:	**Primary Input:** _Axis[*n*].CustomCounts **Secondary Input:** _Axis[*n*].SecCustomCounts where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback Axes Status Registers Pane, All tab: P/F/A Feedback
Data Type:	REAL
Units:	counts

Description

The Primary and Secondary Custom Counts registers are for use with Custom Feedback. For axes with custom feedback, these registers are used for assigning the position, velocity, pressure, or force value to the axis. The user must make a user program that continuously assigns some value to the Custom Counts register.

The Counts value will be scaled by the Scale and Offset. However, in typical applications, the Scale and Offset will be set at the default values of 1 and 0 respectively, meaning that no scaling takes place, and the value written to the Counts will be applied directly to the Actual Position, Velocity, Pressure, or Force register.

For details, see the Custom Feedback topic.

See Also

Custom Feedback

9.2.2.3.18. Custom Error Bits

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.64, where *n* = 8 + the axis number **RMC150:** %MD*n*.64, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.51, where *n* = 256 + the axis number Secondary Input: %MD*n*.111, where *n* = 256 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].CustomErrorBits, where *n* is the axis number **RMC200:** Primary Input: _Axis[*n*].CustomStatus Secondary Input: _Axis[*n*].SecCustomStatus
How to Find:	Axes Status Registers Pane, All tab: Feedback, Custom - No Transducer Axes Status Registers Pane, All tab: P/F/A Feedback, Custom - P/F No Transducer
Data Type:	DWORD

Description

The Custom Error Bits register is a collection of bits that provide a summary of the state of the Custom Feedback.

In RMCTools, individual Status bits can be addressed by appending the bit tag name to the system tag name.
For example:

_Axis[1].CustomErrorBits.NoTrans is the RMC75/150 Axis 1 Custom Feedback No Transducer Error bit.

_Axis[3].CustomStatus.NoTrans is the RMC200 Axis 3 Custom Feedback No Transducer Error bit.

_Axis[4].SecCustomStatus.NoTrans is the RMC200 Axis 4 Secondary Input Custom Feedback No Transducer Error bit.

For more details, see Custom Feedback.

Custom Error Bits

RMC75/150

Bit #	Tag	Register Name

6	NoTrans	Custom Feedback No Transducer Error bit This bit is intended to be written to in order to specify whether the Custom Counts value is valid.
18	PFNoTrans	Custom Feedback Pressure/Force No Transducer Error bit Identical to the NoTrans bit, but applies to the secondary input.

RMC200

For both _Axis[*n*].CustomStatus and _Axis[*n*].SecCustomStatus:

Bit #	Tag	Register Name
0	NoTrans	Custom Feedback No Transducer Error bit This bit is intended to be written to in order to specify whether the Custom Counts value is valid.

See Also

Custom Feedback | Custom Counts

9.2.2.3.19. Transducer Status A and B

Type:	Axis Status Register
Transducer Status A Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Inputs: %MD*n*.41, where *n* = 256+ the axis number %MD*n*.43 (channel B, if applicable), where *n* = 256+ the axis number Secondary Inputs: %MD*n*.101, where *n* = 256+ the axis number %MD*n*.103 (channel B, if applicable), where *n* = 256+ the axis number
Transducer Status B Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Inputs: %MD*n*.42, where *n* = 256+ the axis number %MD*n*.44 (channel B, if applicable), where *n* = 256+ the axis number Secondary Inputs: %MD*n*.102, where *n* = 256+ the axis number %MD*n*.104 (channel B, if applicable), where *n* = 256+ the axis number
System Tags:	**Primary Inputs:** _Axis[*n*].Xdcr0StatusA _Axis[*n*].Xdcr1StatusA (channel B, if applicable)

	_Axis[*n*].Xdcr0StatusB _Axis[*n*].Xdcr1StatusB (channel B, if applicable) **Secondary Inputs:** _Axis[*n*].SecXdcr0StatusA _Axis[*n*].SecXdcr1StatusA (channel B, if applicable) _Axis[*n*].SecXdcr0StatusB _Axis[*n*].SecXdcr1StatusB (channel B, if applicable) where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback or Pressure/Force/Accel Feedback
Data Type:	DWORD

Description

The Transducer Status A and Transducer Status B status registers provide advanced diagnostic bits relating to the input. Here are the main uses for this information:

- Determine the precise cause for a No Transducer**,** Transducer Overflow, or Transducer Noise error.
- On SSI transducers, extract extra data from measurement data that includes multiple data items, such as position and temperature. See **SSI/MDT Input (S8 Module), Transducer Status B** below.
- On SSI transducers, capturing the data signals using plots to see the actual data on the wire. See **SSI/MDT Input (S8 Module), Capturing an SSI/MDT Scope Plot** below.

The contents of these registers is specific to each I/O module and feedback type and is described below:

- Analog Input (A8 Module)
- Analog Input (U14 Module)
- SSI/MDT Input (S8 Module)
- SSI/MDT Input (U14 Module)
- Load Cell Input (LC8 Module)
- Quadrature Input (Q4 Module)
- Quadrature Input (S8 Module)
- Quadrature Input (U14 Module)
- Quadrature Input (D24 Module)

Analog Input (A8 Module):

Transducer Status A:

Bits	Description
31:1	Reserved.
0	**Out of Range Error.** This bit will be set when either In+ or In- is outside the valid range so that the analog reading is unreliable.

Transducer Status B:

Bits	Description
31:0	Raw Filtered Analog Input. This 32-bit value represents the filtered analog input reading. This unsigned integer represents the following ranges approximately, since the analog calibration values will adjust the value slightly. Voltage: ±10.54 V

	Current: ±42.18 mA (with ±10 V range and 250 Ω resistor). Notice that the current input is not intended for use beyond ±20 mA.

Analog Input (U14 Module):

Transducer Status A:

Bits	Description
31:18	Reserved.
17:14	**Out of Range Error.** For analog inputs 3-0. These bits will be set when either In+ or In- for the corresponding input is outside the valid range so that the analog reading is unreliable.
13:12	**Output Fault.** For analog outputs 1-0. Indicates a over-current fault in voltage mode or an output saturation fault in current mode for the corresponding output.
11:8	**Discrete Input State.** For discrete inputs 3-0.
7:0	**Module Temperature.** Signed module temperature in degrees C.

Transducer Status B:

Bits	Description
31:0	**Raw Filtered Analog Input.** This 32-bit value represents the filtered analog input reading. This unsigned integer represents the following ranges approximately, since the analog calibration values will adjust the value slightly. Voltage: ±10.24 V Current: ±33.11 mA (with ±5 V range and 151 Ω resistor). Notice that the current input is not intended for use beyond ±20 mA.

SSI/MDT Input (S8 Module):

Transducer Status A:

Bits	Description
31:29	Reserved.
28:20	**Loop-Back Test Counter.** Number of 60 MHz clock cycles between rising edge of Interrogate signal and first rising edge of return signal. Used for Loopback test in production testing.
19:17	Reserved.
16	**Scope Trigger.** Used for capturing a scope plot of the SSI/MDT data. See **Capturing an SSI/MDT Scope Plot** below.
15	**Scope Clock/Interrogate Signal.** Used for capturing a scope plot of the SSI/MDT data. See **Capturing an SSI/MDT Scope Plot** below.
14	**Scope Data/Return Signal.** Used for capturing a scope plot of the SSI/MDT data. See **Capturing an SSI/MDT Scope Plot** below.
13	**Quadrature Error (channel 6 only).** When in Quad mode, quadrature encoder A and B signals transitioned at the same time.
12	Reserved.
11	**MDT No Stop Pulse.** Will be set in MDT mode if the Stop pulse or PWM falling edge was not seen from the MDT before the end of the loop time. This will trigger a Transducer Overflow error.
10	**MDT No Start Pulse.** Will be set in MDT mode if the Start pulse or PWM rising edge was not seen from the MDT before the end of the loop time. This will trigger a No Transducer error.

9	**SSI Data High at End of Cycle.** Will be set in SSI or SSI Monitor mode if the SSI Data line was not pulled low as expected after the last clock pulse, which is required by the SSI specification. This will trigger a No Transducer error unless Wire Break Detection is disabled.
8	**SSI Data Low at Beginning of Cycle.** Will be set in SSI or SSI Monitor mode if the SSI Data line was not high as expected before the first clock pulse, which is required by the SSI specification. This will trigger a No Transducer error unless Wire Break Detection is disabled.
7	**SSI Monitor Excessive Clock Pulses.** Will be set in SSI Monitor mode if more clock pulses were detected on the Clock input than are specified for this channel. Make sure that the number of data bits matches the system. This will trigger a No Transducer error.
6	**SSI Monitor Insufficient Clock Pulses.** Will be set in SSI Monitor mode if fewer clock pulses were detected on the Clock input than are specified for this channel. Make sure that the number of data bits matches the system. This will trigger a No Transducer error.
5	**S8 Channel Pair Error.** Will be set for SSI Monitor channels if there is an internal configuration error. Contact Delta if this occurs.
4:2	**Feedback Type.** Will be None (000), SSI (001), MDT (010), SSI Monitor (011), or Quad (100) (chan 6 only, channel 7 will be 0).
1	**Input State (channel 6 and 7 only)**. When in Quad mode, this bit indicates the state of the Data/Return line (captured at the loop tick) for reporting the state of input A or B.
0	**Data Valid.** Indicates that the input data is valid, as defined based on the mode as shown below. This bit being clear will trigger a No Transducer error. • **MDT mode:** Will be set if the measurement cycle completed without any MDT errors (bits 10 and 11 are zero). • **SSI Mode:** Will be set if the measurement cycle completed without any SSI errors (bits 8 and 9 are zero). • **SSI Monitor mode:** Will be set if the measure cycle completed within the last loop time, whether or not any of the SSI or SSI Monitor error bits above are set.

Transducer Status B:

Bits	Description
31:0	**Transducer Counts.** For MDT encoders, this will hold the number of 240 MHz clock cycles between the Start and Stop pulses or the high time of a PWM signal. For SSI encoders, the raw data value read from the transducer. If Gray Code is selected, this register will hold the value after being converted to binary. Unused bits will be zero; for example, bits 24-31 will be zero for a 24-bit encoder. Use this register to examine the raw transducer feedback in cases where the data includes multiple data items, such as position and temperature, or position and status bits. The SSI High Bits to Ignore and SSI Low Bits to Ignore parameters are used to obtain the position data for the axis, and the remaining data can be parsed from this register.

Capturing an SSI/MDI Scope Plot:

Bits 14-16 can be used to capture an oscilloscope-like trace of the SSI or MDT signal in the plots. This trace is an interleaved signal over time and therefore, if the SSI or MDT data is changing significantly during the capture, the data will not appear valid. If the SSI or MDT data is constant

during the capture time, the data on the plot will appear like an oscilloscope trace of the SSI or MDT input.

The **Scope Trigger** bit is set when a new trace is started. This will give the user a reference point for the image of the **Scope Clock/Interrogate Signal** and **Scope Data/Return Signal** bits seen in plots.

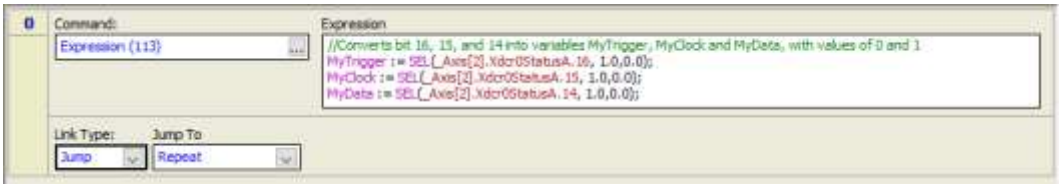
The **Scope Clock/Interrogate Signal** and **Scope Data/Return Signal** bits are the state of the SSI Clock output, MDT Interrogate output, SSI Data input, or MDT Return input. The inputs are captured in the I/O module at progressively larger delays (in increments of 200 ns) from the loop tick. In the plot, each sample will indicate a capture. (This signal shows what should be going on the physical inputs, but does not capture the signals on the wire itself. If the driver chip is damaged or the line is shorted this bit will not show that.)

The trace duration will be: loop time / 200 ns. When the trace completes, it will restart immediately.

Loop Time	2 ms	1 ms	500 µs	250 µs
Trace Duration	10 s	5 s	2.5 s	1.25 s

To capture an SSI/MDT oscilloscope plot:

1. Create and run a user program that converts the **Scope Trigger** (bit 16), **Scope Clock/Interrogate Signal** (bit 15) and **Scope Data/Return Signal** (bit 14) bits into three individual registers that can be plotted. Make sure the user program runs continuously. An example program:



2. Add the three individual registers to a plot template.
3. Start trending the plot. Alternately, you can use the **Scope Trigger** bit to trigger the start of a captured plot.
4. In the plot, the **Scope Trigger** variable will indicate the beginning of a scope trace. Internally, the I/O module will capture one sample of the SSI/MDT input every 200 ns. This is interleaved through the entire loop time. For SSI, you will see the Clock and Data bits. For MDT, you will see the Interrogate and Return bits.

SSI/MDT Input (U14 Module):

Transducer Status A:

Bits	Description
31:29	Reserved.
28:20	**Loop-Back Test Counter**. Number of 60 MHz clock cycles between rising edge of Interrogate signal and first rising edge of return signal. Used for Loopback test in production testing.
19:17	Reserved.
16	**Scope Trigger.** Used for capturing a scope plot of the SSI/MDT data. See **Capturing an SSI/MDT Scope Plot** above.
15	**Scope Clock/Interrogate Signal.** Used for capturing a scope plot of the SSI/MDT data. See **Capturing an SSI/MDT Scope Plot** above.
14	**Scope Data/Return Signal.** Used for capturing a scope plot of the SSI/MDT data. See **Capturing an SSI/MDT Scope Plot** above.
13:12	Reserved.

11	**MDT No Stop Pulse.** Will be set in MDT mode if the Stop pulse or PWM falling edge was not seen from the MDT before the end of the loop time. This will trigger a Transducer Overflow error.
10	**MDT No Start Pulse.** Will be set in MDT mode if the Start pulse or PWM rising edge was not seen from the MDT before the end of the loop time. This will trigger a No Transducer error.
9	**SSI Data High at End of Cycle.** Will be set in SSI or SSI Monitor mode if the SSI Data line was not pulled low as expected after the last clock pulse, which is required by the SSI specification. This will trigger a No Transducer error unless Wire Break Detection is disabled.
8	**SSI Data Low at Beginning of Cycle.** Will be set in SSI or SSI Monitor mode if the SSI Data line was not high as expected before the first clock pulse, which is required by the SSI specification. This will trigger a No Transducer error unless Wire Break Detection is disabled.
7	**SSI Monitor Excessive Clock Pulses.** Will be set in SSI Monitor mode if more clock pulses were detected on the Clock input than are specified for this channel. Make sure that the number of data bits matches the system. This will trigger a No Transducer error.
6	**SSI Monitor Insufficient Clock Pulses.** Will be set in SSI Monitor mode if fewer clock pulses were detected on the Clock input than are specified for this channel. Make sure that the number of data bits matches the system. This will trigger a No Transducer error.
5	Reserved.
4:2	**Feedback Type.** SSI (001), MDT (010), SSI Monitor (011), Quad (100), SSI Echo (101) (chan 1 only).
1	Reserved.
0	**Data Valid.** Indicates that the input data is valid, as defined based on the mode as shown below. This bit being clear will trigger a No Transducer error. • **MDT mode:** Will be set if the measurement cycle completed without any MDT errors (bits 10 and 11 are zero). • **SSI Mode:** Will be set if the measurement cycle completed without any SSI errors (bits 8 and 9 are zero). • **SSI Monitor mode:** Will be set if the measure cycle completed within the last loop time, whether or not any of the SSI or SSI Monitor error bits above are set. • **Quadrature mode:** Will always be set.

Transducer Status B:

Bits	Description
31:0	**Transducer Counts.** For MDT encoders, this will hold the number of 240 MHz clock cycles between the Start and Stop pulses or the high time of a PWM signal. For SSI encoders, the raw data value read from the transducer. If Gray Code is selected, this register will hold the value after being converted to binary. Unused bits will be zero; for example, bits 24-31 will be zero for a 24-bit encoder. Use this register to examine the raw transducer feedback in cases where the data includes multiple data items, such as position and temperature, or position and status bits. The SSI High Bits to Ignore and SSI Low Bits to Ignore parameters are used to obtain the position data for the axis, and the remaining data can be parsed from this register.

Load Cell Input (LC8 Module):

Transducer Status A, Input 0:

Bits	Description
31	**A2D Converter Initialized.** This module-wide bit is set when the Sample Rate Converter value (which controls the Output Data Rate (ODR) from the ADC) has been configured. Analog values on all channels are invalid if this bit is not set. When the ODR is changed, this bit will be off for 10 ms (20 ms for 4 ms loop times).
12:30	Reserved.
0:11	**Wire Sense.** This 12-bit unsigned value represents the voltage drop in the reference voltage wire to the load cell. This value is updated every 18 ms or every 9 loop times, whichever is greater. Scaling for this value is: WS(Volts) = (Counts – 2048) * 3.185 / (2048 x 30.12) Resolution is 51.63 µV per LSB.

Transducer Status A, Input 1:

Bits	Description
14:31	**Data Latency.** Module-wide 18-bit Data Latency register. The Data Latency measures the time, in increments of 33.333 ns, from the ADC data receipt to the next loop tick. This value should be about 50.
12:13	Reserved.
0:11	**Wire Sense.** See **Transducer A Status, Input 0** above.

Transducer Status A, Input 2:

Bits	Description
20:31	**Temperature.** 12-bit unsigned Temperature register. The ADC temperature in °C is calculated from this value: Temp = 25 + ((TEMP – 2048) * 3.185 / 2048 – 0.6) / 0.002 Resolution is 0.778 °C per LSB.
12:19	Reserved
0:11	**Wire Sense.** See **Transducer A Status, Input 0** above.

Transducer Status A, Input 3-7:

Bits	Description
12:31	Reserved.
0:11	**Wire Sense.** See **Transducer A Status, Input 0** above.

Transducer Status B:

Bits	Description
8:31	**Analog Input.** 24-bit twos complement value representing the input voltage to the channel. The range is nominally -34.8606 mV to +34.8606 mV.
7	**Data Valid.** Indicates that no ADC error bits are on (bits 5:1 below).
6	Reserved
5	**Channel ID Error.** Channel number read does not match expected channel number. This is an internal error in with the ADC interface.
4	**Reset Detected.** This bit indicates if the ADC was reset. This bit is not channel specific. Data is not valid until this bit is cleared.

3	**ADC Saturated.** This bit indicates that the ADC modulator saturated or the ADC filter saturated. Data is not valid if this bit is set.
2	**Reference Detect Error.** Reference voltage has dropped below 0.7V (Nominal is 2.5V).
1	**Overvoltage or Undervoltage Error.** This bit indicates that there is an overvoltage or undervoltage condition on the input.
0	**Input Signal Out of Range Error.** 0 = in range, 1 = out of range. This bit is set when either the + or – input signal is outside the valid range, so the analog reading is unreliable. This is detected at the module input rather than at the ADC input. The valid range is nominally 0.6 V to 6.15 V. This allows use of our 6.75 V exciter output or an externally supplied 10 V exciter.

Quadrature Input (Q4 Module):

Transducer Status A:

Bits	Description
31	**A State.** State of the A signal exactly at the loop tick.
30	**A Fault.** Single-ended fault on A input. Voltage is out of range.
29	**A Fault.** Single-ended fault on A input. Voltage is out of range.
28	**A Differential Fault.** Differential voltage on A is out of range.
27	**B State.** State of the B signal exactly at the loop tick.
26	**B Fault.** Single-ended fault on B input. Voltage is out of range.
25	**B Fault.** Single-ended fault on B input. Voltage is out of range.
24	**B Differential Fault.** Differential voltage on B is out of range.
23	**Quadrature Error.** Simultaneous transitions on A and B. This will trigger a Transducer Noise error.
22	**Timer Reset Event.** Timer was reset during last loop time.
21	**Timer Latch Event.** Timer value was latched during last loop time.
20	**Counter Direction.** Direction of last quadrature count.
19	**Counter Changed Direction.** Last count was in opposite direction from previous count.
18	**Timer Overflow.** Timer exceeded 18 bits since last reset.
17:0	**Quadrature Counter.** This holds a 18-bit accumulation of quadrature counts since power-up, starting at zero. It increases with each change in A or B where A leads B and decreases when A trails B. This value wraps when it overflows in the positive and negative direction. In pulse counter mode, this value will only count up.

Transducer Status B:

Bits	Description
31	**Z State.** State of the Z signal exactly at the loop tick.
30	**Z Fault.** Single-ended fault on Z input. Voltage is out of range.
29	**Z Fault.** Single-ended fault on Z input. Voltage is out of range.
28	**Z Differential Fault.** Differential voltage on Z is out of range.
27	**Home Input State.** State of the Home input exactly at the loop tick.
26	**Home Fault.** Fault on the Home input. Voltage is out of range.
25	**Registration Input State.** State of the Reg input exactly at the loop tick.

24	**Home Latch Valid.** Home event occurred this loop time. The value is latched in **Home Count.** This bit is active even if Home is not armed. There is a 2 loop time delay from when the CPU firmware arms the home to when it will acknowledge an active Home Latch Valid bit.
23	**Home Direction.** Direction of counter when Home latched.
22	**Registration_X Latch Valid.** Registration X event occurred this loop time.
21	**Registration_Y Latch Valid.** Registration Y event occurred this loop time.
20	**Z Pulse.** Edge of Z pulse was seen during last loop time.
19	**Z Location Latched.** This bit will be set after a Learn Z Alignment (54) command is requested to indicate that the state of **Z Location** is valid.
18	**Z Location.** This bit will be set or cleared when **Z Location Latched** is set indicating whether the Z Location is A Leading (0) or A Trailing (1).
17:0	**Home Count.** Value of the Quadrature Counter when the Home event occurred. Only valid when **Home Latch Valid** is set.

Quadrature Input (S8 Module):

Transducer Status A:

Bits	Description
31:0	**Quadrature Counter.** This holds a 32-bit accumulation of quadrature counts since power-up, starting at zero. It increases with each change in A or B where A leads B and decreases when A trails B. This value wraps when it overflows in the positive and negative direction.

Transducer Status B:

Bits	Description
31:14	Reserved.
13	**Quadrature Error.** Will be set when quadrature encoder A and B signals transitioned at the same time. This will trigger a Transducer Noise error.
12:6	Reserved.
5	**S8 Channel Pair Error.** Will be set if there is an internal configuration error. Contact Delta if this occurs.
4:2	**Feedback Type.** Will be 4 (binary 100) for quadrature channel.
1	**Input State.** Will hold the state of the quadrature A line. The state of A and B inputs are available in the Encoder Status register.
0	**Reserved.** Will always be 1.

Quadrature Input (U14 Module):

Transducer Status A:

Bits	Description
31	**Z State.** State of the Z signal exactly at the loop tick.
30	**Z Fault.** Single-ended fault on Z input. Voltage is out of range.
29	**Z Fault.** Single-ended fault on Z input. Voltage is out of range.
28	**Z Differential Fault.** Differential voltage on Z is out of range.
27:26	Reserved.
25	**Home Latch Valid.** Home event occurred this loop time. The value is latched in **Home Count.** This bit is active even if Home is not armed. There is a 2 loop

	time delay from when the CPU firmware arms the home to when it will acknowledge an active Home Latch Valid bit.
24	**Home Direction.** Direction of counter when Home latched.
23	**Registration_X Latch Valid.** Registration X event occurred this loop time.
22	**Registration_Y Latch Valid.** Registration Y event occurred this loop time.
21	**Z Pulse.** Edge of Z pulse was seen during last loop time.
20	**Z Location Latched.** This bit will be set after a Learn Z Alignment (54) command is requested to indicate that the state of **Z Location** is valid.
19	**Z Location.** This bit will be set or cleared when **Z Location Latched** is set indicating whether the Z Location is A Leading (0) or A Trailing (1).
18	Reserved.
17:0	**Quadrature Counter.** This holds an 18-bit accumulation of quadrature counts since power-up, starting at zero. It increases with each change in A or B where A leads B and decreases when A trails B. This value wraps when it overflows in the positive and negative direction. In pulse counter mode, this value will only count up.

Transducer Status B:

Bits	Description
31	**A State.** State of the A signal exactly at the loop tick.
30	**A Fault.** Single-ended fault on A input. Voltage is out of range.
29	**A Fault.** Single-ended fault on A input. Voltage is out of range.
28	**A Differential Fault.** Differential voltage on A is out of range.
27	**B State.** State of the B signal exactly at the loop tick.
26	**B Fault.** Single-ended fault on B input. Voltage is out of range.
25	**B Fault.** Single-ended fault on B input. Voltage is out of range.
24	**B Differential Fault.** Differential voltage on B is out of range.
23	**Quadrature Error.** Simultaneous transitions on A and B. This will trigger a Transducer Noise error.
22:21	Reserved.
20	**Counter Direction.** Direction of last quadrature count.
19	**Counter Changed Direction.** Last count was in opposite direction from previous count.
18	Reserved.
17:0	**Home Count.** Value of the Quadrature Counter when the Home event occurred. Only valid when Home Latch Valid is set.

Quadrature Input (D24 Module):

Transducer Status A:

Bits	Description
31	**Quadrature Error.** Simultaneous transitions on A and B. This will trigger a Transducer Noise error.
30	**A Wire Break.** Wire break detected on A.
29	**B Wire Break.** Wire break detected on B.
28	**Home Latch Valid.** Home event occurred this loop time. The value is latched in **Home Count**. This bit is active even if Home is not armed. There is a 2 loop

	time delay from when the CPU firmware arms the home to when it will acknowledge an active Home Latch Valid bit.
27	**Registration Latch Valid.** Registration event occurred this loop time.
26	**Z Location Latched.** This bit will be set after a Learn Z Alignment (54) command is requested to indicate that the state of **Z Location** is valid.
25	**Z Location.** This bit will be set or cleared when **Z Location Latched** is set indicating whether the Z Location is A Leading (0) or A Trailing (1).
24	**Home Direction.** Direction of counter when Home latched.
23	**Timer 0/2 Reset Event.** Timer was reset during last loop time.
22	**Timer 1/3 Reset Event.** Timer was reset during last loop time.
21	**Counter Direction.** Direction of last quadrature count.
20	**Counter Changed Direction.** Last count was in opposite direction from previous count.
19:16	Reserved.
15:0	**Quadrature Counter.** This holds a 16-bit accumulation of quadrature counts since power-up, starting at zero. It increases with each change in A or B where A leads B and decreases when A trails B. This value wraps when it overflows in the positive and negative direction. In pulse counter mode, this value will only count up.

Transducer Status B:

Bits	Description
31:16	Reserved.
15:0	**Home Count.** Value of the Quadrature Counter when the Home event occurred. Only valid when **Home Latch Valid** is set.

See Also

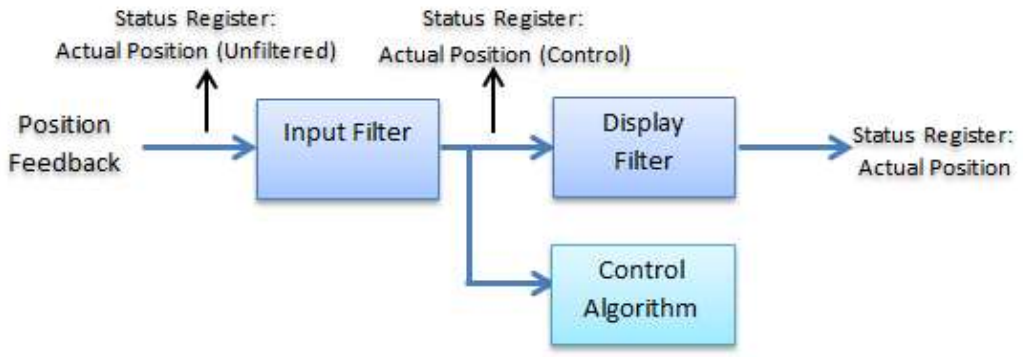
A8 Module (RMC200) | S8 Module (RMC200) | Q4 Module (RMC200) | D24 Module (RMC200) | LC8 Module (RMC200)

9.2.2.3.20. Actual Position (Unfiltered)

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.32, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ActPosUnfiltered, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	pu

Description

The Actual Position (Unfiltered) value is the unfiltered Actual Position. The control algorithms are not performed on the Actual Position (Unfiltered). Therefore, the Actual Position (Unfiltered) value is mostly for troubleshooting when verifying the filtering performance.



See the Actual Position and Filtering topics for more details.

See Also

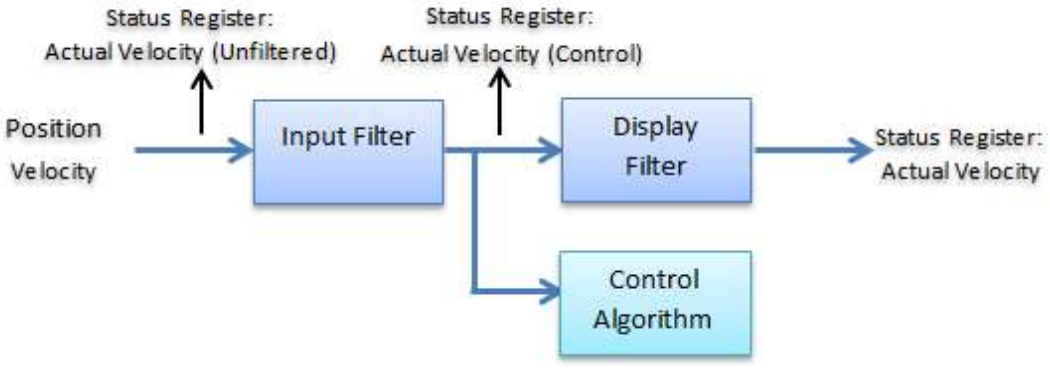
Actual Position | Actual Position (Control) | Filtering

9.2.2.3.21. Actual Velocity (Unfiltered)

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.33, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ActVelUnfiltered, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	pu/sec^2

Description

The Actual Velocity (Unfiltered) value is the unfiltered Actual Velocity. The control algorithms are not performed on the Actual Velocity (Unfiltered). Therefore, the Actual Velocity (Unfiltered) value is mostly for troubleshooting when verifying the filtering performance.



See the Actual Velocity and Filtering topics for more details.

See Also

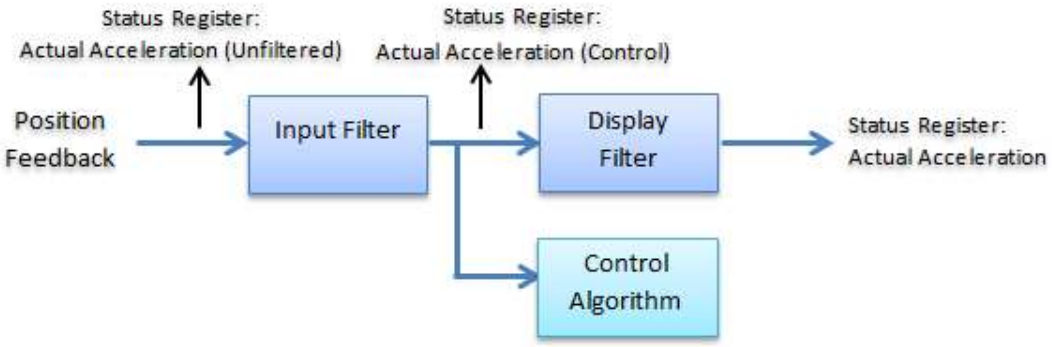
Help Overview

9.2.2.3.22. Actual Acceleration (Unfiltered)

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** **Position/Velocity Input:** %MD*n*.34, where *n* = 256 + the axis number **Primary Acceleration Input:** %MD*n*.32, where *n* = 256 + the axis number **Secondary Acceleration Input:** %MD*n*.92, where *n* = 256 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].ActAccUnfiltered **Secondary Input:** _Axis[*n*].SecActAccUnfiltered where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	pu/sec^2

Description

The Actual Acceleration (Unfiltered) value is the unfiltered Actual Acceleration. The control algorithms are not performed on the Actual Acceleration (Unfiltered). Therefore, the Actual Acceleration (Unfiltered) value is mostly for troubleshooting when verifying the filtering performance.



See the Actual Acceleration and Filtering topics for more details.

See Also

Actual Acceleration | Actual Acceleration (Control) | Filtering

9.2.2.3.23. Actual Jerk (Unfiltered)

Type:	Axis Status Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** **Primary Acceleration Input:** %MD*n*.33, where *n* = 256 + the axis number **Secondary Acceleration Input:** %MD*n*.93, where *n* = 256 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].ActJerkUnfiltered **Secondary Input:** _Axis[*n*].SecActJerkUnfiltered where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	pu/sec^3

Description

The Actual Jerk (Unfiltered) value is the unfiltered Actual Jerk. The control algorithms are not performed on the Actual Jerk (Unfiltered). Therefore, the Actual Jerk (Unfiltered) value is mostly for troubleshooting when verifying the filtering performance.

See the Actual Jerk and Filtering topics for more details.

See Also

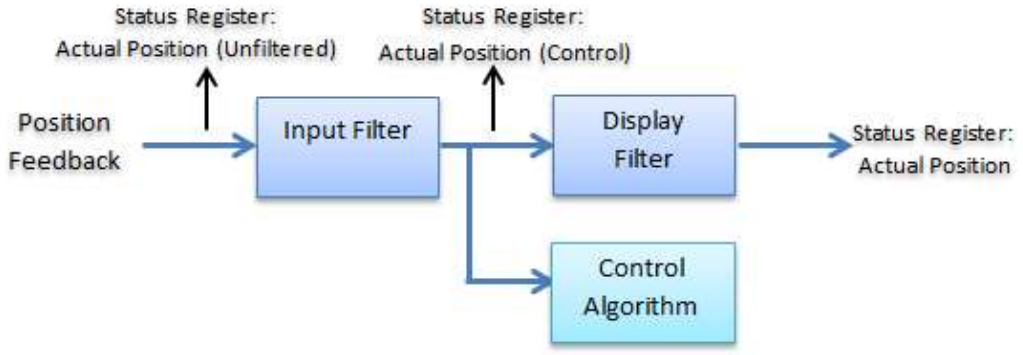
Actual Jerk | Filtering

9.2.2.3.24. Actual Position (Control)

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.36, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ActPosControl, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	pu

Description

The Actual Position (Control) value is the position feedback with the control filter applied. This value is used by the control algorithm. The value may be filtered again by the display filter, and the final filtered value is given by the Actual Position status register.



The Actual Position (Control) value is valuable for detailed troubleshooting because it shows the exact value that the control-algorithm is operating on, whereas the commonly used Actual Position may be filtered and therefore differ slightly from the Actual Velocity (Control) value.

See Actual Position and Filtering topics for more details.

See Also

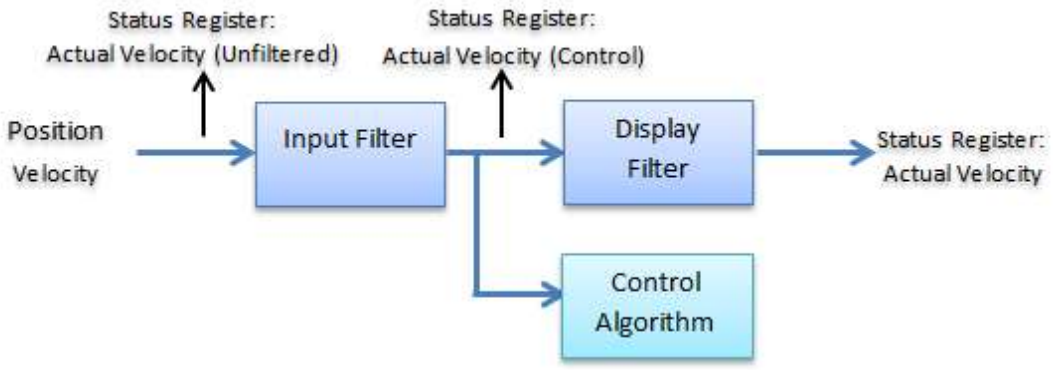
Actual Position | Actual Position (Unfiltered) | Filtering

9.2.2.3.25. Actual Velocity (Control)

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.37, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ActVelControl, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	pu/sec

Description

The Actual Velocity (Control) value is the velocity feedback with the control filter applied. This value is used by the control algorithm for the PID calculations. The value may be filtered again by the display filter, and the final filtered value is given by the Actual Velocity status register.



The Actual Velocity (Control) value is valuable for detailed troubleshooting because it shows the exact value that the control-algorithm is operating on, whereas the commonly used Actual Velocity is typically a filtered value that will differ slightly from the Actual Velocity (Control) value.

See Actual Velocity and Filtering topics for more details.

See Also

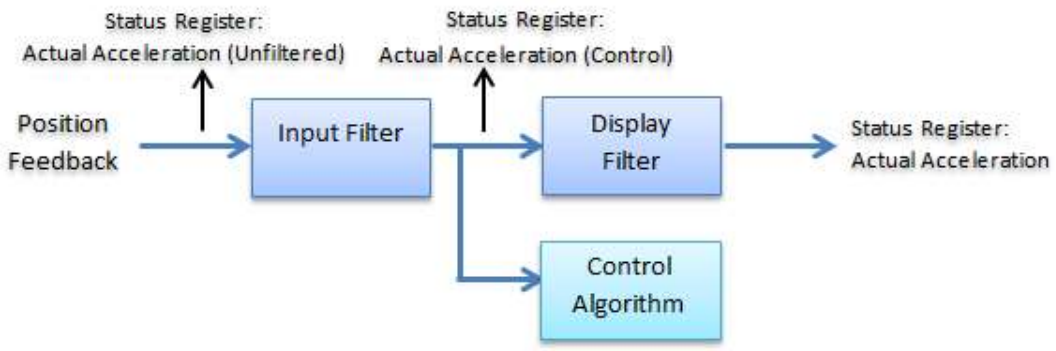
Actual Velocity | Actual Velocity (Unfiltered) | Filtering

9.2.2.3.26. Actual Acceleration (Control)

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** **Position/Velocity Input:** %MD*n*.38, where *n* = 256 + the axis number **Primary Acceleration Input:** %MD*n*.36, where *n* = 256 + the axis number **Secondary Acceleration Input:** %MD*n*.96, where *n* = 256 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].ActAccControl **Secondary Input:** _Axis[*n*].SecActAccControl where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	pu/sec^2

Description

The Actual Acceleration (Control) value is the acceleration feedback with the control filter applied. This value is used by the control algorithm for the PID calculations. The value may be filtered again by the display filter, and the final filtered value is given by the Actual Acceleration status register.



The Actual Acceleration (Control) value is valuable for detailed troubleshooting because it shows the exact value that the control-algorithm is operating on, whereas the commonly used Actual Acceleration is typically a filtered value that will differ slightly from the Actual Acceleration (Control) value.

See Actual Acceleration and Filtering topics for more details.

See Also

Actual Acceleration | Actual Acceleration (Unfiltered) | Filtering

9.2.2.3.27. Actual Jerk (Control)

Type:	Axis Status Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** **Primary Acceleration Input:** %MD*n*.37, where *n* = 256 + the axis number **Secondary Acceleration Input:** %MD*n*.97, where *n* = 256 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].ActJerkControl **Secondary Input:** _Axis[*n*].SecActJerkControl where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	pu/sec^3

Description

The Actual Jerk (Control) value is the Jerk feedback with the control filter applied. This value is used by the control algorithm for the PID calculations. The value may be filtered again by the display filter, and the final filtered value is given by the Actual Jerk status register.

The Actual Jerk (Control) value is valuable for detailed troubleshooting because it shows the exact value that the control-algorithm is operating on, whereas the commonly used Actual Jerk is typically a filtered value that will differ slightly from the Actual Jerk (Control) value.

See Actual Jerk and Filtering topics for more details.

See Also

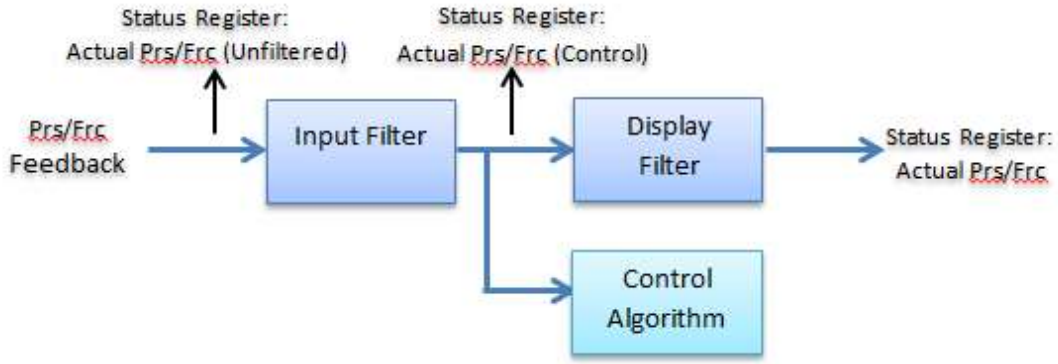
Actual Jerk | Filtering

9.2.2.3.28. Actual Pressure/Force (Unfiltered)

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.32, where *n* = 256 + the axis number Secondary Input: %MD*n*.92, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].ActPrsUnfiltered, where *n* is the axis number **Force Input:** _Axis[*n*].ActFrcUnfiltered, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	Fr or Pr

Description

The Actual Pressure (Unfiltered) or Actual Force (Unfiltered) value is the unfiltered Actual Pressure or Actual Force. The control algorithms are not performed on the Actual Pressure (Unfiltered) or Actual Force (Unfiltered). Therefore, the Actual Pressure (Unfiltered) and Actual Force (Unfiltered) values are mostly for troubleshooting when verifying the filtering performance.



See the Actual Pressure/Force and Filtering topics for more details.

See Also

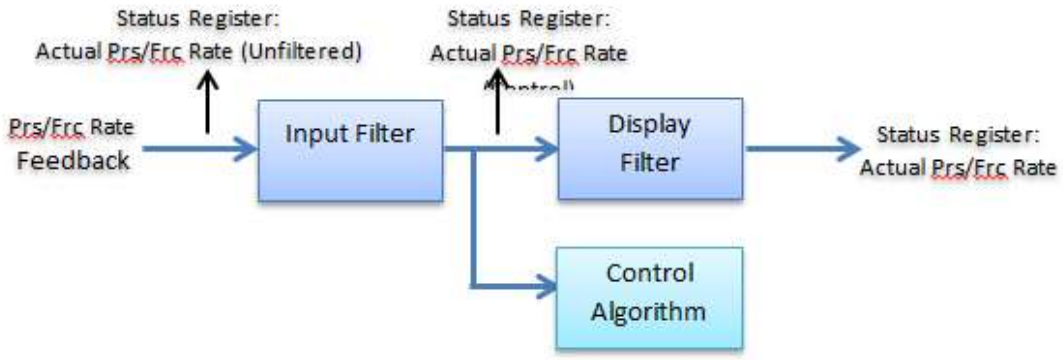
Actual Pressure/Force | Actual Pressure/Force (Control) | Filtering

9.2.2.3.29. Actual Pressure/Force Rate (Unfiltered)

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.33, where *n* = 256 + the axis number Secondary Input: %MD*n*.93, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].ActPrsRateUnfiltered, where *n* is the axis number **Force Input:** _Axis[*n*].ActFrcRateUnfiltered, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	Fr/s or Pr/s

Description

The Actual Pressure Rate (Unfiltered) or Actual Force Rate (Unfiltered) value is the unfiltered Actual Pressure Rate or Actual Force Rate. The control algorithms are not performed on the Actual Pressure Rate (Unfiltered) or Actual Force Rate (Unfiltered). Therefore, the Actual Pressure Rate (Unfiltered) and Actual Force Rate (Unfiltered) values are mostly for troubleshooting when verifying the filtering performance.



See the Actual Pressure/Force Rate and Filtering topics for more details.

See Also

Actual Pressure/Force Rate | Actual Pressure/Force Rate (Control) | Filtering

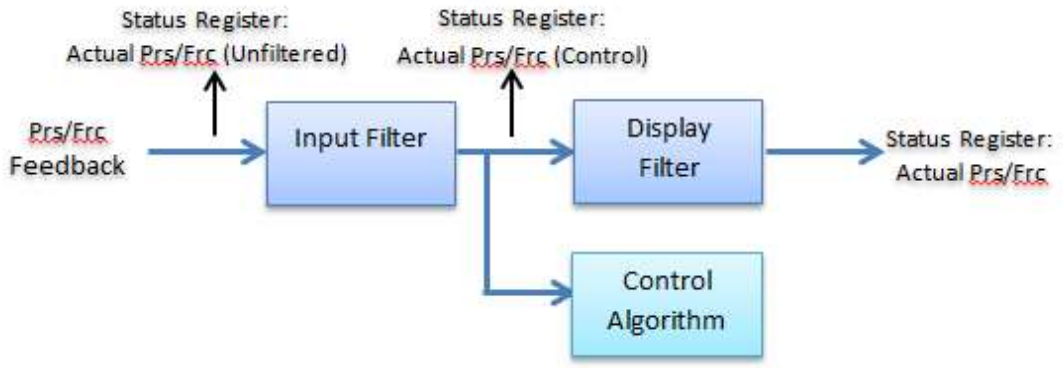
9.2.2.3.30. Actual Pressure/Force (Control)

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.36, where *n* = 256 + the axis number Secondary Input: %MD*n*.96, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].ActPrsControl, where *n* is the axis number

	Force Input: _Axis[*n*].ActFrcControl, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	Fr or Pr

Description

The Actual Pressure (Control) or Actual Force (Control) value is the pressure or force feedback with the control filter applied. This value is used by the control algorithm. The value may be filtered again by the display filter, and the final filtered value is given by the Actual Pressure/Force Rate status register.



The Actual Pressure (Control) or Actual Force (Control) is valuable for detailed troubleshooting because it shows the exact value that the control-algorithm is operating on, whereas the commonly used Actual Pressure or Actual Force is typically a filtered value that will differ slightly from the Actual Pressure (Control) or Actual Force (Control) value.

See the Actual Pressure/Force and Filtering topics for more details.

See Also

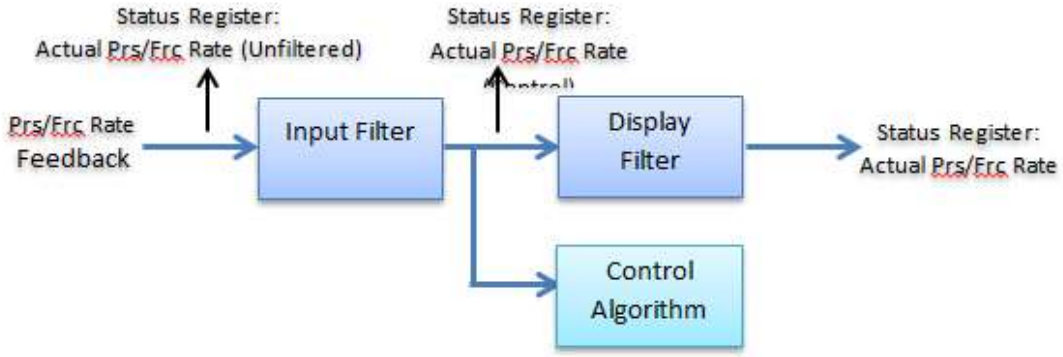
Actual Pressure/Force | Actual Pressure/Force (Unfiltered) | Filtering

9.2.2.3.31. Actual Pressure/Force Rate (Control)

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.37, where *n* = 256 + the axis number Secondary Input: %MD*n*.97, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].ActPrsRateControl, where *n* is the axis number **Force Input:** _Axis[*n*].ActFrcRateControl, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab, Feedback section
Data Type:	REAL
Units:	Fr/s or Pr/s

Description

The Actual Pressure Rate (Control) or Actual Force Rate (Control) value is the pressure or force rate with the control filter applied. This value is used by the control algorithm. The value may be filtered again by the display filter, and the final filtered value is given by the Actual Pressure/Force Rate status register.



The Actual Pressure/Force Rate (Control) is valuable for detailed troubleshooting because it shows the exact value that the control-algorithm is operating on, whereas the commonly used Actual Pressure Rate or Actual Force Rate is typically a filtered value that will differ slightly from the Actual Pressure Rate (Control) or Actual Force Rate (Control) value.

See the Actual Pressure/Force Rate and Filtering topics for more details.

See Also

Actual Pressure/Force Rate | Actual Pressure/Force Rate (Unfiltered) | Filtering

---

9.2.2.3.32. Millivolts/Volt

Type:	Axis Status Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** %MD*n*.26, where *n* = 256+ the axis number
System Tag:	_Axis[*n*].MillivoltsPerVolt
How to Find:	Axes Status Registers Pane, All tab:Feedback
Data Type:	REAL
Units:	mV/V

Description

The Millivolts/Volt status register is the feedback value from a load cell. It is calculated as follows:

Millivolts/Volt = Millivolt Input / Exciter Voltage

The Exciter Voltage depends on the Exciter Mode:

- **Nominal**

  The Exciter Voltage is the value of the load cell input module's Exciter Output as measured during the factory calibration process. For the LC8 module, the value may be in the range of 6.75 ±0.003 V. For more details, see LC8 Calibration.

- **Adaptive**

  The Exciter Voltage is the Effective Exciter Voltage, which uses the voltage of the Sense pin to calculate the actual excitation voltage at the load cell, to compensate for voltage drop.

- **Fixed**

  The Exciter Voltage is the Fixed Exciter Voltage. This is the value the user obtained by measuring the excitation voltage at the load cell.

The Millivolts/Volt status register is used to calculate the Actual Force of a load cell as follows:

If (**Millivolts/Volt +** Millivolts/Volt Offset) < 0 Then

Force = (**Millivolts/Volt** + Millivolts/Volt Offset) x NegCorrFactor x Force Scale + Force Offset

Else

Force = (**Millivolts/Volt** + Millivolts/Volt Offset) x Force Scale + Force Offset

EndIf

See Also

Load Cell Fundamentals | Millivolt Input | Millivolts/Volt Offset | Exciter Mode

---

9.2.2.3.33. Millivolt Input

Type:	Axis Status Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** %MD*n*.45, where *n* = 256+ the axis number
System Tag:	_Axis[*n*].MillivoltInput
How to Find:	Axes Status Registers Pane, All tab:Feedback
Data Type:	REAL
Units:	mV

Description

The Millivolt Input status register is the directly measured voltage of the In+ pin relative to the In- pin of the load cell input. This value is used to calculate the millivolt/volt feedback value as follows:

Millivolts/Volt = Millivolt Input / Exciter Voltage

The Exciter Voltage depends on the Exciter Mode, as described in the Millivolts/Volt topic.

The Millivolt Input status register may be used to verify that the RMC is reporting an accurate value as compared to a measurement with a precision voltmeter. Because the voltage is so small, a high-impedance voltmeter must be used to accurately measure the voltage externally.

See Also

Load Cell Fundamentals | Millivolts/Volt | Millivolts/Volt Offset | Exciter Mode

---

9.2.2.3.34. Effective Exciter Voltage

Type:	Axis Status Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** %MD*n*.46, where *n* = 256+ the axis number
System Tag:	_Axis[*n*].EffExciterVoltage
How to Find:	Axes Status Registers Pane, All tab:Feedback
Data Type:	REAL
Units:	V

Description

The Effective Exciter Voltage is the value of the exciter voltage at the load cell. This value is derived from the measured Wire Sense voltage as follows:

Effective Exciter Voltage = Exciter Voltage – 2 X Wire Sense Voltage

Where:

Exciter Voltage = Nominal exciter voltage of the LC8 module

Wire Sense Voltage = Voltage drop between Load Cell Cmn (Sense pin) and the RMC exciter Cmn (Exc-). The Wire Sense voltage itself is reported in raw form in Transducer Status A.

The Effective Exciter Voltage assumes the voltage drop of the Exc+ and Exc- wires is identical. Therefore, this value is valid only if the length and gauge of the Exc+ and Exc- wires are identical.

The Effective Exciter Voltage is always reported, but is only used by the calculation for Millivolts/Volt if the Exciter Mode is set to Adaptive. The Effective Exciter Voltage value can also be used to help set the Fixed Exciter Voltage parameter if the **Fixed Value** mode is selected.

See Also

Load Cell Fundamentals | Fixed Exciter Voltage | Exciter Mode | Millivolts/Volt

9.2.2.4. Output

9.2.2.4.1. Control Output

Type:	Axis Status Register
Address:	**RMC75/150:** %MD*n*.33, where *n* = 8 + the axis number **RMC200:** %MD*n*.140, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].ControlOutput, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	**RMC75/150:** V **RMC200:** %

Description

The Control Output status register is the output of the axis:

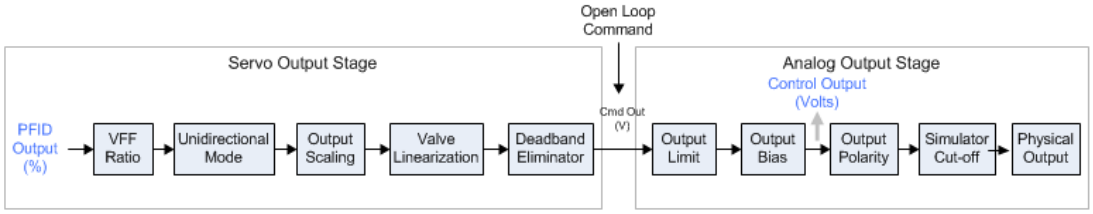
RMC75/150: The Control Output is volts, and can range from -10 to +10 V. This status register specified the physical voltage output, although it may also be inverted due to the Invert Output Polarity.

RMC200: The Control Output is percent, and can range from -100 to +100%. The Control Output percentage is then converted to the physical voltage or current output, which is given by the Final Output status register. The Output Type axis parameter defines the mapping of the Control Output to the Final Output.

Control Diagrams

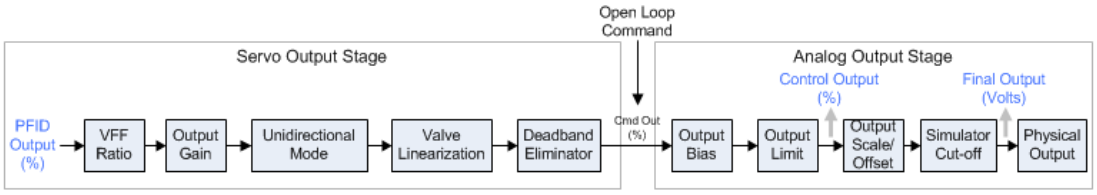
RMC75/150

The Control Output is the value after the Feed Forward Ratio, Output Scale, Valve Linearization, Output Deadband, Output Limit, and Output Bias, but before the Output Polarity. Therefore, if the Invert Output Polarity parameter is set, the measured voltage will be negative of what is displayed in this register.



RMC200

The Control Output is the value after the Feed Forward Ratio, Output Gain, Unidirectional Mode, Valve Linearization, Output Deadband, Output Bias, and Output Limit. The Control Output is subsequently mapped to the Final Output as indicated in the Output Scale/Offset block below. This mapping is defined by the Output Type parameter. The Output Type parameter offers pre-defined mappings, and custom mappings.



See Also

Final Output | Output Type

9.2.2.5. Primary Control

9.2.2.5.1. Position Error

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.35, where *n* = 8 + the axis number **RMC150:** %MD*n*.35, where *n* = 8 + the axis number **RMC200:** %MD*n*.180, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].PosError, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	pu

Description

The Position Error is the difference between the Target Position and Actual Position. If this value exceeds the Position Error Tolerance while controlling in Position PID, the Following Error bit will be set. If the Following Error bit is set, a Halt will occur if the Following Error Auto Stop is configured to do so and the Direct Output Status bit is off.

The Position Error Term is useful for troubleshooting when included in a plot.

See Also

Status Registers | Following Error

9.2.2.5.2. Velocity Error

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.36, where *n* = 8 + the axis number **RMC150:** %MD*n*.36, where *n* = 8 + the axis number **RMC200:** %MD*n*.181, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].VelError, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	pu/s

Description

The Velocity Error is the difference between the Target Velocity and Actual Velocity. If this value exceeds the Velocity Error Tolerance while controlling in Velocity PID, the Following Error bit will be set. If the Following Error bit is set, a Halt will occur if the Following Error Auto Stop is configured to do so and the Direct Output Status bit is off.

The Velocity Error is useful for troubleshooting when included in a plot.

See Also

Status Registers | Following Error

9.2.2.5.3. Proportional Output Term

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.37, where *n* = 8 + the axis number **RMC150:** %MD*n*.37, where *n* = 8 + the axis number **RMC200:** %MD*n*.182, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].PropOutputTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Proportional Output Term is the portion of control output contributed by the Proportional Gain, in units of percent per maximum Control Output. The Proportional Output Term is useful for troubleshooting when included in a plot.

See Proportional Gain for more details.

See Also

Status Registers | Proportional Gain

9.2.2.5.4. Integral Output Term

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.38, where *n* = 8 + the axis number **RMC150:** %MD*n*.38, where *n* = 8 + the axis number **RMC200:** %MD*n*.183, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].IntOutputTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Integral Output Term is the portion of the PFID Output contributed by the Integral Gain, in units of percent per maximum Control Output. The Integral Output Term is useful for troubleshooting when included in a plot.

See the Integral Gain topic for more details.

See Also

Status Registers | Integral Gain

9.2.2.5.5. Double Differential Output Term

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.44, where *n* = 8 + the axis number **RMC150:** %MD*n*.44, where *n* = 8 + the axis number **RMC200:** %MD*n*.185, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].Kd3OutputTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Double Differential Output Term is the portion of the PFID Output contributed by the Double Differential Gain, if Acceleration Control is enabled, or the Active Damping Differential Gain if Active Damping is enabled. The Double Differential Output Term is useful for troubleshooting when included in a plot.

See the Double Differential Gain and Active Damping Differential Gain topics for details.

See Also
Status Registers | Double Differential Gain

9.2.2.5.6. Differential Output Term

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.39, where *n* = 8 + the axis number **RMC150:** %MD*n*.39, where *n* = 8 + the axis number **RMC200:** %MD*n*.184, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].DiffOutputTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	% of maximum Control Oootput

Description

The Differential Output Term is the portion of the PFID Output contributed by the Differential Gain, in units of percent per maximum Control Output. The Differential Output Term is useful for troubleshooting when included in a plot.

See the Differential Gain topic for more details.

See Also
Status Registers | Differential Gain

9.2.2.5.7. Triple Differential Output Term

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.40, where *n* = 8 + the axis number **RMC150:** %MD*n*.40, where *n* = 8 + the axis number **RMC200:** %MD*n*.186, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].TrpDiffOutputTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Triple Differential Output Term is the portion of the PFID Output contributed by the Triple Differential Gain if Acceleration Control is enabled, or the Active Damping Differential Gain if Active Damping is enabled. The Triple Differential Output Term is useful for troubleshooting when included in a plot.

See the Triple Differential Gain and Active Damping Differential Gain topics for more details.

See Also

Status Registers | Triple Differential Gain

9.2.2.5.8. Acceleration Feed Forward Term

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.42, where *n* = 8 + the axis number **RMC150:** %MD*n*.42, where *n* = 8 + the axis number **RMC200:** %MD*n*.189, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].AccFFwdTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	% of maximum control output

Description

The Acceleration Feed Forward Term is the portion of the PFID Output contributed by the Acceleration Feed Forward. The Acceleration Feed Forward Term is useful for troubleshooting when included in a plot.

See the Acceleration Feed Forward topic for more details.

See Also

Status Registers | Acceleration Feed Forward

9.2.2.5.9. Velocity Feed Forward Term

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.41, where *n* = 8 + the axis number **RMC150:** %MD*n*.41, where *n* = 8 + the axis number **RMC200:** %MD*n*.188, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].VelFFwdTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Velocity Feed Forward Term is the portion of control output contributed by the Velocity Feed Forward, in units of percent per maximum Control Output. The Velocity Feed Forward Term is useful for troubleshooting when included in a plot.

See the Velocity Feed Forward topic for more details.

See Also

Status Registers | Velocity Feed Forward

9.2.2.5.10. Jerk Feed Forward Term

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.43, where *n* = 8 + the axis number **RMC150:** %MD*n*.43, where *n* = 8 + the axis number **RMC200:** %MD*n*.190, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].JerkFFwdTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Jerk Feed Forward Term is the portion of the PFID Output contributed by the Jerk Feed Forward, in units of percent per maximum Control Output. The Jerk Feed Forward Term is useful for troubleshooting when included in a plot.

See the Jerk Feed Forward topic for more details.

See Also

Status Registers | Jerk Feed Forward

9.2.2.5.11. PFID Output

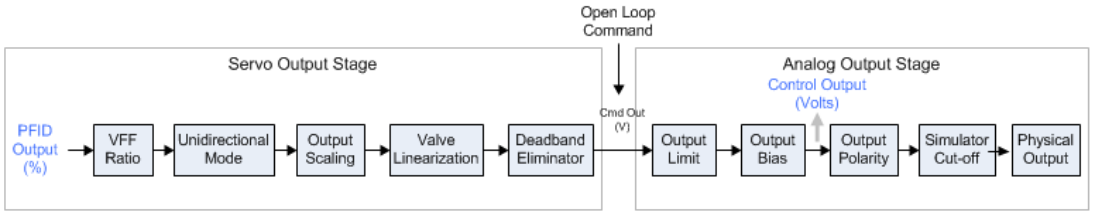
Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.45, where *n* = 8 + the axis number **RMC150:** %MD*n*.45, where *n* = 8 + the axis number **RMC200:** %MD*n*.151, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].PFIDOutput, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	REAL
Units:	% of maximum control output

Description

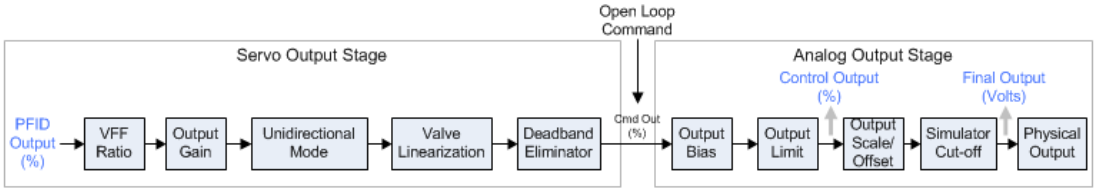
The PFID Output register is the sum of all control gains and feed forwards including proportional, integral, differential, and feed forward terms, plus any pressure/force limiting, acceleration control, or active damping contributions. In short, this register gives the intended control output percentage.

This value will be affected by the Output Scale, Feed Forward Ratio, Output Deadband, Output Limit, and Output Bias before generating an actual voltage on the Control Output. If the Invert Output Polarity parameter is set, the measured physical output voltage will be negative of what is displayed in this register.

RMC75/150 Output Diagram:



RMC200 Diagram:



See Also

Status Registers

9.2.2.5.12. Current Control Mode

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.6, where *n* = 8 + the axis number **RMC150:** %MD*n*.6, where *n* = 8 + the axis number **RMC200:** %MD*n*.150, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].CtlMode, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	DINT

Description

The **Current Control Mode** is the control mode that the axis is currently in. The **Current Control Mode** register does not indicate whether the axis is in pressure or force limit. Use the Status Bits register for pressure or force limit information.

Values	Control Mode
0	None
1	Direct Output
2	Open Loop
3	Position PID
4	Position I-PD
5	Velocity PID
6	Velocity I-PD
7	Pressure control (does not indicate pressure limit)

Any motion command that switches the control mode will affect this register. For example, issuing Direct Output (9) or Open Loop Rate (10) commands will change this register to values of **Direct Output** (1) and **Open Loop** (2). Also, issuing closed loop motion commands will switch the control mode to the control mode specified by the Next Pos/Vel Control Mode register. Use the Set Pos/Vel Ctrl Mode (68) command to change the value of this register, and thus control which closed loop control mode will be used.

See Also

Status Registers | Next Pos/Vel Control Mode | Default Pos/Vel Control Mode | Set Pos/Vel Ctrl Mode (68)

9.2.2.5.13. Next Pos/Vel Control Mode

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.7, where *n* = 8 + the axis number **RMC150:** %MD*n*.7, where *n* = 8 + the axis number **RMC200:** %MD*n*.171, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].NextPVCtlMode, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
Data Type:	DINT

Description

The **Next Pos/Vel Control Mode** status register indicates which control mode will be used when the next closed-loop position or velocity motion command is issued. The **Next Pos/Vel Control Mode** register applies only to control modes for position and velocity commands, not pressure or force limit.

Values	Control Mode
0	Position PID
1	Position I-PD
4	Velocity PID
5	Velocity I-PD
100	None (for virtual axes)

Use the Set Pos/Vel Ctrl Mode (68) command to change this register and select which control mode the next closed loop command will use. The Current Control Mode register shows the current control of the axis.

See Also

Status Registers | Current Control Mode | Default Pos/Vel Control Mode | Set Pos/Vel Ctrl Mode (68)

9.2.2.5.14. Current Integrator Mode

Type:	Axis Status Register

Address:	**RMC75:** %MD*n*.47, where *n* = 8 + the axis number **RMC150:** %MD*n*.47, where *n* = 8 + the axis number **RMC200:** %MD*n*.170, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].CurIntMode, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control-Primary
Data Type:	DINT

Description

The Current Integrator Mode register shows the current Mode of the Integrator, which can be set by the Default Integrator Mode axis parameter and the Set Integrator Mode (71) command.

The values correspond to the Integrator Mode as shown in the following table:

Value	Integrator Mode
0	Always Held
1	Always Active
2	Always Zero
3	TGDone
4	Decel

See the Default Integrator Mode topic for details on the modes.

The Current Integrator Mode status register is not available in firmware versions prior to 3.66.0 on the RMC75 and RMC150.

See Also

Default Integrator Mode | Set Integrator Mode (71)

9.2.2.5.15. Current Gain Set

Type:	Axis Status Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.192, where *n* = 256+ the axis number
System Tag:	_Axis[*n*].CurGainSet, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab
Data Type:	REAL

Description

The **Current Gain Set** axis status register shows which Gain Set is currently being applied by the axis. The current gain set may be selected with the Select Gain Set (75) command.

See Also

Gain Sets | Select Gain Set (75) Command

9.2.2.6. Secondary Control

9.2.2.6.1. Pressure/Force Error

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.35, where *n* = 8 + the axis number Secondary Input: %MD*n*.46, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.35, where *n* = 8 + the axis number Secondary Input: %MD*n*.46, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.180, where *n* = 8 + the axis number Secondary Input: %MD*n*.290, where *n* = 8 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].PrsError, where *n* is the axis number **Force Input:** _Axis[*n*].FrcError, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control Axes Status Registers Pane, All tab: Pressure/Force Control
Data Type:	REAL
Units:	Pressure units (Pr) or Force units (Fr)

Description

The Pressure/Force Error is the difference between the Target Pressure/Force and the Actual Pressure/Force. If this value exceeds the Pressure/Force Error Tolerance while in pressure/force control or while pressure/force limit is enabled in Open Loop, Position PID, or Velocity PID control modes, the Pressure/Force Following Error bit will be set. If the Pressure or Force Following Error bit is set, a Halt will occur if the Pressure/Force Following Error Auto Stop is configured to do so and the Direct Output Status bit is off.

In the Negative and Bidirectional modes of Pressure/Force Limit, the Pressure/Force Error will not give the error between the negated Pressure/Force Target and the Actual Pressure/Force. If you need to monitor this error, use the expressions in a user program.

This register is useful for troubleshooting when included in a plot.

See Also

Status Registers | Pressure/Force Following Error bit

9.2.2.6.2. Pressure/Force Proportional Term

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.37, where *n* = 8 + the axis number Secondary Input: %MD*n*.48, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.37, where *n* = 8 + the axis number Secondary Input: %MD*n*.48, where *n* = 8 + the axis number **RMC200:**

	Primary Input: %MD*n*.182, where *n* = 256 + the axis number Secondary Input: %MD*n*.292, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].PrsPropGainTerm, where *n* is the axis number **Force Input:** _Axis[*n*].FrcPropGainTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control Axes Status Registers Pane, All tab: Pressure/Force Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Pressure/Force Proportional Output Term is the portion of control output contributed by the Pressure/Force Proportional Gain, in units of percent per maximum Control Output. The Pressure/Force Proportional Output Term is useful for troubleshooting when included in a plot.

See Pressure/Force Proportional Gain for more details.

See Also

Status Registers | Pressure/Force Proportional Gain

9.2.2.6.3. Pressure/Force Integral Term

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.38, where *n* = 8 + the axis number Secondary Input: %MD*n*.47, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.38, where *n* = 8 + the axis number Secondary Input: %MD*n*.47, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.183, where *n* = 256 + the axis number Secondary Input: %MD*n*.293, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].PrsIntGainTerm, where *n* is the axis number **Force Input:** _Axis[*n*].FrcIntGainTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control Axes Status Registers Pane, All tab: Pressure/Force Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Pressure/Force Integral Term is the portion of control output contributed by the Pressure/Force Integral Gain, in units of percent per maximum Control Output. The Pressure/Force Integral Term is useful for troubleshooting when included in a plot.

See Pressure/Force Integral Gain for more details.

See Also

Status Registers | Pressure/Force Integral Gain

9.2.2.6.4. Pressure/Force Differential Term

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.39, where *n* = 8 + the axis number Secondary Input: %MD*n*.50, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.39, where *n* = 8 + the axis number Secondary Input: %MD*n*.50, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.184, where *n* = 256 + the axis number Secondary Input: %MD*n*.294, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].PrsDiffGainTerm, where *n* is the axis number **Force Input:** _Axis[*n*].FrcDiffGainTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control Axes Status Registers Pane, All tab: Pressure/Force Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Pressure/Force Differential Term is the portion of the PFID output contributed by the Pressure or Force Differential Gain, in units of percent per maximum Control Output. The Pressure/Force Differential Term is useful for troubleshooting when included in a plot.

See Pressure/Force Differential Gain for more details.

See Also

Status Registers | Pressure/Force Differential Gain

9.2.2.6.5. Pressure/Force Feed Forward Term

Type:	Axis Status Register
Address:	**RMC75:** Primary Input: %MD*n*.40, where *n* = 8 + the axis number Secondary Input: %MD*n*.51, where *n* = 8 + the axis number **RMC150:** Primary Input: %MD*n*.40, where *n* = 8 + the axis number Secondary Input: %MD*n*.51, where *n* = 8 + the axis number **RMC200:** Primary Input: %MD*n*.187, where *n* = 256 + the axis number Secondary Input: %MD*n*.297, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].PrsFFwdTerm, where *n* is the axis number

	Force Input: _Axis[*n*].FrcFFwdTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
	Axes Status Registers Pane, All tab: Pressure/Force Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Pressure/Force Feed Forward Term is the portion of the control output contributed by the Pressure/Force Feed Forward Term, in units of percent per maximum Control Output. The Pressure/Force Feed Forward Term is useful for troubleshooting when included in a plot.

See Pressure/Force Feed Forward Gain for more details.

See Also

Status Registers | Pressure/Force Feed Forward

9.2.2.6.6. Pressure/Force Rate Feed Forward Term

Type:	Axis Status Register
Address:	**RMC75:**
	Primary Input: %MD*n*.41, where *n* = 8 + the axis number
	Secondary Input: %MD*n*.52, where *n* = 8 + the axis number
	RMC150:
	Primary Input: %MD*n*.41, where *n* = 8 + the axis number
	Secondary Input: %MD*n*.52, where *n* = 8 + the axis number
	RMC200:
	Primary Input: %MD*n*.188, where *n* = 256 + the axis number
	Secondary Input: %MD*n*.298, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].PrsRateFFwdTerm, where *n* is the axis number
	Force Input: _Axis[*n*].FrcRateFFwdTerm, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Control
	Axes Status Registers Pane, All tab: Pressure/Force Control
Data Type:	REAL
Units:	% of maximum Control Output

Description

The Pressure/Force Rate Feed Forward Term is the portion of control output contributed by the Pressure/Force Rate Feed Forward Gain, in units of percent per maximum Control Output. The Pressure/Force Rate Feed Forward Term is useful for troubleshooting when included in a plot.

See the Pressure/Force Rate Feed Forward topic for more details.

See Also

Status Registers | Pressure/Force Rate Feed Forward

9.2.2.7. Target

9.2.2.7.1. Command Position

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.56, where *n* = 8 + the axis number **RMC150:** %MD*n*.56, where *n* = 8 + the axis number **RMC200:** %MD*n*.404, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].CmdPos, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	pu

Description

The Command Position is the requested position with travel limits applied. If the requested position is outside the Positive or Negative Travel Limit, the Command Position will be set to the value of the limit, and the axis will go only to the limit. The Command Position is updated when any motion command is issued with a Requested Position command parameter.

For commands that do not have a requested position (velocity, gearing, or open loop commands), this register will follow the Target Position.

Why Bother?

If the Command Position is not the same as the requested position then one of two things has happened:

1. A program error has asked the axis to go to an invalid position. In this case the program error should be corrected.
2. The Command Value field has just been changed and the Motion Controller has not had a chance to acknowledge the new request.

See Also

Status Registers

9.2.2.7.2. Command Velocity

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.57, where *n* = 8 + the axis number **RMC150:** %MD*n*.57, where *n* = 8 + the axis number **RMC200:** %MD*n*.405, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].CmdVel, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	pu/sec

Description

For velocity commands, the Command Velocity is the requested final speed. In any other control type, it is the same value as the Target Velocity.

See Also
Status Registers

9.2.2.7.3. Target Position

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.53, where *n* = 8 + the axis number **RMC150:** %MD*n*.53, where *n* = 8 + the axis number **RMC200:** %MD*n*.400, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].TarPos, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	pu

Description

When the axis is in Closed Loop control, the Target Position is the calculated instantaneous ideal position for the axis. The Target Position is calculated every loop by the target generator of the RMC. During a move the path of the Target Position toward the Command Position will be the perfect profile for the Actual Position to follow.

The control algorithm uses the difference between the Target Position and the Actual Position to compute any required corrective Control Output.

When an axis is stopped, the Target Position should be the same as the Command Position unless an error or halt has occurred. When an axis is in Open Loop control, the Target Position is set to the Actual Position.

Why Bother?

Knowing the relationship between the Target Position and Actual Position is key to tuning the axis. The main goal in tuning the axis is to minimize the error between the Target and Actual Positions. The plot function is a very useful visual aid in tuning the axis.

See Also
Status Registers

9.2.2.7.4. Target Velocity

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.54, where *n* = 8 + the axis number **RMC150:** %MD*n*.54, where *n* = 8 + the axis number **RMC200:** %MD*n*.401, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].TarVel, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	pu/sec

Description

When the axis is in Closed Loop control, the Target Velocity is the desired instantaneous velocity for the axis, based on the last motion command that was sent. The Target Velocity is calculated every loop by the target generator of the RMC.

When a velocity command is sent to the axis, the Command Velocity will immediately be set to the final requested velocity while the Target Velocity will ramp according to the command.

When a position command is issued, the Command Velocity will be set to the same value as the Target Velocity.

See Also

Status Registers

9.2.2.7.5. Target Acceleration

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.55, where *n* = 8 + the axis number **RMC150:** %MD*n*.55, where *n* = 8 + the axis number **RMC200:** %MD*n*.402, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].TarAcc, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Target
Data Type:	REAL
Units:	pu/sec^2

Description

When the axis is in Closed Loop control, the Target Acceleration is the calculated instantaneous ideal acceleration for the axis. The Target Acceleration is calculated every loop by the target generator of the RMC.

See Also

Status Registers | Actual Acceleration

9.2.2.7.6. Target Jerk

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.58, where *n* = 8 + the axis number **RMC150:** %MD*n*.58, where *n* = 8 + the axis number **RMC200:** %MD*n*.403, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].TarJrk, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Target
Data Type:	REAL
Units:	pu/sec^3

Description

When the axis is in Closed Loop control, the Target Jerk is the calculated instantaneous ideal jerk (rate of change of acceleration) for the axis. The Target Jerk is calculated every loop by the target generator of the RMC.

See Also

Status Registers | Actual Jerk

9.2.2.7.7. Cycles

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.59, where *n* = 8 + the axis number **RMC150:** %MD*n*.59, where *n* = 8 + the axis number **RMC200:** %MD*n*.410, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].Cycles, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Target
Data Type:	DINT

Description

The Cycles status register gives the whole number of cycles that the current position sine move or curve has completed. After a sine move or curve has completed or been interrupted, this will remain at its current value. This value is reset each time a position sine move or curve is started.

For continuous moves (without a fixed number of cycles), this value will wrap to zero after it reaches 10,000,000 and then continue incrementing. For moves with a fixed number of cycles, this value will not go beyond the requested cycle count.

If you need more status information on the sine move or curve, such as the cycle fraction, you can use the Status Block defined by the Sine Start (72) and Curve Start Advanced (88) commands.

See Also

Status Registers | Sine Start (72) | Sine Start (Prs/Frc) (76) | Curve Start (86) | Curve Start (Prs/Frc) (87)

9.2.2.7.8. Command Pressure/Force

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.61, where *n* = 8 + the axis number **RMC150:** %MD*n*.61, where *n* = 8 + the axis number **RMC200:** %MD*n*.444, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].CmdPrs, where *n* is the axis number **Force Input:** _Axis[*n*].CmdFrc, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	Pr or Fr

Description

The Command Pressure or Force is the requested pressure or force. For Pressure or Force Limit, this will hold the requested Pressure/Force Limit value.

See Also

Status Registers

9.2.2.7.9. Target Pressure/Force

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.60, where *n* = 8 + the axis number **RMC150:** %MD*n*.60, where *n* = 8 + the axis number **RMC200:** %MD*n*.440, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].TarPrs, where *n* is the axis number **Force Input:** _Axis[*n*].TarFrc, where *n* is the axis number
How to Find:	Axes Status Registers Pane, Basic tab
Data Type:	REAL
Units:	Pr or Fr

Description

In Pressure/Force control, the Target Pressure/Force is the instantaneous desired Pressure/Force value for the axis.

In Pressure/Force Limit, the Target Pressure/Force is the pressure/force value that should not be exceeded. The control algorithm will try to limit the motion such that the Actual Pressure or Force does not exceed the Target Pressure/Force.

The Target Pressure/Force is calculated every loop by the target generator of the RMC. When the axis is not in Pressure/Force Control or Pressure/Force Limit, the Target Pressure/Force can still be ramped up or down, but will have no effect on control.

See Pressure/Force Control Overview for more details.

See Also

Status Registers

9.2.2.7.10. Target Pressure/Force Rate

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.63, where *n* = 8 + the axis number **RMC150:** %MD*n*.63, where *n* = 8 + the axis number **RMC200:** %MD*n*.441, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].TarPrsRate, where *n* is the axis number **Force Input:** _Axis[*n*].TarFrcRate, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Feedback or Pressure/Force/Accel Feedback

Data Type:	REAL
Units:	Pr/s or Fr/s

Description

Target Pressure/Force Rate is the rate of change of the Target Pressure/Force. The Target Pressure/Force Rate is calculated every loop by the target generator of the RMC.

See Also

Status Registers

9.2.2.7.11. Cycles (Pressure or Force)

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.62, where *n* = 8 + the axis number **RMC150:** %MD*n*.62, where *n* = 8 + the axis number **RMC200:** %MD*n*.450, where *n* = 256 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].CyclesPrs, where *n* is the axis number **Force Input:** _Axis[*n*].CyclesFrc, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Target
Data Type:	DINT

Description

The Cycles (Pressure or Force) status register gives the whole number of cycles that the current Pressure or Force sine move or curve has completed. After a sine move or curve has completed or been interrupted, this register will remain at its current value. This value is reset each time a position sine move or curve is started.

For continuous moves (without a fixed number of cycles), this value will wrap to zero after it reaches 10,000,000 and then continue incrementing. For moves with a fixed number of cycles, this value will not go beyond the requested cycle count.

If you need more status information on the sine move or curve, such as the cycle fraction, you can use the Status Block defined by the Sine Start (Prs/Frc) (76) or and Curve Start Advanced (Prs/Frc) (89) commands.

See Also

Status Registers | Sine Start (72) | Sine Start (Prs/Frc) (76) | Curve Start (86) | Curve Start (Prs/Frc) (87)

9.2.2.8. Home/Registration

9.2.2.8.1. Registration 0 Position

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.19, where *n* = 8 + the axis number **RMC150:** %MD*n*.19, where *n* = 8 + the axis number **RMC200:** %MD*n*.56, where *n* = 256 + the axis number

System Tag:	_Axis[*n*].Reg0Pos, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Registration
Data Type:	REAL

Description

This status register stores the "registration position 0" obtained by a registration event. This value will be valid until another "registration 0" event occurs.

See Also

Status Registers | Registration 1 Position | Registration

9.2.2.8.2. Registration 1 Position

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.20, where *n* = 8 + the axis number **RMC150:** %MD*n*.20, where *n* = 8 + the axis number **RMC200:** %MD*n*.57, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].Reg1Pos, where *n* is the axis number
How to Find:	Axes Status Registers Pane, All tab: Registration
Data Type:	REAL

Description

This status register stores the "registration position 1" obtained by a registration event. This value will be valid until another "registration 1" event occurs.

See Also

Status Registers | Registration 0 Position | Registration

Show All

9.2.2.8.3. Encoder Status

Note to Help Editor:
To add a Bit:

- **Make anew row, then copy and from an existing row**
- **Modify the bookmark.**
- **In the TRUE CODE, make sure the bookmark has quotes around it.**
- **Change the id of the dropdown text to the same as the bookmark, but with an "x" in front of it. E.g., "Home Input" and "xHome Input"**
- **Make sure to test it.**

Type:	Axis Status Register
Address:	**RMC75:** %MD*n*.18, where *n* = 8 + the axis number **RMC150:** %MD*n*.18, where *n* = 8 + the axis number

	RMC200: %MD*n*.55, where *n* = 256 + the axis number
System Tag:	_Axis[*n*].EncStatus, where *n* is the axis number
How to Find:	See individual parameters below
Data Type:	DWORD – See **Bits** below

Description

The Encoder Status register is an Axis Status Register. It is a collection of bits that provide a summary of a quadrature encoder. This register is only available for the following modules:

- RMC75 QA and Q1
- RMC150 Quadrature (Q) and Universal I/O
- RMC200 Q4, S8, U14, and D24

Bits

RMC75/150

Bit	Tag Name	Register Name	How to Find in Axis Status Registers Pane
0	AIn	A Input Indicates the state of the quadrature A input (0=Off, 1=On). Available on all axes with quadrature inputs. This status bit is valid only if the A Wire Break bit is off. The A Input and the A Wire Break are combined into one tag name, AIn, as described below. In the Axis Tools, these are represented as **On**, **Off**, or **Break**.	All tab: Feedback
1		A Wire Break Indicates whether a broken wire is detected on the quadrature A input. Not available on the RMC150 Quad module. 0=No break 1=Break Detected In the Axis Tools, the A Input status bit and the A Wire Break are combined into one cell with the values **On**, **Off**, or **Break**.	All tab: Feedback

A Wire Break (bit 1)	A Input (bit 0)	Value of AIn Tag	Description
0	0	0	A Input On
0	1	1	A Input Off
1	0	2	Wire Break Detected
1	1	3	Wire Break Detected

2 BIn B Input

Indicates the state of the quadrature B input (0=Off, 1=On). Available on all axes with quadrature inputs.

This status bit is valid only if the B Wire Break bit is off.

The B Input and the B Wire Break are combined into one tag name, BIn, as described below. In the Axis Tools, these are represented as **On**, **Off**, or **Break**.

B Wire Break (bit 1)	B Input (bit 0)	Value of BIn Tag	Description
0	0	0	B Input On
0	1	1	B Input Off
1	0	2	Wire Break Detected
1	1	3	Wire Break Detected

All tab: Feedback

3 B Wire Break

Indicates whether a broken wire is detected on the quadrature B input. Not available on the RMC150 Quad module.

0=No break

1=Break Detected

In the Axis Tools, the B Input status bit and the B Wire Break are combined into one cell with the values **On**, **Off**, or **Break**.

All tab: Feedback

4 ZIn Index (Z) Input

Indicates the state of the Index (Z) input (0=Off, 1=On). The status of this bit is valid only if the Index (Z) Wire Break bit is off.

The index (Z) Input and the Index (Z) Wire Break are combined into one tag name, ZIn, as described below. In the Axis Tools, these are represented as **On**, **Off**, or **Break**.

The Index (Z) input is used for Homing.

Z Wire Break (bit 1)	Z Input (bit 0)	Value of ZIn Tag	Description
0	0	0	Z Input On
0	1	1	Z Input Off
1	0	2	Wire Break Detected

All tab: Home

1	1	3	Wire Break Detected

5		Index (Z) Wire Break Indicates whether a broken wire is detected on the quadrature Index (Z) input. Not available on the RMC150 Quad module. 0=No break 1=Break Detected In the Axis Tools, the Index (Z) Input status bit and the Index (Z) Wire Break are combined into one cell with the values **On**, **Off**, or **Break**.	All tab: Home
6	HomeIn	Home Input Indicates the state of the Home input (0=Off, 1=On). See the Homing topic for more details.	All tab: Home
7	RegXIn	Reg or RegX Input Indicates the state of the **Reg** or **RegX** input for this axis. See the Registration topic for details.	All tab: Registration
8	RegYIn	RegY Input Indicates the state of the **RegY** input for this axis. See the Registration topic for details.	All tab: Registration
9	HomeArmed	Home Armed This bit is set when the Home is armed.	All tab: Home
10	HomeLatched	Home Latched This bit is set when a Home event has been triggered. Once the home event occurred, this bit will remain set until the axis is re-armed if the homing is set for manual re-arm.	All tab: Home
11	Reg0Armed	Registration 0 Armed This bit is set when Registration 0 is armed.	All tab: Registration
12	Reg0Latched	Registration 0 Latched This bit is set when the Registration 0 event has occurred. This bit will remain set until the registration is re-armed.	All tab: Registration
13	Reg1Armed	Registration 1 Armed This bit is set when Registration 1 is armed.	All tab: Registration
14	Reg1Latched	Registration 1 Latched This bit is set when the Registration 1 event has occurred. This bit will remain set until the registration is re-armed.	All tab: Registration
15	LearnZAlign	Learning Z Alignment	All tab: Home

This bit is set after the Learn Z Alignment (54) command is issued and remains on until the axis has learned the Index (Z) alignment.

RMC200

Bit	Tag Name	Register Name	Applies to	How to Find in Axis Status Registers
0	AIn	A Input Indicates the state of the quadrature A input (0=Off, 1=On). Available on all axes with quadrature inputs. This status bit is valid only if the A Wire Break and A Fault bits are off. The A Input, A Wire Break, and A Fault are combined into one tag name, AIn, as described below. At most one of these bits will be set at at any given time. In the Axis Tools, these states are represented as **On**, **Off**, **Break**, or **Fault.**	Q4, S8, U14, D24	All tab: Feedback

A Fault (bit 2)	A Wire Break (bit 1)	A Input (bit 0)	Value of AIn Tag	Description
0	0	0	0	A Input On
0	0	1	1	A Input Off
0	1	0	2	Wire Break Detected
1	0	0	4	Fault Detected

1		A Wire Break Indicates whether a broken wire is detected on the quadrature A input. Available on the RMC200 Q4 and U14 and also on the D24 module when the wire break option is selected. 0=No break 1=Break Detected In the Axis Tools, the A Input, A Wire Break, and A Fault status bits are combined into one cell with the values **On**, **Off**, **Break**, or **Fault**.	Q4, U14, D24	All tab: Feedback

2		A Fault Indicates that the A+ or A- input voltage exceeded the allowable range. Refer to the Fault Voltage specification of the RMC200 Q4 and U14 modules. 0 = No Fault 1 = Fault In the Axis Tools, the A Input, A Wire Break, and A Fault status bits are combined into one cell with the values **On**, **Off**, **Break**, or **Fault**.	Q4, U14	All tab: Feedback
3	BIn	B Input Indicates the state of the quadrature B input (0=Off, 1=On). Available on all axes with quadrature inputs. This status bit is valid only if the B Wire Break and B Fault bits are off. The B Input, B Wire Break, and B Fault are combined into one tag name, BIn, as described below. At most one of these bits will be set at at any given time. In the Axis Tools, these states are represented as **On**, **Off**, **Break**, or **Fault.**	Q4, U14, S8, D24	All tab: Feedback
4		B Wire Break Indicates whether a broken wire is detected on the quadrature B input. Available on the RMC200 Q4 and U14 and also on the D24 module when the wire break option is selected. 0=No break 1=Break Detected In the Axis Tools, the B Input, B Wire Break, and B Fault status bits are combined into one cell with the values **On**, **Off**, **Break**, or **Fault**.	Q4, U14, D24	All tab: Feedback

B Fault (bit 5)	**B Wire Break (bit 4)**	**B Input (bit 3)**	**Value of BIn Tag**	**Description**
0	0	0	0	B Input On
0	0	1	1	B Input Off
0	1	0	2	Wire Break Detected
1	0	0	4	Fault Detected

5 — B Fault — Q4, U14 — All tab: Feedback

Indicates that the B+ or B- input voltage exceeded the allowable range. Refer to the Fault Voltage specification of the RMC200 Q4 and U14 modules.

0 = No Fault

1 = Fault

In the Axis Tools, the B Input, B Wire Break, and B Fault status bits are combined into one cell with the values **On**, **Off**, **Break**, or **Fault**.

6 — ZIn — Index (Z) Input — Q4, U14, D24 — All tab: Home

Indicates the state of the Index (Z) input (0=Off, 1=On). The status of this bit is valid only if the Index (Z) Wire Break and Index (Z) Fault bits are off.

The index (Z) Input, Index (Z) Wire Break, and Index (Z) Fault are combined into one tag name, ZIn, as described below. At most one of these bits will be set at at any given time. In the Axis Tools, these states are represented as **On**, **Off**, **Break**, or **Fault.**

The Index (Z) input is used for Homing.

Z Fault (bit 8)	**Z Wire Break (bit 7)**	**Z Input (bit 6)**	**Value of ZIn Tag**	**Description**
0	0	0	0	Z Input On
0	0	1	1	Z Input Off
0	1	0	2	Wire Break Detected
1	0	0	4	Fault Detected

7 — Index (Z) Wire Break — Q4, U14 — All tab: Home

Indicates whether a broken wire is detected on the quadrature Index (Z) input. Available on the RMC200 Q4 and U14.

0=No break

1=Break Detected

In the Axis Tools, the Index (Z) Input, Index (Z) Wire Break, and Index (Z) Fault status bits are combined into one cell with the values **On**, **Off**, **Break**, or **Fault**.

8		Index (Z) Fault Indicates that the Z+ or Z- input voltage exceeded the allowable range. Refer to the Fault Voltage specification of the RMC200 Q4 and U14 modules. 0 = No Fault 1 = Fault In the Axis Tools, the Index (Z) Input, Index (Z) Wire Break, and Index (Z) Fault status bits are combined into one cell with the values **On**, **Off**, **Break**, or **Fault**.	Q4, U14	All tab: Home
9	LearnZAlign	Learning Z Alignment This bit is set after the Learn Z Alignment (54) command is issued and remains on until the axis has learned the Index (Z) alignment.	Q4, U14, D24	All tab: Home
10	HomeIn	Home Input Indicates the state of the Home input (0=Off, 1=On). This status bit is only valid if the Home Fault status bit is off. The Home Input and Home Fault bits are combined into one tag name, HomeIn, as described below. In the Axis Tools, these states are represented as **On**, **Off**, and **Fault.** **Home Fault (bit 11) / Home Input (bit 10) / Value of HomeIn Tag / Description** 0 / 0 / 0 / Home Input Off 0 / 1 / 1 / Home Input On 1 / 0 / 2 / Fault Detected See the Homing topic for more details.	Q4	All tab: Home
11		Home Fault Indicates that the Home input voltage exceeded the allowable range. Refer to the Home input specification of the RMC200 Q4 module. 0 = No Fault 1 = Fault In the Axis Tools, the Home Input and Home Fault status bits are combined into one cell with the values **On**, **Off**, or **Fault**.	Q4	All tab: Home
12	HomeArmed	Home Armed This bit is set when the Home is armed.	Q4, U14, D24	All tab: Home
13	HomeLatched	Home Latched This bit is set when a Home event has been triggered.	Q4, U14, D24	All tab: Home

		Once the home event occurred, this bit will remain set until the axis is re-armed if the homing is set for manual re-arm.		
14	RegIn	Registration Input Indicates the state of the registration input for this axis.	Q4	All tab: Registration
15	Reg0Armed	Registration 0 Armed This bit is set when Registration 0 is armed.	Q4, U14, D24	All tab: Registration
16	Reg0Latched	Registration 0 Latched This bit is set when the Registration 0 event has occurred. This bit will remain set until the registration is re-armed.	Q4, U14, D24	All tab: Registration
17	Reg1Armed	Registration 1 Armed This bit is set when Registration 1 is armed.	Q4, U14	All tab: Registration
18	Reg1Latched	Registration 1 Latched This bit is set when the Registration 1 event has occurred. This bit will remain set until the registration is re-armed.	Q4, U14	All tab: Registration

See Also

Status Registers | Homing | Registration

9.2.3. Axis Parameter Registers

9.2.3.1. Axis Parameter Registers Overview

The Axis Parameter Registers contain configuration information for each axis. These registers are editable. Each Register is a 32-bit word.

In general, the parameter registers can be changed at any time. Certain registers, especially the axis feedback setup parameters, require the axis to be in a disabled state in order to change them. Tuning parameters can be changed at any time, including during motion.

For a list of the Parameter Registers, see the Register Maps topic.

Viewing and Editing Parameter Registers

Use the Axis Tools to view, edit and download the Axis Parameters.

To edit a parameter:

1. In the Axis Tools, in the Axes Parameters Pane, click the cell of the parameter you want to edit and enter a valid entry.
2. Click the **Download** button to download the parameter to the RMC.

Note:
Certain parameters that affect motion require that the Direct Output Status bit be on, or the Enabled bit be off before the parameter can be changed. If you are editing the parameter from RMCTools, this will automatically be done for you.

This does not apply if the parameter is being changed with a command, such as with the Offset Position (47) command.

Tag Names

Tag names for axis status and parameter registers use the format _Axis[*x*].*reg*, where *x* specifies the axis number and *reg* is the tag name for that register. For example, _Axis[2].ActPos is the Actual Position of Axis 2.

If there is no number in the brackets, such as _Axis[].ActPos, the axis is the current axis for the current task. See the Tasks topic for more details.

See Also

Register Maps | Status Registers

9.2.3.2. Feedback

9.2.3.2.1. Position Scale

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.0, where *n* = 12 + the axis number **RMC150:** %MD*n*.0, where *n* = 24 + the axis number **RMC200:** %MD*n*.20, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].PosScale, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	pu/Counts or pu/V or pu/mA
Range:	<> 0
Default Value:	1

Description

This parameter is used together with the Position Offset parameter to convert the transducer feedback Counts to an Actual Position. The Position Scale specifies how many position units equal one transducer feedback Count.

To reverse the direction of the feedback, use a negative Position Scale.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Setting the Scale

In general, to set the scale, determine how many Counts correspond to one position unit (inches, meters, etc., as desired). The Position Scale is the inverse of that number. For specific details on converting counts to position units refer to the appropriate topic:

- MDT Scaling
- SSI Scaling
- Quadrature Scaling
- Analog Position Scaling

- Resolver Scaling

Why bother?

It is important to specify the measurement units to be used for each axes. Setting the Scale converts the information from the transducer into meaningful measurement units.

See Also

Parameter Registers

9.2.3.2.2. Position Offset

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.1, where *n* = 12 + the axis number **RMC150:** %MD*n*.1, where *n* = 24 + the axis number **RMC200:** %MD*n*.21, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].PosOffset, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	pu
Range:	all
Default Value:	0

Description

Linear Axes

This parameter is used on position axes together with the Position Scale parameter to convert the transducer feedback Counts to an Actual Position. After the Counts are scaled to position units, this parameter is used to shift the position units.

Rotary Axes

This parameter is used on rotary position axes together with the Position Unwind and Count Unwind parameters to convert the transducer feedback Counts to an Actual Position.

The Count Unwind defines the number of counts in one machine cycle. The Position Unwind defines the number of position units per machine cycle. The Position Offset defines the value at which those position units begin.

Examples:

Position Unwind	Position Offset	Position Range
360	0	0 to 360
360	-180	-180 to 180
360	60	60 to 420

Absolute Rotary axes also use the Count Offset parameter to move the zero point of the Counts. The position range begins where the Counts are zero.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the

axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Setting the Offset

To set the Position Offset, first set the Position Scale. Then, determine the position, P_0, at which the Actual Position should be zero. Set the Position Offset to $-P_0$. For details on scaling counts to Position Units, see the appropriate topic:

- MDT Scaling
- SSI Scaling
- Analog Position Scaling
- Resolver Scaling

See Also

Parameter Registers

9.2.3.2.3. Velocity Scale

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.0, where *n* = 12 + the axis number **RMC150:** %MD*n*.0, where *n* = 24 + the axis number **RMC200:** %MD*n*.20, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].VelScale, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	pu/sec/V or pu/sec/mA
Range:	<> 0
Default Value:	1

Description

This parameter is used together with the Velocity Offset parameter to convert the transducer Voltage or Current to an Actual Velocity. The Velocity Scale specifies how many position units equal one transducer feedback Count. To reverse the direction of the feedback, use a negative Velocity Scale.

Actual Velocity = Velocity Scale x [(Voltage or Current) + Velocity Offset]

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Setting the Velocity Scale

Set the Velocity Offset before setting the Velocity Scale. To set the Velocity Offset parameter, make sure the tachometer is stopped. Then adjust the Velocity offset until the Actual Velocity is zero.

To set the scale, determine how many volts or milliamps correspond to one velocity unit (inches/sec, meters/sec, etc., as desired). The Velocity Scale is the inverse of that number.

For more details, see the Analog Velocity Scaling topic.

Why bother?

It is important to specify the measurement units to be used for each axes. Setting the Scale converts the information from the transducer into meaningful measurement units.

See Also

Velocity Offset | Velocity Deadband | Velocity Control | Actual Velocity Filter

9.2.3.2.4. Velocity Offset

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.1, where *n* = 12 + the axis number **RMC150:** %MD*n*.1, where *n* = 24 + the axis number **RMC200:** %MD*n*.21, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].VelOffset, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	V or mA
Range:	any
Default Value:	0

Description

On velocity input axes, this parameter is used together with the Velocity Scale to calculate the Actual Velocity from the input.

Actual Velocity = Velocity Scale x [(Voltage or Current) + Velocity Offset]

Setting the Velocity Offset

Set the Velocity Offset before setting the Velocity Scale. To set the Velocity Offset parameter, make sure the tachometer is stopped. Then adjust the Velocity offset until the Actual Velocity is zero. For more details, see the Analog Velocity Scaling topic.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Velocity Scale | Velocity Deadband | Velocity Control | Actual Velocity Filter

9.2.3.2.5. Velocity Deadband

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.2, where *n* = 12 + the axis number **RMC150:** %MD*n*.2, where *n* = 24 + the axis number **RMC200:** %MD*n*.27, where *n* = 384 + the axis number

System Tag:	_Axis[*n*].VelDeadband, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	REAL
Units:	Volts or mA
Range:	any
Default Value:	0

Description

This parameter is used on velocity input axes to specify the voltage or current input range in which the velocity is zero. If the absolute value of the voltage or current is outside of this value, this value is always subtracted from (added to, if the voltage or current is negative) the voltage or current input before being converted to the Actual Velocity.

Since velocity feedback is an analog signal, it is impossible to be exactly zero. Therefore, this parameter allows you to specify a range in which the velocity is assumed to be zero. This helps determine when the axis is actually stopped.

If this value is zero the deadband is disabled.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Velocity Scale | Velocity Offset | Velocity Control | Actual Velocity Filter

---

9.2.3.2.6. Acceleration Scale

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.0, where *n* = 12 + the axis number Secondary Input: %MD*n*.18, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.0, where *n* = 24 + the axis number Secondary Input: %MD*n*.18, where *n* = 24 + the axis number **RMC200:** Primary Input: %MD*n*.20, where *n* = 384 + the axis number Secondary Input: %MD*n*.80, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].AccScale, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	$(pu/s^2)/V$ or $(pu/s^2)/mA$
Range:	<> 0
Default Value:	1

Description

This parameter is used together with the Acceleration Offset parameter to convert the Voltage or Current of a single-input acceleration to an Actual Acceleration. The Acceleration Scale specifies how many position units equal one transducer voltage or current unit.

The Acceleration Scale and Acceleration Offset parameters apply only to acceleration inputs. They are not used to calculate the derived Actual Acceleration on position or velocity inputs.

To reverse the direction of the feedback, use a negative Acceleration Scale.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Setting the Acceleration Scale

In general, to set the scale, determine how many voltage or current units correspond to one acceleration unit (inches/sec^2, meters/sec^2, etc., as desired). The Acceleration Scale is the inverse of that number.

See the Analog Acceleration Scaling topic for more details.

Why bother?

It is important to specify the measurement units to be used for each axes. Setting the Scale converts the information from the transducer into meaningful measurement units.

See Also

Parameter Registers | Acceleration Offset | Channel A, B Acceleration Scale

9.2.3.2.7. Acceleration Offset

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.1, where *n* = 12 + the axis number Secondary Input: %MD*n*.19, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.1, where *n* = 24 + the axis number Secondary Input: %MD*n*.19, where *n* = 24 + the axis number **RMC200:** Primary Input: %MD*n*.21, where *n* = 384 + the axis number Secondary Input: %MD*n*.81, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].AccOffset, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Feedback Axes Parameters Pane, Setup tab: Secondary Feedback
Data Type:	REAL
Units:	V or mA
Range:	any
Default Value:	0

Description

This parameter is used on acceleration inputs together with the Acceleration Scale parameter to convert the Voltage or Current of a single-input acceleration to an Actual Acceleration. This parameter is used to shift the input voltage or current before they are the volts scaled to acceleration units.

The Acceleration Scale and Acceleration Offset parameters apply only to acceleration inputs. They are not used to calculate the derived Actual Acceleration on position or velocity inputs.

Actual Acceleration = Acceleration Scale x ((Voltage or Current) + Acceleration Offset)

See the Analog Acceleration Scaling topic for details on setting the Acceleration Offset.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Acceleration Scale | Channel A, B Acceleration Offset

9.2.3.2.8. Channel A, B Acceleration Scale

Type:	Axis Parameter Register
RMC75 Address:	**Channel A Acceleration Scale:** Primary Input: %MD*n*.0, where *n* = 12 + the axis number Secondary Input: %MD*n*.18, where *n* = 12 + the axis number **Channel B Acceleration Scale:** Primary Input: %MD*n*.2, where *n* = 12 + the axis number Secondary Input: %MD*n*.20, where *n* = 12 + the axis number
RMC150 Address:	**Channel A Acceleration Scale:** Primary Input: %MD*n*.0, where *n* = 24 + the axis number Secondary Input: %MD*n*.18, where *n* = 24 + the axis number **Channel B Acceleration Scale:** Primary Input: %MD*n*.2, where *n* = 24 + the axis number Secondary Input: %MD*n*.20, where *n* = 24 + the axis number
RMC200 Address:	**Channel A Acceleration Scale:** Primary Input: %MD*n*.20, where *n* = 384 + the axis number Secondary Input: %MD*n*.82, where *n* = 384 + the axis number **Channel B Acceleration Scale:** Primary Input: %MD*n*.22, where *n* = 384 + the axis number Secondary Input: %MD*n*.82, where *n* = 384 + the axis number
System Tag:	**Channel A Acceleration Scale:** _Axis[*n*].AccAScale **Channel B Acceleration Scale:** _Axis[*n*].AccBScale where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL

Units:	$(pu/s^2)/V$ or $(pu/s^2)/mA$
Range:	<> 0
Default Value:	1

Description

These parameters are used together with the Channel A Acceleration Offset and Channel B Acceleration Offset parameters to calculate the Actual Acceleration from inputs 0 and 1 of a dual-input (differential) acceleration feedback.

After the Channel A Acceleration Offset is added to the Channel A voltage or current, it is multiplied by the Acceleration A Scale to produce the Channel A Acceleration.

After the Channel B Acceleration Offset is added to the Channel B voltage or current, it is multiplied by the Acceleration B Scale to produce the Channel B Acceleration.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Channel A, B Acceleration Offset | Acceleration Scale

9.2.3.2.9. Channel A, B Acceleration Offset

Type:	Axis Parameter Register
RMC75 Address:	**Channel A Acceleration Offset:** Primary Input: %MD*n*.1, where *n* = 12 + the axis number Secondary Input: %MD*n*.19, where *n* = 12 + the axis number **Channel B Acceleration Offset:** Primary Input: %MD*n*.3, where *n* = 12 + the axis number Secondary Input: %MD*n*.21, where *n* = 12 + the axis number
RMC150 Address:	**Channel A Acceleration Offset:** Primary Input: %MD*n*.1, where *n* = 24 + the axis number Secondary Input: %MD*n*.19, where *n* = 24 + the axis number **Channel B Acceleration Offset:** Primary Input: %MD*n*.3, where *n* = 24 + the axis number Secondary Input: %MD*n*.21, where *n* = 24 + the axis number
RMC200 Address:	**Channel A Acceleration Offset:** Primary Input: %MD*n*.21, where *n* = 384 + the axis number Secondary Input: %MD*n*.81, where *n* = 384 + the axis number **Channel B Acceleration Offset:** Primary Input: %MD*n*.23, where *n* = 384 + the axis number Secondary Input: %MD*n*.83, where *n* = 384 + the axis number
System Tag:	**Channel A Acceleration Offset:** _Axis[*n*].AccAOffset **Channel B Acceleration Offset:** _Axis[*n*].AccBOffset where *n* is the axis number

How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL
Units:	V, mA
Range:	any
Default Value:	0

Description

These parameters are used together with the Channel A Acceleration Scale and Channel B Acceleration Offset parameters to calculate the Actual Acceleration from inputs 0 and 1 of a dual-input (differential) acceleration axis. For each channel, the Acceleration Offset shifts the voltage or current input as shown below:

Actual Acceleration = Acceleration Scale x ((Voltage or Current) + Acceleration Offset)

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Channel A, B Acceleration Scale | Acceleration Offset

9.2.3.2.10. Position Unwind

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.12, where *n* = 12 + the axis number **RMC150:** %MD*n*.12, where *n* = 24 + the axis number **RMC200:** %MD*n*.28, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].PosUnwind, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	pu
Range:	not equal to zero
Default Value:	1000

Description

This parameter is used on rotary position axes together with the Count Unwind, Position Offset and Count Offset parameters to convert the transducer feedback Counts to an Actual Position.

The Position Unwind parameter defines the valid range of positions for the axis. When the position units reach the end of the valid range, they will wrap around to the beginning of the range. For example, consider an axis with a Position Unwind of 360. If the axis rotates continuously, the position will go from 0 up to, but not including, 360. As it reaches 360, it will "wrap" to 0 and keep going.

For absolute rotary axes, the Counts are in the opposite direction if the Position Unwind is negative.

The Counts formula for a positive Position Unwind is:

Counts = (RawCounts + CountOffset) MOD MaxCounts

The Counts formula for a positive Negative Unwind is:

Counts = ([MaxCounts-1] - RawCounts + CountOffset) MOD MaxCounts

The Position Offset defines the value at which the position units begin.

Examples:

Position Unwind	Position Offset	Position Range
360	0	0 to 360
360	-180	-180 to 180
360	60	60 to 420

Absolute Rotary axes also use the Count Offset parameter to move the zero point of the Counts. The position range begins where the Counts are zero.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Rotary Scaling

9.2.3.2.11. Count Unwind

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.13, where *n* = 12 + the axis number **RMC150:** %MD*n*.13, where *n* = 24 + the axis number **RMC200:** %MD*n*.29, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].CntUnwind, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	DINT
Units:	pu
Range:	> 0
Default Value:	1024

Description

This parameter is used on rotary position axes together with the Position Unwind, Position Offset and Count Offset parameters to convert the transducer feedback Counts to an Actual Position.

The Count Unwind parameter defines the range of feedback Counts that maps to the Position Unwind. As the encoder rotates through the range of counts de fined by the Count Unwind, the Actual Position will go through the entire range of numbers specified by the Position Unwind.

For axes configured as Rotary Absolute, the Count Unwind parameter must be a power of 2, such as 1024, 8192, etc.

Example:
Consider an SSI Rotary Absolute encoder with 8192 counts per revolution. The user wants to set it up so that it goes from 0 to 360 on one revolution. To do this, the Count Unwind must be set to 8192, and the Position Unwind must be set to 360.

Valid Range

The valid range of the Count Unwind parameter is limited for certain feedback types.

- **SSI Feedback on Absolute Rotary Axes**
  The **Count Unwind** parameter must be less than or equal to 2^N, where N is the SSI Data Bits. For example, for a 13-bit SSI encoder, the SSI Data Bits would be set to 13, and the maximum Count Unwind value would be 2^{13}, which is 8192.
- **Resolver Feedback on Absolute Rotary Axes**
  The maximum value of the Count Unwind parameter for resolver feedback is 65536.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Rotary Scaling

9.2.3.2.12. Count Offset

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.11, where *n* = 12 + the axis number **RMC150:** %MD*n*.11, where *n* = 24 + the axis number **RMC200:** %MD*n*.30, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].CntOffset, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	DINT
Units:	pu
Range:	any
Default Value:	0

Description

This parameter is used on SSI and Resolver position axes with absolute positions. The Count Offset offsets the transducer Counts before the Counts are converted to position units. This parameter has no effect if the axis is set to incremental.

The Count Offset fits in the conversion of counts to an actual position as follows:

Rotary Axes:

Actual Position [pu] = ((Counts [cnt] + **Count Offset** [cnt]) x Position Unwind [pu] / Count Unwind [pu/cnt]) + Position Offset [pu]

Linear Axes:

Actual Position [pu] = ((Counts [cnt] + **Count Offset** [cnt]) x Position Scale [pu/cnt]) + Position Offset [pu]

The Count Offset is typically used in the following situations:

- On rotary axes with absolute encoders, to set the zero point of the axis.
- On linear axes that use an absolute encoder, the Count Offset is needed to insure that the total counts (Counts + Count Offset) never cross the roll-over point (below zero or above max count).

Example: Rotary Axis with Rotary Absolute Encoder
Consider an SSI rotary absolute encoder with 8192 counts per revolution, with the Count Unwind to 8192, and the Position Unwind to 360. Therefore, the positions will go from 0 up to, but not including, 360. You want the zero point (zero counts) to be at the top of the rotation so it will read 0 degrees, but perhaps the way the transducer is mounted gives you 1000 counts at the top. You can set the Count Offset to -1000, which will give you zero counts at the top, and therefore, 0 degrees.

Example: Linear Axis with Rotary Absolute Encoder
Consider a rack and pinion system with 10000 mm of travel and an encoder mounted on the pinion. 1 turn on the encoder will move 100 mm. The entire travel will therefore span 100 turns. The encoder is an SSI multi-turn rotary absolute encoder with 4096 counts per revolution and 4096 turns.

The maximum counts is 16,777,216.

The entire span of counts over the travel range will be 100 turns x 4096 counts = 409,600 counts.

The Position Scale is 100 mm / 4096 counts = 0.024414 mm/cnt

When setting up the axis, the user started with the pinion all the way to the negative end of the travel. At that point, the encoder counts happened to be 16,561,242. Moving to the positive end of travel would add 409,600 counts resulting in a value of 16,970,842, which is greater than the maximum 16,777,216 counts of the encoder. This will cause the encoder to pass the zero point, which is unacceptable on a linear axis.

The Count Offset can be used to adjust the zero point such that the axis will not reach it. For example, setting the Count Offset to -1,000,000 will change the counts at the negative end of travel to 15,561,242. The value at the positive end will then be 15,561,242 + 409,600 = 15,970,842, which is less than the maximum of 16,777,216, and the axis will not reach the zero point.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Scaling Overview

9.2.3.2.13. Linear/Rotary

Type:	Axis Parameter Register Bit Parameter
Address:	**RMC75:** %MD*n*.9.0, where *n* = 12 + the axis number

	RMC150: %MD*n*.9.0, where *n* = 24 + the axis number
	RMC200: %MD*n*.9.0, where *n* = 24 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].PriInputBits.Rotary
	RMC200: _Axis[*n*].Rotary
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	**RMC75/150:** bit
	RMC200: DINT
Range:	Linear (0), Rotary (1)
Default Value:	Linear (0)

Description

The Linear/Rotary Bit Parameter specifies whether the axis is set as a Rotary axis or a Linear axis. Each axis's input is either linear or rotary. When this bit is set, the axis is rotary. When this bit is cleared, the axis is linear.

To determine whether you should set your axis as Rotary, see the Rotary Axis topic.

See the Primary Input Bits Register for details about the register containing this bit.

Note: For rotary axes, the encoder counts per revolution must be a power of 2.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers

9.2.3.2.14. Stop Threshold

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.5, where *n* = 12 + the axis number
	RMC150: %MD*n*.5, where *n* = 24 + the axis number
	RMC200: %MD*n*.31, where *n* = 24 + the axis number
System Tag:	_Axis[*n*].StopThreshold, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	REAL
Units:	pu/s
Range:	≥ 0
Default Value:	0.1

Description

The Stop Threshold specifies a velocity threshold for considering an axis stopped. When the absolute value of the velocity falls below this threshold, the axis is considered stopped, and the Stopped Status bit will be set. Notice that axis may still be moving slightly, because the criterium

is a velocity threshold. The Stopped bit is not latched and will clear when the velocity exceeds the threshold.

The Stop Threshold does not apply to axes with only pressure or force control because the axis may be physically moving even though the pressure does not change. The stopped status of axes with position-pressure or velocity-pressure control is determined by the position or velocity axis.

Why Bother?

If an axis is considered stopped when the velocity is exactly zero, it would be impossible for it to ever be considered stopped, since there is always noise in real-life systems. For example, perhaps a system (scaled in inches) has velocities up to 0.08 in/sec when it's standing still. Setting the Stop Threshold to 0.1 in/sec will let you know when the axis is approximately stopped.

See Also

Parameter Registers | Stopped Status bit

9.2.3.2.15. Noise Error Rate

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.6, where *n* = 12 + the axis number Secondary Input: %MD*n*.24, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.6, where *n* = 24 + the axis number Secondary Input: %MD*n*.24, where *n* = 24 + the axis number **RMC200:** Primary Input: %MD*n*.32, where *n* = 24 + the axis number Secondary Input: %MD*n*.92, where *n* = 24 + the axis number
System Tag:	**Primary Input:** _Axis[*n*].NoiseErrorRate **Secondary Input:** _Axis[*n*].SecNoiseErrorRate where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	REAL
Units:	See Below
Range:	≥ 0
Default Value:	0 A value of zero means the Noise Error detection is disabled.

Description

This parameter is designed to specify the noise detection on the feedback. It is primarily designed for MDT inputs, but also works well for analog voltage or current inputs. If a large change in the feedback is detected, the Noise Error bit will be set.

The Noise Error Rate defines the maximum allowable rate of change in the feedback of the axis. The behavior depends on the feedback type:

- **Position Inputs**
  If the rate of change of position exceeds the **Noise Error Rate** for more than 3 loop times, then the Noise Error bit will be set. While the input is estimated, the Input Estimated status bit

will be set. When the Noise Error bit becomes set, it may cause a Halt if so configured by the Auto Stops and if the Direct Output Status bit is off. During noise less than 3 loop times, the Actual Position (or the value measured by the axis' input) is estimated to allow recovery from electrical noise.

- **Velocity, Pressure, Force Inputs**
  If the rate of change of position exceeds the **Noise Error Rate**, the Noise Error bit will be set immediately.
- **Quadrature Inputs**
  The Noise Error Rate parameter does not apply to quadrature axes. Instead, if the RMC detects an illegal transition (both A and B signals transition simultaneously), or an overspeed condition (pulse frequency exceeds maximum specifications), the Noise Error bit will be set immediately.

Make sure to set the Noise Error Rate to a value much higher than the maximum expected rate of change in the axis' units. For example, if the axis will be traveling at up to 200 pu/s, then set the Noise Error Rate to something much higher, such as 800.

A value of zero means the Noise Error detection is disabled. For RMC75/150 controllers prior to 3.64.0 firmware, zero was not supported. To disable the feature for these controllers, set the Noise Error Rate to a very large value, such as 1e10.

Note:
The Noise Error bit may turn on because the Noise Error Rate parameter is set too low. The Noise Error Rate should be set to a value much higher than the expected velocity (or rate of change of pressure or force) of the axis. If your axis is experiencing Noise Errors, this is the first item you should check. Setting the Noise Error Auto Stop to Status Only to avoid halting due to an overly low Noise Error Rate parameter can cause significant control problems, because the Actual Position and Actual Velocity may still be incorrectly calculated.

Units

The Noise Error Rate applies to all axes types. However, the Noise Error Rate units for each axis type may differ. The following table shows the Noise Error Rate units for each axis type:

Axis Type	Noise Error Rate Units
Position	pu/s
Velocity	pu/s^2
Acceleration	pu/s^3
Pressure/Force	Pr/s or Fr/s

Special Notes

The Noise Error Rate does factor out the current actual rate of change of the axis, and therefore, it won't trigger for simply going over speed, although in this situation, a high acceleration can trigger the Noise Error. In any case, the user should set the Noise Error Rate much higher than the expected velocity of the axis.

See Also

Parameter Registers | Noise Error bit

9.2.3.2.16. Display Units

Type: Axis Parameter Register

Address:	**RMC75:** Primary Input: %MD*n*.166, where *n* = 12 + the axis number Secondary Input: %MD*n*.168, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.166, where *n* = 24 + the axis number Secondary Input: %MD*n*.168, where *n* = 24 + the axis number **RMC200:** Primary Input: %MD*n*.35, where *n* = 384 + the axis number Secondary Input: %MD*n*.95, where *n* = 384 + the axis number
System Tag:	Primary Input: _Axis[*n*].PriUnits Secondary Input: _Axis[*n*].SecUnits, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup or Secondary Control Setup
Data Type:	DINT
Range:	See **Values** below
Default Value:	0

Description

The Display Units axis parameter defines the feedback units that are displayed for that axis in RMCTools. The Display Units are solely for display purposes and do not provide any automatic feedback scaling or conversion between units.

To set the Display Units for an axis:

1. In the Axis Tools, in the Axis Parameters, go to the **Setup** tab.
2. In the **Primary Control Setup** or **Secondary Control Setup** sections, find the **Display Units** parameter.
3. Choose the desired units from the list, or choose Custom. If you choose Custom, you can define your own units by entering up to four characters in the Custom Units axis parameter.

Example 1: On a position feedback axis, the user selects **mm**. The position units will be displayed as (mm).

Example 2: On a position-force feedback axis, the user selects **in** for the primary feedback and **lbs** for the secondary feedback. The position units will be displayed as (in) and the force units will be displayed as (lbs).

Example 3: On a position axis, the user has connected a torque sensor. In the Display Units parameter, the user selects **Custom**. In the Custom Units parameter, the user enters "Nm". The axis units will be displayed as (Nm).

Values

The table below lists the display units for each value of this register.

Value	Position Feedback	Velocity Feedback	Acceleration Feedback	Pressure Feedback	Force Feedback
-1	Custom	Custom	Custom	Custom	Custom
0	pu (default)	pu/s (default)	pu/s^2 (default)	Pr (default)	Fr (default)
1	mm	mm/s	mm/s^2	psi	lb
2	cm	cm/s		bar	kg
3	m	m/s	m/s^2	atm	N

4	in	in/s	in/s²	kPa	ton (short, US)
5	ft	ft/s	ft/s²	MPa	ton (long, UK)
6	deg	deg/s			ton (metric)
7	rev	rps			kN
8	rpm·s	rpm			
9	fpm·s				
10	thou				

*The text in parentheses will not be displayed where the units are used, although the Display Units register itself will display the entire units string.

See Also

Custom Units | Pressure Display Units

9.2.3.2.17. Custom Units

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.167, where *n* = 12 + the axis number Secondary Input: %MD*n*.169, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.167, where *n* = 24 + the axis number Secondary Input: %MD*n*.169, where *n* = 24 + the axis number **RMC200:** No address. The custom units are stored in non-user-addressable memory space.
System Tag:	None
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup *or* Secondary Control Setup
Data Type:	RMC75/150: DWORD RMC200: STRING

Description

This axis parameter is used to define custom units for the feedback of the axis. This parameter is used only if the Display Units parameter is set to Custom. If Custom is selected, but no characters are entered in the Custom Units parameter, the axis will use the default units.

In the Custom Units parameter, you may enter up to four valid characters as listed below.

RMC75/150:

In the Custom Units parameter, you may enter up to four valid characters as listed below.

!"#$%&'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopq
rstuvwxyz

{|}~ ¡¢£¤¥¦§¨©ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿
ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæçèéêëìíîïðñòóôõö÷øùúûüýþÿ

These characters correspond to the Unicode character ranges U+0021 to U+007E and U+00A0 to U+00FF. The NUL character (U+0000) cannot be entered, but RMCTools will pad trailing unused characters with NUL. The space character (U+0020) may not be used for the first or last character. For details on Unicode characters, see www.unicode.org.

RMC200:

In the Custom Units parameter, you may enter up to four Unicode characters.

Register Format

RMC75/150:

The Custom Units parameter is a 32-bit register of data type DWORD. Each byte (8 bits) encodes one character. If less than 4 characters are entered, the unused low bytes will be padded with the NUL character.

Example

The user entered "in²". This string is stored in the register as follows:

Bits	31-24	23-16	15-8	7-0
Character	i	n	²	
Unicode Value	U+0069	U+006E	U+00B2	U+0000
Hexadecimal Value	16#69	16#6E	16#B2	16#00
Complete 32-Bit Hexadecimal Value	16#696EB200			

RMC200:

The Custom Units parameter is not accessible as a register and therefore does not have a register format.

See Also

Display Units | Pressure Custom Units | Pressure Display Units

9.2.3.2.18. Primary Input Bits Register

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.9, where *n* = 12 + the axis number **RMC150:** %MD*n*.9, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].PriInputBits, where *n* is the axis number
How to Find:	See individual parameters listed below
Data Type:	DWORD - see below

Description

The RMC75/150 Primary Input Bits register contains the bit-addressable Feedback configuration parameters. This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

The corresponding tags in the RMC200 are stored in the Limit Input Configuration, Filter Configuration and Linear/Rotary registers.

Tag Names

This register contains the following parameters. The tag names and bits for each are given below.

RMC75 **RMC150**

Parameter	Tag	Bit(s	Parameter	Tag	Bit(
Rotary vs. Linear	Rotary	0	Rotary vs. Linear	Rotary	0
Positive Limit Input	PosLimitIn	4-7	Limit Input Polarity	LimitInPol	1
Negative Limit Input	NegLimitIn	8-11	Velocity Filter Type	VelFilterType	4-5
Limit Input Polarity	LimitInPol	12	Acceleration Filter Type	AccelFilterType	6-7
Velocity Filter Type	VelFilterType	13-1	Positive Limit Input	PosLimitIn	8-15
Acceleration Filter Type	AccelFilterType	15-1	Negative Limit Input	NegLimitIn	16-2

For details on the values that each bit represents, see the respective parameter topic.

See Also

Parameter Registers

9.2.3.2.19. Filtering - RMC70/150

9.2.3.2.19.1. Actual Position Filter (RMC75/150)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.2, where *n* = 12 + the axis number **RMC150:** %MD*n*.2, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].ActPosFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** See **Limits** below
Default Value:	0

Description

This parameter specifies the cut-off frequency of the Actual Position input filter in hertz. The filter is applied to the Actual Position before it is used in the control algorithm. The filter is a low-pass fourth order Butterworth filter. Setting this value to zero (0) disables the Actual Position filter.

Filtering of the Actual Position is not recommended on position control axes. The control algorithms use the filtered Actual Position. If filtering is enabled, it will slow the response of the system. The lower the cut-off frequency, the slower the system will respond.

Notice that this Actual Position filter is independent of the Actual Velocity Filter and Actual Acceleration Filter. If the Actual Velocity and Actual Acceleration filters are being used on the controlled values (Velocity Filter Type or Acceleration Filter Type are set to Low-Pass or Model), then you should make sure to always set the Actual Position filter, Actual Velocity Filter, and Actual Acceleration filter to the same values.

Limits

The filter is limited to greater than 0.01 due to inaccuracies in the calculations for lower values. The filter is disabled (without notice to the user) when the frequency is set above 1/4 the sample frequency (500Hz for 500us loop, 250Hz for 1ms loop, 125Hz for 2ms loop, 62.5Hz for 4ms loop).

Why Bother?

Filtering the input can reduce noise in the feedback. It also makes the plots look cleaner. Filtering of the position should not be done on control axes.

See Also

Parameter Registers | Filtering Overview

9.2.3.2.19.2. Actual Velocity Filter (RMC75/150)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.3, where *n* = 12 + the axis number **RMC150:** %MD*n*.3, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].ActVelFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** See **Limits** below
Default Value:	100

Description

This parameter specifies the cut-off frequency of the Actual Velocity input filter in hertz. The filter is applied to the Actual Velocity before it is used in the control algorithm. The filter is a low-pass fourth order Butterworth filter.

For most position or velocity control applications, there is no need to change this parameter from its default setting. If you are using the velocity of a reference input, you may wish to decrease this value if the velocity is noisy.

Setting the **Actual Velocity Filter** to zero (0) disables the filter. If the Velocity Filter Type parameter is set to **Model**, the Actual Velocity Filter will not be used (see below).

How the RMC75/150 Use the Filtered Velocity

The Velocity Filter parameter specifies how the **Actual Velocity Filter** is used in the RMC:

- **Status Only (default setting)**
  The **Actual Velocity Filter** is applied to the Velocity value used for status and for plots, but it is not applied to the value used for control. This is the default setting because the filtering introduces phase that negatively affects control. This option should be selected for systems that need to react to quick changes.
- **Low-Pass**
  The **Actual Velocity Filter** applies to both the Actual Velocity status register and the velcoity value used by the control algorithm. If you want the filter to apply to the control, choose this option.

- **Model**
  The **Actual Velocity Filter** is not used. Instead, the Model is used to calculate the Actual Velocity. This value is then used for both the Actual Velocity status register and the control algorithm.

Filtering increases the phase delay in the filtered value. By default, filtering is applied to the Velocity and Acceleration values used for status and for plots, but it is not applied to the values used for control because of the phase delay filtering introduces. Therefore, filtering should be left out of systems that need to react to quick changes. You can select to use filtered values for control with the Velocity Filter Type and Acceleration Filter Type parameters.

Notice that this Actual Velocity filter is independent of the Actual Position Filter and Actual Acceleration Filter. If the Actual Velocity and Actual Acceleration filters are being used on the controlled values (Velocity Filter Type or Acceleration Filter Type are set to Low-Pass or Model), then you should make sure to always set the Actual Position filter, Actual Velocity Filter, and Actual Acceleration filter to the same values.

Limits

The filter is limited to greater than 0.01 due to inaccuracies in the calculations for lower values. The filter is disabled (without notice to the user) when the frequency is set above 1/4 the sample frequency (500Hz for 500us loop, 250Hz for 1ms loop, 125Hz for 2ms loop, 62.5Hz for 4ms loop).

Why Bother?

- Filtering makes the plots look cleaner.
- Filtering can be used to "smooth" the velocity of a reference input.
- Filtering the input can reduce noise in the feedback which may improve system control.

See Also

Velocity Filter | Parameter Registers | Filtering | Velocity Filter Type

9.2.3.2.19.3. Actual Acceleration Filter (RMC75/150)

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.4, where *n* = 12 + the axis number Secondary Input: %MD*n*.22, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.4, where *n* = 24 + the axis number Secondary Input: %MD*n*.22, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	Position/Velocity Axes: _Axis[*n*].ActAccFilter Primary Accel Input: _Axis[*n*].AccFilter Secondary Accel Input: _Axis[*n*].SecAccFilter where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01

	Maximum: See **Limits** below
Default Value:	25

Description

This parameter specifies the cut-off frequency of the Actual Acceleration input filter in hertz. The filter is a low-pass fourth order Butterworth filter. For most position or velocity control applications, there is no need to change this parameter from its default setting.

Setting the Actual Acceleration Filter to zero (0) disables the filter.

How the RMC75/150 Use the Filtered Acceleration

Acceleration Inputs

The Actual Acceleration will be filtered as specified by the Actual Acceleration Filter parameter. The Acceleration Filter Type parameter does not apply.

Position or Velocity Input Axes

If the Acceleration Filter Type parameter is set to **Model**, the Actual Acceleration Filter will not be used (see below).

The Acceleration Filter Type parameter specifies how the **Actual Acceleration Filter** is used in the RMC:

- **Status Only (default setting)**
  The **Actual Acceleration Filter** is applied to the Acceleration value used for status and for plots, but it is not applied to the value used for control. This is the default setting because the filtering introduces phase that negatively affects control. This option should be selected for systems that need to react to quick changes.
- **Low-Pass**
  The **Actual Acceleration Filter** applies to both the Actual Acceleration status register and the acceleration value used by the control algorithm. If you want the filter to apply to the control, choose this option.
- **Model**
  The **Actual Acceleration Filter** is not used. Instead, the Model is used to calculate the Actual Acceleration. This value is then used for both the Actual Acceleration status register and the control algorithm.

Filtering increases the phase delay in the filtered value. By default, filtering is applied to the Velocity and Acceleration values used for status and for plots, but it is not applied to the values used for control because of the phase delay filtering introduces. Therefore, filtering should be left out of systems that need to react to quick changes. You can select to use filtered values for control with the Velocity Filter Type and Acceleration Filter Type parameters.

Notice that this Actual Acceleration filter is independent of the Actual Position Filter and Actual Velocity Filter. If the Actual Velocity and Actual Acceleration filters are being used on the controlled values (Velocity Filter Type or Acceleration Filter Type are set to Low-Pass or Model), then you should make sure to always set the Actual Position filter, Actual Velocity Filter, and Actual Acceleration filter to the same values.

Limits

The filter is limited to greater than 0.01 due to inaccuracies in the calculations for lower values. The filter is disabled (without notice to the user) when the frequency is set above 1/4 the sample frequency (500Hz for 500us loop, 250Hz for 1ms loop, 125Hz for 2ms loop, 62.5Hz for 4ms loop).

Why Bother?

- Filtering makes the plots look cleaner.
- Filtering can be used to "smooth" a reference input.
- Filtering the input can reduce noise in the feedback which may improve system control.

See Also

Parameter Registers | Filtering Overview | Acceleration Filter Type

9.2.3.2.19.4. Actual Jerk Filter (RMC75/150)

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.5, where *n* = 12 + the axis number Secondary Input: %MD*n*.23, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.5, where *n* = 24 + the axis number Secondary Input: %MD*n*.23, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	Primary Input: _Axis[*n*].JerkFilter Secondary Input: _Axis[*n*].SecJerkFilter where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** See **Limits** below
Default Value:	25

Description

This parameter specifies the cut-off frequency of the Actual Jerk input filter in hertz. The filter is a low-pass fourth order Butterworth filter. This parameter applies only to acceleration inputs. Setting the **Actual Jerk Filter** to zero (0) disables the filter.

This filter will apply to both the Actual Jerk status register *and* the jerk value used by the higher-order Active Damping and Acceleration Control algorithms.

Limits

The filter frequency range is limited to greater than 0.01 due to inaccuracies in the calculations for lower values. The filter is disabled (without notice to the user) when the frequency is set above 1/4 the sample frequency (500Hz for 500us loop, 250Hz for 1ms loop, 125Hz for 2ms loop, 62.5Hz for 4ms loop).

Why Bother?

- Filtering makes the plots look cleaner.
- Filtering can be used to "smooth" a reference input.
- Filtering the input can reduce noise in the feedback which may improve system control.

See Also

Parameter Registers | Filtering Overview

9.2.3.2.19.5. Velocity Filter Type (RMC75/150)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.9.13-14, where *n* = 12 + the axis number **RMC150:** %MD*n*.9.4-5, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].PriInputBits.VelFilterType, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling
Data Type:	Bits
Range:	Status Only (0), Low-Pass (1), Model (2)
Default Value:	Status Only (0)

Description

This parameter specifies how the Actual Velocity Filter is used by the RMC75:

- **Status Only (default setting)**
  The Actual Velocity Filter applies only to the Actual Velocity status register. The control algorithm uses the unfiltered velocity.
- **Low-Pass**
  The Actual Velocity Filter applies to both the Actual Velocity status register and the velocity value used by the control algorithm. If you want the filter to apply to the control, choose this option.
- **Model**
  The Actual Velocity Filter is not used. Instead, the Model is used to calculate the Actual Velocity. This value is then used for both the Actual Velocity status register and the control algorithm. This option is only available if the Model Order parameter is defined.
  Notice that the model will be used only if a valid model can be calculated. See the modeling topic for details.

Format Details

This section is primarily for addressing the Velocity Filter Type parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

RMC75

Bit 14	Bit 13	Value	Filter Type
0	0	0	Status Only
0	1	1	Low-Pass
1	0	2	Model

RMC150

Bit 5	Bit 4	Value	Filter Type
0	0	0	Status Only
0	1	1	Low-Pass
1	0	2	Model

See the Primary Input Bits Register for details about the register containing these bits.

See Also

Actual Velocity Filter | Filtering Overview | Velocity Control

9.2.3.2.19.6. Acceleration Filter Type (RMC75/150)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.9/15-16, where *n* = 12 + the axis number **RMC150:** %MD*n*.9/6-7 , where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].PriInputBits.AccelFilterType, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling
Data Type:	Bits
Range:	Status Only (0), Low-Pass (1), Model (2)
Default Value:	Status Only (0)

Description

This parameter specifies how the Actual Acceleration Filter is used by the RMC:

- **Status Only (default setting)**
  The Actual Acceleration Filter applies only to the Actual Acceleration status register. The control algorithm uses the unfiltered acceleration.
- **Low-Pass**
  The Actual Acceleration Filter applies to both the Actual Acceleration status register and the acceleration value used by the control algorithm. If you want the filter to apply to the control, choose this option.
- **Model**
  The Actual Acceleration Filter is not used. Instead, the Model is used to calculate the Actual Acceleration. This value is then used for both the Actual Acceleration status register and the control algorithm. This option is only available if the Model Order parameter is defined. Notice that the model will be used only if a valid model can be calculated. See the modeling topic for details.

Format Details

This section is primarily for addressing the Acceleration Filter Type parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

RMC75

Bit 16	Bit 15	Value	Filter Type
0	0	0	Status Only
0	1	1	Low-Pass
1	0	2	Model

RMC150

Bit 7	Bit 6	Value	Filter Type
0	0	0	Status Only
0	1	1	Low-Pass
1	0	2	Model

See the Primary Input Bits Register for details about the register containing these bits.

See Also

Actual Acceleration Filter | Parameter Registers | Filtering Overview

9.2.3.2.19.7. Actual Pressure/Force Filter (RMC75/150)

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.4, where *n* = 12 + the axis number Secondary Input: %MD*n*.22, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.4, where *n* = 24 + the axis number Secondary Input: %MD*n*.22, where *n* = 24 + the axis number
System Tag:	Pressure Input: _Axis[*n*].PrsFilter Force Input: _Axis[*n*].FrcFilter where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** See **Limits** below
Default Value:	0

Description

This parameter specifies the cut-off frequency of the Actual Pressure/Force input filter in hertz. The filter is applied to the Actual Pressure/Force before it is used in the control algorithm. Therefore, changes to this filter value will affect control, and may require retuning the axis. The filter is a low-pass fourth order Butterworth filter.

Setting this value to zero (0) disables the filter.

Notice that this Actual Pressure/Force filter is independent of the Actual Pressure/Force Rate Filter. The Actual Pressure/Force Rate Filter does not affect control and only affects the displayed Actual Force Rate.

Limits

The filter is limited to greater than 0.01 due to inaccuracies in the calculations for lower values. The filter is disabled (without notice to the user) when the frequency is set above 1/4 the sample frequency (500Hz for 500us loop, 250Hz for 1ms loop, 125Hz for 2ms loop, 62.5Hz for 4ms loop).

Why Bother?

Filtering the pressure/force input can reduce noise in the feedback which may improve system control. It also makes the plots look cleaner.

See Also

Parameter Registers | Filtering Overview

9.2.3.2.19.8. Actual Pressure/Force Rate Filter (RMC75/150)

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.5, where *n* = 12 + the axis number Secondary Input: %MD*n*.23, where *n* = 12 + the axis number

	RMC150: Primary Input: %MD*n*.5, where *n* = 24 + the axis number Secondary Input: %MD*n*.23, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	**Pressure Input:** _Axis[*n*].PrsRateFilter **Force Input:** _Axis[*n*].FrcRateFilter where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback or Sec Feedback → Filtering/Modeling
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** See **Limits** below
Default Value:	500

Description

This parameter specifies the cut-off frequency of the Actual Pressure/Force Rate input filter in hertz. The filter is NOT applied to the Actual Pressure/Force Rate used in the control algorithm. It is only applied to the value displayed in the Axis Tools and in the plot. Therefore, changes to the Actual Pressure/Force Rate Filter value will not affect the control or the tuning. The filter is a low-pass fourth order Butterworth filter.

Setting this value to zero (0) disables the filter.

Notice that this Actual Pressure/Force Rate filter is independent of the Actual Pressure/Force Filter. The Actual Pressure/Force Filter *does* affect control.

Limits

The filter is limited to greater than 0.01 due to inaccuracies in the calculations for lower values. The filter is disabled (without notice to the user) when the frequency is set above 1/4 the sample frequency (500Hz for 500us loop, 250Hz for 1ms loop, 125Hz for 2ms loop, 62.5Hz for 4ms loop).

Why Bother?

Filtering the pressure/force rate makes the plots look cleaner.

See Also

Parameter Registers | Filtering Overview

9.2.3.2.20. Filtering - RMC200

9.2.3.2.20.1. Filter Configuration Register (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.40, where *n* = 384+ the axis number Secondary Input: %MD*n*.100, where *n* = 384+ the axis number

System Tag:	**Primary Input:** _Axis[*n*].FilterConfig, where *n* is the axis number **Secondary Input:** _Axis[*n*].SecFilterConfig, where *n* is the axis number
How to Find:	See individual parameters listed below
Data Type:	DWORD - see below

The RMC200 Filter Configuration register contains the bit-addressable filter configuration parameters. This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Tag Names

This register contains the following parameters. The tag names and bits for each are given below.

Parameter	Tag	Bit(s)
Input Filter Type	Type	0-3

For details on the values that each bit represents, see the respective parameter topic.

See Also

Filtering | Filter Algorithm Types

9.2.3.2.20.2. Input Filter Type (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.40.0-3, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].FilterConfig.Type, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	Bits
Default Value:	0 (none)

Description

This parameter selects the RMC200 input Filter Algorithm Type. The input filter applies to the feedback values that the control algorithm uses. The RMC200 also has a separate display filter, which further filters the values that have already been filtered by the input filter.

The input filter options are:

- **None (0)**
  No input filter is applied. Notice that a Display Filter may still be active.
- **Low Pass 1P (1)**
  Low-pass single-pole filter, with individual cut-off frequencies for the Position, Velocity, and Acceleration, or the Pressure/Force and Pressure/Force Rate.
- **Low Pass BW2 (2)**
  Low-pass two-pole Butterworth filter, with individual cut-off frequencies for the Position, Velocity, and Acceleration, or the Pressure/Force and Pressure/Force Rate.
- **Low Pass BW4 (3)**
  Low-pass four-pole Butterworth filter, with individual cut-off frequencies for the Position, Velocity, and Acceleration, or the Pressure/Force and Pressure/Force Rate.

- **Low Pass ABG (4)**
  Low-pass Alpha-Beta-Gamma filter, with a single cut-off frequency for the Position, Velocity, and Acceleration, or the Pressure/Force and Pressure/Force Rate.
- **Low Pass ABGD (5)**
  Low-pass Alpha-Beta-Gamma-Delta filter, with a single cut-off frequency for the Position, Velocity, and Acceleration, or the Pressure/Force and Pressure/Force Rate.
- **Model (10)**
  Uses the system model to calculate the Actual Velocity and Actual Acceleration. The Actual Position remains unfiltered. Only available for position feedback.

For more details, see Filter Algorithm Types.

Format Details

This section is primarily for addressing the Input Filter Type parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

Bit 3	Bit 2	Bit 1	Bit 0	Value	Filter Type
0	0	0	0	0	None
0	0	0	1	1	Low Pass 1P
0	0	1	0	2	Low Pass BW2
0	0	1	1	3	Low Pass BW4
0	1	0	0	4	Low Pass ABG
0	1	0	1	5	Low Pass ABGD
1	0	1	0	10	Model

See the Filter Configuration Register for details about the register containing these bits.

See Also

Filter Configuration Register | Modeling | Filter Algorithm Types

9.2.3.2.20.3. Position Input Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.45, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].PosInputFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Input Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** See **Limits** below
Default Value:	0

Description

The RMC200 Position Input Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Position that is used in the control algorithm. Setting this value to zero (0) disables the Position Input Filter. This filter is only available if the Input Filter Type has been set to a filter type that uses the Position Input Filter.

The Position Input Filter is part of the Input Filter, which filters the feedback values used by the control algorithm. This differs from the Position Display Filter axis parameter, which is part of the Display Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

For more details, see Filtering.

Limits

The filter is disabled (without notice to the user) when the frequency is set above the maximum frequencies listed below, or below the minimum value.

Filter Algorithm Type	Min Cut-off Frequency	Max Cut-off Frequency
Low Pass, Single Pole	0.01	50% x loop frequency
Low Pass, 2-pole Butterworth	0.01	25% x loop frequency
Low Pass, 4-pole Butterworth	0.01	25% x loop frequency
Low Pass ABG	0.01	10% x loop frequency
Low Pass ABGD	0.01	7.5% x loop frequency

See Also

Filtering | Position Display Filter

9.2.3.2.20.4. Position Display Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.41, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].ActPosFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Display Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** 25% x loop frequency
Default Value:	0

Description

The RMC200 Position Display Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Position that is displayed to the user. Setting this value to zero (0) disables the Position Display filter.

The Position Display Filter is part of the Display Filter. This differs from the Position Input Filter axis parameter, which is part of the Input Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

For more details, see Filtering.

See Also

Filtering | Position Input Filter

9.2.3.2.20.5. Velocity Input Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.46, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].VelInputFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Input Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** See **Limits** below
Default Value:	0

Description

The RMC200 Velocity Input Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Velocity that is used in the control algorithm. Setting this value to zero (0) disables the Velocity Input Filter. This filter is only available if the Input Filter Type has been set to a filter type that uses the Velocity Input Filter.

The Velocity Input Filter is part of the Input Filter, which filters the feedback values used by the control algorithm. This differs from the Velocity Display Filter axis parameter, which is part of the Display Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

This parameter does not apply to the filter algorithm types that use only a single cut-off frequency for position, velocity, and acceleration. This includes the ABG and ABGD filters.

For more details, see Filtering.

Limits

The filter is disabled (without notice to the user) when the frequency is set above the maximum frequencies listed below, or below the minimum value.

Filter Algorithm Type	Min Cut-off Frequency	Max Cut-off Frequency
Low Pass, Single Pole	0.01	50% x loop frequency

Low Pass, 2-pole Butterworth	0.01	25% x loop frequency
Low Pass, 4-pole Butterworth	0.01	25% x loop frequency

See Also

Filtering | Velocity Display Filter

9.2.3.2.20.6. Velocity Display Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.42, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].ActVelFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Display Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** 25% x loop frequency
Default Value:	100

Description

The RMC200 Velocity Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Velocity that is displayed to the user. Setting this value to zero (0) disables the Velocity Display filter.

The Velocity Display Filter is part of the Display Filter. This differs from the Velocity Input Filter axis parameter, which is part of the Input Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

For more details, see Filtering.

See Also

Filtering | Velocity Input Filter

9.2.3.2.20.7. Acceleration Input Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a

	RMC200:
	Primary Position/Velocity Input: %MD*n*.47, where *n* = 384+ the axis number
	Primary Acceleration Input: %MD*n*.45, where *n* = 384+ the axis number
	Secondary Input: %MD*n*.105, where *n* = 384+ the axis number
System Tag:	Primary Position/Velocity Input: _Axis[*n*].AccInputFilter, where *n* is the axis number
	Primary Acceleration Input: _Axis[*n*].AccInputFilter, where *n* is the axis number
	Secondary Input: _Axis[*n*].SecAccInputFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Input Filter
	Axes Parameters Pane, All tab: Secondary Feedback→Input Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off
	Minimum: 0.01
	Maximum: See **Limits** below
Default Value:	0

Description

The RMC200 Acceleration Input Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Acceleration that is used in the control algorithm. Setting this value to zero (0) disables the Acceleration Input Filter. This filter is only available if the Input Filter Type has been set to a filter type that uses the Acceleration Input Filter.

The Acceleration Input Filter is part of the Input Filter, which filters the feedback values used by the control algorithm. This differs from the Acceleration Display Filter axis parameter, which is part of the Display Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

This parameter does not apply to the filter algorithm types that use only a single cut-off frequency for position, velocity, and acceleration. This includes the ABG and ABGD filters.

For more details, see Filtering.

Limits

The filter is disabled (without notice to the user) when the frequency is set above the maximum frequencies listed below, or below the minimum value.

Filter Algorithm Type	Min Cut-off Frequency	Max Cut-off Frequency
Low Pass, Single Pole	0.01	50% x loop frequency
Low Pass, 2-pole Butterworth	0.01	25% x loop frequency
Low Pass, 4-pole Butterworth	0.01	25% x loop frequency

See Also

Filtering | Acceleration Display Filter

9.2.3.2.20.8. Acceleration Display Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Position/Velocity Input: %MD*n*.43, where *n* = 384+ the axis number Primary Acceleration Input: %MD*n*.41, where *n* = 384+ the axis number Secondary Input: %MD*n*.101, where *n* = 384+ the axis number
System Tag:	Primary Position/Velocity Input: _Axis[*n*].ActAccFilter, where *n* is the axis number Primary Acceleration Input: _Axis[*n*].AccFilter, where *n* is the axis number Secondary Input: _Axis[*n*].SecAccFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Display Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** 25% x loop frequency
Default Value:	25

Description

The RMC200 Acceleration Display Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Acceleration that is displayed to the user. Setting this value to zero (0) disables the Acceleration Display filter.

The Acceleration Display Filter applies to the Display Filter. This differs from the Acceleration Input Filter axis parameter, which is part of the Input Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

For more details, see Filtering.

See Also

Filtering | Acceleration Input Filter

9.2.3.2.20.9. Jerk Input Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a

	RMC200: Primary Input: %MD*n*.46, where *n* = 384+ the axis number Secondary Input: %MD*n*.106, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].JerkInputFilter, where *n* is the axis number Secondary Input: _Axis[*n*].SecJerkInputFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Input Filter Axes Parameters Pane, All tab: Secondary Feedback→Input Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** See **Limits** below
Default Value:	0

Description

The RMC200 Jerk Input Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Jerk that is used in the control algorithm. Setting this value to zero (0) disables the Jerk Input Filter. This filter is only available if the Input Filter Type has been set to a filter type that uses the Jerk Input Filter.

The Jerk Input Filter is part of the Input Filter, which filters the feedback values used by the control algorithm. This differs from the Jerk Display Filter axis parameter, which is part of the Display Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

For more details, see Filtering.

Limits

The filter is disabled (without notice to the user) when the frequency is set above the maximum frequencies listed below.

Filter Algorithm Type	Min Cut-off Frequency	Max Cut-off Frequency
Low Pass, Single Pole	0.01	50% x loop frequency
Low Pass, 2-pole Butterworth	0.01	25% x loop frequency
Low Pass, 4-pole Butterworth	0.01	25% x loop frequency
Low Pass ABG	0.01	10% x loop frequency
Low Pass ABGD	0.01	7.5% x loop frequency

See Also

Help Overview | Jerk Display Filter

9.2.3.2.20.10. Jerk Display Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.42, where *n* = 384+ the axis number Secondary Input: %MD*n*.102, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].JerkFilter, where *n* is the axis number Secondary Input: _Axis[*n*].SecJerkFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	pu/sec^2
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** 25% x loop frequency
Default Value:	0

Description

The RMC200 Jerk Display Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Jerk that is displayed to the user. Setting this value to zero (0) disables the Jerk Display filter.

The Jerk Display Filter is part of the Display Filter. This differs from the Jerk Input Filter axis parameter, which is part of the Input Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

For more details, see Filtering.

See Also

Filtering | Jerk Input Filter

9.2.3.2.20.11. Pressure/Force Input Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.45, where *n* = 384+ the axis number Secondary Input: %MD*n*.105, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].PrsInputFilter, where *n* is the axis number _Axis[*n*].FrcInputFilter, where *n* is the axis number

	Secondary Input: _Axis[*n*].SecPrsInputFilter, where *n* is the axis number _Axis[*n*].SecFrcInputFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Input Filter Axes Parameters Pane, All tab: Secondary Feedback→Input Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** See **Limits** below
Default Value:	0

Description

The RMC200 Pressure/Force Input Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Pressure/Force that is used in the control algorithm. Setting this value to zero (0) disables the Pressure/Force Input filter. This filter is only available if the Input Filter Type has been set to a filter type that uses the Pressure/Force Input Filter.

The Pressure/Force Input Filter is part of the Input Filter, which filters the feedback values used by the control algorithm. This differs from the Pressure/Force Display Filter axis parameter, which is part of the Display Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

For more details, see Filtering.

Limits

The filter is disabled (without notice to the user) when the frequency is set above the maximum frequencies listed below, or below the minimum value.

Filter Algorithm Type	Min Cut-off Frequency	Max Cut-off Frequency
Low Pass, Single Pole	0.01	50% x loop frequency
Low Pass, 2-pole Butterworth	0.01	25% x loop frequency
Low Pass, 4-pole Butterworth	0.01	25% x loop frequency
Low Pass ABG	0.01	10% x loop frequency
Low Pass ABGD	0.01	7.5% x loop frequency

See Also

Filtering | Pressure/Force Display Filter

9.2.3.2.20.12. Pressure/Force Rate Input Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.46, where *n* = 384+ the axis number Secondary Input: %MD*n*.106, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].PrsRateInputFilter, where *n* is the axis number _Axis[*n*].FrcRateInputFilter, where *n* is the axis number Secondary Input: _Axis[*n*].SecPrsRateInputFilter, where *n* is the axis number _Axis[*n*].SecFrcRateInputFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Input Filter Axes Parameters Pane, All tab: Secondary Feedback→Input Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** See **Limits** below
Default Value:	0

Description

The RMC200 Pressure/Force Rate Input Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Pressure/Force Rate that is used in the control algorithm. Setting this value to zero (0) disables the Pressure/Force Rate Input filter. This filter is only available if the Input Filter Type has been set to a filter type that uses the Pressure/Force Rate Input Filter.

The Pressure/Force Rate Input Filter is part of the Input Filter, which filters the feedback values used by the control algorithm. This differs from the Pressure/Force Rate Display Filter axis parameter, which is part of the Display Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

This parameter does not apply to the filter algorithm types that use only a single cut-off frequency for the pressure or force and the pressure rate or force rate. This includes the ABG and ABGD filters.

For more details, see Filtering.

Limits

The filter is disabled (without notice to the user) when the frequency is set above the maximum frequencies listed below, or below the minimum value.

Filter Algorithm Type	Min Cut-off Frequency	Max Cut-off Frequency
Low Pass, Single Pole	0.01	50% x loop frequency
Low Pass, 2-pole Butterworth	0.01	25% x loop frequency

Low Pass, 4-pole Butterworth	0.01	25% x loop frequency

See Also

Filtering | Pressure/Force Rate Display Filter

9.2.3.2.20.13. Pressure/Force Display Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.41, where *n* = 384+ the axis number Secondary Input: %MD*n*.101, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].PrsInputFilter, where *n* is the axis number _Axis[*n*].FrcInputFilter, where *n* is the axis number Secondary Input: _Axis[*n*].SecPrsInputFilter, where *n* is the axis number _Axis[*n*].SecFrcInputFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Display Filter Axes Parameters Pane, All tab: Secondary Feedback→Display Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** 25% x loop frequency
Default Value:	0

Description

The RMC200 Pressure/Force Display Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Pressure/Force that is displayed to the user. Setting this value to zero (0) disables the Pressure/Force Display Filter.

The Pressure/Force Display Filter is part of the Display Filter. This differs from the Pressure/Force Input Filter axis parameter, which is part of the Input Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

For more details, see Filtering.

See Also

Filtering | Pressure/Force Input Filter

9.2.3.2.20.14. Pressure/Force Rate Display Filter (RMC200)

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.41, where *n* = 384+ the axis number Secondary Input: %MD*n*.101, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].PrsRateFilter, where *n* is the axis number _Axis[*n*].FrcRateFilter, where *n* is the axis number Secondary Input: _Axis[*n*].SecPrsRateFilter, where *n* is the axis number _Axis[*n*].SecFrcRateFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback→Display Filter Axes Parameters Pane, All tab: Secondary Feedback→Display Filter
Data Type:	REAL
Units:	Hz
Range:	**0:** Off **Minimum:** 0.01 **Maximum:** 25% x loop frequency
Default Value:	0

Description

The RMC200 Pressure/Force Rate Display Filter parameter specifies the cut-off frequency of the filter in Hertz. This filter is applied to the Actual Pressure/Force Rate that is displayed to the user. Setting this value to zero (0) disables the Pressure/Force Rate Display filter.

The Pressure/Force Rate Display Filter is part of the Display Filter. This differs from the Pressure/Force Rate Input Filter axis parameter, which is part of the Input Filter. Feedback values are first filtered by the Input Filter, if it is enabled, and then also by the Display Filter, before being displayed as the final feedback values.

For more details, see Filtering.

See Also

Filtering | Pressure/Force Rate Input Filter

9.2.3.2.21. Modeling

9.2.3.2.21.1. Model Order

Type:	Axis Parameter Register
Address:	**RMC75:** Position: %MD*n*.149, where *n* = 12 + the axis number Pressure/Force: %MD*n*.160, where *n* = 12 + the axis number **RMC150:**

	Position: %MD*n*.149, where *n* = 24 + the axis number Pressure/Force: %MD*n*.160, where *n* = 24 + the axis number **RMC200:** Primary: %MD*n*.50, where *n* = 384 + the axis number Secondary: %MD*n*.110, where *n* = 384 + the axis number
System Tag:	**Position:** _Axis[*n*].ModOrder, where *n* is the axis number **Pressure/Force:** _Axis[*n*].PFModelOrder, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling Axes Parameters Pane, All tab: Secondary Feedback → Filtering/Modeling
Data Type:	DINT
Range:	Zero (0), First (1), Second (2), Undefined (255)
Default Value:	Undefined (255)

Description

This parameter specifies the order of the feedback model. Separate models exist for position and for pressure or force feedback. For more details, see the modeling topic.

Zero Order (0)

A zero-order system is a linear system with virtually no delay between the Control Output and motion. The velocity of the system is proportional to the Control Output.

The following parameters define the zero-order model:

Position	Pressure or Force
Model Gain Positive Model Gain Negative	Model Gain Pressure/Force

First Order (1)

A first-order system is a linear system with a delay between the Control Output and motion. At steady state, the velocity of the system is proportional to the Control Output.

The following parameters define the zero-order model:

Position	Pressure or Force
Model Gain Positive Model Gain Negative Model Time Constant	Model Gain Pressure/Force Model Time Constant

Second Order (2)

A second order system behaves like a mass on a spring.

The following parameters define the second-order model:

Position	Pressure or Force
Model Gain Positive Model Gain Negative Model Natural Frequency Model Damping Factor	Model Gain Pressure/Force Model Natural Frequency Model Damping Factor

Changing this Parameter

On position axes, always change this parameter only from Axis Tools for best results. If the position feedback model is being used, it will be suspended until the last position feedback model

parameter register has been updated (Time Constant for first order or Damping Factor for second order). At this point, the new feedback model will be calculated and the model will be activated.

Certain position model parameter settings may result in an invalid model. If this occurs, change your settings. See the modeling topic for more details.

See Also

Modeling

9.2.3.2.21.2. Model Gain Positive

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.150, where *n* = 12 + the axis number **RMC150:** %MD*n*.150, where *n* = 24 + the axis number **RMC200:** %MD*n*.51, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].ModGainPos, where *n* is the axis number
How to Find:	**RMC75/150:** Axes Parameters Pane, All tab: Feedback → Filtering/Modeling **RMC200:** Axes Parameters Pane, All tab: Feedback → Model
Data Type:	REAL
Units:	**RMC75/150:** (pu/s)/volt **RMC200:** (pu/s)/%
Default Value:	**RMC75/150:** 1 **RMC200:** 0.1

Description

Note: The units differ between the RMC75/150 and RMC200, since the RMC75/150 Control Output is voltage, and the RMC200 Control Output is percent.

This parameter specifies the gain of the position feedback model in the positive direction. This means how fast the position will change for 1 volt or percent of Control Output. This parameter is valid for models with a Model Order of 0, 1, or 2. This parameter is valid only for position axes. For more details, see the modeling topic.

Changing this Parameter

On position axes, always change this parameter only from Axis Tools for best results. If the position feedback model is being used, it will be suspended until the last position feedback model parameter register has been updated (Time Constant [Fx:152] for first order or Damping Factor [Fx:153] for second order). At this point, the new feedback model will be calculated and the model will be activated.

Certain position model parameter settings may result in an invalid model. If this occurs, change your settings. See the modeling topic for more details.

See Also

Modeling | Model Gain Negative

9.2.3.2.21.3. Model Gain Negative

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.151, where *n* = 12 + the axis number **RMC150:** %MD*n*.151, where *n* = 24 + the axis number **RMC200:** %MD*n*.52, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].ModGainNeg, where *n* is the axis number
How to Find:	**RMC75/150:** Axes Parameters Pane, All tab: Feedback → Filtering/Modeling **RMC200:** Axes Parameters Pane, All tab: Feedback → Model
Data Type:	REAL
Units:	**RMC75/150:** (pu/s)/volt **RMC200:** (pu/s)/%
Default Value:	**RMC75/150:** 1 **RMC200:** 0.1

Description

Note: The units differ between the RMC75/150 and RMC200, since the RMC75/150 Control Output is voltage, and the RMC200 Control Output is percent.

This parameter specifies the gain of the position feedback model in the negative direction. This means how fast the position will change for 1 volt or percent of Control Output. This parameter is valid for models with a Model Order of 0, 1, or 2. This parameter is valid only for position axes. For more details, see the modeling topic.

Changing this Parameter

On position axes, always change this parameter only from Axis Tools for best results. If the position feedback model is being used, it will be suspended until the last position feedback model parameter register has been updated (Time Constant [Fx:152] for first order or Damping Factor [Fx:153] for second order). At this point, the new feedback model will be calculated and the model will be activated.

Certain position model parameter settings may result in an invalid model. If this occurs, change your settings. See the modeling topic for more details.

See Also

Modeling | Model Gain Positive

9.2.3.2.21.4. Model Gain (Pressure or Force)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.161, where *n* = 12 + the axis number **RMC150:** %MD*n*.161, where *n* = 24 + the axis number **RMC200:** Primary: %MD*n*.51, where *n* = 384 + the axis number Secondary: %MD*n*.111, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].PFModelGain, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Secondary Feedback → Filtering/Modeling
Data Type:	REAL

Units:	**RMC75/150:** (Prs/s)/V or (Fr/s)/V **RMC200:** (Prs/s)/% or (Fr/s)/%
Default Value:	1

Description

This parameter specifies the gain of the pressure or force model. This parameter is valid for models with a Model Order of 0, 1, or 2. This parameter is valid only for pressure or force axes. For more details, see the modeling topic.

See Also

Modeling

9.2.3.2.21.5. Model Time Constant

Type:	Axis Parameter Register
Address:	**RMC75:** Position: %MD*n*.152, where *n* = 12 + the axis number Pressure/Force: %MD*n*.162, where *n* = 12 + the axis number **RMC150:** Position: %MD*n*.152, where *n* = 24 + the axis number Pressure/Force: %MD*n*.162, where *n* = 24 + the axis number **RMC200:** Primary: %MD*n*.53, where *n* = 24 + the axis number Secondary: %MD*n*.113, where *n* = 24 + the axis number
System Tag:	**Position:** _Axis[*n*].ModTimeConst, where *n* is the axis number **Pressure/Force:** _Axis[*n*].PFModelTimeConst, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling Axes Parameters Pane, All tab: Secondary Feedback → Filtering/Modeling
Data Type:	REAL
Units:	seconds
Range:	See the **Range** section below.
Default Value:	20

Description

This parameter specifies the Time Constant of the model. The Time Constant is only valid for models with a Model Order of one (1). Separate models exist for position and for pressure or force feedback. For more details, see the modeling topic.

Changing this Parameter

On position axes, always change this parameter only from Axis Tools for best results. If the position feedback model is being used, it will be suspended until the last position feedback model parameter register has been updated (Time Constant [Fx:152] for first order or Damping Factor

[Fx:153] for second order). At this point, the new feedback model will be calculated and the model will be activated.

Certain position model parameter settings may result in an invalid model. If this occurs, change your settings. See the modeling topic for more details.

Range

The Time Constant must be greater than or equal to the control loop time:

Control Loop Time	Minimum Time Constant
0.5ms	0.0005sec
1.0ms	0.001sec
2.0ms	0.002sec
4.0ms	0.004sec

See Also

Modeling

9.2.3.2.21.6. Model Natural Frequency

Type:	Axis Parameter Register
Address:	**RMC75:** Position: %MD*n*.152, where *n* = 12 + the axis number Pressure/Force: %MD*n*.162, where *n* = 12 + the axis number **RMC150:** Position: %MD*n*.152, where *n* = 24 + the axis number Pressure/Force: %MD*n*.162, where *n* = 24 + the axis number **RMC200:** Primary: %MD*n*.54, where *n* = 384 + the axis number Secondary: %MD*n*.114, where *n* = 384 + the axis number
System Tag:	**Position:** _Axis[*n*].ModNatFreq, where *n* is the axis number **Pressure/Force:** _Axis[*n*].PFModelNatFreq, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling Axes Parameters Pane, All tab: Secondary Feedback → Filtering/Modeling
Data Type:	REAL
Units:	Hz
Range:	See the **Range** section below.
Default Value:	20

Description

This parameter specifies the Natural Frequency of the feedback model. The Natural Frequency is only valid for models with a Model Order of two (2). Separate models exist for position and for pressure or force feedback. For more details, see the modeling topic.

Changing this Parameter

On position axes, always change this parameter only from Axis Tools for best results. If the position feedback model is being used, it will be suspended until the last position feedback model parameter register has been updated (Time Constant [Fx:152] for first order or Damping Factor [Fx:153] for second order). At this point, the new feedback model will be calculated and the model will be activated.

Certain position model parameter settings may result in an invalid model. If this occurs, change your settings. See the modeling topic for more details.

Range

The Natural Frequency must be less than or equal to the control loop time frequency divided by 4:

Control Loop Time	Max Natural Frequency
0.5ms	500Hz
1ms	250Hz
2ms	125Hz
4ms	62.5Hz

See Also

Modeling

9.2.3.2.21.7. Model Damping Factor

Type:	Axis Parameter Register
Address:	**RMC75:** Position: %MD*n*.153, where *n* = 12 + the axis number Pressure/Force: %MD*n*.163, where *n* = 12 + the axis number **RMC150:** Position: %MD*n*.153, where *n* = 24 + the axis number Pressure/Force: %MD*n*.163, where *n* = 24 + the axis number **RMC200:** Primary: %MD*n*.55, where *n* = 384 + the axis number Secondary: %MD*n*.115, where *n* = 384 + the axis number
System Tag:	**Position:** _Axis[*n*].ModDampFactor, where *n* is the axis number **Pressure/Force:** _Axis[*n*].PFModelDampFactor, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling Axes Parameters Pane, All tab: Secondary Feedback → Filtering/Modeling
Data Type:	REAL
Units:	seconds
Range:	[0..2]
Default Value:	0.75

Description

This parameter specifies the Damping Factor of the model. The Damping Factor is only valid for models with a Model Order of 2. Separate models exist for position and for pressure or force feedback. For more details, see the modeling topic.

Changing this Parameter

On position axes, always change this parameter only from Axis Tools for best results. If the position feedback model is being used, it will be suspended until the last position feedback model parameter register has been updated (Time Constant [Fx:152] for first order or Damping Factor [Fx:153] for second order). At this point, the new feedback model will be calculated and the model will be activated.

Certain position model parameter settings may result in an invalid model. If this occurs, change your settings. See the modeling topic for more details.

See Also

Modeling

9.2.3.2.21.8. Model Response

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.148, where *n* = 12 + the axis number **RMC150:** %MD*n*.148, where *n* = 24 + the axis number **RMC200:** %MD*n*.44, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].ModResponse, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback → Filtering/Modeling
Data Type:	REAL
Units:	Hz
Range:	5-150
Default Value:	20

Description

This parameter specifies the rate at which the model is updated to match the position feedback. The faster the response, the closer the model output will be to the feedback. In that sense, this parameter is like a filter cut-off frequency. For more details, see the modeling topic.

This parameter is valid only on position axes.

If the system model is accurate, the Model Response can be set to a low value. This will remove most of the quantization noise from the feedback which will allow the high-order gains to be increased without causing excessive noise on the Control Output.

With less accurate system models, the Model Response must be set to higher values which will leave more of the quantization noise on the feedback. The higher order gains must be kept lower to avoid excessive noise on the Control Output.

See Also

Modeling

9.2.3.2.22. Pressure & Force
9.2.3.2.22.1. Pressure/Force Scale

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.0, where *n* = 12 + the axis number Secondary Input: %MD*n*.18, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.0, where *n* = 24 + the axis number Secondary Input: %MD*n*.18, where *n* = 24 + the axis number **RMC200:** Primary Input: %MD*n*.20, where *n* = 384 + the axis number Secondary Input: %MD*n*.80, where *n* = 384 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].PrsScale **Force Input:** _Axis[*n*].FrcScale where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL
Units:	Pr/V, Pr/mA, Fr/V, F/mA, F/mV/V
Range:	<>0
Default Value:	1

Description

The Pressure Scale or Force Scale parameter is used together with the Pressure/Force Offset parameter to calculate the Actual Force or Actual Pressure from the voltage, current, or mV/V of the input.

On the RMC200:

- For single-input force axes, the Voltage/Current Offset is first added to the volts or current, then that value is used with the Force Scale and Force Offset to calculate the Actual Force.
- For load cell axes, the Millivolts/Volt Offset is first added to the Millivolts/Volt, then that value is used with the Force Scale and Force offsets to calculate the Actual Force.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Analog Pressure/Force Scaling

9.2.3.2.22.2. Pressure/Force Offset

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.1, where *n* = 12 + the axis number Secondary Input: %MD*n*.19, where *n* = 12 + the axis number **RMC150:**

	Primary Input: %MD*n*.1, where *n* = 24 + the axis number Secondary Input: %MD*n*.19, where *n* = 24 + the axis number **RMC200:** Primary Input: %MD*n*.21, where *n* = 384 + the axis number Secondary Input: %MD*n*.81, where *n* = 384 + the axis number
System Tag:	**Pressure Input:** _Axis[*n*].PrsOffset **Force Input:** _Axis[*n*].FrcOffset where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL
Units:	Pr or Fr
Range:	any
Default Value:	0

Description

The Pressure Offset or Force Offset parameter is used together with the Pressure Scale or Force Scale parameter to calculate the Actual Force or Actual Pressure from the voltage or current of the input.

The Offset is added to the scaled Pressure or Force.

On the RMC200:

- For single-input force axes, the Voltage/Current Offset is first added to the volts or current, then that value is used with the Force Scale and Force Offset to calculate the Actual Force.
- For load cell axes, the Millivolts/Volt Offset is first added to the Millivolts/Volt, then that value is used with the Force Scale and Force offsets to calculate the Actual Force.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Analog Pressure/Force Scaling

9.2.3.2.22.3. Voltage/Current Offset

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.30, where *n* = 384+ the axis number Secondary Input: %MD*n*.90, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].VCOffset, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback or Secondary Feedback
Data Type:	REAL

Units:	V or mA
Range:	any
Default Value:	0

Description

The Voltage/Current Offset parameter offsets the input voltage or current before the voltage or current is converted to force units. This parameter is available only on single-input force axes and is intended to provide an easy zero-signal adjustment for bipolar load cells with ±10 V, ±5 V, or ±20 mA signals. For mV/V inputs, see the Millivolts/Volt Offset parameter. The Negative Correction Factor parameter can also be useful for scaling load cell signals.

For bipolar load cells (tension and compression) with ±10 V, ±5 V, or ±20 mA signals, the Current/Voltage Offset is expected to be approximately 0, and the Negative Correction Factor close to 1. If using the Negative Correction Factor with a bipolar load cell with 4-20 mA feedback, the Current/Voltage Offset must be set to approximately -12 so that the Negative Correction Factor can apply to the negative side.

The Voltage/Current Offset is used in the calculation of the Actual Force as follows:

If ((Voltage + **Voltage/Current Offset**) < 0) Then
  Force = (Voltage + **Voltage/Current Offset**) x Negative Correction Factor x Force Scale + Force Offset
Else
  Force = (Voltage + **Voltage/Current Offset**) x Force Scale + Force Offset
EndIf

See Also

Negative Correction Factor | Force Scale | Force Offset

9.2.3.2.22.4. Negative Correction Factor

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary feedback: %MD*n*.22, where *n* = 384+ the axis number Secondary feedback: %MD*n*.82, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].NegCorrFactor, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback or Secondary Feedback
Data Type:	REAL
Units:	none
Range:	> 0
Default Value:	1.0

Description

The Negative Correction Factor is intended for minor adjustment of the feedback signal gain in the negative direction of a load cell input. When the voltage or current is negative, it is multiplied by the Negative Correction Factor. The Negative Correction Factor should typically be very close

to 1. This parameter is available only on single-input force axes. The Voltage/Current Offset or Millivolts/Volt Offset parameter can also be useful for load cell signals.

For bipolar load cells (tension and compression) with ±10 V, ±5 V, ±20 mA, or millivolt signals, the Voltage/Current Offset or Millivolts/Volt Offset is expected to be close to 0, and the Negative Correction Factor close to 1. If using the Negative Correction Factor with a bipolar load cell with 4-20 mA feedback, the Voltage/Current Offset must be set to approximately -12 so that the Negative Correction Factor can apply to the negative side.

The Negative Correction Factor is used in the calculation of the Actual Force as follows:

Voltage or Current Input (±10 V, ±5 V, ±20 mA)

If ((Voltage + Voltage/Current Offset) < 0) Then
  Force = (Voltage + Voltage/Current Offset) x **NegCorrFactor** x Force Scale + Force Offset
Else
  Force = (Voltage + Voltage/Current Offset) x Force Scale + Force Offset
EndIf

Millivolts/Volt Input

If ((Millivolts/Volt + Millivolts/Volt Offset) < 0) Then
  Force = (Millivolts/Volt + Millivolts/Volt Offset) x **NegCorrFactor** x Force Scale + Force Offset
Else
  Force = (Millivolts/Volt + Millivolts/Volt Offset) x Force Scale + Force Offset
EndIf

See Also

Voltage/Current Offset | Force Scale | Force Offset | Analog Pressure/Force Scaling | Load Cell Scaling

9.2.3.2.22.5. Channel A Force Scale, Channel B Force Scale

Type:	Axis Parameter Register
RMC75 Address:	**Channel A Force Scale:**
	Primary Input: %MD*n*.0, where *n* = 12 + the axis number
	Secondary Input: %MD*n*.18, where *n* = 12 + the axis number
	Channel B Force Scale:
	Primary Input: %MD*n*.0, where *n* = 12 + the axis number
	Secondary Input: %MD*n*.20, where *n* = 12 + the axis number
RMC150 Address:	**Channel A Force Scale:**
	Primary Input: %MD*n*.0, where *n* = 24 + the axis number
	Secondary Input: %MD*n*.18, where *n* = 24 + the axis number
	Channel B Force Scale:
	Primary Input: %MD*n*.0, where *n* = 24 + the axis number
	Secondary Input: %MD*n*.20, where *n* = 24 + the axis number
RMC200 Address:	**Channel A Force Scale:**

	Primary Input: %MD*n*.0, where *n* = 384 + the axis number Secondary Input: %MD*n*.18, where *n* = 384 + the axis number **Channel B Force Scale:** Primary Input: %MD*n*.0, where *n* = 384 + the axis number Secondary Input: %MD*n*.20, where *n* = 384 + the axis number
System Tag:	**Channel A Force Scale:** _Axis[*n*].FrcAScale **Channel B Force Scale:** _Axis[*n*].FrcBScale where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL
Units:	Fr/V or Fr/mA
Range:	<> 0
Default Value:	1

Description

The Channel A and B Force Scale parameters are used together with other parameters to calculate the Actual Force from the two inputs of a dual-input (differential) force axis. The calculation method varies by controller.

RMC75/150:

The Channel A and B Force Scale parameters convert the Channel A and B voltage or current to force units. The Channel A and B Force Offset are then added resulting in the individual Channel A and B Force values. The difference becomes the Actual Force:

Channel A Force = Channel A Voltage (or Current) * Channel A Scale + Channel A Offset

Channel B Force = Channel B Voltage (or Current) * Channel B Scale + Channel B Offset

Actual Force = Channel A Force - Channel B Force

RMC200:

The RMC200 calculates the pressures for each channel, then the forces for each channel, then takes the difference and applies the Dual Channel Force Offset to arrive at the Actual Force.

Channel A Pressure = Channel A Voltage (or Current) * Channel A Pressure Scale + Channel A Pressure Offset

Channel B Pressure = Channel B Voltage (or Current) * Channel B Pressure Scale + Channel B Pressure Offset

Channel A Force = Channel A Pressure * Channel A Force Scale

Channel B Force = Channel B Pressure * Channel B Force Scale

Actual Force = Channel A Force - Channel B Force + Dual Channel Force Offset

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Input Type: Dual-Input Force | Analog Pressure/Force Scaling

9.2.3.2.22.6. Channel A, B Force Offset

Type:	Axis Parameter Register
RMC75 Address:	**Channel A Force Offset:** Primary Input: %MD*n*.1, where *n* = 12 + the axis number Secondary Input: %MD*n*.19, where *n* = 12 + the axis number **Channel B Force Offset:** Primary Input: %MD*n*.3, where *n* = 12 + the axis number Secondary Input: %MD*n*.21, where *n* = 12 + the axis number
RMC150 Address:	**Channel A Force Offset:** Primary Input: %MD*n*.1, where *n* = 24 + the axis number Secondary Input: %MD*n*.19, where *n* = 24 + the axis number **Channel B Force Offset:** Primary Input: %MD*n*.3, where *n* = 12 + the axis number Secondary Input: %MD*n*.21, where *n* = 24 + the axis number
RMC200 Address:	n/a
System Tag:	**Channel A Force Offset:** _Axis[*n*].FrcAOffset **Channel B Force Offset:** _Axis[*n*].FrcBOffset where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL
Units:	Fr
Range:	any
Default Value:	0

Description

These parameters are used on the RMC75/150 together with the Channel A Force Scale and Channel B Force Scale parameters to calculate the Actual Force from inputs 0 and 1 of a dual-input (differential) force axis.

Channel A Force Offset shifts the Channel A Force.

Channel B Force Offset shifts the Channel B Force.

The RMC200 uses a different method of calculating the differential force, as described in Input Type: Dual-Input Force. Instead of the Channel A and B Force Offset, it uses the Dual Channel Force Offset.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Input Type: Dual-Input Force | Analog Pressure/Force Scaling

9.2.3.2.22.7. Channel A, B Pressure Scale

Type:	Axis Parameter Register
RMC75 Address:	n/a
RMC150 Address:	n/a
RMC200 Address:	**Channel A Pressure Scale:** Primary Input: %MD*n*.22, where *n* = 384 + the axis number Secondary Input: %MD*n*.82, where *n* = 384 + the axis number **Channel B Pressure Scale:** Primary Input: %MD*n*.25, where *n* = 384 + the axis number Secondary Input: %MD*n*.85, where *n* = 384 + the axis number
System Tag:	**Channel A Pressure Scale:** _Axis[*n*].PrsAScale **Channel B Pressure Scale:** _Axis[*n*].PrsBScale where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL
Units:	Prs/V or Prs/mA
Range:	<> 0
Default Value:	0

Description

The Channel A and B Pressure Scale parameters are used on the RMC200 together with other parameters to calculate the Actual Force from the two inputs of a dual-input (differential) force axis. The Channel A and B Pressure Scale parameters convert the Channel A and B voltage or current to Channel A and B pressures, as shown below. The RMC75/150 controllers use a different method of calculating the differential force, as described in Input Type: Dual-Input Force, and do not use the Channel A and B Pressure Scale parameters.

RMC200 Differential Force Calculation

The RMC200 calculates the pressures for each channel, then the forces for each channel, then takes the difference and applies the Dual Channel Force Offset to arrive at the Actual Force.

Channel A Pressure = Channel A Voltage (or Current) * **Channel A Pressure Scale** + Channel A Pressure Offset

Channel B Pressure = Channel B Voltage (or Current) * **Channel B Pressure Scale** + Channel B Pressure Offset

Channel A Force = Channel A Pressure * Channel A Force Scale

Channel B Force = Channel B Pressure * Channel B Force Scale

Actual Force = Channel A Force - Channel B Force + Dual Channel Force Offset

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Status Registers | Input Type: Dual-Input Force

9.2.3.2.22.8. Channel A, B Pressure Offset

Type:	Axis Parameter Register
RMC75 Address:	n/a
RMC150 Address:	n/a
RMC200 Address:	**Channel A Pressure Offset:** Primary Input: %MD*n*.21, where *n* = 384 + the axis number Secondary Input: %MD*n*.81, where *n* = 384 + the axis number **Channel B Pressure Offset:** Primary Input: %MD*n*.24, where *n* = 384 + the axis number Secondary Input: %MD*n*.84, where *n* = 384 + the axis number
System Tag:	**Channel A Pressure Offset:** _Axis[*n*].PrsAOffset **Channel B Pressure Offset:** _Axis[*n*].PrsBOffset where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Feedback Axes Parameters Pane, Setup tab: Secondary Feedback
Data Type:	REAL
Units:	Prs
Range:	any
Default Value:	0

Description

The Channel A and B Pressure Offset parameters are used on the RMC200 together with other parameters to calculate the Actual Force from the two inputs of a dual-input (differential) force axis. The Channel A and B Pressure Offset parameters convert the Channel A and B voltage or current to Channel A and B pressures, as shown below. The RMC75/150 controllers use a different method of calculating the differential force, as described in Input Type: Dual-Input Force, and do not use the Channel A and B Pressure Offset parameters.

RMC200 Differential Force Calculation

The RMC200 calculates the pressures for each channel, then the forces for each channel, then takes the difference and applies the Dual Channel Force Offset to arrive at the Actual Force.

Channel A Pressure = Channel A Voltage (or Current) * Channel A Pressure Scale + **Channel A Pressure Offset**

Channel B Pressure = Channel B Voltage (or Current) * Channel B Pressure Scale + Channel B Pressure Offset

Channel A Force = Channel A Pressure * Channel A Force Scale

Channel B Force = Channel B Pressure * Channel B Force Scale

Actual Force = Channel A Force - Channel B Force + Dual Channel Force Offset

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Status Registers | Input Type: Dual-Input Force

9.2.3.2.22.9. Dual Channel Force Offset

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.26, where *n* = 384 + the axis number Secondary Input: %MD*n*.86, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].FrcOffset, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Primary Control Setup Axes Parameters Pane, All tab: Secondary Control Setup
Data Type:	REAL
Units:	Frc
Range:	any
Default Value:	0

Description

The Dual Channel Force Offset parameter is used on the RMC200 together with other parameters to calculate the Actual Force from the two inputs of a dual-input (differential) force axis. The Dual Channel Force Offset applies a final offset to the difference of the Channel A Force and Channel B Force. The Dual Channel Force Offset is useful for taking into account the weight of tooling attached to the cylinder. On a single-rod cylinder, the first input, Channel A, must be wired to the cap end of the cylinder, and the second input, Channel B, must be wired to the rod end of the cylinder.

RMC200 Differential Force Calculation

The RMC200 calculates the pressures for each channel, then the forces for each channel, then takes the difference and applies the Dual Channel Force Offset to arrive at the Actual Force.

Channel A Pressure = Channel A Voltage (or Current) * Channel A Pressure Scale + Channel A Pressure Offset

Channel B Pressure = Channel B Voltage (or Current) * Channel B Pressure Scale + Channel B Pressure Offset

Channel A Force = Channel A Pressure * Channel A Force Scale

Channel B Force = Channel B Pressure * Channel B Force Scale

Actual Force = Channel A Force - Channel B Force + **Dual Channel Force Offset**

The RMC75/150 controllers use a different method of calculating the differential force, as described in Input Type: Dual-Input Force.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Parameter Registers | Input Type: Dual-Input Force | Analog Pressure/Force Scaling

9.2.3.2.23. Limit Inputs
9.2.3.2.23.1. Positive Limit Input

Type:	Axis Parameter Register Bit Parameter
Address:	**RMC75:** %MD*n*.9.4-7, where *n* = 12 + the axis number **RMC150:** %MD*n*.9.8-15, where *n* = 24 + the axis number **RMC200:** %MD*n*.34.4-12, where *n* = 24 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].PriInputBits.PosLimIn **RMC200:** _Axis[*n*].LimitInputs.PosLimitIn
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	bits
Default Value:	0 (none)

Description

Each RMC axis allows for optional physical limit inputs to specify the boundaries in which the axis is allowed to operate. Each axis can have two physical limit inputs: Positive Limit Input and Negative Limit Input. Typically, these inputs are wired to limit switches.

The physical limit inputs differ from the Travel Limits, which limit the range of travel of an axis based on the feedback (position, pressure, etc.). The Travel Limits are required to be set on an axis; physical limit inputs are optional.

The Positive Limit Input bit parameter selects which discrete input, if any, should be used as a 'Positive Limit Input'. For details on the operation of the physical limit inputs, see the Physical Limit Inputs topic.

In RMCTools, this parameter will allow you choose an input from the drop-down box. Inputs will be listed by their address. Use the Discrete I/O Configuration editor to find the addresses of inputs.

The following options are possible:

RMC75/150:

Option	Description
none	This is the default setting.
Fault Input	**RMC75**: The Fault Input of the axis. **RMC150**: The Fault Input of the axis. Only available on the Quadrature Module.

Dedicated	Available on RMC75 QA and RMC150 Quadrature modules only. This is the **PosLim** input on the module.
general input	**RMC75**: any input from a D8 module, but only from the first 12 I/O points as listed in the Discrete I/O Monitor. **RMC150**: any general discrete input from any module.

RMC200:

Option	Description
none	This is the default setting.
general input	Any discrete input from any module. Use the The Discrete I/O Configuration Editor to find which addresses correspond to which modules and discrete inputs.

Positive Limit Format Details

This section is primarily for addressing the Positive Limit Input parameter when communicating with the RMC from an external device. This information is not necessary when configuring the Positive Limit Input in RMCTools.

The Positive Limit Input is selected with bits in the Primary Input Bits register, as shown in the following table:

RMC75

Value of bits 4-7	Positive Limit Input
0	none
1	Fault Input
2	Dedicated (RegX/PosLim)
3	-
4	%IX0
5	%IX1
6	%IX2
7	%IX3
8	%IX4
9	%IX5
10	%IX6
11	%IX7
12	%IX8
13	%IX9
14	%IX10
15	%IX11

RMC150

Value of bits 8-15	Postive Limit Input
0	none
1	Fault Input
2	Dedicated (RegX/PosLim)
3	-
(32 x *slot)* + (*i/o point)* + 4	%IX*slot*.(*i/o point)*

The slot numbering starts with 0 for the left-most module in the RMC150.

The Discrete I/O Monitor also displays the slot number and I/O number of each I/O point. For example, the I/O point 4 in slot 3 is displayed as 3.4.

For example, input 4 in slot 0 is:

(32 x 0*)* + (4*)* + 4 = 8

For example, input 7 in slot 5 is:

(32 x 5*)* + (7*)* + 4 = 171

RMC200

Value of bits 4-12	Positive Limit Input
0	none
(32 x *slot)* + (*i/o point)*	%IX*slot*.(*i/o point)*

The slot numbering starts with 0 for the power supply module, 1 for the CPU, and 2 for the first I/O module on the RMC200.

The Discrete I/O Configuration Editor also displays the slot number and I/O number of each I/O point. For example, the I/O point 4 in slot 3 is displayed as %IX3.4.

For example, input 1 in slot 1 (the CPU) is:

(32 x 1*)* + (1*)* = 33

For example, input 7 in slot 5 is:

(32 x 5*)* + (7*)* = 167

For details about the register containing the Positive Limit Input bit, see the RMC75/150 Primary Input Bits Register or RMC200 Limit Inputs Configuration.

See Also

Parameter Registers | Negative Limit Input | Travel Limits | Physical Limit Inputs

9.2.3.2.23.2. Negative Limit Input

Type:	Axis Parameter Register Bit Parameter
Address:	**RMC75:** %MD*n*.9.8-11, where *n* = 12 + the axis number **RMC150:** %MD*n*.9.16-23, where *n* = 24 + the axis number **RMC200:** %MD*n*.34.16-24, where *n* = 384 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].PriInputBits.NegLimitIn **RMC200:** _Axis[*n*].LimitInputs.NegLimitIn
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	bits
Default Value:	0 (none)

Description

Each RMC axis allows for optional physical limit inputs to specify the boundaries in which the axis is allowed to operate. Each axis can have two physical limit inputs: Positive Limit Input and Negative Limit Input. Typically, these inputs are wired to limit switches.

The physical limit inputs differ from the Travel Limits, which limit the range of travel of an axis based on the feedback (position, pressure, etc.). The Travel Limits are required to be set on an axis; physical limit inputs are optional.

The Negative Limit Input bit parameter selects which discrete input, if any, should be used as a 'Negative Limit Input'. For details on physical limit inputs, and the options of this parameter, see the Physical Limit Inputs topic.

In RMCTools, this parameter will allow you choose an input from the drop-down box. Inputs will be listed by their address. Use the Discrete I/O Configuration editor to find the addresses of inputs.

The following options are possible:

RMC75/150:

Option	Description
none	This is the default setting.
Fault Input	**RMC75**: The Fault Input of the axis. **RMC150**: The Fault Input of the axis. Only available on the Quadrature Module.
Dedicated	Available on RMC75 QA and RMC150 Quadrature modules only. This is the **NegLim** input on the module.
general input	**RMC75**: any input from a D8 module, but only from the first 12 I/O points as listed in the Discrete I/O Monitor. **RMC150**: any general discrete input from any module.

RMC200:

Option	Description
none	This is the default setting.

general input	Any discrete input from any module. Use the The Discrete I/O Configuration Editor to find which addresses correspond to which modules and discrete inputs.

Negative Limit Format Details

This section is primarily for addressing the Negative Limit Input parameter when communicating with the RMC from an external device. This information is not necessary when configuring the Negative Limit Input in RMCTools.

The Negative Limit Input is selected with bits in the Primary Input Bits register, as shown in the following table:

RMC75

Value of bits 8-11	Negative Limit Input
0	none
1	Fault Input
2	Dedicated (RegY/NegLim)
3	-
4	%IX0
5	%IX1
6	%IX2
7	%IX3
8	%IX4
9	%IX5
10	%IX6
11	%IX7
12	%IX8
13	%IX9
14	%IX10
15	%IX11

RMC150

Value of bits 16-23	Negative Limit Input
0	none
1	Fault Input
2	Dedicated (RegY/NegLim)
3	-
(32 x *slot)* + (*i/o point)* + 4	%IX*slot*.(*i/o point)*

The slot numbering starts with 0 for the left-most module in the RMC150.

The Discrete I/O Monitor also displays the slot number and I/O number of each I/O point. For example, the I/O point 4 in slot 3 is displayed as 3.4.

For example, input 4 in slot 0 is:

(32 x 0*)* + (4*)* + 4 = 8

For example, input 7 in slot 5 is:

(32 x 5*)* + (7*)* + 4 = 171

RMC200

Value of bits 16-24	Negative Limit Input
0	none
(32 x *slot)* + (*i/o point)*	%IX*slot*.(*i/o point)*

The slot numbering starts with 0 for the power supply module, 1 for the CPU, and 2 for the first I/O module on the RMC200.

The Discrete I/O Configuration Editor also displays the slot number and I/O number of each I/O point. For example, the I/O point 4 in slot 3 is displayed as %IX3.4.

For example, input 1 in slot 1 (the CPU) is:

(32 x 1*)* + (1*)* = 33

For example, input 7 in slot 5 is:

(32 x 5*)* + (7*)* = 167

For details about the register containing the Negative Limit Input bit, see the RMC75/150 Primary Input Bits Register or RMC200 Limit Inputs Configuration.

See Also

Parameter Registers | Positive Limit Input | Travel Limits | Physical Limit Inputs

9.2.3.2.23.3. Limit Input Polarity

Type:	Axis Parameter Register Bit Parameter
Address:	**RMC75:** %MD*n*.9.12, where *n* = 12 + the axis number **RMC150:** %MD*n*.9.1, where *n* = 24 + the axis number **RMC200:** %MD*n*.34.0, where *n* = 384 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].PriInputBits.LimitInPol **RMC200:** _Axis[*n*].LimitInputs.LimitInPol
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	bit
Range:	Active High (0), Active Low (1)
Default Value:	Active High (0)

Description

This parameter specifies the polarity of the Positive Limit Input and Negative Limit Input. It can be set to the following:

- **Active High**
  When the input is physically "on" (voltage is applied), the Positive Limit Input is active.
- **Active Low**
  When the input is physically "off" (voltage is not applied), the Negative Limit Input is active.

The Positive Limit Input or Negative Limit Input status bits indicate the state of the corresponding limit input. For details on voltage levels, see the specifications for the module the input is located on.

For details about the register containing the Limit Input Polarity bit, see the RMC75/150 Primary Input Bits Register or the RMC200 Limit Inputs Configuration Register.

See Also

Parameter Registers | Positive Limit Input | Negative Limit Input

---

9.2.3.2.23.4. Limit Inputs Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.34, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].LimitInputs, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	DWORD

Description

The RMC200 Limit Inputs Configuration register contains the bit-addressable limit input configuration parameters. The limit inputs are optional discrete inputs that indicate that the axis has reached the limit of travel on for position or velocity axes. This topic lists the bit address for each parameter. Each

parameter is accessible in RMCTools via the Axis Parameters Pane. For details on each parameter, see the respective links.

The corresponding tags in the RMC75/150 are stored in the Primary Input Bits Register.

Tag Names

This register contains the following parameters. The tag names and bits for each are given below.

RMC200

Parameter	Tag	Bits
Limit Input Polarity	LimitInPol	0
Positive Limit Input	PosLimitIn	4-12
Negative Limit Input	NegLimitIn	16-24

For details on the values that each parameter represents, see the respective parameter topic.

See Also

Positive Limit Input | Negative Limit Input | Limit Input Polarity

9.2.3.2.24. Transducer Specific
9.2.3.2.24.1. MDT Type

Type:	Axis Parameter Register
Address:	**RMC75:** %MX*n*.10.0-2, where *n* = 12 + the axis number **RMC150:** %MX*n*.10.0-2, where *n* = 24 + the axis number **RMC200:** %MX*n*.60.2-4, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].MDTSSICfg.MDTType
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	Bits - see below

Description

This parameter is valid only on axes with MDT feedback. The MDT Type parameter specifies the type of Magnetostrictive Displacement Transducer (MDT). The following options are available:

- **St/St Rising**
  The MDT feedback is Start/Stop, measured on the rising edge.
- **St/St Falling**
  The MDT feedback is Start/Stop, measured on the falling edge.
- **PWM**
  The MDT feedback is Pulse Width Modulated.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Format Details

This section is primarily for addressing the MDT Type parameter when communicating with the RMC from an external device. This information is not necessary when configuring the MDT Type in RMCTools.

RMC75/150:

The MDT Type is selected with bits 0-2 in the SSI/MDT Configuration Register. These bits correspond to the MDT Type as shown in the following table:

Bit 2	Bit 1	Bit 0	Value	MDT Type
0	0	0	0	Start/Stop Rising
0	0	1	1	Start/Stop Falling
0	1	0	2	PWM

RMC200:

The MDT Type is selected with bits 2-4 in the SSI/MDT Configuration Register. These bits correspond to the MDT Type as shown in the following table:

Bit 4	Bit 3	Bit 2	Value	MDT Type
0	0	0	0	Start/Stop Rising
0	0	1	1	Start/Stop Falling
0	1	0	2	PWM

See Also

Parameter Registers | MA Module (RMC75) | MDT Module (RMC150) | S8 Module (RMC200) | U14 Module (RMC200) | MDT Fundamentals

---

9.2.3.2.24.2. MDT Blanking Parameter

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** %MX*n*.10.3, where *n* = 24+ the axis number **RMC200:** %Mx*n*.60.5-6, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].MDTSSICfg.MDTBP, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	**RMC75/150:** boolean **RMC200:** bits
Range:	5µs (0), 21µs (1)
Default Value:	5µs (0)

Description

The MDT Blanking Period parameter specifies the amount of time that the MDT input waits after the start pulse before looking for the stop pulse. This parameter is valid only on RMC150 and RMC200 axes with MDT feedback.

Typically, this parameter should be left at its default setting. For older Temposonics I and II transducers with a neuter output, this parameter needs to be set to 21µs, since the start pulses contain ringing that lasts longer then the default 5µs.

The RMC75 blanking period is fixed at 5µsec. Transducers that require the longer 21µsec blanking period will require using the RMC150 or RMC200.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the

axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See the SSI/MDT Configuration Register for details about the register containing these bits.

See Also

Parameter Registers | MDT Module (RMC150) | S8 Module (RMC200) | U14 Module (RMC200) | MDT Fundamentals

9.2.3.2.24.3. MDT Interrogation Period

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MX*n*.60.8-11, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].MDTSSICfg.MDTIntPeriod, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	bits
Default Value:	0 (loop time)

Description

This parameter is valid only on axes with MDT feedback. The MDT Interrogation Period defines the time period between sampling of the transducer. The RMC200 allows this time to be different than the controller's Loop Time, whereas the RMC75 and RMC150 sample only at the set loop time. The following options are available:

- Loop Time: this will interrogate at the same time period as the loop time.
- 125 µs
- 250 µs
- 500 µs
- 1 ms
- 2 ms
- 4 ms
- 8 ms

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Format Details

This section is primarily for addressing the MDT Interrogation Period parameter when communicating with the RMC from an external device. This information is not necessary when configuring the MDT Type in RMCTools.

RMC200:

The MDT Type is selected with bits 8-11 in the SSI/MDT Configuration Register. These bits correspond to the MDT Type as shown in the following table:

Bit 11	Bit 10	Bit 9	Bit 8	Value	MDT Interrogation Period

0	0	0	0	0	Loop Time
0	0	0	1	1	125 µs
0	0	1	0	2	250 µs
0	0	1	1	3	500 µs
0	1	0	0	4	1 ms
0	1	0	1	5	2 ms
0	1	1	0	6	4 ms
0	1	1	1	7	8 ms

See Also

Parameter Registers | MDT Module (RMC150) | S8 Module | MDT Fundamentals

9.2.3.2.24.4. Absolute/Incremental

Type:	Axis Parameter Register Bit Parameter
Address:	**RMC75:** %MD*n*.10.18, where *n* = 12 + the axis number **RMC150:** %MD*n*.10.18 , where *n* = 24 + the axis number **RMC200:** %MD*n*.60.28 , where *n* = 384 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].MDTSSICfg.Inc or _Axis[*n*].ResCfg.Inc **RMC200:** _Axis[*n*].MDTSSICfg.Inc
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	bit
Range:	Absolute (0), Incremental (1)
Default Value:	Absolute (0)

Description

The Absolute/Incremental bit parameter specifies whether the feedback is incremental or absolute. This parameter is valid only for axes with SSI or Resolver feedback. Other axes types are by definition absolute or incremental, and a choice is not available. Typically, this parameter should be set to Absolute.

When this bit is set, the axis is incremental. When this bit is cleared, the axis is absolute.

See the SSI/MDT Configuration Register or Resolver Configuration Register for details about the register containing this bit.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Definition of Incremental and Absolute

Absolute encoders have a unique value (voltage, binary count, SSI etc) for each mechanical position. When an absolute encoder is turned on, the position of an absolute encoder is known.

An **incremental** encoder does not have a unique value for each mechanical position. When it is turned on, the position of an incremental encoder is not known. An incremental encoder measures the distance traveled and does not provide absolute positions. An example is a Quadrature encoder.

See Also

Parameter Registers | Axis Type: Incremental/Absolute

9.2.3.2.24.5. SSI/MDT Feedback Type

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.10.8, where *n* = 12 + the axis number **RMC150:** not available **RMC200:** %MD*n*.60.0-1, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].MDTSSICfg.SSI
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	Bits - see below

Description

This parameter is valid only for axes with feedback from the RMC75 MA axis module, and with feedback from an RMC200 S8 Module or U14 Module channel that is an SSI/MDT input. The Feedback Type parameter specifies the feedback type of the axis. The following options are available:

- **SSI**
  Synchronous Serial Interface.
- **MDT**
  Magnetostrictive Displacement Transducer. Supports Start/Stop rising edge, Start/Stop falling edge, and Pulse-Width Modulated (PWM).

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Format Details

This section is primarily for addressing the Feedback Type parameter when communicating with the RMC from an external device. This information is not necessary when configuring the Feedback in RMCTools.

RMC75/150:

Bit 8	MDT Type
0	MDT
1	SSI

RMC200:

Bit 1	Bit 0	MDT Type
0	0	MDT
0	1	SSI

See the SSI/MDT Configuration Register for details about the register containing these bits.

See Also

Parameter Registers | MA module (RMC75) | S8 Module (RMC200) | U14 Module (RMC200) | MDT Fundamentals | SSI Fundamentals

9.2.3.2.24.6. SSI Format

Type:	Axis Parameter Register
Address:	**RMC75:** %MX*n*.10.9, where *n* = 12 + the axis number **RMC150:** %MX*n*.10.9, where *n* = 24 + the axis number **RMC200:** %MX*n*.60.14-15, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].MDTSSICfg.GrayCode
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	Bit - see below

Description

This parameter is valid only on axes with SSI feedback. You must set this parameter to match your SSI transducer or encoder. The following options are available:

- **Binary**
- **Gray**

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Format Details

This section is primarily for addressing the SSI Format parameter when communicating with the RMC from an external device. This information is not necessary when configuring the SSI Format in RMCTools.

RMC75/150:

The SSI Format is selected with bit 9 in the SSI/MDT Configuration Register. This bit corresponds to the SSI Format as shown in the following table:

Bit 9	SSI Format
0	Binary
1	Gray Code

RMC200:

The SSI Format is selected with bits 14-15 in the SSI/MDT Configuration Register. This bit corresponds to the SSI Format as shown in the following table:

Bit 15	Bit 14	SSI Format
0	0	Binary
0	1	Gray Code

See Also

Parameter Registers | SSI/MDT Configuration Register | SSI Fundamentals

9.2.3.2.24.7. SSI Data Bits

Type:	Axis Parameter Register
Address:	**RMC75:** %MX*n*.10/12-17, where *n* = 12 + the axis number **RMC150:** %Mx*n*.10.12-17, where *n* = 24 + the axis number **RMC200:** %Mx*n*.60.20-25, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].MDTSSICfg.SSIDataBits
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	Bits - see below

Description

This parameter is valid only on axes with SSI feedback. It tells the RMC how many data bits your SSI encoder has. You must set this parameter to match your SSI transducer or encoder. The SSI Data Bits range depends on the controller:

Module	SSI Data Bits
RMC75 MA2	8-32
RMC150 SSI	8-31
RMC150 UI/O	8-32
RMC200 S8	8-32
RMC200 U14	8-32

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

24-Bit Limitation

See the **Exceeding 24 Bits** section of the Feedback Resolution topic for details on the 24-bit resolution limitation of the SSI input.

Address Format Details

This section is for addressing the SSI Data Bits parameter when communicating with the RMC from an external device. This information is not necessary when configuring the SSI Data Bits in RMCTools.

The RMC75/150 SSI Data Bits are selected with bits 12-17 in the SSI/MDT Configuration Register. The RMC200 SSI Data Bits are selected with bits 20-25 in the SSI/MDT Configuration Register. These bits are the binary representation of the number of data bits.

See Also

Parameter Registers | SSI/MDT Configuration Register | SSI Fundamentals

---

9.2.3.2.24.8. SSI Clock Rate

Type:	Axis Parameter Register

Address:	**RMC75:** %MX*n*.10.10-11, where *n* = 12 + the axis number **RMC150:** %MX*n*.10.10-11, where *n* = 24 + the axis number **RMC200:** %MX*n*.60.16-19, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].MDTSSICfg.SSIClockRate
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	Bits - see below

Description

This parameter defines the output clock rate of the SSI clock signal. Normally, this parameter need not be changed.

The clock signal clocks out as many bits as defined by the SSI Data Bits. This clocking is done each loop time of the controller. Therefore, the SSI Clock Rate does not define how often the SSI data is queried, it only defines the rate of the clock bits. This rate must be fast enough that all the bits can be clocked within one loop time.

This parameter is valid only on axes with SSI feedback. This parameter must be set to match your SSI transducer or encoder, but usually, the transducer or encoder handles a wide range of clock rates, so this parameter need not be changed from its default setting. The clock rate can be set to the following values:

RMC75 MA Module	RMC150 SSI Module	RMC150 UI/O Module	RMC200 S8 Module	RMC200 U14 Module
150 kHz	230 kHz	250 kHz	100 kHz	100 kHz
250 kHz	921 kHz	500 kHz	150 kHz	150 kHz
375 kHz		971 kHz	250 kHz	250 kHz
			400 kHz	400 kHz
			625 kHz	625 kHz
			1000 kHz	1000 kHz
			1500 kHz	1500 kHz
			2500 kHz	2500 kHz

SSI Monitor Mode

In SSI Monitor mode, the RMC receives a clock signal, and therefore this parameter does not define an output clock signal. However, this parameter is used to calculate the clock period, which in turn is used to determine when to sample the Data line at the end of the SSI transaction to check for wire break.

Select the closest frequency to the one the SSI master is using. When using long cables and the closest frequency is higher than the master's actual frequency, it may be necessary to select the next lower frequency.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Choosing the SSI Clock Rate

The clock rate affects the maximum cable length allowed for SSI signals. A faster clock rate helps obtain the most accurate feedback values. Before selecting the clock rate, verify that your encoder or transducer supports the clock rate.

For modules that support wire delay compensation, longer cables lengths may be used. See **Wire Delay Compensation** below.

Clock Rate	Maximum Cable Length*
100 kHz	2100 ft (640 m)
150 kHz	1360 ft (415 m)
230 kHz	850 ft (255 m)
250 kHz	770 ft (235 m)
375 kHz	475 ft (145 m)
400 kHz	450 ft (135 m)
500 kHz	325 ft (99 m)
625 kHz	225 ft (70 m)
921 kHz	120 ft (37 m)
971 kHz	110 ft (34 m)
1000 kHz	100 ft (30 m)
1500 kHz	25 ft (7.5 m)
2500 kHz	3 ft (1 m)

The cable lengths are approximate, and may be affected by the type of wire and transducer. Here is an approximate formula used to calculate the wire lengths in the table:

Length = ((1,000,000 / Clock_Rate) – Delay) * Wire_Speed / 2

where:

Clock_Rate is in kHz

Delay is the sum of the propagation delays in the RMC and the transducer in nanoseconds (approximately 550 ns).

Wire_Speed is the speed at which the signal travels down the wire in ft/ns (approximately 0.45 ft/ns).

Wire Delay Compensation

For SSI inputs on the RMC150 UI/O module, wire delay compensation is available. If your cable length exceeds the length for your clock rate given in the table above, you should use the SSI Wire Delay parameter to account for the time delay of the SSI signal.

Format Details

This section is primarily for addressing the SSI Clock Rate parameter when communicating with the RMC from an external device. This information is not necessary when configuring the SSI Clock Rate in RMCTools.

The SSI Clock Rate is selected with bits 10-11 in the SSI/MDT Configuration Register. These bits correspond to the SSI Clock Rate as shown in the table below. The RMC150 clock rate is not selectable, and the bits should be zero.

RMC75 MA Module

Bit 11	Bit 10	Value	SSI Clock Rate
0	0	0	375 kHz
0	1	1	250 kHz
1	0	2	150 kHz

RMC150 SSI Module

Bit 11	Bit 10	Value	SSI Clock Rate

0	0	0	230 kHz
0	1	1	921 kHz

RMC150 UI/O Module

Bit 11	Bit 10	Value	SSI Clock Rate
0	0	0	250 kHz
0	1	1	500 kHz
1	0	2	971 kHz

RMC200 S8 and U14 Modules

Bit 19	Bit 18	Bit 17	Bit 16	Value	SSI Clock Rate
0	0	0	0	0	100 kHz
0	0	0	1	1	150 kHz
0	0	1	0	2	250 kHz
0	0	1	1	3	400 kHz
0	1	0	0	4	625 kHz
0	1	0	1	5	1000 kHz
0	1	1	0	6	1500 kHz
0	1	1	1	7	2500 kHz

See Also

Parameter Registers | SSI/MDT Configuration Register | SSI Fundamentals

9.2.3.2.24.9. SSI Clock Mode

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** %Mx*n*.10.1, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].MDTSSICfg.SSIMonitor
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	boolean
Range:	standard (0), monitor (1)
Default Value:	standard (0)

Description

This parameter is valid on axes with SSI feedback on the UI/O Module. It provides two options for the clock on an SSI input assigned to an axis:

- **Standard Mode**
  In standard mode, the SSI channel sends out a clock signal to the SSI device, and the SSI device responds with the data. This is the typical behavior when an SSI device is connected to an RMC.

- **Monitor Mode**
  In monitor mode, the SSI channel clock is an input. This mode is used to monitor the communication between an RMC and an SSI device. This is useful for daisy-chaining multiple UI/O modules to a single SSI device.

See Also

Parameter Registers | SSI/MDT Configuration Register | SSI Fundamentals | UI/O Module

9.2.3.2.24.10. SSI Termination Parameter

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** %MD*n*.10.0, where *n* = 24+ the axis number **RMC200:** %MD*n*.60.12-13, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].MDTSSICfg.SSITerm, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	bits
Range:	SSI Input: none (0), ±Data (1) SSI Monitor: none (0), ±Data/±Clock (1)
Default Value:	SSI Input: ±Data (1) SSI Monitor: none (0)

Description

The SSI Termination parameter applies to axes with SSI feedback on an RMC150 Universal I/O (UI/O) and RMC200 S8 and U14 modules. These modules have internal software-selectable termination.

SSI signals need to be terminated at the input end. For a typical RMC SSI input that is connected directly to an SSI device (the transducer or encoder), the ±Data signals are an input to the RMC, and need to be terminated. The ±Clock signals are an outputs from the RMC, so ±Clock termination on the RMC side doesn't apply.

For SSI Monitor, both the ±Data and ±Clock are inputs, so any termination applies to both ±Data and ±Clock. If the SSI Monitor is in the middle of a daisy-chained SSI configuration, it should not have termination, and only the last input in the chain should have termination. If the SSI Monitor is the endpoint of the wiring, then termination should be applied. See the wiring topics for each module for details.

See the SSI/MDT Configuration Register for details about the register containing these bits.

See Also

Parameter Registers | Universal I/O Module

9.2.3.2.24.11. SSI Wire Delay

Type:	Axis Parameter Register
Address:	**RMC75:** n/a

	RMC150: %MD*n*.14, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].WireDelay
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	DINT

Description

This parameter is valid on axes with SSI feedback on the UI/O Module. The SSI Wire Delay allows the SSI input to take into account the delay of the signal in the wire. To ensure error-free feedback, this parameter should be used if the wire length exceeds the values given in the table below.

Clock Rate	Maximum Cable Length*
250 kHz	770 ft (235 m)
500 kHz	325 ft (99 m)
971 kHz	110 ft (34 m)

***Note:**
The cable lengths depend on the type of wire used and on the internal delays in the transducers. The transducer delays are usually not specified so the wire delay may need to be found empirically.

Setting the Wire Delay

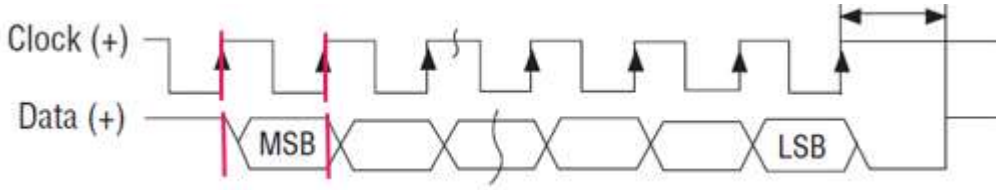
To set the SSI Wire Delay parameter:

1. In the Axis Parameters, select the **SSI Wire Delay** parameter and then click the ellipsis button .
2. Enter the Delay Time directly, or choose to estimate the delay based on the cable length. The Wizard will calculate and/or adjust the Wire Delay Time to account for resolution of the internal delay timer. The wizard will display the value that will be applied.
3. Click **OK**. The time to delay will be converted to the nearest time delay that can be represented with [1..8] * [0..31] / 33MHz.

Details

Minimal Delay

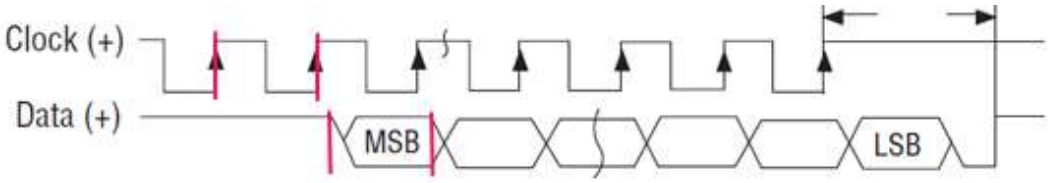
The timing diagram below shows an SSI system with very little delay. On the first rising edge of the Clock, the SSI device puts the first bit of data on the Data line. By the next rising edge of the Clock, when the RMC samples the data, the data is valid, and the read is successful.



Excessive Delay

The timing diagram below shows an SSI system with a time delay of more then one clock period. On the first rising edge of the Clock, the SSI device puts the first bit of data on the Data line. By the next rising edge of the Clock, when the RMC samples the data, the data from the SSI device has not yet arrived, and the SSI input will not return the correct value.

To compensate for the delay, set SSI Wire Delay parameter. You can enter the wire length or enter the time delay directly. The SSI input will then use the delay value to correctly read the SSI input data.



See Also

Parameter Registers | SSI/MDT Configuration Register | SSI Fundamentals

9.2.3.2.24.12. SSI Home Source

Type:	Axis Parameter Register
Address:	**RMC75:** %MX*n*.10.20-23, where *n* = 12 + the axis number **RMC150:** n/a **RMC200:** n/a
System Tag:	_Axis[*n*].MDTSSICfg.SSIHomeSrc
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	Bits - see below

Description

This parameter is available only on RMC75 SSI axes configured as Incremental. This parameter specifies the input to use as a home input. By default, this parameter is set to **none**.

Inputs that can be used as an SSI Home Source are:

- The Fault Input of the axis
- %IX0 - %IX11 (discrete inputs 0 - 11)

See the Homing topic for details on homing.

Format Details

This section is primarily for addressing the SSI Home Source parameter when communicating with the RMC from an external device. This information is not necessary when configuring the SSI Home Source in RMCTools.

The SSI Home Source is selected with bits 20-23 in the SSI/MDT Configuration Register. These bits correspond to the SSI Home Source as shown in the following table:

RMC75

Value of bits 20-23	SSI Home Source
0	none
1	Fault Input
2	-
3	-
4	%IX0
5	%IX1
6	%IX2
7	%IX3
8	%IX4

9	%IX5
10	%IX6
11	%IX7
12	%IX8
13	%IX9
14	%IX10
15	%IX11

See the SSI/MDT Configuration Register for details about the register containing these bits.

See Also

Parameter Registers | SSI/MDT Configuration Register | SSI Fundamentals | Homing

---

9.2.3.2.24.13. SSI Overflow Mode

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.10.4-6, where *n* = 12 + the axis number **RMC150:** %MD*n*.10.4-6, where *n* = 24 + the axis number **RMC200:** %MD*n*.61.0-2, where *n* = 24 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].MDTSSICfg.SSIOverflowMode **RMC200:** _Axis[*n*].SSIDataCfg.SSIOverflowMode
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	Bits - see below

Description

This parameter is valid only on axes with SSI feedback. This parameter selects which, if any, SSI value causes the Transducer Overflow error bit to turn on. For absolute linear SSI axes, choose from the options listed below. This parameter does not apply to and will be ignored by SSI axes configured in rotary or incremental modes because all SSI values are valid in those modes.

- **None**
  No transducer overflow error is ever triggered.
- **Zeros**
  A transducer overflow is triggered if all the bits in the SSI value from the feedback are zero.
- **Ones**
  A transducer overflow is triggered if all the bits in the SSI value from the feedback are ones.
- **Bit 21**
  A transducer overflow is triggered if bit 21 is 1.

Format Details

This section is primarily for addressing the SSI Overflow Mode parameter when communicating with the RMC from an external device. This information is not necessary when configuring the SSI Overflow Mode in RMCTools.

RMC75/150:

The SSI Overflow Mode is selected with bits 4-6 in the SSI/MDT Configuration Register. These bits correspond to the SSI Overflow Mode as shown in the following table:

Bit 6	Bit 5	Bit 4	SSI Format
0	0	0	None
0	0	1	Zeros
0	1	0	Ones
0	1	1	Bit 21

See the SSI/MDT Configuration Register for details about the register containing these bits.

RMC200:

The SSI Overflow Mode is selected with bits 0-2 in the SSI Data Configuration Register. These bits correspond to the SSI Overflow Mode as shown in the following table:

Bit 2	Bit 1	Bit 0	SSI Format
0	0	0	None
0	0	1	Zeros
0	1	0	Ones
0	1	1	Bit 21

See the SSI Data Configuration Register for details about the register containing these bits.

See Also

Parameter Registers | SSI/MDT Configuration Register | SSI Fundamentals

9.2.3.2.24.14. SSI High Bits to Ignore

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.61.4-7, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].MDTSSICfg.SSIDataMaskHi
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	DINT Bits - see below

Description

The **SSI High Bits to Ignore** parameter applies to RMC200 SSI inputs. Together with SSI Low Bits to Ignore, it is used to ignore bits in the SSI data.

This parameter is useful for SSI signals that, in addition to the feedback value, include other information such as parity or error bits, or temperature. In order for the axis input to use the feedback data, the bits containing the additional data must be ignored.

The axis' Raw Counts will become the value of the SSI Data bits that are not ignored.

The **SSI High Bits to Ignore** parameter ignores the specified number of high bits of the SSI data. This starts with the highest bit number as defined by the SSI Data Bits parameter.

The entire SSI data, including the ignored bits, is given in the Transducer Status B axis status parameter.

Format Details

The binary value of bits 4-7 defines how many of the high SSI bits to ignore.

See the SSI/MDT Configuration Register for details about the register containing these bits.

See Also
Parameter Registers | SSI/MDT Configuration Register | SSI Fundamentals | Homing

9.2.3.2.24.15. SSI Low Bits to Ignore

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.61.8-11, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].MDTSSICfg.SSIDataMaskLow
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	DINT Bits - see below

Description

The **SSI Low Bits to Ignore** parameter applies to RMC200 SSI inputs. Together with SSI High Bits to Ignore parameter, it is used to ignore bits in the SSI data.

This parameter is useful for SSI signals that, in addition to the feedback value, include other information such as parity or error bits, or temperature. In order for the axis input to use the feedback data, the bits containing the additional data must be ignored.

The axis' Raw Counts will become the value of the SSI Data bits that are not ignored.

The **SSI Low Bits to Ignore** parameter ignores the specified number of low bits of the SSI data. This starts with the lowest bit number of the SSI data.

The entire SSI data, including the ignored bits, is given in the Transducer Status B axis status parameter.

Format Details

The binary value of bits 8-11 defines how many of the low SSI bits to ignore.

See the SSI/MDT Configuration Register for details about the register containing these bits.

See Also
Parameter Registers | SSI/MDT Configuration Register | SSI Fundamentals | Homing

9.2.3.2.24.16. Exciter Mode

Type:	Axis Parameter Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** %MD*n*.60.0-3, where *n* = 384+ the axis number Primary Input: %MD*n*.60.0-3, where *n* = 384+ the axis number Secondary Input: %MD*n*.120.0-3, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].LoadCellCfg.ExciterMode Secondary Input: _Axis[*n*].SecLoadCellCfg.ExciterMode

	where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	Bits
Range:	Nominal (0), Fixed Value (1), Adaptive (2)
Default Value:	Nominal (0)

Description

The Exciter Mode parameter defines the exciter voltage value that is used in calculating the Millivolts/Volt input value of a load cell input. The two main use cases for this parameter are compensating for wire voltage drop or using an external excitation voltage.

Compensating for Wire Voltage Drop

The LC8 offers two methods of compensating for the voltage drop in the Exc+ and Exc- wires:

- **Adaptive:**

  The LC8 measures the voltage of the Sense pin and calculates the voltage drop. It does this every 18 ms or 9 loop times, whichever is greater. The calculated voltage at the load cell is given in the **Effective Exciter Voltage** status register. This method assumes the voltage drop of the Exc+ and Exc- wires is the same. Therefore, the wire length and gauge of the Exc+ and Exc- wires must be identical.

  To choose this option:

  1. Set the Exciter Mode axis parameter to **Adaptive**.

- **Fixed Value:**

  The user can manually enter the voltage at the load cell.

  To choose this option:

  1. Set the Exciter Mode axis parameter to **Fixed Value**.
  2. Use a voltmeter to measure the voltage of the Exc+ wire at the load cell relative to the Exc- wire at the load cell,

     or,

     For 6-wire load cells, if the Sense input is wired, optionally choose the value given by the **Effective Exciter Voltage** status register.
  3. Enter the measured voltage into the **Fixed Exciter Voltage** axis parameter.

Using an External Excitation Voltage

When using an external excitation voltage:

1. Set the Exciter Mode axis parameter to **Fixed Value**.
2. Enter the excitation voltage value into the **Fixed Exciter Voltage** axis parameter. For a more precise value, use a voltmeter to measure the voltage of the Exc+ wire at the load cell relative to the Exc- wire at the load cell, or, optionally, for 6-wire load cells with the Sense pin connected, use the value given by the **Effective Exciter Voltage** status register.

Format Details

This section is primarily for addressing the Exciter Mode parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

RMC200

Bit 3	Bit 2	Bit 1	Bit 0	Value	Exciter Mode
0	0	0	0	0	Nominal
0	0	0	1	1	Fixed Value

0	0	1	0	2	Adaptive

See the Load Cell Configuration Register topic for details about the register containing these bits.

See Also

Millivolts/Volt | Millivolts/Volt Offset | Fixed Exciter Voltage | Load Cell Scaling

---

9.2.3.2.24.17. Fixed Exciter Voltage

Type:	Axis Parameter Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.63, where *n* = 384+ the axis number Secondary Input: %MD*n*.123, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].FixedExciterVoltage Secondary Input: _Axis[*n*].SecFixedExciterVoltage where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	REAL
Units:	V
Range:	> 0 V
Default Value:	6.75 V

Description

The Fixed Exciter Voltage axis parameter applies when the Exciter Mode is set to **Fixed Value**.

The Fixed Value mode is one method of compensating for the wire drop for a load cell input. During commissioning, the user must enter the measured applied voltage at the load cell in the Fixed Exciter Voltage axis parameter. The measured voltage may be obtained from a voltmeter, or from the Effective Exciter Voltage axis status register.

When the Exciter Mode is set to **Fixed Value**, the Fixed Exciter Voltage parameter is used in the calculation the mV/V input of the load cell as follows:

Millivolts/Volt = Millivolt Input / **Fixed Exciter Voltage**

See Also

Load Cell Fundamentals | Exciter Mode | Load Cell Scaling

9.2.3.2.24.18. Load Cell Configuration Register

Type:	Axis Parameter Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.60, where *n* = 384+ the axis number

	Secondary Input: %MD*n*.120, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].LoadCellCfg Secondary Input: _Axis[*n*].SecLoadCellCfg where *n* is the axis number
How to Find:	See individual parameter registers listed below
Data Type:	DWORD

Description

The Load Cell Configuration Register contains the bit-addressable feedback configuration parameters for load cell inputs. This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Changing this Parameter Register

Because this parameter register affects motion, the axis must be disabled or in Direct Output before writing to this register. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Bits

This register contains the following parameters. The bits for each are given in the right-hand column.

RMC200

Parameter	Tag Name	Bit Number(s)
Exciter Mode	Exciter Mode	0-3

Note:
For details on the values that each bit represents, see the respective parameter topic.

See Also

Parameter Registers

9.2.3.2.24.19. Millivolts/Volts Offset

Type:	Axis Parameter Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.30, where *n* = 384+ the axis number Secondary Input: %MD*n*.90, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].MvPerVoltOffset
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	REAL
Units:	mV/V
Range:	Real Number

Default Value:	0 mV/V

Description

The Millivolts/Volt Offset parameter offsets the Millivolts/Volt input before it is converted to force units. This parameter is available only on load-cell input axes and is intended to provide an easy zero-signal adjustment for bipolar load cell feedback. The Negative Correction Factor parameter can also be useful for load cell signals.

For bipolar load cells (tension and compression), the Millivolts/Volt Offset is expected to be approximately 0, and the Negative Correction Factor close to 1.

The Millivolts/Volt Offset is used in the calculation of the Actual Force as follows:

If (Millivolts/Volt **+ Millivolts/Volt Offset**) < 0 Then

Force = (Millivolts/Volt + **Millivolts/Volt Offset**) x NegCorrFactor x Force Scale + Force Offset

Else

Force = (Millivolts/Volt + **Millivolts/Volt Offset**) x Force Scale + Force Offset

EndIf

See Also

Load Cell Fundamentals | Millivolt Input | Millivolts/Volt | Exciter Mode | Load Cell Scaling

---

9.2.3.2.24.20. Wire Break Detection

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** %MD*n*.10.19, where *n* = 24 + the axis number **RMC200:** %MD*n*.60.26-27, where *n* = 24 + the axis number
System Tag:	_Axis[*n*].MDTSSICfg.WBDetect
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	Bit - see below

Description

This parameter is used to turn off the SSI wire break detection. This should only be done if the SSI transducer or encoder you are using requires it. The SSI specification includes wire break detection, but some devices are not compliant. This option is available on the RMC in order to use non-compliant SSI devices. If an SSI wire break is detected, it will be reported as a No Transducer error.

This parameter is only available on the RMC150 SSI, RMC200 S8, and RMC200 U14 modules.

Format Details

This section is primarily for addressing the Wire Break Detection parameter when communicating with the RMC from an external device. This information is not necessary when configuring the Wire Break Detection in RMCTools.

The Wire Break Detection is selected with bit 19 in the SSI/MDT Configuration Register. This bit corresponds to the Wire Break Detection as shown in the following table:

RMC75/150

Bit 19	Wire Break Detection
0	Enabled

1	Disabled

RMC200

Bit 27	Bit 26	Wire Break Detection
0	0	Enabled
0	1	Disabled

See the SSI/MDT Configuration Register for details about the register containing these bits.

See Also

Parameter Registers | SSI/MDT Configuration Register | SSI Fundamentals

9.2.3.2.24.21. Load Cell Underflow Limit

Type:	Axis Parameter Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.62, where *n* = 384+ the axis number Secondary Input: %MD*n*.122, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].LoadCellLimitLow Secondary Input: _Axis[*n*].SecLoadCellLimitLow where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	REAL
Units:	mV/V
Range:	Real Number
Default Value:	-5.05 mV/V

Description

The Load Cell Overflow Limit and Load Cell Underflow Limit define the valid range of the Millivolt/Volt load cell input. If the Millivolt/Volt value is outside of this range, the Transducer Overflow bit will be set. If the Transducer Overflow error bit is set, a Halt will occur if the Transducer Overflow Auto Stop is configured to do so and the Direct Output Status bit is off.

If the feedback is exceeding the valid range due to acceptable noise, you can expand the range with these parameters.

The Load Cell Overflow Limit and Load Cell Underflow Limit may be set to values that will properly indicate an error if the load cell output indicates an unacceptably large load is being applied. Note that the Positive and Negative Force Limits do not have a corresponding error bit, so setting the Load Cell Overflow Limit and Load Cell Underflow Limit is a more certain method of halting the axis in the event of an unacceptably large load.

See Also

Load Cell Overflow Limit | Load Cell Fundamentals | Error Bits

9.2.3.2.24.22. Load Cell Overflow Limit

Type:	Axis Parameter Register
Address:	**RMC70:** n/a **RMC150:** n/a **RMC200:** Primary Input: %MD*n*.61, where *n* = 384+ the axis number Secondary Input: %MD*n*.121, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].LoadCellLimitHigh Secondary Input: _Axis[*n*].SecLoadCellLimitHigh where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	REAL
Units:	mV/V
Range:	Real Number
Default Value:	5.05 mV/V

Description

The Load Cell Overflow Limit and Load Cell Underflow Limit define the valid range of the Millivolt/Volt load cell input. If the Millivolt/Volt value is outside of this range, the Transducer Overflow bit will be set. If the Transducer Overflow error bit is set, a Halt will occur if the Transducer Overflow Auto Stop is configured to do so and the Direct Output Status bit is off.

If the feedback is exceeding the valid range due to acceptable noise, you can expand the range with these parameters.

The Load Cell Overflow Limit and Load Cell Underflow Limit may be set to values that will properly indicate an error if the load cell output indicates an unacceptably large load is being applied. Note that the Positive and Negative Force Limits do not have a corresponding error bit, so setting the Load Cell Overflow Limit and Load Cell Underflow Limit is a more certain method of halting the axis in the event of an unacceptably large load.

See Also

Load Cell Underflow Limit | Load Cell Fundamentals | Error Bits

9.2.3.2.24.23. Analog Input Type

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.10.0-2, where *n* = 12 + the axis number Secondary Input: %MD*n*.28.0-2, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.10.0-2, where *n* = 24 + the axis number Secondary Input: %MD*n*.28.0-2, where *n* = 24 + the axis number **RMC200:** Primary Input: %MD*n*.60.0-3, where *n* = 384 + the axis number Secondary Input: %MD*n*.120.0-3, where *n* = 384 + the axis number

System Tag:	Primary Input: _Axis[*n*].AnalogCfg.InputRange Secondary Input: _Axis[*n*].SecAnalogCfg.InputRange
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	Bits
Range:	±10V (0), 4-20mA (1), ±5V (2)
Default Value:	Voltage (±10V) (0)

Description

The Analog Input Type register specifies the type of analog feedback from the transducer:

- **±10V**
  If the feedback is voltage, choose this option.
- **±5V (RMC150 A and H modules only)**
  If you are using an input on the RMC150 A or H modules, and the analog input voltage does not exceed ±5V, choose this option for higher resolution.
- **4-20 mA**
  If the feedback is 4-20 mA, choose this option.

The input types of differential force or differential acceleration inputs must be identical and are determined by a single parameter.

Notice that voltage and current require different wiring.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you.

Format Details

This section is primarily for addressing the Analog Input Type parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

RMC75/150

Bit 2	Bit 1	Bit 0	Value	Control Mode
0	0	0	0	±10V
0	0	1	1	4-20mA
0	1	0	2	±5V (RMC150 Only)

RMC200

Bit 3	Bit 2	Bit 1	Bit 0	Value	Control Mode
0	0	0	0	0	±10V
0	0	0	1	1	4-20mA

See the Analog Configuration Register topic for details about the register containing these bits.

See Also

Analog Configuration Register

9.2.3.2.24.24. Analog Input Filter

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.60.4-7, where *n* = 384+ the axis number
System Tag:	Primary Input: _Axis[*n*].AnalogCfg.AnalogFilter Secondary Input: _Axis[*n*].SecAnalogCfg.AnalogFilter where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback or Secondary Feedback
Data Type:	bits
Default Value:	0 (4 kHz)

Description

The Analog Input Filter parameter applies to the RMC200 analog inputs and selects the cutoff frequency of the single-pole IIR digital filter that is applied in the analog input module. This analog input filter differs from the axis feedback filtering, which is applied after the feedback is received from the input module. The analog input filter applies directly to the signal, whereas the axis feedback filtering has multiple options such as selecting whether to filter the signal, the rate of change of the signal, and filtering just the displayed value versus the value used for control.

For best noise immunity, this filter should be set to the lowest frequency possible for the application. When considering filtering, this filter should always be set before setting the axis feedback filtering.

Format Details

This section is primarily for addressing the Analog Input Type parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

RMC200

Bit 7	Bit 6	Bit 5	Bit 4	Value	Filter
0	0	0	0	0	4 kHz
0	0	0	1	1	2 kHz
0	0	1	0	2	1 kHz
0	0	1	1	3	500 Hz
0	1	0	0	4	250 Hz

See the Analog Configuration Register topic for details about the register containing these bits.

See Also

Analog Configuration Register

9.2.3.2.24.25. Analog Overflow Limit

Type:	Axis Parameter Register
Address:	**RMC75:** n/a

	RMC150: n/a **RMC200:**
	Primary Feedback: %MD*n*.61, where *n* = 384+ the axis number
	Secondary Feedback: %MD*n*.121, where *n* = 384+ the axis number
System Tag:	Primary Feedback: _Axis[*n*].PriAnaLimitHigh
	Secondary Feedback: _Axis[*n*].SecAnaLimitHigh, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback or Secondary Feedback
Data Type:	REAL
Units:	V or mA
Range:	-25 to +25 (see below)
Default Value:	Voltage: 10.1 V
	Current: 21 mA

Description

The Analog Overflow Limit and Analog Underflow Limit parameters define the valid range of analog input. If the analog input exceeds this range, the Transducer Overflow error bit will be set. If the Transducer Overflow error bit is set, a Halt will occur if the Transducer Overflow Auto Stop is configured to do so and the Direct Output Status bit is off.

If the feedback is exceeding the valid range due to acceptable noise, you can expand the range with these parameters.

The Analog Overflow Limit and Analog Underflow limits should be set to values within the maximum and minimum valid feedback ranges:

Module	Feedback Type	Max Value	Min Values
RMC200 A8	4-20 mA	25.0	-25.0
	±10 V	10.5	-10.5
RMC200 U14	4-20 mA	25.0	-25.0
	±10 V	10.2	-10.2

See Also

Analog Underflow Limit | Transducer Overflow Error Bit

9.2.3.2.24.26. Analog Underflow Limit

Type:	Axis Parameter Register
Address:	**RMC75:** n/a
	RMC150: n/a
	RMC200:
	Primary Feedback: %MD*n*.62, where *n* = 384+ the axis number
	Secondary Feedback: %MD*n*.122, where *n* = 384+ the axis number
System Tag:	Primary Feedback: _Axis[*n*].PriAnaLimitLow
	Secondary Feedback: _Axis[*n*].SecAnaLimitLow, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback or Secondary Feedback

Data Type:	REAL
Units:	V or mA
Range:	-25 to +25 (see below)
Default Value:	Voltage: -10.1 V Current: 3.6 mA

Description

The Analog Underflow Limit and Analog Overflow Limit parameters define the valid range of analog input. If the analog input exceeds this range, the Transducer Overflow error bit will be set. If the Transducer Overflow error bit is set, a Halt will occur if the Transducer Overflow Auto Stop is configured to do so and the Direct Output Status bit is off.

If the feedback is exceeding the valid range due to acceptable noise, you can expand the range with these parameters.

The Analog Overflow Limit and Analog Underflow limits should be set to values within the maximum and minimum valid feedback ranges:

Module	Feedback Type	Max Value	Min Values
RMC200 A8	4-20 mA	25.0	-25.0
	±10 V	10.5	-10.5
RMC200 U14	4-20 mA	25.0	-25.0
	±10 V	10.2	-10.2

See Also

Analog Overflow Limit | Transducer Overflow Error Bit

9.2.3.2.24.27. Input Termination Parameter

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** %MD*n.*10, bit 1, where *n* = 24+ the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].QuadCfg.Term, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	boolean
Range:	**UI/O module:** none (0), ±A and ±B (1) **QA module:** none (1), ±A and ±B (0)
Default Value:	**UI/O module:** none (0) **QA module:** ±A and ±B (0)

Description

The Input Termination parameter applies to axes with quadrature feedback on the RMC150 Universal I/O (UI/O) Module or RMC75 QA Module. These modules have internal software-selectable termination.

Input termination should always be applied when connecting a single encoder directly to a single quadrature input. For daisy-chained quadrature feedback (connecting one encoder to multiple inputs), only the last input in the chain should have termination.

See the Quadrature Configuration Register for details about the register containing these bits.

See Also

Parameter Registers | Universal I/O Module | QA Module (RMC75) | Quadrature Fundamentals | AB Termination

9.2.3.2.24.28. Quadrature Counter Mode Register

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.60.0-2, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].QuadCfg.CountMode, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	Bits - see below
Range:	A-Quad-B (0), Pulse A Rising (1), Pulse A Falling (2), Pulse A Both (3)
Default Value:	A-Quad-B (0)

Description

The Quadrature Counter Mode applies to the RMC200 Q4, U14 and D24 modules and defines the method by which counts are determined from the A and B inputs. It can be set for a typical quadrature counter, or a pulse counter. The following options are available:

- **A-Quad-B**
  The is the typical quadrature encoder counting method. The counts increment on each transition of A or B, and the order of the transition defines the direction. This option should be used with a quadrature A and B encoder.
- **Pulse A Rising**
  The counts increment each time a rising edge occurs on input A.
- **Pulse A Falling**
  The counts increment each time a falling edge occurs on input A.
- **Pulse A Both**
  The counts increment each time a rising edge or a falling edge occurs on input A.

For the pulse options, input B is not used. The pulse counter options cannot detect direction of motion. Therefore, a pulse counter can be useful for tracking the velocity of a machine component that is know to always move in only one direction, such as a feed belt.

Format Details

This section is primarily for addressing this parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

The Quadrature Counter Mode is selected with bits 0-2 in the Quadrature Configuration Register. These bits correspond to the Quadrature Counter Mode as shown in the table below.

RMC200

Bit 2	Bit 1	Bit 0	Value	
0	0	0	0	A-Quad-B
0	0	1	1	Pulse A Rising
0	1	0	2	Pulse A Falling
0	1	1	3	Pulse A Both

See Also

Quadrature Configuration Register

9.2.3.2.24.29. Quadrature AB Input Type

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.60.4-6, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].QuadCfg.ABInputType, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	Bits - see below
Range:	RS-422 (0), TTL (1), Diff HTL (2), SE HTL (3)
Default Value:	RS-422 (0)

Description

The Quadrature AB Input Type parameter applies to the RMC200 Q4 and U14 modules and defines the signal type for the A and B signals with various options for voltage levels and single-ended or differential. Choose the option that corresponds to your encoder. Delta always recommends quadrature encoders with RS-422 outputs due to high noise immunity and speed performance. The Index (Z) input type is separately set with the parameter Quadrature Z Input Type.

The following Quadrature AB Input Type options are available:

- **RS-422**
- **TTL**
- **Differential HTL**
- **Single-ended HTL**

See the Q4 or U14 modules for details on these input types.

Format Details

This section is primarily for addressing this parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

The Quadrature AB Input Type is selected with bits 4-6 in the Quadrature Configuration Register. These bits correspond to the Quadrature AB Input Type as shown in the table below.

Bit 6	Bit 5	Bit 4	Value	

0	0	0	0	RS-422
0	0	1	1	TTL
0	1	0	2	Differential HTL
0	1	1	3	Single-ended HTL

See Also

Quadrature Configuration Register

9.2.3.2.24.30. Quadrature AB Termination

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.60.7, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].QuadCfg.ABTerm, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	boolean
Range:	Disabled (0), Enabled (1)
Default Value:	Enabled (1)

Description

The AB Termination parameter applies to axes with quadrature feedback on the RMC200 Q4, U14 and S8 modules. These modules have internal software-selectable termination. Termination for the Z input may be set separately via the Z Termination parameter.

On the Q4 and U14 modules, the AB Termination parameter is available only if the AB Input Type is RS-422 or TTL.

Input termination should always be applied when connecting an RS-422 single encoder directly to a single quadrature input. For daisy-chained quadrature feedback (connecting one encoder to multiple inputs), only the last input in the chain should have termination.

See the Quadrature Configuration Register for details about the register containing this parameter.

See Also

Quadrature Configuration Register | Quadrature AB Input Type | Z Termination

9.2.3.2.24.31. Quadrature Z Input Type

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.60.10-12, where *n* = 384+ the axis number

System Tag:	_Axis[*n*].QuadCfg.ZInputType, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	Bits - see below
Range:	RS-422 (0), TTL (1), Diff HTL (2), SE HTL (3), DI (4)
Default Value:	RS-422 (0)

Description

The Quadrature Z Input Type parameter applies to the RMC200 Q4 module and U14 module and defines the signal type for the Z input with various options for voltage levels and single-ended or differential. Choose the option that corresponds to your encoder. Delta always recommends quadrature encoders with RS-422 outputs due to high noise immunity and speed performance. The A and B input type is separately set with the parameter Quadrature AB Input Type.

The following Quadrature AB Input Type options are available:

- **RS-422**
- **TTL**
- **Differential HTL**
- **Single-ended HTL**
- **DI**

See the Q4 Module and U14 module for details on these input types.

Format Details

This section is primarily for addressing this parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

The Quadrature Z Input Type is selected with bits 10-12 in the Quadrature Configuration Register. These bits correspond to the Quadrature Z Input Type as shown in the table below.

Bit 12	Bit 11	Bit 10	Value	
0	0	0	0	RS-422
0	0	1	1	TTL
0	1	0	2	Differential HTL
0	1	1	3	Single-ended HTL
1	0	0	4	DI

See Also

Quadrature Configuration Register| Quadrature AB Input Type

9.2.3.2.24.32. Quadrature Z Termination

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.60.13, where *n* = 384+ the axis number

System Tag:	_Axis[*n*].QuadCfg.ZTerm, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	boolean
Range:	Disabled (0), Enabled (1)
Default Value:	Enabled (1)

Description

The Z Termination parameter applies to axes with quadrature feedback on the RMC200 Q4 and U14 modules. These modules have internal software-selectable termination. Termination for the A and B inputs may be set separately via the AB Termination parameter.

The Z Termination parameter is available only if the Z Input Type is RS-422 or TTL.

Input termination should always be applied when connecting an RS-422 signal directly to a single Z input. For daisy-chained quadrature feedback (connecting one encoder to multiple inputs), only the last input in the chain should have termination.

See the Quadrature Configuration Register for details about the register containing this parameter.

See Also

Quadrature Configuration Register | Quadrature Z Input Type | Quadrature AB Termination

9.2.3.2.24.33. Quadrature H Input Type

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.60.16, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].QuadCfg.HInputType, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	boolean
Range:	TTL (0), DI (1)
Default Value:	DI (1)

Description

The Quadrature H Input Type parameter applies to the RMC200 Q4 module and defines the signal type for the home input signal with various options for voltage levels. Choose the option that corresponds to your home input signal.

The following Quadrature H Input Type options are available:

- **TTL**
- **DI**

See the Q4 Module for details on these input types.

See Also

Quadrature Configuration Register

9.2.3.2.24.34. Quadrature R Input Type

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.60.17, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].QuadCfg.RInputType, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	boolean
Range:	5 V (0), 12-24 V (1)
Default Value:	12-24 V (1)

Description

The Quadrature R Input Type parameter applies to the RMC200 Q4 module and defines the signal type for the registration input signal with various options for voltage levels. Choose the option that corresponds to your registration input signal.

The following Quadrature H Input Type options are available:

- **5V**
- **12-24 V**

See the Q4 Module for details on these input types.

See Also

Quadrature Configuration Register

9.2.3.2.24.35. Quadrature HTL Threshold

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.60.18, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].QuadCfg.HTLLevel, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	boolean
Range:	7 V (0), 12 V (1)
Default Value:	7 V (0)

Description

The Quadrature HTL Threshold parameter applies to the RMC200 Q4 module and U14 Module and defines the threshold for the Single-ended HTL signal type as selected by the AB Input Type or Z Input Type parameters. Choose the option that corresponds to the voltage level of your A and B or Z signal.

The following Quadrature HTL Threshold options are available:

- **7V**
  Suitable for 12V signal levels.

- **12V**
  Suitable for 24V signal levels.

See the Q4 or U14 modules for details on these input types.

See Also

Quadrature Configuration Register

9.2.3.2.24.36. Index (Z) Home Location

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.10.0, where *n* = 12 + the axis number **RMC150:** %MD*n*.10.0, where *n* = 24 + the axis number **RMC200:** %MD*n*.60.14, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].QuadCfg.ZLocation
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	Bit
Range:	A Leading (0), A Trailing (1)
Default Value:	A Leading (0)

Description

This parameter is valid for axes with quadrature inputs that include the Z (Index) signal. The Z Home Location parameter specifies the edge of the quadrature A signal on which a Z home is triggered. The Z Home Location parameter is important for accurate directional homing with the Index (Z) pulse on a quadrature encoder. Once the Z Home Location has been correctly set, you can home an axis with the Z pulse in either direction of motion and it will set the home position at the same count position each time. This type of homing is the most accurate homing method. See the Homing topic for details.

Use the Learn Z Alignment (54) command to set the Index Z Home Location parameter.

Manually Setting the Index (Z) Home Location

The Learn Z Alignment (54) command is the easiest way to set the Z Home Location parameter. If you wish to set this parameter manually instead, read this section for details on the Z Home Location options.

The Z Home Location parameter can be set to the following:

- **A Leading (0)**
- **A Trailing (1)**

Choosing the Index (Z) Home Location

Choosing the Index (Z) Home Location option involves determining the state of the B input at the leading edge of Z. 'Leading' is defined as the edge that will be rising when the encoder is moving in the direction of increasing counts and falling when in the direction of decreasing counts.

- If the B signal is high at the leading edge of Z, or has a transition at the leading edge of Z, choose **A Leading**.
- If the B signal is low at the leading edge of Z, then choose **A Trailing**.

A Leading

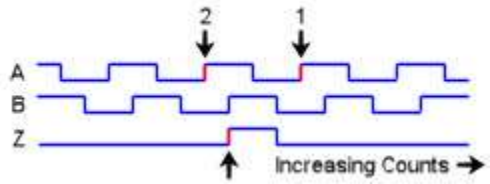
The home is triggered on the leading edge of the A pulse after the leading edge of the Z pulse. 'Leading' is defined as the edge that will be rising when the encoder is moving in the direction of

increasing counts and falling when in the direction of decreasing counts. The leading edge of Z is indicated by the arrow in the figure below.

Therefore, when moving in the direction of *increasing* counts, the home is triggered on the next rising edge of the A pulse after the rising edge Z pulse, as indicated by arrow number **1** in the figure below.

When moving in the direction of *decreasing* counts, the home is triggered on the leading edge (as seen in the direction of increasing counts) of the A pulse after the leading edge Z pulse. Since the axis is moving in the direction of decreasing counts, it will trigger on the falling edge indicated by arrow **2** in the figure below.

Notice that when the home is triggered, the RMC will automatically adjust the home position for the direction so that the Home Position is set at the same count, regardless of direction.



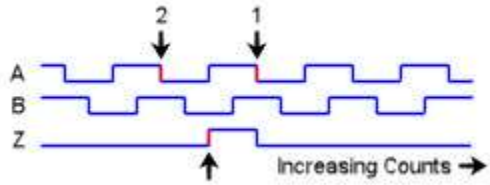
A Trailing

The home is triggered on the trailing edge of the A pulse after the leading edge of the Z pulse. 'Leading' is defined as the edge that will be rising when the encoder is moving in the direction of increasing counts and falling when in the direction of decreasing counts. The leading edge of Z is indicated by the arrow in the figure below.

Therefore, when moving in the direction of *increasing* counts, the home is triggered on the next falling edge of the A pulse after the rising edge Z pulse, as indicated by arrow number **1** in the figure below.

When moving in the direction of *decreasing* counts, the home is triggered on the trailing edge (as seen in the direction of increasing counts) of the A pulse after the leading edge Z pulse. Since the axis is moving in the direction of decreasing counts, it will trigger on the rising edge indicated by arrow **2** in the figure below.

Notice that when the home is triggered, the RMC will automatically adjust the home position for the direction so that the Home Position is set at the same count, regardless of direction.



See Also

Parameter Registers | Quadrature Configuration Register | Learning Z Alignment Status Bit | Learn Z Alignment (54) Command

9.2.3.2.24.37. Filter Reg Input Parameter

Type:	Axis Parameter Register
Address:	**RMC75:** n/a

	RMC150: %MD*n*.10/2, where *n* = 24+ the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].QuadCfg.RFilt, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Feedback
Data Type:	boolean
Range:	off (0), on (1)
Default Value:	off (0)

Description

The Filter Reg Input parameter applies to axes with quadrature feedback on an RMC150 Universal I/O (UI/O) Module. All the discrete inputs on the UI/O module—including the Reg 0 and Reg 1 inputs—are filtered, which increases the time delay of the filter, and hence, the registration.

This parameter gives the option of disabling the filtering on the Reg input associated with the axis to decrease the response time of the registration to 150µsec. With the filtering applied, the response time of the registration is 220µsec.

See the Quadrature Configuration Register for details about the register containing this bit.

See Also

Parameter Registers | Universal I/O Module | Quadrature Fundamentals

9.2.3.2.24.38. Resolver Resolution

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** %MD*n*.10.0-1, where *n* = 24+ the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].ResCfg.Resolution, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup Tab: Primary Control Setup
Data Type:	Bits - see below

Description

This parameter is valid only on axes with Resolver feedback. It defines the resolution of the resolver input. The following resolutions are available:

- 14 bits
- 16 bits

The resolution of the resolver input affects the maximum speed and acceleration of the resolver. If the resolver exceeds the maximum speed or acceleration, a No Transducer error will occur. The following table lists that maximum speeds and accelerations:

	14 Bits	16 Bits
Maximum Speed	3000 RPM	600 RPM
Maximum Acceleration	1200 RPS per second	60 RPS per second

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Format Details

This section is primarily for addressing the Resolver Resolution parameter when communicating with the RMC from an external device. This information is not necessary when configuring the Resolver Resolution in RMCTools.

The Resolver Resolution is selected with bits 0 and 1 in the Resolver Configuration register. These bits correspond to the Resolver Resolution as shown in the following table:

Bit 1	Bit 0	Value	Resolution
0	0	0	14 bits
0	1	1	16 bits

See the Resolver Configuration Register for details about the register containing these bits.

See Also

Resolver Configuration Register

9.2.3.2.24.39. Reference Amplitude

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** %MD*n*.15, where *n* = 24+ the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].RefAmp, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	V RMS
Range:	1.42 to 4.80
Default Value:	2V

Description

This parameter applies only to the Resolver (R) module. This parameter specifies the RMS amplitude of the reference signal transmitted by the Resolver (R) module to the resolver device.

Note: This parameter does not apply to the RW module.

Changing this Parameter

To change this parameter, click the cell, then click the Ellipsis button.

Your resolver specification sheet should indicate either a Transformation Ratio or the Input and Output Voltages. Use the radio button to select the type of information you will use to configure the interface and then enter the appropriate information in the appropriate edit boxes.

Notice that the RMC expects a 2 Vrms signal from the resolver, and this may not match up with your resolver's specification. This is generally OK. RMCTools will use the information you supplied to calculate the correct reference output voltage to obtain 2 Vrms back from the resolver. Some

resolvers are specified for operation at 11 or 26 Volts. These resolvers will generally still work well at 2 Volts, but there is some potential loss of accuracy and the signal-to-noise ratio will be reduced. Contact Delta Computer Systems, Inc. to discuss options for your application.

Both Resolver interfaces on a Resolver (R) module must always be set to the identical Reference Amplitude. Changing the Reference Amplitude on one input will automatically change it on the other input.

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Reference Frequency | Resolver (R) Module

9.2.3.2.24.40. Reference Frequency

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** %MD*n*.14, where *n* = 24+ the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].RefFreq, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	Hz
Range:	800-5000
Default Value:	2500

Description

This parameter applies only to the Resolver (R) module. This parameter specifies the frequency of the reference signal transmitted by the Resolver (R) module to the resolver device. This value should be set to the frequency specified by the datasheet of the resolver device.

Note: This parameter does not apply to the RW module.

Changing this Parameter

Both Resolver interfaces on a Resolver (R) module must always set to the identical Reference Frequency. Changing the Reference Frequency on one input will automatically change it on the other input.

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

See Also

Reference Amplitude | Resolver (R) Module

9.2.3.2.24.41. Analog Feedback Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.10, where *n* = 12 + the axis number Secondary Input: %MD*n*.28, where *n* = 12 + the axis number **RMC150:** Primary Input: %MD*n*.10, where *n* = 24 + the axis number Secondary Input: %MD*n*.28, where *n* = 24 + the axis number **RMC200:** Primary Input: %MD*n*.60, where *n* = 384 + the axis number Secondary Input: %MD*n*.120, where *n* = 384 + the axis number
System Tag:	Primary Input: _Axis[*n*].AnalogCfg Secondary Input: _Axis[*n*].SecAnalogCfg where *n* is the axis number
How to Find:	See individual parameter registers listed below
Data Type:	DWORD - see below

Description

The Analog Feedback Register contains the bit-addressable feedback configuration parameters for analog inputs. This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Changing this Parameter Register

Because this parameter register affects motion, the axis must be disabled or in Direct Output before writing to this register. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Bits

This register contains the following parameters. The bits for each are given in the right-hand column.

RMC75/150

Parameter	Tag Name	Bit Number(s)
Analog Input Type	InputRange	0-2

RMC200

Parameter	Tag Name	Bit Number(s)
Analog Input Type	InputRange	0-3
Analog Input Filter	AnalogFilter	4-7

Note:
For details on the values that each bit represents, see the respective parameter topic.

See Also

Parameter Registers

9.2.3.2.24.42. SSI/MDT Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.10, where *n* = 12 + the axis number **RMC150:** %MD*n*.10, where *n* = 24 + the axis number **RMC200:** %MD*n*.60, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].MDTSSICfg, where *n* is the axis number
How to Find:	See individual parameters listed below
Data Type:	DWORD

Description

The SSI/MDT Configuration Register contains the bit-addressable feedback configuration parameters for MDT and SSI inputs. The RMC200 includes a SSI Data Configuration register with additional SSI parameters.

This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Changing this Parameter Register

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this register. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Parameters

This register contains the following parameters.

RMC75/150:

Not all bits are available for all modules. For details on the values that each bit represents, see the respective parameter topic.

Parameter	Tag Names	Bits	RMC75 MA	RMC150 MDT	RMC150 SSI	RMC150 UI/O
Feedback Type	SSI	8	✓			
MDT Type	MDTType	0-2	✓	✓		
MDT Blanking Period	MDTBP	3		✓		
SSI Termination	SSITerm	0				✓
SSI Overflow Mode	SSIOverflowMode	4-6	✓		✓	✓
SSI Format	GrayCode	9	✓		✓	✓
SSI Data Bits	SSIDataBits	12-17	✓		✓	✓
SSI Clock Rate	SSIClockRate	10-11	✓		✓	✓
Absolute/Incremental	Inc	18	✓		✓	✓
Wire Break Detection	WBDetect	19			✓	
SSI Home Source	SSIHomeSrc	20-23	✓			

RMC200:

In addition to the items listed below, more SSI parameters are contained in the SSI Data Configuration register, such as SSI Overflow Mode, SSI High Bits to Ignore, and SSI Low Bits to Ignore.

Parameter	Tag Names	Bits
Feedback Type	SSI	0-1
MDT Type	MDTType	2-4
MDT Blanking Period	MDTBP	5-6
SSI Termination	SSITerm	12-13
SSI Format	GrayCode	14-15
SSI Clock Rate	SSIClockRate	16-19
SSI Data Bits	SSIDataBits	20-25
Wire Break Detection	WBDetect	26-27
Absolute/Incremental	Inc	28

See Also

SSI Data Configuration | MA Module (RMC75) | MDT Module (RMC150) | SSI Module (RMC150) | U14 Module (RMC200) | S8 Module (RMC200)

9.2.3.2.24.43. SSI Data Configuration

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.61, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].SSIDataCfg, where *n* is the axis number
How to Find:	See individual parameters listed below
Data Type:	DWORD

Description

The SSI Data Configuration Register contains bit-addressable feedback configuration parameters for RMC200 SSI inputs. Additional bit-addressable parameters are provided by the SSI/MDT Configuration Register.

This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Changing this Parameter Register

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this register. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Parameters

This register contains the following parameters.

RMC200:

For details on the values that each bit represents, see the respective parameter topic.

Parameter	Tag Names	Bit No(s)
SSI Overflow Mode	SSIOverflowMode	0-2
SSI High Bits to Ignore	SSIDataMaskHi	4-7
SSI Low Bits to Ignore	SSIDataMaskLow	8-11

See Also

SSI/MDT Configuration Register | S8 Module | U14 Module

9.2.3.2.24.44. Quadrature Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.10, where *n* = 12 + the axis number **RMC150:** %MD*n*.10, where *n* = 24 + the axis number **RMC200:** %MD*n*.60, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].QuadCfg, where *n* is the axis number
How to Find:	See individual parameters listed below
Data Type:	DWORD - see below

Description

The Quadrature Configuration Register contains the bit-addressable feedback configuration parameters for axes with a quadrature input from any of the following modules:

- RMC75 QA
- RMC150 Quadrature and Universal I/O
- RMC200 Q4, S8, U14 and D24

This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Bits

This register contains the following parameters. The bits for each are given in the second column. Not all bits are available for all modules. For details on the values that each bit represents, see the respective parameter topic.

RMC75/150

Parameter	Tag Names	Bit No	RMC75 QA	RMC150 QUAD	RMC150 UI/O
Index (Z) Home Location	ZLocation	0	✓	✓	
Input Termination	Term	1	✓		✓
Filter Reg Input	RFilt	2			✓

RMC200

Parameter	Tag Names	Bit No(s)	RMC200 Q4	RMC200 S8	RMC200 U14	RMC200 D24

Counter Mode	CountMode	0-2	✓		✓	✓
AB Input Type	ABInputType	4-6	✓		✓	
AB Termination	ABTerm	7	✓	✓	✓	
Z Input Type	ZInputType	10-12	✓		✓	
Z Termination	ZTerm	13	✓		✓	
Index (Z) Home Location	ZLocation	14	✓		✓	✓
H Input Type	HInputType	16	✓			
R Input Type	RInputType	17	✓			
HTL Threshold	HTLLevel	18	✓		✓	

See Also

Parameter Registers | Quadrature Module (RMC150) | QA Module (RMC75) | Q1 Module (RMC75) | Q4 Module (RMC200) | S8 Module (RMC200) | U14 Module (RMC200) | D24 Module (RMC200)

9.2.3.2.24.45. Resolver Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** %MD*n*.10, where *n* = 24+ the axis number **RMC150:** n/a
System Tag:	_Axis[*n*].ResCfg, where *n* is the axis number
How to Find:	See individual parameters listed below.
Data Type:	DWORD - see below

Description

The Resolver Configuration Register contains the bit-addressable feedback configuration parameters for the Resolver (R) module.

This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Changing this Parameter Register

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this register. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Bits

This register contains the following parameters. The bits for each are given in the second column. Not all bits are available for all modules. For details on the values that each bit represents, see the respective parameter topic.

Parameter	Tag Name	Bit Number(s)
Resolver Resolution	Resolution	0-1
Absolute/Incremental	Inc	18

See Also

Resolver (R) Module

9.2.3.2.25. Custom
9.2.3.2.25.1. Custom Feedback Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Input: %MD*n*.8, where *n* = 12+ the axis number Secondary Input: %MD*n*.26, where *n* = 12+ the axis number **RMC150:** Primary Input: %MD*n*.8, where *n* = 24+ the axis number Secondary Input: %MD*n*.26, where *n* = 24+ the axis number **RMC200:** Primary Input: %MD*n*.59, where *n* = 384+ the axis number Secondary Input: %MD*n*.119, where *n* = 384+ the axis number
System Tag:	**Primary Input:** _Axis[*n*].CustomFBConfig **Secondary Input:** _Axis[*n*].SecCustomFBConfig where *n* is the axis number
How to Find:	See individual parameters listed below
Data Type:	DWORD - see below

Description

The Custom Feedback Configuration Register contains the bit-addressable Custom Feedback configuration parameters. This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Tag Names

This register contains the following parameters. The tag names and bits for each are given below.

Parameter	Tag	Bit(s)
Custom Feedback Auto-Fault Mode	AutoFault	0-3

For details on the values that each bit represents, see the respective parameter topic.

See Also

Custom Feedback | Custom Feedback Auto-Fault Mode

9.2.3.2.25.2. Custom Feedback Auto-Fault Mode

Type:	Axis Parameter Register Bit Parameter
Address:	**RMC75:** Primary Input: %MD*n*.8/0-3, where *n* = 12+ the axis number Secondary Input: %MD*n*.26/0-3, where *n* = 12+ the axis number **RMC150:** Primary Input: %MD*n*.8/0-3, where *n* = 24+ the axis number Secondary Input: %MD*n*.26/0-3, where *n* = 24+ the axis number **RMC200:** Primary Input: %MD*n*.59/0-3, where *n* = 24+ the axis number Secondary Input: %MD*n*.119/0-3, where *n* = 24+ the axis number
System Tag:	**Secondary Input:** _Axis[*n*].CustomFBConfig.AutoFault **Primary Input:** _Axis[*n*].SecCustomFBConfig.AutoFault where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Feedback
Data Type:	Bits - see below
Range:	Missed Update (0), PROGRAM Mode (1), Disabled (2)
Default Value:	Missed Update (0)

Description

This parameter is valid only on axes with custom feedback on the primary and/or secondary input. This parameter provides options to automatically set the No Transducer or Prs/Frc No Transducer error bit. The following options are available:

- **Missed Update**

  If the Custom Counts register is not written to in a given loop time, the No Transducer error will be set. This is the preferred mode for most applications. In this mode, you should make sure that the Custom Counts register is written to each loop time.

- **PROGRAM Mode**

  If the controller is in PROGRAM mode, the No Transducer error will be set. This mode may be useful in situations where you do not need to update the Custom Counts very often, but you do wish to halt the axis if the program that updates the Custom Counts is stopped due to the controller entering PROGRAM mode.

- **Disabled**

  The No Transducer error will not be set by any of the above options. In this mode, the updating of the Custom Counts register is not monitored in any way. This could be used in cases where the Custom Counts register is being modified from an external device, such as a PLC.

The No Transducer Error bit can still be set by using the No Transducer and Secondary No Transducer bits in the Custom Error Bits register. Additionally, if the Custom Counts is NaN, INf, -Inf, the No Transducer error bit will also be set.

Format Details

The Custom Feedback Auto-Fault Mode is selected with bits 0-3. These bits correspond to the Auto-Fault Mode as shown in the following table:

Bit 3	Bit 2	Bit 1	Bit 0	Value	Auto-Fault Mode
0	0	0	0	0	Missed Update
0	0	0	1	1	PROGRAM Mode
0	0	1	0	2	Disabled

See the Custom Feedback Configuration Register for details about the register containing these bits.

See Also

Custom Feedback | Custom Feedback Configuration Register

9.2.3.3. Simulator

9.2.3.3.1. Simulate Mode

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.116/0, where *n* = 12 + the axis number **RMC150:** %MD*n*.116/0, where *n* = 24 + the axis number **RMC200:** %MD*n*.480, where *n* = 384 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].SimulationBits.Simulate **RMC200:** _Axis[*n*].SimMode
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	**RMC75/150:** BOOL **RMC200:** DINT
Default Value:	**RMC75/150:** False (off) **RMC200:** 0 (off)

Description

This parameter puts the axis in simulate mode. See the Simulating Motion topic for details.

If any axis is in simulate mode, the status of this parameter is displayed in the Axis Status Registers Pane to clearly indicate that to the operator.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you.

Format Details

For the RMC75/150, see the Simulator Configuration Register for details about addressing this bit.

For the RMC200, a DINT value of 0 is off, and a value of 1 is on.

See Also

Simulating Motion | Simulation Register

9.2.3.3.2. System Gain (Simulator)
9.2.3.3.3. Positive System Gain (Simulator)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.117, where *n* = 12 + the axis number **RMC150:** %MD*n*.117, where *n* = 24 + the axis number **RMC200:** %MD*n*.482, where *n* = 384 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].SystemGain, where *n* is the axis number **RMC200:** _Axis[*n*].SimPosGain, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	**RMC75/150:** (pu/s)/V **RMC200:** (pu/s)/%
Range:	> 0
Default Value:	**RMC75/150:** 1 **RMC200:** 0.1

Description

Note: The units differ between the RMC75/150 and RMC200, since the RMC75/150 Control Output is voltage, and the RMC200 Control Output is percent.

This parameter is used for the motion simulator in the RMC. It is used with both 1st order and 2nd order simulators. For details on using simulate mode, see the Simulate Mode topic.

For the RMC75/150, this parameter defines the gain in both directions. For the RMC200, this parameter defines the gain in the positive direction.

The System Gain is pu/s/V, in other words, how fast the system moves for 1 V of control output. To find this value, give 1 volt of Control Output (use the Open Loop Rate (10) command). Use the plot to find the Actual Velocity of the system. This is your System Gain.

Changing this Parameter

For best results, always change this parameter only from Axis Tools. If the axis is in Simulate mode, and this parameter is changed, simulate model will be suspended until the last simulate mode parameter register (Max Compression) has been updated. At this point, the new simulator model will be calculated and the simulator model will be activated. When the simulator model is suspended, the Actual Position will not move.

Certain simulator settings may result in an invalid model. If this occurs, change your settings. See the Simulating Motion topic for more details.

See Also

Simulating Motion | Negative System Gain

9.2.3.3.4. Negative System Gain (Simulator)

Type:	Axis Parameter Register
Address:	**RMC75/150:** n/a

	RMC200: %MD*n*.483, where *n* = 384 + the axis number
System Tag:	**RMC70/150:** n/a **RMC200:** _Axis[*n*].SimNegGain, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	(pu/s)/%
Range:	> 0
Default Value:	0.1

Description

This parameter is used for the motion simulator in the RMC200. It is used with both 1st order and 2nd order simulators. For details on using simulate mode, see the Simulate Mode topic.

The Negative System Gain is pu/s/%, in other words, how fast the system moves in the negative direction for 1% of control output. To find this value, give a significant amount of negative Control Output such as -20% (use the Open Loop Rate (10) command). Use the plot to find the Actual Velocity. Divide the Actual Velocity by the amount of Control Output. This is your Negative System Gain.

Changing this Parameter

For best results, always change this parameter only from Axis Tools. If the axis is in Simulate mode, and this parameter is changed, simulate model will be suspended until the last simulate mode parameter register (Max Compression) has been updated. At this point, the new simulator model will be calculated and the simulator model will be activated. When the simulator model is suspended, the Actual Position will not move.

Certain simulator settings may result in an invalid model. If this occurs, change your settings. See the Simulating Motion topic for more details.

See Also

Simulating Motion | Positive System Gain

9.2.3.3.5. Time Constant (Simulator)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*:118, where *n* = 12 + the axis number **RMC150:** %MD*n*:118, where *n* = 24 + the axis number **RMC200:** %MD*n*:484, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].TimeConstant, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	s
Range:	See the **Range** section below.
Default Value:	0.75

Description

This parameter is used for the motion simulator in the RMC. For details on using simulate mode, see the Simulate Mode topic.

The Time Constant is used with a 1st order simulator.

Range

The Time Constant must be greater than or equal to the control loop time:

Control Loop Time	Minimum Time Constant
0.125 ms	0.000125 s
0.25 ms	0.00025 s
0.5 ms	0.0005 s
1.0 ms	0.001 s
2.0 ms	0.002 s
4.0 ms	0.004 s

Changing this Parameter

For best results, always change this parameter only from Axis Tools. If the axis is in Simulate mode, and this parameter is changed, simulate model will be suspended until the last simulate mode parameter register (Max Compression) has been updated. At this point, the new simulator model will be calculated and the simulator model will be activated. When the simulator model is suspended, the Actual Position will not move.

Certain simulator settings may result in an invalid model. If this occurs, change your settings. See the Simulating Motion topic for more details.

See Also

Simulating Motion

9.2.3.3.6. Natural Frequency (Simulator)

Type:	Axis Parameter Register
RMC75 Address:	**RMC75:** %MD*n*.118, where *n* = 12 + the axis number **RMC150:** %MD*n*.118, where *n* = 24 + the axis number **RMC200:** %MD*n*.485, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].NaturalFreq, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	pu/s^2
Range:	See the **Range** section below
Default Value:	20

Description

This parameter is used with a 2nd order simulator. For details on using simulate mode, see the Simulate Mode topic.

The Natural Frequency for a hydraulic system is normally between 1 and 30. Use the following formula to approximate the natural frequency for hydraulic system:

ω = sqrt[(4*200000 *A^2) /(mass * volume)]

where

A = area of the piston (in^2)

mass = the mass moved by the system (lb)

volume = the volume of trapped oil in the cylinder (in^3)

Range

The Natural Frequency must be less than or equal to the control loop time frequency divided by 4:

Control Loop Time	Max Natural Frequency
0.125 ms	2000 Hz
0.25 ms	1000 Hz
0.5 ms	500 Hz
1 ms	250 Hz
2 ms	125 Hz
4 ms	62.5 Hz

Changing this Parameter

For best results, always change this parameter only from Axis Tools. If the axis is in Simulate mode, and this parameter is changed, simulate model will be suspended until the last simulate mode parameter register (Max Compression) has been updated. At this point, the new simulator model will be calculated and the simulator model will be activated. When the simulator model is suspended, the Actual Position will not move.

Certain simulator settings may result in an invalid model. If this occurs, change your settings. See the Simulating Motion topic for more details.

See Also

Simulating Motion

9.2.3.3.7. Damping Factor (Simulator)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.119, where *n* = 12 + the axis number **RMC150:** %MD*n*.119, where *n* = 24 + the axis number **RMC200:** %MD*n*.486, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].DampingFactor, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	pu/sec^2
Range:	[0..2]
Default Value:	0.75

Description

This parameter is used with a 2nd order simulator. For details on using simulate mode, see the Simulate Mode topic.

The damping factor is a unitless number that specifies how much friction there is in the system. Hydraulic systems typically range from 0.3 to 0.8. If the load has a lot of friction, this value will become larger. A lower value makes the system more difficult to control.

Changing this Parameter

For best results, always change this parameter only from Axis Tools. If the axis is in Simulate mode, and this parameter is changed, simulate model will be suspended until the last simulate mode parameter register (Max Compression) has been updated. At this point, the new simulator model will be calculated and the simulator model will be activated. When the simulator model is suspended, the Actual Position will not move.

Certain simulator settings may result in an invalid model. If this occurs, change your settings. See the Simulating Motion topic for more details.

See Also

Simulating Motion

9.2.3.3.8. Positive Physical Limit (Simulator)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.120, where *n* = 12 + the axis number **RMC150:** %MD*n*.120, where *n* = 24 + the axis number **RMC200:** %MD*n*.487, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].SimPosPhysLim, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	pu
Range:	any
Default Value:	0

Description

This parameter specifies the maximum position the simulator can move to. Notice that this parameter is independent of the Positive Travel Limit.

When the simulator is first enabled, the position is set to the average of the Positive and Negative Physical Limits.

At the Positive and Negative Physical Limit, the simulator simulates a spring force that extends outside the limit by the distance specified by the Maximum Compression parameter. The Maximum Force specifies the max force at the max compression. The spring force increases linearly from zero at the limit to the max force at the max compression. This is useful for simulating pressure or force control.

If the Positive Physical Limit is less than or equal to the Negative Physical Limit, there is no endpoint and therefore the spring force is not simulated.

See Also

Simulating Motion | Negative Physical Limit

9.2.3.3.9. Negative Physical Limit (Simulator)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.121, where *n* = 12 + the axis number **RMC150:** %MD*n*.121, where *n* = 24 + the axis number **RMC200:** %MD*n*.488, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].SimNegPhysLim, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	pu
Range:	any
Default Value:	0

Description

This parameter specifies the minimum position the simulator can move to. Notice that this parameter is independent of the Negative Travel Limit.

When the simulator is first enabled, the position is set to the average of the Positive and Negative Physical Limits.

At the Positive Physical Limit and Negative Physical Limits, the simulator simulates a spring force that extends outside the limit by the distance specified by the Maximum Compression parameter. The Maximum Force specifies the max force at the max compression. The spring force increases linearly from zero at the limit to the max force at the max compression. This is useful for simulating pressure or force control.

If the Positive Physical Limit is less than or equal to the Negative Physical Limit, there is no endpoint and therefore the spring force is not simulated.

See Also

Simulating Motion | Positive Physical Limit

9.2.3.3.10. Output Deadband (Simulator)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.122, where *n* = 12 + the axis number **RMC150:** %MD*n*.122, where *n* = 24 + the axis number **RMC200:** %MD*n*.489, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].SimDeadband, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	**RMC75/150:** V **RMC200:** %
Range:	**RMC75/150:** 0 to 10 **RMC200:** 0 to 100
Default Value:	0

Description

Note: The units differ between the RMC75/150 and RMC200, since the RMC75/150 Control Output is voltage, and the RMC200 Control Output is percent.

This parameter specifies the deadband of the simulator. The deadband is the value of Control Output voltage or percentage at which the simulator begins to move. Use this parameter to simulate overlapped spools on hydraulic valves.

See Also

Simulating Motion | Output Null

9.2.3.3.11. Output Null (Simulator)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.123, where *n* = 12 + the axis number **RMC150:** %MD*n*.123, where *n* = 24 + the axis number **RMC200:** %MD*n*.490, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].SimNullOffset, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	**RMC75/150:** V **RMC200:** %
Range:	**RMC75/150:** -10<V<10 **RMC200:** -100<%<100
Default Value:	0

Description

Note: The units differ between the RMC75/150 and RMC200, since the RMC75/150 Control Output is voltage, and the RMC200 Control Output is percent.

This parameter specifies the null offset of the simulator. The null offset is the value of voltage at which the simulator does not move. Use this parameter to simulate the null offset on hydraulic valves.

See Also

Simulating Motion | Output Deadband

9.2.3.3.12. Weight (Simulator)

Type:	Axis Parameter Register
RMC75 Address:	**RMC75:** %MD*n*.125, where *n* = 12 + the axis number **RMC150:** %MD*n*.125, where *n* = 24 + the axis number **RMC200:** %MD*n*.491, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].SimWeight, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	lb

Range:	>0
Default Value:	1000

Description

The simulator weight parameter is the weight of the mass that the system is moving. This weight is used when calculating the acceleration forces and viscous forces of the simulator. When setting the weight, make sure to set it to a value low enough to keep the actual force within the Maximum Force at the maximum acceleration and velocity of the axis.

Changing this Parameter

For best results, always change this parameter only from Axis Tools. If the axis is in Simulate mode, and this parameter is changed, simulate model will be suspended until the last simulate mode parameter register (Max Compression) has been updated. At this point, the new simulator model will be calculated and the simulator model will be activated. When the simulator model is suspended, the Actual Position will not move.

Certain simulator settings may result in an invalid model. If this occurs, change your settings. See the Simulating Motion topic for more details.

See Also

Simulating Motion | Maximum Force | Maximum Compression

9.2.3.3.13. Maximum Force (Simulator)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.126, where *n* = 12 + the axis number **RMC150:** %MD*n*.126, where *n* = 24 + the axis number **RMC200:** %MD*n*.492, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].SimMaxForce, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	lb
Range:	>0
Default Value:	10000

Description

This parameter specifies the maximum force of the system. The simulator simulates the force it takes to accelerate and move the system (based on the Weight parameter) and also simulates the spring force at the Positive Physical Limit and Negative Physical Limit.

The maximum force will limit the maximum acceleration of the axis. It will also affect the max speed of the axis, similarly to a hydraulic system.

At the Positive and Negative Physical Limits, the simulator simulates a spring force that extends outside the limit by the distance specified by the Maximum Compression parameter. The Maximum Force specifies the max force at the max compression. The spring force increases linearly from zero at the limit to the max force at the max compression. This is useful for simulating pressure or force control.

If the Positive and Negative Physical Limits are both zero, the spring force is not simulated and this parameter is not used.

Changing this Parameter

For best results, always change this parameter only from Axis Tools. If the axis is in Simulate mode, and this parameter is changed, simulate model will be suspended until the last simulate mode parameter register (Max Compression) has been updated. At this point, the new simulator model will be calculated and the simulator model will be activated. When the simulator model is suspended, the Actual Position will not move.

Certain simulator settings may result in an invalid model. If this occurs, change your settings. See the Simulating Motion topic for more details.

See Also

Simulating Motion | Maximum Compression | Weight

9.2.3.3.14. Maximum Compression (Simulator)

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.127, where *n* = 12 + the axis number **RMC150:** %MD*n*.127, where *n* = 24 + the axis number **RMC200:** %MD*n*.493, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].SimMaxComp, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	REAL
Units:	pu
Range:	>0
Default Value:	0.01

Description

At the Positive and Negative Physical Limits, the simulator simulates a spring force that extends outside the limit by the distance specified by the Maximum Compression parameter. The Maximum Force specifies the max force at the max compression. The spring force increases linearly from zero at the limit to the max force at the max compression. This is useful for simulating pressure or force control.

If the Positive and Negative Physical Limits are both zero, the spring force is not simulated and this parameter is not used.

Changing this Parameter

For best results, always change this parameter only from Axis Tools. If the axis is in Simulate mode, and this parameter is changed, simulate model will be suspended until the last simulate mode parameter register (Max Compression) has been updated. At this point, the new simulator model will be calculated and the simulator model will be activated. When the simulator model is suspended, the Actual Position will not move.

Certain simulator settings may result in an invalid model. If this occurs, change your settings. See the Simulating Motion topic for more details.

See Also

Simulating Motion | Maximum Force | Weight

9.2.3.3.15. Simulator Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.116, where *n* = 12 + the axis number **RMC150:** %MD*n*.116, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].SimulationBits, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Simulate
Data Type:	DWORD - see below

Description

For the RMC75/150, the Simulate Configuration register contains bit-addressable parameters for Simulate Mode. For the RMC200, these parameters are individual registers and not part of a configuration register.

This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Bits

This register contains the following parameters. The bits for each are given in the right-hand column.

Parameter	Tag Name	Bit Number(s)
Simulate Mode	Simulate	0
Simulator Order	SimulatorOrder	1-2

For details on the values that each bit represents, see the respective parameter topic.

See Also

Simulating Motion

9.2.3.4. Position/Velocity Control

9.2.3.4.1. In Position Tolerance

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.56, where *n* = 12 + the axis number **RMC150:** %MD*n*.56, where *n* = 24 + the axis number **RMC200:** %MD*n*.180, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].InPosTolernce, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	pu
Range:	≥ 0
Default Value:	0.05

Description

The In Position Tolerance specifies a tolerance around the Command Position. The In Position status bit is set when the Target Position has reached the Command Position and the Actual Position gets within this window. Notice that this bit will not be active until the Target Position has

completed its move. The In Position bit is not latched and will clear if the axis moves back outside the In Position window.

This bit is only used when controlling position in a mode that has a requested position. Therefore, it will be clear in Open Loop, Velocity Control, or Position Control in modes such as gearing that have no final requested position.

Example

If an axis Command Position is 10.000 and the In Position parameter is 0.030, the In Position bit will be set when the Target has reached 10.000 and the Actual Position is between 9.971 and 10.029. The bit will be cleared whenever the Actual Position is outside the range.

See Also

Parameter Registers | In Position

9.2.3.4.2. Position Error Tolerance

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.57, where *n* = 12 + the axis number **RMC150:** %MD*n*.57, where *n* = 24 + the axis number **RMC200:** %MD*n*.181, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].PosErrorTolerance, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	pu
Range:	≥ 0
Default Value:	0.25

Description

The Position Error Tolerance specifies how large the Position Error may become before the Following Error bit is set. If the Following Error bit is set, a Halt will occur if the Following Error Auto Stop is configured to do so and the Direct Output Status bit is off.

The Position Error Tolerance applies only when controlling in Position PID.

See Also

Parameter Registers | Following Error

9.2.3.4.3. At Velocity Tolerance

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.58, where *n* = 12 + the axis number **RMC150:** %MD*n*.58, where *n* = 24 + the axis number **RMC200:** %MD*n*.182, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].AtVelTolerance, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup

Data Type:	REAL
Units:	pu/s
Range:	≥ 0
Default Value:	0.1

Description

The At Velocity Tolerance specifies a tolerance around the Command Velocity. When the Target Velocity has reached the Command Velocity and the Actual Velocity gets within this window, the At Velocity status bit is set. Notice that this bit will not be active until the Target Velocity has completed its move. The At Velocity bit is not latched and will clear if the axis speed moves back outside the At Velocity window.

This bit is only used when controlling velocity in a mode that has a requested velocity. Therefore, it will be clear in Open Loop, Position Control, or Velocity Control in modes such as gearing that have no final requested velocity.

Example

If an axis Command Velocity is 20.000 in/s and the At Velocity Tolerance parameter is 0.30, the At Velocity bit will be set when the Target Velocity has reached the Command Velocity and the axis is at a constant velocity between 19.71 and 20.29. The bit will be cleared whenever the Actual Velocity is outside the range.

See Also

Parameter Registers | At Velocity | Velocity Error Tolerance | Stop Threshold

9.2.3.4.4. Velocity Error Tolerance

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.59, where *n* = 12 + the axis number **RMC150:** %MD*n*.59, where *n* = 24 + the axis number **RMC200:** %MD*n*.183, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].VelErrorTolerance, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	pu/s
Range:	≥ 0
Default Value:	1

Description

In velocity control (when the Current Control Mode status register is velocity), the Velocity Error Tolerance specifies how large the Velocity Error may become before the Following Error status bit is set. If the Following Error bit is set, a Halt will occur if the Following Error Auto Stop is configured to do so and the Direct Output Status bit is off.

The Velocity Error Tolerance applies only when controlling in Velocity PID.

Note:
Notice that the Following Error status bit will be set due to exceeding the Velocity Error Tolerance only in *velocity* control. If you issue a Move Velocity command in *position* control, the Following Error bit

will not be set due to exceeding the Velocity Error Tolerance! It will only be set due to exceeding the Position Error Tolerance.

See Also

Parameter Registers | Following Error | At Velocity Tolerance

9.2.3.4.5. Default Integrator Mode

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.60.0-3, where *n* = 12 + the axis number **RMC150:** %MD*n*.60.0-3, where *n* = 24 + the axis number **RMC200:** %MD*n*.170 where *n* = 384 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].PriControlBits.IntMode, where *n* is the axis number **RMC200:** _Axis[*n*].DefIntMode, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Position/Velocity Control
Data Type:	RMC75/150: Bits RMC200: DINT
Range:	Always Held (0), Always Active (1), Always Zero (2), TGDone (3), Decel (4)
Default Value:	Always Active (1)

Description

The Default Integrator Mode specifies the functionality of the Integral Gain on both the primary and secondary control loops. The following modes are available:

- **Always Held (0)**
  The Integral gain is not active. The Integral Output Term will be held at the value at which it was when the Always Held mode is selected.
- **Always Active (1)**
  The Integral gain is always active.
- **Always Zero (2)***
  The Integral gain is not active and the Integral Output Term will be held at zero.
- **TGDone (3)***
  The Integratral gain is only active when the TGDone bit is set and the final target velocity and acceleration are both zero. For a typical Move Absolute command, this means that the Integral gain is not active during the move and becomes active after the move completes, as indicated by the Target Generator Done (TGDone) or Pressure/Force Target Generator Done (PFTGDone) status bit. During the move, the Integral Output Term will be held at the value at which it was when the move started.
- **Decel (4)***
  The Integral gain is not active during the move and becomes active when the final deceleration portion of the move begins. Deceleration is indicated by the Target Generator State A and Target Generator State B status bits, or the Pressure/Force Target Generator State A and Pressure/Force Target Generator State B status bits, but notice that it is only in the *final* deceleration of a commanded move that the integral gain will be active. For motion commands with no defined deceleration phase, the integrator will become active when the

TGDone bit is set. The integral gain also remains active after the move complete. Otherwise during the move, the Integral Output Term will be held at the value at which it was when the move started.

*Modes 2-4 are available in RMC75/150 firmware 3.66.0 and newer, and RMC200 firmware 1.05.0 and newer.

Why Bother?

The integral gain has advantages and disadvantages. It performs very well when holding position. It helps the Actual Position track the Target Position during a move. If the Feed Forwards are not set correctly, or if the system is non-linear, then the integral gain can make up for this. However, if the integral gain needs to make up for these situations, then the integral term may 'wind up' to a large value. If the integral term has a large value in it when a move completes (the Target Velocity is zero), then that term will result in extra Control Output, and will cause the Actual Position to overshoot or undershoot the Commanded Position. This is also true for pressure and force control.

The Integrator Mode options can be used prevent the integrator from overshooting or undershooting. In situations when the system response is known to vary, the integrator can be turned off, or can be cleared.

The TGDone option keeps the integrator from winding up during the move, reducing final undershoot or overshoot, yet allows the integrator to do what it does best, namely keeping the axis in position.

The Decel option is similar to the TGDone option, but it turns on the integrator when the target generator reaches the deceleration phase of the move. This allows a bit more time for the integral gain to start getting the axis into position.

Which option you select varies based on the system. The TGDone and Decel options are most useful for systems where the position need not track so well during the move system, but final position is important. For systems that require precise control during the entire move, turning off the integrator is not recommended.

The Integrator Adjust (70) command can also be used for situations that require modifying the integrator.

Format Details

This section is primarily for addressing the Integrator Mode when communicating with the RMC from an external device. This information is not necessary when configuring the Integrator Mode in RMCTools.

RMC75/150:

The Integrator Mode is selected with bits 0-3 in the register. These bits correspond to the Integrator Mode as shown in the following table:

Bit 3	Bit 2	Bit 1	Bit 0	Value	Integrator Mode
0	0	0	0	0	Always Held
0	0	0	0	1	Always Active
0	0	0	1	0	Always Zero
0	0	0	1	1	TGDone
0	0	1	0	0	Decel

See the Primary Control Register topic for details about the register containing these bits.

RMC200:

The values correspond to the Integrator Mode as shown in the following table:

Value	Integrator Mode
0	Always Held
1	Always Active
2	Always Zero

3	TGDone
4	Decel

See Also

Parameter Registers | Primary Control Register | Position Integral Gain | Set Integrator Mode (71) Command

9.2.3.4.6. Proportional Gain

Type:	Axis Parameter Register
Address:	**RMC75:** Gain Set #0: %MD*n*.61 Gain Set #1: %MD*n*.128 where *n* = 12 + the axis number **RMC150:** Gain Set #0: %MD*n*.61 Gain Set #1: %MD*n*.128 where *n* = 24 + the axis number **RMC200:** Gain Set #0: %MD*n*.200 Gain Set #1: %MD*n*.230 where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].PropGain Gain Set #1: _Axis[*n*].PropGain_2 where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Position/Velocity Control
Data Type:	REAL
Units:	Position Control: %/pu Velocity control: %/(pu/sec) % = percent of maximum Control Output (default is 10V)
Range:	≥ 0
Default Value:	1

Description

The Proportional Gain is the most important tuning parameter. It affects the responsiveness of the system. A low gain makes the system sluggish and unresponsive. A gain that is too high makes the axis oscillate or vibrate.

The Proportional Gain controls how much of the Control Output is added to the PFID Output due to the Position Error or Velocity Error for position or velocity control, respectively. Position control is defined as when the Current Control Mode is Position PID. Velocity control is defined as when the Current Control Mode is Velocity PID.

See the Tuning topic for details on how to properly adjust the Proportional Gain.

Each gain of the axis contributes to the Control Output. The contribution from the Proportional Gain is called the Proportional Output Term.

Advanced: I-PD Control

The Differential Gain controls how much the PFID Output is adjusted based on the change in the Actual Position or Actual Velocity for position or velocity control, respectively. Position control is defined as when the Current Control Mode is Position I-PD. Velocity control is defined as when the Current Control Mode is Velocity I-PD.

Mathematical Definition

The Proportional Gain units are: percent of the maximum Control Output per control unit (cu = position-units or velocity-units). The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

Position PID

$D_n = (P_{Target}n - P_{Actual}n) \times K_P$

Velocity PID

$D_n = (V_{Target}n - V_{Actual}n) \times K_P$

Position I-PD

$D_n = - (P_{Actual}n - P_{Actual}n\text{-}1) \times K_P$

Velocity I-PD

$D_n = - (V_{Actual}n - V_{Actual}n\text{-}1) \times K_P$

where

D_n = Proportional Term at sample n [% of maximum Control Output]

P = Position [pu]

V = Velocity [pu/sec]

K_P = Proportional Gain [%/(pu) or %/(pu/sec)]

pu = position-unit

See Also

Parameter Registers | Proportional Output Term |

9.2.3.4.7. Integral Gain

Type:	Axis Parameter Register
Address:	**RMC75:**
	Gain Set #0: %MD*n*.62
	Gain Set #1: %MD*n*.129
	where *n* = 12 + the axis number
	RMC150:
	Gain Set #0: %MD*n*.62
	Gain Set #1: %MD*n*.129
	where *n* = 24 + the axis number
	RMC200:
	Gain Set #0: %MD*n*.201
	Gain Set #1: %MD*n*.231
	where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].IntGain
	Gain Set #1: _Axis[*n*].IntGain_2

	where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Position/Velocity Control
Data Type:	REAL
Units:	Position Control: %/(pu·sec) Velocity control: %/((pu/sec)·sec) % = percent of maximum Control Output (default is 10V)
Range:	≥ 0
Default Value:	0

Description

General

This gain helps compensate for changing system dynamics, such as varying loads, and often aids the axis in rapidly reaching the Command Position or Command Velocity. A low gain slows the response of the axis to changes and may keep the axis from reaching the Command Position or Command Velocity. A gain that is too high may make the axis oscillate.

The Integral Gain controls how much of the Control Output is generated due to the accumulated Position Error or Velocity Error while in position control or velocity control, respectively. Position control is defined as when the Current Control Mode is Position PID. Velocity control is defined as when the Current Control Mode is Velocity PID.

Each gain of the axis contributes to the Control Output. The contribution from the Integral Gain is called the Integral Output Term. For both PID and I-PD, each control loop, the Integral Gain is multiplied by the control loop time and Position Error or Velocity Error and added to the Integral Output Term. The Integral Output term is added to the PFID Output.

See the Tuning topic for details on how to properly adjust the Integral Gain.

The Integral Gain may be disabled or only be allowed to decrease the Integral output Term under certain conditions. See below for more details.

Advanced: I-PD

This gain is the most important gain for I-PD control. It must be non-zero in I-PD mode. If it is zero, it will cause a run-time error. A higher value will increase the response. A value that is too high will cause oscillation and instability.

Special Conditions

Integrator Hold

The Integral Gain can be disabled by the user by setting the Integrator Mode to Always Held. The Integrator Output Term will be held at the value at the time it was set.

The Integrator Hold does not apply to Position I-PD or Velocity I-PD, since the integrator is an important part of the I-PD control.

Integrator Adjust

The Integral Output Term can be manually adjusted with the Integrator Adjust (70) command. This is useful for clearing it if it has "wound up" for some reason. See the Integrator Adjust (70) topic for more details.

Integral Gain is Zero

If the Integral Gain is zero, the Integral Output Term will be set to zero.

Output Limiting

When the sum of the Proportional Term and the Integral Term reaches 100%, the Integral Output Term may be reduced to limit the PFID Output. This helps to avoid overshoot. This applies to all control modes.

Example:

If the Proportional Output Term is 60%, and the Integral Output Term is 50%, then the total is 110%. This will be limited to 100%, so the Integral Output Term will be set to 40%.

Wind-Down Only

In the Position PID control mode, the Integral Output Term will only be allowed to decrease in the following conditions:.

- **Within Deadband Window**
  When the target is stopped, and the Position Error is less than the Deadband Tolerance, the Integral Output Term will only be allowed to wind down. In systems with a deadband, this avoids ratcheting the axis back and forth around the deadband. See the Output Deadband topic for more details.
- **Within one-half count of Command Position**
  When the axis is within one-half count of the Command Position, the Integral Gain is only allowed to decrease the Integral Output Term. This is similar to the deadband case. Except in the rare case when the axis is scaled such that the transducer feedback counts match up exactly with the position units, there will always be a small difference between the Command Position and the nearest feedback count. Normally, this would cause the axis to "hunt" for the command position. Constraining the Integral Output Term to only decrease prevents the axis from "hunting".

Mathematical Definition

PID and I-PD

The Integral Output Term is calculated from the Integral Gain as follows:

$$I_n = (e_n \times K_i \times T) + I_{n-1}$$

where

I_n = Integral Term [% of maximum Control Output]

e_n = Position Error at sample n [cu]

K_i = Integral Gain [% / (cu x sec)]

T = Time period of control loop [sec]

cu = control unit (position-unit or velocity-unit)

The Integral Gain units are: percent of the maximum Control Output per unit of control (position, velocity, pressure, force etc) of Position Error times time. The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

See Also

Parameter Registers | Integral Output Term

9.2.3.4.8. Differential Gain

Type:	Axis Parameter Register
Address:	**RMC75:**
	Gain Set #0: %MD*n*.63
	Gain Set #1: %MD*n*.130
	where *n* = 12 + the axis number
	RMC150:
	Gain Set #0: %MD*n*.63
	Gain Set #1: %MD*n*.130
	where *n* = 24 + the axis number

	RMC200:
	Gain Set #0: %MD*n*.202
	Gain Set #1: %MD*n*.232
	where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].DiffGain
	Gain Set #1: _Axis[*n*].DiffGain_2
	where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Position/Velocity Control
Data Type:	REAL
Units:	Position Control: %/(pu/sec)
	Velocity control: %/(pu/sec[2)]
	% = percent of maximum Control Output (default is 10V)
Range:	≥ 0
Default Value:	0

Description

The Differential Gain may greatly enhance performance on many motion systems. On velocity drives or hydraulic systems, Differential Gain will tend to dampen out oscillations and help the axis track during acceleration and deceleration. On torque drives, the differential gain is essential for providing damping to the motor.

The Differential Gain controls how much of the Control Output is added to the PFID Output due to the difference between the target and actual velocity or target and actual acceleration for position or velocity control, respectively. Position control is defined as when the Current Control Mode is Position PID. Velocity control is defined as when the Current Control Mode is Velocity PID.

A disadvantage of Differential Gain is that it amplifies position measurement noise. If there is too much noise or the gain is too high, this can cause the system to chatter or oscillate.

Each gain of the axis contributes to the Control Output. The contribution from the Differential Gain is called the Differential Output Term.

See Tuning topic for details on how to properly adjust the Differential Gain.

Advanced: I-PD Control

The Differential Gain controls how much the PFID Output is adjusted based on the change in the Actual Velocity or Actual Acceleration for position or velocity control, respectively. Position control is defined as when the Current Control Mode is Position I-PD. Velocity control is defined as when the Current Control Mode is Velocity I-PD.

Definition

The Differential Gain units are: percent of the maximum Control Output per control units (position-units or velocity-units) per second (cu/sec). The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

The Differential Output Term control is calculated from the Integral Gain as follows:

Position PID

$D_n = (V_{Target}n - V_{Actual}n) \times K_D$

Velocity PID

$D_n = (A_{Target}n - A_{Actual}n) \times K_D$

Position I-PD

$D_n = - (V_{Actual}n - V_{Actual}n\text{-}1) \times K_D$

Velocity I-PD

$$D_n = - (A_{Actual}n - A_{Actual}n\text{-}1) \times K_D$$

where

D_n = Differential Term at sample n [% of maximum Control Output]

V = Velocity [pu/sec]

A = Acceleration [pu/sec^2]

K_D = Differential Gain [%/(pu/sec) or %/(pu/sec^2)]

pu = position-unit

See Also

Parameter Registers | Differential Output Term

9.2.3.4.9. Velocity Feed Forward

Type:	Axis Parameter Register
Address:	**RMC75:** Gain Set #0: %MD*n*.65, where *n* = 12 + the axis number Gain Set #1: %MD*n*.132, where *n* = 12 + the axis number **RMC150:** Gain Set #0: %MD*n*.65, where *n* = 24 + the axis number Gain Set #1: %MD*n*.132, where *n* = 24 + the axis number **RMC200:** Gain Set #0: %MD*n*.206, where *n* = 384 + the axis number Gain Set #1: %MD*n*.236, where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].VelFFwd Gain Set #1: _Axis[*n*].VelFFwd_2 where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Position/Velocity Gains
Data Type:	REAL
Units:	%/(pu/s)
Range:	≥ 0
Default Value:	0

Description

The Velocity Feed Forward is one of the gains used in Position PID and Velocity PID control. It causes the controller to give an amount of Control Output proportional to the speed of the velocity of the axis. The Velocity Feed Forward is only applied when the axis is in Closed Loop control. It helps the axis maintain the Target Position while the axis is moving at constant velocity.

The Velocity Feed Forward is available on position or velocity axes with symmetrical gains. Position or velocity axes with ratioed gains have a Velocity Feed Forward (Positive) and a Velocity Feed Forward (Negative).

The Feed Forward Adjust (69) command can be used to set the Velocity Feed Forward during machine operation to adjust the system for changing system dynamics.

The Velocity Feed Forward *Term* is the portion of the Control Output resulting from the Velocity Feed Forward, which is a gain.

I-PD

In Position I-PD and Velocity I-PD control, the Velocity Feed Forward is not used. If the gains are set to Ratioed, the Velocity Feed Forward (Positive) and Velocity Feed Forward (Negative) are used, but they do not contribute any Control Output. They are used only to ratio the gains for either direction of travel.

Definition

The Velocity Feed Forward units are: percent of the maximum Control Output per position-units per second (pu/sec). The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

The Velocity Feed Forward Term is calculated from the Velocity Feed Forward as follows:

$$D_{VFF}n = K_{VFF}V_{Target}n$$

where

$D_{VFF}n$ = Velocity Feed Forward Term at sample n [% of maximum Control Output]

K_{VFF} = Velocity Feed Forward [%/(pu/s)]

$V_{Target}n$ = Target Velocity at sample n [pu/s]

Tuning

The Velocity Feed Forward helps the axis maintain the Target Position while the axis is moving. This makes it "easier" for the other gains to control the position of the axis. See the Tuning topic for details on how to properly adjust the Velocity Feed Forward.

The Feed Forward Adjust (69) command can also be used to set the Velocity Feed Forward.

See Also

Parameter Registers | Velocity Feed Forward Term |Feed Forward Adjust (69)

9.2.3.4.10. Velocity Feed Forward (Positive)

Type:	Axis Parameter Register
Address:	**RMC75:** Gain Set #0: %MD*n*.65, where *n* = 12 + the axis number Gain Set #1: %MD*n*.132, where *n* = 12 + the axis number **RMC150:** Gain Set #0: %MD*n*.65, where *n* = 24 + the axis number Gain Set #1: %MD*n*.132, where *n* = 24 + the axis number **RMC200:** Gain Set #0: %MD*n*.206, where *n* = 384 + the axis number Gain Set #1: %MD*n*.236, where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].VelFFwdP Gain Set #1: _Axis[*n*].VelFFwdP_2 where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Position/Velocity Gains
Data Type:	REAL
Units:	%/(pu/s)
Range:	≥ 0

Default Value: 0

Description

The Positive Velocity Feed Forward is one of the gains. It causes the controller to give an amount of Control Output proportional to the speed of the velocity of the axis.

The Positive Velocity Feed Forward applies when the Control Output is positive. The Negative Velocity Feed Forward applies when the Control Output is negative. The Velocity Feed Forwards are only applied when the axis is in Closed Loop control. They help the axis maintain the Target Position while the axis is moving at constant velocity.

The Positive and Negative Velocity Feed Forwards are available only on position or velocity axes with ratioed gains. Position or velocity axes with symmetrical gains have only one Velocity Feed Forward.

All the gains of the axis are ratioed by the Positive and Negative Velocity Feed Forwards. Once the Positive and Negative Velocity Feed Forwards have been tuned, the gains in the direction of the highest Velocity Feed Forward will be applied as they are. The gains in the direction of the lowest Velocity Feed Forward will be decreased by the ratio of the Velocity Feed Forwards. This results in very good control of the cylinder in both directions.

Special Case: If either Feed Forward is set to zero, or they are identical, the gain ratio will be one-to-one. The Feed Forward contribution to the Control Output will still be as specified by the Feed Forward parameter.

Note:
Ratioed gains are automatically selected by the Tuning Wizard if moves in both directions are used.

Example

Consider a single-rod hydraulic cylinder that moves at 3 in/sec when a Control Output of 1V is applied and -2.3 in/sec when a Control Output of -1V is applied. Since it moves faster in the positive direction, those gains need to be smaller.

The correct Positive Velocity Feed Forward for this system is 3.33. The correct Negative Velocity Feed Forward is 4.35. By choosing Ratioed gains, the gains in the negative direction will be applied as they are. The gains in the positive direction will be multiplied by 3.33/4.35, which is 0.766.

The Velocity Feed Forward *Term* is the portion of the Control Output resulting from the Velocity Feed Forwards, which are gains.

I-PD

In Position I-PD and Velocity I-PD control, the Velocity Feed Forwards do not contribute to the Control Output; they are used only to ratio the gains for either direction of travel.

Definition

The Velocity Feed Forward units are: percent of the maximum Control Output per position-units per second (pu/sec). The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

The Velocity Feed Forward Term is calculated from the Velocity Feed Forward as follows:

$$D_{VFF}n = K_{VFF}V_{Target}n$$

where

$D_{VFF}n$ = Velocity Feed Forward Term at sample n [% of maximum Control Output]

K_{VFF} = Velocity Feed Forward [%/(pu/s)]

$V_{Target}n$ = Target Velocity at sample n [pu/s]

Tuning

The Velocity Feed Forwards help the axis maintain the Target Position while the axis is moving. This makes it "easier" for the other gains to control the position of the axis. See the Tuning topic for details on how to properly adjust the Velocity Feed Forward.

See Also

Symmetrical/Ratioed | Velocity Feed Forward Term |Velocity Feed Forward (Negative)

9.2.3.4.11. Velocity Feed Forward (Negative)

Type:	Axis Parameter Register
Address:	**RMC75:** Gain Set #0: %MD*n*.68, where *n* = 12 + the axis number Gain Set #1: %MD*n*.135, where *n* = 12 + the axis number **RMC150:** Gain Set #0: %MD*n*.68, where *n* = 24 + the axis number Gain Set #1: %MD*n*.135, where *n* = 24 + the axis number **RMC200:** Gain Set #0: %MD*n*.207, where *n* = 384 + the axis number Gain Set #1: %MD*n*.237, where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].VelFFwdN Gain Set #1: _Axis[*n*].VelFFwdN_2 where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Position/Velocity Gains
Data Type:	REAL
Units:	%/(pu/s)
Range:	≥ 0
Default Value:	0

Description

The Negative Velocity Feed Forward is one of the gains. It causes the controller to give an amount of Control Output proportional to the speed of the velocity of the axis.

The Negative Velocity Feed Forward applies when the Control Output is negative. The Positive Velocity Feed Forward applies when the Control Output is positive. The Velocity Feed Forwards are only applied when the axis is in Closed Loop control. They help the axis maintain the Target Position while the axis is moving at constant velocity.

The Positive and Negative Velocity Feed Forwards are available only on position or velocity axes with ratioed gains. Position or velocity axes with symmetrical gains have only one Velocity Feed Forward.

All the gains of the axis are ratioed by the Positive Negative Velocity Feed Forwards. Once the Positive and Negative Velocity Feed Forwards have been tuned, the gains in the direction of the highest Velocity Feed Forward will be applied as they are. The gains in the direction of the lowest Velocity Feed Forward will be decreased by the ratio of the Velocity Feed Forwards. This results in very good control of the cylinder in both directions.

Note:
Ratioed gains are automatically selected by the Tuning Wizard if moves in both directions are used.

Example

Consider a single-rod hydraulic cylinder that moves at 3 in/sec when a Control Output of 1V is applied and -2.3 in/sec when a Control Output of -1V is applied. Since it moves faster in the positive direction, those gains need to be smaller.

The correct Positive Velocity Feed Forward for this system is 3.33. The correct Negative Velocity Feed Forward is 4.35. By choosing Ratioed gains, the gains in the negative direction will be applied as they are. The gains in the positive direction will be multiplied by 3.33/4.35, which is 0.766.

The Velocity Feed Forward *Term* is the portion of the Control Output resulting from the Velocity Feed Forwards, which are gains.

I-PD

In Position I-PD and Velocity I-PD control, the Velocity Feed Forwards are used, only if the gains are set to Ratioed. In that case, the Velocity Feed Forward (Positive) and Velocity Feed Forward (Negative) do not contribute to the Control Output. They are used only to ratio the gains for either direction of travel.

If either Feed Forward is set to zero, or they are identical, the gain ratio will be one. Otherwise, for the direction of the largest Feed Forward, the gains will be applied as they. In the direction of the smallest Feed Forward, the gains will be multiplied by the ratio of the smallest Feed Forward divided by the largest feed forward.

Definition

The Velocity Feed Forward units are: percent of the maximum Control Output per position-units per second (pu/sec). The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

The Velocity Feed Forward Term is calculated from the Velocity Feed Forward as follows:

$$D_{VFF}n = K_{VFF}V_{Target}n$$

where

$D_{VFF}n$ = Velocity Feed Forward Term at sample n [% of maximum Control Output]

K_{VFF} = Velocity Feed Forward [%/(pu/s)]

$V_{Target}n$ = Target Velocity at sample n [pu/s]

Tuning

The Velocity Feed Forwards help the axis maintain the Target Position while the axis is moving. This makes it "easier" for the other gains to control the position of the axis. See the Tuning topic for details on how to properly adjust the Velocity Feed Forward.

See Also

Parameter Registers | Velocity Feed Forward Term

9.2.3.4.12. Acceleration Feed Forward

Type:	Axis Parameter Register
Address:	**RMC75:**
	Gain Set #0: %MD*n*.66
	Gain Set #1: %MD*n*.133
	where *n* = 12 + the axis number
	RMC150:
	Gain Set #0: %MD*n*.66

	Gain Set #1: %MD*n*.133 where *n* = 24 + the axis number **RMC200:** Gain Set #0: %MD*n*.208 Gain Set #1: %MD*n*.238 where *n* = 384 + the axis number
System Tag:	Gain Set #0:_Axis[*n*].AccFFwd Gain Set #1:_Axis[*n*].AccFFwd_2 where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Position/Velocity Gains
Data Type:	REAL
Units:	%/(pu/s^2)
Range:	≥ 0
Default Value:	0

Description

The Acceleration Feed Forward causes the controller to give Control Output proportional to the Target Acceleration. Acceleration Feed Forward is only applied when the axis is in Closed Loop control and accelerating or decelerating. It helps the axis maintain the Target Position.

The Acceleration Feed Forward Term is the portion of the Control Output resulting from the Acceleration Feed Forward Gain.

This parameter is not used in the Position I-PD or Velocity I-PD control modes.

Definition

The Acceleration Feed Forward Term is calculated from the Acceleration Feed Forward as follows:

$$D_{AFF}n = K_{AFF}A_{Target}n$$

where

$D_{AFF}n$ = Acceleration Feed Forward Term at sample n [% of maximum Control Output]

K_{AFF} = Acceleration Feed Forward [%/(pu/sec^2)]

$A_{Target}n$ = Target Acceleration at sample n [pu/sec^2]

Tuning

The Acceleration Feed Forward helps the axis track the Target Position while the axis is accelerating and decelerating. This makes it "easier" for the other gains to control the position of the axis.

The Acceleration Feed Forward should not be adjusted until the Velocity Feed Forward is adjusted correctly. See the Tuning topic for more details on how to properly adjust the Acceleration Feed Forward.

See Also

Parameter Registers | Acceleration Feed Forward Term

9.2.3.4.13. Jerk Feed Forward

Type:	Axis Parameter Register

Address:	**RMC75:** Gain Set #0: %MD*n*.67, Gain Set #1: %MD*n*.134, where *n* = 12 + the axis number **RMC150:** Gain Set #0: %MD*n*.67, Gain Set #1: %MD*n*.134, %MD*n*.67, where *n* = 24 + the axis number **RMC200:** Gain Set #0: %MD*n*.209, Gain Set #1: %MD*n*.239, %MD*n*.67, where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].JerkFFwd, where *n* is the axis number Gain Set #1: _Axis[*n*].JerkFFwd_2, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Position/Velocity Control → Gain Set#
Data Type:	REAL
Units:	%/(pu/s^3)
Range:	≥0
Default Value:	0

Description

The Jerk Feed Forward causes the controller to give extra Control Output proportional to the change in acceleration. Acceleration Feed Forward is only applied when the axis is in Closed Loop control and the axis is accelerating. It helps the axis maintain the Target Position. Jerk Feed Forward is helpful in systems that can be modeled as a second-order system. Many systems do not require any Jerk Feed Forward.

The Jerk Feed Forward Term is the portion of the Control Output resulting from the Acceleration Feed Forward Gain.

This parameter is not used in the Position I-PD or Velocity I-PD control modes.

Tuning

The Jerk Feed Forward helps the axis track the Target Position while the axis is accelerating. This makes it "easier" for the other gains to control the position of the axis. Many systems will not need any Jerk Feed Forward.

The Jerk Feed Forward should not be adjusted until the Velocity Feed Forward and Acceleration Feed Forward are adjusted correctly. See Tuning topic for more details on how to properly adjust the Jerk Feed Forward.

See Also

Parameter Registers | Jerk Feed Forward Term

9.2.3.4.14. Double Differential Gain

Type:	Axis Parameter Register
Address:	**RMC75:** Gain Set #0: %MD*n*.69

	Gain Set #1: %MD*n*.136 where *n* = 12 + the axis number **RMC150:** Gain Set #0: %MD*n*.69 Gain Set #1: %MD*n*.136 where *n* = 24 + the axis number **RMC200:** Gain Set #0: %MD*n*.203 Gain Set #1: %MD*n*.233 where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].DblDiffGain Gain Set #1: _Axis[*n*].DblDiffGain_2 where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Position/Velocity Control → Gain Set#
Data Type:	REAL
Units:	Position Control: %/(pu/sec^2) Velocity control: %/(pu/sec^3) % = percent of maximum Control Output (default is 10V)
Range:	≥ 0
Default Value:	0

Description

The Double Differential Gain is available only on position and velocity axes with the High-Order Control parameter set to Accel Control. The Double Differential Gain controls how much of the Control Output is is added to the PFID Output due to the difference between the Target Acceleration and the Actual Acceleration or Target and Actual Jerk for position or velocity control, respectively. Position control is defined as when the Current Control Mode is Position PID. Velocity control is defined as when the Current Control Mode is Velocity PID.

Each gain of the axis contributes to the Control Output. The contribution from the Double Differential Gain is called the Double Differential Output Term.

Advanced: I-PD Control

The Double Differential Gain controls how much the PFID Output is adjusted based on the change in the Actual Acceleration or Actual Jerk for position or velocity control, respectively. Position control is defined as when the Current Control Mode is Position I-PD. Velocity control is defined as when the Current Control Mode is Velocity I-PD.

Tuning

See the Tuning Active Damping and Acceleration Control topic for details.

Definition

PositionPID

$D_n = (A_{Target}n - A_{Actual}n) \times K_{DD}$

Velocity PID

$D_n = (J_{Target}n - J_{Actual}n) \times K_{DD}$

Position I-PD

$D_n = - (A_{Actual}n - A_{Actual}n\text{-}1) \times K_{DD}$

Velocity I-PD

$D_n = - (J_{Actual}n - J_{Actual}n\text{-}1) \times K_{DD}$

where

D_n = Double Differential Term at sample n [% of maximum Control Output]

A = Acceleration [pu/sec^2]

J = Acceleration [pu/sec^3]

K_{DD} = Double Differential Gain [%/(pu/sec^2) or %/(pu/sec^3)]

pu = position-unit

The Double Differential Gain units are: Percent of the maximum Control Output per position units per second squared (pu/sec^2). The default value of the maximum Control Output is 10V, but can be changed using the Output Scale parameter.

See Also

Parameter Registers | Double Differential Output Term

9.2.3.4.15. Active Damping Proportional Gain

Type:	Axis Parameter Register
Address:	**RMC75:** Gain Set #0: %MD*n*.69 Gain Set #1: %MD*n*.136 where *n* = 12 + the axis number **RMC150:** Gain Set #0: %MD*n*.69 Gain Set #1: %MD*n*.136 where *n* = 24 + the axis number **RMC200:** Gain Set #0: %MD*n*.215 Gain Set #1: %MD*n*.245 where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].ADPropGain Gain Set #1: _Axis[*n*].ADPropGain_2 where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Position/Velocity Gains
Data Type:	REAL
Units:	%/(pu/sec^2) % = percent of maximum Control Output (default is 10V)
Range:	≥ 0
Default Value:	0

Description

This gain is only valid when Active Damping is selected. The Active Damping Proportional Gain controls how much the Control Output is adjusted based on the change in the Actual Acceleration or the Actual Force.

Each gain of the axis contributes to the Control Output. In position control, the contribution from the Active Damping Proportional Gain is called the Double Differential Output Term. In velocity control, the contribution from the Active Damping Proportional Gain is called the Differential Output Term. Notice that in velocity control with active damping, the Differential Gain is not used.

Position control is defined as when the Current Control Mode is Position PID or Position I-PD. Velocity control is defined as when the Current Control Mode is Velocity PID or Velocity I-PD.

Tuning

See the Tuning Active Damping and Acceleration Control topic for details.

Mathematical Definition

The Active Damping Proportional Gain units are: percent of the maximum Control Output per position unit per second squared. The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

PID with Active Damping

Using Force Input

$D_n = - F_{Actual}n \times K_{ADP}$

Using Acceleration, Filter, or Model

$D_n = - A_{Actual}n \times K_{ADP}$

I-PD with Active Damping

Using Force Input

$D_n = - (F_{Actual}n - F_{Actual}n\text{-}1) \times K_{ADP}$

Using Acceleration, Filter, or Model

$D_n = - (A_{Actual}n - A_{Actual}n\text{-}1) \times K_{ADP}$

where

D_n = Active Damping Proportional Term at sample n [% of maximum Control Output]

F = Force [Fr]

A = Acceleration [pu/sec^2]

K_{ADP} = Active Damping Proportional Gain [%/(Fr])] or [%/(pu/sec^2])]

See Also

Parameter Registers | Double Differential Output Term | Active Damping Differential Gain

---

9.2.3.4.16. Triple Differential Gain

Type:	Axis Parameter Register
Address:	**RMC75:**
	Gain Set #0: %MD*n*.70
	Gain Set #1: %MD*n*.137
	where *n* = 12 + the axis number
	RMC150:
	Gain Set #0: %MD*n*.70
	Gain Set #1: %MD*n*.137
	where *n* = 24 + the axis number
	RMC200:
	Gain Set #0: %MD*n*.204

	Gain Set #1: %MD*n*.234 where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].TrpDiffGain Gain Set #1: _Axis[*n*].TrpDiffGain_2 where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Position/Velocity Control → Gain Set#
Data Type:	REAL
Units:	%/(pu/sec^3) % = percent of maximum Control Output (default is 10V)
Range:	≥ 0
Default Value:	0

Description

The Triple Differential Gain is available only on position-acceleration axes with the High-Order Control parameter set to Accel Control. It controls how much of the Control Output is generated proportional to the Actual Jerk.

Each gain of the axis contributes to the Control Output. The contribution from the Triple Differential Gain is called the Triple Differential Output Term.

Tuning

See the Tuning Active Damping and Acceleration Control topic for details.

Definition

Position PID

$D_n = (J_{Target}n - J_{Actual}n) \times K_{TD}$

Position I-PD

$D_n = - (J_{Actual}n - J_{Actual}n\text{-}1) \times K_{TD}$

where

D_n = Triple Differential Term at sample n [% of maximum Control Output]

J = Jerk [pu/sec^3]

K_{TD} = Triple Differential Gain [%/(pu/sec^3)]

pu = position-unit

The Triple Differential Gain units are: Percent of the maximum Control Output per position units per second cubed (pu/sec^3). The default value of the maximum Control Output is 10V, but can be changed using the Output Scale parameter.

See Also

Parameter Registers | Triple Differential Output Term

9.2.3.4.17. Active Damping Differential Gain

Type:	Axis Parameter Register
Address:	**RMC75:** Gain Set #0: %MD*n*.70

	Gain Set #1: %MD*n*.137 where *n* = 12 + the axis number **RMC150:** Gain Set #0: %MD*n*.70 Gain Set #1: %MD*n*.137 where *n* = 24 + the axis number **RMC200:** Gain Set #0: %MD*n*.216 Gain Set #1: %MD*n*.246 where *n* = 384 + the axis number
System Tag:	Gain Set #0: _Axis[*n*].ADDiffGain Gain Set #1: _Axis[*n*].ADDiffGain_2 where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Position/Velocity Gains
Data Type:	REAL
Units:	%/(pu/sec^3) % = percent of maximum Control Output (default is 10V)
Range:	≥ 0
Default Value:	0

Description

This gain is only valid when Active Damping is selected. The Active Damping Differential Gain controls how much the Control Output is reduced due to the Actual Jerk or Actual Force Rate.

Each gain of the axis contributes to the Control Output. In position control, the contribution from the Active Damping Differential Gain is called the Triple Differential Output Term. In velocity control, the contribution from the Active Damping Differential Gain is called the Double Differential Output Term.

Position control is defined as when the Current Control Mode is Position PID or Position I-PD. Velocity control is defined as when the Current Control Mode is Velocity PID or Velocity I-PD.

Tuning

See the Tuning Active Damping and Acceleration Control topic for details.

Mathematical Definition

The Active Damping Differential Gain units are: percent of the maximum Control Output per position unit per second cubed. The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

PID with Active Damping

Using Acceleration Input, Filter, or Model

$D_n = - J_{Actual}n \times K_{ADD}$

Using Force Input

$D_n = - \Delta F_{Actual}n \times K_{ADD}$

I-PD with Active Damping

Using Acceleration Input, Filter, or Model

$D_n = - (J_{Actual}n - J_{Actual}n\text{-}1) \times K_{ADD}$

Using Force Input

$D_n = - (\Delta F_{Actual}n - \Delta F_{Actual}n\text{-}1) \times K_{ADD}$

where

D_n = Active Damping Differential Term at sample n [% of maximum Control Output]

ΔF = Force Rate [Fr/sec]

J = Jerk [pu/sec^3]

K_{ADP} = Active Damping Differential Gain [%/(Fr/sec])] or [%/(pu/sec^3])]

See Also

Parameter Registers | Double Differential Output Term | Active Damping Differential Gain

9.2.3.4.18. Gain Sets

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.60/7-11, where *n* = 12 + the axis number **RMC150:** %MD*n*.60/7-11, where *n* = 24 + the axis number **RMC200:** %MD*n*.185, where *n* = 384 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].PriControlBits.GainSets, where *n* is the axis number **RMC200:** _Axis[*n*].GainSets, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Position/Velocity Control
Data Type:	**RMC75/150:** Bits **RMC200:** DINT
Range:	Single (0), "Pos, Vel" (1), "PID, I-PD" (2), Dual (3)
Default Value:	Single (0)

Description

This parameter specifies the Gain Sets options for the axis:

- **Single**
  This is sufficient for most applications.
- **Dual (RMC200 Only)**
  This is useful if you need to switch between two gains, for example, when switching from a high-flow valve to a low-flow valve.
- **Pos, Vel**
  The Pos, Vel option is useful if you are using both position and velocity control (Velocity PID or I-PD) on the same axis.
- **PID, I-PD**
  The PID, I-PD option is useful if you are using both a PID and and I-PD option on the same axis.

When the Gain Sets parameter is set to **Pos, Vel** or **PID, I-PD**, the RMC automatically switches between them according to the option. For example, if you select the **Pos, Vel** gain sets, the RMC will use Gain Set #0 when it is in position control (as defined by the Current Control Mode status register), and Gain Set #1 when it is in velocity control.

When the Gain Sets parameter is set to **Dual**, the Select Gain Set (75) command must be used to select the gain set.

See the Gain Sets Overview for a more detailed discussion.

Format Details

This section is primarily for addressing the Gain Sets parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

RMC75/150:

Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Value	Control Mode
0	0	0	0	0	0	Single
0	0	0	0	1	1	Pos, Vel
0	0	0	1	0	2	PID, I-PD

See the Primary Control Register topic for details about the register containing these bits.

RMC200:

Value	Control Mode
0	Single
1	Pos, Vel
2	PID, I-PD
3	Dual

See Also

Gain Sets Overview

9.2.3.4.19. Symmetrical/Ratioed

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.60.4, where *n* = 12 + the axis number **RMC150:** %MD*n*.60.4, where *n* = 24 + the axis number **RMC200:** %MD*n*.184, where *n* = 384 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].PriControlBits.Symm, where *n* is the axis number **RMC200:** _Axis[*n*].Symm, where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Position/Velocity Gains
Data Type:	**RMC75/150:** Bits **RMC200:** DINT
Range:	Ratioed (0), Symmetrical (1)
Default Value:	Ratioed (0)

Description

This parameter specifies whether to use ratioed or symmetrical gains. Ratioed gains are usually necessary on systems that behave differently in each direction of motion, such as hydraulic cylinders. Many systems, such as electric motors, usually do not require ratioed gains.

When this parameter is set to **Symmetrical**, the position or velocity gains are applied the same in both directions of motion. With the Symmetrical option, only one Velocity Feed Forward is available. If multiple gain sets are used, this parameter applies to both gain sets.

When this parameter is set to **Ratioed**, the position or velocity gains are applied differently in each direction of motion. With the Ratioed option, two Velocity Feed Forwards are available: Velocity Feed Forward (Positive) and Velocity Feed Forward (Negative). Once the Positive and Negative Velocity Feed Forwards have been tuned, the gains in the direction of the highest Velocity Feed Forward will be applied as they are. The gains in the direction of the lowest Velocity Feed Forward will be decreased by the ratio of the Velocity Feed Forwards. This results in very good control of the cylinder in both directions.

The RMC uses the Target Velocity and Position Error to determine whether to use the forward or reverse gains. If there is a non-zero Target Velocity, then its sign determines which directional gains to use. If the Target Velocity is zero, then the Position Error (or Velocity Error for Velocity Control) determines the intended direction of movement.

If the Symmetrical/Ratioed parameter is set to Ratioed and any of the Velocity Feed Forwards are set to zero, the gains will have a ratio of 1:1, and the Velocity Feed Forwards will be applied as they are.

Note:
Ratioed gains are automatically selected by the Tuning Wizard if moves in both directions are used.

Example

Consider a single-rod hydraulic cylinder that moves at 3 in/sec when a Control Output of 1V is applied and -2.3 in/sec when a Control Output of -1V is applied. Since it moves faster in the positive direction, those gains need to be smaller.

The correct Positive Velocity Feed Forward for this system is 3.33. The correct Negative Velocity Feed Forward is 4.35. By choosing Ratioed gains, the gains in the negative direction will be applied as they are. The gains in the positive direction will be multiplied by 3.33/4.35, which is 0.766.

Format Details

This section is primarily for addressing this parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

RMC75/150:

Bit 4	Control Mode
0	Ratioed
1	Symmetrical

See the Primary Control Register topic for details about the register containing these bits.

RMC200:

Value	Control Mode
0	Ratioed
1	Symmetrical

See Also

Ratioed Gains | Control Register

9.2.3.4.20. High-Order Control

Type:	Axis Parameter Register
Address:	**RMC75:** Gain Set #1: %MD*n*.71, where *n* = 12 + the axis number Gain Set #2: %MD*n*.138, where *n* = 12 + the axis number **RMC150:** Gain Set #1: %MD*n*.71, where *n* = 24 + the axis number Gain Set #2: %MD*n*.138, where *n* = 24 + the axis number **RMC200:** Gain Set #1: %MD*n*.214, where *n* = 384 + the axis number Gain Set #2: %MD*n*.244, where *n* = 384 + the axis number
System Tag:	Gain Set #1: _Axis[*n*].AccCtrlMode, where *n* is the axis number Gain Set #2: _Axis[*n*].AccCtrlMode_2, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Position/Velocity Control → Gain Set#
Data Type:	DINT
Range:	None (0), Acceleration Control (1), and Active Damping (2)
Default Value:	None (0)

Description

This parameter specifies the type of high-order control to use on the position or velocity control axis.

Options:

- **None**
  No high-order control is used.
- **Acceleration Control**
  This option adds the higher-order Double Differential Gain and Triple Differential Gain to the PID or I-PD algorithm. For axes with a secondary Acceleration Input, the acceleration and jerk from this input will be used instead of the Acceleration Input derived from the position or velocity input. See the Acceleration Control topic for details.
- **Active Damping**
  This option adds the higher-order Active Damping Proportional Gain and Active Damping Differential Gain. This option is useful for controlling systems with low damping that tend to oscillate. For axes with a secondary Force Input, the active damping gains will be applied to the force input. Similarly, for axes with a secondary Acceleration Input, the gains will apply to the acceleration and jerk from this input instead of the Acceleration Input derived from the position or velocity input. See the Active Damping topic for details.

Values	Control Mode
0	None
1	Acceleration Control
2	Active Damping

See Also

Acceleration Control | Active Damping

9.2.3.4.21. Default Pos/Vel Control Mode

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.43, where *n* = 12 + the axis number **RMC150:** %MD*n*.43, where *n* = 24 + the axis number **RMC200:** %MD*n*.171, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].DefPVCtrlMode, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Position/Velocity Control
Data Type:	DINT
Range:	Pos PID (0), Pos I-PD (1), Vel PID (4), Vel I-PD (5)
Default Value:	Pos PID (0) for Position Control Axes Vel PID (4) for Velocity Control Axes

Description

The Default Pos/Vel Control Mode parameter determines which control mode the Next Pos/Vel Control Mode for that axis will be set to when the RMC powers up. The Next Pos/Vel Control Mode register determines which control mode will be applied to the next position or velocity motion command that is issued. Use the Set Pos/Vel Ctrl Mode (68) command to change the Next Pos/Vel Control Mode register during machine control without affecting the power-up mode.

Values	Control Mode
0	Position PID
1	Position I-PD
4	Velocity PID
5	Velocity I-PD

See Also

Parameter Registers | Current Control Mode | Next Pos/Vel Control Mode

9.2.3.4.22. Primary Control Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.60, where *n* = 12 + the axis number **RMC150:** %MD*n*.60, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	_Axis[*n*].PriControlBits, where *n* is the axis number
How to Find:	See individual parameters below
Data Type:	DWORD - see below

Description

The RMC75/150 Control Configuration register contains the bit-addressable Control configuration parameters. This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Bits

This register contains the following parameters. The bits for each are given in the right-hand column.

Parameter	Tag Name	Bit Number(s)
Default Integrator Mode	IntMode	0-3
Symmetrical/Ratioed	Symm	4
Pressure/Force Orientation	PFOrientation	5-6
Gain Sets	GainSets	7-11
Unidirectional Mode	UniMode	12-14

For details on the values that each bit represents, see the respective parameter topic.

See Also
Parameter Registers

9.2.3.5. Pressure/Force Control

9.2.3.5.1. At Pressure/Force Tolerance

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Control Axis: %MD*n*.56, where *n* = 12 + the axis number Secondary Control Axis: %MD*n*.76, where *n* = 12 + the axis number **RMC150:** Primary Control Axis: %MD*n*.56, where *n* = 24 + the axis number Secondary Control Axis: %MD*n*.76, where *n* = 24 + the axis number **RMC200:** Primary Control Axis: %MD*n*.180, where *n* = 384 + the axis number Secondary Control Axis: %MD*n*.290, where *n* = 384 + the axis number
System Tag:	**Pressure Axis:** _Axis[*n*].AtPrsTolerance, where *n* is the axis number **Force Axis:** _Axis[*n*].AtFrcTolerance, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL
Units:	Pr or Fr
Range:	≥ 0
Default Value:	10.0

Description

This parameter specifies a tolerance around the Command Pressure or Command Force. In pressure/force control or pressure/force limit , when the Target Pressure/Force is at the Command Pressure/Force, and the Actual Pressure/Force is within this tolerance, the At Pressure/Force Status bit is set. The At Pressure/Force bit is not latched and will be cleared if the axis moves outside the At Pressure/Force Tolerance.

See Also
Parameter Registers | At Pressure/Force

9.2.3.5.2. Pressure/Force Error Tolerance

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Control Axis: %MD*n*.56, where *n* = 12 + the axis number Secondary Control Axis: %MD*n*.76, where *n* = 12 + the axis number **RMC150:** Primary Control Axis: %MD*n*.56, where *n* = 24 + the axis number Secondary Control Axis: %MD*n*.76, where *n* = 24 + the axis number **RMC200:** Primary Control Axis: %MD*n*.181, where *n* = 384 + the axis number Secondary Control Axis: %MD*n*.291, where *n* = 384 + the axis number
System Tag:	**Pressure Axis:** _Axis[*n*].PrsErrorTolerance, where *n* is the axis number **Force Axis:** _Axis[*n*].FrcErrorTolerance, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL
Units:	Pr or Fr
Range:	≥ 0
Default Value:	100.0

Description

This parameter specifies how large the Pressure/Force Error may become before the Pressure/Force Following Error bit is set. The axis must be in pressure/force control or pressure/force limit for the Pressure/Force Following Error bit to be set. If the Pressure/Force Following Error bit is set, a Halt will occur if the Pressure/Force Following Error Auto Stop is configured to do so and the Direct Output Status bit is off.

See Also

Parameter Registers | Pressure/Force Following Error

9.2.3.5.3. Pressure/Force Proportional Gain

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Control Axis: %MD*n*.61, where *n* = 12 + the axis number Secondary Control Axis: %MD*n*.81, where *n* = 12 + the axis number **RMC150:** Primary Control Axis: %MD*n*.61, where *n* = 24 + the axis number Secondary Control Axis: %MD*n*.81, where *n* = 24 + the axis number **RMC200:** Primary Control Axis: %MD*n*.200, where *n* = 384 + the axis number Secondary Control Axis: %MD*n*.310, where *n* = 384 + the axis number

System Tag:	**Pressure Axis:** _Axis[*n*].PrsPropGain, where *n* is the axis number **Force Axis:** _Axis[*n*].FrcPropGain, where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Pressure/Force Tuning
Data Type:	REAL
Units:	%/(Pr) or %/(Fr)
Range:	≥ 0
Default Value:	RMC75/150: 1.0 RMC200: 0.001

Description

The Pressure/Force Proportional Gain generates Control Output proportional to the Pressure/Force Error. The Proportional Gain is the most important pressure or force tuning parameter. It affects the responsiveness of the system. A low gain makes the system sluggish and unresponsive. A gain that is too high makes the axis oscillate or vibrate.

The Pressure/Force Proportional Term is the portion of the Control Output resulting from the Pressure/Force Proportional gain.

Advanced: Pressure/Force I-PD

In pressure/force I-PD, the Pressure/Force Proportional Gain adjusts the Control Output based on the change in the Actual Pressure or Actual Force. The Pressure/Force I-PD is used in pressure/force limit on dual-loop axes with I-PD position control or I-PD velocity control.

Mathematical Definition

Pressure/Force PID

$D_n = (P_{Target}n - P_{Actual}n) \times K_P$

Pressure/Force I-PD

$D_n = (P_{Actual}n - P_{Actual}n\text{-}1) \times K_P$

where

D_n = Proportional Term at sample n [% of maximum Control Output]

P = Pressure or Force [Pr or Fr]

K_P = Pressure/Force Proportional Gain [%/(Pr) or %/(Fr)]

The Pressure/Force Proportional Gain units are: percent of the maximum Control Output per pressure or force unit. The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

See Also

Parameter Registers | Pressure/Force Proportional Term

9.2.3.5.4. Pressure/Force Integral Gain

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Control Axis: %MD*n*.62, where *n* = 12 + the axis number Secondary Control Axis: %MD*n*.82, where *n* = 12 + the axis number **RMC150:** Primary Control Axis: %MD*n*.62, where *n* = 24 + the axis number

	Secondary Control Axis: %MD*n*.82, where *n* = 24 + the axis number **RMC200:** Primary Control Axis: %MD*n*.201, where *n* = 384 + the axis number Secondary Control Axis: %MD*n*.311, where *n* = 384 + the axis number
System Tag:	**Pressure Axis:** _Axis[*n*].PrsIntGain, where *n* is the axis number **Force Axis:** _Axis[*n*].FrcIntGain, where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Pressure/Force Tuning
Data Type:	REAL
Units:	%/(Pr·s) or %/(Fr·s)
Range:	≥ 0
Default Value:	0

Description

The Pressure/Force Integral Gain generates Control Output proportional to the Pressure/Force Error. The Integral Gain helps compensate for changing system dynamics, such as varying loads, and often aids the axis in rapidly reaching the Command Pressure/Force. A low gain slows the response of the axis to changes and may keep the axis from reaching the Command Position. A gain that is too high may make the axis oscillate.

Each gain of the axis contributes to the Control Output. The contribution from the Pressure/Force Integral Gain is called the Pressure/Force Integral Term. For both PID and I-PD, each control loop, the Integral Gain is multiplied by the control loop time and Pressure/Force Error and added to the Pressure/Force Integral Output Term.

Integrator Adjust

The Integral Output Term can be manually adjusted with the Integrator Adjust (70) command. This is useful for clearing it if it has "wound up" for some reason. See the Integrator Adjust (70) topic for more details.

Advanced: Pressure/Force I-PD

In pressure/force I-PD, this gain is the most important gain for pressure/force I-PD control. This gain must be non-zero in I-PD mode. If it is zero, it will cause a run-time error. A higher value will increase the response. A value that is too high will cause oscillation and instability. The Pressure/Force I-PD is used in pressure/force limit on dual-loop axes with I-PD position control or I-PD velocity control.

Mathematical Definition

PID and I-PD

The Pressure/Force Integral Gain is calculated from the Pressure/Force Integral Gain as follows:

$$I_n = (e_n \times K_i \times T) + I_{n-1}$$

where

I_n = Pressure/Force Integral Term [% of maximum Control Output]

e_n = Pressure/Force Error at sample n [Pr or Fr]

K_i = Pressure/Force Integral Gain [%/(Pr x sec) or %/(Fr x sec)]

T = Time period of control loop [sec]

The Integral Gain units are: percent of the maximum Control Output per pressure or force units times time. The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

See Also

Parameter Registers | Pressure/Force Integral Term

9.2.3.5.5. Pressure/Force Differential Gain

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Control Axis: %MD*n*.63, where *n* = 12 + the axis number Secondary Control Axis: %MD*n*.83, where *n* = 12 + the axis number **RMC150:** Primary Control Axis: %MD*n*.63, where *n* = 24 + the axis number Secondary Control Axis: %MD*n*.83, where *n* = 24 + the axis number **RMC200:** Primary Control Axis: %MD*n*.200, where *n* = 24 + the axis number Secondary Control Axis: %MD*n*.310, where *n* = 24 + the axis number
System Tag:	**Pressure Axis:** _Axis[*n*].PrsDiffGain, where *n* is the axis number **Force Axis:** _Axis[*n*].FrcDiffGain
How to Find:	Axes Parameters Pane, Tune tab: Pressure/Force Tuning
Data Type:	REAL
Units:	%/(Pr/s) or %/(Fr/s)
Range:	≥ 0
Default Value:	0

Description

The Pressure/Force Differential Gain generates Control Output proportional to the difference between the Target Pressure/Force Rate and the Actual Pressure/Force Rate. The Differential Gain helps the pressure or force track quick changes.

The Pressure/Force Differential Term is the portion of the Control Output resulting from the Pressure/Force Differential gain.

Advanced: Pressure/Force I-PD

In Pressure/Force I-PD, the Pressure/Force Differential Gain adjusts the Control Output based on the change in the Actual Pressure/Force Rate. The Pressure/Force I-PD is used in pressure/force limit on dual-loop axes with I-PD Position or Velocity Control.

Mathematical Definition

Pressure/Force PID

$$D_n = (PR_{Target}n - PR_{Actual}n) \times K_D$$

Pressure/Force I-PD

$$D_n = (PR_{Actual}n - PR_{Actual}n\text{-}1) \times K_D$$

where

D_n = Pressure/Force Differential Term at sample n [% of maximum Control Output]

PR = Pressure or Force Rate [Pr/sec or Fr/sec]

K_P = Pressure/Force Proportional Gain [%/(Pr/sec) or %/(Fr/sec)]

The Pressure/Force Differential Gain units are: percent of the maximum Control Output per pressure or force units per second. The maximum Control Output is 10V, but can be changed using the Output Scale parameter.

See Also

Parameter Registers | Pressure/Force Differential *term*

9.2.3.5.6. Pressure/Force Feed Forward

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Control Axis: %MD*n*.64, where *n* = 12 + the axis number Secondary Control Axis: %MD*n*.84, where *n* = 12 + the axis number **RMC150:** Primary Control Axis: %MD*n*.64, where *n* = 24 + the axis number Secondary Control Axis: %MD*n*.84, where *n* = 24 + the axis number **RMC200:** Primary Control Axis: %MD*n*.205, where *n* = 384 + the axis number Secondary Control Axis: %MD*n*.315, where *n* = 384 + the axis number
System Tag:	**Pressure Axis:** _Axis[*n*].PrsFFwd, where *n* is the axis number **Force Axis:** _Axis[*n*].FrcFFwd, where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Pressure/Force Tuning
Data Type:	REAL
Units:	%/(Pr) or %/(Fr)
Range:	≥ 0
Default Value:	0

Description

The Pressure/Force Feed Forward gives extra Control Output proportional to the Target Pressure or Target Force. This feed forward is useful on systems where the actual pressure or force is proportional to the Control Output, such as a pressure control or relief valve. This feed forward is typically not used with systems employing a 4-way proportional valve.

The Pressure/Force Feed Forward Term is the portion of the Control Output resulting from this gain.

Advanced: Pressure/Force I-PD

In pressure/force I-PD, the Pressure/Force Feed Forward will not be applied. The Pressure/Force I-PD is used in pressure/force limit on dual-loop axes with I-PD position control or I-PD velocity control.

See Also

Parameter Registers | Pressure/Force Feed Forward Term

9.2.3.5.7. Pressure/Force Rate Feed Forward

Type:	Axis Parameter Register
Address:	**RMC75:**

	Primary Control Axis: %MD*n*.65, where *n* = 12 + the axis number Secondary Control Axis: %MD*n*.85, where *n* = 12 + the axis number **RMC150:** Primary Control Axis: %MD*n*.65, where *n* = 24 + the axis number Secondary Control Axis: %MD*n*.85, where *n* = 24 + the axis number **RMC200:** Primary Control Axis: %MD*n*.206, where *n* = 384 + the axis number Secondary Control Axis: %MD*n*.316, where *n* = 384 + the axis number
System Tag:	**Pressure Axis:** _Axis[*n*].PrsRateFFwd, where *n* is the axis number **Force Axis:** _Axis[*n*].FrcRateFFwd, where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab: Pressure/Force Tuning
Data Type:	REAL
Units:	%/(Pr/s) or %/(Fr/s)
Range:	≥ 0
Default Value:	0

Description

The Pressure/Force Rate Feed Forward gives extra Control Output proportional to the rate of change in the Target Pressure or Target Force. It helps the axis maintain the target when the Target Pressure or Force is changing.

Pressure/Force Rate Feed Forward is applied in both pressure/force control and in pressure/force limit.

The Pressure/Force Rate Feed Forward Term is the portion of the Control Output resulting from this gain.

Advanced: Pressure/Force I-PD

In pressure/force I-PD, the Pressure/Force Rate Feed Forward will not be applied. The Pressure/Force I-PD is used in pressure/force limit on dual-loop axes with I-PD position control or I-PD velocity control.

See Also

Parameter Registers | Pressure/Force Rate Feed Forward Term

9.2.3.5.8. Pressure/Force Orientation

Type:	Axis Parameter Register
Address:	**RMC75:** Primary Control Axis: %MD*n*.60/5-6, where *n* = 12 + the axis number Secondary Control Axis: %MD*n*.80/5-6, where *n* = 12 + the axis number **RMC150:** Primary Control Axis: %MD*n*.60/5-6, where *n* = 24 + the axis number Secondary Control Axis: %MD*n*.80/5-6, where *n* = 24 + the axis number **RMC200:**

	Primary Control Axis: %MD*n*.186, where *n* = 384 + the axis number Secondary Control Axis: %MD*n*.296, where *n* = 384 + the axis number
System Tag:	**RMC75:** Primary Control Axis: _Axis[*n*].PriControlBits.PFOrientation Secondary Control Axis: _Axis[*n*].SecControlBits.PFOrientation **RMC200:** _Axis[*n*].PFOrientation
How to Find:	Axes Parameters Pane, Tune tab: Pressure/Force Tuning
Data Type:	**RMC75/150:** Bits **RMC200:** DINT
Range:	Same (0), Opposite (1), Bidirectional (2)
Default Value:	Same (0)

Description

This register defines the orientation of the pressure/force relative to the orientation of the position or velocity when in closed-loop control, or of the Control Output when in open-loop control. It is important that this parameter is set correctly so that the Control Output will be the correct polarity.

For Closed-Loop Control

Orientation	Description
Same	The pressure/force increases as the position units increase.
Opposite	The pressure/force decreases as the position units increase.

For Open-Loop Control

Orientation	Description
Same	The pressure/force increases with positive Control Output.
Opposite	The pressure/force decreases with positive Control Output.

Changing this Parameter

Because this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Enabled and Direct Output Status Bits indicate these states of the axis. When changing this parameter from RMCTools, the software will automatically do this for you. This may cause a halt on the axis, which is expected.

Format Details

RMC75/150:

The Pressure/Force Orientation is selected with bits 5-6. These bits correspond to the Pressure/Force Orientation as shown in the following table:

Bit 6	Bit 5	Value	Pressure/Force Orientation
0	0	0	Same
0	1	1	Opposite

See the Primary Control Register and Secondary Control Register topics for details about the register containing these bits.

RMC200:

The Pressure/Force Orientation values are as follows:

Value	Pressure/Force Orientation
0	Same

1	Opposite

See Also
Parameter Registers | Pressure/Force Control Overview

9.2.3.5.9. Secondary Control Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.80, where *n* = 12 + the axis number **RMC150:** %MD*n*.80, where *n* = 24 + the axis number **RMC200:** n/a
System Tag:	_Axis[0].SecControlBits, where *n* is the axis number
How to Find:	See individual parameters below
Data Type:	DWORD - see below

Description

The RMC75/150 Control Configuration register contains the bit-addressable control configuration parameters. This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Bits

This register contains the following parameters. The bits for each are given in the right-hand column.

Parameter	Tag Name	Bit Number(s)
Pressure/Force Orientation	PFOrientation	5-6

For details on the values that each bit represents, see the respective parameter topic.

See Also
Parameter Registers

9.2.3.6. Output

9.2.3.6.1. Invert Output Polarity

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.34.0, where *n* = 12 + the axis number **RMC150:** %MD*n*.34.0, where *n* = 24 + the axis number **RMC200:** %MD*n*.140.8, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].OutputBits.InvertOutPol, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	bit
Range:	Not Inverted (0), Inverted (1)

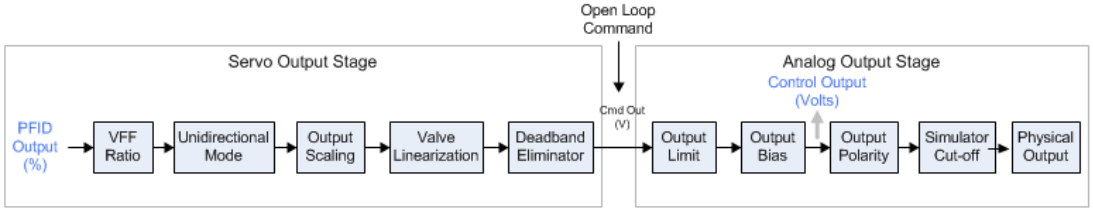
Default Value:	Not Inverted (0)

Description

The Invert Output Polarity parameter bit specifies the polarity of the Control Output. For closed-loop motion control, it is imperative that the output polarity is correct. A positive Control Output must move the axis in the direction of increasing Actual Position. The Invert Output Polarity register can be used to set this. If the Invert Output Polarity bit is cleared, the physical voltage on the Control Output or Drive will be of the same polarity as the Control Output. If the Invert Output Polarity bit is set, the Control Output pins of the RMC module will be the negative of the voltage of the Control Output.

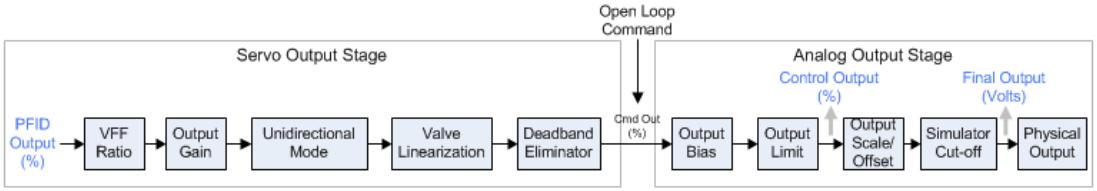
RMC75/150:

The Invert Output Polarity is applied in the Output Polarity block below.



RMC200:

The Invert Output Polarity is applied in the Output Scale/Offset block below. The Invert Output Polarity applies only when one of the standard Output Types is selected. For a custom output type, inverting the output is done by swapping the Output at 100% and Output at -100% axis parameter values.



Changing the Invert Output Polarity

Since this parameter affects motion, the axis must be disabled or in Direct Output before writing to this parameter. The Status Bits indicate the state of the axis.

Why Bother?

If you wire up the system and find out that a positive Control Output moves the axis in the negative direction, instead of rewiring the Control Output, you can just set this bit. This is often useful on motors, when it isn't always obvious which way it should be wired.

See Also

Parameter Registers | Control Output | Output Register

9.2.3.6.2. Output Bias

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.33, where *n* = 12 + the axis number
	RMC150: %MD*n*.33, where *n* = 24 + the axis number

	RMC200: %MD*n*.146, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].OutputBias, where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab
Data Type:	REAL
Units:	**RMC75/150:** V **RMC200: %**
Range:	**RMC75/150:** -10 to 10 **RMC200:** -100 to 100
Default Value:	0

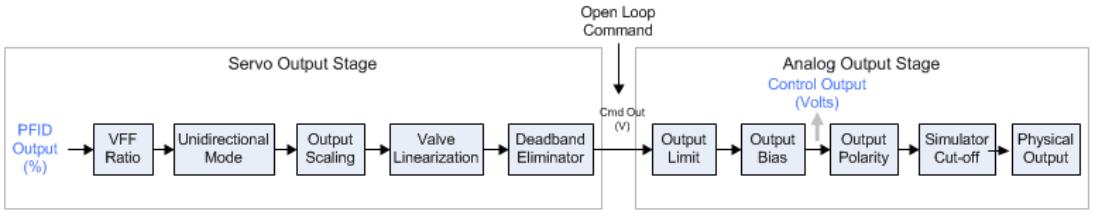
Description

The Output Bias voltage is always added to the Control Output. Use the Output Bias to compensate for hydraulic valves or other systems that have an offset. To set the Output Bias, use the Open Loop Rate (10) command to find out how much Control Output is required to keep the axis from moving. This is called the null drive. Set the Output Bias parameter to that value.

Note:
There may also be a null or offset adjustment on the valve or amplifier that can be adjusted so the null drive is 0.

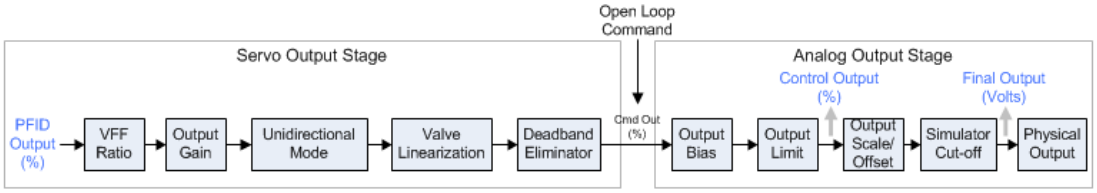
RMC75/150:

The Output Bias is in units of Volts and is applied after the Output Limit.



RMC200:

The Output Bias is in units of percent and is applied before the Output Limit.



See Also

Parameter Registers | Control Output

9.2.3.6.3. Output Deadband

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.42, where *n* = 12 + the axis number **RMC150:** %MD*n*.42, where *n* = 24 + the axis number

	RMC200: %MD*n*.152, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].OutputDeadband, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	**RMC75/150:** V **RMC200:** %
Range:	**RMC75/150:** 0 to 10 V **RMC200:** 0 to 100 %
Default Value:	0 (disabled)

Description

The Output Deadband is the amount added or subtracted from the Control Output to compensate for a "deadband" in the system. Valves with overlapped spools, and some drives do not react to small changes in output; this effect is termed "dead band". Deadband compensation can be performed in closed-loop control only. No deadband compensation is performed in open-loop control.

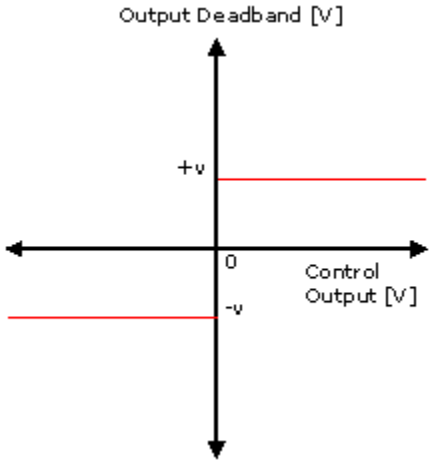
The Output Deadband works together with the Deadband Tolerance parameter. These are essential tuning parameters if the system exhibits a deadband. Care must be taken to not make this value too large or the axis may oscillate. Normally, the Output Deadband value can be determined from the valve datasheet, which indicates how large the overlap is.

Many systems do not require any deadband compensation.

How it Works

The Output Deadband is added to or subtracted from the Control Output (depending on its sign) so that the Control Output is outside the dead band.

- If the Control Output is positive, the Output Deadband (v) is added to the Control Output.
- If the Control Output is negative, the Output Deadband (v) is subtracted from the Control Output.



The Output Deadband also works together with the following components:

- **Deadband Tolerance parameter**
  If the Position Error is less than the Deadband Tolerance, then only a fraction of the Control Output, proportional to the Position Error, is applied. This helps keep the axis from chattering. See the Deadband Tolerance topic for details.
  The Deadband Tolerance has no effect in Velocity PID or Velocity I-PD control. The Deadband Output is applied based on the direction, but always at 100% in one direction or the other.

- **Integrator**
  The Output Deadband also affects the functionality of the Integral Term. If the target is stopped and the position error is less than the Deadband Tolerance window or less than the position corresponding to one half of a transducer Raw Count, then the integrator is only allowed to wind down. This reduces "hunting". See the Integral Term and Integral Gain topics for more details.

Determining the Output Deadband

Follow these steps to determine what the Output Deadband should be set to:

1. Give increasing amounts of Control Output to the system with the Direct Output (9) command until the system starts to move. Start with a Requested Output of 0 V. The value of Control Output at which the system starts to move is your deadband.
2. If the deadband is less than a few hundred millivolts (approx. 0.4V), your system probably does not need the deadband eliminator. Set the Output Deadband to 0.
3. If the deadband is greater than a few hundred millivolts (approx. 0.4V), you probably need the deadband eliminator. Set the Output Deadband to the value of your deadband.

Control Modes

The Deadband compensation applies to all the closed-loop control modes: Position PID, Position I-PD, Velocity PID, Velocity I-PD.

Velocity Control

The Output Deadband applies to velocity control axes. If the Deadband Tolerance value is non-zero, then the Output Deadband will not apply if the Target Pressure or Velocity is zero. Otherwise, the value of the Deadband Tolerance has no meaning.

Pressure/Force Control

The Output Deadband applies to pressure/force control. If the Deadband Tolerance value is non-zero, then the Output Deadband will not apply if the Target Pressure or Target Force is zero. Otherwise, the value of the Deadband Tolerance has no meaning.

Pressure/Force Limit on Position-Pressure or Position-Force Axes

The Output Deadband is applied as normal, based on position. The Deadband Tolerance is applied based on the position as normal. Notice that if the axis is pressure/force limited, the position may be outside the tolerance window, and the full compensation will apply.

The Deadband Tolerance controls the integrator for the position component. If the target is stopped and the position error is less than the window then the position of the integrator is only allowed to wind down. When the axis is pressure/force limited then the position integrator is not used and the Deadband Tolerance has no effect.

Why Bother?

On an axis with overlapped spools or a large amount of static friction, the axis may oscillate slowly around the Command Position. This happens when the axis does not respond to small changes in Control Output. The integrator winds up until the axis moves and then has too much Control Output, overshooting the Command Position. The axis then winds the integrator down until the axis overshoots the other way.

If this happens, look at the Control Output. The Control Output will oscillate slowly as the axis oscillates. Notice the peak-to-peak values of the Control Output oscillation and subtract the minimum value from the maximum value and then divide the result by two. Enter this value in the Output Deadband register as a starting point.

A spool that has 10% overlap will require a value of about 1 Volt.

See Also

Parameter Registers | Deadband Tolerance

9.2.3.6.4. Deadband Tolerance

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.41, where *n* = 12 + the axis number **RMC150:** %MD*n*.41, where *n* = 24 + the axis number **RMC200:** %MD*n*.151, where *n* = 24 + the axis number
System Tag:	_Axis[*n*].DeadbandTolerance, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	pu
Range:	≥ 0
Default Value:	0

Description

In closed-loop position control (Position PID or Position I-PD), the Deadband Tolerance parameter works together with the Output Deadband to reduce "hunting" and ratio the deadband for smoother motion. The Deadband Tolerance has no effect on control in open-loop or velocity (Velocity PID or Velocity I-PD) control.

The Deadband Tolerance affects two areas:

- **Anti-Hunting**
  When the target is stopped, and the Position Error is less than the Deadband Tolerance or less than the position corresponding to one half of a transducer Raw Count, then the Integral Output Term will only be allowed to wind down. In systems with a deadband, this avoids ratcheting the axis back and forth around the deadband. Notice that this works even if the Output Deadband is 0. If the Deadband Tolerance is zero, this still works with the window defined by the position corresponding to one half of a transducer Raw Count.
- **Deadband Ratioing**
  The Deadband Tolerance specifies a Position Error window inside of which only a fraction of the Output Deadband is applied. This minimizes chattering, which might otherwise be caused by the Output Deadband. If the Position Error is greater than the Deadband Tolerance, the Output Deadband is applied normally.

  Deadband Ratioing works as shown in the graph below:
  - If the Deadband Output is non-zero, the target velocity is zero, and the absolute value of the Position Error is less than the Deadband Tolerance, only a fraction of the Output Deadband, proportional to the Position Error, is applied, as shown in the graph below.
  - If the absolute value of the Position Error is greater than the Deadband Tolerance, the full Output Deadband is applied, as shown in the graph below.

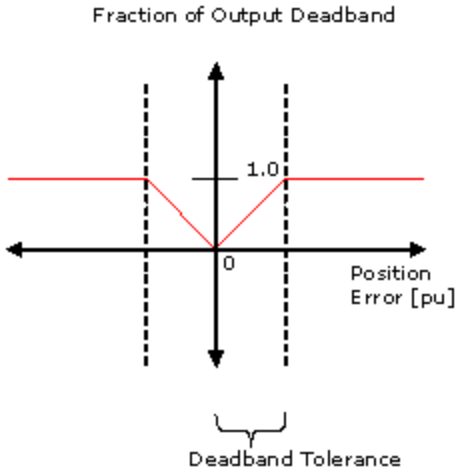


Figure 1: Deadband Tolerance

Control Modes

The Deadband compensation applies to all the closed-loop control modes: Position PID, Position I-PD, Velocity PID, Velocity I-PD.

Pressure/Force Limit on Position-Pressure or Position-Force Axes

The Output Deadband is applied as normal, based on position. The Deadband Tolerance controls the integrator for the position component. If the target is stopped and the position error is less than the window then the position of the integrator is only allowed to wind down. When the axis is pressure/force limited then the position integrator is not used and the Deadband Tolerance has no effect.

Pressure-Only or Force-Only Axes

The Output Deadband applies to pressure/force control. If the Deadband Tolerance value is non-zero, then the Output Deadband will not apply if the Target Pressure or Target Force is zero. Otherwise, the value of the Deadband Tolerance has no meaning.

Velocity Control

The Output Deadband applies to velocity control. If the Deadband Tolerance value is non-zero, then the Output Deadband will not apply if the Target Velocity is zero. Otherwise, the value of the Deadband Tolerance has no meaning.

See Also

Parameter Registers | Output Deadband

9.2.3.6.5. Output Filter

Type:	Axis Parameter Register
Address:	**RMC75:**
	Primary Input: %MD*n*.39, where *n* = 12+ the axis number
	Secondary Input: %MD*n*.40, where *n* = 12+ the axis number
	RMC150:
	Primary Input: %MD*n*.39, where *n* = 24+ the axis number

	Secondary Input: %MD*n*.40, where *n* = 24+ the axis number **RMC200:** Primary Input: %MD*n*.222, where *n* = 384+ the axis number Secondary Input: %MD*n*.332, where *n* = 384+ the axis number
System Tag:	**RMC75/150:** Primary Input: _Axis[*n*].OutputFilterFreq, where *n* is the axis number Secondary Input: _Axis[*n*].PFOutputFilter, where *n* is the axis number **RMC200:** Primary Input: _Axis[*n*].OutputPDFilter, where *n* is the axis number Secondary Input: _Axis[*n*].PFOutputFilter, where *n* is the axis number
How to Find:	Axes Parameters Pane, Tune tab
Data Type:	REAL
Units:	Hz
Range:	0, ≥ 0.01
Default Value:	0

Description

This parameter specifies the cut-off frequency of the low-pass single pole filter that is applied to the output. This filter is applied only to the Proportional, Differential, Double Differential, and Triple Differential gains in closed-loop control.

Setting the Output Filter to zero (0) disables the filter.

Dual-loop axes have an output filter for each loop, that is, one for the primary control and one for the secondary control.

Use for Difficult Systems

The Output Filter can significantly improve control of difficult systems. Without the Output Filter, the Differential Gain can cause the Control Output to oscillate, causing oscillation of the axis. By using the output filter, the differential gain can be increased significantly to help the Actual Velocity track the Target Velocity.

Typically, the Output Filter can be set to a value close top the natural frequency of the system. For example, if a system tends to oscillate at 10 Hz, a good starting value for the Output Filter is 10 or higher.

See Also

Parameter Registers | Filtering

9.2.3.6.6. Output Limit

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.32, where *n* = 12 + the axis number **RMC150:** %MD*n*.32, where *n* = 24 + the axis number **RMC200:** %MD*n*.144, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].OutputLimit, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	**RMC75/150:** V

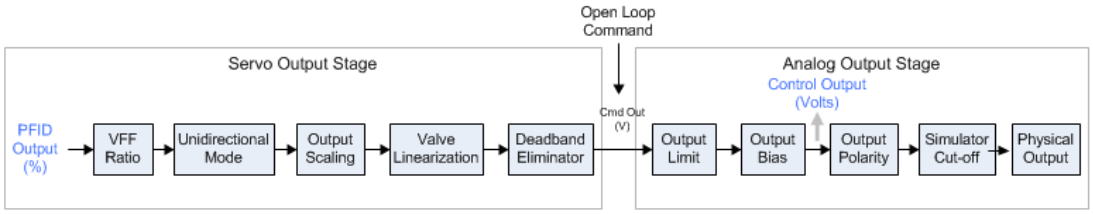
	RMC200: %
Range:	**RMC75/150:** 0 to 10 V
	RMC200: 0 to 100 %
Default Value:	**RMC75/150:** 10 V
	RMC200: 100 %

Description

This parameter determines the maximum Control Output. When the Control Output reaches the Output Limit (positive or negative), it will not increase (or decrease) further and, if the axis is in closed loop control, the Output Saturated error bit will be set.

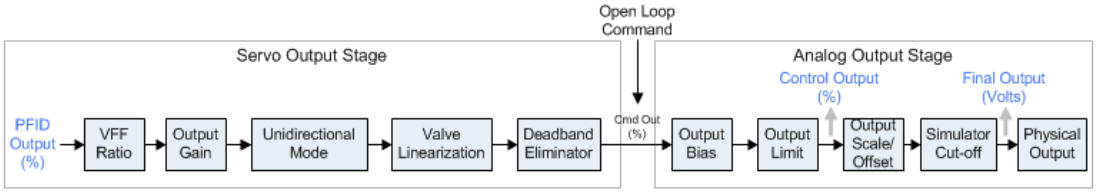
RMC75/150:

The Output Limit is in units of volts, and is applied before the Output Bias. The actual maximum voltage measured on the Control Output pins is the Output Limit *plus* the Output Bias.



RMC200:

The Output Limit is in units of percent, and is applied immediately before the conversion from percent to the physical units, such as V or mA. For the standard Output Types, the Output Limit is applied in both the positive and negative direction. To apply asymmetric limits, choose a custom Output Type and use the Positive Output Limit and Negative Output Limit axis parameters.



See Also

Parameter Registers | Control Output | Output Saturated Error bit

9.2.3.6.7. Output Scale

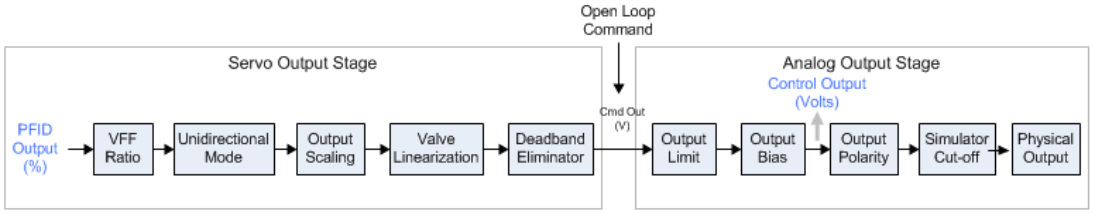
Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.38, where *n* = 12 + the axis number
	RMC150: %MD*n*.38, where *n* = 24 + the axis number
	RMC200: n/a
System Tag:	_Axis[*n*].OutputScale, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output

Data Type:	REAL
Units:	V/100%
Range:	<> 0
Default Value:	10

Description

The Output Scale on the RMC75/150 scales the Control Output to the actual Control Output voltage. This scale is defined as volts per 100% of Control Output. For most systems, this value should be left at its default of 10.

RMC75/150 Output Diagram:



The RMC200 output flow is different from the RMC75/150 and does not use the Output Scale parameter. The RMC200 does use the related Output Gain parameter.

There are three main reasons for setting this parameter to a value other than 10:

1. **Output Range Less Than ±10V**
   For systems that require a Control Output range of less than ± 10 V, Output Scale should be set to the maximum required Control Output voltage. The Output Limit parameter should also be set to the same value.
   For example:
   For a system requiring a Control Output range of only ±5V, set this parameter to 5. The Output Limit parameter should also be set to 5V so that the Control Output will never exceed ±5V. Thus, you can work with the full scale output range and won't have to worry about the Control Output exceeding ±5V.
2. **Convert ±10V to 4-20 mA**
   The Output Scale and Output Bias parameters can be used on conjunction with the VC2124 voltage-to-current converter to convert the Control Output from ±10V to 4-20 mA. Delta does not recommend using valves or drives that require a 4-20mA input. Delta recommends using a valve or drive that requires a ±10V input voltage or, for servo valves, a current centered around 0, such as ±90mA. However, for the cases where 4-20mA cannot be avoided, the Using the VC2124 topic explains how to set up the RMC parameters to work together with the VC2124 to produce a 4-20mA output.
3. **Compensate for Oil Temperature Changes**
   The system gain may change with temperature for hydraulic system, requiring different values of gains and feed forwards. Changing the Output Scale parameter will in effect scale the gains. The Output Scale parameter provides a simple method to scale the gains and feed forwards to compensate for oil temperature changes.

See Also

Parameter Registers | Control Output | Output Limit | Output Bias | Output Gain

9.2.3.6.8. Output Type

Type:	Axis Parameter Register

Address:	**RMC75:** n/a **RMC150:** /a **RMC200:** %MD*n*.140.0-3, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].OutputType, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	DINT
Range:	Bits - see below
Default Value:	**±10V**

Description

The RMC200 Output Type axis parameter defines the type of the Final Output and how the Control Output percentage range maps to the Final Output range. The RMC200 Control Output is very flexible. If the standard options do not match the application, choose a custom option to define your own output range and limits.

Which output types are available depends on the features of the module on which the physical output is located. The CA4 and U14 modules support both voltage and current ranges. The CV8 module supports only voltage ranges.

This axis parameter is not available on the RMC75/150.

Standard Output Types

The standard output types define a preset range and direction. The Invert Output Polarity and the Output Limit parameters may be used together with these types.

- **±10V**
  The most common output type. This output type for systems that will control in both directions and assumes that a Control Output signal of 0 V is approximately zero control signal, a positive voltage moves the axis one direction, and a negative voltage moves it in the other direction.
- **0-10V unidirectional**
  To be used when the Control Output signal will be within the range 0-10 V and will only regulate the motion in one direction. On such systems, if the direction is to be changed at all, it is usually done with a separate directional valve. See Unidirectional Mode for more details.
- **±20mA**
  Typically for use with servo valves that take a ±20mA input. This output type is for systems that will control in both directions and assumes that a Control Output signal of 0 mA is approximately zero control signal, positive current moves the axis one direction, and negative current moves it in the other direction.
  Available only on the CA4 and U14 modules.
- **4-20mA bidirectional**
  This output type is for systems that will control in both directions and assumes that a Control Output signal of 12 mA is approximately zero control signal, positive current moves the axis one direction, and negative current moves it in the other direction.
  Available only on the CA4 and U14 modules.
- **4-20mA unidirectional**
  To be used when the 4-20 mA Control Output signal will regulate the motion in only one direction, where 4 mA is assumed to be zero control signal. On such systems, if the direction is to be changed at all, it is done with a separate directional valve. See Unidirectional Mode for more details.
  Available only on the CA4 and U14 modules.
- **±100%**
  Available on Outer Loop axes only.

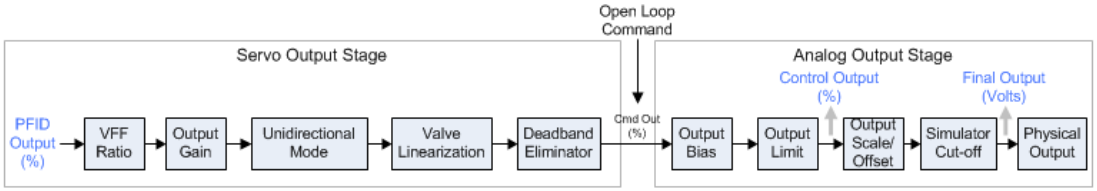
Custom Output Types

The custom output types give the user the flexibility to define asymmetric output ranges, and individual positive and negative output limits.

- **Custom V**
  If a custom voltage output range is desired, select this option.
- **Custom mA**
  If a custom current output range is desired, select this option. Available only on the CA4 and U14 modules.
- **Custom %**
  Available on Outer Loop axes only. If custom percentage units are desired, select this option.
- **Custom Units**
  Available on Outer Loop axes only. If custom output units are desired, select this option. For example, you may wish for the outer loop output to be in velocity units, such as in/sec. Choosing the Custom Units option will enable the Custom Units parameter. Enter the desired units in the **Custom Units** parameter. The units will appear in the Final Output status register for the axis.

Output Stage

The Output Type defines the **Output Scale/Offset** block in the output diagram below. The result of the **Output Scale/Offset** will be reflected in the Final Output, which gives the value of the physical output, unless the axis is in simulate mode, in which case the Final Output will be zero.



Using Custom Output Types

For any custom Output Type, the following parameters must be used to define the output range:

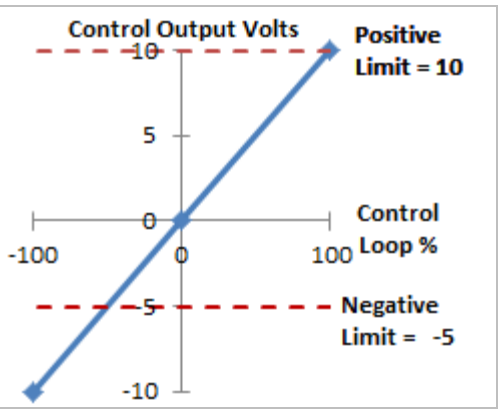
- Control Range
  Defines the directional range of the control loop output, either **±100%** for typical systems that move in both the positive and negative direction, or **0-100%** for systems that move in only one direction.
- Output at 100%
  Defines the Final Output value at 100% of Control Output.
- Output at 0%
  Defines the Final Output value at 0% of Control Output.
- Output at -100%
  Defines the Final Output value at -100% of Control Output.
- Positive Output Limit
  Defines the maximum Control Output value in the positive direction.
- Negative Output Limit
  Defines the minimum Control Output value in the negative direction.

The Invert Output Polarity parameter does not apply to custom output types. To invert the polarity on a custom output type, swap the Output at 100% and Output at -100% values.

Some examples of custom ranges:

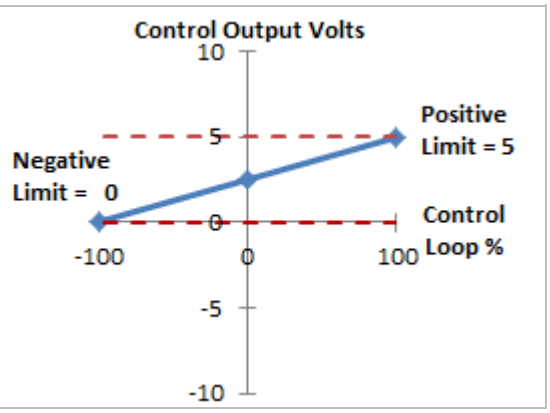
1. Normal control, but limit only the negative voltage to -5 V:

Parameter	Setting
Output Type	Custom V
Control Range	±100%
Output at 100%	10 V
Output at 0%	0 V
Output at -100%	-10 V
Positive Output Limit	10 V
Negative Output Limit	-5 V



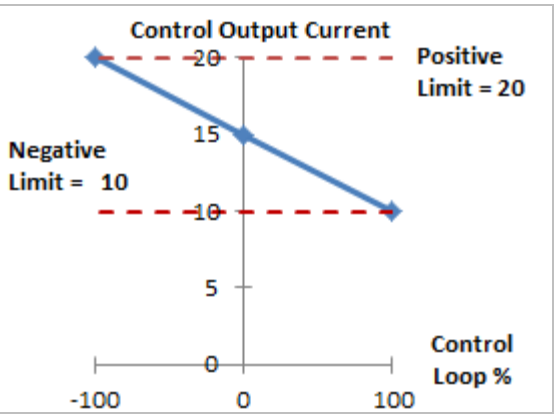
2. Use 0-5 V with 2.5 V as zero control signal:

Parameter	Setting
Output Type	Custom V
Control Range	±100%
Output at 100%	10 V
Output at 0%	2.5 V
Output at -100%	0 V
Positive Output Limit	5 V
Negative Output Limit	0 V



3. Use 10-20 mA, with 15 mA as zero control signal, and inverted:

Parameter	Setting
Output Type	Custom mA
Control Range	±100%
Output at 100%	10 mA
Output at 0%	15 mA
Output at -100%	20 mA
Positive Output Limit	20 mA
Negative Output Limit	10 mA



Enumeration Values

This information is only necessary when reading or writing the register from somewhere other than RMCTools. The options of this axis register take on the following values:

Option	Value of Bits
±10V	0

0-10V uni	1
±20mA	2
4-20mA bi	3
4-20mA uni	4
±100%	5
custom V	10
custom mA	11
custom %	12
custom units	13

See Also

Invert Output Polarity | Output Limit | Control Range | Output at 100% | Output at 0% | Output at -100% | Positive Output Limit | Negative Output Limit

9.2.3.6.9. Control Range

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.140.4-7, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].OutputBits.ControlRange, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Range:	See Below
Default Value:	0

Description

The **Control Range** axis parameter is available when the Output Type axis parameter is set to **Custom V**, **Custom mA**, or **Custom %**. The mapping of the Control Output percentage to the Final Output is then defined by the Control Range, Output at 100%, Output at 0%, and Output at -100% parameters, and limits are applied by the Positive Output Limit and Negative Output Limit.

The **Control Range** defines the directional range of the control loop output that will be used. The options are:

- **±100%**
  This will use the full positive and negative range of the PID output, and assumes the axis will move in both the positive and negative direction.
- **0-100%**
  This will use only half of the Control Output, and assumes the axis will only move in one direction. This is intended to be used together with Unidirectional Mode, where the axis will move in one direction but will not try to reverse direction to maintain position.

Format Details

This section is primarily for addressing the Control Range parameter when communicating with the RMC from an external device. This information is not necessary when configuring this

parameter in RMCTools. This information is only necessary when reading or writing the register from somewhere other than RMCTools.

Option	Value
±100%	0
0-100%	1

See Also

Output Type | Unidirectional Mode

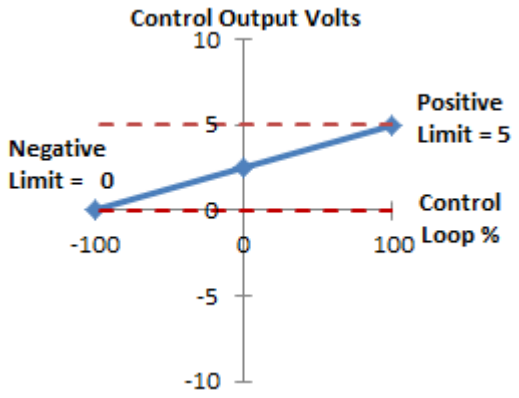
9.2.3.6.10. Output at 100%

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.141, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].OutputAtMax, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	V, mA, or %

Description

The **Output at 100%** axis parameter is available when the Output Type axis parameter is set to **Custom V**, **Custom mA**, or **Custom %**. The mapping of the Control Output percentage to the Final Output is then defined by the Output at 100%, Output at 0%, and Output at -100% parameters, and limits are applied by the Positive Output Limit and Negative Output Limit.

The **Output at 100%** axis parameter defines the output at 100% of the Control Output. For example, in the following diagram of the mapping of the Control Output to Final Output, the **Output at 100%** is set to 5:



For more details on configuring the output range, see the Output Type topic.

See Also

Output Type | Output at 0% | Output at -100% | Positive Output Limit | Negative Output Limit

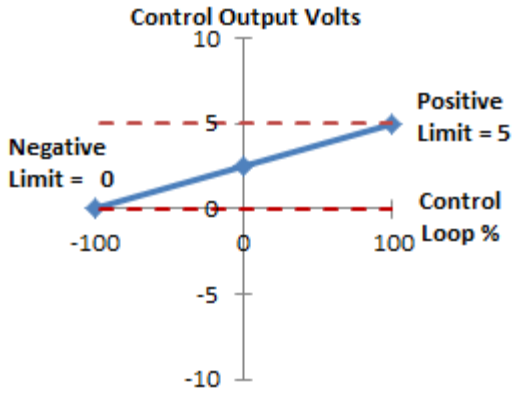
9.2.3.6.11. Output at 0%

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.142, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].OutputAt0, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	V, mA, or %

Description

The **Output at 0%** axis parameter is available when the Output Type axis parameter is set to **Custom V**, **Custom mA**, or **Custom %**. The mapping of the Control Output percentage to the Final Output is then defined by the Output at 100%, Output at 0%, and Output at -100% parameters, and limits are applied by the Positive Output Limit and Negative Output Limit.

The **Output at 0%** axis parameter defines the output at 0% of the Control Output. For example, in the following diagram of the mapping of the Control Output to Final Output, the **Output at 0%** is set to 2.5:



For more details on configuring the output range, see the Output Type topic.

See Also

Output Type | Control Range | Output at 100% | Output at -100% | Positive Output Limit | Negative Output Limit

9.2.3.6.12. Output at -100%

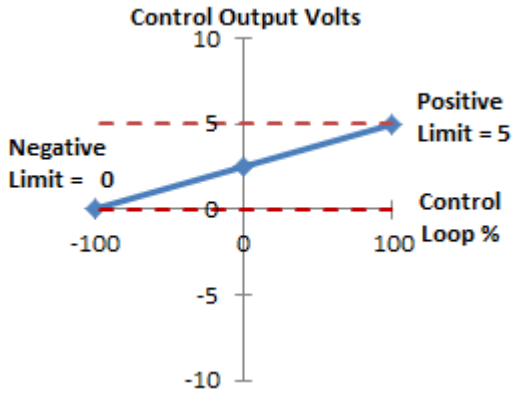
Type:	Axis Parameter Register

Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** n/a
System Tag:	_Axis[*n*].OutputAtMin, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	V, mA, or %

Description

The **Output at -100%** axis parameter is available when the Output Type axis parameter is set to **Custom V**, **Custom mA**, or **Custom %**. The mapping of the Control Output percentage to the Final Output is then defined by the Output at 100%, Output at 0%, and Output at -100% parameters, and limits are applied by the Positive Output Limit and Negative Output Limit.

The **Output at -100%** axis parameter defines the output at -100% of the Control Output. For example, in the following diagram of the mapping of the Control Output to Final Output, the **Output at -100%** is set to 0:



For more details on configuring the output range, see the Output Type topic.

See Also

Output Type | Control Range | Output at 100% | Output at 0% | Positive Output Limit | Negative Output Limit

9.2.3.6.13. Positive Output Limit

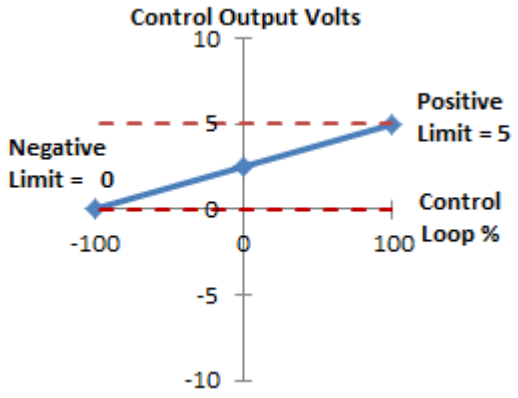
Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.144, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].OutputLimitPos, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL

Units:	V, mA, or %

Description

The **Positive Output Limit** axis parameter is available when the Output Type axis parameter is set to **Custom V**, **Custom mA**, or **Custom %**. The mapping of the Control Output percentage to the Final Output is then defined by the Output at 100%, Output at 0%, and Output at -100% parameters, and limits are applied by the Positive Output Limit and Negative Output Limit.

The **Positive Output Limit** axis parameter limits the Final Output in the positive direction. For example, in the following diagram of the mapping of the Control Output to Final Output, the **Positive Output Limit** is set to 5:



For more details on configuring the output range, see the Output Type topic.

See Also

Output Type | Negative Output Limit | Output at 100% | Output at 0% | Output at -100%

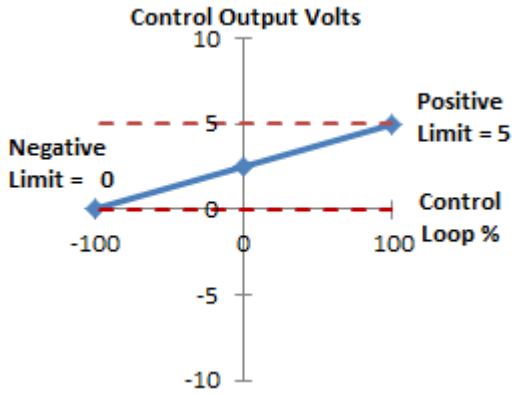
9.2.3.6.14. Negative Output Limit

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.145, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].OutputLimitNeg, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	V, mA, or %

Description

The **Negative Output Limit** axis parameter is available when the Output Type axis parameter is set to **Custom V**, **Custom mA**, or **Custom %**. The mapping of the Control Output percentage to the Final Output is then defined by the Output at 100%, Output at 0%, and Output at -100% parameters, and limits are applied by the Positive Output Limit and Negative Output Limit.

The **Negative Output Limit** axis parameter limits the Final Output in the negative direction. For example, in the following diagram of the mapping of the Control Output to Final Output, the **Negative Output Limit** is set to 0:



For more details on configuring the output range, see the Output Type topic.

See Also

Output Type | Positive Output Limit | Output at 100% | Output at 0% | Output at -100%

9.2.3.6.15. Custom Output Units

Type:	Axis Parameter Register
Address:	none
System Tag:	none
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	STRING

Description

The Custom Output Units parameter is available on Outer Loop axes when the Output Type is set to Custom Units. Enter the desired units in the **Custom Output Units** parameter. The units will appear in the Final Output status register for the axis.

An example is a cascading loop axis where the output of the outer loop is a velocity that will be the command for the inner loop. By setting the Output Type to Custom Units, the outer loop axis' output can be scaled from ±100% to the desired velocity range, and the Custom Output Units can be given the correct units, such as mm/s. These units will appear in the Final Output status register of the outer loop axis.

See Also

Output Type | Cascading Outer Loop Axes | Cascade Control

9.2.3.6.16. Output Gain

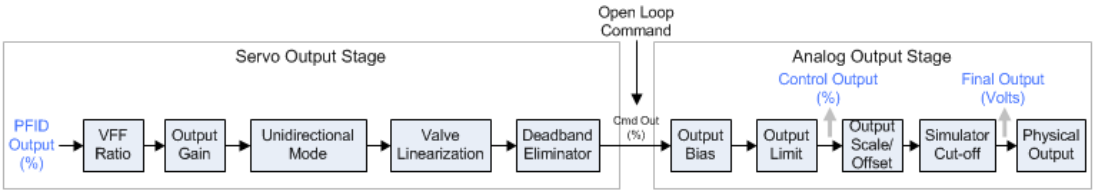
Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.153, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].OutputGain, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	none
Range:	>0
Default Value:	1

Description

The Output Gain offers a method of amplifying or attenuating the output. Changing the Output Gain parameter will in effect scale all the gains equally. The Output Gain can be used to adjust the entire control response based on some change in the system, such as a change in the hydraulic supply pressure. In most cases, this value should be left at the default value of 1.

The Output Gain is applied after the PFID Output and Feed Forward ratioing, and before all the non-linear effects of the output stage, such as unidirectional mode, valve linearization, deadband compensation, etc.

RMC200 Diagram:



See Also

PFID Output

9.2.3.6.17. Unidirectional Mode

Type:	Axis Parameter Register
RMC75 Address:	**RMC75:** %MD*n*.60/12-14, where *n* = 12+ the axis number **RMC150:** %MD*n*.60/12-14, where *n* = 24+ the axis number **RMC200:** %MD*n*.154, where *n* = 384+ the axis number
System Tag:	**RMC75/150:** _Axis[*n*].PriControlBits.UniMode, where *n* is the axis number **RMC200:** _Axis[*n*].UniMode, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	See **Format Details** below
Range:	None (0), Positive (1), Negative (2), Automatic (3)

Default Value: None (0)

Description

Unidirectional Mode, also known as Absolute Mode, is specifically designed for systems that require a unipolar control signal. Some common cases are:

- **Two Hydraulic Valves**
  Some hydraulic systems have two valves, one for direction control, the other for flow control. Unidirectional mode is for controlling the flow valve. The directional valve must be controlled by other means, but Unidirectional Mode does provide for easy switching of the control direction based on the directional valve setting.
- **Unidirectional Belt**
  A belt that must always move in the same direction.

Unidirectional Mode will prevent the Control Output from going negative even if the Actual overshoots the Target. When this occurs, the Control Output will be truncated at the Output Bias, and the Integral Term will not wind up. How the direction of control is determined is described below.

Unidirectional Mode applies only to closed-loop position, velocity, pressure, and force control. It does not affect open loop commands, but it does affect the Quick Move commands. The open loop portion of the Quick Move is adjusted in the same way as a closed loop move.

Unidirectional mode should not be used on torque mode systems.

RMC200:

On the RMC200, Undirectional Mode can be used together with any of the Output Type options, although the unidirectional options are specifically intended for use with unidirectional mode. The unidirectional Output Type options will limit the output to one side of the range. Or, a custom output range can be selected.

Direction Determination

Unidirectional Mode requires that the RMC knows the desired direction of motion on the axis. This is specified with the Unidirectional Mode axis parameter and/or the Set Control Direction (96) command.

Using the Unidirectional Mode Parameter

The Unidirectional Mode axis parameter specifies how the direction of control is determined:

- **None (0)**
  Unidirectional Mode is disabled.
- **Positive (1)**
  The control direction will be *positive* when the controller is started up or when Unidirectional Mode is first enabled. It can however be overridden using the Set Control Direction (96) command, as described below.
- **Negative (2)**
  The control direction will be *negative* when the controller is started up or when Unidirectional Mode is first enabled. It can however be overridden using the Set Control Direction (96) command, as described below.
- **Automatic (3)**
  The control direction is automatically set based on the current Target Velocity. Specifically, when the axis is in closed-loop position or velocity control, the control direction is set to match the direction of the current Target Velocity. When the axis is stopped or in the open loop control, the control direction holds its last value, but may be overridden using the Set Control Direction (96) command.

  If, after automatic Unidirectional Mode is first enabled, the axis enters closed loop using a Hold Current Position (5) command or similar means whereby the Target Velocity is zero, then the control direction will be assumed to be positive until a directional move is started.

  This option assumes that the user has switched the directional valve prior to changing the direction of the axis's target profile. Failure to do so may result in runaway conditions.

The axis must be in open loop control when the Unidirectional Mode parameter is changed.

Using the Set Control Direction Command

In Positive or Negative mode, issue the Set Control Direction (96) command at any time to set the desired direction.

In Automatic mode, the direction is determined based on the Target Velocity. The Set Control Direction (96) command may still be used while the axis is stopped to indicate a change in the system direction prior to a move command. Notice that issuing this command will have no effect in Automatic mode while the target profile is moving.

For many systems, this command is unnecessary, because the Automatic option handles direction changes very well for most two-valve systems, and for a unidirectional belt, the direction is set by the Unidirectional Mode parameter on startup and never needs to be changed after that.

The Set Control Direction command can be issued while the axis in closed loop control.

Pressure and Force Control

Pressure/Force Control

Unidirectional Mode can be used in Pressure/Force Control mode, although automatic direction detection is not supported. Therefore, Automatic mode will behave the same as Positive mode.

Pressure/Force Limit

Unidirectional Mode works normally when Pressure/Force Limit is active. That is, for open loop with Pressure/Force Limit, Unidirectional Mode does not affect the output, and for position or velocity control with Pressure/Force Limit, the Control Output will have the Unidirectional Mode applied to it based on the perceived position/velocity direction.

Interaction with other Output Parameters

The Unidirectional Mode is applied before the Output Bias and the Invert Output Polarity. Therefore, the Output Bias can be adjusted to any value as usual. The Invert Output Polarity can be used to change the Control Output to the negative range.

This chart summarizes how the Invert Output Polarity works with Unidirectional Mode:

		Control Output	
Control Direction	**PFID Output**	**Normal Output Polarity**	**Inverted Output Polarity**
Positive	<0	0	0
Positive	≥0	0..10V	0..-10V
Negative	≤0	0..10V	0..-10V
Negative	>0	0	0

Format Details

This section is primarily for addressing the Unidirectional Mode parameter when communicating with the RMC from an external device. This information is not necessary when configuring the Unidirectional Mode in RMCTools.

RMC75/150:

The Unidirectional Mode is selected with bits 12-14 in the Primary Control Register. These bits correspond to the Unidirectional Mode as shown in the following table:

Bit 14	Bit 13	Bit 12	Value	Unidirectional Mode
0	0	0	0	None
0	0	1	1	Positive
0	1	0	2	Negative
0	1	1	3	Automatic

RMC200:

The Unidirectional Mode is a DINT register. These values correspond to the Unidirectional Mode as shown in the following table:

Value	Unidirectional Mode
0	None
1	Positive
2	Negative
3	Automatic

See Also

Set Control Direction (96) | Primary Control Register

9.2.3.6.18. Output Bits Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.34, where *n* = 12 + the axis number **RMC150:** %MD*n*.34, where *n* = 24 + the axis number **RMC200:** %MD*n*.140, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].OutputBits, where *n* is the axis number
How to Find:	See individual parameters listed below
Data Type:	DWORD - see below

Description

The Output Bits Register contains the bit-addressable output configuration parameters. This topic lists the bit address for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Tag Names and Bits

RMC75/150

This register contains the following parameters. The bits for each are given in the right-hand column.

Parameter	Tag Name	Bit Number(s)
Invert Output Polarity	InvertOutPol	0
Enable Output Behavior	EnableOutBehavior	2
Fault Input Polarity	FaultInPol	3
Valve Linearization Type	ValveLin	4-5

RMC200

This register contains the following parameters. The bits for each are given in the right-hand column.

Parameter	Tag Name	Bit Number(s)
Output Type	OutputType	0-3
Control Range	ControlRange	4-7
Invert Output Polarity	InvertOutPol	8

See Also
Parameter Registers

9.2.3.6.19. Fault Input Source

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.11.0-8, where *n* = 384+ the axis number
System Tag:	**RMC200:** _Axis[*n*].FaultInCfg.FaultInSrc, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	bits-see below
Default Value:	0 (none)

Description

The Fault Input Source parameter specifies which discrete input is to be used for the axis' Fault Input . Any discrete input on the RMC200 may be used. Typically, the dedicated input on the module containing the Control Output is selected.

This parameter is only available on the RMC200. The RMC75 and RMC150 support only the dedicated Fault Input.

Selecting a Discrete Input as the Fault Input

1. In the Axis Parameters, on the **All** tab, expand the **Output** section.
2. For the desired axis, click the **Fault Input** parameter.
3. From the drop-down list, choose a discrete input.
    a. If you wish to use the dedicated Fault Input on the CA4, choose **Dedicated**.
    b. The other discrete inputs on the RMC are listed by the IEC address of **%IX*s*.*n***, where ***s*** is the slot number and ***n*** is the number of the IO point on the module.
    The slot numbering ***s*** begins with 0 for the power supply, 1 for the CPU, 2 for the first IO module slot, etc.
4. Make sure to also set the Fault Input Polarity.

Format Details

This section is primarily for addressing the Fault Input Source when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

The value of the Fault Input Source register is:

0 = none

1 = dedicated

n = ***s*** x 32 + ***n***

where ***s*** is the slot number of the module containing the discrete input and ***n*** is the number of the IO point on the module.

See Also

Fault Input | Fault Input Polarity

9.2.3.6.20. Fault Input Polarity

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.34.3, where *n* = 12 + the axis number **RMC150:** %MD*n*.34.3, where *n* = 24 + the axis number **RMC200:** %MD*n*.11.16, where *n* = 384 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].OutputBits.FaultInPol, where *n* is the axis number **RMC200:** _Axis[*n*].FaultInCfg.FaultInPol
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	bit
Range:	Active High (0), Active Low (1)
Default Value:	Active High (0)

Description

This parameter specifies the polarity of the Fault Input on modules with a Fault Input. It can be set to the following:

- **Active High (0)**
  When the Fault Input is on, the Fault input is active.
- **Active Low (1)**
  When the Fault Input is off, the Fault input is active.

For details on the Fault Input, see the Fault Input topic.

Note:
The Fault input turns on when a current flows. The polarity of the voltage is unimportant.

For the RMC75/150, see the Output Register for details about the register containing these bits. For the RMC200, see the Fault Input Configuration register for details about the register containing these bits.

See Also

Parameter Registers | Fault Input | Fault Input Source

9.2.3.6.21. Fault Input Configuration Register

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.11, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].FaultInCfg, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	DWORD - see below

Description

The Fault Input Configuration register for the RMC200 contains the bit-addressable Fault Input configuration parameters. This topic lists the bit addresses for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Tag Names and Bits

This register contains the following parameters. The bits for each are given in the right-hand column.

Parameter	Tag Name	Bit Number(s)
Fault Input Polarity	EnableOutBehavior	16
Fault Input Source	EnableOutSrc	0-8

See Also

Fault Input Source | Fault Input Polarity

9.2.3.6.22. Enable Output Source

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.10.0-8, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].EnOutCfg.EnableOutSrc, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	bits-see below
Default Value:	0 (none)

Description

The Enable Output Source parameter specifies which discrete output is to be used for the axis' Enable Output. Any discrete output on the RMC may be used. Typically, the dedicated output on the module containing the Control Output is selected.

This parameter is only available on the RMC200.

Selecting a Discrete Output as the Enable Output

1. In the Axis Parameters, on the **All** tab, expand the **Output** section.
2. For the desired axis, click the **Enable Output** parameter.
3. From the drop-down list, choose a discrete output. If you wish to use the dedicated Enable Output on the CA4, choose **Dedicated**.

   The other discrete outputs are listed by the IEC address of **%QX*s*.*n***, where ***s*** is the slot number and ***n*** is the number of the IO point on the module.
   The slot numbering ***s*** begins with 0 for the power supply, 1 for the CPU, 2 for the first IO module slot, etc.

Format Details

This section is primarily for addressing the Enable Output Source when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

The value of the Enable Output Source register is:

0 = none
1 = dedicated
n = ***s*** x 32 + ***n***

where ***s*** is the slot number of the module containing the discrete output and ***n*** is the number of the IO point on the module.

See Also

Enable Output | Enable Output Behavior

9.2.3.6.23. Enable Output Behavior

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.34.2, where *n* = 12 + the axis number **RMC150:** %MD*n*.34.2, where *n* = 24 + the axis number **RMC200:** %MD*n*.10.16, where *n* = 384 + the axis number
System Tag:	RMC75/150: _Axis[*n*].OutputBits.EnableOutBehavior RMC200: _Axis[*n*].EnOutCfg.EnableOutBehavior
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	bit
Range:	Active Closed (0), Active Open (1)
Default Value:	Active Closed (0)

Description

This parameter determines the behavior of the Enable Output on an axis module with an Enable Output. It can be set to the following:

- **Active Closed (0)**
  When the Enable Output is set, the Enable output switch is closed (on, conducting).
- **Active Open (1)**
  When the Enable Output is set, the Enable output switch is opened (off, non-conducting).

The Enable Output can be set with the Set Enable Output (67) command.

See the Output Register (RMC75/150) or Enable Output Configuration (RMC200) for details about the register containing these bits.

See Also

Parameter Registers | Enable Output

9.2.3.6.24. Enable Output Configuration

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.10, where *n* = 384+ the axis number

System Tag:	_Axis[*n*].EnOutCfg
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	DWORD - see below

Description

The Enable Output Configuration register contains the bit-addressable Enable Output configuration parameters. This topic lists the bit addresses for each parameter. Each parameter is accessible in RMCTools via the Axis Parameter Editor. For details on each parameter, see the respective links.

Tag Names and Bits

This register contains the following parameters. The bits for each are given in the right-hand column.

Parameter	Tag Name	Bit Number(s)
Enable Output Behavior	EnableOutBehavior	16
Enable Output Source	EnableOutSrc	0-8

See Also

Enable Output Behavior | Enable Output Source

9.2.3.6.25. Valve Linearization Type

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.34/4-5, where *n* = 12 + the axis number **RMC150:** %MD*n*.34/4-5, where *n* = 24 + the axis number **RMC200:** %MD*n*.155, where *n* = 384 + the axis number
System Tag:	**RMC75/150:** _Axis[*n*].OutputBits.ValveLin **RMC200:** _Axis[*n*].ValveLin
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	RMC75/150: bits RMC200: DINT
Default Value:	None (0)

Description

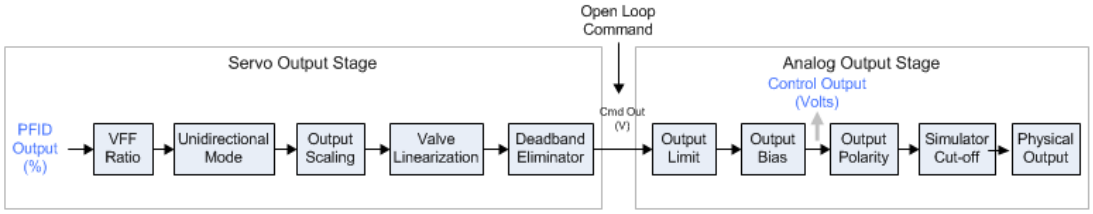
This parameter specifies the type of Valve Linearization for the axis. It can be set to the following:

- **None (0)**
  No valve linearization is applied to the axis.
- **Single-Point (1)**
  Single-point linearization is applied to the output of the axis. The Knee Command Voltage and Knee Flow Percentage axis parameters define the single-point valve linearization.
- **Curve (2)**
  Linearization is applied to the output of the axis by using the curve specified by the Valve Linearization Curve ID axis parameter.

For more details, see Valve Linearization.

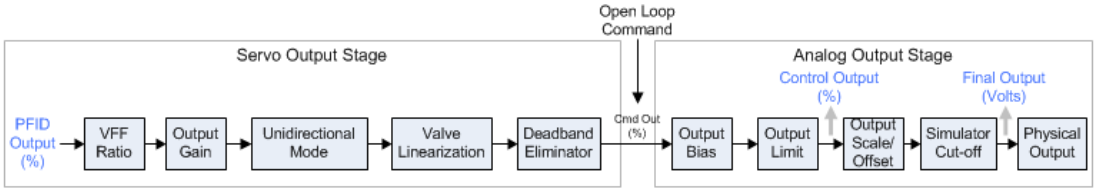
RMC75 and RMC150

The Valve Linearization is applied after the PFID Output, Feed Forward ratioing, and Output Scaling.



RMC200

The Valve Linearization is applied after the PFID Output, Output Gain, and Feed Forward ratioing.



Format Details

This section is primarily for addressing the Valve Linearization Type parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

RMC75/150

The value is defined by bits 4 and 5 of the Output Register.

Bit 5	Bit 4	Value	Valve Linearization Type
0	0	0	None
0	1	1	Single-Point
1	0	2	Curve

RMC200

The value is a DINT register.

Value	Valve Linearization Type
0	None
1	Single-Point
2	Curve

See the Output Register for details about the register containing these bits.

See Also

Valve Linearization | Output Register | Knee Command Voltage | Knee Flow Percentage | Valve Linearization Curve ID

9.2.3.6.26. Valve Linearization Curve ID

Type:	Axis Parameter Register
Address:	**RMC70:** %MD*n*.46, where *n* = 12+ the axis number **RMC150:** %MD*n*.46, where *n* = 24+ the axis number **RMC200:** %MD*n*.158, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].ValveLinCurveID, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	DINT
Range:	Valid curve number
Default Value:	0

Description

This parameter specifies the curve to use for valve linearization. This parameter is available when the Valve Linearization Type axis parameter is set to **Curve (2)**.

The Curve ID must refer to a valid valve linearization curve. This means:

- In the Curve Tool, the **Curve Type** property of the curve must be set to **Valve Linearization**.
- The curve must meet the other requirements of a Valve Linearization Curve.

See Also

Valve Linearization Type | Valve Linearization

9.2.3.6.27. Knee Command Input Knee Command Voltage

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.44, where *n* = 12+ the axis number **RMC150:** %MD*n*.44, where *n* = 24+ the axis number **RMC200:** %MD*n*.156, where *n* = 384+ the axis number
System Tag:	**RMC75/150:** _Axis[*n*].KneeVoltage **RMC200:** _Axis[*n*].KneeCmdIn
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	**RMC75/150:** V **RMC200:** %
Range:	**RMC75/150:** 0 < V < 10 **RMC200:** 0 < % < 100
Default Value:	**RMC75/150:** 1 **RMC200:** 10

Description

The Knee Command Voltage, together with the Knee Flow Percentage, defines the Single-Point Valve Linearization. These parameters are valid only if the Valve Linearization Type is set to **Single-Point**.

See the Valve Linearization topic for details on how to use these parameters.

See Also

Valve Linearization | Knee Flow Percentage | Valve Linearization Type

9.2.3.6.28. Knee Flow Output Knee Flow Percentage

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.45, where *n* = 12+ the axis number **RMC150:** %MD*n*.45, where *n* = 24+ the axis number **RMC200:** %MD*n*.157, where *n* = 384+ the axis number
System Tag:	**RMC75/150:** _Axis[*n*].KneeFlow **RMC200:** _Axis[*n*].KneeFlowOut
How to Find:	Axes Parameters Pane, All tab: Output
Data Type:	REAL
Units:	%
Range:	0 < % < 100
Default Value:	10

Description

The Knee Flow Output, together with the Knee Command Input, define the Single-Point Valve Linearization. These parameters are valid only if the Valve Linearization Type is set to **Single-Point**.

See the Valve Linearization topic for details on how to use these parameters.

See Also

Valve Linearization | Knee Command Voltage | Valve Linearization Type

9.2.3.7. Target

9.2.3.7.1. Positive Travel Limit

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.92, where *n* = 12 + the axis number **RMC150:** %MD*n*.92, where *n* = 24 + the axis number **RMC200:** %MD*n*.400, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].PosTravelLimit, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	pu
Range:	any
Default Value:	0

Description

The Positive Travel Limit and the Negative Travel Limit specify the position boundaries in which the axis is allowed to operate. The Positive Travel Limit must be greater than or equal to the Negative Travel Limit. These limits are not available on Rotary axes.

In Closed Loop Control

If the Target Position exceeds the Positive Travel Limit, the Positive Overtravel error bit will be set. If the Target Position is less than the Negative Travel Limit, the Negative Overtravel error bit will be set. While the axis is in an overtravel condition, motion commands that move the axis *toward* the valid range will be accepted, even if they don't move the axis entirely into the valid range.

In Open Loop Control

If the Actual Position exceeds the Positive Travel Limit, the Positive Overtravel error bit will be set. If the Actual Position is less than the Negative Travel Limit, the Negative Overtravel error bit will be set. While the axis is in an overtravel condition, motion commands that move the axis *toward* the valid range will be accepted, even if they don't move the axis entirely into the valid range.

The error bits will cause a Halt to occur if the corresponding Auto Stops are configured to do so and the Direct Output Status bit is off.

Note:
If the Direct Output Status bit is on, the Overtravel bits will not be set.

Moving Toward the Valid Travel Range

While the axis is outside the valid travel range, as defined by the Travel Limits, any motion commands the move the axis toward the valid travel range will be accepted, even if they don't move the axis entirely into the valid range.

Notice that if the axis is outside the valid travel range, in open loop, and is drifting slightly away from the valid range, issuing a closed loop motion command may again trigger an overtravel error. This is because the target position starts at the actual position, velocity and acceleration at the time the command is issued. This means the target will continue moving away from the valid range while accelerating to turn around. This will trigger an overtravel error. To avoid this, first issue a Hold Current Position (5) command, then the motion command. Or, simply use an open loop move to move into the valid range.

Truncating a Motion Command

Any motion command issued with a Requested Position that is outside of the travel limits will be set to the closest travel limit, and the Command Modified error bit will be set. The error bit will cause a Halt to occur if the Command Modified Auto Stop is configured to do so and the Direct Output Status bit is off. If the axis is outside of the travel limits when a move command is issued, the axis will only be allowed to move in the direction that brings it closer to the limit.

See Also

Parameter Registers | Positive Overtravel Error bit | Negative Overtravel Error bit | Limit Inputs

9.2.3.7.2. Negative Travel Limit

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.93, where *n* = 12 + the axis number
	RMC150: %MD*n*.93, where *n* = 24 + the axis number
	RMC200: %MD*n*.401, where *n* = 384 + the axis number

System Tag:	_Axis[*n*].NegTravelLimit, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
Data Type:	REAL
Units:	pu
Range:	any
Default Value:	0

Description

The Negative Travel Limit and the Positive Travel Limit specify the position boundaries in which the axis is allowed to operate. The Positive Travel Limit must be greater than or equal to the Negative Travel Limit. These limits are not available on Rotary axes.

In Closed Loop Control

If the Target Position exceeds the Positive Travel Limit, the Positive Overtravel error bit will be set. If the Target Position is less than the Negative Travel Limit, the Negative Overtravel error bit will be set. While the axis is in an overtravel condition, motion commands that move the axis *toward* the valid range will be accepted, even if they don't move the axis entirely into the valid range.

In Open Loop Control

If the Actual Position exceeds the Positive Travel Limit, the Positive Overtravel error bit will be set. If the Actual Position is less than the Negative Travel Limit, the Negative Overtravel error bit will be set. While the axis is in an overtravel condition, motion commands that move the axis *toward* the valid range will be accepted, even if they don't move the axis entirely into the valid range.

The error bits will cause a Halt to occur if the corresponding Auto Stops are configured to do so and the Direct Output Status bit is off.

Note:
If the Direct Output Status bit is on, the Overtravel bits will not be set.

Any motion command issued with a Requested Position that is outside of the travel limits will be set to the closest travel limit, and the Command Modified error bit will be set. The error bit will cause a Halt to occur if the Command Modified Auto Stop is configured to do so and the Direct Output Status bit is off. If the axis is outside of the travel limits when a move command is issued, the axis will only be allowed to move in the direction that brings it closer to the limit.

See Also

Parameter Registers | Positive Overtravel Error bit | Negative Overtravel Error bit | Limit Inputs

9.2.3.7.3. Positive Pressure/Force Limit

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.100, where *n* = 12 + the axis number **RMC150:** %MD*n*.100, where *n* = 24 + the axis number **RMC200:** %MD*n*.440, where *n* = 384 + the axis number
System Tag:	**Pressure Axis:** _Axis[*n*].PosPrsLimit, where *n* is the axis number **Force Axis:** _Axis[*n*].PosFrcLimit, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup

	Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL
Units:	Pr or Fr
Range:	any
Default Value:	0

Description

The Positive Pressure/Force Limit and the Negative Pressure/Force Limit specify the pressure or force boundaries in which the axis is allowed to operate. The Positive Pressure/Force Limit must be greater than or equal to the Negative Pressure/Force Limit.

If a command is issued that sets the Command Pressure/Force outside of the Positive and Negative Pressure/Force Limits, the pressure or force will be truncated to the closest Pressure/Force Limit and the Command Modified error bit will be set. The error bit will cause a Halt to occur if the Command Modified Auto Stop is configured to do so and the Direct Output Status bit is off.

If the current Pressure or Force Target is outside of the pressure/force limits, commands that move it closer to the valid range will be allowed.

Note:
No error bits or auto stop will be set if the Target Pressure/Force or Actual Pressure/Force exceed the Positive or Negative Pressure/Force Limit. These limits only apply to the Command Pressure/Force at the time a command is issued to the axis.

See Also

Parameter Registers | Positive Overtravel Error bit | Negative Overtravel Error bit | Limit Inputs

9.2.3.7.4. Negative Pressure/Force Limit

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.101, where *n* = 12 + the axis number
	RMC150: %MD*n*.101, where *n* = 24 + the axis number
	RMC200: %MD*n*.441, where *n* = 384 + the axis number
System Tag:	**Pressure Axis:** _Axis[*n*].NegPrsLimit, where *n* is the axis number
	Force Axis: _Axis[*n*].NegFrcLimit, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Primary Control Setup
	Axes Parameters Pane, Setup tab: Secondary Control Setup
Data Type:	REAL
Units:	Pr or Fr
Range:	any
Default Value:	0

Description

The Negative Pressure/Force Limit and the Positive Pressure/Force Limit specify the pressure or force boundaries in which the axis is allowed operate. The Negative Pressure/Force Limit must be less than or equal to the Positive Pressure/Force Limit.

If a command is issued that sets the Command Pressure/Force outside of the Positive and Negative Pressure/Force Limits, the pressure or force will be truncated to the closest Pressure/Force Limit and the Command Modified error bit will be set. The error bit will cause a Halt to occur if the Command Modified Auto Stop is configured to do so and the Direct Output Status bit is off.

If the current Pressure or Force Target is outside of the pressure/force limits, commands that move it closer to the valid range will be allowed.

Note:
No error bits or auto stop will be set if the Target Pressure/Force or Actual Pressure/Force exceed the Positive or Negative Pressure/Force Limit. These limits only apply to the Command Pressure/Force at the time a command is issued to the axis.

See Also

Parameter Registers | Positive Overtravel Error bit | Negative Overtravel Error bit | Limit Inputs

9.2.3.7.5. Requested Jerk

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.94, where *n* = 12 + the axis number **RMC150:** %MD*n*.94, where *n* = 24 + the axis number **RMC200:** %MD*n*.403, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].ReqJerk, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Target
Data Type:	REAL
Units:	pu/s^3
Range:	≥0
Default Value:	100000

Description

The Requested Jerk parameter is used for point-to-point and velocity closed loop motion commands. The term 'jerk' is the rate of acceleration. The Requested Jerk specifies the requested average Target Jerk. Therefore, the peak Target Jerk will be 1.5 times the Requested Jerk.

The Requested Jerk parameter is used as follows:

- **Specify the Target Type (RMC75/150 only, firmware 1.50 and newer)**

  If the Requested Jerk is set to a non-zero number, the target generator will generate speed profiles with s-curves. If the Requested Jerk is set to 0, the target generator will generate trapezoidal target speed profiles. Notice that the value of the jerk doesn't matter when specifying a target type, only non-zero versus zero.

  Notice that the RMC200 has a separate Target Type parameter for selecting trapezoidal or s-curve speed profiles.

- **Speed at Position (36) Command**

  The Requested Jerk is used in the calculation of the speed at position command when the S-curve target generator is active.

- **Short Moves**

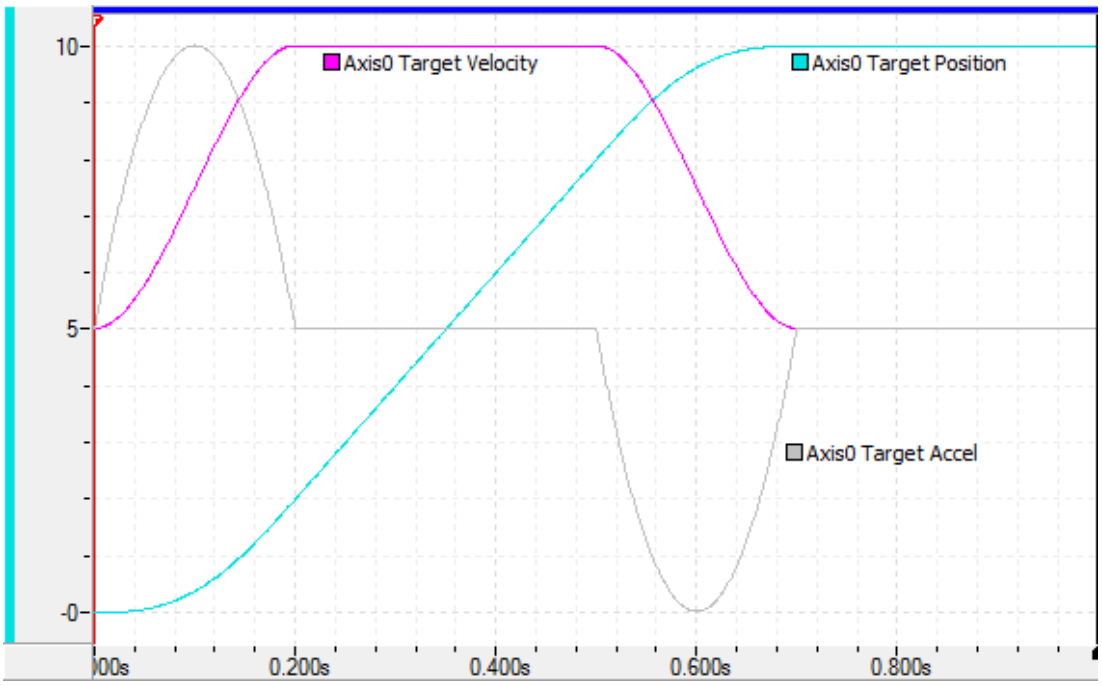
  The Requested Jerk is used in the calculations for point-to-point moves when the following criteria are met:

- The target is stopped but not at the final position.
- The requested velocity is non-zero (as it usually is unless the user wanted to just stop at the current position).
- The distance to move is less than the acceleration distance plus deceleration distance.

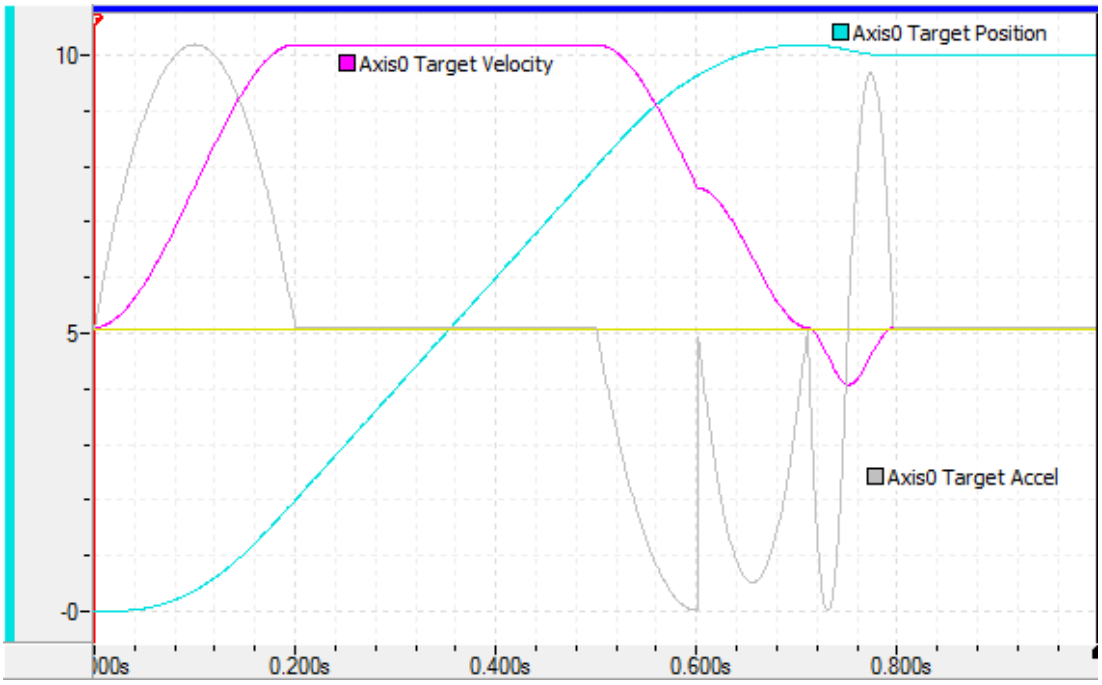
S-Curve Target Profile

The S-curve profile generates a 5th-order profile for the Target Velocity during the acceleration and deceleration phases, as shown by the magenta line below. This results in a smooth Target Acceleration (gray line), with no sudden jumps. A smooth Target Acceleration will result in a smooth control output contribution by the Acceleration Feed Forward, for optimal control of the axis motion.

With the S-curve target profile, the commanded acceleration specifies the *average* Target Acceleration. The maximum instantaneous Target Acceleration will be 1.5 times the commanded acceleration.



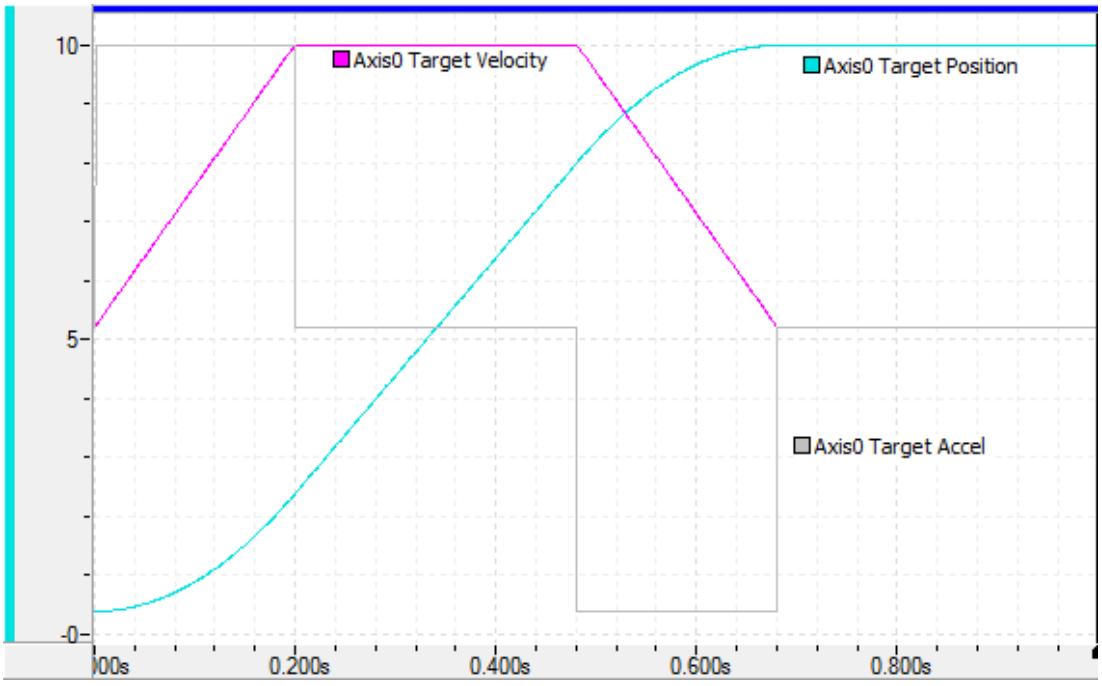
A disadvantage of the RMC's S-curve target profile is that if another motion command is sent to the axis during the deceleration portion of a point-to-point command, the Target Acceleration will be reset to zero before continuing. This will extend the deceleration time and may cause the Target Position to overshoot the requested position, as shown below. If commands are continuously sent to the axis while commands are in progress, Delta recommends using the trapezoidal target profile instead. Notice that if an identical command is sent during the deceleration portion, the RMC will recognize this and not reset the Target Acceleration, and the axis will not overshoot.



Trapezoidal Target Profile

The Trapezoidal profile generates linear segments for the Target Velocity, as shown by the magenta line below. This results in a Target Acceleration (gray line), that suddenly jumps to its requested value. This will result in a sudden jump in the control output contribution by the Acceleration Feed Forward, and may prevent optimal control of the axis motion, especially for highly dynamic systems.

A trapezoidal target profile should be used if commands are continuously sent to the axis while commands are in progress, to prevent the target position from overshooting.



See Also

Target Type | Move Absolute (20)

9.2.3.7.6. Target Type

Type:	Axis Parameter Register
Address:	**RMC75:** n/a **RMC150:** n/a **RMC200:** %MD*n*.402, where *n* = 384+ the axis number
System Tag:	_Axis[*n*].TgtType, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Target
Data Type:	DINT
Range:	5th Order (0), Trapezoidal (1)
Default Value:	5th Order (0)

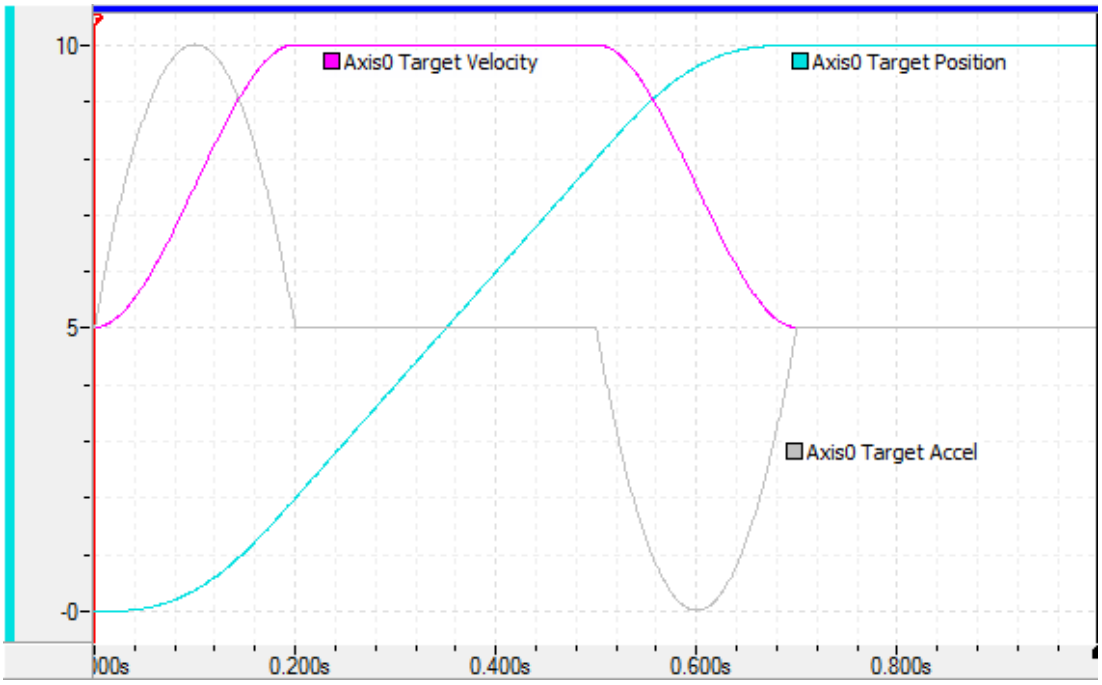
Description

The RMC200 Target Type parameter specifies the type of target velocity profile to be used for point-to-point and velocity closed loop motion commands. The RMC75/150 use the Requested Jerk parameter for the same purpose.

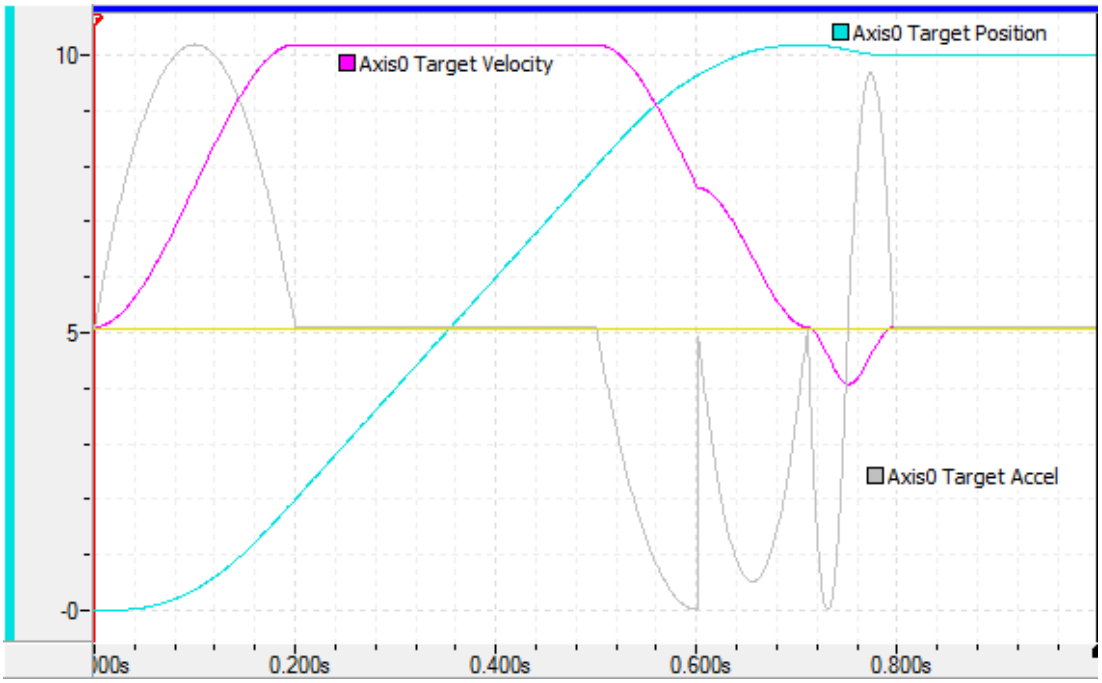
5th-Order Target Profile (S-curve)

The 5th-order profile generates an s-curve-like profile for the Target Velocity during the acceleration and deceleration phases, as shown by the magenta line below. This results in a smooth Target Acceleration (gray line), with no sudden jumps. A smooth Target Acceleration will result in a smooth control output contribution by the Acceleration Feed Forward, for optimal control of the axis motion.

With the 5th-order target profile, the commanded acceleration specifies the *average* Target Acceleration. The maximum instantaneous Target Acceleration will be 1.5 times the commanded acceleration.



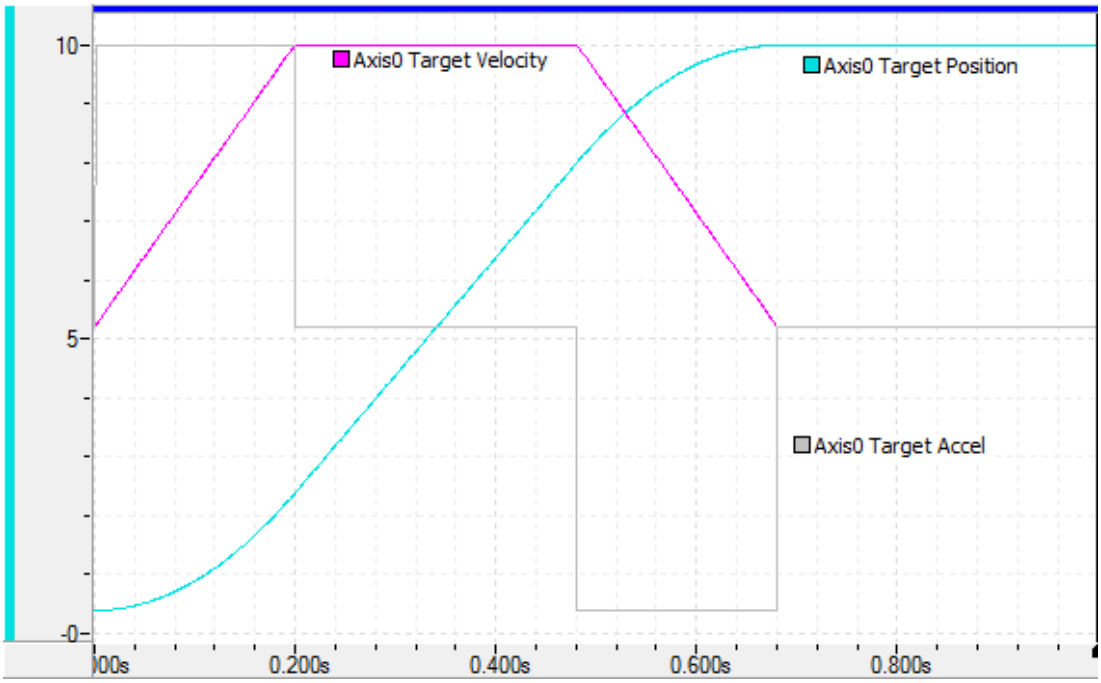
A disadvantage of the RMC's 5th-order target profile is that if another motion command is sent to the axis during the deceleration portion of a point-to-point command, the Target Acceleration will be reset to zero before continuing. This will extend the deceleration time and may cause the Target Position to overshoot the requested position, as shown below. If commands are continuously sent to the axis while commands are in progress, Delta recommends using the trapezoidal target profile instead. Notice that if an identical command is sent during the deceleration portion, the RMC will recognize this and not reset the Target Acceleration, and the axis will not overshoot.



Trapezoidal Target Profile

The Trapezoidal profile generates linear segments for the Target Velocity, as shown by the magenta line below. This results in a Target Acceleration (gray line), that suddenly jumps to its requested value. This will result in a sudden jump in the control output contribution by the Acceleration Feed Forward, and may prevent optimal control of the axis motion, especially for highly dynamic systems.

A trapezoidal target profile should be used if commands are continuously sent to the axis while commands are in progress, to prevent the target position from overshooting.



See Also

Requested Jerk | Move Absolute (20)

9.2.3.8. Halts

Show All

9.2.3.8.1. Auto Stop Configuration

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.106 to %MD*n*.108, where *n* = 12 + the axis number **RMC150:** %MD*n*.106 to %MD*n*.108, where *n* = 24 + the axis number **RMC200:** %MD*n*.0 to %MD*n*.2, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].AutoStopCfg1, _Axis[*n*].AutoStopCfg2, and _Axis[*n*].AutoStopCfg3, where *n* is the axis number

How to Find:	Axes Parameters Pane, Setup tab: Halts → Auto Stop Configuration
Data Type:	DWORD - See Format Details below

Description

Auto Stops are Halts that are defined to occur automatically on rising edges of the axis' error bits. The user has control over which error bits cause which levels of halts, or whether an error will cause a halt at all. The default setting of all the Auto Stops is Direct Output Halt. This means that when an error bit turns on, the axis will receive a Direct Output halt.

During startup and tuning, you will typically need to set some Auto Stops to Status Only to keep halts from interfering with the tuning. After completing the startup procedure, make sure to set the Auto Stops to setting that are safe for the machine operation.

Auto Stops are always active unless the axis is in the Direct Output state (whether it was caused the Direct Output (9) command, or a halt), in which case no Auto Stops will be triggered even though error bits may be set.

Note: Auto Stops will not occur if the axis is in Direct Output (as indicated by the Direct Output Status bit)!

Each Auto Stop may be configured to one of the following levels:

AutoStop Level	Action
Status Only	No action is taken. **Note:** The No Transducer Auto Stop cannot be set to Status Only. The Transducer Overflow Auto Stop cannot be set to Status Only for certain feedback types.
External Halt	An External Halt is initiated. An External Halt only sets the External Halt status bit. No other action is taken.
Closed Loop Halt	If the axis is currently in Position or Velocity Closed Loop Control, then a Closed Loop Halt is initiated, otherwise an Open Loop Halt is initiated. **Note:** The No Transducer, and Noise Error Auto Stop bits cannot be set to Closed Loop Halt. The Transducer Overflow Auto Stop cannot be set to Closed Loop Halt for certain feedback types.
Open Loop Halt	An Open Loop Halt is initiated.
Direct Output Halt	A Direct Output Halt is initiated.
Disable Axis Halt	A Disable Axis Halt is initiated.

See the Halts topic for details on the steps taken by the RMC when a Halt occurs.

Configuring the Auto Stops

To configure the Auto Stops:

- Click the **Axis Tools** button on the RMCTools toolbar.
- In the **Axis Parameters** pane, on the **Setup** tab, expand the **Halts** section and expand the **Auto Stop Configuration** section.
- Choose a halt type for each error bit.

- Click the **download** button  in the **Axis Tools** toolbar.

Why Bother?

A good system designer not only considers how a machine reacts in normal operation, but also considers how it reacts when something fails. The Auto Stops help the designer avoid personal injury or equipment damage when something does fail.

Example

A system designer decided to set all the Auto Stops to Status Only to avoid the hassle of having the system stop when he was tuning it up. Unfortunately, the feedback transducer on one of the hydraulic cylinders failed. The RMC thought the cylinder was at 0 inches when it really was at 20 inches.

The feedback error caused the RMC to apply full drive to the cylinder. The cylinder slammed to the end of the stroke, smashing sensitive machine parts. This cylinder had also been synchronized with another cylinder to push a machine part back and forth. Of course, the cylinders went out of synchronization, bending the linkages between them.

If the Auto Stop for the No Transducer Error Bit and Transducer Overflow had been set to Open Loop Halt, this accident could have been avoided. However, if it had been set to Closed Loop halt, the accident would still have occurred, because a Closed Loop Halt still requires feedback.

Format Details

This section is primarily for addressing the Auto Stops parameter when communicating with the RMC from an external device. This information is not necessary when configuring this parameter in RMCTools.

Each Auto Stop level is configured with 3 bits. These bits correspond to the Auto Stop Levels as shown in the following table:

High Bit	Mid Bit	Low Bit	Value	Level
0	0	0	0	Status Only
0	0	1	1	External Halt
0	1	0	2	Closed Loop Halt
0	1	1	3	Open Loop Halt
1	0	0	4	Direct Output Halt
1	0	1	5	Disable Axis Halt

Auto Stop Bit Mapping

RMC75/150 Axis Parameter	RMC200 Axis Parameter	Data Type	Tag Name and Bits		
106	0	DWORD	AutoStopCfg1.FollowErr	bits 0-2	Following Error
			-	bits 3-5	Reserved
			AutoStopCfg1.OutSat	bits 6-8	Output Saturated
			AutoStopCfg1.FaultIn	bits 9-11	Fault Input
			AutoStopCfg1.PosLimitIn	bits 12-14	Positive Limit Input
			AutoStopCfg1.NegLimitIn	bits 15-17	Negative Limit Input

			AutoStopCfg1.NoTrans	bits 18-20	No Transducer
			AutoStopCfg1.TransOverflow	bits 21-23	Transducer Overflow
107	1	DWORD	AutoStopCfg2.NoiseErr	bits 0-2	Noise Error
			AutoStopCfg2.PosOvertravel	bits 3-5	Positive Overtravel
			AutoStopCfg2.NegOvertravel	bits 6-8	Negative Overtravel
			AutoStopCfg2.CmdErr	bits 9-11	Command Error
			AutoStopCfg2.CmdMod	bits 12-14	Command Modified
			AutoStopCfg2.CfgErr	bits 15-17	Configuration Error
			AutoStopCfg2.RunErr	bits 18-20	Runtime Error
			AutoStopCfg2.OutFault	bits 21-23	Output Fault
108	2	DWORD	AutoStopCfg3.PFNoTrans	bits 6-8	No Transducer (Pressure/Force)
			AutoStopCfg3.PFTransOverflow	bits 9-11	Transducer Overflow ((Pressure/Force))
			AutoStopCfg3.PFNoiseErr	bits 12-14	Noise Error (Pressure/Force)
			AutoStopCfg3.PFFollowErr	bits 15-17	Following Error (Pressure/Force)

See Also

Parameter Registers | Halts Overview | Error bits

9.2.3.8.2. Halt Group Number

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.112, where *n* = 12 + the axis number **RMC150:** %MD*n*.112, where *n* = 24 + the axis number **RMC200:** %MD*n*.6, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].HaltGrpNo, where *n* is the axis number
How to Find:	Axes Parameters Pane, All tab: Halt Group Number

Data Type:	DINT
Units:	none
Range:	**RMC75:** 0 (none), 1 to 2, whole numbers only **RMC150:** 0 (none), 1 to 8, whole numbers only **RMC200:** 0 (none), 1 to 32, whole numbers only
Default Value:	0

Description

This parameter defines which halt group the axis is in. If any member of a halt group halts, all axes in that group will halt with the same halt type. Zero indicates that the axis is not a member of a halt group.

Sync Groups for synchronized axes have similar grouped halt functionality. Halts in any group will propagate recursively to other groups. If an axis halts, all axes in that group will halt, and so will the axes in any other sync group or halt group that the other axes are part of.

Why Bother?

For systems such as presses with 2 or more cylinders, where it is very important that the cylinders never get skewed, you typically want all the axes to halt when one axis halts. This parameter allows you to easily do that.

See Also

Parameter Registers | Halts Overview

9.2.3.8.3. Closed Loop Halt Deceleration

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.110, where *n* = 12 + the axis number **RMC150:** %MD*n*.110, where *n* = 24 + the axis number **RMC200:** %MD*n*.4, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].CLHaltDecel, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Halts
Data Type:	REAL
Units:	pu/sec^2
Range:	> 0
Default Value:	100

Description

This register specifies the rate at which an axis is decelerated to zero velocity in closed loop control due to a Closed Loop Halt. A Closed Loop Halt can be initiated by the Closed Loop Halt (1) command or an Auto Stop.

Note:
The deceleration specified by the Closed Loop Halt Deceleration is the *average* deceleration. The instantaneous deceleration may exceed this value.

See Also

Parameter Registers | Closed Loop Halt

9.2.3.8.4. Open Loop Halt Ramp

Type:	Axis Parameter Register
Address:	**RMC75:** %MD*n*.111, where *n* = 12 + the axis number **RMC150:** %MD*n*.111, where *n* = 24 + the axis number **RMC200:** %MD*n*.5, where *n* = 384 + the axis number
System Tag:	_Axis[*n*].OLHaltRamp, where *n* is the axis number
How to Find:	Axes Parameters Pane, Setup tab: Halts
Data Type:	REAL
Units:	**RMC75/150:** V/s **RMC200:** %/s
Range:	≥ 0
Default Value:	**RMC75/150:** 100 **RMC200:** 1000

Description

Note: The units differ between the RMC75/150 and RMC200, since the RMC75/150 Control Output is voltage, and the RMC200 Control Output is percent.

This register specifies the rate at which the output is ramped to zero volts due to an Open Loop Halt or Direct Output Halt. An Open Loop Halt can be initiated by the Open Loop Halt (2) command or an Auto Stop. A Direct Output Halt can be initiated by the Direct Output Halt (3) command or an Auto Stop.

See Also

Parameter Registers | Open Loop Halt

9.2.4. Communication Registers

9.2.4.1. Ethernet Status

Type:	Communications Register
RMC75 Address:	%MD21.10
RMC150 Address:	%MD45.10
RMC200 Address:	n/a (not accessible via addressing)
System Tag:	_Enet.Status
How to Find:	Address Selection Tool → Controller → Communication Settings → Ethernet
Data Type:	DWORD

Accessibility:	Read Only

This register indicates the state of various Ethernet communication items.

Bit	Tag Name	Description
0	_Enet.Status.LinkUp	**Link** **True:** The Ethernet link is up, and the Ethernet Link/Act LED will be green (steady or flashing). **False:** The Ethernet link is down.
1	_Enet.Status.100Mbps	**100 Mbps** **True:** The Ethernet speed is 100Mbps **False:** The Ethernet speed is 10Mbps
2	_Enet.Status.FullDuplex	**Full Duplex** **True:** The Ethernet is full-duplex. **False:** The Ethernet is half-duplex.
4-7	_Enet.Status.IPState	**Current IP Address State** **0:** Static Method, but an unusable address is assigned (e.g. 0x00000000) **1:** IP Address assigned and in use **2:** IP Address assigned, but disabled due to duplicate detected **3:** IP Address assigned, but still doing Address Collision Detection **4:** BOOTP: Still querying, no address yet assigned **5:** DHCP: Still querying, no address yet assigned

See Also

Ethernet Configuration Bits

9.2.4.2. Ethernet Configuration Bits

Type:	Communications Register
RMC75 Address:	%MD21.11
RMC150 Address:	%MD45.11
RMC200 Address:	n/a (not accessible via addressing)
System Tag:	_Enet.Config
How to Find:	Address Selection Tool → Controller → Communication Settings → Ethernet
Data Type:	DWORD
Accessibility:	Read/Write

This register indicates the state of various Ethernet communication items.

Bit	Tag Name	Description

0	_Enet.Config.ManConfig	**Disable Auto-Negotiation** **True:** Auto-Negotation is disabled. **False:** Use Auto-Negotation (use bits 1 and 2).
1	_Enet.Config.Man100Mbps	**100 Mbps** Valid only if _Enet.Config.ManConfig is True. **True:** Ethernet speed is set to 100 Mpbs. **False:** Ethernet speed is set to 10 Mpbs.
2	_Enet.Config.ManFDX	**Full Duplex** Valid only if _Enet.Config.ManConfig is True. **True:** Ethernet is set to full-duplex. **False:** Ethernet is set to half-duplex.
3-4	_Enet.Config.IPAddrMode	**IP Address Assignment Mode** Specifies how the RMC obtains its IP address. **0:** Statically Assigned **1:** BOOTP **2:** DHCP
5	_Enet.Config.LockIP	**Lock IP Settings** **True:** IP address settings cannot be changed via Ethernet. **False:** IP address settings can be changed via Ethernet.
6	_Enet.Config.LockFW	**Lock Firmware Update via Ethernet** **True:** Firmware cannot be updated via Ethernet. **False:** Firmware can be updated via Ethernet.

See Also

Ethernet Status Bits

9.2.4.3. I/O Connection Status

Type:	Communications Register
RMC75 Address:	%MD21.23
RMC150 Address:	%MD45.23
RMC200 Address:	EtherNet/IP Connection #1: %MD112.0 EtherNet/IP Connection #2: %MD112.4 EtherNet/IP Connection #3: %MD112.8 PROFINET I/O Connection: %MD112.12
System Tag:	See table below
How to Find:	Address Selection Tool → Controller → Communication Settings → Ethernet
Data Type:	DWORD
Accessibility:	Read Only

These registers indicate the state of the EtherNet/IP and PROFINET I/O communications. For more details, see Handling Broken EtherNet/IP I/O Connections and Handling Broken PROFINET Connections.

For the RMC75 and RMC150, there is a single I/O Connection Status register shared by EtherNet/IP and PROFINET. If there is an EtherNet/IP Exclusive Owner I/O connection open, this register will reflect the state of that connection. If there is a PROFINET I/O connection open, then this register will reflect the state of that connection.

For the RMC200, there is one I/O Connection Status register dedicated to each EtherNet/IP and PROFINET I/O connection.

The bits within each register are defined as follows for the respective connection:

Bit	RMC75/150 Tag Name	RMC200 Tag Name	Description
0	_Enet.CCStatus.Active	_Enet.IOConn1Status.Active _Enet.IOConn2Status.Active _Enet.IOConn3Status.Active _Enet.PNIOConnStatus.Active	**I/O Connection Active** This bit is set as long as an I/O connection is currently active. If the connection is closed or timed out, this bit will be cleared.
1	_Enet.CCStatus.TimedOut	_Enet.IOConn1Status.TimedOut _Enet.IOConn2Status.TimedOut _Enet.IOConn3Status.TimedOut _Enet.PNIOConnStatus.TimedOut	**I/O Connection Timed Out** This bit is set when an I/O connection timed out. Notice that this is only one method of a connection being closed (another example is the PLC intentionally closing it). This bit is cleared when the I/O connection is re-opened. The user can look for the rising edge of this bit in the Program Triggers to respond to a time-out. A time-out can occur when the cable is disconnected, or when the client is powered off or reset. This bit is not used for PROFINET on the RMC75/150.

See Also

Handling Broken EtherNet/IP I/O Connections | Handling Broken PROFINET Connections

9.2.4.4. I/O Connection PLC Status

Type:	Communications Register
RMC75 Address:	%MD21.24
RMC150 Address:	%MD45.24
RMC200 Address:	EtherNet/IP Connection #1: %MD112.1 EtherNet/IP Connection #2: %MD112.5 EtherNet/IP Connection #3: %MD112.9 PROFINET I/O Connection: %MD112.13
RMC75/150 System Tag:	_Enet.PLCStatus
RMC200 System Tag:	EtherNet/IP Connection #1: _Enet.IOConn1PLCState EtherNet/IP Connection #2: _Enet.IOConn2PLCState EtherNet/IP Connection #3: _Enet.IOConn3PLCState PROFINET I/O Connection: _Enet.PNIOConnPLCState
How to Find:	Address Selection Tool → Controller → Communication Settings → Ethernet
Data Type:	DINT
Accessibility:	Read Only

These registers indicate the RUN/STOP (or RUN/PROGRAM) state of the I/O controller for EtherNet/IP and PROFINET I/O communications. For more details, see Handling Broken EtherNet/IP I/O Connections and Handling Broken PROFINET Connections.

For the RMC75 and RMC150, there is a single PLC Status register shared by EtherNet/IP and PROFINET. If there is an Exclusive Owner EtherNet/IP I/O connection open, this register will reflect the state of the PLC controlling that connection. If there is a PROFINET I/O connection open, then this register will reflect the state of the I/O controller (PLC) for that connection.

For the RMC200, there is one PLC Status register dedicated to each EtherNet/IP and PROFINET I/O connection.

Each register can hold one of the following three values:

Value	Description
0	**Unknown** Applies when the I/O connection is not currently active.
1	**RUN**
2	**PROGRAM or STOP**

See Also

Handling Broken EtherNet/IP I/O Connections | Handling Broken PROFINET Connections

9.2.4.5. PROFIBUS Connection Status

Type:	Communications Register
RMC75 Address:	%MD21.6
RMC150 Address:	%MD45.6
RMC200 Address:	n/a
System Tag:	_PROFI.Status
How to Find:	Address Selection Tool → Controller → Communication Settings → PROFIBUS
Data Type:	DWORD
Accessibility:	Read Only

This register indicates the state of the PROFIBUS connection.

Bit	Tag Name	Description
0	_PROFI.Status.Connected	**Connection Established** This bit will be set when the PROFIBUS interface is in the Data Exchange mode.

See Also

PROFIBUS Overview

9.2.5. Task Registers

9.2.5.1. Task Status

Type:	Task Register
RMC75 Address:	%MD24.16*n*, *n* = task number. See Register Map topic for other address formats.
RMC150 Address:	%MD48.16*n*, *n* = task number. See Register Map topic for other address formats.
RMC200 Address:	%MD.[192+*n*].0, *n* = task number.
System Tag:	_Task[*n*].Status, where *n* is the Task number
How to Find:	Address Selection Tool → Tasks → Task #
Data Type:	DWORD
Accessibility:	Read Only

This register indicates the state of Task *n*. You can use this register to determine whether a specific Task is running and whether it is allocated.

After issuing a Start Task command from a PLC, the Running bit, together with the Current Program register, is useful for determining if the User Program really did start.

Bit	Tag Name	Description
0	Running	0 = Stopped, 1 = Running
1	Allocated	0 = Not Allocated, 1 = Allocated A task is *allocated* if it exists. To specify the number of tasks that exist, use the **General** page on the Programming Properties dialog. The **Number of User Tasks** box defines how many tasks exist. For example, if you set the **Number of User Tasks** to 3, then tasks 0 to 2 will be allocated.

See Also
Tasks

9.2.5.2. Current Program

Type:	Task Register
RMC75 Address:	%MD24.3+16*n*, *n* = task number. See Register Map topic for other address formats.
RMC150 Address:	%MD48.3+16*n*, *n* = task number. See Register Map topic for other address formats.
RMC200 Address:	%MD.[192+*n*].3, *n* = task number.
System Tag:	_Task[*n*].CurProg, where *n* is the Task number
How to Find:	Address Selection Tool → Tasks → Task #
Data Type:	DINT
Accessibility:	Read Only

This register indicates the current program that is running on Task *n*. The Current Step register indicates the current step.

After issuing a Start Task command from a PLC, this register, together with the Running bit in the Task Status register, is useful for determining if the User Program really did start.

Note:
If this register is referenced in the action portion of a step in a user program, it will give the number of the program that was running in the previous step of the task. If referenced from the Link or from any other location, such as the Program Triggers or from a host controller, it will always indicate the current program.
Therefore, this register is best not used in the action portion of a step in a user program.

See Also
Tasks

9.2.5.3. Current Step

Type:	Task Register
RMC75 Address:	%MD24.4+16*n, n = task number. See Register Map topic for other address formats.
RMC150 Address:	%MD48.4+16*n, n = task number. See Register Map topic for other address formats.
RMC200 Address:	%MD.[192+n].4, n = task number.
System Tag:	_Task[n].CurStep, where n is the Task number
How to Find:	Address Selection Tool → Tasks → Task #
Data Type:	DINT
Accessibility:	Read Only

This register indicates the current step that is running on Task *n*. The Current Program register indicates the current program.

Note:
If this register is referenced in the action portion of a step in a user program, it will give the number of the step that was previously executed by the task. If referenced from the Link or from any other location, such as the Program Triggers or from a host controller, it will indicate the current step. Therefore, this register is best not used in the action portion of a step in a user program.

See Also
Tasks

9.2.5.4. Current Axis

Type:	Task Register
RMC75 Address:	%MD24.2+16*n, n = task number. See Register Map topic for other address formats.
RMC150 Address:	%MD48.2+16*n, n = task number. See Register Map topic for other address formats.
RMC200 Address:	%MD.[192+n].0, n = task number.
System Tag:	_Task[n].CurAxis, where n is the Task number _CurAxis (indicates the current axis of the current Task)
How to Find:	Address Selection Tool → Tasks → Task #
Data Type:	DINT
Accessibility:	Read/Write

This register indicates the current axis (Default Axis) of Task *n*.

Each task has a Default Axis associated with it. If a command in a User Program does not have a selected Commanded Axis, the command will be issued to this Default Axis.

See the Tasks topic for more details on the Default Axis.

See Also

Tasks

9.2.5.5. Current Program/Step

Type:	Task Register
RMC75 Address:	%MD24.1+16*n, n = task number. See Register Map topic for other address formats.
RMC150 Address:	%MD48.1+16*n, n = task number. See Register Map topic for other address formats.
RMC200 Address:	%MD.[192+n].1, n = task number.
System Tag:	_Task[n].CurProgStep, where n is the Task number
How to Find:	Address Selection Tool → Tasks → Task #
Data Type:	DINT
Accessibility:	Read Only

This register indicates the current program and step that is running on Task *n*. This register combines the information provided in both the Current Program and Current Step registers. Therefore, it provides information more compactly, which is useful when including Task information in plots.

Other than using this register for plots, this register is difficult to use! Use the Current Program and Current Step registers instead.

Note:
If this register is referenced in the action portion of a step in a user program, it will give the numbers of the program and step that were executed in the previous step by the task. If referenced from the Link or from any other location, such as the Program Triggers or from a host controller, it will always indicate the current program and step.
Therefore, this register is best not used in the action portion of a step in a user program.

This register includes the current program and step in the following bits of the register:

Bit	Description
0-11	Step Number
12-23	Program Number

See Also
Tasks

9.2.6. Controller Registers

9.2.6.1. Controller Status

Type:	Controller Info Register

RMC75 Address:	%MD20.7
RMC150 Address:	%MD44.7
RMC200 Address:	%MD18.7
System Tag:	_Controller.Status
How to Find:	Address Selection Tool → Controller → Status
Data Type:	DWORD
Accessibility:	Read Only

This register indicates the state of various controller items. Use this register to determine whether the controller is enabled, if it is in Run or Program mode, or to check the First Scan bit.

Bit	Tag Name	Description
0	_Controller.Status.Run	**RUN/PROGRAM Mode** This bit indicates whether the controller is in RUN (1) or PROGRAM (0) mode. See the RUN/PROGRAM Mode topic for details. On the RMC200, the next bit defines Disabled mode.
1	_Controller.Status.Enabled	**Controller Enabled** This bit is initially cleared on power-up, but will be set after the Enable Controller (7) or RUN Mode (98) command is issued. It will also be set immediately if the controller is set up to start in RUN mode. On the RMC200, when this bit is off, the RMC is in Disabled mode.
2	_Controller.Status.FirstScan	**First Scan** Goes high (1) on first scan after entering RUN mode, otherwise it is low (0). See the Program Triggers topic for usage details. The system tag _FirstScan is identical to using _Controller.FirstScan.
3	_Controller.Status.WDReset	**Restart due to Watchdog Timeout** If the last controller startup was the result of a watchdog timeout causing the RMC to restart, this bit will be set until the RMC is powered off. A watchdog timeout is typically caused by a bug in the firmware. If the RMC did not restart due to a watchdog timeout or unhandled exception, then it was because of a power problem.

		After any restart caused by a watchdog timeout, the RMC CPU LED will alternately flash red and green for 10 seconds.
4	RMC75/150: no tag name RMC200: _Controller.Status.SysFault	**Restart due to Unhandled Exception** If the last controller startup was the result of an unhandled exception causing the RMC to restart, this bit will be set until the RMC is powered off. After any restart caused by an unhandled exception, the RMC CPU LED will alternately flash red and green for 10 seconds.

See Also

Help Overview

9.2.6.2. Controller Tags

In addition to the controller registers listed in the RMC150 register map (file 7), RMC75 register map (file 7), and RMC200 register map (file 18), the RMC includes the following system tags. These are not directly accessible from outside the RMC. From within the RMC, they can be accessed using the tag names.

Tag Name	Data Type	How to Find in Tags Browser	Description
_FirstScan	BOOL	Controller→ First Scan	True on the first loop time after entering RUN mode, otherwise it is false. See the Program Triggers topic for usage details.

See Also

Tags Overview | Registers Overview

9.2.6.3. Controller Loop Time Status Registers

The controller loop time status registers provide information on the utilization of the loop time.

Register Name	Units	Data Type	Tag Name	RMC200 Address	RMC75 Address	150 Address
Loop Time Used, Last	s	REAL	_Controller.LoopTimeUsedLast	%MD18.2	%MD20.2	%MD44.2
Loop Time	s	REAL	_Controller.LoopTimeUsedMax	%MD18.3	%MD20.3	%MD44.3

Used, Maximum						
Loop Time Used, Minimum	s	REAL	_Controller.LoopTimeUsedMin	%MD18.4	n/a	n/a
Loop Time Used, Total	µs	REAL	_Controller.LoopTimeUsedTotal	%MD18.24	%MD20.34	%MD44.34
Loop Time Used, Axes	µs	REAL	_Controller.LoopTimeUsedAxes	%MD18.25	%MD20.35	%MD44.35
Loop Time Used, Programs	µs	REAL	_Controller.LoopTimeUsedPrograms	%MD18.26	%MD20.36	%MD44.36
Loop Time Used, Plots	µs	REAL	_Controller.LoopTimeUsedPlots	%MD18.27	%MD20.37	%MD44.37
Loop Time Used, Comm	µs	REAL	_Controller.LoopTimeUsedComm	%MD18.28	%MD20.38	%MD44.38
Loop Time Used, Overhead	µs	REAL	_Controller.LoopTimeUsedOverhead	%MD18.29	%MD20.39	%MD44.39

Using the Controller Loop Time Status Registers

The controller loop time status registers are useful for determining how much of the loop time is being used. Adding the registers to a plot provides a good visualization. See Plotting the Loop Time for more details.

General Loop Time Status

The following registers are in units of seconds, and help provide general information:

- **Loop Time Used, Last**
  The utilization of the last loop time. Including this item on a plot gives a clear indication of the utilization of each loop time. However, it is typically easier to use **Loop Time Used, Total**, since it is in units of microseconds, which scales better on the plot.
- **Loop Time Used, Maximum**
  The maximum utilization of all loop times, in seconds, since the RMC powered up or since this value was reset.
- **Loop Time Used, Minimum (RMC200 only)**
  The minimum utilization of all loop times, in seconds, since the RMC powered up or since this value was reset.

To reset the **Loop Time Used, Maximum** and **Loop Time Used, Minimum**:

- In the Project View, expand **Modules** and double-click the CPU.
- On the **Control Loop** page, in the **Loop Time Usage Statistics** section, click **Clear**.

Detailed Loop Time Status

Including these registers in a plot provides detailed information on the utilization of each section of the loop time. This helps determine how to reduce the loop time utilization if it is high. These registers are in units of microseconds.

- **Loop Time Used, Total**
  The total utilization of the last loop time, which is the sum of the following registers.
- **Loop Time Used, Axes**
  The loop time utilization of axes, including processing inputs, generating the target profiles, computing the control output, and processing commands.
- **Loop Time Used, Programs**
  The loop time utilization of the user programs and Program Triggers.
- **Loop Time Used, Plots**
  The loop time utilization of the plots.
- **Loop Time Used, Comm**
  The loop time utilization of the communication. This is only the portion of the communications that occurs in the motion loop, which is small compared to the overall communications.
- **Loop Time Used, Overhead**
  The loop time utilization of all other items.

See Also

Loop Time | Plotting the Loop Time

9.2.6.4. System Time Registers

These time registers keep track of system time (time since the RMC started up) or real time. These registers can be used for functions such as delay timers, calculating time between events, and comparing motion controller events with the times of other devices.

All registers listed below are Read Only.

Register Name	Data Type	Time Base	Rollover	Tag Name	RMC75 Address	RMC150 Address	RMC200 Address
System Time - Time base starts when RMC powers up							
Time Gear Master	REAL	1.0 s	1 s	_Time	%MD20.10	%MD44.10	%MD18.10
Time, 16th Milliseconds	DINT	0.000062 5 s	1.5 days (2^{31} ms/16)	_Controller.SysTime_16thmsec	%MD20.11	%MD44.11	%MD18.11
Time, Seconds	DINT	1.0 s	68 years (2^{31} s)	_Controller.SysTime_sec	%MD20.12	%MD44.12	%MD18.12
Time, Milliseconds	DINT	0.001 s	24.8 days (2^{31} ms)	_Controller.SysTime_msec Or: _SysMS	%MD20.13	%MD44.13	%MD18.14
Time, Microseconds	DINT	0.000001 s	35.7 minutes (2^{31} µs)	_Controller.SysTime_usec	%MD20.14	%MD44.14	%MD18.15

Time, Nanoseconds	DINT	0.000 000 001 s	1 second (1 billion ns)	_Controller.SysTime_nsec	%MD20.15	%MD44.15	%MD18.16
Time, Loop Ticks	DINT	Same as the Loop Time (0.000 125 s to 0.008 s)	3.1 - 198 days (2^{31} loop times)	_Controller.SysTime_loops Or: _SysTicks	%MD20.33	%MD44.33	%MD18.17
Real Time - Time base starts Jan. 1, 1970							
Real Time UTC, Seconds	DINT	1.0 s	February 7, 2106 (2^{32} s)	_Controller.RealTimeUTC_sec	n/a	n/a	%MD18.18
Real Time Local, Seconds	DINT	1.0 s	February 7, 2106 (2^{32} s)	_Controller.RealTimeLocal_sec	n/a	n/a	%MD18.20
Real Time, Nanoseconds	DINT	0.000 000 001 s	1 second (1 billion ns)	_Controller.RealTime_nsec	n/a	n/a	%MD18.19

For details on using these registers, see the System Time topic.

See Also

System Time | RMC200 Real-Time Clock

9.2.6.5. Loader Command (RMC75/150) Reset Command (RMC200)

Type:	Controller Info register
RMC75/150 Address:	%MD7.28
RMC200:	%MD18.9
Data Type:	RMC75/150 Internal: DINT RMC75/150 External: REAL RMC200: DINT

Description

The Loader Command/Reset Command register is for sending certain commands to the loader. Currently, it supports commands for restarting the RMC75, RMC150 and RMC200.

Restarting the RMC

The RMC can be restarted from RMCTools, on the **Controller** menu, using the **Restart Controller** item. The RMC can also be restarted by writing certain values to the Loader Command/Reset Command register. Restarting the RMC in this manner is only done in advanced applications, such as programmatically changing the axis definitions during machine operation.

There are several ways to restart the RMC:

1. **Cold Restart with Flash Update**
   This method first updates Flash, then waits one second before doing a cold restart of the RMC, which is the same as cycling power. Use this method if you want to automatically update Flash and restart. To do this:
   a. Write 16#55AA to the Loader Command/Reset Command register.
   b. Write 16#000D to the Loader Command/Reset Command register.
2. **Cold Restart without Flash Update**
   The controller will restart in approximately 1 second. This is a cold restart of the RMC, which is the same as cycling power. Anything not saved to Flash will be lost.
   a. Write 16#55AA to the Loader Command/Reset Command register.
   b. Write 16#000C to the Loader Command/Reset Command register.
3. **Warm Restart**
   The controller will do a warm restart in approximately 1 second. A warm restart, which retains all the data in the RMC, but doesn't save to Flash. Notice that the restart will apply any new axis definitions and will set all variables to initial values.
   a. Write 16#55AA to the Loader Command/Reset Command register.
   b. Write 16#000E to the Loader Command/Reset Command register.

Note: On the RMC75/150, if the writes are done externally, such as from a host controller such as a PLC, the values must be in floating-point format. 16#55AA would be written as 21930 in floating-point, and 16#000C, 16#000D, and 16#000E are 12, 13 and 14, respectively, in floating-point.

After the RMC restarts, it will take several seconds before it begins to communicate again. Over USB or serial, it will be approximately 4 seconds, over Ethernet, approximately 8 seconds.

See Also

Axis Definition Registers

9.2.7. Variables

9.2.7.1. Variable Attributes

This topic describes the attributes registers for the variables in the Variable Table.

RMC75/150

RMC75 Variable Attribute registers: %MD68.0-255

RMC150 Variable Attribute registers: %MD88.0-255

Each attribute register contains the attributes for four variables:

Attribute Register	Bits	Description

%MD68.*n* or **%MD88.*n***	0-3	**Data Type for Variable *n*×4:** - REAL (0) - DINT (1) - DWORD (2)
	4	**Retained for Variable *n*×4:** - Not retained (0) - Retained (1)
	5-7	Reserved
	8-11	**Data Type for Variable *n*×4+1:** - REAL (0) - DINT (1) - DWORD (2)
	12	**Retained for Variable *n*×4+1:** - Not retained (0) - Retained (1)
	13-15	Reserved
	16-19	**Data Type for Variable *n*×4+2:** - REAL (0) - DINT (1) - DWORD (2)
	20	**Retained for Variable *n*×4+2:** - Not retained (0) - Retained (1)
	21-23	Reserved
	24-27	**Data Type for Variable *n*×4+3:** - REAL (0) - DINT (1) - DWORD (2)
	28	**Retained for Variable *n*×4+3:** - Not retained (0) - Retained (1)
	29-31	Reserved

RMC200

RMC200 Variable Attribute registers: %MD1536.0-4095
Each attribute register contains the attributes for one variable:

Attribute Register	Bits	Description
%MD1536.*n*	0-3	**Data Type for Variable *n*:** - REAL (0) - DINT (1) - DWORD (2)
	4	**Retained for Variable *n*:** - Not retained (0)

		- Retained (1)
	5-31	Reserved

See Also

Variables

9.3. Address Formats

9.3.1. Address Formats Overview

Each register in the RMC can be addressed with several different address formats. The table below lists the formats. For more details on each format, see the respective format.

Address Format	Example	View in RMCTools	Use In Expressions	From External Devices	Multi-level/ Flat
DF1	F56:0	✓		✓	Multi-level
Modbus	28673	✓		✓	Flat
FINS	D28672	✓		✓	Flat
IEC	%MD56.0	✓	✓		Multi-level

Viewing Addresses in RMCTools

RMCTools displays the addresses of many registers, such as in the Axis Tools. To change the address format, right-click any register address cell, choose **Address Formats**, and choose the desired format.

Using Addresses in Expressions

The Expressions in RMCTools typically use tag names for the RMC registers, although the IEC address format can also be used. Certain infrequently-used registers can only be addressed with the IEC format.

Address Formats from External Devices

Several of the address formats can be used from an external controller, such as a PLC. The communication protocol determines which format will be used. For details, see the specific address format in the table above.

Multi-level vs. Flat Addresses

Some of the address types are *multi-level*. This means that the address of any 32-bit register consists of two numbers, for example, %MD8.0. The first number is called the *file*. The second number is called the *element*.

A *flat* address format has only one number. For example, 4192.

Finding Addresses for RMC Registers

When configuring the host controller (such as a PLC or HMI) to communicate with an RMC, you will need to enter the addresses of the RMC's registers in the correct format for the addressing method that the protocol uses.

Ways to find RMC addresses:

1. In RMCTools
   RMCTools displays the addresses of many registers, such as in the Axis Tools. To change the address format, right-click any register address cell, choose **Address Formats**, and choose the desired format.
2. Use the Address Maps in RMCTools to browse all the register addresses for any addressing method.
3. Use the Register Map help topic to browse all the register addresses for any addressing method.

See Also

DF1 Addressing | FINS Addressing | IEC Addressing | Modbus Addressing

9.3.2. DF1 (Allen Bradley) Addressing

This topic describes the Allen-Bradley DF1 addressing format as used by the RMC for the following communications:

- CSP or EtherNet/IP with the RMC75E, RMC150E, or RMC200
- DF1 (Full- and Half-Duplex) with the RMC75S.

Finding DF1 Addresses for RMC Registers

When configuring the host controller (such as a PLC or HMI) to communicate with an RMC, you will need to enter the DF1 addresses of the RMC's registers that you wish to read from or write to.

Ways to find RMC addresses:

1. In RMCTools editors:
   Some register addresses are displayed in RMCTools, such as in the **Reg #** column in Axis Tools, Indirect Data Map, and the Variable Table. To view those addresses in various formats, in the **Reg#** column, right-click the address and choose Address Formats.
2. Use the Address Maps in RMCTools to browse all the register addresses for any addressing method.
3. Use the Register Map help topic to browse all the register addresses for any addressing method.

RMC75 and RMC150 DF1 Addresses

The DF1 address format uses the same two-level numbering as the RMC's IEC addresses. Any RMC2 DF1 address can be accessed as an F or L register. L register addressing is useful for DINTs and DWORDS.The format for the registers in the RMC is:

F*n*.*x* or L*n*:*x*

where *n* = File number and *x* = the register number.

Example

For the RMC75, F8:8 is the Axis 0 Actual Position, and F8:0 (or L8:0) is the Axis 0 Status.

Supported File Types

The RMC uses F and L files for the DF1 protocol. The DF1 protocol specifies F files as 32-bit floating point registers and L files 32-bit integers.

Every register in the RMC can be addressed as either an L or F file register. Typically, it only makes sense to address REAL registers in the RMC as F file registers, and address DINT or DWORD registers as L file registers.

Since the only AB PLC at the time of this writing to support L files is the MicroLogix, you may have to set up the DF1 communications on your host system as if the RMC were a MicroLogix if you wish to use L file addressing.

Internal versus External Data Types

Notice that some RMC75 and RMC150 registers have different data types depending on whether they are accessed from user programs or externally, such as from a PLC. The RMC150 Register Map and RMC75 Register Map topics list the external and internal data types of the registers. The RMC200 does not have separate internal and external data types.

RMC200 DF1 Addresses

The RMC200 DF1 addressing includes the pre-defined addresses as listed below. The remaining DF1 addresses may be configured in the Address Maps.

Any RMC200 DF1 address can be accessed as an F or L register. L register addressing is useful for DINTs and DWORDS.

Some registers can be addressed with two different addresses, depending on whether the communication master device can address only 256 elements per file, or up to 1024 or 4096 elements per file. For example, axis 24 command area starts at F12:240 or F13:0, and variable 512 can be addressed as F20:512, or as F22:0.

Description	RMC200 IEC Address	RMC200 DF1 Address
Reserved		F8-F11
Command Area, axes 0-127 (10 registers per axis)	%MD16.0-1279	F12:0-1279
Command Area, axes 0-23 (10 registers per axis)	%MD16.0-239	F12:0-239
Command Area, axes 24-47 (10 registers per axis)	%MD16.240-479	F13:0-239
Image Upload/Download Area	%MD23.0-4095	F14:0-4095
Indirect Data Map 0-1023	%MD8.0-1023	F16:0-1023
Indirect Data Map 0-255	%MD8.0-255	F16:0-255
Indirect Data Map 256-511	%MD8.256-511	F17:0-255
Indirect Data Map 512-767	%MD8.512-767	F18:0-255
Indirect Data Map 768-1023	%MD8.768-1023	F19:0-255
Variables (Current Values) 0-4095	%MD1024.0-4095	F20:0-4095
Variables (Current Values) 0-255	%MD1024.0-255	F20:0-255
Variables (Current Values) 256-511	%MD1024.256-511	F21:0-255
Variables (Current Values) 512-767	%MD1024.512-767	F22:0-255
Variables (Current Values) 768-1023	%MD1024.768-1023	F23:0-255
Variables (Current Values) 1024-2047	%MD1024.1024-2047	F24:0-1023
Variables (Current Values) 1024-1279	%MD1024.1024-1279	F24:0-255
Variables (Current Values) 1280-1535	%MD1024.1280-1535	F25:0-255
Variables (Current Values) 1536-1791	%MD1024.1536-1791	F26:0-255
Variables (Current Values) 1792-2047	%MD1024.1792-2047	F27:0-255

Variables (Current Values) 2048-3071	%MD1024.2048-3071	F28:0-1023
Variables (Current Values) 2048-2303	%MD1024.2048-2303	F28:0-255
Variables (Current Values) 2304-2559	%MD1024.2304-2559	F29:0-255
Variables (Current Values) 2560-2815	%MD1024.2560-2815	F30:0-255
Variables (Current Values) 2816-3071	%MD1024.2816-3071	F31:0-255
Variables (Current Values) 3072-4095	%MD1024.3072-4095	F32:0-1023
Variables (Current Values) 3072-3327	%MD1024.3072-3327	F32:0-255
Variables (Current Values) 3328-3583	%MD1024.3328-3583	F33:0-255
Variables (Current Values) 3584-3839	%MD1024.3584-3839	F34:0-255
Variables (Current Values) 3840-4095	%MD1024.3840-4095	F35:0-255

Addressing Individual Bits

Some RMC registers contain individual bits that you may wish to address. The DF1 address format for a specific bit is shown below. Notice that many host controllers cannot address individual bits in an F file.

F*n*.*x*/*b* or L*n*:*x*/*b*, where *n* and *x* are defined above, and *b* = bit number.

Note:
The RMC does not support reading or writing of individual bits. You must read or write an entire 32-bit word. Some HMIs allow individual bit addressing as described above, but still read the entire word.

Example

In the RMC75, the user would like the address for the Enable Output Status Bits on axis 0. Then,

n = 8, the file number for Axis 0 Status registers,

x = 0, the number of the Status Bits register,

and

b = 7, the bit number for the Enable Output bit.

Therefore, the address is F8:0/7 or L8:0/7.

Discrete I/O

Discrete I/O are mapped to the Discrete I/O registers and can be addressed by addressing the bits in the register. See the RMC150 DI/O, RMC75 DI/O, and RMC200 DI/O register map topics for the addresses of the I/O.

Example

For the RMC75, F23:0/5 references the state of discrete I/O point 5.

For the RMC150, F47:6/3 references the state of discrete input 3 in slot 0.

For the RMC200, add the discrete input or discrete output register to the Indirect Data Map, then address that register and bit from the Indirect Data Map.

Register Addresses in Integer Format

Occasionally, an address may need to be represented in integer format, for example when used in commands that contain a Master Register parameter. This section describes how to convert register addresses from the standard register representation to an integer.

RMC addresses are normally represented in the following Data Type:

Fn:x, where n = File number, and x = Element number.

Use the following equation to convert a register address to integer format, N:

N = n * 4096 + x

Example

Register address F8:33 is 8*4096 + 33 = 32801.

See Also

Register Map Overview

9.3.3. IEC-61131 Addressing

This topic describes the IEC addressing format as used in the RMC. For other addressing formats, see the Register Map Overview topic.

The RMC uses the IEC-61131 standard for its register addresses. This format can be used in User Programs and Expressions, although Delta recommends using tag names where possible.

Address Format

The IEC address format for the registers in the RMC is:

%MD*file*.*element*

where

M indicates it is a location in memory.

D indicates it is a 32 bit word.

file = the file number. This file number corresponds to the file number in the DF1 addressing.

element = the element number. This element number corresponds to the element number in the DF1 addressing.

Example

For the RMC75, %MD8.8 is the Axis 0 Actual Position, and %MD8.0 is the Axis 0 Status.

Individual Bits

Some RMC registers contain individual bits that you may wish to address. The IEC address for a specific bit in the RMC is:

%MX*file*.*element*.*bit*

where

M indicates it is a location in memory.

X indicates it is a single bit.

file = the file number.

element = the element number.

bit = the bit number (0-31).

Example

The user would like the address for the RMC75 Enable Output Status Bit on axis 0. Then,

file = 8, the file number for Axis 0 Status registers,

element = 0, the element number of the Status Bits register,

and

bit = 7, the bit number for the Enable Output bit.

Therefore, the address is %MX8.0.7

Discrete I/O

Discrete Inputs and Outputs can be addressed in IEC format as shown below. The IEC address is also displayed in the Discrete I/O Monitor and the Discrete I/O Configuration dialog.

RMC75	RMC150 and RMC200
%IX*number* or %QX*number* where **I** indicates it is an input. **Q** indicates it is an output. **X** indicates it is a single bit. ***number*** = the I/O number.	%IX*slot*.*number* or %QX*slot*.*number* where **I** indicates it is an input. **Q** indicates it is an output. **X** indicates it is a single bit. ***slot*** = the slot number of the RMC, starting with slot 0 to the left. ***number*** = the input or output number.

Discrete I/O are also mapped to the Discrete I/O registers and can be addressed by addressing the bit in the register. See the RMC150 DI/O, RMC75 DI/O, and RMC200 DI/O register map topics for the addresses of the I/O.

Register Addresses in Integer Format

Occasionally, an addresses may need to be represented in integer format. This section describes how to convert register addresses from the standard register representation to an integer.

RMC addresses are represented in IEC 61131 address format as:

%MD*file*.*element*, where *file* = file number, and *element* = element number.

Use the following equation to convert a register address to integer format, N:

N = *file* * 4096 + *element*

Example

Register address %MD8.33 is 8*4096 + 33 = 32801.

See Also

Register Map Overview

9.3.4. FINS (Omron) Addressing

This topic describes the FINS addressing format as used by the RMC75, RMC150, and RMC200. For other addressing formats, see the Register Map Overview topic. The FINS address format must be used when communicating with the RMC via the FINS protocol. The FINS protocol is used by Omron PLCs.

Address Format

The RMC memory uses the D and E memory areas. The FINS addressing assigns a single number to each register in RMC memory. In the FINS protocol, the addresses are 0-based. Because the FINS protocol is 16-bit based, and the RMC has 32-bit registers, the addresses of RMC registers are even.

The RMC interprets the current E memory as the first extended memory E0_. For example, E01200 is interpreted as E0_01200.

Finding FINS Addresses for RMC Registers

When configuring the host controller (such as a PLC or HMI) to communicate with an RMC, you will need to enter the FINS addresses of the RMC's registers that you wish to read from or write to.

Ways to find RMC addresses:

1. In RMCTools editors:
   Some register addresses are displayed in RMCTools, such as in the **Reg #** column in Axis Tools, Indirect Data Map, and the Variable Table. To view those addresses in various formats, in the **Reg#** column, right-click the address and choose Address Formats.
2. Use the Address Maps in RMCTools to browse all the register addresses for any addressing method.
3. Use the Register Map help topic to browse all the register addresses for any addressing method.

FINS Address Sections

Listed below are the starting FINS addresses of each major register section. To obtain the FINS address of a specific RMC75 or RMC150 register, use the Register Maps or the **Address Calculation Utility** section below. The register maps provide the FINS address for each register. In addition, RMCTools itself displays the FINS address for many items such as the Axis Status Registers, Axis Parameters, and Variables. In RMCTools, to choose the FINS format, right-click a cell in the **Reg #** column, and choose **Omron**.

RMC75

Description	RMC75 File	RMC75 Starting FINS Address
Indirect Data Map	18	D00000
Indirect Data Map Definition	17	D00512
Command Area	25	D01024
Variables - Current Values	56-59	D01536
Controller Info	7	D03584
Axis Status Registers	8-11	D04096
Axis Parameters	12-15	D06144
Command Area - Small	16	D08192
Indirect Data Map Definition (duplicate area)	17	D08704
Indirect Data Map (duplicate area)	18	D09216
Axis Definitions	19	D09728
Controller Status/Parameters	20	D10240
Communication Configuration	21	D10752
Event Log Configuration	22	D11264
Discrete I/O	23	D11776
Task Status/Configuration	24	D12288
Command Area (duplicate area)	25	D12800
Analog Inputs	26	D13312
Axis Names	27	D13824
Image Upload/Download Area	30	D15360

Plot Layout	31	D15872
Plot Status/Configuration	32-39	D16384
Dynamic Plot Upload Area	40-47	D20480
Static Plot Upload Area	48-55	D24576
Variables - Current Values (duplicate area)	56-59	D28672
Variables - Initial Values	64-67	E0_00000
Variables - Attributes	68	E0_02048

RMC150

Description	RMC150 File	RMC150 Starting FINS Address
Indirect Data Map	42	D00000
Indirect Data Map Definition	41	D00512
Command Area	40	D01024
Variables - Current Values	56-59	D01536
Controller Info	7	D03584
Axis Status Registers	8-23	D04096
Axis Parameters	24-39	D12288
Command Area (duplicate area)	40	D20480
Indirect Data Map Definition (duplicate area)	41	D20992
Indirect Data Map (duplicate area)	42	D21504
Axis Definitions	43	D22016
Controller Parameters/Status	44	D22528
Communication Configuration	45	D23040
Event Log Configuration	46	D23552
Discrete I/O	47	D24064
Task Status/Configuration	48	D24576
Axis Names	49	D25088
Variables - Current Values (duplicate area)	56-59	D28672
Variables - Initial Values	72-75	E0_04096
Variables - Attributes	88	E0_12288
Image Upload/Download Area	94	E0_15360
Plot Layout	95	E0_15872
Plot Status/Configuration	96-103	E0_16384
Dynamic Plot Upload Area	104-111	E0_20480
Static Plot Upload Area	112-143	E0_24576
Slot Settings	144-149	E1_08192

RMC200

The RMC200 FINS addressing includes the pre-defined addresses as listed below. The remaining FINS addresses may be configured in the Address Maps.

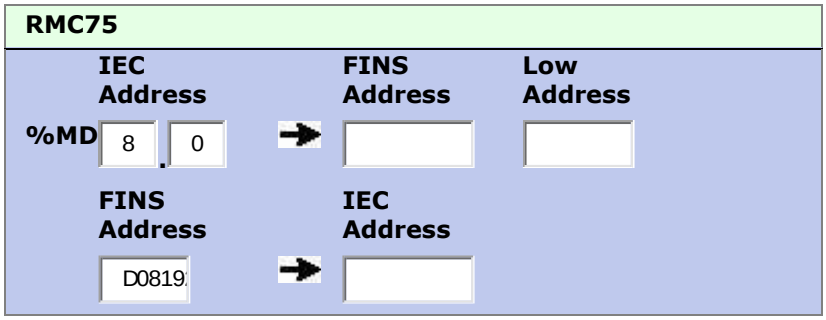
Description	RMC200 IEC Address	RMC200

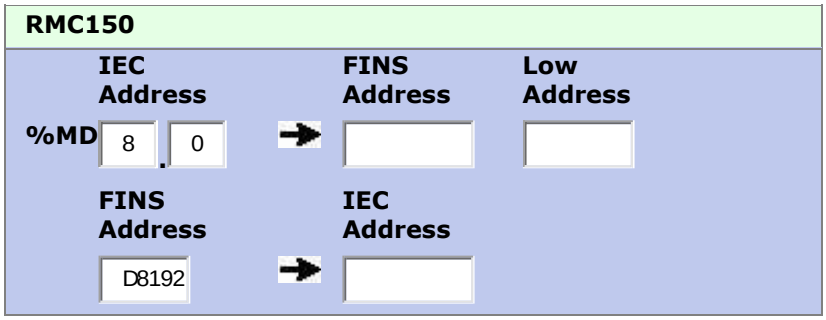
		FINS Address
Indirect Data Map	%MD8.0-1023	D00000-D02046
Command Area, axes 0-*n*[1]	%MD16.0-1019	D02048-D04086
Axis 0 Command Area	%MD16.0-9	D02048-D02066
Axis 1 Command Area	%MD16.10-19	D02068-D02086
Axis 2 Command Area	%MD16.20-29	D02088-D02106
...	...	D02048 + (20 x Axis Number)...
Variables - Current Values	%MD1024.0-4095	D04096-D12286
Image Upload/Download Area	%MD23.0-1023	D12288-D14334

[1]The maximum axis number *n* is 101 for the CPU40 and 47 for the CPU20L.

Address Calculation Utility (RMC75 and RMC150)

For the RMC75 and RMC150, you can use the utility or calculation below to determine the FINS address, given the IEC address.





Manual Calculation (RMC75 and RMC150)

The FINS address is calculated according to the following equation:

FINS Address = D*nnnnn* or E*nnnnn*-32768

nnnnn = 2 x [(256 x *file*) + *element*]

where the RMC IEC addressing format is %MD*file.element*

if *nnnnn* < 32768, then the prefix D is used with *nnnnn*.
if *nnnnn* >= 32768, then the prefix E is used with *nnnnn* - 32768

Examples:

%MD12:3 = 2 x [(256 x 12) + 3] = 6150. Therefore, the address is D06151.
%MD9:56 = 2 x [(256 x 9) + 56] = 4720. Therefore, the address is D04721.

%MD64:5 = 2 x [(256 x 64) + 5] = 32778. 32778-32768 = 10. Therefore, the address is E00010.

See Also

Register Map Overview | FINS Protocol | Using Omron Controllers via FINS

9.3.5. Modbus Addressing

This topic describes the Modbus addressing format as used by the RMC. For other addressing formats, see the Register Map Overview topic.

The Modbus/RTU and Modbus/TCP address format must be used when communicating with the RMC75S via Modbus/RTU and when communicating with the RMC75E or RMC150E via Modbus/TCP.

Address Format

The Modbus addressing assigns a single number to each register.

There are three different ways to refer to Modicon holding registers:

- **Holding Register Address** (e.g. 400059)
  Holding registers are labeled with the 4 prefix to differentiate them from other types of registers in the Modicon PLCs. Originally, they were called 40 thousand registers and were five-digit numbers starting with a 4. However, as PLC memory grew, an extra digit was added. So, 40059 became 400059.
- **Holding Register Offset** (e.g. 59)
  Holding Register Offset addresses are 1-based. This is basically the Holding Register Address stripped of the leading 4 and any leading zeros. Therefore, 400059 becomes 59. The RMC Modbus addresses are given in this format. You will need to prepend it with a 4 and any leading zeros.
- **Modbus Protocol Address** (e.g. 58)
  In the Modbus/RTU and Modbus/TCP protocols, the addresses are encoded using 16 bits with a number between 0 and 65,535. These are 0-based addresses. Therefore, the Modbus protocol address is equal to the Holding Register Offset minus one. This is mainly used internal to devices and is typically not seen by the end-user.

Finding Modbus Addresses for RMC Registers

When configuring the host controller (such as a PLC or HMI) to communicate with an RMC, you will need to enter the Modbus addresses of the RMC's registers that you wish to read from or write to.

Ways to find RMC addresses:

1. In RMCTools editors:
   Some register addresses are displayed in RMCTools, such as in the **Reg #** column in Axis Tools, Indirect Data Map, and the Variable Table. To view those addresses in various formats, in the **Reg#** column, right-click the address and choose Address Formats.
2. Use the Address Maps in RMCTools to browse all the register addresses for any addressing method.
3. Use the Register Map help topic to browse all the register addresses for any addressing method.

Modbus Address Sections

The Modbus address space for the RMC's is divided in sections as listed below. See each section for addresses of individual items within each section.

RMC75

Description	RMC75 IEC File	RMC75 Starting Modbus Address
Indirect Data Map	18	1
Indirect Data Map Definition	17	513
Command Area	25	1025
Variables - Current Values	56-59	1537
Controller Info	7	3585
Axis Status Registers	8-11	4097
Axis Parameters	12-15	6145
Command Area - Small	16	8193
Indirect Data Map Definition (duplicate area)	17	8705
Indirect Data Map (duplicate area)	18	9217
Axis Definitions	19	9729
Controller Parameters/Status	20	10241
Communication Configuration	21	10753
Event Log Configuration	22	11265
Discrete I/O	23	11777
Task Status/Configuration	24	12289
Command Area (duplicate area)	25	12801
Analog Inputs	26	13313
Axis Names	27	13825
Image Upload/Download Area	30	15361
Plot Layout	31	15873
Plot Status/Configuration	32-39	16385
Dynamic Plot Upload Area	40-47	20481
Static Plot Upload Area	48-55	24577
Variables - Current Values (duplicate area)	56-59	28673
Variables - Initial Values	64-67	32769
Variables - Attributes	68	34817

RMC150

Description	RMC150 IEC File	RMC150 Starting Modbus Address
Indirect Data Map	42	1
Indirect Data Map Definition	41	513
Command Area	40	1025
Variables - Current Values	56-59	1537
Controller Info	7	3585
Axis Status Registers	8-23	4097
Axis Parameters	24-39	12289

Command Area (duplicate area)	40	20481
Indirect Data Map Definition (duplicate area)	41	20993
Indirect Data Map (duplicate area)	42	21505
Axis Definitions	43	22017
Controller Parameters/Status	44	22529
Communication Configuration	45	23041
Event Log Configuration	46	23553
Discrete I/O	47	24065
Task Status/Configuration	48	24577
Axis Names	49	25089
Variables - Current Values (duplicate area)	56-59	28673
Variables - Initial Values	72-75	36865
Variables - Attributes	88	45057
Image Upload/Download Area	94	48129
Plot Layout	95	48641
Plot Status/Configuration	96-103	49153
Dynamic Plot Upload Area	104-111	53249
Static Plot Upload Area (the Modbus address space doesn't cover the entire range of 112-143)	112-127	57345

RMC200

The RMC200 Modbus addressing includes the pre-defined addresses as listed below. The remaining Modbus addresses may be configured in the Address Maps.

Description	RMC200 IEC Address	RMC200 Modbus Address
Indirect Data Map	%MD8.0-1023	1 - 2047
Command Area, axes 0-*n*[1]	%MD16.0-1019	2049 - 4087
Axis 0 Command Area	%MD16.0-9	2049 - 2067
Axis 1 Command Area	%MD16.10-19	2069 - 2087
Axis 2 Command Area	%MD16.20-29	2089 - 2107
...	...	2049 + (20 x Axis Number)...
Variables - Current Values	%MD1024.0-4095	4097 - 12287
Image Upload/Download Area	%MD23.0-1023	12289-14335

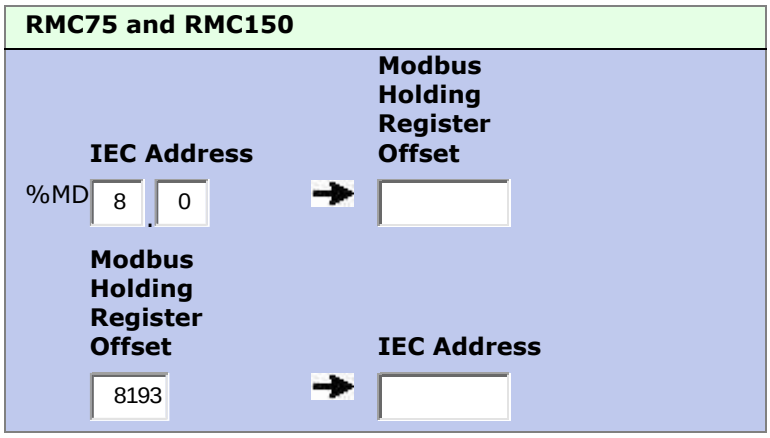
[1]The maximum axis number *n* is 101 for the CPU40 and 47 for the CPU20L.

Address Calculation Utility (RMC75 and RMC150)

For the RMC75 and RMC150, the standard Modbus addresses can be calculated from the IEC addresses as described here, with the exception of the first four items in the table.

To obtain the Modbus holding register offset address of RMC registers, use the Register Maps. The maps provide the Modbus offset for each register. The complete address is the holding register offset prepended with 4 and padded with zeros if necessary. For example, holding register offset 8193 will then become the address 408193 or 48193, depending on the host controller.

In addition, you can use the utility or calculation below to determine the Modbus holding register offset, given the IEC address.



Manual Calculation

The Modbus address is calculated according to the following equation:

Modbus RTU Address = 2 x [(256 x *file*) + *element*] + 1

where the RMC IEC addressing format is %MD*file.element*

Examples:

%MD12:3 = 2 x [(256 x 12) + 3] + 1 = 6151

%MD9:56 = 2 x [(256 x 9) + 56] + 1 = 4721

See Also

Register Map Overview

9.4. RMC75 Register Map

9.4.1. RMC75 Register Map

The RMC75 Register Map lists the addresses of all the registers in the RMC75. Typically, you will need to use the register map to find addresses when setting up communications with the RMC75 from a host controller such as a PLC. When referencing registers from *within* the RMC, such as in user programs, you do not need to use register addresses. You can use tag names instead. See the Tags Overview topic for details.

Tip: The Address Maps in RMCTools provide any easy way to browse all the registers in the RMC, along with their addresses.

For each register, the RMC75 register map provides addresses in the formats listed below. The address type you use will depend on the communication method you use. For more details on the addressing formats, see the respective topics.

- DF1 (Allen-Bradley) Addressing
- Modbus Addressing
- FINS (Omron) Addressing
- IEC-61131 Addressing

For PROFINET Record Data addressing, see PROFINET Data Records Addressing.

Follow the links below for the addresses of each section of registers.
The registers are divided into the following sections:

File	Description
7	Controller Info
8-11	Axis 0-3 Status Registers
12-15	Axis 0-3 Parameters
16	Command Area (Small)
17	Indirect Data Map Definition
18	Indirect Data Map
19	Axis Definitions
20	Controller Status/Parameters
21	Communication Configuration
22	Event Log Configuration
23	Discrete I/O
24	Tasks 0-3 Status/Configuration
25	Command Area
26	Analog Inputs
27	Axis Names
30	Image Upload/Download Area
31	Plot Layout
32-39	Plots 0-7 Status/Configuration
40-47	Dynamic Plot Upload Area
48-55	Static Plot Upload Area
56-59	Variables - Current Values
64-67	Variables - Initial Values
68	Variables - Attributes

See Also

Register Map Overview | RMC150 Register Map | RMC200 Register Map

Show All Back

RMC75 Registers, File 7: Controller Info

All Controller Information registers are **Read Only**.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
F7:0	3585	D03584	REAL	%MD7.0	DINT	Product ID

						1: RMC75
F7:1	3587	D03586	REAL	%MD7.1	DINT	State 1: Running Control Program 2: Running Control Program (in debugger) 3: Running Loader 4: Running Loader (FPGA not loaded)
F7:2	3589	D03588	REAL	%MD7.2	DINT	CPU Module ID 1: RMC75S 2: RMC75P 3: RMC75E
F7:3	3591	D03590	REAL	%MD7.3	DINT	CPU Module Rev Major * 256 + Minor
F7:4	3593	D03592	REAL	%MD7.4	DINT	Axis Module ID 0: None 32: MA1 33: MA2 34: QA1 35: QA2 36: AA1 37: AA2
F7:5	3595	D03594	REAL	%MD7.5	DINT	Axis Module Rev Major * 256 + Minor
F7:6	3597	D03596	REAL	%MD7.6	DINT	Expansion 1 Module ID 0: None 64: AP2 65: A2 66: D8 68: Q1
F7:7	3599	D03598	REAL	%MD7.7	DINT	Expansion 1 Module Rev Major * 256 + Minor
F7:8	3601	D03600	REAL	%MD7.8	DINT	Expansion 2 Module ID
F7:9	3603	D03602	REAL	%MD7.9	DINT	Expansion 2 Module Rev
F7:10	3605	D03604	REAL	%MD7.10	DINT	Expansion 3 Module ID
F7:11	3607	D03606	REAL	%MD7.11	DINT	Expansion 3 Module Rev
F7:12	3609	D03608	REAL	%MD7.12	DINT	Expansion 4 Module ID
F7:13	3611	D03610	REAL	%MD7.13	DINT	Expansion 4 Module Rev
F7:14	3613	D03612	DINT	%MD7.14	DINT	Serial Number 8-digit serial number (e.g. 71034039)
F7:15	3615	D03614	REAL	%MD7.15	DINT	Firmware Rev Major * 256 + Minor The patch release number is given in F7:32.

F7:16	3617	D03616	REAL	%MD7.16	DINT	Firmware Special Release Code 0: Standard 1-127: Special Release (S1-S127) 128: Beta Standard 129-255: Beta Special Release (S1-S127)
F7:17	3619	D03618	REAL	%MD7.17	DINT	Firmware Configuration ID 0: Discovery Image 1: A (RMC75S, revision 2.1C or older) 2: B (RMC75P, revision 2.1D or older) 3: C (RMC75E) 4: D (RMC75S, revision 2.1D or newer, and RMC75P, revision 2.1E or newer)
F7:18	3621	D03620	REAL	%MD7.18	DINT	Firmware Year and Month Year (4-digit) * 16 + Month (1=Jan, 2=Feb, ...)
F7:19	3623	D03622	REAL	%MD7.19	DINT	Firmware Day and Time Date (1-31) * 2048 + Hour (0-23) * 64 + Minute
F7:20	3625	D03624	REAL	%MD7.20	DINT	FPGA Rev Major * 256 + Minor
F7:21	3627	D03626	REAL	%MD7.21	DINT	FPGA Configuration ID 0: Discovery Image 1: A (RMC75S and RMC75P 0.1-1.x) 2: B (RMC75S and RMC75P 2.x)
F7:22	3629	D03628	REAL	%MD7.22	DINT	Flash Rev Major * 256 + Minor
F7:23	3631	D03630	REAL	%MD7.23	DINT	Required RMCTools Ver Major * 256 + Minor (patch ignored)
F7:24	3633	D03632	REAL	%MD7.24	DINT	Suggested RMCTools Ver Major * 256 + Minor (patch ignored)
F7:25	3635	D03634	REAL	%MD7.25	DINT	Loader Rev Major * 256 + Minor
F7:26	3637	D03636	REAL	%MD7.26	DINT	Loader Year and Month Year (4-digit) * 16 + Month (1=Jan, 2=Feb, ...)
F7:27	3639	D03638	REAL	%MD7.27	DINT	Loader Day and Time Date (1-31) * 2048 + Hour (0-23) * 64 + Minute
F7:28	3641	D03640	REAL	%MD7.28	DINT	Loader Command

						For sending certain commands to the loader, such as restart the RMC. See Loader Command for more details.
F7:29	3643	D03642	REAL	%MD7.29	DINT	Reason in Loader 0: Not in Loader 1: Explicitly Requested by Firmware (e.g. Firmware Update) 2: Bad FW Image Checksum 3: Invalidate FW Image Header 4: Base Module not supported by FW Image 5: Axis Module not supported by FW Image 6: Watchdog Timeout 7: Boot of FW Image Failed 8: FPGA Configuration Failed 9: Force-to-Loader Jumper is set 10: Unexpected External Reset
F7:30	3645	D03644	REAL	%MD7.30	DINT	Loader State Only used when updating. Confidential
F7:31	3647	D03646	REAL	%MD7.31	DINT	CPU Board Version This indicates the actual CPU board revision. Bits 0-7: Mod Level (0=A, 1=B, etc.) Bits 8-15: Minor Revision Bits 16-23: Major Revision Bits 24-31: Reserved For example, 2.1E will be 0x00020104.
F7:32	3649	D03648	REAL	%MD7.32	DINT	Firmware Patch Number Holds the patch level of the firmware version. For example, for 3.30.0, this value will be 0, and for 3.30.1, it will be 1.

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, Files 8-11: Axis Status Registers

All Axis Status Registers are **Read Only**.

Axis 0

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Common Registers: All Axes						
F8:0	4097	D04096	DWORD	%MD8.0	DWORD	Status Bits
F8:1	4099	D04098	DWORD	%MD8.1	DWORD	Error Bits
F8:2	4101	D04100	REAL	%MD8.2	DINT	Last Error Number
F8:4	4105	D04104	REAL	%MD8.4	REAL	Read Response
Position/Velocity Control						
F8:6	4109	D04108	REAL	%MD8.6	DINT	Current Control Mode
F8:7	4111	D04110	REAL	%MD8.7	DINT	Next Pos/Vel Control Mode
Primary Input: Position/Velocity Axes						
F8:8	4113	D04112	REAL	%MD8.8	REAL	Actual Position
F8:9	4115	D04114	REAL	%MD8.9	REAL	Actual Velocity
F8:10	4117	D04116	REAL	%MD8.10	REAL	Actual Acceleration
F8:11	4119	D04118	REAL	%MD8.11	REAL	Counts/Current/Voltage
F8:12	4121	D04120	DINT	%MD8.12	DINT	Raw Counts
Primary Input: Single-Input Pressure, Force, or Acceleration Axes						
F8:8	4113	D04112	REAL	%MD8.8	REAL	Actual Pressure/Force, Actual Acceleration
F8:9	4115	D04114	REAL	%MD8.9	REAL	Actual Pressure/Force Rate, Actual Jerk
F8:11	4119	D04118	REAL	%MD8.11	REAL	Current/Voltage
F8:12	4121	D04120	DINT	%MD8.12	DINT	Raw Counts
Primary Input: Dual-Input Force or Acceleration Axes						
F8:8	4113	D04112	REAL	%MD8.8	REAL	Actual Differential Force, Actual Acceleration
F8:9	4115	D04114	REAL	%MD8.9	REAL	Actual Differential Force Rate, Actual Jerk
F8:10	4117	D04116	REAL	%MD8.10	REAL	Actual Force A, Channel A Acceleration
F8:11	4119	D04118	REAL	%MD8.11	REAL	Voltage A/Current A
F8:12	4121	D04120	DINT	%MD8.12	DINT	Raw Counts A
F8:13	4123	D04122	REAL	%MD8.13	REAL	Actual Force B, Channel B Acceleration
F8:14	4125	D04124	REAL	%MD8.14	REAL	Voltage B/Current B
F8:15	4127	D04126	DINT	%MD8.15	DINT	Raw Counts B
Home/Registration: Quadrature Axes						
F8:18	4133	D04132	DWORD	%MD8.18	DWORD	Encoder Status Bits
F8:19	4135	D04134	REAL	%MD8.19	REAL	Registration 0 Position
F8:20	4137	D04136	REAL	%MD8.20	REAL	Registration 1 Position
Secondary Input: Single-Input Pressure, Force, or Acceleration Axes						

F8:23	4143	D04142	REAL	%MD8.23	REAL	Actual Pressure/Force, Actual Acceleration
F8:24	4145	D04144	REAL	%MD8.24	REAL	Actual Pressure/Force Rate, Actual Jerk
F8:26	4149	D04148	REAL	%MD8.26	REAL	Current/Voltage
F8:27	4151	D04150	DINT	%MD8.27	DINT	Raw Counts
Secondary Input: Dual-Input Force or Acceleration Axes						
F8:23	4143	D04142	REAL	%MD8.23	REAL	Actual Differential Force, Actual Acceleration
F8:24	4145	D04144	REAL	%MD8.24	REAL	Actual Differential Force Rate, Actual Jerk
F8:25	4147	D04146	REAL	%MD8.25	REAL	Actual Force A, Channel A Acceleration
F8:26	4149	D04148	REAL	%MD8.26	REAL	Voltage A/Current A
F8:27	4151	D04150	DINT	%MD8.27	DINT	Raw Counts A
F8:28	4153	D04152	REAL	%MD8.28	REAL	Actual Force B, Channel B Acceleration
F8:29	4155	D04154	REAL	%MD8.29	REAL	Voltage B/Current B
F8:30	4157	D04156	DINT	%MD8.30	DINT	Raw Counts B
Output: Analog Control Output Axes						
F8:33	4163	D04162	REAL	%MD8.33	REAL	Control Output
Primary Control: Position/Velocity Axes						
F8:35	4167	D04166	REAL	%MD8.35	REAL	Position Error
F8:36	4169	D04168	REAL	%MD8.36	REAL	Velocity Error
F8:37	4171	D04170	REAL	%MD8.37	REAL	Proportional Term
F8:38	4173	D04172	REAL	%MD8.38	REAL	Integral Term
F8:39	4175	D04174	REAL	%MD8.39	REAL	Differential Term
F8:40	4177	D04176	REAL	%MD8.40	REAL	Double Differential Output Term
F8:41	4179	D04177	REAL	%MD8.41	REAL	Velocity Feed Forward Term
F8:42	4181	D04180	REAL	%MD8.42	REAL	Acceleration Feed Forward Term
F8:43	4183	D04182	REAL	%MD8.43	REAL	Jerk Feed Forward Term
F8:44	4185	D04184	REAL	%MD8.44	REAL	Triple Differential Output Term
F8:45	4187	D04186	REAL	%MD8.45	REAL	PFID Output
F8:47	4191	D04190	REAL	%MD8.47	DINT	Current Integrator Mode
Primary Control: Pressure or Force Axes						
F8:35	4167	D04166	REAL	%MD8.35	REAL	Pressure/Force Error
F8:37	4171	D04170	REAL	%MD8.37	REAL	Pressure/Force Proportional Term
F8:38	4173	D04172	REAL	%MD8.38	REAL	Pressure/Force Integral Term
F8:39	4175	D04174	REAL	%MD8.39	REAL	Pressure/Force Differential Term
F8:40	4177	D04176	REAL	%MD8.40	REAL	Pressure/Force Feed Forward Term
F8:41	4179	D04178	REAL	%MD8.41	REAL	Pressure/Force Rate Feed Forward Term
F8:45	4187	D04186	REAL	%MD8.45	REAL	PFID Output

F8:47	4191	D04190	REAL	%MD8.47	DINT	Current Integrator Mode
Secondary Control: Pressure or Force Axes						
F8:46	4189	D04188	REAL	%MD8.46	REAL	Pressure/Force Error
F8:48	4193	D04192	REAL	%MD8.48	REAL	Pressure/Force Proportional Term
F8:49	4195	D04194	REAL	%MD8.49	REAL	Pressure/Force Integral Term
F8:50	4197	D04196	REAL	%MD8.50	REAL	Pressure/Force Differential Term
F8:51	4199	D04198	REAL	%MD8.51	REAL	Pressure/Force Feed Forward Term
F8:52	4201	D04200	REAL	%MD8.52	REAL	Pressure/Force Rate Feed Forward Term
Target						
F8:53	4203	D04202	REAL	%MD8.53	REAL	Target Position
F8:54	4205	D04204	REAL	%MD8.54	REAL	Target Velocity
F8:55	4207	D04206	REAL	%MD8.55	REAL	Target Acceleration
F8:56	4209	D04208	REAL	%MD8.56	REAL	Command Position
F8:57	4211	D04210	REAL	%MD8.57	REAL	Command Velocity
F8:58	4213	D04212	REAL	%MD8.58	REAL	Target Jerk
F8:59	4215	D04214	REAL	%MD8.59	DINT	Cycles
F8:60	4217	D04216	REAL	%MD8.60	REAL	Target Pressure/Force
F8:61	4219	D04218	REAL	%MD8.61	REAL	Command Pressure/Force
F8:62	4221	D04220	REAL	%MD8.62	DINT	Cycles (Pressure/Force)
Custom Feedback						
F89:64	4225 + *b*	D04224 + *b*	DWORD	%MD8.64	DWORD	Custom Error Bits
F8:65	4227 + *b*	D04226 + *b*	REAL	%MD8.65	REAL	Primary Custom Counts
F8:66	4229 + *b*	D04228 + *b*	REAL	%MD8.66	REAL	Secondary Custom Counts

Axis 1

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Common Registers: All Axes						
F9:0	4609	D04608	DWORD	%MD9.0	DWORD	Status Bits
F9:1	4611	D04610	DWORD	%MD9.1	DWORD	Error Bits
F9:2	4613	D04612	REAL	%MD9.2	DINT	Last Error Number
F9:4	4617	D04616	REAL	%MD9:4	REAL	Read Response
Position/Velocity Control						
F9:6	4621	D04620	REAL	%MD9.6	DINT	Current Control Mode
F9:7	4623	D04622	REAL	%MD9.7	DINT	Next Pos/Vel Control Mode
Primary Input: Position/Velocity Axes						
F9:8	4625	D04624	REAL	%MD9.8	REAL	Actual Position
F9:9	4627	D04626	REAL	%MD9.9	REAL	Actual Velocity

F9:10	4629	D04628	REAL	%MD9.10	REAL	Actual Acceleration
F9:11	4631	D04630	REAL	%MD9.11	REAL	Counts/Current/Voltage
F9:12	4633	D04632	DINT	%MD9.12	DINT	Raw Counts
Primary Input: Single-Input Pressure, Force, or Acceleration Axes						
F9:8	4625	D04624	REAL	%MD9.8	REAL	Actual Pressure/Force, Actual Acceleration
F9:9	4627	D04626	REAL	%MD9.9	REAL	Actual Pressure/Force Rate, Actual Jerk
F9:11	4631	D04630	REAL	%MD9.11	REAL	Current/Voltage
F9:12	4633	D04632	DINT	%MD9.12	DINT	Raw Counts
Primary Input: Dual-Input Force or Acceleration Axes						
F9:8	4625	D04624	REAL	%MD9.8	REAL	Actual Differential Force, Actual Acceleration
F9:9	4627	D04626	REAL	%MD9.9	REAL	Actual Differential Force Rate, Actual Jerk
F9:10	4629	D04628	REAL	%MD9.10	REAL	Actual Force A, Channel A Acceleration
F9:11	4631	D04630	REAL	%MD9.11	REAL	Voltage A/Current A
F9:12	4633	D04632	DINT	%MD9.12	DINT	Raw Counts A
F9:13	4635	D04634	REAL	%MD9.13	REAL	Actual Force B, Channel B Acceleration
F9:14	4637	D04636	REAL	%MD9.14	REAL	Voltage B/Current B
F9:15	4639	D04638	DINT	%MD9.15	DINT	Raw Counts B
Home/Registration: Quadrature Axes						
F9:18	4645	D04644	DWORD	%MD9.18	DWORD	Home/Registration Bits
F9:19	4647	D04646	REAL	%MD9.19	REAL	Registration 0 Position
F9:20	4649	D04648	REAL	%MD9.20	REAL	Registration 1 Position
Secondary Input: Single-Input Pressure, Force, or Acceleration Axes						
F9:23	4655	D04654	REAL	%MD9.23	REAL	Actual Pressure/Force, Actual Acceleration
F9:24	4657	D04656	REAL	%MD9.24	REAL	Actual Pressure/Force Rate, Actual Jerk
F9:26	4661	D04660	REAL	%MD9.26	REAL	Current/Voltage
F9:28	4663	D04662	DINT	%MD9.27	DINT	Raw Counts
Secondary Input: Dual-Input Force or Acceleration Axes						
F9:23	4655	D04654	REAL	%MD9.23	REAL	Actual Differential Force, Actual Acceleration
F9:24	4657	D04656	REAL	%MD9.24	REAL	Actual Differential Force Rate, Actual Jerk
F9:25	4659	D04658	REAL	%MD9.25	REAL	Actual Force A, Channel A Acceleration
F9:26	4661	D04660	REAL	%MD9.26	REAL	Voltage A/Current A
F9:27	4663	D04662	DINT	%MD9.27	DINT	Raw Counts A
F9:28	4665	D04664	REAL	%MD9.28	REAL	Actual Force B, Channel B Acceleration

F9:29	4667	D04666	REAL	%MD9.29	REAL	Voltage B/Current B
F9:30	4669	D04668	DINT	%MD9.30	DINT	Raw Counts B
Output: Analog Control Output Axes						
F9:33	4675	D04674	REAL	%MD9.33	REAL	Control Output
Primary Control: Position/Velocity Axes						
F9:35	4679	D04678	REAL	%MD9.35	REAL	Position Error
F9:36	4681	D04680	REAL	%MD9.36	REAL	Velocity Error
F9:37	4683	D04682	REAL	%MD9.37	REAL	Proportional Term
F9:38	4685	D04684	REAL	%MD9.38	REAL	Integral Term
F9:39	4687	D04686	REAL	%MD9.39	REAL	Differential Term
F9:40	4689	D04688	REAL	%MD9.40	REAL	Double Differential Output Term
F9:41	4691	D04690	REAL	%MD9.41	REAL	Velocity Feed Forward Term
F9:42	4693	D04692	REAL	%MD9.42	REAL	Acceleration Feed Forward Term
F9:43	4695	D04694	REAL	%MD9.43	REAL	Jerk Feed Forward Term
F9:44	4697	D04696	REAL	%MD9.44	REAL	Triple Differential Output Term
F9:45	4699	D04698	REAL	%MD9.45	REAL	PFID Output
F9:47	4703	D04702	REAL	%MD9.47	DINT	Current Integrator Mode
Primary Control: Pressure or Force Axes						
F9:35	4679	D04678	REAL	%MD9.35	REAL	Pressure/Force Error
F9:37	4683	D04682	REAL	%MD9.37	REAL	Pressure/Force Proportional Term
F9:38	4685	D04684	REAL	%MD9.38	REAL	Pressure/Force Integral Term
F9:39	4687	D04686	REAL	%MD9.39	REAL	Pressure/Force Differential Term
F9:40	4689	D04688	REAL	%MD9.40	REAL	Pressure/Force Feed Forward Term
F9:41	4691	D04690	REAL	%MD9.41	REAL	Pressure/Force Rate Feed Forward Term
F9:45	4699	D04698	REAL	%MD9.45	REAL	PFID Output
F9:47	4703	D04702	REAL	%MD9.47	DINT	Current Integrator Mode
Secondary Control: Pressure or Force Axes						
F9:46	4701	D04700	REAL	%MD9.46	REAL	Pressure/Force Error
F9:48	4705	D04704	REAL	%MD9.48	REAL	Pressure/Force Proportional Term
F9:49	4707	D04706	REAL	%MD9.49	REAL	Pressure/Force Integral Term
F9:50	4709	D04708	REAL	%MD9.50	REAL	Pressure/Force Differential Term
F9:51	4711	D04710	REAL	%MD9.51	REAL	Pressure/Force Feed Forward Term
F9:52	4713	D04712	REAL	%MD9.52	REAL	Pressure/Force Rate Feed Forward Term
Target						
F9:53	4715	D04714	REAL	%MD9.53	REAL	Target Position
F9:54	4717	D04716	REAL	%MD9.54	REAL	Target Velocity
F9:55	4719	D04718	REAL	%MD9.55	REAL	Target Acceleration
F9:56	4721	D04720	REAL	%MD9.56	REAL	Command Position
F9:57	4723	D04722	REAL	%MD9.57	REAL	Command Velocity
F9:58	4725	D04724	REAL	%MD9.58	REAL	Target Jerk

F9:59	4727	D04726	REAL	%MD9.59	DINT	Cycles
F9:60	4729	D04728	REAL	%MD9.60	REAL	Target Pressure/Force
F9:61	4731	D04730	REAL	%MD9.61	REAL	Command Pressure/Force
F9:62	4733	D04732	REAL	%MD9.62	DINT	Cycles (Pressure/Force)
Custom Feedback						
F9:64	4737 + *b*	D04736 + *b*	DWORD	%MD9.64	DWORD	Custom Error Bits
F9:65	4739 + *b*	D04738 + *b*	REAL	%MD9.65	REAL	Primary Custom Counts
F9:66	4741 + *b*	D04740 + *b*	REAL	%MD9.66	REAL	Secondary Custom Counts

Axis 2

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Common Registers: All Axes						
F10:0	5121	D05120	DWORD	%MD10.0	DWORD	Status Bits
F10:1	5123	D05122	DWORD	%MD10.1	DWORD	Error Bits
F10:2	5125	D05124	REAL	%MD10.2	DINT	Last Error Number
F10:4	5129	D05128	REAL	%MD10.4	REAL	Read Response
Position/Velocity Control						
F10:6	5133	D05132	REAL	%MD10.6	DINT	Current Control Mode
F10:7	5135	D05134	REAL	%MD10.7	DINT	Next Pos/Vel Control Mode
Primary Input: Position/Velocity Axes						
F10:8	5137	D05136	REAL	%MD10.8	REAL	Actual Position
F10:9	5139	D05138	REAL	%MD10.9	REAL	Actual Velocity
F10:10	5141	D05140	REAL	%MD10.10	REAL	Actual Acceleration
F10:11	5143	D05142	REAL	%MD10.11	REAL	Counts/Current/Voltage
F10:12	5145	D05144	DINT	%MD10.12	DINT	Raw Counts
Primary Input: Single-Input Pressure, Force, or Acceleration Axes						
F10:8	5137	D05136	REAL	%MD10.8	REAL	Actual Pressure/Force, Actual Acceleration
F10:9	5139	D05138	REAL	%MD10.9	REAL	Actual Pressure/Force Rate, Actual Jerk
F10:11	5143	D05142	REAL	%MD10.11	REAL	Current/Voltage
F10:12	5145	D05144	DINT	%MD10.12	DINT	Raw Counts
Primary Input: Dual-Input Force or Acceleration Axes						
F10:8	5137	D05136	REAL	%MD10.8	REAL	Actual Differential Force, Actual Acceleration
F10:9	5139	D05138	REAL	%MD10.9	REAL	Actual Differential Force Rate, Actual Jerk
F10:10	5141	D05140	REAL	%MD10.10	REAL	Actual Force A, Channel A Acceleration

F10:11	5143	D05142	REAL	%MD10.11	REAL	Voltage A/Current A
F10:12	5145	D05144	DINT	%MD10.12	DINT	Raw Counts A
F10:13	5147	D05146	REAL	%MD10.13	REAL	Actual Force B, Channel B Acceleration
F10:14	5149	D05148	REAL	%MD10.14	REAL	Voltage B/Current B
F10:15	5151	D05150	DINT	%MD10.15	DINT	Raw Counts B
Home/Registration: Quadrature Axes						
F10:18	5157	D05156	DWORD	%MD10.18	DWORD	Home/Registration Bits
F10:19	5159	D05158	REAL	%MD10.19	REAL	Registration 0 Position
F10:20	5161	D05160	REAL	%MD10.20	REAL	Registration 1 Position
Secondary Input: Single-Input Pressure, Force, or Acceleration Axes						
F10:23	5167	D05166	REAL	%MD10.23	REAL	Actual Pressure/Force, Actual Acceleration
F10:24	5169	D05168	REAL	%MD10.24	REAL	Actual Pressure/Force Rate, Actual Jerk
F10:26	5173	D05172	REAL	%MD10.26	REAL	Current/Voltage
F10:27	5175	D05174	DINT	%MD10.27	DINT	Raw Counts
Secondary Input: Dual-Input Force or Acceleration Axes						
F10:23	5167	D05166	REAL	%MD10.23	REAL	Actual Differential Force, Actual Acceleration
F10:24	5169	D05168	REAL	%MD10.24	REAL	Actual Differential Force Rate, Actual Jerk
F10:25	5171	D05170	REAL	%MD10.25	REAL	Actual Force A, Channel A Acceleration
F10:26	5173	D05172	REAL	%MD10.26	REAL	Voltage A/Current A
F10:27	5175	D05174	DINT	%MD10.27	DINT	Raw Counts A
F10:28	5177	D05176	REAL	%MD10.28	REAL	Actual Force B, Channel B Acceleration
F10:29	5179	D05178	REAL	%MD10.29	REAL	Voltage B/Current B
F10:30	5181	D05180	DINT	%MD10.30	DINT	Raw Counts B
Output: Analog Control Output Axes						
F10:33	5187	D05186	REAL	%MD10.33	REAL	Control Output
Primary Control: Position/Velocity Axes						
F10:35	5191	D05190	REAL	%MD10.35	REAL	Position Error
F10:36	5193	D05192	REAL	%MD10.36	REAL	Velocity Error
F10:37	5195	D05194	REAL	%MD10.37	REAL	Proportional Term
F10:38	5197	D05196	REAL	%MD10.38	REAL	Integral Term
F10:39	5199	D05198	REAL	%MD10.39	REAL	Differential Term
F10:40	5201	D05200	REAL	%MD10.40	REAL	Double Differential Output Term
F10:41	5203	D05202	REAL	%MD10.41	REAL	Velocity Feed Forward Term
F10:42	5205	D05204	REAL	%MD10.42	REAL	Acceleration Feed Forward Term
F10:43	5207	D05206	REAL	%MD10.43	REAL	Jerk Feed Forward Term
F10:44	5209	D05208	REAL	%MD10.44	REAL	Triple Differential Output Term

F10:45	5211	D05210	REAL	%MD10.45	REAL	PFID Output
F10:47	5215	D05214	REAL	%MD10.47	DINT	Current Integrator Mode
Primary Control: Pressure or Force Axes						
F10:35	5191	D05190	REAL	%MD10.35	REAL	Pressure/Force Error
F10:37	5195	D05194	REAL	%MD10.37	REAL	Pressure/Force Proportional Term
F10:38	5197	D05196	REAL	%MD10.38	REAL	Pressure/Force Integral Term
F10:39	5199	D05198	REAL	%MD10.39	REAL	Pressure/Force Differential Term
F10:40	5201	D05200	REAL	%MD10.40	REAL	Pressure/Force Feed Forward Term
F10:41	5203	D05202	REAL	%MD10.41	REAL	Pressure/Force Rate Feed Forward Term
F10:45	5211	D05210	REAL	%MD10.45	REAL	PFID Output
F10:47	5215	D05214	REAL	%MD10.47	DINT	Current Integrator Mode
Secondary Control: Pressure or Force Axes						
F10:46	5213	D05212	REAL	%MD10.46	REAL	Pressure/Force Error
F10:48	5217	D05216	REAL	%MD10.48	REAL	Pressure/Force Proportional Term
F10:49	5219	D05218	REAL	%MD10.49	REAL	Pressure/Force Integral Term
F10:50	5221	D05220	REAL	%MD10.50	REAL	Pressure/Force Differential Term
F10:51	5223	D05222	REAL	%MD10.51	REAL	Pressure/Force Feed Forward Term
F10:52	5225	D05224	REAL	%MD10.52	REAL	Pressure/Force Rate Feed Forward Term
Target						
F10:53	5227	D05226	REAL	%MD10.53	REAL	Target Position
F10:54	5229	D05228	REAL	%MD10.54	REAL	Target Velocity
F10:55	5231	D05230	REAL	%MD10.55	REAL	Target Acceleration
F10:56	5233	D05232	REAL	%MD10.56	REAL	Command Position
F10:57	5235	D05234	REAL	%MD10.57	REAL	Command Velocity
F10:58	5237	D05236	REAL	%MD10.58	REAL	Target Jerk
F10:59	5239	D05238	REAL	%MD10.59	DINT	Cycles
F10:60	5241	D05240	REAL	%MD10.60	REAL	Target Pressure/Force
F10:61	5243	D05242	REAL	%MD10.61	REAL	Command Pressure/Force
F10:62	5245	D05444	REAL	%MD10.62	DINT	Cycles (Pressure/Force)
Custom Feedback						
F10:64	5249 + *b*	D05448 + *b*	DWORD	%MD10.64	DWORD	Custom Error Bits
F10:65	5251 + *b*	D05450 + *b*	REAL	%MD10.65	REAL	Primary Custom Counts
F10:66	5253 + *b*	D05452 + *b*	REAL	%MD10.66	REAL	Secondary Custom Counts

Axis 3

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name

Common Registers: All Axes						
F11:0	5633	D05632	DWORD	%MD11.0	DWORD	Status Bits
F11:1	5635	D05634	DWORD	%MD11.1	DWORD	Error Bits
F11:2	5637	D05636	REAL	%MD11.2	DINT	Last Error Number
F11:4	5641	D05640	REAL	%MD11.4	REAL	Read Response
Position/Velocity Control						
F11:6	5645	D05644	REAL	%MD11.6	DINT	Current Control Mode
F11:7	5647	D05646	REAL	%MD11.7	DINT	Next Pos/Vel Control Mode
Primary Input: Position/Velocity Axes						
F11:8	5649	D05648	REAL	%MD11.8	REAL	Actual Position
F11:9	5651	D05650	REAL	%MD11.9	REAL	Actual Velocity
F11:10	5653	D05652	REAL	%MD11.10	REAL	Actual Acceleration
F11:11	5655	D05654	REAL	%MD11.11	REAL	Counts/Current/Voltage
F11:12	5657	D05656	DINT	%MD11.12	DINT	Raw Counts
Primary Input: Single-Input Pressure, Force, or Acceleration Axes						
F11:8	5649	D05648	REAL	%MD11.8	REAL	Actual Pressure/Force, Actual Acceleration
F11:9	5651	D05650	REAL	%MD11.9	REAL	Actual Pressure/Force Rate, Actual Jerk
F11:11	5655	D05654	REAL	%MD11.11	REAL	Current/Voltage
F11:12	5657	D05656	DINT	%MD11.12	DINT	Raw Counts
Primary Input: Dual-Input Force or Acceleration Axes						
F11:8	5649	D05648	REAL	%MD11.8	REAL	Actual Differential Force, Actual Acceleration
F11:9	5651	D05650	REAL	%MD11.9	REAL	Actual Differential Force Rate, Actual Jerk
F11:10	5653	D05652	REAL	%MD11.10	REAL	Actual Force A, Channel A Acceleration
F11:11	5655	D05654	REAL	%MD11.11	REAL	Voltage A/Current A
F11:12	5657	D05656	DINT	%MD11.12	DINT	Raw Counts A
F11:13	5659	D05658	REAL	%MD11.13	REAL	Actual Force B, Channel B Acceleration
F11:14	5661	D05660	REAL	%MD11.14	REAL	Voltage B/Current B
F11:15	5663	D05662	DINT	%MD11.15	DINT	Raw Counts B
Home/Registration: Quadrature Axes						
F11:18	5669	D05668	DWORD	%MD11.18	DWORD	Home/Registration Bits
F11:19	5671	D05670	REAL	%MD11.19	REAL	Registration 0 Position
F11:20	5673	D05672	REAL	%MD11.20	REAL	Registration 1 Position
Secondary Input: Single-Input Pressure, Force, or Acceleration Axes						
F11:23	5679	D05678	REAL	%MD11.23	REAL	Actual Pressure/Force, Actual Acceleration
F11:24	5681	D05680	REAL	%MD11.24	REAL	Actual Pressure/Force Rate, Actual Jerk
F11:26	5685	D05684	REAL	%MD11.26	REAL	Current/Voltage

F11:27	5687	D05686	DINT	%MD11.27	DINT	Raw Counts
Secondary Input: Dual-Input Force or Acceleration Axes						
F11:23	5679	D05678	REAL	%MD11.23	REAL	Actual Differential Force, Actual Acceleration
F11:24	5681	D05680	REAL	%MD11.24	REAL	Actual Differential Force Rate, Actual Jerk
F11:25	5683	D05682	REAL	%MD11.25	REAL	Actual Force A, Channel A Acceleration
F11:26	5685	D05684	REAL	%MD11.26	REAL	Voltage A/Current A
F11:27	5687	D05686	DINT	%MD11.27	DINT	Raw Counts A
F11:28	5689	D05688	REAL	%MD11.28	REAL	Actual Force B, Channel B Acceleration
F11:29	5691	D05690	REAL	%MD11.29	REAL	Voltage B/Current B
F11:30	5693	D05692	DINT	%MD11.30	DINT	Raw Counts B
Output: Analog Control Output Axes						
F11:33	5699	D05698	REAL	%MD11.33	REAL	Control Output
Primary Control: Position/Velocity Axes						
F11:35	5703	D05702	REAL	%MD11.35	REAL	Position Error
F11:36	5705	D05704	REAL	%MD11.36	REAL	Velocity Error
F11:37	5707	D05706	REAL	%MD11.37	REAL	Proportional Term
F11:38	5709	D05708	REAL	%MD11.38	REAL	Integral Term
F11:39	5711	D05710	REAL	%MD11.39	REAL	Differential Term
F11:40	5713	D05712	REAL	%MD11.40	REAL	Double Differential Output Term
F11:41	5715	D05714	REAL	%MD11.41	REAL	Velocity Feed Forward Term
F11:42	5717	D05716	REAL	%MD11.42	REAL	Acceleration Feed Forward Term
F11:43	5719	D05718	REAL	%MD11.43	REAL	Jerk Feed Forward Term
F11:44	5721	D05720	REAL	%MD11.44	REAL	Triple Differential Output Term
F11:45	5723	D05722	REAL	%MD11.45	REAL	PFID Output
F11:47	5727	D05726	REAL	%MD11.47	DINT	Current Integrator Mode
Primary Control: Pressure or Force Axes						
F11:35	5703	D05702	REAL	%MD11.35	REAL	Pressure/Force Error
F11:37	5707	D05706	REAL	%MD11.37	REAL	Pressure/Force Proportional Term
F11:38	5709	D05708	REAL	%MD11.38	REAL	Pressure/Force Integral Term
F11:39	5711	D05710	REAL	%MD11.39	REAL	Pressure/Force Differential Term
F11:40	5713	D05712	REAL	%MD11.40	REAL	Pressure/Force Feed Forward Term
F11:41	5715	D05714	REAL	%MD11.41	REAL	Pressure/Force Rate Feed Forward Term
F11:45	5723	D05722	REAL	%MD11.45	REAL	PFID Output
F11:47	5727	D05726	REAL	%MD11.47	DINT	Current Integrator Mode
Secondary Control: Pressure or Force Axes						
F11:46	5725	D05724	REAL	%MD11.46	REAL	Pressure/Force Error
F11:48	5729	D05728	REAL	%MD11.48	REAL	Pressure/Force Proportional Term
F11:49	5731	D05730	REAL	%MD11.49	REAL	Pressure/Force Integral Term

F11:50	5733	D05732	REAL	%MD11.50	REAL	Pressure/Force Differential Term
F11:51	5735	D05734	REAL	%MD11.51	REAL	Pressure/Force Feed Forward Term
F11:52	5737	D05736	REAL	%MD11.52	REAL	Pressure/Force Rate Feed Forward Term
Target						
F11:53	5739	D05738	REAL	%MD11.53	REAL	Target Position
F11:54	5741	D05740	REAL	%MD11.54	REAL	Target Velocity
F11:55	5743	D05742	REAL	%MD11.55	REAL	Target Acceleration
F11:56	5745	D05744	REAL	%MD11.56	REAL	Command Position
F11:57	5747	D05746	REAL	%MD11.57	REAL	Command Velocity
F11:58	5749	D05748	REAL	%MD11.58	REAL	Target Jerk
F11:59	5751	D05750	REAL	%MD11.59	DINT	Cycles
F11:60	5753	D05752	REAL	%MD11.60	REAL	Target Pressure/Force
F11:61	5755	D05754	REAL	%MD11.61	REAL	Command Pressure/Force
F11:62	5757	D05756	REAL	%MD11.62	DINT	Cycles (Pressure/Force)
Custom Feedback						
F11:64	5761 + *b*	D05760 + *b*	DWORD	%MD11.64	DWORD	Custom Error Bits
F11:65	5763 + *b*	D05762 + *b*	REAL	%MD11.65	REAL	Primary Custom Counts
F11:66	5765 + *b*	D05764 + *b*	REAL	%MD11.66	REAL	Secondary Custom Counts

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, Files 12-15: Axis Parameter Registers

All Axis Parameter registers are **Read/Write**.

Axis 0

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Primary Feedback: Position Axes						
F12:0	6145	D06144	REAL	%MD12.0	REAL	Position Scale
F12:1	6147	D06146	REAL	%MD12.1	REAL	Position Offset
F12:2	6149	D06148	REAL	%MD12.2	REAL	Actual Position Filter
F12:3	6151	D06150	REAL	%MD12.3	REAL	Actual Velocity Filter
F12:4	6153	D06152	REAL	%MD12.4	REAL	Actual Acceleration Filter
F12:5	6155	D06154	REAL	%MD12.5	REAL	Stop Threshold
F12:6	6157	D06156	REAL	%MD12.6	REAL	Noise Error Rate
F12:8	6161	D06160	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register

F12:9	6163	D06162	REAL	%MD12.9	DWORD	Primary Input Bits Register
Primary Feedback: Velocity Axes						
F12:0	6145	D06144	REAL	%MD12.0	REAL	Velocity Scale
F12:1	6147	D06146	REAL	%MD12.1	REAL	Velocity Offset
F12:2	6149	D06148	REAL	%MD12.2	REAL	Velocity Deadband
F12:3	6151	D06150	REAL	%MD12.3	REAL	Actual Velocity Filter
F12:4	6153	D06152	REAL	%MD12.4	REAL	Actual Acceleration Filter
F12:5	6155	D06154	REAL	%MD12.5	REAL	Stop Threshold
F12:6	6157	D01656	REAL	%MD12.6	REAL	Noise Error Rate
F12:8	6161	D06160	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register
Primary Feedback: Single-Input Pressure, Force or Acceleration						
F12:0	6145	D06144	REAL	%MD12.0	REAL	Pressure/Force Scale, Acceleration Scale
F12:1	6147	D06146	REAL	%MD12.1	REAL	Pressure/Force Offset, Acceleration Offset
F12:4	6153	D06152	REAL	%MD12.4	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F12:5	6155	D06154	REAL	%MD12.5	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F12:6	6157	D06156	REAL	%MD12.6	REAL	Noise Error Rate
F12:8	6161	D06160	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register
Primary Feedback: Dual-Input Force or Acceleration						
F12:0	6145	D06144	REAL	%MD12.0	REAL	Force A Scale, Channel A Acceleration Scale
F12:1	6147	D06146	REAL	%MD12.1	REAL	Force A Offset, Channel A Acceleration Offset
F12:2	6149	D06148	REAL	%MD12.2	REAL	Force B Scale, Channel B Acceleration Scale
F12:3	6151	D06150	REAL	%MD12.3	REAL	Force B Offset, Channel B Acceleration Offset
F12:4	6153	D06152	REAL	%MD12.4	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F12:5	6155	D06154	REAL	%MD12.5	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F12:6	6157	D06156	REAL	%MD12.6	REAL	Noise Error Rate
Primary Feedback: SSI/MDT Transducer						
F12:10	6165	D06164	REAL	%MD12.10	DWORD	SSI/MDT Config Register
F12:11	6167	D06166	REAL	%MD12.11	REAL	Count Offset
F12:12	6169	D06168	REAL	%MD12.12	REAL	Position Unwind
F12:13	6171	D06170	REAL	%MD12.13	DINT	Count Unwind
Primary Feedback: Analog Transducer						
F12:10	6165	D06164	REAL	%MD12.10	DWORD	Analog Config Register
Primary Feedback: Quadrature Transducer						

F12:10	6165	D06164	REAL	%MD12.10	DWORD	Quadrature Config Register
F12:12	6169	D06168	REAL	%MD12.12	REAL	Position Unwind
F12:13	6171	D06170	REAL	%MD12.13	DINT	Count Unwind
Secondary Feedback: Single-Input Pressure, Force, or Acceleration						
F12:18	6181	D06180	REAL	%MD12.18	REAL	Pressure/Force Scale, Acceleration Scale
F12:19	6183	D06182	REAL	%MD12.19	REAL	Pressure/Force Offset, Acceleration Offset
F12:22	6189	D06188	REAL	%MD12.22	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F12:23	6191	D06190	REAL	%MD12.23	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F12:24	6193	D06192	REAL	%MD12.24	REAL	Noise Error Rate
F12:26	6197	D06196	REAL	%MD12.26	DWORD	Custom Feedback Configuration Register
Secondary Feedback: Dual-Input Force or Acceleration						
F12:18	6181	D06180	REAL	%MD12.18	REAL	Force A Scale, Channel A Acceleration Scale
F12:19	6183	D06182	REAL	%MD12.19	REAL	Force A Offset, Channel A Acceleration Offset
F12:20	6185	D06184	REAL	%MD12.20	REAL	Force B Scale, Channel B Acceleration Scale
F12:21	6187	D06186	REAL	%MD12.21	REAL	Force B Offset, Channel B Acceleration Offset
F12:22	6189	D06188	REAL	%MD12.22	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F12:23	6191	D06190	REAL	%MD12.23	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F12:24	6193	D06192	REAL	%MD12.24	REAL	Noise Error Rate
Secondary Feedback: Analog Transducer						
F12:28	6201	D06200	REAL	%MD12.28	DWORD	Analog Config Register
Analog Control Output						
F12:32	6209	D06208	REAL	%MD12.32	REAL	Output Limit
F12:33	6211	D06210	REAL	%MD12.33	REAL	Output Bias
F12:34	6213	D06212	REAL	%MD12.34	DWORD	Output Register
Final Output Stage						
F12:38	6221	D06220	REAL	%MD12.38	REAL	Output Scale
F12:39	6223	D06222	REAL	%MD12.39	REAL	Primary Output Filter
F12:40	6225	D06224	REAL	%MD12.40	REAL	Secondary Output Filter
F12:41	6227	D06226	REAL	%MD12.41	REAL	Deadband Tolerance
F12:42	6229	D06228	REAL	%MD12.42	REAL	Output Deadband
F12:44	6233	D06232	REAL	%MD12.44	REAL	Knee Command Voltage
F12:45	6235	D06234	REAL	%MD12.45	REAL	Knee Flow Percentage
F12:46	6237	D06236	REAL	%MD12.46	DINT	Valve Linearization Curve ID
Primary Control: Servo Position Axes						

F12:43	6231	D06230	REAL	%MD12.43	DINT	Default Pos/Vel Control Mode
F12:56	6257	D06256	REAL	%MD12.56	REAL	In Position Tolerance
F12:57	6259	D06258	REAL	%MD12.57	REAL	Position Error Tolerance
F12:58	6261	D06260	REAL	%MD12.58	REAL	At Velocity Tolerance
F12:59	6263	D06262	REAL	%MD12.59	REAL	Velocity Error Tolerance
F12:60	6265	D06264	REAL	%MD12.60	DWORD	Primary Control Register
Position/Velocity Gain Set #0						
F12:61	6267	D06266	REAL	%MD12.61	REAL	Proportional Gain
F12:62	6269	D06268	REAL	%MD12.62	REAL	Integral Gain
F12:63	6271	D06270	REAL	%MD12.63	REAL	Differential Gain
F12:65	6275	D06274	REAL	%MD12.65	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)
F12:66	6277	D06276	REAL	%MD12.66	REAL	Acceleration Feed Forward
F12:67	6279	D06278	REAL	%MD12.67	REAL	Jerk Feed Forward
F12:68	6281	D06280	REAL	%MD12.68	REAL	Velocity Feed Forward (Negative)
F12:69	6283	D06282	REAL	%MD12.69	REAL	Double Differential Gain, Active Damping Proportional Gain
F12:70	6285	D06284	REAL	%MD12.70	REAL	Triple Differential Gain, Active Damping Differential Gain
F12:71	6287	D06286	REAL	%MD12.71	DINT	High-Order Control
Position/Velocity Gain Set #1						
F12:128	6401	D06400	REAL	%MD12.128	REAL	Proportional Gain
F12:129	6403	D06402	REAL	%MD12.129	REAL	Integral Gain
F12:130	6405	D06404	REAL	%MD12.130	REAL	Differential Gain
F12:132	6409	D06408	REAL	%MD12.132	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)
F12:133	6411	D06410	REAL	%MD12.133	REAL	Acceleration Feed Forward
F12:134	6413	D06412	REAL	%MD12.134	REAL	Jerk Feed Forward
F12:135	6415	D06414	REAL	%MD12.135	REAL	Velocity Feed Forward (Negative)
F12:136	6417	D06416	REAL	%MD12.136	REAL	Double Differential Gain, Active Damping Proportional Gain
F12:137	6419	D06418	REAL	%MD12.137	REAL	Triple Differential Gain, Active Damping Differential Gain
F12:138	6421	D06420	REAL	%MD12.138	DINT	High-Order Control
Primary Control: Servo Pressure or Force Axes						
F12:56	6257	D06256	REAL	%MD12.56	REAL	At Pressure/Force Tolerance
F12:57	6259	D06258	REAL	%MD12.57	REAL	Pressure/Force Error Tolerance
F12:60	6265	D06264	REAL	%MD12.60	DWORD	Primary Control Register
F12:61	6267	D06266	REAL	%MD12.61	REAL	Pressure/Force Proportional Gain
F12:62	6269	D06268	REAL	%MD12.62	REAL	Pressure/Force Integral Gain
F12:63	6271	D06270	REAL	%MD12.63	REAL	Pressure/Force Differential Gain
F12:64	6273	D06272	REAL	%MD12.65	REAL	Pressure/Force Feed Forward

F12:65	6275	D06274	REAL	%MD12.66	REAL	Pressure/Force Rate Feed Forward
Secondary Control: Servo Pressure or Force						
F12:76	6297	D06296	REAL	%MD12.76	REAL	At Pressure/Force Tolerance
F12:77	6299	D06268	REAL	%MD12.77	REAL	Pressure/Force Error Tolerance
F12:80	6305	D06304	REAL	%MD12.80	DWORD	Secondary Control Register
F12:81	6307	D06306	REAL	%MD12.81	REAL	Pressure/Force Proportional Gain
F12:82	6309	D06308	REAL	%MD12.82	REAL	Pressure/Force Integral Gain
F12:83	6311	D06310	REAL	%MD12.83	REAL	Pressure/Force Differential Gain
F12:84	6313	D06312	REAL	%MD12.84	REAL	Pressure/Force Feed Forward
F12:85	6315	D06314	REAL	%MD12.85	REAL	Pressure/Force Rate Feed Forward
Position Target						
F12:92	6329	D06328	REAL	%MD12.92	REAL	Positive Travel Limit
F12:93	6331	D06330	REAL	%MD12.93	REAL	Negative Travel Limit
F12:94	6333	D06332	REAL	%MD12.94	REAL	Requested Jerk
Pressure/Force Target						
F12:100	6345	D06344	REAL	%MD12.100	REAL	Positive Pressure/Force Limit
F12:101	6347	D06346	REAL	%MD12.101	REAL	Negative Pressure/Force Limit
Halts						
F12:106	6357	D06356	REAL	%MD12.106	DWORD	Auto Stops
F12:107	6359	D06358	REAL	%MD12.107	DWORD	Auto Stops
F12:108	6361	D06360	REAL	%MD12.108	DWORD	Auto Stops
F12:110	6365	D06364	REAL	%MD12.110	REAL	Closed Loop Halt Deceleration
F12:111	6367	D06366	REAL	%MD12.111	REAL	Open Loop Halt Ramp
F12:112	6369	D06368	REAL	%MD12.112	DINT	Halt Group Number
Simulator						
F12:116	6377	D06376	REAL	%MD12.116	DWORD	Simulator Configuration Register
F12:117	6379	D06378	REAL	%MD12.117	REAL	System Gain
F12:118	6381	D06380	REAL	%MD12.118	REAL	Natural Frequency
F12:119	6383	D06382	REAL	%MD12.119	REAL	Damping Factor
F12:120	6385	D06384	REAL	%MD12.120	REAL	Positive Physical Limit
F12:121	6387	D06386	REAL	%MD12.121	REAL	Negative Physical Limit
F12:122	6389	D06388	REAL	%MD12.122	REAL	Output Deadband
F12:123	6391	D06390	REAL	%MD12.123	REAL	Output Null
F12:125	6395	D06394	REAL	%MD12.125	REAL	Weight
F12:126	6397	D06396	REAL	%MD12.126	REAL	Maximum Force
F12:127	6399	D06398	REAL	%MD12.127	REAL	Maximum Compression
Position/Velocity Modeling						
F12:148	6441	D06440	REAL	%MD12.148	REAL	Model Response
F12:149	6443	D06442	REAL	%MD12.149	DINT	Model Order
F12:150	6445	D06444	REAL	%MD12.150	REAL	Model Gain Positive
F12:151	6447	D06446	REAL	%MD12.151	REAL	Model Gain Negative

F12:152	6449	D06448	REAL	%MD12.152	REAL	Model Time Constant, Model Natural Frequency
F12:153	6451	D06450	REAL	%MD12.153	REAL	Model Damping Factor
Pressure/Force Modeling						
F12:160	6465	D06464	REAL	%MD12.160	DINT	Model Order
F12:161	6467	D06466	REAL	%MD12.161	REAL	Model Gain Pressure/Force
F12:162	6469	D06468	REAL	%MD12.162	REAL	Model Time Constant, Model Natural Frequency
F12:163	6471	D06470	REAL	%MD12.163	REAL	Model Damping Factor
Display Units						
F12:166	6477	D06476	REAL	%MD12.166	DINT	Primary Display Units
F12:167	6479	D06478	DWORD	%MD12.167	DWORD	Primary Custom Units
F12:168	6481	D06480	REAL	%MD12.168	DINT	Secondary Display Units
F12:169	6483	D06482	DWORD	%MD12.169	DWORD	Secondary Custom Units

Axis 1

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Primary Feedback: Position Axes						
F13:0	6657	D06656	REAL	%MD13.0	REAL	Position Scale
F13:1	6659	D06658	REAL	%MD13.1	REAL	Position Offset
F13:2	6661	D06660	REAL	%MD13.2	REAL	Actual Position Filter
F13:3	6663	D06662	REAL	%MD13.3	REAL	Actual Velocity Filter
F13:4	6665	D06664	REAL	%MD13.4	REAL	Actual Acceleration Filter
F13:5	6667	D06666	REAL	%MD13.5	REAL	Stop Threshold
F13:6	6669	D06668	REAL	%MD13.6	REAL	Noise Error Rate
F13:8	6673	D06672	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register
F13:9	6675	D06674	REAL	%MD13.9	DWORD	Primary Input Bits Register
Primary Feedback: Velocity Axes						
F13:0	6657	D06656	REAL	%MD13.0	REAL	Velocity Scale
F13:1	6659	D06658	REAL	%MD13.1	REAL	Velocity Offset
F13:2	6661	D06660	REAL	%MD13.2	REAL	Velocity Deadband
F13:3	6663	D06662	REAL	%MD13.3	REAL	Actual Velocity Filter
F13:4	6665	D06664	REAL	%MD13.4	REAL	Actual Acceleration Filter
F13:5	6667	D06666	REAL	%MD13.5	REAL	Stop Threshold
F13:6	6669	D06668	REAL	%MD13.6	REAL	Noise Error Rate
F13:8	6673	D06672	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register
Primary Feedback: Single-Input Pressure, Force, or Acceleration Axes						
F13:0	6657	D06656	REAL	%MD13.0	REAL	Pressure/Force Scale, Acceleration Scale

F13:1	6659	D06658	REAL	%MD13.1	REAL	Pressure/Force Offset, Acceleration Offset
F13:4	6665	D06664	REAL	%MD13.4	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F13:5	6667	D06666	REAL	%MD13.5	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F13:6	6669	D06668	REAL	%MD13.6	REAL	Noise Error Rate
F13:8	6673	D06672	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register
Primary Feedback: Dual-Input Force or Acceleration Axes						
F13:0	6657	D06656	REAL	%MD13.0	REAL	Force A Scale, Channel A Acceleration Scale
F13:1	6659	D06658	REAL	%MD13.1	REAL	Force A Offset, Channel A Acceleration Offset
F13:2	6661	D06660	REAL	%MD13.2	REAL	Force B Scale, Channel B Acceleration Scale
F13:3	6663	D06662	REAL	%MD13.3	REAL	Force B Offset, Channel B Acceleration Offset
F13:4	6665	D06664	REAL	%MD13.4	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F13:5	6667	D06666	REAL	%MD13.5	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F13:6	6669	D06668	REAL	%MD13.6	REAL	Noise Error Rate
Primary Feedback: SSI/MDT Transducer						
F13:10	6677	D06676	REAL	%MD13.10	DWORD	SSI/MDT Config Register
F13:11	6679	D06678	REAL	%MD13.11	REAL	Count Offset
F13:12	6681	D06680	REAL	%MD13.12	REAL	Position Unwind
F13:13	6683	D06682	REAL	%MD13.13	DINT	Count Unwind
Primary Feedback: Analog Transducer						
F13:10	6677	D06676	REAL	%MD13.10	DWORD	Analog Config Register
Primary Feedback: Quadrature Transducer						
F13:10	6677	D06676	REAL	%MD13.10	DWORD	Quadrature Config Register
F13:12	6681	D06680	REAL	%MD13.12	REAL	Position Unwind
F13:13	6683	D06682	REAL	%MD13.13	DINT	Count Unwind
Secondary Feedback: Single-Input Pressure, Force, or Acceleration Axes						
F13:18	6693	D06692	REAL	%MD13.18	REAL	Pressure/Force Scale, Acceleration Scale
F13:19	6695	D06694	REAL	%MD13.19	REAL	Pressure/Force Offset,Acceleration Offset
F13:22	6701	D06700	REAL	%MD13.22	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F13:23	6703	D06702	REAL	%MD13.23	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F13:24	6705	D06704	REAL	%MD13.24	REAL	Noise Error Rate
F13:26	6709	D06708	REAL	%MD13.26	DWORD	Custom Feedback Configuration Register

Secondary Feedback: Dual-Input Force or Acceleration Axes						
F13:18	6693	D06692	REAL	%MD13.18	REAL	Force A Scale, Channel A Acceleration Scale
F13:19	6695	D06694	REAL	%MD13.19	REAL	Force A Offset, Channel A Acceleration Offset
F13:20	6697	D06696	REAL	%MD13.20	REAL	Force B Scale, Channel B Acceleration Scale
F13:21	6699	D06698	REAL	%MD13.21	REAL	Force B Offset, Channel B Acceleration Offset
F13:22	6701	D06700	REAL	%MD13.22	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F13:23	6703	D06702	REAL	%MD13.23	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F13:24	6705	D06704	REAL	%MD13.24	REAL	Noise Error Rate
Secondary Feedback: Analog Transducer						
F13:28	6713	D06712	DWORD	%MD13.28	DWORD	Analog Config Register
Analog Control Output						
F13:32	6721	D06720	REAL	%MD13.32	REAL	Output Limit
F13:33	6723	D06722	REAL	%MD13.33	REAL	Output Bias
F13:34	6725	D06724	DWORD	%MD13.34	DWORD	Output Register
Final Output Stage						
F13:38	6733	D06732	REAL	%MD13.38	REAL	Output Scale
F13:39	6735	D06734	REAL	%MD13.39	REAL	Primary Output Filter
F13:40	6737	D06736	REAL	%MD13.40	REAL	Secondary Output Filter
F13:41	6739	D06738	REAL	%MD13.41	REAL	Deadband Tolerance
F13:42	6741	D06740	REAL	%MD13.42	REAL	Output Deadband
F13:44	6745	D06744	REAL	%MD13.44	REAL	Knee Command Voltage
F13:45	6747	D06746	REAL	%MD13.45	REAL	Knee Flow Percentage
F13:46	6749	D06748	REAL	%MD13.46	DINT	Valve Linearization Curve ID
Primary Control: Servo Position/Velocity Axes						
F13:43	6743	D06742	REAL	%MD13.43	DINT	Default Pos/Vel Control Mode
F13:56	6769	D06768	REAL	%MD13.56	REAL	In Position Tolerance
F13:57	6771	D06770	REAL	%MD13.57	REAL	Position Error Tolerance
F13:58	6773	D06772	REAL	%MD13.58	REAL	At Velocity Tolerance
F13:59	6775	D06774	REAL	%MD13.59	REAL	Velocity Error Tolerance
F13:60	6777	D06776	DWORD	%MD13.60	DWORD	Primary Control Register
Position/Velocity Gain Set #1						
F13:61	6779	D06778	REAL	%MD13.61	REAL	Proportional Gain
F13:62	6781	D06780	REAL	%MD13.62	REAL	Integral Gain
F13:63	6783	D06782	REAL	%MD13.63	REAL	Differential Gain
F13:65	6787	D06786	REAL	%MD13.65	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)
F13:66	6789	D06788	REAL	%MD13.66	REAL	Acceleration Feed Forward

F13:67	6791	D06790	REAL	%MD13.67	REAL	Jerk Feed Forward
F13:68	6793	D06792	REAL	%MD13.68	REAL	Velocity Feed Forward (Negative)
F13:69	6795	D06794	REAL	%MD13.69	REAL	Double Differential Gain, Active Damping Proportional Gain
F13:70	6797	D06796	REAL	%MD13.70	REAL	Triple Differential Gain, Active Damping Differential Gain
F13:71	6799	D06798	DINT	%MD13.71	DINT	High-Order Control
Position/Velocity Gain Set #2						
F13:128	6913	D06912	REAL	%MD13.128	REAL	Proportional Gain
F13:129	6915	D06914	REAL	%MD13.129	REAL	Integral Gain
F13:130	6917	D06916	REAL	%MD13.130	REAL	Differential Gain
F13:132	6921	D06921	REAL	%MD13.132	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)
F13:133	6923	D06922	REAL	%MD13.133	REAL	Acceleration Feed Forward
F13:134	6925	D06924	REAL	%MD13.134	REAL	Jerk Feed Forward
F13:135	6927	D06926	REAL	%MD13.135	REAL	Velocity Feed Forward (Negative)
F13:136	6929	D06928	REAL	%MD13.136	REAL	Double Differential Gain, Active Damping Proportional Gain
F13:137	6931	D06930	REAL	%MD13.137	REAL	Triple Differential Gain, Active Damping Differential Gain
F13:138	6933	D06932	DINT	%MD13.138	DINT	High-Order Control
Primary Control: Servo Pressure or Force Axes						
F13:56	6769	D06768	REAL	%MD13.56	REAL	At Pressure/Force Tolerance
F13:57	6771	D06770	REAL	%MD13.57	REAL	Pressure/Force Error Tolerance
F13:60	6777	D06776	REAL	%MD13.60	DWORD	Primary Control Register
F13:61	6779	D06778	REAL	%MD13.61	REAL	Pressure/Force Proportional Gain
F13:62	6781	D06780	REAL	%MD13.62	REAL	Pressure/Force Integral Gain
F13:63	6783	D06782	REAL	%MD13.63	REAL	Pressure/Force Differential Gain
F13:64	6785	D06784	REAL	%MD13.65	REAL	Pressure/Force Feed Forward
F13:65	6787	D06786	REAL	%MD13.66	REAL	Pressure/Force Rate Feed Forward
Secondary Control: Servo Pressure or Force						
F13:76	6809	D06808	REAL	%MD13.76	REAL	At Pressure/Force Tolerance
F13:77	6811	D06810	REAL	%MD13.77	REAL	Pressure/Force Error Tolerance
F13:80	6817	D06816	REAL	%MD13.80	DWORD	Secondary Control Register
F13:81	6819	D06818	REAL	%MD13.81	REAL	Pressure/Force Proportional Gain
F13:82	6821	D06820	REAL	%MD13.82	REAL	Pressure/Force Integral Gain
F13:82	6823	D06822	REAL	%MD13.83	REAL	Pressure/Force Differential Gain
F13:84	6825	D06824	REAL	%MD13.84	REAL	Pressure/Force Feed Forward
F13:85	6827	D06826	REAL	%MD13.85	REAL	Pressure/Force Rate Feed Forward
Position Target						
F13:92	6841	D06840	REAL	%MD13.92	REAL	Positive Travel Limit
F13:93	6843	D06842	REAL	%MD13.93	REAL	Negative Travel Limit

F13:94	6845	D06844	REAL	%MD13.94	REAL	Requested Jerk
Pressure/Force Target						
F13:100	6857	D06856	REAL	%MD13.100	REAL	Positive Pressure/Force Limit
F13:101	6859	D06858	REAL	%MD13.101	REAL	Negative Pressure/Force Limit
Halts						
F13:106	6869	D06868	REAL	%MD13.106	DWORD	Auto Stops
F13:107	6871	D06870	REAL	%MD13.107	DWORD	Auto Stops
F13:108	6873	D06872	REAL	%MD13.108	DWORD	Auto Stops
F13:110	6877	D06876	REAL	%MD13.110	REAL	Closed Loop Halt Deceleration
F13:111	6879	D06878	REAL	%MD13.111	REAL	Open Loop Halt Ramp
F13:112	6881	D06880	REAL	%MD13.112	DINT	Halt Group Number
Simulator						
F13:116	6889	D06888	REAL	%MD13.116	DWORD	Simulator Configuration Register
F13:117	6891	D06890	REAL	%MD13.117	REAL	System Gain
F13:118	6893	D06892	REAL	%MD13.118	REAL	Natural Frequency
F13:119	6895	D06895	REAL	%MD13.119	REAL	Damping Factor
F13:120	6897	D06896	REAL	%MD13.120	REAL	Positive Physical Limit
F13:121	6899	D06898	REAL	%MD13.121	REAL	Negative Physical Limit
F13:122	6901	D06900	REAL	%MD13.122	REAL	Output Deadband
F13:123	6903	D06902	REAL	%MD13.123	REAL	Output Null
F13:125	6907	D06906	REAL	%MD13.125	REAL	Weight
F13:126	6909	D06908	REAL	%MD13.126	REAL	Maximum Force
F13:127	6911	D06910	REAL	%MD13.127	REAL	Maximum Compression
Position/Velocity Modeling						
F13:148	6953	D06952	REAL	%MD13.148	REAL	Model Response
F13:149	6955	D06954	REAL	%MD13.149	DINT	Model Order
F13:150	6957	D06956	REAL	%MD13.150	REAL	Model Gain Positive
F13:151	6959	D06958	REAL	%MD13.151	REAL	Model Gain Negative
F13:152	6961	D06960	REAL	%MD13.152	REAL	Model Time Constant, Model Natural Frequency
F13:153	6963	D06962	REAL	%MD13.153	REAL	Model Damping Factor
Pressure/Force Modeling						
F13:160	6977	D06976	REAL	%MD13.160	DINT	Model Order
F13:161	6979	D06978	REAL	%MD13.161	REAL	Model Gain Pressure/Force
F13:162	6981	D06980	REAL	%MD13.162	REAL	Model Time Constant, Model Natural Frequency
F13:163	6983	D06982	REAL	%MD13.163	REAL	Model Damping Factor
Display Units						
F13:166	6989	D06988	REAL	%MD13.166	DINT	Primary Display Units
F13:167	6991	D06990	DWORD	%MD13.167	DWORD	Primary Custom Units
F13:168	6993	D06992	REAL	%MD13.168	DINT	Secondary Display Units
F13:169	6995	D06994	DWORD	%MD13.169	DWORD	Secondary Custom Units

Axis 2

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Primary Feedback: Position Axes						
F14:0	7169	D07168	REAL	%MD14.0	REAL	Position Scale
F14:1	7171	D07170	REAL	%MD14.1	REAL	Position Offset
F14:2	7173	D07172	REAL	%MD14.2	REAL	Actual Position Filter
F14:3	7175	D07174	REAL	%MD14.3	REAL	Actual Velocity Filter
F14:4	7177	D07176	REAL	%MD14.4	REAL	Actual Acceleration Filter
F14:5	7179	D07178	REAL	%MD14.5	REAL	Stop Threshold
F14:6	7181	D07180	REAL	%MD14.6	REAL	Noise Error Rate
F12:8	7185	D07184	REAL	%MD12.89	DWORD	Custom Feedback Configuration Register
F14:9	7187	D07186	REAL	%MD14.9	DWORD	Primary Input Bits Register
Primary Feedback: Velocity Axes						
F14:0	7169	D07168	REAL	%MD14.0	REAL	Velocity Scale
F14:1	7171	D07170	REAL	%MD14.1	REAL	Velocity Offset
F14:2	7173	D07172	REAL	%MD14.2	REAL	Velocity Deadband
F14:3	7175	D07174	REAL	%MD14.3	REAL	Actual Velocity Filter
F14:4	7177	D07176	REAL	%MD14.4	REAL	Actual Acceleration Filter
F14:5	7179	D07178	REAL	%MD14.5	REAL	Stop Threshold
F14:6	7181	D07180	REAL	%MD14.6	REAL	Noise Error Rate
F12:8	7185	D07184	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register
Primary Feedback: Single-Input Pressure, Force or Acceleration Axes						
F14:0	7169	D07168	REAL	%MD14.0	REAL	Pressure/Force Scale, Acceleration Scale
F14:1	7171	D07170	REAL	%MD14.1	REAL	Pressure/Force Offset, Acceleration Offset
F14:4	7177	D07176	REAL	%MD14.4	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F14:5	7179	D07178	REAL	%MD14.5	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F14:6	7181	D07180	REAL	%MD14.6	REAL	Noise Error Rate
F12:8	7185	D07184	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register
Primary Feedback: Dual-Input Force or Acceleration Axes						
F14:0	7169	D07168	REAL	%MD14.0	REAL	Force A Scale, Channel A Acceleration Scale
F14:1	7171	D07170	REAL	%MD14.1	REAL	Force A Offset, Channel A Acceleration Offset
F14:2	7173	D07172	REAL	%MD14.2	REAL	Force B Scale, Channel B Acceleration Scale

F14:3	7175	D07174	REAL	%MD14.3	REAL	Force B Offset, Channel B Acceleration Offset
F14:4	7177	D07176	REAL	%MD14.4	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F14:5	7179	D07178	REAL	%MD14.5	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F14:6	7181	D07180	REAL	%MD14.6	REAL	Noise Error Rate
Primary Feedback: SSI/MDT Transducer						
F14:10	7189	D07188	REAL	%MD14.10	DWORD	SSI/MDT Config Register
F14:11	7191	D07190	REAL	%MD14.11	REAL	Count Offset
F14:12	7193	D07192	REAL	%MD14.12	REAL	Position Unwind
F14:13	7195	D07194	REAL	%MD14.13	DINT	Count Unwind
Primary Feedback: Analog Transducer						
F14:10	7189	D07188	REAL	%MD14.10	DWORD	Analog Config Register
Primary Feedback: Quadrature Transducer						
F14:10	7189	D07188	REAL	%MD14.10	DWORD	Quadrature Config Register
F14:12	7193	D07192	REAL	%MD14.12	REAL	Position Unwind
F14:13	7195	D07194	REAL	%MD14.13	DINT	Count Unwind
Secondary Feedback: Single-Input Pressure, Force, or Acceleration Axes						
F14:18	7205	D07204	REAL	%MD14.18	REAL	Pressure/Force Scale, Acceleration Scale
F14:19	7207	D07206	REAL	%MD14.19	REAL	Pressure/Force Offset, Acceleration Offset
F14:22	7213	D07212	REAL	%MD14.22	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F14:23	7215	D07214	REAL	%MD14.23	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F14:24	7217	D07216	REAL	%MD14.24	REAL	Noise Error Rate
F14:26	7221	D07220	REAL	%MD12.26	DWORD	Custom Feedback Configuration Register
Secondary Feedback: Dual-Input Force or Acceleration Axes						
F14:18	7205	D07204	REAL	%MD14.18	REAL	Force A Scale, Channel A Acceleration Scale
F14:19	7207	D07206	REAL	%MD14.19	REAL	Force A Offset, Channel A Acceleration Offset
F14:20	7209	D07208	REAL	%MD14.20	REAL	Force B Scale, Channel B Acceleration Scale
F14:21	7211	D07210	REAL	%MD14.21	REAL	Force B Offset, Channel B Acceleration Offset
F14:22	7213	D07212	REAL	%MD14.22	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F14:23	7215	D07214	REAL	%MD14.23	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F14:24	7217	D07216	REAL	%MD14.24	REAL	Noise Error Rate
Secondary Feedback: Analog Transducer						
F14:28	7225	D07224	REAL	%MD14.28	DWORD	Analog Config Register

Analog Control Output						
F14:32	7233	D07232	REAL	%MD14.32	REAL	Output Limit
F14:33	7235	D07234	REAL	%MD14.33	REAL	Output Bias
F14:34	7237	D07236	REAL	%MD14.34	DWORD	Output Register
Final Output Stage						
F14:38	7245	D07244	REAL	%MD14.38	REAL	Output Scale
F14:39	7247	D07246	REAL	%MD14.39	REAL	Primary Output Filter
F14:40	7249	D07248	REAL	%MD14.40	REAL	Secondary Output Filter
F14:41	7251	D07250	REAL	%MD14.41	REAL	Deadband Tolerance
F14:42	7253	D07252	REAL	%MD14.42	REAL	Output Deadband
F14:44	7257	D07256	REAL	%MD14.44	REAL	Knee Command Voltage
F14:45	7259	D07258	REAL	%MD14.45	REAL	Knee Flow Percentage
F14:46	7261	D07260	REAL	%MD14.46	DINT	Valve Linearization Curve ID
Primary Control: Servo Position Axes						
F14:43	7255	D07254	REAL	%MD14.43	DINT	Default Pos/Vel Control Mode
F14:56	7281	D07280	REAL	%MD14.56	REAL	In Position Tolerance
F14:57	7283	D07282	REAL	%MD14.57	REAL	Position Error Tolerance
F14:58	7285	D07284	REAL	%MD14.58	REAL	At Velocity Tolerance
F14:59	7287	D07286	REAL	%MD14.59	REAL	Velocity Error Tolerance
F14:60	7289	D07288	REAL	%MD14.60	DWORD	Primary Control Register
Position/Velocity Gain Set #1						
F14:61	7291	D07290	REAL	%MD14.61	REAL	Proportional Gain
F14:62	7293	D07292	REAL	%MD14.62	REAL	Integral Gain
F14:63	7295	D07294	REAL	%MD14.63	REAL	Differential Gain
F14:65	7299	D07298	REAL	%MD14.65	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)
F14:66	7301	D07300	REAL	%MD14.66	REAL	Acceleration Feed Forward
F14:67	7303	D07302	REAL	%MD14.67	REAL	Jerk Feed Forward
F14:68	7305	D07607	REAL	%MD14.68	REAL	Velocity Feed Forward (Negative)
F14:69	7307	D07309	REAL	%MD14.69	REAL	Double Differential Gain, Active Damping Proportional Gain
F14:70	7309	D07308	REAL	%MD14.70	REAL	Triple Differential Gain, Active Damping Differential Gain
F14:71	7311	D07310	REAL	%MD14.71	DINT	High-Order Control
Position/Velocity Gain Set #2						
F14:128	7425	D07424	REAL	%MD14.128	REAL	Proportional Gain
F14:129	7427	D07426	REAL	%MD14.129	REAL	Integral Gain
F14:130	7429	D07428	REAL	%MD14.130	REAL	Differential Gain
F14:132	7433	D07432	REAL	%MD14.132	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)
F14:133	7435	D07434	REAL	%MD14.133	REAL	Acceleration Feed Forward
F14:134	7437	D07436	REAL	%MD14.134	REAL	Jerk Feed Forward

F14:135	7439	D07438	REAL	%MD14.135	REAL	Velocity Feed Forward (Negative)
F14:136	7441	D07440	REAL	%MD14.136	REAL	Double Differential Gain, Active Damping Proportional Gain
F14:137	7443	D07442	REAL	%MD14.137	REAL	Triple Differential Gain, Active Damping Differential Gain
F14:138	7445	D07444	REAL	%MD14.138	DINT	High-Order Control
Primary Control: Servo Pressure or Force Axes						
F14:56	7281	D07280	REAL	%MD14.56	REAL	At Pressure/Force Tolerance
F14:57	7283	D07282	REAL	%MD14.57	REAL	Pressure/Force Error Tolerance
F14:60	7289	D07288	REAL	%MD14.60	DWORD	Primary Control Register
F14:61	7291	D07290	REAL	%MD14.61	REAL	Pressure/Force Proportional Gain
F14:62	7293	D07292	REAL	%MD14.62	REAL	Pressure/Force Integral Gain
F14:63	7295	D07294	REAL	%MD14.63	REAL	Pressure/Force Differential Gain
F14:64	7297	D07296	REAL	%MD14.65	REAL	Pressure/Force Feed Forward
F14:65	7299	D07298	REAL	%MD14.66	REAL	Pressure/Force Rate Feed Forward
Secondary Control: Servo Pressure or Force						
F14:76	7321	D07320	REAL	%MD14.76	REAL	At Pressure/Force Tolerance
F14:77	7323	D07322	REAL	%MD14.77	REAL	Pressure/Force Error Tolerance
F14:80	7329	D07328	REAL	%MD14.80	DWORD	Secondary Control Register
F14:81	7331	D07330	REAL	%MD14.81	REAL	Pressure/Force Proportional Gain
F14:82	7333	D07332	REAL	%MD14.82	REAL	Pressure/Force Integral Gain
F14:83	7335	D07334	REAL	%MD14.83	REAL	Pressure/Force Differential Gain
F14:84	7337	D07336	REAL	%MD14.84	REAL	Pressure/Force Feed Forward
F14:85	7339	D07338	REAL	%MD14.85	REAL	Pressure/Force Rate Feed Forward
Position Target						
F14:92	7353	D07352	REAL	%MD14.92	REAL	Positive Travel Limit
F14:93	7355	D07354	REAL	%MD14.93	REAL	Negative Travel Limit
F14:94	7357	D07356	REAL	%MD14.94	REAL	Requested Jerk
Pressure/Force Target						
F14:100	7369	D07368	REAL	%MD14.100	REAL	Positive Pressure/Force Limit
F14:101	7371	D07370	REAL	%MD14.101	REAL	Negative Pressure/Force Limit
Halts						
F14:106	7381	D07380	REAL	%MD14.106	DWORD	Auto Stops
F14:107	7383	D07382	REAL	%MD14.107	DWORD	Auto Stops
F14:108	7385	D07384	REAL	%MD14.108	DWORD	Auto Stops
F14:110	7389	D07388	REAL	%MD14.110	REAL	Closed Loop Halt Deceleration
F14:111	7391	D07390	REAL	%MD14.111	REAL	Open Loop Halt Ramp
F14:112	7393	D07392	REAL	%MD14.112	DINT	Halt Group Number
Simulator						
F14:116	7401	D07400	REAL	%MD14.116	DWORD	Simulator Configuration Register
F14:117	7403	D07402	REAL	%MD14.117	REAL	System Gain
F14:118	7405	D07404	REAL	%MD14.118	REAL	Natural Frequency

F14:119	7407	D07409	REAL	%MD14.119	REAL	Damping Factor
F14:120	7409	D07408	REAL	%MD14.120	REAL	Positive Physical Limit
F14:121	7411	D07410	REAL	%MD14.121	REAL	Negative Physical Limit
F14:122	7413	D07412	REAL	%MD14.122	REAL	Output Deadband
F14:123	7415	D07417	REAL	%MD14.123	REAL	Output Null
F14:125	7419	D07418	REAL	%MD14.125	REAL	Weight
F14:126	7421	D07420	REAL	%MD14.126	REAL	Maximum Force
F14:127	7423	D07422	REAL	%MD14.127	REAL	Maximum Compression
Position/Velocity Modeling						
F14:148	7465	D07464	REAL	%MD14.148	REAL	Model Response
F14:149	7467	D07466	REAL	%MD14.149	DINT	Model Order
F14:150	7469	D07468	REAL	%MD14.150	REAL	Model Gain Positive
F14:151	7471	D07470	REAL	%MD14.151	REAL	Model Gain Negative
F14:152	7473	D07472	REAL	%MD14.152	REAL	Model Time Constant, Model Natural Frequency
F14:153	7475	D07474	REAL	%MD14.153	REAL	Model Damping Factor
Pressure/Force Modeling						
F14:160	7489	D07488	REAL	%MD14.160	DINT	Model Order
F14:161	7491	D07490	REAL	%MD14.161	REAL	Model Gain Pressure/Force
F14:162	7493	D07492	REAL	%MD14.162	REAL	Model Time Constant, Model Natural Frequency
F14:163	7495	D07494	REAL	%MD14.163	REAL	Model Damping Factor
Display Units						
F14:166	7501	D07500	REAL	%MD14.166	DINT	Primary Display Units
F14:167	7503	D07502	DWORD	%MD14.167	DWORD	Primary Custom Units
F14:168	7505	D07504	REAL	%MD14.168	DINT	Secondary Display Units
F14:169	7507	D07506	DWORD	%MD14.169	DWORD	Secondary Custom Units

Axis 3

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Primary Feedback: Position Axes						
F15:0	7681	D07680	REAL	%MD15.0	REAL	Position Scale
F15:1	7683	D07682	REAL	%MD15.1	REAL	Position Offset
F15:2	7685	D07685	REAL	%MD15.2	REAL	Actual Position Filter
F15:3	7687	D07687	REAL	%MD15.3	REAL	Actual Velocity Filter
F15:4	7689	D07688	REAL	%MD15.4	REAL	Actual Acceleration Filter
F15:5	7691	D07690	REAL	%MD15.5	REAL	Stop Threshold
F15:6	7693	D07692	REAL	%MD15.6	REAL	Noise Error Rate
F12:8	7697	D07694	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register
F15:9	7699	D07698	REAL	%MD15.9	DWORD	Primary Input Bits Register

Primary Feedback: Velocity Axes						
F15:0	7681	D07680	REAL	%MD15.0	REAL	Velocity Scale
F15:1	7683	D07682	REAL	%MD15.1	REAL	Velocity Offset
F15:2	7685	D07684	REAL	%MD15.2	REAL	Velocity Deadband
F15:3	7687	D07686	REAL	%MD15.3	REAL	Actual Velocity Filter
F15:4	7689	D07688	REAL	%MD15.4	REAL	Actual Acceleration Filter
F15:5	7691	D07690	REAL	%MD15.5	REAL	Stop Threshold
F15:6	7693	D07692	REAL	%MD15.6	REAL	Noise Error Rate
F12:8	7697	D07698	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register
Primary Feedback: Single-Input Pressure, Force or Acceleration Axes						
F15:0	7681	D07680	REAL	%MD15.0	REAL	Pressure/Force Scale, Acceleration Scale
F15:1	7683	D07682	REAL	%MD15.1	REAL	Pressure/Force Offset, Acceleration Offset
F15:4	7689	D07688	REAL	%MD15.4	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F15:5	7691	D07690	REAL	%MD15.5	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F15:6	7693	D07692	REAL	%MD15.6	REAL	Noise Error Rate
F12:8	7697	D07696	REAL	%MD12.8	DWORD	Custom Feedback Configuration Register
Primary Feedback: Dual-Input Force or Acceleration Axes						
F15:0	7681	D07680	REAL	%MD15.0	REAL	Force A Scale, Channel A Acceleration Scale
F15:1	7683	D07682	REAL	%MD15.1	REAL	Force A Offset, Channel A Acceleration Offset
F15:2	7685	D07684	REAL	%MD15.2	REAL	Force B Scale, Channel B Acceleration Scale
F15:3	7687	D07686	REAL	%MD15.3	REAL	Force B Offset, Channel B Acceleration Offset
F15:4	7689	D07688	REAL	%MD15.4	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F15:5	7691	D07690	REAL	%MD15.5	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F15:6	7693	D07692	REAL	%MD15.6	REAL	Noise Error Rate
Primary Feedback: SSI/MDT Transducer						
F15:10	7701	D07700	REAL	%MD15.10	DWORD	SSI/MDT Config Register
F15:11	7703	D07702	REAL	%MD15.11	REAL	Count Offset
F15:12	7705	D07704	REAL	%MD15.12	REAL	Position Unwind
F15:13	7707	D07706	REAL	%MD15.13	DINT	Count Unwind
Primary Feedback: Analog Transducer						
F15:10	7701	D07700	REAL	%MD15.10	DWORD	Analog Config Register
Primary Feedback: Quadrature Transducer						
F15:10	7701	D07700	REAL	%MD15.10	DWORD	Quadrature Config Register

F15:12	7705	D07704	REAL	%MD15.12	REAL	Position Unwind
F15:13	7707	D07706	REAL	%MD15.13	DINT	Count Unwind
Secondary Feedback: Single-Input Pressure, Force, or Acceleration Axes						
F15:18	7717	D07716	REAL	%MD15.18	REAL	Pressure/Force Scale, Acceleration Scale
F15:19	7719	D07718	REAL	%MD15.19	REAL	Pressure/Force Offset, Acceleration Offset
F15:22	7725	D07724	REAL	%MD15.22	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F15:23	7727	D07726	REAL	%MD15.23	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F15:24	7729	D07728	REAL	%MD15.24	REAL	Noise Error Rate
F15:26	7733	D07732	REAL	%MD15.26	DWORD	Custom Feedback Configuration Register
Secondary Feedback: Dual-Input Force or Acceleration Axes						
F15:18	7717	D07716	REAL	%MD15.18	REAL	Force A Scale, Channel A Acceleration Scale
F15:19	7719	D07718	REAL	%MD15.19	REAL	Force A Offset, Channel A Acceleration Offset
F15:20	7721	D07720	REAL	%MD15.20	REAL	Force B Scale, Channel B Acceleration Scale
F15:21	7723	D07722	REAL	%MD15.21	REAL	Force B Offset, Channel B Acceleration Offset
F15:22	7725	D07724	REAL	%MD15.22	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F15:23	7727	D07726	REAL	%MD15.23	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F15:24	7729	D07728	REAL	%MD15.24	REAL	Noise Error Rate
Secondary Feedback: Analog Transducer						
F15:28	7737	D07726	REAL	%MD15.28	DWORD	Analog Config Register
Analog Control Output						
F15:32	7745	D07744	REAL	%MD15.32	REAL	Output Limit
F15:33	7747	D07746	REAL	%MD15.33	REAL	Output Bias
F15:34	7749	D07748	REAL	%MD15.34	DWORD	Output Register
Final Output Stage						
F15:38	7757	D07756	REAL	%MD15.38	REAL	Output Scale
F15:39	7759	D07758	REAL	%MD15.39	REAL	Primary Output Filter
F15:40	7761	D07760	REAL	%MD15.40	REAL	Secondary Output Filter
F15:41	7763	D07762	REAL	%MD15.41	REAL	Deadband Tolerance
F15:42	7765	D07764	REAL	%MD15.42	REAL	Output Deadband
F15:44	7769	D07768	REAL	%MD15.44	REAL	Knee Command Voltage
F15:45	7771	D07770	REAL	%MD15.45	REAL	Knee Flow Percentage
F15:46	7773	D07772	REAL	%MD15.46	DINT	Valve Linearization Curve ID
Primary Control: Servo Position Axes						
F15:43	7767	D07766	REAL	%MD15.43	DINT	Default Pos/Vel Control Mode

F15:56	7793	D07792	REAL	%MD15.56	REAL	In Position Tolerance
F15:57	7795	D07794	REAL	%MD15.57	REAL	Position Error Tolerance
F15:58	7797	D07796	REAL	%MD15.58	REAL	At Velocity Tolerance
F15:59	7799	D07798	REAL	%MD15.59	REAL	Velocity Error Tolerance
F15:60	7801	D07800	REAL	%MD15.60	DWORD	Primary Control Register
Position/Velocity Gain Set #1						
F15:61	7803	D07802	REAL	%MD15.61	REAL	Proportional Gain
F15:62	7805	D07804	REAL	%MD15.62	REAL	Integral Gain
F15:63	7807	D07806	REAL	%MD15.63	REAL	Differential Gain
F15:65	7811	D07810	REAL	%MD15.65	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)
F15:66	7813	D07812	REAL	%MD15.66	REAL	Acceleration Feed Forward
F15:64	7815	D07814	REAL	%MD15.67	REAL	Jerk Feed Forward
F15:68	7817	D07816	REAL	%MD15.68	REAL	Velocity Feed Forward (Negative)
F15:69	7819	D07818	REAL	%MD15.69	REAL	Double Differential Gain, Active Damping Proportional Gain
F15:70	7821	D07820	REAL	%MD15.70	REAL	Triple Differential Gain, Active Damping Differential Gain
F15:71	7823	D07822	REAL	%MD15.71	DINT	High-Order Control
Position/Velocity Gain Set #2						
F15:128	7937	D07936	REAL	%MD15.128	REAL	Proportional Gain
F15:129	7939	D07938	REAL	%MD15.129	REAL	Integral Gain
F15:130	7941	D07940	REAL	%MD15.130	REAL	Differential Gain
F15:132	7945	D07944	REAL	%MD15.132	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)
F15:133	7947	D07946	REAL	%MD15.133	REAL	Acceleration Feed Forward
F15:134	7949	D07948	REAL	%MD15.134	REAL	Jerk Feed Forward
F15:135	7951	D07950	REAL	%MD15.135	REAL	Velocity Feed Forward (Negative)
F15:136	7953	D07952	REAL	%MD15.136	REAL	Double Differential Gain, Active Damping Proportional Gain
F15:137	7955	D07954	REAL	%MD15.137	REAL	Triple Differential Gain, Active Damping Differential Gain
F15:138	7957	D07956	REAL	%MD15.138	DINT	High-Order Control
Primary Control: Servo Pressure or Force Axes						
F15:56	7793	D07792	REAL	%MD15.56	REAL	At Pressure/Force Tolerance
F15:57	7795	D07794	REAL	%MD15.57	REAL	Pressure/Force Error Tolerance
F15:60	7801	D07800	REAL	%MD15.60	DWORD	Primary Control Register
F15:61	7803	D07802	REAL	%MD15.61	REAL	Pressure/Force Proportional Gain
F15:62	7805	D07804	REAL	%MD15.62	REAL	Pressure/Force Integral Gain
F15:63	7807	D07806	REAL	%MD15.63	REAL	Pressure/Force Differential Gain
F15:64	7809	D07808	REAL	%MD15.65	REAL	Pressure/Force Feed Forward
F15:65	7811	D07810	REAL	%MD15.66	REAL	Pressure/Force Rate Feed Forward

Secondary Control: Servo Pressure or Force						
F15:76	7833	D07832	REAL	%MD15.76	REAL	At Pressure/Force Tolerance
F15:77	7835	D07834	REAL	%MD15.77	REAL	Pressure/Force Error Tolerance
F15:80	7841	D07840	REAL	%MD15.80	DWORD	Secondary Control Register
F15:81	7843	D07842	REAL	%MD15.81	REAL	Pressure/Force Proportional Gain
F15:82	7845	D07844	REAL	%MD15.82	REAL	Pressure/Force Integral Gain
F15:83	7847	D07846	REAL	%MD15.83	REAL	Pressure/Force Differential Gain
F15:84	7849	D07848	REAL	%MD15.84	REAL	Pressure/Force Feed Forward
F15:85	7851	D07850	REAL	%MD15.85	REAL	Pressure/Force Rate Feed Forward
Position Target						
F15:92	7865	D07864	REAL	%MD15.92	REAL	Positive Travel Limit
F15:93	7867	D07866	REAL	%MD15.93	REAL	Negative Travel Limit
F15:94	7869	D07868	REAL	%MD15.94	REAL	Requested Jerk
Pressure/Force Target						
F15:100	7881	D07880	REAL	%MD15.100	REAL	Positive Pressure/Force Limit
F15:101	7883	D07882	REAL	%MD15.101	REAL	Negative Pressure/Force Limit
Halts						
F15:106	7893	D07892	REAL	%MD15.106	DWORD	Auto Stops
F15:107	7895	D07894	REAL	%MD15.107	DWORD	Auto Stops
F15:108	7897	D07896	REAL	%MD15.108	DWORD	Auto Stops
F15:110	7901	D07900	REAL	%MD15.110	REAL	Closed Loop Halt Deceleration
F15:111	7903	D07902	REAL	%MD15.111	REAL	Open Loop Halt Ramp
F15:112	7905	D07904	REAL	%MD15.112	DINT	Halt Group Number
Simulator						
F15:116	7913	D07912	REAL	%MD15.116	DWORD	Simulator Configuration Register
F15:117	7915	D07914	REAL	%MD15.117	REAL	System Gain
F15:118	7917	D07916	REAL	%MD15.118	REAL	Natural Frequency
F15:119	7919	D07918	REAL	%MD15.119	REAL	Damping Factor
F15:120	7921	D07920	REAL	%MD15.120	REAL	Positive Physical Limit
F15:121	7923	D07922	REAL	%MD15.121	REAL	Negative Physical Limit
F15:122	7925	D07924	REAL	%MD15.122	REAL	Output Deadband
F15:123	7927	D07926	REAL	%MD15.123	REAL	Output Null
F15:125	7931	D07930	REAL	%MD15.125	REAL	Weight
F15:126	7933	D07932	REAL	%MD15.126	REAL	Maximum Force
F15:127	7935	D07934	REAL	%MD15.127	REAL	Maximum Compression
Position/Velocity Modeling						
F15:148	7977	D07976	REAL	%MD15.148	REAL	Model Response
F15:149	7979	D07978	REAL	%MD15.149	DINT	Model Order
F15:150	7981	D07980	REAL	%MD15.150	REAL	Model Gain Positive
F15:151	7983	D07982	REAL	%MD15.151	REAL	Model Gain Negative
F15:152	7985	D07984	REAL	%MD15.152	REAL	Model Time Constant, Model Natural Frequency

F15:153	7987	D07986	REAL	%MD15.153	REAL	Model Damping Factor
Pressure/Force Modeling						
F15:160	8001	D8000	REAL	%MD15.160	DINT	Model Order
F15:161	8003	D8002	REAL	%MD15.161	REAL	Model Gain Pressure/Force
F15:162	8005	D8004	REAL	%MD15.162	REAL	Model Time Constant, Model Natural Frequency
F15:163	8007	D8006	REAL	%MD15.163	REAL	Model Damping Factor
Display Units						
F15:166	8013	D8012	REAL	%MD15.166	DINT	Primary Display Units
F15:167	8015	D8014	DWORD	%MD15.167	DWORD	Primary Custom Units
F15:168	8017	D8016	REAL	%MD15.168	DINT	Secondary Display Units
F15:169	8019	D8018	DWORD	%MD15.169	DWORD	Secondary Custom Units

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, File 16: Command Area (Small)

All Command Area Registers are **Write Only**.

Note: The RMC75 command area registers were originally located only in file 16. However, to add more command parameters, the command area was changed to file 25. The file 16 command area is still available for backwards compatibility with earlier versions of RMC75 firmware that did not support the new larger command area (9 command parameters per command). The file 16 registers should not be used to issue commands that require more than 5 command parameters.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Axis 0 Command						
F16:0	8193	D8192	REAL	%MD16.0	REAL	Axis 0 Command
F16:1	8195	D8194	REAL	%MD16.1	REAL	Axis 0 Command Parameter 1
F16:2	8197	D8196	REAL	%MD16.2	REAL	Axis 0 Command Parameter 2
F16:3	8199	D8198	REAL	%MD16.3	REAL	Axis 0 Command Parameter 3
F16:4	8201	D8200	REAL	%MD16.4	REAL	Axis 0 Command Parameter 4
F16:5	8203	D8202	REAL	%MD16.5	REAL	Axis 0 Command Parameter 5
Axis 1 Command						
F16:6	8205	D8204	REAL	%MD16.6	REAL	Axis 1 Command
F16:7	8207	D8206	REAL	%MD16.7	REAL	Axis 1 Command Parameter 1
F16:8	8209	D8208	REAL	%MD16.8	REAL	Axis 1 Command Parameter 2
F16:9	8211	D8210	REAL	%MD16.9	REAL	Axis 1 Command Parameter 3
F16:10	8213	D8212	REAL	%MD16.10	REAL	Axis 1 Command Parameter 4
F16:11	8215	D8214	REAL	%MD16.11	REAL	Axis 1 Command Parameter 5

Axis 2 Command						
F16:12	8217	D8216	REAL	%MD16.12	REAL	Axis 2 Command
F16:13	8219	D8218	REAL	%MD16.13	REAL	Axis 2 Command Parameter 1
F16:14	8221	D8220	REAL	%MD16.14	REAL	Axis 2 Command Parameter 2
F16:15	8223	D8222	REAL	%MD16.15	REAL	Axis 2 Command Parameter 3
F16:16	8225	D8224	REAL	%MD16.16	REAL	Axis 2 Command Parameter 4
F16:17	8227	D8226	REAL	%MD16.17	REAL	Axis 2 Command Parameter 5
Axis 3 Command						
F16:18	8229	D8228	REAL	%MD16.18	REAL	Axis 3 Command
F16:19	8231	D8230	REAL	%MD16.19	REAL	Axis 3 Command Parameter 1
F16:20	8233	D8232	REAL	%MD16.20	REAL	Axis 3 Command Parameter 2
F16:21	8235	D8234	REAL	%MD16.21	REAL	Axis 3 Command Parameter 3
F16:22	8237	D8236	REAL	%MD16.22	REAL	Axis 3 Command Parameter 4
F16:23	8239	D8238	REAL	%MD16.23	REAL	Axis 3 Command Parameter 5

See Also

RMC75 Register Map Overview | RMC75 Register Map - File 25 Commands

Back

RMC75 Registers, Files 17-18: Indirect Data Map Registers

Use the Indirect Data Map Editor to edit the Indirect Data Map.

The **Indirect Data Map** registers (%MD18.0 - %MD18.31) are the registers that should be accessed at runtime by a PLC when reading or writing the Indirect Data. The **Indirect Data Map Definition** (registers %MD17.0 - %MD17.31) contains the *addresses* of the referenced registers, not the *data*. Reading or writing to these registers will not access the Indirect Data via a PLC.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types.

n = Data Item (0-63)

b = 2 x *n*

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
Indirect Data Map							
F18:*n*	9217 + *b*	D09216 + *b*	*	%MD18.*n*	*	*	Indirect Data Value *n*
Indirect Data Map Definition							
F17:*n*	8705 + *b*	D08704 + *b*	REAL	%MD17.*n*	DINT	Read/Write	Indirect Data Map Entry 0

*The Access and Data Types of the Indirect Data Values registers are determined by the mapped registers.

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, File 19: Axis Definitions

The Axis Definitions are not intended to be directly accessed by the user. The preferred method of changing the axis definitions is to use the Axis Definitions dialog.

For highly advanced users, see the Axis Definition Registers topic for more details.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
F19:0-15	9729-9759	D09728-D09758	DWORD	%MD19.0-15	DWORD	Current Axis Definitions (Read-Only)
F19:16-31	9761-9791	D09760-D09790	DWORD	%MD19.16-31	DWORD	Requested Axis Definitions

The Current Axis Definitions and the Requested Axis Definitions will generally be the same except in two cases:

(1) The user has written to the requested block and intends to do a warm restart or burn to flash and do a cold restart, or

(2) The requested axis definitions found on startup are invalid for the current hardware configuration; in this case the Current Axis Definitions will be the default for the current hardware configuration.

See Also

RMC75 Register Map

Back

RMC75 Registers, File 20: Controller Status/Parameters

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
F20:0	10241	D10240	REAL	%MD20.0	REAL	Read Only	Loop Time, Set (sec)
F20:1	10243	D10242	REAL	%MD20.1	REAL	none	Loop Time, Requested (sec)
F20:2	10245	D10244	REAL	%MD20.2	REAL	Read Only	Loop Time Used, Last (sec)
F20:3	10247	D10246	REAL	%MD20.3	REAL	Read/Write	Loop Time Used, Maximum (sec)
F20:4	10249	D10248	REAL	%MD20.4	DINT	Read/Write	Number of Tasks Allocated 0-4 (default = 2)
F20:5	10251	D10250	REAL	%MD20.5	DINT	Read/Write	Program Triggers Task Enabled?

							0=false, 1= true (default = 1)
F20:6	10253	D10252	REAL	%MD20.6	DWORD	Read/Write	Stop All Tasks on Any Axis Halt? 0=false, 1= true (default = 1)
F20:7	10255	D10254	REAL	%MD20.7	DWORD	Read Only	Controller Status Bits
F20:8	10257	D10256	REAL	%MD20.8	DINT	Read/Write	Startup Mode 0=PROGRAM, 1-RUN (default = 0)
F20:9	10259	D10258	REAL	%MD20.9	DINT	Read/Write	RUN/PROGRAM Input 0=none, 1-64=%IX(n-1), 256+*a*=Axis *a* Fault Input)
F20:10	10261	D10260	REAL	%MD20.10	REAL	Read Only	Time Gear Master
F20:11	10263	D10262	DINT	%MD20.11	DINT	Read Only	Time, 16th Milliseconds
F20:12	10265	D10264	DINT	%MD20.12	DINT	Read Only	Time, Seconds
F20:13	10267	D10266	DINT	%MD20.13	DINT	Read Only	Time, Milliseconds
F20:14	10269	D10268	DINT	%MD20.14	DINT	Read Only	Time, Microseconds
F20:15	10271	D10270	DINT	%MD20.15	DINT	Read Only	Time, Nanoseconds
F20:16-31	10273-10303	D10272-D10302	DWORD	%MD20.16-31	DWORD	Read/Write	Controller Name Each 32-bit register holds 2 UTF-16 characters. The MSW is the first and the LSW is the second. For example, 0x004D0061 is ‘Ma’.
F20:32	10305	D10304	DWORD	%MD20.32	DWORD	Read/Write	Controller Config Bits Bit 0: High Control Loop Utilization (1=enabled, 0=disabled) Bits 1-31: Reserved
F20:33	10307	D10306	DINT	%MD20.33	DINT	Read Only	Time, Loop Ticks
F20:34	10309	D10308	REAL	%MD20.34	REAL	Read Only	Loop Time Used, Total (µs)
F20:35	10311	D10310	REAL	%MD20.35	REAL	Read Only	Loop Time Used, Axes (µs)
F20:36	10313	D10312	REAL	%MD20.36	REAL	Read Only	Loop Time Used, Programs (µs)
F20:37	10315	D10314	REAL	%MD20.37	REAL	Read Only	Loop Time Used, Plots (µs)

F20:38	10317	D10316	REAL	%MD20.38	REAL	Read Only	Loop Time Used, Comm (µs)
F20:39	10319	D10318	REAL	%MD20.39	REAL	Read Only	Loop Time Used, Overhead (µs)

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, File 21: Communication Configuration

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
RMC75E							
F21:0-7	10753-10767	D10752 - D10766	-	%MD21.0-7	-	Read Only	Reserved
F21:8	10769	D10768	DWORD	%MD21.8	DWORD	Read Only	MAC Address (bytes 0-3) big Endian, (for 00-50-A0-A1-23-45, this would be 0x0050A0A1)
F21:9	10771	D10770	DWORD	%MD21.9	DWORD	Read Only	MAC Address (bytes 4-5) big Endian, in upper 16 bits (for 00-50-A0-A1-23-45, this would be 0x23450000)
F21:10	10773	D10772	DWORD	%MD21.10	DWORD	Read Only	Ethernet Status Bits
F21:11	10775	D10774	DWORD	%MD21.11	DWORD	Read/Write	Ethernet Configuration Bits
F21:12	10777	D10776	DWORD	%MD21.12	DWORD	Read/Write	IP Address
F21:13	10779	D10778	DWORD	%MD21.13	DWORD	Read/Write	Subnet Mask (0=use standard network masks)
F21:14	10781	D10780	DWORD	%MD21.14	DWORD	Read/Write	Default Gateway (0=none)
F21:15	10783	D10782	DWORD	%MD21.15	DWORD	Read Only	DHCP (or BOOTP) server IP Address
F21:16	10785	D10784	DINT	%MD21.16	DINT	Read Only	DHCP Lease Start (seconds since powerup)
F21:17	10787	D10786	DINT	%MD21.17	DINT	Read Only	DHCP Lease End (seconds since powerup)
F21:18	10789	D10788	DINT	%MD21.18	DINT	Read/Write	EtherNet/IP Class 1 TTL (time-to-live)
F21:19	10791	D10790	DINT	%MD21.19	DINT	Read/Write	EtherNet/IP Class 1 Start of Multicast Address Block

F21:20	10793	D10792	DINT	%MD21.20	DINT	Read/Write	EtherNet/IP Class 1 Size of Multicast Address Block
F21:21	10795	D10794	DWORD	%MD21.21	DWORD	Read/Write	Ethernet Outgoing Cyclic I/O Data Source
F21:22	10797	D10796	DWORD	%MD21.22	DWORD	Read/Write	Ethernet Incoming Cyclic I/O Data Destination
F21:23	10799	D10798	DWORD	%MD21.23	DWORD	Read Only	I/O Connection Status
F21:24	10801	D10800	DINT	%MD21.24	DINT	Read Only	I/O Connection PLC Status
F21:25	10803	D10802	DINT	%MD21.25	DINT	Read/Write	Ethernet App Flags **Bits 0-3 - I/O Data Mode** 0=Use a Sync Register 1=Do not use a Sync Register Applies to both EtherNet/IP and PROFINET. **Bits 4 - 5 - PROFINET Byte Order** 0=MSB first 1=LSB first Applies to PROFINET.
F21:26-55	10805-10863	D10804-D10862	-	%MD21.26-55	-	-	Reserved
F21:56-115	10865-10983	D10864-D10982	DWORD	%MD21.56-115	DWORD	Read/Write	PROFINET Device Name Each register in this range holds four characters of the PROFINET device name.
F21:116	10985	D10984	DWORD	%MD21.116	DWORD	Read/Write	PROFINET Custom Data Record 1000 Address
F21:117	10987	D10986	DWORD	%MD21.117	DWORD	Read/Write	PROFINET Custom Data Record 1001 Address
F21:118	10989	D10988	DWORD	%MD21.118	DWORD	Read/Write	PROFINET Custom Data Record 1002 Address
F21:119	10991	D10990	DWORD	%MD21.119	DWORD	Read/Write	PROFINET Custom Data Record 1003 Address
RMC75S							
F21:0	10753	D10752	REAL	%MD21.0	DINT	Read Only	RS-232 Monitor Baud Rate
F21:1	10755	D10754	REAL	%MD21.1	DWORD	Read Only	RS-232 Monitor Configuration Bits
F21:2	10757	D10756	REAL	%MD21.2	DINT	Read Only	RS-232 Monitor Protocol
F21:3	10759	D10758	REAL	%MD21.3	DINT	Read Only	RS-232 Monitor Address
F21:4	10761	D10760	REAL	%MD21.4	DINT	Read/Write	Comm. Serial Baud Rate

F21:5	10763	D10762	REAL	%MD21.5	DWORD	Read/Write	Comm. Serial Bits 0-1 Comm. Serial Data Bits 2 Comm. Serial Stop Bits 3-4 Comm. Serial Parity 5-6 Comm. Serial Line Driver 7 Comm. Serial Checksum Type 8-9 Comm. Serial Data Type 10 FlowControl 11 DF1: Allow Duplicate Packet Detection
F21:6	10765	D10764	REAL	%MD21.6	DINT	Read/Write	Comm. Serial Protocol
F21:7	10767	D10766	REAL	%MD21.7	DINT	Read/Write	Comm. Serial Address
RMC75P							
F21:0	10753	D10752	REAL	%MD21.0	DINT	Read Only	RS-232 Monitor Baud Rate
F21:1	10755	D10754	REAL	%MD21.1	DWORD	Read Only	RS-232 Monitor Configuration Bits
F21:2	10757	D10756	REAL	%MD21.2	DINT	Read Only	RS-232 Monitor Protocol
F21:3	10759	D10758	REAL	%MD21.3	DINT	Read Only	RS-232 Monitor Address
F21:4	10761	D10760	REAL	%MD21.4	DINT	Read Only	PROFIBUS: Station Address Indicates the current PROFIBUS station address, as selected by the rotary switches on the front of the RMC75P. Possible values are from 0-99, although zero (0) is not a valid station address.
F21:5	10763	D10762	-	%MD21.5	-	-	Reserved
F21:6	10765	D10764	DWORD	%MD21.6	DWORD	Read Only	PROFIBUS Connection Status

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, File 22: Event Log Configuration

This file is used by RMCTools to configure Event Log filtering. It should not be edited directly. Instead use the Event Log Properties to change these settings from within RMCTools.

See Also

RMC75 Register Map

Back

RMC75 Registers, File 23: Discrete I/O

All Discrete I/O registers are **Read/Write**.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Tag Name	Register Name
F23:0	11777	D11776	DWORD	%MD23.0	DWORD	_DIO.State[0]	Current state of discrete I/O - bits 0-31 The I/O points are numbered in order, with 0 indicating the first I/O point in the first D8 module. The Discrete I/O Configuration editor lists the I/O numbers as part of the address, for example %IX0 or %OX4.
F23:1	11779	D11778	DWORD	%MD23.1	DWORD		Reserved
F23:2	11781	D11780	DWORD	%MD23.2	DWORD	_DIO.Type[0]	I/O Type - bits 0-31
F23:3	11783	D11782	DWORD	%MD23.3	DWORD	_DIO.Type[1]	I/O Type - bits 32-64
F23:6	11789	D11788	DWORD	%MD23.6	DWORD	_DIO.OffInProgram[0]	Output to Off in PROGRAM mode - bits 0-31
F23:7	11791	D11790	DWORD	%MD23.7	DWORD	_DIO.OffInProgram[1]	Output to Off in PROGRAM mode - bits 32-64
F23:8	11793	D11792	DWORD	%MD23.8	DWORD	_DIO.OnInProgram[0]	Output to On in PROGRAM

							mode - bits 0-31
F23:9	11795	D11794	DWORD	%MD23.9	DWORD	_DIO.OnInProgram[1]	Output to On in PROGRAM mode - bits 32-64
F23:10	11797	D11796	DWORD	%MD23.10	DWORD	_DIO.OffInFault[0]	Output to Off in FAULT mode - bits 0-31
F23:11	11799	D11798	DWORD	%MD23.11	DWORD	_DIO.OffInFault[1]	Output to Off in FAULT mode - bits 32-64
F23:12	11801	D11800	DWORD	%MD23.12	DWORD	_DIO.OnInFault[0]	Output to On in FAULT mode - bits 0-31
F23:13	11803	D11802	DWORD	%MD23.13	DWORD	_DIO.OnInFault[1]	Output to On in FAULT mode - bits 32-64
F23:14	11805	D11804	DWORD	%MD23.14	DWORD	_DIO.ForcedOFF[0]	Forced Off - bits 0-31
F23:15	11807	D11806	DWORD	%MD23.15	DWORD	_DIO.ForcedOFF[1]	Forced Off - bits 32-64
F23:16	11809	D11808	DWORD	%MD23.16	DWORD	_DIO.ForcedON[0]	Forced On - bits 0-31
F23:17	11811	D11810	DWORD	%MD23.17	DWORD	_DIO.ForcedON[1]	Forced On - bits 32-64

%IX and %QX Addressing

In RMCTools, the discrete I/O addresses are shown in the IEC 61131-3 format:

Inputs = %IX.*n*

Outputs = %QX.*n*

where *n* is the I/O number as displayed in the Discrete I/O Monitor.

Examples:

%QX0 is output 0

%IX8 is input 8

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, File 24: Task Status/Configuration

The Task Status can be viewed in the Task Monitor.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
Task 0							
F24:0	12289		REAL	%MD24.0	DWORD	Read Only	Task 0 Task Status
F24:1	12291		REAL	%MD24.1	DWORD	Read Only	Task 0 Current Program/Step
F24:2	12293		REAL	%MD24.2	DINT	Read/Write	Task 0 Current Axis
F24:3	12295		REAL	%MD24.3	DINT	Read Only	Task 0 Current Program
F24:4	12297		REAL	%MD24.4	DINT	Read Only	Task 0 Current Step
F24:5-15	12299-12319		-	%MD24.5-15	-	-	Reserved
Task 1							
F24:16	12321		REAL	%MD24.16	DWORD	Read Only	Task 1 Task Status
F24:17	12323		REAL	%MD24.17	DWORD	Read Only	Task 1 Current Program/Step
F24:18	12325		REAL	%MD24.18	DINT	Read/Write	Task 1 Current Axis
F24:19	12327		REAL	%MD24.19	DINT	Read Only	Task 1 Current Program
F24:20	12329		REAL	%MD24.20	DINT	Read Only	Task 1 Current Step
F24:21-31	12331-12351		-	%MD24.21-31	-	-	Reserved
Task 2							
F24:32	12353		REAL	%MD24.32	DWORD	Read Only	Task 2 Task Status
F24:33	12355		REAL	%MD24.33	DWORD	Read Only	Task 2 Current Program/Step
F24:34	12357		REAL	%MD24.34	DINT	Read/Write	Task 2 Current Axis
F24:35	12359		REAL	%MD24.35	DINT	Read Only	Task 2 Current Program
F24:36	12361		REAL	%MD24.36	DINT	Read Only	Task 2 Current Step
F24:37-47	12363-12383		-	%MD24.37-47	-	-	Reserved
Task 3							
F24:48	12385		REAL	%MD24.48	DWORD	Read Only	Task 3 Task Status
F24:49	12387		REAL	%MD24.49	DWORD	Read Only	Task 3 Current Program/Step
F24:50	12389		REAL	%MD24.50	DINT	Read/Write	Task 3 Current Axis
F24:51	12391		REAL	%MD24.51	DINT	Read Only	Task 3 Current Program
F24:52	12393		REAL	%MD24.52	DINT	Read Only	Task 3 Current Step

F24:53-63	12395-12415		-	%MD24.53-63	-	-	Reserved

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, File 25: Command Area Registers

All Command Area Registers are **Write Only**.

Note: The RMC75 command area registers were originally located only in file 16. However, to add more command parameters, the command area was changed to file 25. The file 16 command area is still available for backwards compatibility with earlier versions of RMC75 firmware that did not support the new larger command area (9 command parameters per command).

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Axis 0 Command						
F25:0	12801	D12800	REAL	%MD25.0	REAL	Axis 0 Command
F25:1	12803	D12802	REAL	%MD25.1	REAL	Axis 0 Command Parameter 1
F25:2	12805	D12804	REAL	%MD25.2	REAL	Axis 0 Command Parameter 2
F25:3	12807	D12806	REAL	%MD25.3	REAL	Axis 0 Command Parameter 3
F25:4	12809	D12808	REAL	%MD25.4	REAL	Axis 0 Command Parameter 4
F25:5	12811	D12810	REAL	%MD25.5	REAL	Axis 0 Command Parameter 5
F25:6	12813	D12812	REAL	%MD25.6	REAL	Axis 0 Command Parameter 6
F25:7	12815	D12814	REAL	%MD25.7	REAL	Axis 0 Command Parameter 7
F25:8	12817	D12816	REAL	%MD25.8	REAL	Axis 0 Command Parameter 8
F25:9	12819	D12818	REAL	%MD25.9	REAL	Axis 0 Command Parameter 9
Axis 1 Command						
F25:10	12821	D12820	REAL	%MD25.10	REAL	Axis 1 Command
F25:11	12823	D12822	REAL	%MD25.11	REAL	Axis 1 Command Parameter 1
F25:12	12825	D12824	REAL	%MD25.12	REAL	Axis 1 Command Parameter 2
F25:13	12827	D12826	REAL	%MD25.13	REAL	Axis 1 Command Parameter 3
F25:14	12829	D12828	REAL	%MD25.14	REAL	Axis 1 Command Parameter 4
F25:15	12831	D12830	REAL	%MD25.15	REAL	Axis 1 Command Parameter 5
F25:16	12833	D12832	REAL	%MD25.16	REAL	Axis 1 Command Parameter 6
F25:17	12835	D12834	REAL	%MD25.17	REAL	Axis 1 Command Parameter 7
F25:18	12837	D12836	REAL	%MD25.18	REAL	Axis 1 Command Parameter 8
F25:19	12839	D12838	REAL	%MD25.19	REAL	Axis 1 Command Parameter 9
Axis 2 Command						
F25:20	12841	D12840	REAL	%MD25.20	REAL	Axis 2 Command
F25:21	12843	D12842	REAL	%MD25.21	REAL	Axis 2 Command Parameter 1

F25:22	12845	D12844	REAL	%MD25.22	REAL	Axis 2 Command Parameter 2
F25:23	12847	D12846	REAL	%MD25.23	REAL	Axis 2 Command Parameter 3
F25:24	12849	D12848	REAL	%MD25.24	REAL	Axis 2 Command Parameter 4
F25:25	12851	D12850	REAL	%MD25.25	REAL	Axis 2 Command Parameter 5
F25:26	12853	D12852	REAL	%MD25.26	REAL	Axis 2 Command Parameter 6
F25:27	12855	D12854	REAL	%MD25.27	REAL	Axis 2 Command Parameter 7
F25:28	12857	D12856	REAL	%MD25.28	REAL	Axis 2 Command Parameter 8
F25:29	12859	D12858	REAL	%MD25.29	REAL	Axis 2 Command Parameter 9
Axis 3 Command						
F25:30	12861	D12830	REAL	%MD25.30	REAL	Axis 3 Command
F25:31	12863	D12862	REAL	%MD25.31	REAL	Axis 3 Command Parameter 1
F25:32	12865	D12864	REAL	%MD25.32	REAL	Axis 3 Command Parameter 2
F25:33	12867	D12866	REAL	%MD25.33	REAL	Axis 3 Command Parameter 3
F25:34	12869	D12868	REAL	%MD25.34	REAL	Axis 3 Command Parameter 4
F25:35	12871	D12870	REAL	%MD25.35	REAL	Axis 3 Command Parameter 5
F25:36	12873	D12872	REAL	%MD25.36	REAL	Axis 3 Command Parameter 6
F25:37	12875	D12874	REAL	%MD25.37	REAL	Axis 3 Command Parameter 7
F25:38	12877	D12876	REAL	%MD25.38	REAL	Axis 3 Command Parameter 8
F25:39	12879	D12878	REAL	%MD25.39	REAL	Axis 3 Command Parameter 9

See Also

RMC75 Register Map Overview | RMC75 Register Map - File 16 Commands (Small)

Back

RMC75 Registers, File 26: Analog Inputs

The Analog Inputs register file holds the voltage or current of all the analog inputs on the controller. These Analog Input registers exist in order to display all the analog inputs, even those that are not assigned to axes, since the RMC75 can have more analog inputs than can be assigned to axes.

For analog inputs that are not assigned to axes, only the voltage is available. No status or error bits, filtering, scaling, input range selection, raw counts, halt groups, etc. are available. Current transducers can still be used by connecting the 4-20mA jumper, but the user will have make the conversion from voltage to current (1V = 4mA, 5V = 20mA, current in mA = (volts x 4)).

For analog inputs that are assigned to an axis, the value in the existing Current or Voltage axis status register will be copied into this file.

The analog inputs are ordered by slot (left-to-right), and top-to-bottom within each slot. The register values for inputs that do not exist will be zero.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
F26:0	13313	D13312	REAL	%MD26.0	REAL	Read Only	Analog Input 0
F26:1	13315	D13314	REAL	%MD26.1	REAL	Read Only	Analog Input 1

F26:2	13317	D13316	REAL	%MD26.2	REAL	Read Only	Analog Input 2
F26:3	13319	D13318	REAL	%MD26.3	REAL	Read Only	Analog Input 3
F26:4	13321	D13320	REAL	%MD26.4	REAL	Read Only	Analog Input 4
F26:5	13323	D13322	REAL	%MD26.5	REAL	Read Only	Analog Input 5
F26:6	13325	D13324	REAL	%MD26.6	REAL	Read Only	Analog Input 6
F26:7	13327	D13326	REAL	%MD26.7	REAL	Read Only	Analog Input 7
F26:8	13329	D13328	REAL	%MD26.8	REAL	Read Only	Analog Input 8
F26:9	13331	D13330	REAL	%MD26.9	REAL	Read Only	Analog Input 9

Usage Notes

Viewing the Analog Voltage

To view the Analog Input values in RMCTools, enter the address of the input in the **Map To** column in the Indirect Data Map. The analog input value will appear in the **Current** column of the Indirect Data Map.

Uses

These registers can be used for plotting, can be used in expressions, and can be read by an external device such as a PLC. They can also be used as a gear Master Register, just like any other REAL register, but notice that no filtering or advanced rate calculations will be provided. Also, because the input cannot be part of a halt group, the safety of using the input for gearing must be evaluated by the integrator.

Tag Names

The analog inputs can be referenced with the tag names _AI[*i*], where *i* is the input number. For example analog input 3 is _AI[3].

See Also

RMC75 Register Map Overview | RMC75 Register Map - File 16 Commands (Small)

Back

RMC75 Registers, File 30: Image Upload/Download Area

See the Controller Image Upload/Download topic for details.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name

F30:0	15361	D15360	DINT	%MD30.0	DINT	Write Only	**Image Area Command**
F30:1	15363	D15362	DINT	%MD30.1	DINT	Read Only	**Image Area State**
F30:2	15365	D15364	DINT	%MD30.2	DINT	Read Only	**Image Size**
F30:3	15367	D15366	DINT	%MD30.3	DINT	Read/Write	**Current Index**
F30:4 : F30:4095	15369- 15871 (252 items)	D15368- D15870 (252 items)	DINT[252] or DINT[4092]	%MD30.4 : %MD30.4095	DINT[252] or DINT[4092]	Read/Write	**Image Data**

See Also

RMC75 Register Map Overview | Controller Image Upload/Download

Back

RMC75 Registers, File 31: Plot Layout

The following files contain the Plot Layout registers.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading data of DWORD or DINT external data types.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
F31:0	15873	D15872	REAL	%MD31.0	UDINT	Read/Write	Current Plot Layout This register indicates the current plot allocation. **Bit** / **Description** 0-7 Plots (1-8) 8-15 Sample Sets per Plot (1-16) 16-23 Reserved
F31:1	15875	D15874	REAL	%MD31.1	UDINT	Read Only	Maximum Plots (8) This read-only value indicates how many plots can be defined at once.
F31:2	15877	D15876	REAL	%MD31.2	UDINT	Read Only	Maximum Samples (16) This read-only value indicates how many samples sets can be captured

							simultaneously per plot.
F31:3	15879	D15878	REAL	%MD31.3	UDINT	Read Only	Maximum Samples at Once (128) This read-only value indicates how many total different samples can be allocated across all plots.
F31:4	15881	D15880	REAL	%MD31.4	UDINT	Read Only	Maximum Elements RMC75S/P: 32,768 RMC75E: 12,583,912 **Note:** Prior to 2.00 firmware, the RMC75E is limited to 32,768 elements. Prior to 1.02 Loader, the RMC75E is limited to 4,194,304. This read-only value indicates how many total data points can be captured. This is spread out among the number of allocated plots and samples per plot.
F31:5	15883	D15882	-	%MD31.5	UDINT	-	Reserved
F31:6	15885	D15884	-	%MD31.6	UDINT	-	Reserved
F31:7	15887	D15886	-	%MD31.7	UDINT	-	Reserved

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, Files 32-39: Plot Status/Configuration Registers

The following files contain the Plot Configuration registers. See the Reading Plots with a Host Controller topic for details on how some of them can be used.

n = plot # (0-7)

f = 32 + *n*

b = 512 x *n*

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
Plot 0							

F*f*:0	16385 + *b*	D16384 + *b*	REAL	%MD*f*.0	DWORD	not directly	Plot Flags These bits should not be accessed directly. 0 Reserved (Write Only) 1 Trigger (Write only) 2 Rearm (Write Only) 3 Read Active (Read Only) 4 Trigger Enabled (Read Only)
F*f*:1	16387 + *b*	D16386 + *b*	REAL	%MD*f*.1	DINT	Read/Write	Plot Samples (e.g. 1000)
F*f*:2	16389 + *b*	D16388 + *b*	REAL	%MD*f*.2	REAL	Read/Write	Plot Sample Period (seconds)
F*f*:3	16391 + *b*	D16390 + *b*	REAL	%MD*f*.3	DINT	Read/Write	Plot Axis Owner 0-3, -1 = none
F*f*:4	16393 + *b*	D16392 + *b*	REAL	%MD*f*.4	REAL	Read/Write	Plot Trigger Position %, 0-100, -1 = auto rearm
F*f*:5	16395 + *b*	D16394 + *b*	REAL	%MD*f*.5	DWORD	Read/Write	Plot Trigger Type 0-7Trigger Type (0=none, 1=motion command) 8-23 Depends on trigger type Motion Commands: 8-11 Axis 0-3 bits, if all zero, then use Axis Owner 12-23 Reserved
F*f*:6	16397 + *b*	D16396 + *b*	REAL	%MD*f*.6	-	-	Reserved
F*f*:7	16399 + *b*	D16398 + *b*	REAL	%MD*f*.7	-	-	Reserved
F*f*:8	16401 + *b*	D16400 + *b*	REAL	%MD*f*.8	DINT	Read Only	Plot ID
F*f*:9	16403 + *b*	D16402 + *b*	REAL	%MD*f*.9	DINT	Read Only	Plot State 0 = not triggered, 1 = capturing, 2 = complete
F*f*:10	16405 + *b*	D16404 + *b*	REAL	%MD*f*.10	DINT	Read Only	Plot Captured Samples Number of plot samples captured. Only applies for Plot State > 0.
F*f*:11	16407 + *b*	D16406 + *b*	REAL	%MD*f*.11	DINT	Read Only	Plot Sample 0 Time Time that first plot sample was captured. In

							control loops since controller startup (low 24 bits). Only applies for Plot State > 0.
F*f*:12	16409 + *b*	D16408 + *b*	REAL	%MD*f*.12	DINT	Read Only	Plot Trigger Time Time that plot trigger occurred. In control loops since controller startup (low 24 bits). Only applies for Plot State > 0.
F*f*:13	16411 + *b*	D16410 + *b*	REAL	%MD*f*.13	DINT	Read Only	Plot Trigger Index Index of the plot sample at which the plot trigger occurred. Only applies for Plot State > 0.
F*f*:14-15	16413 + *b to* 16415 + *b*	D16412 + *b* D16414 + *b*	-	%MD*f*.14-15	-	-	Reserved
F*f*:16-31	16417 + *b to* 16447 + *b*	D16416 + *b* D16446 + *b*	REAL	%MD*f*.16-31	DWORD	Read/Write	Plot Data Sets 0-15 Addresses Files %MD*n*:16-31 contain the Addresses for plot Data Sets 0-15. Bits 0-11 Element Bits 12-23File

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, Files 40-47: Dynamic Plot Upload Area

The following files contain the Dynamic Plot Upload Area registers. See the Reading Plots with a Host Controller topic for details on how to use them.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading data of DWORD or DINT external data types.

Note:
When communicating via a protocol that uses DF1 addressing, the Plot Data can be accessed with registers 5-4095, if the host controller allows it.
Other protocols can only address Plot Data registers 5-255, as indicated below.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
Plot 0							

F40:0	20481	D20480	DINT	%MD40.0	UDINT	Read/Write	Plot 0 Flags
F40:1	20483	D20482	DINT	%MD40.1	UDINT	Read/Write	Plot 0 Requested Read Samples
F40:2	20485	D20484	DINT	%MD40.2	UDINT	Read/Write	Plot 0 Current Index
F40:3	20487	D20486	DINT	%MD40.3	UDINT	Read Only	Plot 0 ID
F40:4	20489	D20488	DINT	%MD40.4	UDINT	Read Only	Plot 0 Samples Uploaded
F40:5-255	20491-20991	D20490-D20990	*	%MD40.5-255	*	Read Only	Plot 0 Data
Plot 1							
F41:0	20993	D20992	DINT	%MD41.0	UDINT	Read/Write	Plot 1 Flags
F41:1	20995	D20994	DINT	%MD41.1	UDINT	Read/Write	Plot 1 Requested Read Samples
F41:2	20997	D20996	DINT	%MD41.2	UDINT	Read/Write	Plot 1 Current Index
F41:3	20999	D20998	DINT	%MD41.3	UDINT	Read Only	Plot 1 ID
F41:4	21001	D21000	DINT	%MD41.4	UDINT	Read Only	Plot 1 Samples Uploaded
F41:5-255	21003-21503	D21002-D21502	*	%MD41.5-255	*	Read Only	Plot 1 Data
Plot 2							
F42:0	21505	D21504	DINT	%MD42.0	UDINT	Read/Write	Plot 2 Flags
F42:1	21507	D21506	DINT	%MD42.1	UDINT	Read/Write	Plot 2 Requested Read Samples
F42:2	21509	D21508	DINT	%MD42.2	UDINT	Read/Write	Plot 2 Current Index
F42:3	21511	D21510	DINT	%MD42.3	UDINT	Read Only	Plot 2 ID
F42:4	21513	D21512	DINT	%MD42.4	UDINT	Read Only	Plot 2 Samples Uploaded
F42:5-255	21515-22015	D21514 D22014	*	%MD42.5-255	*	Read Only	Plot 2 Data
Plot 3							
F43:0	22017	D22016	DINT	%MD43.0	UDINT	Read/Write	Plot 3 Flags
F43:1	22019	D22018	DINT	%MD43.1	UDINT	Read/Write	Plot 3 Requested Read Samples
F43:2	22021	D22020	DINT	%MD43.2	UDINT	Read/Write	Plot 3 Current Index
F43:3	22023	D22022	DINT	%MD43.3	UDINT	Read Only	Plot 3 ID
F43:4	22025	D22024	DINT	%MD43.4	UDINT	Read Only	Plot 3 Samples Uploaded
F43:5-255	22027-22527	D22026-D22526	*	%MD43.5-255	*	Read Only	Plot 3 Data
Plot 4							
F44:0	22529	D22528	DINT	%MD44.0	UDINT	Read/Write	Plot 4 Flags
F44:1	22531	D22530	DINT	%MD44.1	UDINT	Read/Write	Plot 4 Requested Read Samples
F44:2	22533	D22532	DINT	%MD44.2	UDINT	Read/Write	Plot 4 Current Index
F44:3	22535	D22534	DINT	%MD44.3	UDINT	Read Only	Plot 4 ID
F44:4	22537	D22536	DINT	%MD44.4	UDINT	Read Only	Plot 4 Samples Uploaded
F44:5-255	22539-23039	D22538-	*	%MD44.5-255	*	Read Only	Plot 4 Data

		D23038					
Plot 5							
F45:0	23041	D23040	DINT	%MD45.0	UDINT	Read/Write	Plot 5 Flags
F45:1	23043	D23042	DINT	%MD45.1	UDINT	Read/Write	Plot 5 Requested Read Samples
F45:2	23045	D23044	DINT	%MD45.2	UDINT	Read/Write	Plot 5 Current Index
F45:3	23047	D23046	DINT	%MD45.3	UDINT	Read Only	Plot 5 ID
F45:4	23049	D23048	DINT	%MD45.4	UDINT	Read Only	Plot 5 Samples Uploaded
F45:5-255	23051-23551	D23050 D23550	*	%MD45.5-255	*	Read Only	Plot 5 Data
Plot 6							
F46:0	23553	D23552	DINT	%MD46.0	UDINT	Read/Write	Plot 6 Flags
F46:1	23555	D23554	DINT	%MD46.1	UDINT	Read/Write	Plot 6 Requested Read Samples
F46:2	23557	D23556	DINT	%MD46.2	UDINT	Read/Write	Plot 6 Current Index
F46:3	23559	D23558	DINT	%MD46.3	UDINT	Read Only	Plot 6 ID
F46:4	23561	D23560	DINT	%MD46.4	UDINT	Read Only	Plot 6 Samples Uploaded
F46:5-255	23563-24063	D23562-D24062	*	%MD46.5-255	*	Read Only	Plot 6 Data
Plot 7							
F47:0	24065	D24064	DINT	%MD47.0	UDINT	Read/Write	Plot 7 Flags
F47:1	24067	D24066	DINT	%MD47.1	UDINT	Read/Write	Plot 7 Requested Read Samples
F47:2	24069	D24068	DINT	%MD47.2	UDINT	Read/Write	Plot 7 Current Index
F47:3	24071	D24070	DINT	%MD47.3	UDINT	Read Only	Plot 7 ID
F47:4	24073	D24072	DINT	%MD47.4	UDINT	Read Only	Plot 7 Samples Uploaded
F47:5-255	24075-24575	D24074-D24574	*	%MD47.5-255	*	Read Only	Plot 7 Data

*The Data Types of the Plot Data are determined by the plotted registers.

See Also

RMC75 Register Map Overview

Back

RMC75 Registers, Files 48-55: Static Plot Upload Area

All Static Plot Upload Area Registers are **Read Only**. See the Reading Plots with a Host Controller topic for details on how to use them.

The data types of the plot data is determined by the plotted registers.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading plot data of DWORD or DINT data types.

Note:
When communicating via a protocol that uses DF1 addressing, samples 0-4095 from each plot can be addressed, if the host controller allows it. For example, F112:4095 addresses sample 4095 of plot 0, data set 0.
Other protocols can only address the first 256 samples, as indicated below.

AB DF1,CSP Address	Modbus TCP,RTU Address (only 256 registers)	FINS Address (only 256 registers)	Internal IEC Address	Register Name
Plot 0				
F48:0-4095	24577-25087	D24576-D25086	%MD48.0-4095	Plot 0, Data Set 0, Samples 0-4095
F49:0-4095	25089-25599	D25088-D25568	%MD49.0-4095	Plot 0, Data Set 1, Samples 0-4095
F50:0-4095	25601-26111	D25600-D26110	%MD50.0-4095	Plot 0, Data Set 2, Samples 0-4095
F51:0-4095	26113-26623	D26112-D26622	%MD51.0-4095	Plot 0, Data Set 3, Samples 0-4095
Plot 1				
F52:0-4095	26625-27135	D26624-D27134	%MD52.0-4095	Plot 1, Data Set 0, Samples 0-4095
F53:0-4095	27137-27647	D27136-D27646	%MD53.0-4095	Plot 1, Data Set 1, Samples 0-4095
F54:0-4095	27649-28159	D27648-D28158	%MD54.0-4095	Plot 1, Data Set 2, Samples 0-4095
F55:0-4095	28161-28671	D28160-D28670	%MD55.0-4095	Plot 1, Data Set 3, Samples 0-4095

*The Data Types of the Plot Data Sets are determined by the plotted registers.

See Also
RMC75 Register Map Overview

Back

RMC75 Registers, Files 56-59, 64-67, 68: Variables Registers

All variable registers are **Read/Write**.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading variables defined as DWORD or DINT data types.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Variables - Current Values						
F56:0-255	28673-29183	D28672-D29182	*	%MD56.0-255	*	Variables 0-255 - Current Values
F57:0-255	29185-29695	D29184-D29694	*	%MD57.0-255	*	Variables 256-511 - Current Values
F58:0-255	29697-30207	D29696-D30206	*	%MD58.0-255	*	Variables 512-767 - Current Values

F59:0-255	30209-30719	D30208-D30718	*	%MD59.0-255	*	Variables 768-1023 - Current Values
Variables - Initial Values						
F64:0-255	32769-33279	D32768-D33278	*	%MD64.0-255	*	Variables 0-255 - Initial Values
F65:0-255	33281-33791	D33280-D33790	*	%MD65.0-255	*	Variables 256-511 - Initial Values
F66:0-255	33793-34303	D33792-D34302	*	%MD66.0-255	*	Variables 512-767 - Initial Values
F67:0-255	34305-34815	D34304-D34814	*	%MD67.0-255	*	Variables 768-1023 - Initial Values
Variables - Attributes						
F68:0-255	34817-35327	D34816-D35326	DWORD	%MD68.0-255	DWORD	Variables 0-1023 - Attributes (4 variables per register)

* The Data Types of the variables are specified by the user when defining a variable in the Variable Table.

Allen-Bradley DF1 and CSP

Where allowed by the host controller's communications, all the variables can also be addressed as F56:*n*, up to *n* = 1023.

Modbus/TCP and /RTU

The address of the current value of variable *n* is 28673 + 2 x *n*.

FINS

The address of the current value of variable *n* is D28672 + 2 x *n*.

See Also

RMC75 Register Map Overview | Variable Attributes

9.5. RMC150 Register Map

9.5.1. RMC150 Register Map

The RMC150 Register Map lists the addresses of all the registers in the RMC150. Typically, you will need to use the register map to find addresses when setting up communications with the RMC150 from a host controller such as a PLC. When referencing registers from *within* the RMC, such as in user programs, you do not need to use register addresses. You can use tag names instead. See the Tags Overview topic for details.

Tip: The Address Maps in RMCTools provide any easy way to browse all the registers in the RMC, along with their addresses.

For each register, the RMC150 register map provides addresses in the formats listed below. The address type you use will depend on the communication method you use. For more details on the addressing formats, see the respective topics.

- DF1 (Allen-Bradley) Addressing

- Modbus Addressing
- FINS (Omron) Addressing
- IEC-61131 Addressing

For PROFINET Record Data addressing, see PROFINET Data Records Addressing.

Follow the links below for the addresses of each section of registers.
The registers are divided into the following sections:

File	Description
7	Controller Info
8-23	Axis 0-15 Status Registers
24-39	Axis 0-15 Parameters
40	Command Area
41	Indirect Data Map Definition
42	Indirect Data Map
43	Axis Definitions
44	Controller Status/Parameters
45	Communication Configuration
46	Event Log Configuration
47	Discrete I/O
48	Tasks 0-9 Status/Configuration
49	Axis Names
56-59	Variables - Current Values
72-75	Variables - Initial Values
88	Variables - Attributes
94	Image Upload/Download Area
95	Plot Layout
96-103	Plots 0-7 Status/Configuration
104-111	Dynamic Plot Upload Area
112-143	Static Plot Upload Area
144-149	Slot Settings

See Also
Register Map Overview | RMC75 Register Map | RMC200 Register Map

Show All Back

RMC150 Registers, File 7: Controller Info

All Controller Information registers are **Read Only**.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
F7:0	3585	D03584	REAL	%MD7.0	DINT	Product ID 2: RMC150 series
F7:1	3587	D03586	REAL	%MD7.1	DINT	State 1: Running Control Program 2: Running Control Program (in debugger) 3: Running Loader
F7:2	3589	D03588	REAL	%MD7.2	DINT	CPU Module ID (slot 1) 16: RMC150E 17: RMC151E
F7:3	3591	D03590	REAL	%MD7.3	DINT	CPU Module Rev (slot 1) Major * 256 + Minor
F7:4	3593	D03592	REAL	%MD7.4	DINT	Comm Module ID (slot 0) 0: Blank 38: Serial (not supported) 40: DI/O 41: PROFIBUS 42: ENET (not supported) 43: Modbus Plus (not supported) 45: Universal I/O
F7:5	3595	D03594	REAL	%MD7.5	DINT	Comm Module Rev (slot 0) Major * 256 + Minor
F7:6	3597	D03596	REAL	%MD7.6	DINT	Sensor Slot 1 Module ID (slot 2) 0: None 64: MDT (M) 66: Analog (H) 68: Analog Inputs (A) 69: Quadrature (Q) 70: SSI (S) 72: DI/O (D) 74: Analog (G) 76: Resolver (R) 77: Universal I/O 80: Resolver (RW)
F7:7	3599	D03598	REAL	%MD7.7	DINT	Sensor Slot 1 Module Rev (slot 2) Major * 256 + Minor
F7:8	3601	D03600	REAL	%MD7.8	DINT	Sensor Slot 2 Module ID (slot 3)
F7:9	3603	D03602	REAL	%MD7.9	DINT	Sensor Slot 2 Module Rev (slot 3)

F7:10	3605	D03604	REAL	%MD7.10	DINT	Sensor Slot 3 Module ID (slot 4)
F7:11	3607	D03606	REAL	%MD7.11	DINT	Sensor Slot 3 Module Rev (slot 4)
F7:12	3609	D03608	REAL	%MD7.12	DINT	Sensor Slot 4 Module ID (slot 5)
F7:13	3611	D03610	REAL	%MD7.13	DINT	Sensor Slot 4 Module Rev (slot 5)
F7:14	3613	D03612	DINT	%MD7.14	DINT	Serial Number 8-digit serial number (e.g. 81074039)
F7:15	3615	D03614	REAL	%MD7.15	DINT	Firmware Rev Major * 256 + Minor The patch release number is given in F7:32.
F7:16	3617	D03616	REAL	%MD7.16	DINT	Firmware Special Release Code 0: Standard 1-127: Special Release (S1-S127) 128: Beta Standard 129-255: Beta Special Release (S1-S127)
F7:17	3619	D03618	REAL	%MD7.17	DINT	Firmware Configuration ID 1: A
F7:18	3621	D03620	REAL	%MD7.18	DINT	Firmware Year and Month Year (4-digit) * 16 + Month (1=Jan, 2=Feb, ...)
F7:19	3623	D03622	REAL	%MD7.19	DINT	Firmware Day and Time Date (1-31) * 2048 + Hour (0-23) * 64 + Minute
F7:20	3625	D03624	REAL	%MD7.20	DINT	reserved
F7:21	3627	D03626	REAL	%MD7.21	DINT	reserved
F7:22	3629	D03628	REAL	%MD7.22	DINT	Flash Rev Major * 256 + Minor
F7:23	3631	D03630	REAL	%MD7.23	DINT	Required RMCTools Ver Major * 256 + Minor (patch ignored)
F7:24	3633	D03632	REAL	%MD7.24	DINT	Suggested RMCTools Ver Major * 256 + Minor (patch ignored)
F7:25	3635	D03634	REAL	%MD7.25	DINT	Loader Rev Major * 256 + Minor
F7:26	3637	D03636	REAL	%MD7.26	DINT	Loader Year and Month Year (4-digit) * 16 + Month (1=Jan, 2=Feb, ...)
F7:27	3639	D03638	REAL	%MD7.27	DINT	Loader Day and Time

						Date (1-31) * 2048 + Hour (0-23) * 64 + Minute
F7:28	3641	D03640	REAL	%MD7.28	DINT	Loader Command For sending certain commands to the loader, such as restart the RMC. See Loader Command for more details.
F7:29	3643	D03642	REAL	%MD7.29	DINT	Reason in Loader 0: Not in Loader 1: Explicitly Requested by Firmware (e.g. Firmware Update) 2: Bad FW Image Checksum 3: Invalidate FW Image Header 4: CPU Module not supported by FW Image 6: Watchdog Timeout 7: Boot of FW Image Failed 9: Force-to-Loader Jumper is set
F7:30	3645	D03644	REAL	%MD7.30	DINT	Loader State Only used when updating. Confidential
F7:31	3647	D03646	REAL	%MD7.31	DINT	CPU Board Version This indicates the actual CPU board revision. Bits 0-7: Mod Level (0=A, 1=B, etc.) Bits 8-15: Minor Revision Bits 16-23: Major Revision Bits 24-31: Reserved For example, 2.1E will be 0x00020104.
F7:32	3649	D03648	REAL	%MD7.32	DINT	Firmware Patch Number Holds the patch level of the firmware version. For example, for 3.30.0, this value will be 0, and for 3.30.1, it will be 1.

See Also

RMC150 Register Map

Back

RMC150 Registers, Files 8-23: Axis Status Registers

All Axis Status Registers are **Read Only**.

n = axis number (0-15)

f = 8 + *n*

b = 512 x *n*

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Common Registers: All Axes						
F*f*:0	4097 + *b*	D4096 + *b*	DWORD	%MD*f*.0	DWORD	Status Bits
F*f*:1	4099 + *b*	D4098 + *b*	DWORD	%MD*f*.1	DWORD	Error Bits
F*f*:2	4101 + *b*	D4100 + *b*	REAL	%MD*f*.2	DINT	Last Error Number
F*f*:4	4105 + *b*	D4104 + *b*	REAL	%MD*f*.4	REAL	Read Response
Position/Velocity Control						
F*f*:6	4109 + *b*	D4108 + *b*	REAL	%MD*f*.6	DINT	Current Control Mode
F*f*:7	4111 + *b*	D4110 + *b*	REAL	%MD*f*.7	DINT	Next Pos/Vel Control Mode
Primary Input: Position/Velocity Axes						
F*f*:8	4113 + *b*	D4112 + *b*	REAL	%MD*f*.8	REAL	Actual Position
F*f*:9	4115 + *b*	D4114 + *b*	REAL	%MD*f*.9	REAL	Actual Velocity
F*f*:10	4117 + *b*	D4116 + *b*	REAL	%MD*f*.10	REAL	Actual Acceleration
F*f*:11	4119 + *b*	D4118 + *b*	REAL	%MD*f*.11	REAL	Counts/Current/Voltage
F*f*:12	4121 + *b*	D4120 + *b*	DINT	%MD*f*.12	DINT	Raw Counts
Primary Input: Single-Input Pressure, Force, or Acceleration Axes						
F*f*:8	4113 + *b*	D4112 + *b*	REAL	%MD*f*.8	REAL	Actual Pressure/Force, Actual Acceleration
F*f*:9	4115 + *b*	D4114 + *b*	REAL	%MD*f*.9	REAL	Actual Pressure/Force Rate, Actual Jerk
F*f*:11	4119 + *b*	D4118 + *b*	REAL	%MD*f*.11	REAL	Current/Voltage
F*f*:12	4121 + *b*	D4120 + *b*	DINT	%MD*f*.12	DINT	Raw Counts
Primary Input: Dual-Input Force or Acceleration Axes						
F*f*:8	4113 + *b*	D4112 + *b*	REAL	%MD*f*.8	REAL	Actual Differential Force, Actual Acceleration
F*f*:9	4115 + *b*	D4114 + *b*	REAL	%MD*f*.9	REAL	Actual Differential Force Rate, Actual Jerk
F*f*:10	4117 + *b*	D4116 + *b*	REAL	%MD*f*.10	REAL	Actual Force A, Channel A Acceleration
F*f*:11	4119 + *b*	D4118 + *b*	REAL	%MD*f*.11	REAL	Voltage A/Current A
F*f*:12	4121 + *b*	D4120 + *b*	DINT	%MD*f*.12	DINT	Raw Counts A
F*f*:13	4123 + *b*	D4122 + *b*	REAL	%MD*f*.13	REAL	Actual Force B, Channel B Acceleration
F*f*:14	4125 + *b*	D4124 + *b*	REAL	%MD*f*.14	REAL	Voltage B/Current B
F*f*:15	4127 + *b*	D4126 + *b*	DINT	%MD*f*.15	DINT	Raw Counts B
Home/Registration: Quadrature Axes						
F*f*:18	4133 + *b*	D4132 + *b*	DWORD	%MD*f*.18	DWORD	Encoder Status Bits
F*f*:19	4135 + *b*	D4134 + *b*	REAL	%MD*f*.19	REAL	Registration 0 Position
F*f*:20	4137 + *b*	D4136 + *b*	REAL	%MD*f*.20	REAL	Registration 1 Position
Secondary Input: Single-Input Pressure, Force, or Acceleration Axes						

F*f*:23	4143 + *b*	D4142 + *b*	REAL	%MD*f*.23	REAL	Actual Pressure/Force, Actual Acceleration
F*f*:24	4145 + *b*	D4144 + *b*	REAL	%MD*f*.24	REAL	Actual Pressure/Force Rate, Actual Jerk
F*f*:26	4149 + *b*	D4148 + *b*	REAL	%MD*f*.26	REAL	Current/Voltage
F*f*:27	4151 + *b*	D4150 + *b*	DINT	%MD*f*.27	DINT	Raw Counts
Secondary Input: Dual-Input Force or Acceleration Axes						
F*f*:23	4143 + *b*	D4142 + *b*	REAL	%MD*f*.23	REAL	Actual Differential Force, Actual Acceleration
F*f*:24	4145 + *b*	D4144 + *b*	REAL	%MD*f*.24	REAL	Actual Differential Force Rate, Actual Jerk
F*f*:25	4147 + *b*	D4146 + *b*	REAL	%MD*f*.25	REAL	Actual Force A, Channel A Acceleration
F*f*:26	4149 + *b*	D4148 + *b*	REAL	%MD*f*.26	REAL	Voltage A/Current A
F*f*:27	4151 + *b*	D4150 + *b*	DINT	%MD*f*.27	DINT	Raw Counts A
F*f*:28	4153 + *b*	D4152 + *b*	REAL	%MD*f*.28	REAL	Actual Force B, Channel B Acceleration
F*f*:29	4155 + *b*	D4154 + *b*	REAL	%MD*f*.29	REAL	Voltage B/Current B
F*f*:30	4157 + *b*	D5156 + *b*	DINT	%MD*f*.30	DINT	Raw Counts B
Output: Analog Control Output Axes						
F*f*:33	4163 + *b*	D4162 + *b*	REAL	%MD*f*.33	REAL	Control Output
Primary Control: Position/Velocity Axes						
F*f*:35	4167 + *b*	D4166 + *b*	REAL	%MD*f*.35	REAL	Position Error
F*f*:36	4169 + *b*	D4168 + *b*	REAL	%MD*f*.36	REAL	Velocity Error
F*f*:37	4171 + *b*	D4170 + *b*	REAL	%MD*f*.37	REAL	Proportional Term
F*f*:38	4173 + *b*	D4172 + *b*	REAL	%MD*f*.38	REAL	Integral Term
F*f*:39	4175 + *b*	D4174 + *b*	REAL	%MD*f*.39	REAL	Differential Term
F*f*:40	4177 + *b*	D4176 + *b*	REAL	%MD*f*.40	REAL	Double Differential Output Term
F*f*:41	4179 + *b*	D4178 + *b*	REAL	%MD*f*.41	REAL	Velocity Feed Forward Term
F*f*:42	4181 + *b*	D4180 + *b*	REAL	%MD*f*.42	REAL	Acceleration Feed Forward Term
F*f*:43	4183 + *b*	D4182 + *b*	REAL	%MD*f*.43	REAL	Jerk Feed Forward Term
F*f*:44	4185 + *b*	D4184 + *b*	REAL	%MD*f*.44	REAL	Triple Differential Output Term
F*f*:45	4187 + *b*	D4186 + *b*	REAL	%MD*f*.45	REAL	PFID Output
F*f*:47	4191 + *b*	D4190 + *b*	REAL	%MD*f*.47	DINT	Current Integrator Mode
Primary Control: Pressure or Force Axes						
F*f*:35	4167 + *b*	D4166 + *b*	REAL	%MD*f*.35	REAL	Pressure/Force Error
F*f*:37	4171 + *b*	D4170 + *b*	REAL	%MD*f*.37	REAL	Pressure/Force Proportional Term
F*f*:38	4173 + *b*	D4172 + *b*	REAL	%MD*f*.38	REAL	Pressure/Force Integral Term
F*f*:39	4175 + *b*	D4174 + *b*	REAL	%MD*f*.39	REAL	Pressure/Force Differential Term
F*f*:40	4177 + *b*	D4176 + *b*	REAL	%MD*f*.40	REAL	Pressure/Force Feed Forward Term
F*f*:41	4179 + *b*	D4178 + *b*	REAL	%MD*f*.41	REAL	Pressure/Force Rate Feed Forward Term
F*f*:45	4187 + *b*	D4186 + *b*	REAL	%MD*f*.45	REAL	PFID Output

F*f*:47	4191 + *b*	D4190 + *b*	REAL	%MD*f*.47	DINT	Current Integrator Mode
Secondary Control: Pressure or Force Axes						
F*f*:46	4189 + *b*	D4188 + *b*	REAL	%MD*f*.46	REAL	Pressure/Force Error
F*f*:48	4193 + *b*	D4192 + *b*	REAL	%MD*f*.48	REAL	Pressure/Force Proportional Term
F*f*:49	4195 + *b*	D4194 + *b*	REAL	%MD*f*.49	REAL	Pressure/Force Integral Term
F*f*:50	4197 + *b*	D4196 + *b*	REAL	%MD*f*.50	REAL	Pressure/Force Differential Term
F*f*:51	4199 + *b*	D4198 + *b*	REAL	%MD*f*.51	REAL	Pressure/Force Feed Forward Term
F*f*:52	4201 + *b*	D4200 + *b*	REAL	%MD*f*.52	REAL	Pressure/Force Rate Feed Forward Term
Target						
F*f*:53	4203 + *b*	D4202 + *b*	REAL	%MD*f*.53	REAL	Target Position
F*f*:54	4205 + *b*	D4204 + *b*	REAL	%MD*f*.54	REAL	Target Velocity
F*f*:55	4207 + *b*	D4206 + *b*	REAL	%MD*f*.55	REAL	Target Acceleration
F*f*:56	4209 + *b*	D4208 + *b*	REAL	%MD*f*.56	REAL	Command Position
F*f*:57	4211 + *b*	D4210 + *b*	REAL	%MD*f*.57	REAL	Command Velocity
F*f*:58	4213 + *b*	D4212 + *b*	REAL	%MD*f*.58	REAL	Target Jerk
F*f*:59	4215 + *b*	D4214 + *b*	REAL	%MD*f*.59	DINT	Cycles
F*f*:60	4217 + *b*	D4216 + *b*	REAL	%MD*f*.60	REAL	Target Pressure/Force
F*f*:61	4219 + *b*	D4218 + *b*	REAL	%MD*f*.61	REAL	Command Pressure/Force
F*f*:62	4221 + *b*	D4220 + *b*	REAL	%MD*f*.62	DINT	Cycles (Pressure/Force)
Custom Feedback						
F*f*:64	4225 + *b*	D4224 + *b*	DWORD	%MD*f*.64	DWORD	Custom Error Bits
F*f*:65	4227 + *b*	D4226 + *b*	REAL	%MD*f*.65	REAL	Primary Custom Counts
F*f*:66	4229 + *b*	D4228 + *b*	REAL	%MD*f*.66	REAL	Secondary Custom Counts

See Also

RMC150 Register Map

Back

RMC150 Registers, Files 24-39: Axis Parameter Registers

All Axis Parameter registers are **Read/Write**.

n = axis number (0-15)

f = 24 + *n*

b = 512 x *n*

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Primary Feedback: Position Axes						
F*f*:0	12289 + *b*	D12288 + *b*	REAL	%MD*f*.0	REAL	Position Scale
F*f*:1	12291 + *b*	D12290 + *b*	REAL	%MD*f*.1	REAL	Position Offset

F*f*:2	12293 + *b*	D12292 + *b*	REAL	%MD*f*.2	REAL	Actual Position Filter
F*f*:3	12295 + *b*	D12294 + *b*	REAL	%MD*f*.3	REAL	Actual Velocity Filter
F*f*:4	12297 + *b*	D12296 + *b*	REAL	%MD*f*.4	REAL	Actual Acceleration Filter
F*f*:5	12299 + *b*	D12298 + *b*	REAL	%MD*f*.5	REAL	Stop Threshold
F*f*:6	12301 + *b*	D12300 + *b*	REAL	%MD*f*.6	REAL	Noise Error Rate
F*f*:8	12305 + *b*	D12304 + *b*	REAL	%MD*f*.8	DWORD	Custom Feedback Configuration Register
F*f*:9	12307 + *b*	D12306 + *b*	REAL	%MD*f*.9	DWORD	Primary Input Bits Register
Primary Feedback: Velocity Axes						
F*f*:0	12289 + *b*	D12288 + *b*	REAL	%MD*f*.0	REAL	Velocity Scale
F*f*:1	12291 + *b*	D12290 + *b*	REAL	%MD*f*.1	REAL	Velocity Offset
F*f*:2	12293 + *b*	D12292 + *b*	REAL	%MD*f*.2	REAL	Velocity Deadband
F*f*:3	12295 + *b*	D12294 + *b*	REAL	%MD*f*.3	REAL	Actual Velocity Filter
F*f*:4	12297 + *b*	D12296 + *b*	REAL	%MD*f*.4	REAL	Actual Acceleration Filter
F*f*:5	12299 + *b*	D12298 + *b*	REAL	%MD*f*.5	REAL	Stop Threshold
F*f*:6	12301 + *b*	D12300 + *b*	REAL	%MD*f*.6	REAL	Noise Error Rate
F*f*:8	12305 + *b*	D12304 + *b*	REAL	%MD*f*.8	DWORD	Custom Feedback Configuration
Primary Feedback: Single-Input Pressure, Force or Acceleration						
F*f*:0	12289 + *b*	D12288 + *b*	REAL	%MD*f*.0	REAL	Pressure/Force Scale, Acceleration Scale
F*f*:1	12291 + *b*	D12290 + *b*	REAL	%MD*f*.1	REAL	Pressure/Force Offset, Acceleration Offset
F*f*:4	12297 + *b*	D12296 + *b*	REAL	%MD*f*.4	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F*f*:5	12299 + *b*	D12298 + *b*	REAL	%MD*f*.5	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F*f*:6	12301 + *b*	D12300 + *b*	REAL	%MD*f*.6	REAL	Noise Error Rate

F*f*:8	12305 + *b*	D12304 + *b*	REAL	%MD*f*.8	DWORD	Custom Feedback Configuration
Primary Feedback: Dual-Input Force or Acceleration						
F*f*:0	12289 + *b*	D12288 + *b*	REAL	%MD*f*.0	REAL	Force A Scale, Channel A Acceleration Scale
F*f*:1	12291 + *b*	D12290 + *b*	REAL	%MD*f*.1	REAL	Force A Offset, Channel A Acceleration Offset
F*f*:2	12293 + *b*	D12292 + *b*	REAL	%MD*f*.2	REAL	Force B Scale, Channel B Acceleration Scale
F*f*:3	12295 + *b*	D12294 + *b*	REAL	%MD*f*.3	REAL	Force B Offset, Channel B Acceleration Offset
F*f*:4	< td style="border-right: Solid 1px #bfbfbf; border-bottom: Solid 1px #bfbfbf; padding-left: 3px; padding-right: 0px;" width="70"> 12297 + *b* D12296 + *b*	REAL	%MD*f*.4	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter	
F*f*:5	12299 + *b*	D12298 + *b*	REAL	%MD*f*.5	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F*f*:6	12301 + *b*	D12300 + *b*	REAL	%MD*f*.6	REAL	Noise Error Rate
Primary Feedback: SSI/MDT Transducer						
F*f*:10	12309 + *b*	D12308 + *b*	REAL	%MD*f*.10	DWORD	SSI/MDT Config Register
F*f*:11	12311 + *b*	D12310 + *b*	REAL	%MD*f*.11	REAL	Count Offset
F*f*:12	12313 + *b*	D12312 + *b*	REAL	%MD*f*.12	REAL	Position Unwind
F*f*:13	12315 + *b*	D12314 + *b*	REAL	%MD*f*.13	DINT	Count Unwind
F*f*:14	12317 + *b*	D12316 + *b*	REAL	%MD*f*.14	DINT	SSI Wire Delay
Primary Feedback: Analog Transducer						
F*f*:10	12309 + *b*	D12308 + *b*	REAL	%MD*n*.10	DWORD	Analog Config Register

Primary Feedback: Quadrature Transducer						
F*f*:10	12309 + *b*	D12308 + *b*	REAL	%MD*n*.10	DWORD	Quadrature Config Register
F*f*:12	12313 + *b*	D12312 + *b*	REAL	%MD*n*.12	REAL	Position Unwind
F*f*:13	12315 + *b*	D12314 + *b*	REAL	%MD*n*.13	DINT	Count Unwind
Primary Feedback: Resolver Feedback						
F*f*:10	12309 + *b*	D12308 + *b*	REAL	%MD*n*.10	DWORD	Resolver Configuration Register
F*f*:14	12317 + *b*	D12316 + *b*	REAL	%MD*n*.14	REAL	Reference Frequency
F*f*:15	12319 + *b*	D12318 + *b*	REAL	%MD*n*.15	REAL	Reference Amplitude
Secondary Feedback: Single-Input Pressure, Force, or Acceleration						
F*f*:18	12325 + *b*	D12324 + *b*	REAL	%MD*f*.18	REAL	Pressure/Force Scale, Acceleration Scale
F*f*:19	12327 + *b*	D12326 + *b*	REAL	%MD*f*.19	REAL	Pressure/Force Offset, Acceleration Offset
F*f*:22	12333 + *b*	D12332 + *b*	REAL	%MD*f*.22	REAL	Actual Pressure/Force Filter, Actual Acceleration Filter
F*f*:23	12335 + *b*	D12334 + *b*	REAL	%MD*f*.23	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F*f*:24	12337 + *b*	D12336 + *b*	REAL	%MD*f*.24	REAL	Noise Error Rate
F*f*:26	12341 + *b*	D12340 + *b*	REAL	%MD*f*.26	DWORD	Custom Feedback Configuration Register
Secondary Feedback: Dual-Input Force or Acceleration						
F*f*:18	12325 + *b*	D12324 + *b*	REAL	%MD*f*.18	REAL	Force A Scale, Channel A Acceleration Scale
F*f*:19	12357 + *b*	D12356 + *b*	REAL	%MD*f*.19	REAL	Force A Offset, Channel A Acceleration Offset
F*f*:20	12329 + *b*	D12328 + *b*	REAL	%MD*f*.20	REAL	Force B Scale, Channel B Acceleration Scale
F*f*:21	12331 + *b*	D12330 + *b*	REAL	%MD*f*.21	REAL	Force B Offset, Channel B Acceleration Offset
F*f*:22	12333 + *b*	D12332 + *b*	REAL	%MD*f*.22	REAL	Actual Pressure/Force Filter,

						Actual Acceleration Filter
F*f*:23	12335 + *b*	D12334 + *b*	REAL	%MD*f*.23	REAL	Actual Pressure/Force Rate Filter, Actual Jerk Filter
F*f*:24	12337 + *b*	D12336 + *b*	REAL	%MD*f*.24	REAL	Noise Error Rate
Secondary Feedback: Analog Transducer						
F*f*:28	12345 + *b*	D12344 + *b*	REAL	%MD*f*.28	DWORD	Analog Config Register
Analog Control Output						
F*f*:32	12353 + *b*	D12352 + *b*	REAL	%MD*f*.32	REAL	Output Limit
F*f*:33	12355 + *b*	D12354 + *b*	REAL	%MD*f*.33	REAL	Output Bias
F*f*:34	12357 + *b*	D12356 + *b*	REAL	%MD*f*.34	DWORD	Output Register
Final Output Stage						
F*f*:38	12365 + *b*	D12354 + *b*	REAL	%MD*f*.38	REAL	Output Scale
F*f*:39	12367 + *b*	D12356 + *b*	REAL	%MD*f*.39	REAL	Primary Output Filter
F*f*:40	12369 + *b*	D12368 + *b*	REAL	%MD*f*.40	REAL	Secondary Output Filter
F*f*:41	12371 + *b*	D12370 + *b*	REAL	%MD*f*.41	REAL	Deadband Tolerance
F*f*:42	12373 + *b*	D12372 + *b*	REAL	%MD*f*.42	REAL	Output Deadband
F*f*:44	12377 + *b*	D12376 + *b*	REAL	%MD*f*.44	REAL	Knee Command Voltage
F*f*:45	12379 + *b*	D12378 + *b*	REAL	%MD*f*.45	REAL	Knee Flow Percentage
F*f*:46	12381 + *b*	D12380 + *b*	REAL	%MD*f*.46	DINT	Valve Linearization Curve ID
Primary Control: Servo Position Axes						
F*f*:43	12375 + *b*	D12374 + *b*	REAL	%MD*f*.43	DINT	Default Pos/Vel Control Mode
F*f*:56	12401 + *b*	D12400 + *b*	REAL	%MD*f*.56	REAL	In Position Tolerance
F*f*:57	12403 + *b*	D12402 + *b*	REAL	%MD*f*.57	REAL	Position Error Tolerance
F*f*:58	12405 + *b*	D12404 + *b*	REAL	%MD*f*.58	REAL	At Velocity Tolerance
F*f*:59	12407 + *b*	D12406 + *b*	REAL	%MD*f*.59	REAL	Velocity Error Tolerance
F*f*:60	12409 + *b*	D12408 + *b*	REAL	%MD*f*.60	DWORD	Primary Control Register

Position/Velocity Gain Set #0						
F*f*:61	12411 + *b*	D12410 + *b*	REAL	%MD*f*.61	REAL	Proportional Gain
F*f*:62	12413 + *b*	D12412 + *b*	REAL	%MD*f*.62	REAL	Integral Gain
F*f*:63	12415 + *b*	D12414 + *b*	REAL	%MD*f*.63	REAL	Differential Gain
F*f*:65	12419 + *b*	D12418 + *b*	REAL	%MD*f*.65	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)
F*f*:66	12421 + *b*	D12420 + *b*	REAL	%MD*f*.66	REAL	Acceleration Feed Forward
F*f*:67	12423 + *b*	D12422 + *b*	REAL	%MD*f*.67	REAL	Jerk Feed Forward
F*f*:68	12425 + *b*	D12424 + *b*	REAL	%MD*f*.68	REAL	Velocity Feed Forward (Negative)
F*f*:69	12427 + *b*	D12426 + *b*	REAL	%MD*f*.69	REAL	Double Differential Gain, Active Damping Proportional Gain
F*f*:70	12429 + *b*	D12428 + *b*	REAL	%MD*f*.70	REAL	Triple Differential Gain, Active Damping Differential Gain
F*f*:71	12431 + *b*	D12430 + *b*	REAL	%MD*f*.71	DINT	High-Order Control
Position/Velocity Gain Set #1						
F*f*:128	12545 + *b*	D12544 + *b*	REAL	%MD*f*.128	REAL	Proportional Gain
F*f*:129	12547 + *b*	D12546 + *b*	REAL	%MD*f*.129	REAL	Integral Gain
F*f*:130	12549 + *b*	D12548 + *b*	REAL	%MD*f*.130	REAL	Differential Gain
F*f*:132	12553 + *b*	D12552 + *b*	REAL	%MD*f*.132	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)
F*f*:133	12555 + *b*	D12554 + *b*	REAL	%MD*f*.133	REAL	Acceleration Feed Forward
F*f*:134	12557 + *b*	D12556 + *b*	REAL	%MD*f*.134	REAL	Jerk Feed Forward
F*f*:135	12559 + *b*	D12558 + *b*	REAL	%MD*f*.135	REAL	Velocity Feed Forward (Negative)
F*f*:136	12561 + *b*	D12560 + *b*	REAL	%MD*f*.136	REAL	Double Differential Gain, Active Damping Proportional Gain

F*f*:137	12563 + *b*	D12562 + *b*	REAL	%MD*f*.137	REAL	Triple Differential Gain, Active Damping Differential Gain
F*f*:138	12565 + *b*	D12564 + *b*	REAL	%MD*f*.138	DINT	High-Order Control
Primary Control: Servo Pressure or Force Axes						
F*f*:56	12401 + *b*	D12400 + *b*	REAL	%MD*f*.56	REAL	At Pressure/Force Tolerance
F*f*:57	12403 + *b*	D12402 + *b*	REAL	%MD*f*.57	REAL	Pressure/Force Error Tolerance
F*f*:60	12409 + *b*	D12408 + *b*	REAL	%MD*f*.60	DWORD	Primary Control Register
F*f*:61	12411 + *b*	D12410 + *b*	REAL	%MD*f*.61	REAL	Pressure/Force Proportional Gain
F*f*:62	12413 + *b*	D12412 + *b*	REAL	%MD*f*.62	REAL	Pressure/Force Integral Gain
F*f*:63	12415 + *b*	D12414 + *b*	REAL	%MD*f*.63	REAL	Pressure/Force Differential Gain
F*f*:64	12419 + *b*	D12418 + *b*	REAL	%MD*f*.65	REAL	Pressure/Force Feed Forward
F*f*:65	12421 + *b*	D12420 + *b*	REAL	%MD*f*.66	REAL	Pressure/Force Rate Feed Forward
Secondary Control: Servo Pressure or Force						
F*f*:76	12441 + *b*	D12440 + *b*	REAL	%MD*f*.76	REAL	At Pressure/Force Tolerance
F*f*:77	12443 + *b*	D12442 + *b*	REAL	%MD*f*.77	REAL	Pressure/Force Error Tolerance
F*f*:80	12449 + *b*	D12448 + *b*	REAL	%MD*f*.80	DWORD	Secondary Control Register
F*f*:81	12451 + *b*	D12450 + *b*	REAL	%MD*f*.81	REAL	Pressure/Force Proportional Gain
F*f*:82	12453 + *b*	D12452 + *b*	REAL	%MD*f*.82	REAL	Pressure/Force Integral Gain
F*f*:83	12455 + *b*	D12454 + *b*	REAL	%MD*f*.83	REAL	Pressure/Force Differential Gain
F*f*:84	12457 + *b*	D12456 + *b*	REAL	%MD*f*.84	REAL	Pressure/Force Feed Forward
F*f*:85	12459 + *b*	D12458 + *b*	REAL	%MD*f*.85	REAL	Pressure/Force Rate Feed Forward
Position Target						
F*f*:92	12473 + *b*	D12472 + *b*	REAL	%MD*f*.92	REAL	Positive Travel Limit
F*f*:93	12475 + *b*	D12474 + *b*	REAL	%MD*f*.93	REAL	Negative Travel Limit
F*f*:94	12477 + *b*	D12476 + *b*	REAL	%MD*f*.94	REAL	Requested Jerk
Pressure/Force Target						

F*f*:100	12489 + *b*	D12488 + *b*	REAL	%MD*f*.100	REAL	Positive Pressure/Force Limit
F*f*:101	12491 + *b*	D12490 + *b*	REAL	%MD*f*.101	REAL	Negative Pressure/Force Limit
Halts						
F*f*:106	12501 + *b*	D12500 + *b*	REAL	%MD*f*.106	DWORD	Auto Stops
F*f*:107	12503 + *b*	D12502 + *b*	REAL	%MD*f*.107	DWORD	Auto Stops
F*f*:108	12505 + *b*	D12504 + *b*	REAL	%MD*f*.108	DWORD	Auto Stops
F*f*:110	12509 + *b*	D12508 + *b*	REAL	%MD*f*.110	REAL	Closed Loop Halt Deceleration
F*f*:111	12511 + *b*	D12510 + *b*	REAL	%MD*f*.111	REAL	Open Loop Halt Ramp
F*f*:112	12513 + *b*	D12512 + *b*	REAL	%MD*f*.112	DINT	Halt Group Number
Simulator						
F*f*:116	12521 + *b*	D12520 + *b*	REAL	%MD*f*.116	DWORD	Simulator Configuration Register
F*f*:117	12523 + *b*	D12522 + *b*	REAL	%MD*f*.117	REAL	System Gain
F*f*:118	12525 + *b*	D12524 + *b*	REAL	%MD*f*.118	REAL	Natural Frequency
F*f*:119	12527 + *b*	D12526 + *b*	REAL	%MD*f*.119	REAL	Damping Factor
F*f*:120	12529 + *b*	D12528 + *b*	REAL	%MD*f*.120	REAL	Positive Physical Limit
F*f*:121	12531 + *b*	D12530 + *b*	REAL	%MD*f*.121	REAL	Negative Physical Limit
F*f*:122	12533 + *b*	D12532 + *b*	REAL	%MD*f*.122	REAL	Output Deadband
F*f*:123	12535 + *b*	D12534 + *b*	REAL	%MD*f*.123	REAL	Output Null
F*f*:125	12539 + *b*	D12538 + *b*	REAL	%MD*f*.125	REAL	Weight
F*f*:126	12541 + *b*	D12540 + *b*	REAL	%MD*f*.126	REAL	Maximum Force
F*f*:127	12543 + *b*	D12542 + *b*	REAL	%MD*f*.127	REAL	Maximum Compression
Position/Velocity Modeling						
F*f*:148	12585 + *b*	D12584 + *b*	REAL	%MD*f*.148	REAL	Model Response
F*f*:149	12587 + *b*	D12586 + *b*	REAL	%MD*f*.149	DINT	Model Order
F*f*:150	12589 + *b*	D12588 + *b*	REAL	%MD*f*.150	REAL	Model Gain Positive

F*f*:151	12591 + *b*	D12590 + *b*	REAL	%MD*f*.151	REAL	Model Gain Negative
F*f*:152	12593 + *b*	D12592 + *b*	REAL	%MD*f*.152	REAL	Model Time Constant, Model Natural Frequency
F*f*:153	12595 + *b*	D12594 + *b*	REAL	%MD*f*.153	REAL	Model Damping Factor
Pressure/Force Modeling						
F*f*:160	12609 + *b*	D12608 + *b*	REAL	%MD*f*.160	DINT	Model Order
F*f*:161	12611 + *b*	D12610 + *b*	REAL	%MD*f*.161	REAL	Model Gain Pressure/Force
F*f*:162	12613 + *b*	D12612 + *b*	REAL	%MD*f*.162	REAL	Model Time Constant, Model Natural Frequency
F*f*:163	12615 + *b*	D12614 + *b*	REAL	%MD*f*.163	REAL	Model Damping Factor
Display Units						
F*f*:166	12621 + *b*	D12620 + *b*	REAL	%MD*f*.166	DINT	Primary Display Units
F*f*:167	12623 + *b*	D12622 + *b*	DWORD	%MD*f*.167	DWORD	Primary Custom Units
F*f*:168	12625 + *b*	D12624 + *b*	REAL	%MD*f*.168	DINT	Secondary Display Units
F*f*:169	12627 + *b*	D12626 + *b*	DWORD	%MD*f*.169	DWORD	Secondary Custom Units

See Also

RMC150 Register Map

Back

RMC150 Registers, File 40: Command Area Registers

All Command Area Registers are **Write Only**.

n = axis

b = 10 x *n*

c = 20 x *n*

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Axis *n* Command						
F40:*b*+0	20481 + *c*	D20480 + *c*	REAL	%MD40.*b*+0	REAL	Axis *n* Command
F40:*b*+1	20483 + *c*	D20482 + *c*	REAL	%MD40.*b*+1	REAL	Axis *n* Command Parameter 1
F40:*b*+2	20485 + *c*	D20484 + *c*	REAL	%MD40.*b*+2	REAL	Axis *n* Command Parameter 2
F40:*b*+3	20487 + *c*	D20486 + *c*	REAL	%MD40.*b*+3	REAL	Axis *n* Command Parameter 3
F40:*b*+4	20489 + *c*	D20488 + *c*	REAL	%MD40.*b*+4	REAL	Axis *n* Command Parameter 4

F40:*b*+5	20491 + *c*	D20490 + *c*	REAL	%MD40.*b*+5	REAL	Axis *n* Command Parameter 5
F40:*b*+6	20493 + *c*	D20492 + *c*	REAL	%MD40.*b*+6	REAL	Axis *n* Command Parameter 6
F40:*b*+7	20495 + *c*	D20494 + *c*	REAL	%MD40.*b*+7	REAL	Axis *n* Command Parameter 7
F40:*b*+8	20497 + *c*	D20496 + *c*	REAL	%MD40.*b*+8	REAL	Axis *n* Command Parameter 8
F40:*b*+9	20499 + *c*	D20498 + *c*	REAL	%MD40.*b*+9	REAL	Axis *n* Command Parameter 9

Note:
For the Modbus and FINS protocols, the command registers are duplicated in a lower address area for PLCs that cannot access a wide range of registers. See the Modbus Addressing and FINS Addressing topics for details.

See Also
RMC150 Register Map

Back

RMC150 Registers, Files 41-42: Indirect Data Map Registers

Use the Indirect Data Map Editor to edit the Indirect Data Map. The RMC150 Indirect Data Map includes 256 registers.

The **Indirect Data Map** registers are the registers that should be accessed at runtime by a PLC when reading or writing the **Indirect Data**. The **Indirect Data Map Definition** contains the *addresses* of the referenced registers, not the *data*. Reading or writing to these registers will not access the Indirect Data via a PLC.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
Indirect Data Map							
F42:*n*	21505 + *b*	D21504 + *b*	*	%MD42.*n*	*	*	Indirect Data Value *n*
Indirect Data Map Definition							
F41:*n*	20993 + *b*	D20992 + *b*	REAL	%MD41.*n*	DINT	Read/Write	Indirect Data Map Definition Entry *n*

*The access and data types of the Indirect Data Values registers are determined by the mapped registers.

Note:
For the Modbus and FINS protocols, the Indirect Data and Indirect Data Map registers are duplicated in a lower address area for PLCs that cannot access a wide range of registers. See the Modbus Addressing and FINS Addressing topics for details.

See Also

RMC150 Register Map

Back

RMC150 Registers, File 43: Axis Definitions

The Axis Definitions are not intended to be directly accessed by the user. The preferred method of changing the axis definitions is to use the Axis Definitions dialog.

For highly advanced users, see the Axis Definition Registers topic for more details.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
F43:0-63	22017-22143	D22016-D22142	DWORD	%MD43.0-63	DWORD	Current Axis Definitions (Read-Only)
F43:64-127	22145-22271	D22144-D22270	DWORD	%MD143.64-127	DWORD	Requested Axis Definitions

The Current Axis Definitions and the Requested Axis Definitions will generally be the same except in two cases:

(1) The user has written to the requested block and intends to do a warm restart or burn to flash and do a cold restart, or

(2) The requested axis definitions found on startup are invalid for the current hardware configuration; in this case the Current Axis Definitions will be the default for the current hardware configuration.

See Also

RMC150 Register Map

Back

RMC150 Registers, File 44: Controller Status/Parameters

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
F44:0	22529	D22528	REAL	%MD44.0	REAL	Read Only	Loop Time, Set (sec)
F44:1	22531	D22530	REAL	%MD44.1	REAL	none	Loop Time, Requested (sec)
F44:2	22533	D22532	REAL	%MD44.2	REAL	Read Only	Loop Time Used, Last (sec)
F44:3	22535	D22534	REAL	%MD44.3	REAL	Read/Write	Loop Time Used, Maximum (sec)
F44:4	22537	D22536	REAL	%MD44.4	DINT	Read/Write	Number of Tasks Allocated 0-10 (default = 2)
F44:5	22539	D22538	REAL	%MD44.5	DINT	Read/Write	Program Triggers Task Enabled?

							0=false, 1= true (default = 1)
F44:6	22541	D22540	REAL	%MD44.6	DWORD	Read/Write	Stop All Tasks on Any Axis Halt? 0=false, 1= true (default = 1)
F44:7	22543	D22542	REAL	%MD44.7	DWORD	Read Only	Controller Status Bits
F44:8	22545	D22544	REAL	%MD44.8	DINT	Read/Write	Startup Mode 0=PROGRAM, 1-RUN (default = 0)
F44:9	22547	D22546	REAL	%MD44.9	DINT	Read/Write	RUN/PROGRAM Input 0 = None 1 = %IX1.0 (CPU input 0) 2 = %IX1.1 (CPU input 1)
F44:10	22549	D22548	REAL	%MD44.10	REAL	Read Only	Time Gear Master
F44:11	22551	D22550	DINT	%MD44.11	DINT	Read Only	Time, 16th Milliseconds
F44:12	22553	D22552	DINT	%MD44.12	DINT	Read Only	Time, Seconds
F44:13	22555	D22554	DINT	%MD44.13	DINT	Read Only	Time, Milliseconds
F44:14	22557	D22556	DINT	%MD44.14	DINT	Read Only	Time, Microseconds
F44:15	22559	D22558	DINT	%MD44.15	DINT	Read Only	Time, Nanoseconds
F44:16-31	22561-22591	D22560-D22590	DWORD	%MD44.16-31	DWORD	Read/Write	Controller Name Each 32-bit register holds 2 UTF-16 characters. The MSW is the first and the LSW is the second. For example, 0x004D0061 is 'Ma'.
F44:32	22593	D22592	DWORD	%MD44.32	DWORD	Read/Write	Controller Config Bits Bit 0: High Control Loop Utilization (1=enabled, 0=disabled) Bits 1-31: Reserved
F44:33	22595	D22594	DINT	%MD44.33	DINT	Read Only	Time, Loop Ticks
F44:34	22597	D22596	REAL	%MD44.34	REAL	Read Only	Loop Time Used, Total (µs)
F44:35	22599	D22598	REAL	%MD44.35	REAL	Read Only	Loop Time Used, Axes (µs)
F44:36	22601	D22600	REAL	%MD44.36	REAL	Read Only	Loop Time Used, Programs (µs)
F44:37	22603	D22602	REAL	%MD44.37	REAL	Read Only	Loop Time Used, Plots (µs)

F44:38	22605	D22604	REAL	%MD44.38	REAL	Read Only	Loop Time Used, Comm (µs)
F44:39	22607	D22606	REAL	%MD44.39	REAL	Read Only	Loop Time Used, Overhead (µs)

See Also

RMC150 Register Map

Back

RMC150 Registers, File 45: Communication Configuration

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
F45:0-3	23041-23047	D23040-D23046	-	%MD45.0-3	-	-	Reserved
F45:4	23049	D23048	REAL	%MD45.4	DINT	Read/Write	PROFIBUS: Station Address Indicates the current PROFIBUS station address for this controller. Valid values are 1 to 126. The default is 126, which is reserved for unaddressed PROFIBUS devices. **NOTE:** Only RMC150E controllers with the PROFIBUS communication module installed have this register.
F45:5	23051	D23050	REAL	%MD45.5	DINT	Read/Write	PROFIBUS: Lock Address 0=unlocked, 1=locked This register controls whether the PROFIBUS Station Address can be changed over PROFIBUS by a Class 2 master. Notice that RMCTools can always change the station address, regardless of the

							value of this register. Also, setting the station address over RMCTools will set this register to zero (0). **NOTE:** Only RMC150E controllers with the PROFIBUS communication module installed have this register.
F45:6	23053	D23052	DWORD	%MD45.6	DWORD	Read Only	PROFIBUS Connection Status
F45:7	23055	D23054	DINT	%MD45.7	-	-	Reserved
F45:8	23057	D23056	DWORD	%MD45.8	DWORD	Read Only	MAC Address (bytes 0-3) big Endian, (for 00-50-A0-B1-23-45, this would be 0x0050A0B1)
F45:9	23059	D23058	DWORD	%MD45.9	DWORD	Read Only	MAC Address (bytes 4-5) big Endian, in upper 16 bits (for 00-50-A0-B1-23-45, this would be 0x23450000)
F45:10	23061	D23060	DWORD	%MD45.10	DWORD	Read Only	Ethernet Status Bits
F45:11	23063	D23062	DWORD	%MD45.11	DWORD	Read/Write	Ethernet Configuration Bits
F45:12	23065	D23064	DWORD	%MD45.12	DWORD	Read/Write	IP Address
F45:13	23067	D23066	DWORD	%MD45.13	DWORD	Read/Write	Subnet Mask (0=use standard network masks)
F45:14	23069	D23068	DWORD	%MD45.14	DWORD	Read/Write	Default Gateway (0=none)
F45:15	23071	D23070	DWORD	%MD45.15	DWORD	Read Only	DHCP (or BOOTP) server IP Address
F45:16	23073	D23072	DINT	%MD45.16	DINT	Read Only	DHCP Lease Start (seconds since powerup)
F45:17	23075	D23074	DINT	%MD45.17	DINT	Read Only	DHCP Lease End (seconds since powerup)
F45:18	23077	D23076	DINT	%MD21.18	DINT	Read/Write	EtherNet/IP Class 1 TTL (time-to-live)

F45:19	23079	D23078	DINT	%MD245.19	DINT	Read/Write	EtherNet/IP Class 1 Start of Multicast Address Block
F45:20	23081	D23080	DINT	%MD45.20	DINT	Read/Write	EtherNet/IP Class 1 Size of Multicast Address Block
F45:21	23083	D23082	DWORD	%MD45.21	DWORD	Read/Write	Ethernet Outgoing Cyclic I/O Data Source
F45:22	23085	D23084	DWORD	%MD45.22	DWORD	Read/Write	Ethernet Incoming Cyclic I/O Data Destination
F45:23	23087	D23086	DWORD	%MD45.23	DWORD	Read Only	I/O Connection Status
F45:24	23089	D23088	DINT	%MD45.24	DINT	Read Only	I/O Connection PLC Status
F45:25	23091	D23090	DINT	%MD45.25	DINT	Read/Write	Ethernet App Flags **Bits 0-3 - I/O Data Mode** 0=Use a Sync Register 1=Do not use a Sync Register Applies to both EtherNet/IP and PROFINET. **Bits 4 - 5 - PROFINET Byte Order** 0=MSB first 1=LSB first Applies to PROFINET.
F45:26-55	23093-23151	D23092-D23150	-	%MD45.26-55	-	-	Reserved
F45:56-115	23153-23271	D23152-D23270	DWORD	%MD45.56-115	DWORD	Read/Write	PROFINET Device Name Each register in this range holds four characters of the PROFINET device name.
F45:116	23273	D23272	DWORD	%MD45.116	DWORD	Read/Write	PROFINET Custom Data Record 1000 Address
F45:117	23275	D23274	DWORD	%MD45.117	DWORD	Read/Write	PROFINET Custom Data Record 1001 Address

F45:118	23277	D23276	DWORD	%MD45.118	DWORD	Read/Write	PROFINET Custom Data Record 1002 Address
F45:119	23279	D23278	DWORD	%MD45.119	DWORD	Read/Write	PROFINET Custom Data Record 1003 Address

See Also

RMC150 Register Map

Back

RMC150 Registers, File 46: Event Log Configuration

This file is used by RMCTools to configure Event Log filtering. It should not be edited directly. Instead use the Event Log Properties to change these settings from within RMCTools.

See Also

RMC150 Register Map

Back

RMC150 Registers, File 47: Discrete I/O

All Discrete I/O registers are **Read/Write**. Each output or input is one bit. The inputs or outputs start at bit 0 in each register.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types.

n = RMC slot (0-5)

b = 2 x *n*

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Tag Name	Register Name
F47:0+*n*	24065 + *b*	D24064 + *b*	DWORD	%MD47.0+*n*	DWORD	_DIO.OutState[*n*]	Outputs for Slot *n*
F47:6+*n*	24077 + *b*	D24076 + *b*	DWORD	%MD47.6+*n*	DWORD	_DIO.InState[*n*]	Inputs for Slot *n*
F47:12+*n*	24089 + *b*	D24088 + *b*	DWORD	%MD47.12+*n*	DWORD	_DIO.OffInProgram[*n*]	Output to Off in PROGRAM mode for slot *n*
F47:18+*n*	24101 + *b*	D24100 + *b*	DWORD	%MD47.18+*n*	DWORD	_DIO.OnInProgram[*n*]	Output to On in PROGRAM

							mode for slot *n*
F47:24+*n*	24113 + *b*	D24112 + *b*	DWORD	%MD47.24+*n*	DWORD	_DIO.OffInFault[*n*]	Output to Off in FAULT mode for Slot *n*
F47:30+*n*	24125 + *b*	D24124 + *b*	DWORD	%MD47.30+*n*	DWORD	_DIO.OnInFault[*n*]	Output to On in FAULT mode for Slot *n*
F47:36+*n*	24137 + *b*	D24136 + *b*	DWORD	%MD47.36+*n*	DWORD	_DIO.OutForcedOFF[*n*]	Force Off for Outputs in Slots *n*
F47:42+*n*	24149 + *b*	D24148 + *b*	DWORD	%MD47.42+*n*	DWORD	_DIO.OutForcedON[*n*]	Force On for Outputs in Slots *n*
F47:48+*n*	24161 + *b*	D24160 + *b*	DWORD	%MD47.48+*n*	DWORD	_DIO.InForcedOFF[*n*]	Force Off for Inputs in Slots *n*
F47:54+*n*	24173 + *b*	D24172 + *b*	DWORD	%MD47.54+*n*	DWORD	_DIO.InForcedON[*n*]	Force On for Inputs in Slots *n*
F47:60+*n*	24195 + *b*	D24184 + *b*	DWORD	%MD47.60+*n*	DWORD	_DIO.Type[*n*]	Type for configurable I/O in Slots *n*

%IX and %QX Addressing

In RMCTools, the discrete I/O addresses are shown in the IEC 61131-3 format:

Inputs = %IX*slot.n*

Outputs = %QX*slot.n*

where

slot = numbering of slot starting with 0 for the left-most module in the RMC150.

n = the number of the input or output on that module, starting with zero

Examples:

%QX0.5 is output 5 in slot 0

%IX5.0 is input 0 in slot 5

See Also

RMC150 Register Map

Back

RMC150 Registers, File 48: Task Status/Configuration

The Task Status can be viewed in the Task Monitor.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types.

n = task number

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
F48:0 + 16x*n*	24577 + 32x*n*	D24576 + 32x*n*	REAL	%MD48.0+16x*n*	DWORD	Read Only	Task *n* Task Status
F48:1 + 16x*n*	24579 + 32x*n*	D24578 + 32x*n*	REAL	%MD48.1+16x*n*	DWORD	Read Only	Task *n* Current Program/Step
F48:2 + 16x*n*	24581 + 32x*n*	D24580 + 32x*n*	REAL	%MD48.2+16x*n*	DINT	Read/Write	Task *n* Current Axis
F48:3 + 16x*n*	24583 + 32x*n*	D24582 + 32x*n*	REAL	%MD48.3+16x*n*	DINT	Read Only	Task *n* Current Program
F48:4 + 16x*n*	24585 + 32x*n*	D24584 + *32xn*	REAL	%MD48.4+16x*n*	DINT	Read Only	Task *n* Current Step

See Also
RMC150 Register Map

Back

RMC150 Registers, Files 56-59, 72-75, 88: Variables Registers

All variable registers are **Read/Write**.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading variables defined as DWORD or DINT data types.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Register Name
Variables - Current Values						
F56:0-255	28673-29183	D28672-D29182	*	%MD56.0-255	*	Variables 0-255 - Current Values
F57:0-255	29185-29695	D29184-D29694	*	%MD57.0-255	*	Variables 256-511 - Current Values
F58:0-255	29697-30207	D29696-D30206	*	%MD58.0-255	*	Variables 512-767 - Current Values
F59:0-255	30209-30719	D30208-D30718	*	%MD59.0-255	*	Variables 768-1023 - Current Values

Variables - Initial Values						
F72:0-255	36865-37375	E0_04096-E0_04606	*	%MD72.0-255	*	Variables 0-255 - Initial Values
F73:0-255	37377-37887	E0_04608-E0_05118	*	%MD73.0-255	*	Variables 256-511 - Initial Values
F74:0-255	37889-38399	E0_05120-E0_05630	*	%MD74.0-255	*	Variables 512-767 - Initial Values
F75:0-255	38401-38911	E0_05632-E0_06142	*	%MD75.0-255	*	Variables 768-1023 - Initial Values
Variables - Attributes						
F88:0-255	45057-45567	E0_12288-E0_12798	DWORD	%MD88.0-255	DWORD	Variables 0-1023 - Attributes (4 variables per register)

* The data types of the variables are specified by the user when defining a variable in the Variable Table.

Allen-Bradley DF1 and CSP

Where allowed by the host controller's communications, all the current values of the variables can also be addressed as F56:*n*, up to *n* = 1023.

Modbus/TCP and /RTU

The address of the current value of variable *n* is 28673 + 2 x *n*.

FINS

The address of the current value of variable *n* is D28672 + 2 x *n*.

Note:
For the Modbus and FINS protocols, the command registers are duplicated in a lower address area for PLCs that cannot access a wide range of registers. See the Modbus Addressing and FINS Addressing topics for details.

Tag Names

Tag names are given to registers to be used when accessing the registers from within RMCTools. Tag names make the project easy to read.

The variables can given a user-defined tag name in the Variable Table Editor, which is the preferred method of referencing variables from within RMCTools. The user-defined variable name references the variable's Current Value.

Otherwise, each variable has a tag name as follows:

_VarTbl.CurVal[n] - Current Value of variable n.

_VarTbl.Initial[n] - Initial value of variable n.

See Also

RMC150 Register Map | Variable Attributes

Back

RMC150 Registers, File 94: Image Upload/Download Area

See the Controller Image Upload/Download topic for details.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
F94:0	48129	E0_15360	DINT	%MD94.0	DINT	Write Only	**Image Area Command**
F94:1	48131	E0_15362	DINT	%MD94.1	DINT	Read Only	**Image Area State**
F94:2	48133	E0_15364	DINT	%MD94.2	DINT	Read Only	**Image Size**
F94:3	48135	E0_15366	DINT	%MD94.3	DINT	Read/Write	**Current Index**
F94:4 : F94:4095	48137-48639 (252 items)	E0_15368 - E0_15870 (252 items)	DINT[252] or DINT[4092]	%MD94.4 : %MD94.4095	DINT[252] or DINT[4092]	Read/Write	**Image Data**

See Also

RMC150 Register Map | Controller Image Upload/Download

Back

RMC150 Registers, File 95: Plot Layout

The following files contain the Plot Layout registers.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading data of DWORD or DINT external data types.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
F95:0	48641	E0_15872	REAL	%MD95.0	UDINT	Read/Write	Current Plot Layout This register indicates the current plot allocation. Bit / Description: 0-7 Plots (1-8); 8-15 Sample Sets per Plot (1-16); 16-23 Reserved
F95:1	48643	E0_15874	REAL	%MD95.1	UDINT	Read Only	Maximum Plots (8) This read-only value indicates how many plots can be defined at once.

F95:2	48645	E0_15876	REAL	%MD95.2	UDINT	Read Only	Maximum Data items (16) This read-only value indicates how many data items can be captured per plot.
F95:3	48647	E0_15878	REAL	%MD95.3	UDINT	Read Only	Maximum Samples at Once (128) This read-only value indicates how many total different samples can be allocated across all plots.
F95:4	48649	E0_15880	REAL	%MD95.4	UDINT	Read Only	Maximum Elements (12,582,912) This read-only value indicates how many total data points can be captured. This is spread out among the number of allocated plots and samples per plot.
F95:5	48651	E0_15882	-	%MD95.5	UDINT	-	Reserved
F95:6	48653	E0_15884	-	%MD95.6	UDINT	-	Reserved
F95:7	48655	E0_15886	-	%MD95.7	UDINT	-	Reserved

See Also

RMC150 Register Map

Back

RMC150 Registers, Files 96-103: Plot Status/Configuration Registers

The following files contain the Plot Configuration registers. See the Reading Plots with a Host Controller topic for details on how some of them can be used.

f = 96 + *n*

b = 512 x *n*

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
Plot *n*							
F*f*:0	49153 + *b*	E0_16384 + *b*	REAL	%MD*f*.0	DWORD	not directly	Plot Flags These bits should not be accessed directly. 0 Reserved (Write Only) 1 Trigger (Write only)

							2 Rearm (Write Only) 3 Read Active (Read Only) 4 Trigger Enabled (Read Only)
F*f*:1	49155 + *b*	E0_16386 + *b*	REAL	%MD*f*.1	DINT	Read/Write	Plot Samples (e.g. 1000)
F*f*:2	49157 + *b*	E0_16388 + *b*	REAL	%MD*f*.2	REAL	Read/Write	Plot Sample Period (seconds)
F*f*:3	49159 + *b*	E0_16390 + *b*	REAL	%MD*f*.3	DINT	Read/Write	Plot Axis Owner 0-15, -1 = none
F*f*:4	49161 + *b*	E0_16392 + *b*	REAL	%MD*f*.4	REAL	Read/Write	Plot Trigger Position %, 0-100, -1 = auto rearm
F*f*:5	49163 + *b*	E0_16394 + *b*	REAL	%MD*f*.5	DWORD	Read/Write	Plot Trigger Type 0-7Trigger Type (0=none, 1=motion command) 8-23 Depends on trigger type. Motion Commands: Axis 0-15. If all bits are zero, then use Axis Owner
F*f*:6	49165 + *b*	E0_16396 + *b*	REAL	%MD*f*.6	-	-	Reserved
F*f*:7	49167 + *b*	E0_16398 + *b*	REAL	%MD*f*.7	-	-	Reserved
F*f*:8	49169 + *b*	E0_16400 + *b*	REAL	%MD*f*.8	DINT	Read Only	Plot ID
F*f*:9	49171 + *b*	E0_16402 + *b*	REAL	%MD*f*.9	DINT	Read Only	Plot State 0 = not triggered, 1 = capturing, 2 = complete
F*f*:10	49173 + *b*	E0_16404 + *b*	REAL	%MD*f*.10	DINT	Read Only	Plot Captured Samples Number of plot samples captured. Only applies for Plot State > 0.
F*f*:11	49175 + *b*	E0_16406 + *b*	REAL	%MD*f*.11	DINT	Read Only	Plot Sample 0 Time Time that first plot sample was captured. In control loops since controller startup (low 24 bits). Only applies for Plot State > 0.
F*f*:12	49177 + *b*	E0_16408 + *b*	REAL	%MD*f*.12	DINT	Read Only	Plot Trigger Time Time that plot trigger occurred. In control loops since controller startup

							(low 24 bits). Only applies for Plot State > 0.
F*f*:13	49179 + *b*	E0_16410 + *b*	REAL	%MD*f*.13	DINT	Read Only	Plot Trigger Index Index of the plot sample at which the plot trigger occurred. Only applies for Plot State > 0.
F*f*:14-15	49181 + *b* 49183 + *b*	E0_16412 + *b* E0_16414 + *b*	-	%MD*f*.14-15	-	-	Reserved
F*f*:16-31	49185 + *b* 49215 + *b*	E0_16416 + *b* E0_16446 + *b*	REAL	%MD*f*.16-31	DWORD	Read/Write	Plot Data Sets 0-15 Addresses Files %MD*n*:16-31 contain the Addresses for plot Data Sets 0-15. Bits 0-11 Element Bits 12-23File

See Also

RMC150 Register Map

Back

RMC150 Registers, Files 104-111: Dynamic Plot Upload Area Registers

The following files contain the Dynamic Plot Upload Area registers. See the Reading Plots with a Host Controller topic for details on how to use them.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading data of DWORD or DINT external data types.

Note:
When communicating via a protocol that uses DF1 addressing, the Plot Data can be accessed with registers 5-4095, if the host controller allows it. Other protocols can only address Plot Data registers 5-255, as indicated below.

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
Plot 0							
F104:0	53249	E0_20480	DINT	%MD104.0	UDINT	Read/Write	Plot 0 Upload Mode/Status
F104:1	53251	E0_20482	DINT	%MD104.1	UDINT	Read/Write	Plot 0 Requested Read Samples
F104:2	53253	E0_20484	DINT	%MD104.2	UDINT	Read/Write	Plot 0 Current Index
F104:3	53255	E0_20486	DINT	%MD104.3	UDINT	Read Only	Plot 0 ID

F104:4	53257	E0_20488	DINT	%MD104.4	UDINT	Read Only	Plot 0 Samples Uploaded
F104:5-255	53259-53759	E0_20490-E0_20990	*	%MD104.5-255	*	Read Only	Plot 0 Data
Plot 1							
F105:0	53761	E0_20992	DINT	%MD105.0	UDINT	Read/Write	Plot 1 Upload Mode/Status
F105:1	53763	E0_20994	DINT	%MD105.1	UDINT	Read/Write	Plot 1 Requested Read Samples
F105:2	53765	E0_20996	DINT	%MD105.2	UDINT	Read/Write	Plot 1 Current Index
F105:3	53767	E0_20998	DINT	%MD105.3	UDINT	Read Only	Plot 1 ID
F105:4	53769	E0_21000	DINT	%MD105.4	UDINT	Read Only	Plot 1 Samples Uploaded
F105:5-255	53771-54271	E0_21002-E0_21502	*	%MD105.5-255	*	Read Only	Plot 1 Data
Plot 2							
F106:0	54273	E0_21504	DINT	%MD106.0	UDINT	Read/Write	Plot 2 Upload Mode/Status
F106:1	54275	E0_21506	DINT	%MD106.1	UDINT	Read/Write	Plot 2 Requested Read Samples
F106:2	54277	E0_21508	DINT	%MD106.2	UDINT	Read/Write	Plot 2 Current Index
F106:3	54279	E0_21510	DINT	%MD106.3	UDINT	Read Only	Plot 2 ID
F106:4	54281	E0_21512	DINT	%MD106.4	UDINT	Read Only	Plot 2 Samples Uploaded
F106:5-255	54283-54783	E0_21514-E0_22014	*	%MD106.5-255	*	Read Only	Plot 2 Data
Plot 3							
F107:0	54785	E0_22016	DINT	%MD107.0	UDINT	Read/Write	Plot 3 Upload Mode/Status
F107:1	54787	E0_22018	DINT	%MD107.1	UDINT	Read/Write	Plot 3 Requested Read Samples
F107:2	54789	E0_22020	DINT	%MD107.2	UDINT	Read/Write	Plot 3 Current Index
F107:3	54791	E0_22022	DINT	%MD107.3	UDINT	Read Only	Plot 3 ID
F107:4	54793	E0_22024	DINT	%MD107.4	UDINT	Read Only	Plot 3 Samples Uploaded
F107:5-255	54795-55295	E0_22026-E0_22526	*	%MD107.5-255	*	Read Only	Plot 3 Data
Plot 4							
F108:0	55297	E0_22528	DINT	%MD108.0	UDINT	Read/Write	Plot 4 Upload Mode/Status
F108:1	55299	E0_22530	DINT	%MD108.1	UDINT	Read/Write	Plot 4 Requested Read Samples
F108:2	55301	E0_22532	DINT	%MD108.2	UDINT	Read/Write	Plot 4 Current Index
F108:3	55303	E0_22534	DINT	%MD108.3	UDINT	Read Only	Plot 4 ID
F108:4	55305	E0_22536	DINT	%MD108.4	UDINT	Read Only	Plot 4 Samples Uploaded
F108:5-255	55307-55807	E0_22538-E0_23038	*	%MD108.5-255	*	Read Only	Plot 4 Data
Plot 5							

F109:0	55809	E0_23040	DINT	%MD109.0	UDINT	Read/Write	Plot 5 Upload Mode/Status
F109:1	55811	E0_23042	DINT	%MD109.1	UDINT	Read/Write	Plot 5 Requested Read Samples
F109:2	55813	E0_23044	DINT	%MD109.2	UDINT	Read/Write	Plot 5 Current Index
F109:3	55815	E0_23046	DINT	%MD109.3	UDINT	Read Only	Plot 5 ID
F109:4	55817	E0_23048	DINT	%MD109.4	UDINT	Read Only	Plot 5 Samples Uploaded
F109:5-255	55819-56319	E0_23050-E0_23550	*	%MD109.5-255	*	Read Only	Plot 5 Data
Plot 6							
F110:0	56321	E0_23552	DINT	%MD110.0	UDINT	Read/Write	Plot 6 Upload Mode/Status
F110:1	56323	E0_23554	DINT	%MD110.1	UDINT	Read/Write	Plot 6 Requested Read Samples
F110:2	56325	E0_23556	DINT	%MD110.2	UDINT	Read/Write	Plot 6 Current Index
F110:3	56327	E0_23558	DINT	%MD110.3	UDINT	Read Only	Plot 6 ID
F110:4	56329	E0_23560	DINT	%MD110.4	UDINT	Read Only	Plot 6 Samples Uploaded
F110:5-255	56331-56831	E0_23562-E0_24062	*	%MD110.5-255	*	Read Only	Plot 6 Data
Plot 7							
F111:0	56833	E0_24064	DINT	%MD111.0	UDINT	Read/Write	Plot 7 Upload Mode/Status
F111:1	56835	E0_24066	DINT	%MD111.1	UDINT	Read/Write	Plot 7 Requested Read Samples
F111:2	56837	E0_24068	DINT	%MD111.2	UDINT	Read/Write	Plot 7 Current Index
F111:3	56839	E0_24070	DINT	%MD111.3	UDINT	Read Only	Plot 7 ID
F111:4	56841	E0_24072	DINT	%MD111.4	UDINT	Read Only	Plot 7 Samples Uploaded
F111:5-255	56843-57343	E0_24074-E0_24574	*	%MD111.5-255	*	Read Only	Plot 7 Data

*The data types of the Plot Data are determined by the plotted registers.

See Also

RMC150 Register Map

Back

RMC150 Registers, Files 112-143: Static Plot Upload Area Registers

All Static Plot Upload Area Registers are **Read Only**. See the Reading Plots with a Host Controller topic for details on how to use them.

The data types of the plot data is determined by the plotted registers.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading plot data of DWORD or DINT data types.

Note:
When communicating via a protocol that uses DF1 addressing, samples 0-4095 from each plot can be addressed, if the host controller allows it. For example, F112:4095 addresses sample 4095 of plot 0, data set 0.
Other protocols can only address the first 256 samples, as indicated below.

AB DF1,CSP Address	Modbus TCP,RTU Address (max 256 registers)	FINS Address (max 256 registers)	Internal IEC Address	Register Name
Plot 0				
F112:0-4095	57345-57855	E0_24576-E0_25086	%MD112.0-4095	Plot 0, Data Set 0, Samples 0-4095
F113:0-4095	57857-58367	E0_25088-E0_25598	%MD113.0-4095	Plot 0, Data Set 1, Samples 0-4095
F114:0-4095	58369-58879	E0_25600-E0_26110	%MD114.0-4095	Plot 0, Data Set 2, Samples 0-4095
F115:0-4095	58881-59391	E0_26112-E0_26622	%MD115.0-4095	Plot 0, Data Set 3, Samples 0-4095
Plot 1				
F116:0-4095	59393-59903	E0_26624-E0_27134	%MD116.0-4095	Plot 1, Data Set 1, Samples 0-4095
F117:0-4095	59905-60415	E0_27136-E0_27646	%MD117.0-4095	Plot 1, Data Set 2, Samples 0-4095
F118:0-4095	60417-60927	E0_27648-E0_28158	%MD118.0-4095	Plot 1, Data Set 3, Samples 0-4095
F119:0-4095	60929-61439	E0_28160-E0_28670	%MD119.0-4095	Plot 1, Data Set 4, Samples 0-4095
Plot 2				
F120:0-4095	61441-61951	E0_28672-E0_29182	%MD120.0-4095	Plot 2, Data Set 1, Samples 0-4095
F121:0-4095	61953-62463	E0_29184-E0_29694	%MD121.0-4095	Plot 2, Data Set 2, Samples 0-4095
F122:0-4095	62465-62975	E0_29696-E0_30206	%MD122.0-4095	Plot 2, Data Set 3, Samples 0-4095
F123:0-4095	62977-63487	E0_30208-E0_30718	%MD123.0-4095	Plot 2, Data Set 4, Samples 0-4095
Plot 3				
F124:0-4095	63489-63999	E0_30720-E0_31230	%MD124.0-4095	Plot 3, Data Set 1, Samples 0-4095
F125:0-4095	64001-64511	E0_31232-E0_31742	%MD125.0-4095	Plot 3, Data Set 2, Samples 0-4095
F126:0-4095	64513-65023	E0_31744-E0_32254	%MD126.0-4095	Plot 3, Data Set 3, Samples 0-4095
F127:0-4095	65025-65535	E0_32256-E0_32766	%MD127.0-4095	Plot 3, Data Set 4, Samples 0-4095
Plot 4				

F128:0-4095	65537-66047	E1_00000-E1_00510	%MD128.0-4095	Plot 4, Data Set 1, Samples 0-4095
F129:0-4095	66049-66559	E1_00512-E1_01022	%MD129.0-4095	Plot 4, Data Set 2, Samples 0-4095
F130:0-4095	66561-67071	E1_01024-E1_01534	%MD130.0-4095	Plot 4, Data Set 3, Samples 0-4095
F131:0-4095	67073-67583	E1_01536-E1_02046	%MD131.0-4095	Plot 4, Data Set 4, Samples 0-4095
Plot 5				
F132:0-4095	67585-68095	E1_02048-E1_02558	%MD132.0-4095	Plot 5, Data Set 1, Samples 0-4095
F133:0-4095	68097-68607	E1_02560-E1_03070	%MD133.0-4095	Plot 5, Data Set 2, Samples 0-4095
F134:0-4095	68609-69119	E1_03072-E1_03582	%MD134.0-4095	Plot 5, Data Set 3, Samples 0-4095
F135:0-4095	69121-69631	E1_03584-E1_04094	%MD135.0-4095	Plot 5, Data Set 4, Samples 0-4095
Plot 6				
F136:0-4095	69633-70143	E1_04096-E1_04606	%MD136.0-4095	Plot 6, Data Set 1, Samples 0-4095
F137:0-4095	70145-70655	E1_04608-E1_05118	%MD137.0-4095	Plot 6, Data Set 2, Samples 0-4095
F138:0-4095	70657-71167	E1_05120-E1_05630	%MD138.0-4095	Plot 6, Data Set 3, Samples 0-4095
F139:0-4095	71169-71679	E1_05632-E1_06142	%MD139.0-4095	Plot 6, Data Set 4, Samples 0-4095
Plot 7				
F140:0-4095	71681-72191	E1_06144-E1_06654	%MD140.0-4095	Plot 7, Data Set 1, Samples 0-4095
F141:0-4095	72193-72703	E1_06656-E1_07166	%MD141.0-4095	Plot 7, Data Set 2, Samples 0-4095
F142:0-4095	72705-73215	E1_07168-E1_07678	%MD142.0-4095	Plot 7, Data Set 3, Samples 0-4095
F143:0-4095	73217-73727	E1_07680-E1_08190	%MD143.0-4095	Plot 7, Data Set 4, Samples 0-4095

See Also

RMC150 Register Map

Show All Back

RMC150 Registers, File 144-149: Slot Settings

These registers control the settings for the module in each slot.

Tip:
For the DF1/CSP addressing format, all 'F' type registers (32-bit floating point) can also be read as 'L' type (32-bit word) registers. This is very useful when reading registers with DWORD or DINT external data types.

Modules RMC150E, RMC151E, M, S, Q, A, G, H, DI/O

Slot *n* = 0-5

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
F144+*n*:0	73729 +512 x *n*	E1_08192 +512 x *n*	DINT	%MD144+*n*.0	DINT	Read Only	Slot *n* Module ID 0: None 16: RMC150E 17: RMC151E 38: Serial (not supported) 40: DI/O 41: PROFIBUS 42: ENET (not supported) 43: Modbus Plus (not supported) 45: Universal I/O (comm slot) 64: MDT (M) 66: Analog (H) 68: Analog Inputs (A) 69: Quadrature (Q) 70: SSI (S) 72: DI/O (D) 74: Analog (G) 76: Resolver (R) 77: Universal I/O (sensor slot) 80: Resolver (RW)
F144+*n*:1	73731 +512 x *n*	E1_08194 +512 x *n*	DWORD	%MD144+*n*.1	DWORD	Read Only	Slot *n* Module Rev Major * 256 + Minor

Module UI/O

Slot *n* = 0-5

AB DF1,CSP Address	Modbus TCP,RTU Address	FINS Address	External Data Type	Internal IEC Address	Internal Data Type	Access	Register Name
F144+*n*:0	73729 +512 x *n*	E1_08192 +512 x *n*	DINT	%MD144+*n*.0	DINT	Read Only	Slot *n* Module ID Module ID = 45 (comm slot) Module ID = 77 (sensor slot)

F144+*n*:1	73731 +512 x *n*	E1_08194 +512 x *n*	DWORD	%MD144+*n*.1	DWORD	Read Only	Slot *n* Module Rev Major * 256 + Minor
F144+*n*:2	73733 +512 x *n*	E1_08196 +512 x *n*	DWORD	%MD144+*n*.2	DWORD	Read Only	Board Revision 0x00MMmmLL, where MM is major, mm is minor, and LL is letter (00=A, 01=B, etc.)
F144+*n*:3	73735 +512 x *n*	E1_08198 +512 x *n*	DINT	%MD144+*n*.3	DINT	Read Only	Serial Number Last 6 digits of the serial number (yyqnnn)
F144+*n*:4	73737 +512 x *n*	E1_08200 +512 x *n*	DWORD	%MD144+*n*.4	DWORD	Read Only	FPGA Image Date/Time Bits 20-31: Year (4-digit year) Bits 16-19: Month (1=Jan, etc.) Bits 11-15: Day (1..31) Bits 6-10: Hour (0..23, 0=12am, 1=1am, etc.) Bits 0-5: Minute (00-59)
F144+*n*:16	73739 +512 x *n*	E1_08202 +512 x *n*	DWORD	%MD144+*n*.16	DWORD	Read/Write	Channel 0 Mode **Bits 0-3 - Channel Mode** 0 = Quadrature Axis Input 1 = SSI Axis Input 2 = SSI Register Input 3 = SSI Output **Bit 4 - SSI Master/Slave** [Applies only to SSI Output and SSI Register Input modes] 0 = Master - Clock is an output 1 = Slave - Clock is an input
F144+*n*:17	73741 +512 x *n*	E1_08204 +512 x *n*	DWORD	%MD144+*n*.17	DWORD	Read/Write	Channel 0 SSI Options Applies only to SSI Output and SSI Register Input modes] **Bit 0-5 - SSI Data Bits** Range: 8-32 **Bit 6 - SSI Encoding** 0=Binary, 1=Gray Code **Bit 8 - Terminate Inputs** [Applies to all SSI Output and SSI Register

							Input modes except SSI Output - Master.] SSI Register Input - Standard: Data Inputs SSI Register Input - Monitor: Clock/Data Inputs SSI Output - Slave: Clock Inputs **Bit 16-31 - SSI Clock Rate (kHz)** Specifies the clock rate in kHz. The actual rate will be determined by the available integer divisor (16500/SSIClockRatekHz -1). Initial supported values are 250, 500, and 971, with the default of 250 kHz.
F144+*n*:18	73743 +512 x *n*	E1_0820 6 +512 x *n*	DINT	%MD144+*n*.18	DINT	Read/Write	Channel 0 Wire Delay [Applies only to SSI Register Input modes] Time to delay (ns) from Clock to Data. This value will be converted to the nearest time delay that can be represented with [1..8] * [0..31] / 33MHz.
F144+*n*:19	73745 +512 x *n*	E1_0820 8 +512 x *n*	DWORD	%MD144+*n*.19	DWORD	Read/Write	Channel 0 Source/Dest [Applies only to SSI Output and SSI Register Input modes] For SSI Output mode, this is the address of the register to send out the SSI Output. Ignored if Echo is enabled. For SSI Register Input, this is the address of the register to save the value coming in on the SSI input.
F144+*n*:20	73747 +512 x *n*	E1_0821 0 +512 x *n*	DWORD	%MD144+*n*.20	DWORD	Read Only	Channel 0 SSI Status [Applies only to SSI Output and SSI Register Input modes] **Bit 0 - Clock Wire Fault** Wire Break for Clock inputs, Short for Clock outputs.

							Bit 1 - Data Wire Fault Wire Break for Data inputs, Short for Data outputs. **Bit 2 - No Master Detected - Two or more loops (SSI Slave modes only)** No master device is clocking SSI data--two or more loops in a row. **Bit 3 - No Slave Detected (SSI Register Input only)** Transducer or RMC not responding with data. **Bit 4 - No Master Detected - One or more loops (SSI Slave modes only)** This status register indicates that no clocked data was detected this loop time, even if this is the first loop this occurred. Notice that this can be expected behavior when the loop times are not synchronized.
F144+*n*:24	73749 +512 x *n*	E1_0821 2 +512 x *n*	DWORD	%MD144+*n*.24	DWORD	Read/Write	Channel 1 Mode **Bits 0-3 - Channel Mode** 0 = Quadrature Axis Input 1 = SSI Axis Input 2 = SSI Register Input 3 = SSI Output **Bit 4 - SSI Master/Slave** [Applies only to SSI Output and SSI Register Input modes] 0 = Master - Clock is an output 1 = Slave - Clock is an input
F144+*n*:25	73751 +512 x *n*	E1_0821 4 +512 x *n*	DWORD	%MD144+*n*.25	DWORD	Read/Write	Channel 1 SSI Options Applies only to SSI Output and SSI Register Input modes]

							Bit 0-5 - SSI Data Bits Range: 8-32 **Bit 6 - SSI Encoding** 0=Binary, 1=Gray Code **Bit 7 - Echo channel 0 SSI Input** [Applies only to SSI Output modes on channel 1.] **Bit 8 - Terminate Inputs** [Applies to all SSI Output and SSI Register Input modes except SSI Output - Master.] SSI Register Input - Standard: Data Inputs SSI Register Input - Monitor: Clock/Data Inputs SSI Output - Slave: Clock Inputs **Bit 16-31 - SSI Clock Rate (kHz)** Specifies the clock rate in kHz. The actual rate will be determined by the available integer divisor (16500/SSIClockRatekHz -1). Initial supported values are 250, 500, and 971, with the default of 250 kHz.
F144+*n*:26	73753 +512 x *n*	E1_0821 6 +512 x *n*	DINT	%MD144+*n*.26	DINT	Read/Write	Channel 1 Wire Delay [Applies only to SSI Register Input modes] Time to delay (ns) from Clock to Data. This value will be converted to the nearest time delay that can be represented with [1..8] * [0..31] / 33MHz.
F144+*n*:27	73755 +512 x *n*	E1_0821 8 +512 x *n*	DWORD	%MD144+*n*.27	DWORD	Read/Write	Channel 1 Source/Dest [Applies only to SSI Output and SSI Register Input modes] For SSI Output mode, this is the address of the register to send out the SSI Output. Ignored if Echo is enabled. For SSI Register Input, this is

							the address of the register to save the value coming in on the SSI input.
F144+*n*:28	73757 +512 x *n*	E1_0822 0 +512 x *n*	DWORD	%MD144+*n*.28	DWORD	Read Only	Channel 1 SSI Status [Applies only to SSI Output and SSI Register Input modes] **Bit 0 - Clock Wire Fault** Wire Break for Clock inputs, Short for Clock outputs. **Bit 1 - Data Wire Fault** Wire Break for Data inputs, Short for Data outputs. **Bit 2 - No Master Detected - Two or more loops (SSI Slave modes only)** No master device is clocking SSI data--two or more loops in a row. **Bit 3 - No Slave Detected (SSI Register Input only)** Transducer or RMC not responding with data. **Bit 4 - No Master Detected - One or more loops (SSI Slave modes only)** This status register indicates that no clocked data was detected this loop time, even if this is the first loop this occurred. Notice that this can be expected behavior when the loop times are not synchronized.

See Also
RMC150 Register Map

9.6. RMC200 Register Map

9.6.1. RMC200 Register Map

The RMC200 Register Map lists the addresses of all the registers in the RMC200. For certain communications types, you may need to use the register map to find addresses when setting up communications with the RMC200 from a host controller such as a PLC. When referencing registers from *within* the RMC, such as in user programs, you do not need to use register addresses. You can use tag names instead. See the Tags Overview topic for details.

Tip: The Address Maps in RMCTools provide any easy way to browse all the registers in the RMC, along with their addresses.

Protocols with Direct Access to All RMC200 Registers

The following communications types can access all registers in the RMC200 register map directly via the two-level addressing numbers used by the IEC addressing:

- RMCLink
- DMCP
- Mitsubishi Procedure Exist

Protocols with Limited Access to RMC200 Registers

For the following communications types, some RMC200 registers can be accessed directly, and others can be accessed indirectly via user-defined mapping and via the Indirect Data Map. Refer to the addressing topic for each protocol for details.

- Modbus Addressing
- FINS (Omron) Addressing
- DF1 (Allen-Bradley) Addressing
- PROFINET Data Records Addressing

Register Map

The RMC200 register map lists addresses in the IEC-61131 format. The registers are divided into the following sections:

File	Description
7	Controller Info
8	Indirect Data Map
12	Indirect Data Map Definitions
16	Command Area
18	Controller Status/Parameters
19	Event Log Configuration
20	Discrete I/O
21	Task Manager
22	Plot Manager
23	Image Upload/Download Area
24	Axis Definitions (active)
25	Axis Definitions (new)
26	Module Definitions
112	CPU Communication Status/Configuration
113-116	Address Maps
128-143	Base, Slot Status/Configuration
192-255	Task 0-63 Status/Configuration

256-383	Axis 0-127 Status Registers
384-511	Axis 0-127 Parameters
512-575	Plot 0-63 Status/Configuration
576-607	Dynamic Plot Upload Area
640-703	Static Plot Upload Area
1024	Variables - Current Values
1280	Variables - Initial Values
1536	Variables - Attributes

See Also

Register Map Overview | RMC75 Register Map | RMC150 Register Map

Show All Back

RMC200 Registers, File 7: Controller Info

All Controller Information registers are **Read Only**.

Internal IEC Address	Data Type	All Axes
%MD7.0	DINT	Product ID 4: RMC200 series
%MD7.1	DINT	State 1: Running Control Program

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, Files 8, 12: Indirect Data Map Registers

Use the Indirect Data Map Editor to edit the Indirect Data Map. The RMC200 Indirect Data Map includes 1024 registers.

The **Indirect Data Map** registers are the registers that should be accessed at runtime by a PLC when reading or writing the **Indirect Data**. The **Indirect Data Map Definition** contains the *addresses* of the referenced registers, not the *data*. Reading or writing to the **Indirect Data Map Definition** registers will not access the Indirect Data.

n = 0-1023

Internal IEC Address	Data Type	Access	All Axes
Indirect Data Map			
%MD8.*n*	*	*	Indirect Data Value
Indirect Data Map Definition			
%MD12.*n*	DINT	*	Indirect Data Map Definition

*The data types and access type of the Indirect Data Values registers are determined by the mapped registers.

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, File 16: Command Area

All Command Area Registers are **Write Only**.

Internal IEC Address	Data Type	All Axes
Axis 0 Command		
%MD16.0	REAL	Axis 0 Command
%MD16.1	REAL	Axis 0 Command Parameter 1
%MD16.2	REAL	Axis 0 Command Parameter 2
%MD16.3	REAL	Axis 0 Command Parameter 3
%MD16.4	REAL	Axis 0 Command Parameter 4
%MD16.5	REAL	Axis 0 Command Parameter 5
%MD16.6	REAL	Axis 0 Command Parameter 6
%MD16.7	REAL	Axis 0 Command Parameter 7
%MD16.8	REAL	Axis 0 Command Parameter 8
%MD16.9	REAL	Axis 0 Command Parameter 9
Axis *n* Command		
%MD16.*n* x 10 to *n* x 10 + 9	REAL	Axis *n* Command Registers

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Show All Back

RMC200 Registers, File 18: Controller Status/Parameters

Internal IEC Address	Data Type	Access	Register Name
%MD18.0	REAL	Read Only	Loop Time, Set (sec)
%MD18.1	REAL	Read/Write	Loop Time, Requested (sec)
%MD18.2	REAL	Read Only	Loop Time Used, Last (sec)
%MD18.3	REAL	Read Only	Loop Time Used, Maximum (sec)
%MD18.4	REAL	Read Only	Loop Time Used, Minimum (sec)
%MD18.5	DWORD	Read/Write	Controller Config Bits

			Bit 0: High Control Loop Utilization (1=enabled, 0=disabled) Bits 1-31: Reserved
%MD18.7	DWORD	Read Only	Controller Status Bits
%MD18.8	DINT	Read Only	Flash Update State Indicates whether a flash update is in progress. The following values are defined: 0: not in progress 1: in progress (but no estimate on percentage) 2: reserved for future progress indicator. The client should treat all non-zero values as in progress.
%MD18.9	DINT	Read/Write	Reset Command
%MD18.10	REAL	Read Only	Time Gear Master
%MD18.11	DINT	Read Only	Time, 16th Milliseconds
%MD18.12	DINT	Read Only	Time, Seconds
%MD18.13	DINT	Read Only	Controller Image Command State Indicates the status of the most recently issued Save/Restore Controller Image command. This is also described in the Save Controller Image (120) and Restore Controller Image (121) topics. 0 = No action requested. This will be the value after startup, including after an image is successfully restored and the controller is restarted. 10 = Building or saving controller image. 11 = Controller image saved successfully. 20 = Loading or applying controller image. 21 = Image successfully restored to controller. Controller will hold this state for 500 ms, then automatically restart. 30 = Unable to start command since an SD card exclusive access session was active. 31 = Unable to start command since another SD card operation was in progress. 40 = Save image failed due to an internal error. 41 = Save image failed due to an SD card write error. 42 = Save image failed because no SD card is installed. 43 = Save image failed because the installed SD card is incompatible. 44 = Save image failed because the controller is copy protected. 45 = Save image failed because the installed SD card is not writable. 46 = Save image failed because the controller image could not be built. 50 = Restore image failed due to an internal error. 51 = Restore image failed due to an SD card read error. 52 = Restore image failed because no SD card is installed. 53 = Restore image failed because the installed SD card is incompatible.

			54 = Restore image failed because the controller image file was invalid. 55 = Restore image failed because the controller image could not be applied. 56 = Restore image failed because no controller image file was found on the SD card.
%MD18.14	DINT	Read Only	Time, Milliseconds
%MD18.15	DINT	Read Only	Time, Microseconds
%MD18.16	DINT	Read Only	Time, Nanoseconds
%MD18.17	DINT	Read Only	Time, Loop Ticks
%MD18.18	DINT	Read Only	Real Time UTC, Seconds
%MD18.19	DINT	Read Only	Real Time, Nanoseconds
%MD18.20	DINT	Read Only	Real Time Local, Seconds
%MD18.24	REAL	Read Only	Loop Time Used, Total (µs)
%MD18.25	REAL	Read Only	Loop Time Used, Axes (µs)
%MD18.26	REAL	Read Only	Loop Time Used, Programs (µs)
%MD18.27	REAL	Read Only	Loop Time Used, Plots (µs)
%MD18.28	REAL	Read Only	Loop Time Used, Comm (µs)
%MD18.29	REAL	Read Only	Loop Time Used, Overhead (µs)

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, File 19: Event Log Configuration

This file is used by RMCTools to configure Event Log filtering. It should not be edited directly. Instead use the Event Log Properties to change these settings from within RMCTools.

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, File 20: Discrete I/O

All Discrete I/O registers are **Read/Write**.

Each output or input is one bit in the DWORD register. The inputs or outputs start at bit 0 in each DWORD register.

n = RMC200 slot (0-15)

Discrete Inputs and Outputs:

Internal IEC Address	Data Type	Tag Name	Register Name
%MD20.0+*n*	DWORD	_DIO.OutState[*n*]	Outputs for slot *n*

%MD20.16+*n*	DWORD	_DIO.InState[*n*]	Inputs for slot *n*

Discrete I/O Definitions:

Internal IEC Address	Data Type	Tag Name	Register Name
%MD20.32+*n*	DWORD	_DIO.OffInProgram[*n*]	Output to Off in PROGRAM mode for slot *n*
%MD20.48+*n*	DWORD	_DIO.OnInProgram[*n*]	Output to On in PROGRAM mode for slot *n*
%MD20.64+*n*	DWORD	_DIO.OffInFault[*n*]	Output to Off in FAULT mode for slot *n*
%MD20.80+*n*	DWORD	_DIO.OnInFault[*n*]	Output to On in FAULT mode for slot *n*
%MD20.96+*n*	DWORD	_DIO.OutForcedOFF[*n*]	Force Off for Outputs in slot *n*
%MD20.112+*n*	DWORD	_DIO.OutForcedON[*n*]	Force On for Outputs in slot *n*
%MD20.128+*n*	DWORD	_DIO.InForcedOFF[*n*]	Force Off for Inputs in slot *n*
%MD20.144+*n*	DWORD	_DIO.InForcedON[*n*]	Force On for Inputs in slot *n*
%MD20.160+*n*	DWORD	_DIO.Type[*n*]	Type for configurable I/O in slot *n*

%IX and %QX Addressing

In RMCTools, the discrete I/O addresses are shown in the IEC 61131-3 format:

Inputs = %IX*slot.n*

Outputs = %QX*slot.n*

where

slot = numbering of slot starting with 0 for the left-most module in the RMC200. The CPU module is slot 1.

n = the number of the input or output on that module, starting with zero

Examples:

%QX1.0 is output 0 in slot 1

%IX5.7 is input 7 in slot 5

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Show All Back

RMC200 Registers, File 21: Task Manager

Internal IEC Address	Data Type	Access	Register Name
%MD21.0	DINT	Read/Write	Number of Tasks Allocated 0-32 (default = 2)
%MD21.1	DINT	Read/Write	Program Triggers Task Enabled 0=false, 1= true (default = 1)

%MD21.2	DINT	Read/Write	Task Behavior on Axis Halt Defines the Task Stop Mode: 0: Stop no tasks on axis halt 1: Stop all tasks on axis halt 2: Stop specific task(s) on axis halt (see TaskMask fields defined below (%MD21.10)
%MD21.3	DINT	Read/Write	Startup Mode 0=PROGRAM, 1=RUN (default = 0)
%MD21.4	DINT	Read/Write	RUN/PROGRAM Input 0 = None 1 = %IX1.0 (CPU input 0) 2 = %IX1.1 (CPU input 1)
%MD21.5	DINT	Read/Write	Auto-enable Axes Selects whether all axes should be enabled automatically when the controller is enabled. Otherwise, axes must be enabled individually using the Enable/Disable Axis (97) command. 0=Disabled, 1=Enabled (default = 1)
%MD21.10	DWORD	Read/Write	Axis Halt Affected Task Bitmask Bitmask indicating which tasks (Task0..Task31) are affected by the "Stop Specific Tasks" selection of %MD21.2 above.
%MD21.11	DWORD	Read/Write	Axis Halt Affected Task Bitmask Expansion Bitmask indicating which tasks (Task32..Task63) are affected by the "Stop Specific Tasks" selection of %MD21.2 above.

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Show All Back

RMC200 Registers, File 22: Plot Manager

This file contains the Plot Layout registers.

Internal IEC Address	Data Type	Access	Register Name
%MD22.0	DINT	Read/Write	Number of Plots Controls how many plots are active. Changing this value will cause the firmware to re-evaluate the layout of the plots to ensure that it has room for all the data in the plots. This may cause one or more plots to be reset. The default for this register is 8.
%MD22.1	DINT	Read Only	Maximum Plots This read-only value indicates how many plot templates can be defined at once. CPU20L: 48 CPU40: 64 (32 prior to firmware 1.14.0)

<table>
<tr><td>%MD22.2</td><td>DINT</td><td>Read Only</td><td>Maximum Data Sets per Plots
This read-only value indicates the maximum number of data sets that be captured simultaneously by an individual plot.
CPU20L: 48
CPU40: 128 (32 prior to firmware 1.14.0)</td></tr>
<tr><td>%MD22.3</td><td>DINT</td><td>Read Only</td><td>Maximum Samples per Loop Time
This read-only value indicates the maximum number of total data set samples that are allowed per loop time across all plots.
CPU20L: 512
CPU40: 1024</td></tr>
<tr><td>%MD22.4</td><td>DINT</td><td>Read Only</td><td>Plot Pool Size
This read-only value indicates the size of the plot capture pool shared by all plots, in 32-bit values.
CPU20L: 12x1024x1024 = 12,582,912
CPU40: 48x1024x1024 = 50,331,648 (12,582,912 prior to firmware 1.14.0)</td></tr>
<tr><td>%MD22.16-79</td><td>DINT</td><td>Read/Write</td><td>Static Plot Upload Area Map
Each register in this group corresponds to a Static Plot Upload Area file (%MD640-703), and defines which plot data is referenced by each file.

<table><tr><th>Register</th><th>Configures this Static Plot Upload file:</th></tr><tr><td>%MD22.16</td><td>%MD640</td></tr><tr><td>%MD22.17</td><td>%MD641</td></tr><tr><td>...</td><td>...</td></tr><tr><td>%MD22.79</td><td>%MD703</td></tr></table>
The definition of each register is as follows:
Bits 31-24: Plot Number
Selects which plot this data set comes from (0-31).
Bits 23-16: Plot Data Set
Select the data set within the plot that will be mapped into this file (0-31).
Bits 15-0: Plot Interval Offset
Selects the starting offset into the interval that will be referenced by this file. This offset is multiplied by 256. For example, suppose that a client is only interested in accessing a single very large plot, with 10 data sets and 10,000 samples each. Since each Static Plot Upload Area file is limited to 4096 elements, the user will need to allocate 3 files per data set. The first file for each data set would have a Plot Interval Offset of 0 and would hold samples 0-4095, the second file for each data set would have a Plot Interval Offset of 16 (x256=4096) and would hold samples 4096-8191, and the third file for each data set would have a Plot Interval Offset of 32 (x256=8192) and would hold samples 8192-9999.</td></tr>
</table>

Most users accessing plots statically from an external device should be able to use the default Static Plot Upload Area allocation, which is as follows:

Static Plot Upload Area	Plot	Data Set	Offset
%MD640	0	0	0
%MD641	0	1	0
...	...	...	...
%MD655	0	15	0
%MD656-671	1	0..15	0
%MD672-687	2	0..15	0
%MD688-703	3	0..15	0

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, File 23: Image Upload/Download Area

See the Controller Image Upload/Download topic for details.

Internal IEC Address	Data Type	Access	Register Name
%MD23.0	DINT	Write Only	**Image Area Command**
%MD23.1	DINT	Read Only	**Image Area State**
%MD23.2	DINT	Read Only	**Image Size**
%MD23.3	DINT	Read/Write	**Current Index**
%MD23.4 : %MD23.4095	DINT[4092]	Read/Write	**Image Data**

See Also

RMC200 Register Map | Controller Image Upload/Download | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, Files 24 and 25: Axis Definitions

The Axis Definitions are not intended to be directly accessed by the user. The preferred method of changing the axis definitions is to use the Axis Definitions dialog.

For highly advanced users, see the Axis Definition Registers topic for more details.

Internal IEC Address	Data Type	Register Name
%MD24.0-511	DWORD	Current Axis Definitions (Read-Only)
%MD25.0-511	DWORD	Requested Axis Definitions

The Current Axis Definitions and the Requested Axis Definitions will generally be the same except in two cases:

(1) The user has written to the requested block and intends to do a warm restart or burn to flash and do a cold restart, or

(2) The requested axis definitions found on startup are invalid for the current hardware configuration; in this case the Current Axis Definitions will be the default for the current hardware configuration.

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, File 26: Module Definitions

The read-only Module Definitions (actual) registers (file 26) list the modules that the RMC200 currently has in the base. The Module Definitions (defined) registers (file 27) are identical to file 26 and will support additional functionality in the future.

Internal IEC Address	Data Type	All Axes
%MD26.0	DINT	Base Module Definition Bits 15-0: Module Type 64: R200-B7 65: R200-B11 66: R200-B5 69: R200-B15 72: R200-B5L 73: R200-B7L Bits 31-16: Reserved
%MD26.1	DINT	Slot 0 (Power Supply) Module Definition Bits 15-0: Module Type 96: R200-PS4D 97: R200-PS6D 98: CPU20L integrated power supply Bits 31-16: Reserved
%MD26.2	DINT	Slot 1 (CPU) Module Definition

		Bits 15-0: Module Type 0: R200-CPU40 2: R200-CPU20L Bits 31-16: Reserved
%MD26.3-15	DINT	Slots 2-14 Module Definition Bits 15-0: Module Type 132: R200-A8 136: R200-CA4 140: R200-S8 144: R200-Q4 148: R200-D24 152: R200-CV8 156: R200-U14 160: R200-LC8 65535: Empty Bits 31-16: Reserved

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, File 112: Communication Status/Configuration

The RMC200 Communication Status/Configuration area is much smaller than the RMC75/150 area since the RMC200 accesses most of the communication registers differently. Only the items needed to be accessed by the user programs are included here.

Internal IEC Address	Data Type	Access	Register Name
%MD112.0	DWORD	Read Only	EtherNet/IP I/O Connection #1 Status
%MD112.1	DINT	Read Only	EtherNet/IP I/O Connection #1 PLC Status
%MD112.4	DWORD	Read Only	EtherNet/IP I/O Connection #2 Status
%MD112.5	DINT	Read Only	EtherNet/IP I/O Connection #2 PLC Status
%MD112.8	DWORD	Read Only	EtherNet/IP I/O Connection #3 Status
%MD112.9	DINT	Read Only	EtherNet/IP I/O Connection #3 PLC Status
%MD112.12	DWORD	Read Only	PROFINET I/O Connection Status
%MD112.13	DINT	Read Only	PROFINET I/O Connection PLC Status

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, Files 113-116: Address Maps

The following files define the custom portions of the Address Maps. The pre-defined portions of the address maps are not included.

File 113: DF1 Address Map

File 114: Modbus Address Map

File 115: FINS Address Map

File 116: PROFINET Data Record Address Map

File 113: DF1 Address Map

Supports up to 128 ranges

Internal IEC Address	Data Type	Access	Register Name
%MD113.(*n*·3+0)	DWORD	Read/Write	**DF1 Address: Range n** The DF1 address of the start of this mapping range: • **Bits 31-28: File Type** 0: Unused mapping range 1: Float (F) File Type 2: Long (L) File Type 3-15: Reserved • **Bits 27-24: Reserved** • **Bits 23-12: File Number** 0-4095 • **Bits 11-0: Element Number** 0-4095
%MD113.(*n*·3+1)	DINT	Read/Write	**Register Count: Range n** The number of 32-bit registers in this mapping range.
%MD113.(*n*·3+2)	DINT	Read/Write	**RMC Address: Range n** The RMC's IEC %MD*file.element* address of the start of this mapping range. The address value is: *file* * 4096 + *element*.

File 114: Modbus Address Map

Supports up to 128 ranges

Internal IEC Address	Data Type	Access	Register Name
%MD114.(*n*·3+0)	DWORD	Read/Write	**Modus Address: Range n** The Modus address of the start of this mapping range: • **Bits 31-28: Memory Table** 0: Unused mapping range 1-3: Reserved 4: Holding registers (400001-…) 3-15: Reserved • **Bits 27-16: Reserved** • **Bits 15-0: Data Address** Data address from the start of the specified table. For example, Modbus Address 400001 will have

			Table=4, Data Address=0, and address 402049 will have Table=4, Data Address=2048. Valid values are even integers from 0-65,534.
%MD114.(*n*·3+1)	DINT	Read/Write	**Register Count: Range n** The number of 32-bit RMC registers in this mapping range. Since each Modbus Holding Register is only 16 bits, the range will include twice as many Holding Registers as RMC registers.
%MD114.(*n*·3+2)	DINT	Read/Write	**RMC Address: Range n** The RMC's IEC %MD*file.element* address of the start of this mapping range. The address value is: *file* * 4096 + *element*.

File 115: FINS Address Map

Supports up to 128 ranges

Internal IEC Address	Data Type	Access	Register Name
%MD115.(*n*·3+0)	DWORD	Read/Write	**Modus Address: Range n** The FINS address of the start of this mapping range: • **Bits 31-28: Memory Area** 0: Unused mapping range 1: DM Area 2: EM Area 3-15: Reserved • **Bits 27-24: Reserved** • **Bits 23-16: EM Bank** The EM bank area (0-C) if Memory Area is EM Area (2). • **Bits 15-0: Data Address** Starting address within the specified area above. Valid values are even integers from 0-32,766.
%MD115.(*n*·3+1)	DINT	Read/Write	**Register Count: Range n** The number of 32-bit RMC registers in this mapping range. Since each FINS DM/EM register is only 16 bits, the range will include twice as many FINS DM/EM registers as RMC registers.
%MD115.(*n*·3+2)	DINT	Read/Write	**RMC Address: Range n** The RMC's IEC %MD*file.element* address of the start of this mapping range. The address value is: *file* * 4096 + *element*.

File 116: PROFINET Data Record Address Map

Supports up to 128 ranges

Internal IEC Address	Data Type	Access	Register Name
%MD116.(*n*·3+0)	DWORD	Read/Write	**PROFINET Address: Range n**

			The PROFINET address of the start of this mapping range: • **Bits 31-29: Record Type** 0: Unused mapping range 1: Standard Data Record 2-7: Reserved • **Bits 28-14: Record Index** Record Index for this range (0-32767). • **Bits 13-0: Register Offset** Register offset (in 32-bit registers) from the start of this data record. Currently only a value of zero is supported.
%MD116.(*n*·3+1)	DINT	Read/Write	**Register Count: Range n** The number of 32-bit RMC registers in this mapping range.
%MD116.(*n*·3+2)	DINT	Read/Write	**RMC Address: Range n** The RMC's IEC %MD*file.element* address of the start of this mapping range. The address value is: *file* * 4096 + *element*.

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Show All Back

RMC200 Registers, File 128-143: Base, Slot Status/Configuration

Each of these register files holds module-specific information about the module installed in the given location.

File 128: Base Module

Internal IEC Address	Data Type	Access	Register Name
%MD128.0	DINT	Read Only	Base Module Type (Defined) 66: R200-B5 64: R200-B7 65: R200-B11 69: R200-B15 72: R200-B5L 73: R200-B7L
%MD128.1	DINT	Read Only	Base Module Type (Actual) 66: R200-B5 64: R200-B7 65: R200-B11 69: R200-B15 72: R200-B5L 73: R200-B7L

			65534: Unknown (unable to identify the base)
%MD128.2	DINT	Read Only	Base Module State One of the following: 0 = No module detected 1 = Active 2 = Wrong module. The Module Type does not match the Module Type (Defined). 3 = Unsupported module.
%MD128.3	DINT	Read Only	Base Module Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bits 7-0: Mod level (0=A, 1=B, …)
%MD128.4	DINT	Read Only	Firmware Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bit 7: 1=Beta, 0=Standard Bits 6-0: Config ID (1=A, 2=B, ...)
%MD128.16	DINT	Read Only	Total actual slot count (includes PS, CPU, all I/O slots)

File 129: Slot #0 (Power Supply)

Internal IEC Address	Data Type	Access	Register Name
%MD129.0	DINT	Read Only	PS Module Type (Defined) 96: R200-PS4D 97: R200-PS6D 98: CPU20L integrated power supply
%MD129.1	DINT	Read Only	PS Module Type (Actual) 96: R200-PS4D 97: R200-PS6D 98: CPU20L integrated power supply 65534: Unknown (unable to identify)
%MD129.2	DINT	Read Only	PS Module State 0: No module detected. This will be reported if the module does not respond to communication. The power supply itself is active (or the CPU would not be running) but we have no information from the module. 1: Active. This will be reported when the module was enumerated and is supported and providing us with real-time information.

			2: Wrong module. This will be reported when the module was enumerated, but the Module Type does not match the Module Type (Defined) value. 3: Unsupported module. This will be reported when the module was enumerated, and the Module Type matches the Module Type (Defined) value, but the revision of the module is not supported.
%MD129.3	DINT	Read Only	PS Module Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bits 7-0: Mod level (0=A, 1=B, etc.)
%MD129.4	DINT	Read Only	Firmware Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bit 7: 1=Beta, 0=Standard Bits 6-0: Config ID (1=A, 2=B, ...)
%MD129.7	REAL	Read Only	Module Temperature (ºC)
%MD129.16	REAL	Read Only	External Voltage (V)
%MD129.17	REAL	Read Only	Internal Voltage (V)
%MD129.18	REAL	Read Only	Current (A)
%MD129.19	REAL	Read Only	Power (W)
%MD129.20	REAL	Read Only	Loader Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bit 7: 1=Beta, 0=Standard Bits 6-0: Config ID (1=A, 2=B, ...)

File 130: Slot #1 (CPU)

Internal IEC Address	Data Type	Access	Register Name
%MD130.0	DINT	Read Only	CPU Module Type (Defined) 0: R200-CPU40 2: R200-CPU20L
%MD130.1	DINT	Read Only	CPU Module Type (Actual) 0: R200-CPU40 2: R200-CPU20L
%MD130.2	DINT	Read Only	CPU Module State 1: Active. The CPU will always report the Active state. The CPU can only be loaded with

			firmware that supports that particular model.
%MD130.3	DINT	Read Only	CPU Module Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bits 7-0: Mod level (0=A, 1=B, etc.)
%MD130.4	DINT	Read Only	FPGA Firmware Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bit 7: 1=Beta, 0=Standard Bits 6-0: Config ID (1=A, 2=B, ...)
%MD130.7	REAL	Read Only	Module Temperature (core) (ºC)
%MD130.8	REAL	Read Only	Module Temperature (top) (ºC)
%MD130.16	DINT	Read Only	Main Firmware Revision Bits 31-24: Major revision Bits 23-16: Minor revision Bits 15-8: Patch revision Bit 7-0: Config ID (1=A, 2=B, ...)
%MD130.17	DINT	Read Only	Main Firmware Special Release Code 0: Standard 1: Experimental (EXP) 2-127: Special Release (S2-S127) 128: Beta Standard 129: Beta Experimental (EXP) 130-255: Beta Special Release (S2-S127)
%MD130.18	DINT	Read Only	Recovery Firmware Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bit 7: 1=Beta, 0=Standard Bits 6-0: Config ID (1=A, 2=B, ...)
%MD130.19	DINT	Read Only	Boot Firmware Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bit 7: 1=Beta, 0=Standard Bits 6-0: Config ID (1=A, 2=B, ...)

Files 131-143: Slot #2-14

n = 129 + slot number

Internal IEC Address	Data Type	Access	Register Name
%MD*n*.0	DINT	Read Only	Module Type (Defined) 132: R200-A8

			136: R200-CA4 140: R200-S8 144: R200-Q4 148: R200-D24 152: R200-CV8 156: R200-U14 160: R200-LC8 65535: Empty
%MD*n*.1	DINT	Read Only	Module Type (Actual) 132: R200-A8 136: R200-CA4 140: R200-S8 144: R200-Q4 148: R200-D24 152: R200-CV8 156: R200-U14 160: R200-LC8 65535: Empty
%MD*n*.2	DINT	Read Only	Module State One of the following: 0 = No module detected 1 = Active 2 = Wrong module. The Module Type does not match the Module Type (Defined). 3 = Unsupported module. 4 = Ignored module. The module was detected, but the Module Type (Defined) is Empty (65535). 5 = Module Updating. When a module is taken offline in order to update its firmware or soft ID.
%MD*n*.3	DINT	Read Only	Module Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bits 7-0: Mod level (0=A, 1=B, etc.)
%MD*n*.4	DINT	Read Only	Firmware Revision Bits 23-16: Major revision Bits 15-8: Minor revision Bit 7: 1=Beta, 0=Standard Bits 6-0: Config ID (1=A, 2=B, ...)
%MD*n*.7	REAL	Read Only	Module Temperature (ºC) Supported by the CA4, CV8, A8, U14, and LC8 modules.
%MD*n*.16	DINT	Read/Write	Module Configuration Mode This register selects the configuration mode for this module. This register

only applies to the S8, U14, and D24 modules.

R200-S8 Module:

This register selects the operating mode for channels 6 and 7. The following options are available:

- **Two SSI/MDT inputs (0).** Channels 6 and 7 are standard SSI/MDT inputs.
- **One Quadrature input (1).** Channels 6 and 7 are defined as a single A/B quadrature input with no hardware homing or registration.
- **One SSI Monitor input (2).** Channels 6 and 7 are defined as a single SSI monitor input.

See Configuring S8 Channels for details.

R200-U14 Module:

This register selects the operating mode for the U14 high-speed channels.

Bits 0-7: Channel 0

Value	Mode
0	SSI/MDT
1	Quadrature input
2	SSI Monitor

Bits 8-15: Channel 1

Value	Mode
0	SSI/MDT
1	Quadrature input
2	SSI Monitor
3	SSI Echo

See U14 Module (RMC200) for details.

R200-D24 Module:

This register selects the operating mode for the D24 high-speed inputs (20-23). The following options are available:

- **No quadrature inputs (0).** The inputs are used as general-purpose inputs.

			• **One quadrature input (A, B, Z).** Inputs 20-22 are used as A, B, and Z encoder inputs. Input 23 can be used as a general-purpose input. • **One quadrature input (A, A, B, B) with wire-break detection.** Inputs 20-23 are used as A, B, A, and B encoder inputs. • **Two quadrature inputs (A, B).** Inputs 20-21 are A and B for one encoder and inputs 22-23 are A and B for a second encoder. See D24 Module (RMC200) for details. **R200-LC8 Module:** This register selects the minimum filter frequency for the module-wide hardware filter in the LC8 module: • **0 = 150 Hz (default)** 2000 µs ADC conversion time • **1 = 300 Hz** 1000 µs ADC conversion time • **2 = 600 Hz** 500 µs ADC conversion time • **3 = 1200 Hz** 250 µs ADC conversion time • **4 = 2400 Hz** 125 µs ADC conversion time If a filter is selected whose ADC conversion time exceeds the motion loop time, then the firmware will automatically increase the filter frequency. For example, if 150 Hz is selected on a controller with a 500 µs loop time, then the firmware will force the LC8 module to use the 600 Hz filter frequency. See Configuring the LC8 Module for details.
%MD*n*.100	DINT	Read Only	Module Event Timer Count Holds the number of Event Timers provided by this module. This will be four (4) for the D24 module and zero (0) for all other modules. See Event Timers for details.
%MD*n*.101	DWORD	Read Only	Event Timer 0 Status Holds the status of Event Timer 0 in this module:

			Bit 0: Event Timer Armed. This bit is set when this Event Timer is armed. **Bit 1:** Event Timer Latched. This bit will be set when the Event Timer has latched a time for the event. **Bits 2-31:** Reserved. This register only applies to modules that support event timers. See Event Timers for details.
%MD*n*.102	DINT	Read Only	Event Timer 0 Seconds The seconds portion of the latched timestamp. See Event Timers for details.
%MD*n*.103	DINT	Read Only	Event Timer 0 Nanoseconds The nanoseconds portion of the latched timestamp. See Event Timers for details.
%MD*n*.105	DWORD	Read Only	Event Timer 1 Status
%MD*n*.106	DINT	Read Only	Event Timer 1 Seconds
%MD*n*.107	DINT	Read Only	Event Timer 1 Nanoseconds
%MD*n*.109	DWORD	Read Only	Event Timer 2 Status
%MD*n*.110	DINT	Read Only	Event Timer 2 Seconds
%MD*n*.111	DINT	Read Only	Event Timer 2 Nanoseconds
%MD*n*.113	DWORD	Read Only	Event Timer 3 Status
%MD*n*.114	DINT	Read Only	Event Timer 3 Seconds
%MD*n*.115	DINT	Read Only	Event Timer 3 Nanoseconds

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, Files 192-255: Task Status/Configuration

The Task Status can be viewed in the Task Monitor. Each task has its own status/configuration file, where *n* = task number.

Internal IEC Address	Data Type	Access	Register Name
%MD192+*n*.0	DWORD	Read Only	Task *n* Task Status
%MD192+*n*.1	DWORD	Read Only	Task *n* Current Program/Step
%MD192+*n*.2	DINT	Read/Write	Task *n* Current Axis
%MD192+*n*.3	DINT	Read Only	Task *n* Current Program
%MD192+*n*.4	DINT	Read Only	Task *n* Current Step
%MD193-223			Task 1-31 Status/Configuration
%MD224-255			Task 32-63 Status/Configuration

*Task 32-63 Status/Configuration for CPU40 only.

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, Files 256-383: Axis Status Registers

All Axis Status Registers are **Read Only**, with the exception of the Custom Counts and Custom Error Bits.

f = 256 + axis number (starting with zero)

The CPU20L supports 48 axes. The CPU40 supports 128 axes.

General

Internal IEC Address	Data Type	All Axes
General		
%MD*f*.0	DWORD	Status Bits
%MD*f*.1	DWORD	Error Bits

Feedback

Internal IEC Address	Data Type	Position Axes	Velocity Axes	Single-Ended Pressure/Force	Load Cell Single-Ended Force	Differential Force	Single-Ended Acceleration	Differential Acceleration
Primary Feedback								
%MD*f*.20	REAL	Actual Position		Actual Prs/Frc	Actual Force	Actual Force	Actual Acceleration	Actual Acceleration
%MD*f*.21	REAL	Actual Velocity	Actual Velocity	Actual Force Rate	Actual Force Rate	Actual Force Rate	Actual Jerk	Actual Acceleration
%MD*f*.22	REAL	Actual Acceleration	Actual Acceleration					
%MD*f*.24	REAL					Channel A Force		Channel A Accel
%MD*f*.25	REAL					Channel A Pressure		
%MD*f*.26	REAL	Counts/Volts/Cur.	Counts/Volts/Cur.	Counts/Volts/Cur.	Millivolts/Volt	Volts/Current A	Counts/Volts/Current	Volts/Current A
%MD*f*.27	DINT	Raw Counts	Raw Counts	Raw Counts	Raw Counts	Raw Counts A	Raw Counts	Raw Counts A
%MD*f*.28	REAL					Channel B Force		Channel B Accel

%MD*f*.29	REAL					Channel B Pressure		
%MD*f*.30	REAL					Volts/Current B		Volts/Current B
%MD*f*.31	DINT					Raw Counts B		Raw Counts B
%MD*f*.32	REAL	Actual Position (Unfilt)		Actual P/F (Unfilt)	Actual Force (Unfilt)	Actual Force (Unfilt)	Actual Accel (Unfilt)	Actual Accel (Unfilt)
%MD*f*.33	REAL	Actual Velocity (Unfilt)	Actual Velocity (Unfilt)	Actual P/F Rate (Unfilt)	Act Force Rate (Unfilt)	Act Force Rate (Unfilt)	Actual Jerk (Unfiltered)	Actual Jerk (Unfiltered)
%MD*f*.34	REAL	Actual Accel (Unfilt)	Actual Accel (Unfilt)					
%MD*f*.36	REAL	Actual Pos (Control)		Actual P/F (Control)	Actual Force (Control)	Actual Force (Control)	Actual Accel (Control)	Actual Accel (Control)
%MD*f*.37	REAL	Actual Vel (Control)	Actual Vel (Control)	Actual P/F Rate (Ctrl)	Act Frc Rate (Control)	Act Frc Rate (Control)	Actual Jerk (Control)	Actual Jerk (Control)
%MD*f*.38	REAL	Actual Accel (Control)	Actual Accel (Control)					
%MD*f*.41	DWORD	Trans. 0 Status A	Trans. 0 Status A	Trans. 0 Status A	Trans. 0 Status A	Trans. 0 Status A	Trans. 0 Status A	Trans. 0 Status A
%MD*f*.42	DWORD	Trans. 0 Status B	Trans. 0 Status B	Trans. 0 Status B	Trans. 0 Status B	Trans. 0 Status B	Trans. 0 Status B	Trans. 0 Status B
%MD*f*.43	DWORD					Trans. 1 Status A		Trans. 1 Status A
%MD*f*.44	DWORD					Trans. 1 Status B		Trans. 1 Status B
%MD*f*.45	REAL				Millivolt Input			
%MD*f*.46	REAL				Effective Exciter Voltage			
%MD*f*.50	REAL	Custom Counts	Custom Counts	Custom Counts		Custom Counts	Custom Counts	Custom Counts
%MD*f*.51	DWORD	Custom Error Bits	Custom Error Bits	Custom Error Bits		Custom Error Bits	Custom Error Bits	Custom Error Bits
%MD*f*.55	DWORD	Encoder Status						
%MD*f*.56	DWORD	Registration 0 Position						
%MD*f*.57	DWORD	Registration 1 Position						
Secondary Feedback								

%MD*f*.80	REAL			Actual Prs/Frc	Actual Force	Actual Force	Actual Acceleration	Actual Acceleration
%MD*f*.81	REAL			Actual Prs/Frc Rate	Actual Force Rate	Actual Force Rate	Actual Jerk	Actual Jerk
%MD*f*.84	REAL					Channel A Force		Channel A Accel
%MD*f*.85	REAL					Channel A Pressure		
%MD*f*.86	REAL			Counts/Volts/Cur.	Millivolts/Volt	Volts/Current A	Counts/Volts/Cur.	Volts/Current A
%MD*f*.87	DINT			Raw Counts	Raw Counts	Raw Counts A	Raw Counts	Raw Counts A
%MD*f*.88	REAL					Channel B Force		Channel B Accel
%MD*f*.89	REAL					Channel B Pressure		
%MD*f*.90	REAL					Volts/Current B		Volts/Current B
%MD*f*.91	DINT					Raw Counts B		Raw Counts B
%MD*f*.92	REAL			Actual P/F (Unfilt)	Actual Force (Unfilt)	Actual Force (Unfilt)	Actual Accel (Unfilt)	Actual Accel (Unfilt)
%MD*f*.93	REAL			Actual P/F Rate (Unfilt)	Actual Frc Rate (Unfilt)	Actual Frc Rate (Unfilt)	Actual Jerk (Unfiltered)	Actual Jerk (Unfiltered)
%MD*f*.96	REAL			Actual P/F (Control)	Actual Force (Control)	Actual Force (Control)	Actual Accel (Control)	Actual Accel (Control)
%MD*f*.97	REAL			Actual P/F Rate (Ctrl)	Act Frc Rate (Control)	Act Frc Rate (Control)	Actual Jerk (Control)	Actual Jerk (Control)
%MD*f*.101	DWORD			Trans. 0 Status A	Trans. 0 Status A	Trans. 0 Status A	Trans. 0 Status A	Trans. 0 Status A
%MD*f*.102	DWORD			Trans. 0 Status B	Trans. 0 Status B	Trans. 0 Status B	Trans. 0 Status B	Trans. 0 Status B
%MD*f*.103	DWORD					Trans. 1 Status A		Trans. 1 Status A
%MD*f*.104	DWORD					Trans. 1 Status B		Trans. 1 Status B
%MD*f*.105	REAL				Millivolt Input			
%MD*f*.106	REAL				Effective Exciter Voltage			

%MD*f*.110	REAL			Custom Counts		Custom Counts	Custom Counts	Custom Counts
%MD*f*.111	DWORD			Custom Error Bits		Custom Error Bits	Custom Error Bits	Custom Error Bits

Control (Common)

Internal IEC Address	Data Type	Description
Control (Common)		
%MD*f*.140	REAL	Control Output (%)
%MD*f*.141	REAL	Final Output
%MD*f*.150	DINT	Current Control Mode
%MD*f*.151	REAL	PFID Output
%MD*f*.170	DINT	Current Integrator Mode
%MD*f*.171	DINT	Next Pos/Vel Control Mode

Control (Primary and Secondary)

Internal IEC Address	Data Type	Position	Velocity	Pressure/Force
Primary Control				
%MD*f*.180	REAL	Position Error		Prs/Frc Error
%MD*f*.181	REAL	Velocity Error	Velocity Error	
%MD*f*.182	REAL	Proportional Term	Proportional Term	Prs/Frc Proportional Term
%MD*f*.183	REAL	Integral Term	Integral Term	Prs/Frc Integral Term
%MD*f*.184	REAL	Differential Term	Differential Term	Prs/Frc Differential Term
%MD*f*.185	REAL	Double Differential Output Term	Double Differential Output Term	
%MD*f*.186	REAL	Triple Differential Output Term		
%MD*f*.187	REAL			Prs/Frc Feed Forward Term
%MD*f*.188	REAL	Velocity Feed Forward Term	Velocity Feed Forward Term	Prs/Frc Rate Feed Forward Term
%MD*f*.189	REAL	Acceleration Feed Forward Term	Acceleration Feed Forward Term	
%MD*f*.190	REAL	Jerk Feed Forward Term	Jerk Feed Forward Term	
%MD*f*.192	REAL	Current Gain Set	Current Gain Set	
Secondary Control				
%MD*f*.290	REAL	Position Error		Prs/Frc Error
%MD*f*.291	REAL	Velocity Error	Velocity Error	
%MD*f*.292	REAL	Proportional Term	Proportional Term	Prs/Frc Proportional Term
%MD*f*.293	REAL	Integral Term	Integral Term	Prs/Frc Integral Term
%MD*f*.294	REAL	Differential Term	Differential Term	Prs/Frc Differential Term

%MD*f*.295	REAL	Double Differential Output Term	Double Differential Output Term	
%MD*f*.296	REAL	Triple Differential Output Term		
%MD*f*.297	REAL			Prs/Frc Feed Forward Term
%MD*f*.298	REAL	Velocity Feed Forward Term	Velocity Feed Forward Term	Prs/Frc Rate Feed Forward Term
%MD*f*.299	REAL	Acceleration Feed Forward Term	Acceleration Feed Forward Term	
%MD*f*.300	REAL	Jerk Feed Forward Term	Jerk Feed Forward Term	
%MD*f*.302	REAL	Current Gain Set	Current Gain Set	

Target

Internal IEC Address	Data Type	Position	Velocity	Pressure/Force
Target				
%MD*f*.400	REAL	Target Position		
%MD*f*.401	REAL	Target Velocity	Target Velocity	
%MD*f*.402	REAL	Target Acceleration	Target Acceleration	
%MD*f*.403	REAL	Target Jerk	Target Jerk	
%MD*f*.404	REAL	Command Position		
%MD*f*.405	REAL	Command Velocity	Command Velocity	
%MD*f*.410	DINT	Cycles	Cycles	
%MD*f*.440	REAL			Target Pressure/Force
%MD*f*.441	REAL			Target Pressure/Force Rate
%MD*f*.444	REAL			Command Pressure/Force
%MD*f*.450	DINT			Cycles (Pressure/Force)

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, Files 384-511: Axis Parameters

All Axis Parameter registers are **Read/Write**.

f = 384 + axis number (starting with zero)

The CPU20L supports 48 axes. The CPU40 supports 128 axes.

General

Internal IEC Address	Data Type	All Axes
General		
%MD*f*.0	DWORD	Auto Stops (errors 0-7)

%MD*f*.1	DWORD	Auto Stops (errors 8-15)
%MD*f*.2	DWORD	Auto Stops (errors 16-23)
%MD*f*.4	REAL	Closed Loop Halt Deceleration
%MD*f*.5	REAL	Open Loop Halt Ramp
%MD*f*.6	DINT	Halt Group Number
%MD*f*.10	DWORD	Enable Output Configuration
%MD*f*.11	DWORD	Fault Input Configuration

Feedback

Internal IEC Address	Data Type	Position Axes	Velocity Axes	Single-Ended Pressure/Force	Load Cell Single-Ended Force	Differential Force	Single-Ended Acceleration	Differential Acceleration
Primary Feedback								
%MD*f*.20	REAL	Position Scale	Velocity Scale	Pressure/Force Scale	Force Scale	Force A Scale	Acceleration Scale	Channel A Accel. Scale
%MD*f*.21	REAL	Position Offset	Velocity Offset	Pressure/Force Offset	Force Offset	Prs. A Offset	Acceleration Offset	Channel A Accel. Offset
%MD*f*.22	REAL			Neg. Correction Factor	Neg. Correction Factor	Prs. A Scale		Channel B Accel. Scale
%MD*f*.23	REAL					Force B Scale		Channel B Accel. Offset
%MD*f*.24	REAL					Prs. B Scale		
%MD*f*.25	REAL					Prs. B Offset		
%MD*f*.26	REAL					Dual Chan Force Offset		
%MD*f*.27	REAL		Velocity Deadband					
%MD*f*.28	REAL	Position Unwind						
%MD*f*.29	REAL	Count Unwind						
%MD*f*.30	REAL	Count Offset		Voltage/Current Offset	Millivolt/Volt Offset			
%MD*f*.31	REAL	Stop Threshold	Stop Threshold					
%MD*f*.32	REAL	Noise Error Rate	Noise Error Rate	Noise Error Rate	Noise Error Rate	Noise Error Rate	Noise Error Rate	Noise Error Rate
%MD*f*.33	DINT	Linear/Rotary						

%MD*f*.34	DWORD	Limit Inputs Config	Limit Inputs Config					
%MD*f*.35	DINT	Display Units	Display Units	Display Units	Display Units	Display Units	Display Units	Display Units
%MD*f*.36	DINT					Pressure Display Units		
%MD*f*.40	DWORD	Filter Configuration	Filter Configuration	Filter Configuration	Filter Configuration	Filter Configuration	Filter Configuration	Filter Configuration
%MD*f*.41	REAL	Pos Display Filter		Prs/Frc Display Filter	Frc Display Filter	Frc Display Filter	Accel Display Filter	Accel Display Filter
%MD*f*.42	REAL	Vel Display Filter	Vel Display Filter	Prs/Frc Rate Display Filter	Frc Rate Display Filter	Frc Rate Display Filter	Jerk Display Filter	Jerk Display Filter
%MD*f*.43	REAL	Acc Display Filter	Acc Display Filter					
%MD*f*.44	REAL	Model Response						
%MD*f*.45	REAL	Pos Input Filter		Prs/Frc Input Filter	Force Input Filter	Force Input Filter	Accel Input Filter	Acceleration Input Filter
%MD*f*.46	REAL	Vel Input Filter	Velocity Input Filter	Prs/Frc Rate Input Filter	Force Rate Input Filter	Force Rate Input Filter	Jerk Input Filter	Jerk Input Filter
%MD*f*.47	REAL	Acc Input Filter	Accel Input Filter					
%MD*f*.50	DINT	Model Type		Model Type	Model Type	Model Type		
%MD*f*.51	REAL	Model Gain Positive		Model Gain Prs/Frc	Model Gain Force	Model Gain Force		
%MD*f*.52	REAL	Model Gain Negative						
%MD*f*.53	REAL	Model Time Constant		Model Time Constant	Model Time Constant	Model Time Constant		
%MD*f*.54	REAL	Model Natural Freq.		Model Natural Freq.	Model Natural Freq.	Model Natural Freq.		
%MD*f*.55	REAL	Model Damping Factor		Model Damping Factor	Model Damping Factor	Model Damping Factor		
%MD*f*.59	DWORD	Custom FB Config	Custom FB Config	Custom FB Config			Custom FB Config	
%MD*f*.60	DWORD	SSI/MDT Config			Load Cell Config			
	DWORD	Analog Config	Analog Config	Analog Config		Analog Config	Analog Config	Analog Config

	DWORD	Quadrature Config						
%MD*f*.61	DWORD	SSI Data Config			Load Cell Overflow Lim			
	REAL	Analog Overflow Limit	Analog Overflow Limit	Analog Overflow Limit		Analog Overflow Limit	Analog Overflow Limit	Analog Overflow Limit
%MD*f*.62	REAL	Analog Underflow Limit	Analog Underflow Limit	Analog Underflow Limit	Load Cell Underflow Lim	Analog Underflow Limit	Analog Underflow Limit	Analog Underflow Limit
%MD*f*.63					Fixed Exciter Voltage			
Secondary Feedback								
%MD*f*.80	REAL			Pressure/Force Scale	Force Scale	Force A Scale	Acceleration Scale	Channel A Accel. Scale
%MD*f*.81	REAL			Pressure/Force Offset	Force Offset	Prs A Offset	Acceleration Offset	Channel A Accel. Offset
%MD*f*.82	REAL			Neg. Correction Factor	Neg. Correction Factor	Prs A Scale		Channel B Accel. Scale
%MD*f*.83	REAL					Force B Scale		Channel B Accel. Offset
%MD*f*.84	REAL					Prs B Offset		
%MD*f*.85	REAL					Prs B Scale		
%MD*f*.86	REAL					Dual Chan Frc Offset		
%MD*f*.90	REAL			Voltage/Current Offset	Millivolts/Volt Offset			
%MD*f*.92	REAL			Noise Error Rate	Noise Error Rate	Noise Error Rate	Noise Error Rate	Noise Error Rate
%MD*f*.95	DINT			Display Units	Display Units	Display Units	Display Units	Display Units
%MD*f*.96	DINT					Pressure Display Units		
%MD*f*.100	DWORD			Filter Configuration	Filter Configuration	Filter Configuration	Filter Configuration	Filter Configuration
%MD*f*.101	REAL			Prs/Frc Display Filter	Force Display Filter	Force Display Filter	Accel Display Filter	Accel Display Filter

%MD*f*.102	REAL			Prs/Frc Rate Display Filter	Force Rate Display Filter	Force Rate Display Filter	Jerk Display Filter	Jerk Display Filter
%MD*f*.103	REAL							
%MD*f*.104	REAL							
%MD*f*.105	REAL			Prs/Frc Input Filter	Force Input Filter	Force Input Filter	Accel Input Filter	Accel Input Filter
%MD*f*.106	REAL			Prs/Frc Rate Input Filter	Force Rate Input Filter	Force Rate Input Filter	Jerk Input Filter	Jerk Input Filter
%MD*f*.110	DINT			Model Type	Model Type	Model Type		
%MD*f*.111	REAL			Model Gain Prs/Frc	Model Gain Force	Model Gain Force		
%MD*f*.113	REAL			Model Time Constant	Model Time Constant	Model Time Constant		
%MD*f*.114	REAL			Model Natural Freq.	Model Natural Freq.	Model Natural Freq.		
%MD*f*.115	REAL			Model Damping Factor	Model Damping Factor	Model Damping Factor		
%MD*f*.119	DWORD			Custom FB Config			Custom FB Config	Custom FB Config
%MD*f*.120	DWORD			Analog Config	Load Cell Config	Analog Config	Analog Config	Analog Config
%MD*f*.121	REAL			Analog Overflow Limit	Load Cell Overflow Lim	Analog Overflow Limit	Analog Overflow Limit	Analog Overflow Limit
%MD*f*.122	REAL			Analog Underflow Limit	Load Cell Underflow Lim	Analog Underflow Limit	Analog Underflow Limit	Analog Underflow Limit
%MD*f*.123	REAL				Fixed Exciter Voltage			

Output

Internal IEC Address	Data Type	Description
Control (Common)		
%MD*f*.140	DINT	Output Type
%MD*f*.141	REAL	Output at 100%
%MD*f*.142	REAL	Output at 0%
%MD*f*.143	REAL	Output at -100%
%MD*f*.144	REAL	Output Limit

		Positive Output Limit
%MD*f*.145	REAL	Negative Output Limit
%MD*f*.146	REAL	Output Bias
%MD*f*.151	REAL	Deadband Tolerance
%MD*f*.152	REAL	Output Deadband
%MD*f*.153	REAL	Output Gain
%MD*f*.154	DINT	Unidirectional Mode
%MD*f*.155	DINT	Valve Linearization Type
%MD*f*.156	REAL	Knee Command Input
%MD*f*.157	REAL	Knee Flow Output
%MD*f*.158	DINT	Valve Linearization Curve ID
%MD*f*.170	DINT	Default Integrator Mode
%MD*f*.171	DINT	Default Pos/Vel Control Mode

Control (Primary and Secondary)

Internal IEC Address	Data Type	Position	Velocity	Pressure/Force
Primary Control				
%MD*f*.180	REAL	In Position Tolerance		At Pressure/Force Tolerance
%MD*f*.181	REAL	Position Error Tolerance		Pressure/Force Error Tolerance
%MD*f*.182	REAL	At Velocity Tolerance	At Velocity Tolerance	
%MD*f*.183	REAL	Velocity Error Tolerance	Velocity Error Tolerance	
%MD*f*.184	DINT	Symmetrical/Ratioed	Symmetrical/Ratioed	
%MD*f*.185	DINT	Gain Sets	Gain Sets	
%MD*f*.186				Pressure/Force Orientation
Primary Gain Set 0				
%MD*f*.200	REAL	Proportional Gain	Proportional Gain	Pressure/Force Proportional Gain
%MD*f*.201	REAL	Integral Gain	Integral Gain	Pressure/Force Integral Gain
%MD*f*.202	REAL	Differential Gain	Differential Gain	Pressure/Force Differential Gain
%MD*f*.203	REAL	Double Differential Gain	Double Differential Gain	
%MD*f*.204	REAL	Triple Differential Gain		
%MD*f*.205				Pressure/Force Feed Forward
%MD*f*.206	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)	Velocity Feed Forward, Velocity Feed Forward (Positive)	Pressure/Force Rate Feed Forward
%MD*f*.207	REAL	Velocity Feed Forward (Negative)	Velocity Feed Forward (Negative)	

%MD*f*.208	REAL	Acceleration Feed Forward	Acceleration Feed Forward	
%MD*f*.209	REAL	Jerk Feed Forward	Jerk Feed Forward	
%MD*f*.214	DINT	High-Order Control	High-Order Control	
%MD*f*.215	REAL	Active Damping Proportional Gain	Active Damping Proportional Gain	
%MD*f*.216	REAL	Active Damping Differential Gain	Active Damping Differential Gain	
%MD*f*.222	REAL	Output PD Filter	Output PD Filter	
Primary Gain Set 1				
%MD*f*.230	REAL	Proportional Gain	Proportional Gain	Pressure/Force Proportional Gain
%MD*f*.231	REAL	Integral Gain	Integral Gain	Pressure/Force Integral Gain
%MD*f*.232	REAL	Differential Gain	Differential Gain	Pressure/Force Differential Gain
%MD*f*.233	REAL	Double Differential Gain	Double Differential Gain	
%MD*f*.234	REAL	Triple Differential Gain		
%MD*f*.235				Pressure/Force Feed Forward
%MD*f*.236	REAL	Velocity Feed Forward, Velocity Feed Forward (Positive)	Velocity Feed Forward, Velocity Feed Forward (Positive)	Pressure/Force Rate Feed Forward
%MD*f*.237	REAL	Velocity Feed Forward (Negative)	Velocity Feed Forward (Negative)	
%MD*f*.238	REAL	Acceleration Feed Forward	Acceleration Feed Forward	
%MD*f*.239	REAL	Jerk Feed Forward	Jerk Feed Forward	
%MD*f*.244	DINT	High-Order Control	High-Order Control	
%MD*f*.245	REAL	Active Damping Proportional Gain	Active Damping Proportional Gain	
%MD*f*.246	REAL	Active Damping Differential Gain	Active Damping Differential Gain	
%MD*f*.252	REAL	Output PD Filter	Output PD Filter	
Secondary Control				
%MD*f*.290	REAL			At Pressure/Force Tolerance
%MD*f*.291	REAL			Pressure/Force Error Tolerance
%MD*f*.296	REAL			Pressure/Force Orientation
Secondary Gain Set 0				
%MD*f*.310	REAL			Pressure/Force Proportional Gain
%MD*f*.311	REAL			Pressure/Force Integral Gain
%MD*f*.312	REAL			Pressure/Force Differential Gain

%MD*f*.315	REAL			Pressure/Force Feed Forward
%MD*f*.316	REAL			Pressure/Force Rate Feed Forward

Target

Internal IEC Address	Data Type	Position	Velocity	Pressure/Force
%MD*f*.400	REAL	Positive Travel Limit		
%MD*f*.401	REAL	Negative Travel Limit		
%MD*f*.402	DINT	Target Type	Target Type	
%MD*f*.403	REAL	Requested Jerk	Requested Jerk	
%MD*f*.440				Positive Pressure/Force Limit
%MD*f*.441				Negative Pressure/Force Limit

Simulator

Internal IEC Address	Data Type	Position
%MD*f*.480	DINT	Simulate Mode
%MD*f*.481	DINT	Simulator Order/Type
%MD*f*.482	REAL	Positive System Gain
%MD*f*.483	REAL	Negative System Gain
%MD*f*.484	REAL	Time Constant (1st order)
%MD*f*.485	REAL	Natural Frequency (2nd order)
%MD*f*.486	REAL	Damping Factor (2nd order)
%MD*f*.487	REAL	Positive Physical Limit
%MD*f*.488	REAL	Negative Physical Limit
%MD*f*.489	REAL	Output Deadband
%MD*f*.490	REAL	Output Null
%MD*f*.491	REAL	Weight
%MD*f*.492	REAL	Maximum Force
%MD*f*.493	REAL	Maximum Compression

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Show All Back

RMC200 Registers, Files 512-575: Plot Status/Configuration

The following files contain the Plot Configuration registers. See the Reading Plots with a Host Controller topic for details on how some of them can be used.

n = plot # (0-63)

The CPU20L supports 48 plots. The CPU40 supports 64 plots with firmware 1.14.0 or newer, and 32 plots with older firmware.

Internal IEC Address	Data Type	Access	Register Name
%MD512+*n*.0	DWORD	Read Only	Plot Flags These bits should not be accessed directly. bit 3:Read Active bit 4:Trigger Enabled
%MD512+*n*.1	DINT	Read/Write	Plot Samples (e.g. 1000)
%MD512+*n*.2	REAL	Read/Write	Plot Sample Period (seconds)
%MD512+*n*.3	DINT	Read/Write	Plot Axis Owner 0-127: Axis 0..127 -1: none
%MD512+*n*.4	REAL	Read/Write	Plot Trigger Position %, 0-100, -1 = auto rearm
%MD512+*n*.5	DWORD	Read/Write	Plot Trigger Type Bits 0-7:Trigger Type (0=none, 1=motion command)
%MD512+*n*.6	-	-	Reserved
%MD512+*n*.7	-	-	Reserved
%MD512+*n*.8	DINT	Read Only	Plot ID
%MD512+*n*.9	DINT	Read Only	Plot State 0 = not triggered, 1 = capturing, 2 = complete
%MD512+*n*.10	DINT	Read Only	Plot Captured Samples Number of plot samples captured. Only applies for Plot State > 0.
%MD512+*n*.11	DINT	Read Only	Plot Sample 0 Time Time that first plot sample was captured. In control loops since controller startup (low 24 bits). Only applies for Plot State > 0.
%MD512+*n*.12	DINT	Read Only	Plot Trigger Time Time that plot trigger occurred. In control loops since controller startup (low 24 bits). Only applies for Plot State > 0.
%MD512+*n*.13	DINT	Read Only	Plot Trigger Index Index of the plot sample at which the plot trigger occurred. Only applies for Plot State > 0.
%MD512+*n*.14-15	-	-	Reserved
%MD*f*512+*n*.16-143	DWORD	Read/Write	Plot Data Sets 0-127 Addresses The CPU20L supports 48 data items per plot. The CPU40 supports 128 data items per plot (32 prior to firmware 1.14.0).

			Elements 16-143 contain the addresses for plot data sets 0-127. Bits 0-11 Element Bits 12-23File

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, Files 576-639: Dynamic Plot Upload Area

The following files contain the Dynamic Plot Upload Area registers. See the Reading Plots with a Host Controller topic for details on how to use them.

n = plot # (0-63)

The CPU20L supports 48 plots. The CPU40 supports 64 plots with firmware 1.14.0 or newer, and 32 plots with older firmware.

Internal IEC Address	Data Type	Access	Register Name
%MD576+*n*.0	UDINT	Read/Write	Plot 0 Upload Mode/Status
%MD576+*n*.1	UDINT	Read/Write	Plot 0 Requested Read Samples
%MD576+*n*.2	UDINT	Read/Write	Plot 0 Current Index
%MD576+*n*.3	UDINT	Read Only	Plot 0 ID
%MD576+*n*.4	UDINT	Read Only	Plot 0 Samples Uploaded
%MD576+*n*.5-4096	*	Read Only	Plot 0 Data

*The data types of the Plot Data are determined by the plotted registers.

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, Files 640-703: Static Plot Upload Area

All Static Plot Upload Area Registers are **Read Only**. See the Reading Plots with a Host Controller topic for details on how to use them.

The data types of the plot data is determined by the plotted registers.

Internal IEC Address	Register Name
%MD640.0-4095	Plot 0, Data Set 0, Samples 0-4095
%MD641.0-4095	Plot 0, Data Set 1, Samples 0-4095
...	...

%MD655.0-4095	Plot 0, Data Set 15, Samples 0-4095
%MD656-671	Plot 1, Data Sets 0-15, Samples 0-4095
%MD672-687	Plot 2, Data Sets 0-15, Samples 0-4095
%MD688-703	Plot 3, Data Sets 0-15, Samples 0-4095

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing

Back

RMC200 Registers, Files 1024, 1280, 1536: Variables Registers

All variable registers are **Read/Write**.

Internal IEC Address	Data Type	Register Name
Variables - Current Values		
%MD1024.0-4095	*	Variables 0-4095 - Current Values
Variables - Initial Values		
%MD1280.0-4095	*	Variables 0-4095 - Initial Values
Variables - Attributes		
%MD1536.0-4095	DINT	Variables 0-4095 - Attributes Users will typically never access the attributes.

* The data types of the variables are specified by the user when defining a variable in the Variable Table.

Tag Names

The variables can be given a user-defined tag name in the Variable Table Editor, which is the preferred method of referencing variables from within RMCTools. The user-defined variable name references the variable's Current Value.

Otherwise, each variable has a tag name as follows:

_VarTbl.CurVal[n] - Current Value of variable n.

_VarTbl.Initial[n] - Initial value of variable n.

See Also

RMC200 Register Map | Modbus Addressing | DF1 Addressing | FINS Addressing | Variable Attributes

10.Wiring and Installation

10.1. Wiring Guidelines

Proper wiring of the RMC and of the system is important for proper machine control. Poor wiring is a common source of noisy feedback, drive signals or digital I/O. Follow the guidelines in this topic and the other wiring topics to ensure a low-noise system.

Specific Wiring Instructions

Each RMC module has specific wiring diagrams. Follow the links below for wiring details:

RMC75 Wiring	RMC150 Wiring	RMC200 Wiring
RMC75E	RMC150E	PS4D Wiring
RMC75S	Analog modules	PS6D Wiring
RMC75P	MDT	CPU40 Wiring
AA	SSI	CPU20L Wiring
MA	Quadrature	CA4 Wiring
QA	Discrete I/O	CV8 Wiring
A2	Resolver	S8 Wiring
AP2	Universal I/O	A8 Wiring
D8		LC8 Wiring
Q1		Q4 Wiring
		U14 Wiring
		D24 Wiring

General Wiring Instructions

For CE compliance and to minimize electrical interference:

- Use twisted pairs for all wiring where possible.
- Use shielded cables for all wiring.
- Keep RMC wiring separate from AC mains or conductors carrying high currents, especially high frequency switching power such as conductors between servo drives and motors or amplifiers and proportional valves.

For UL and CUL compliance:

- **RMC75/RMC150:** Power supply must be Class 2.
- **RMC75/RMC150:** All RMC inputs and outputs must be connected to Class 2 circuits only.

Fusing

No fusing is required if a Class 2 power supply is used, as required by UL. Class 2 power supplies are limited to 100W output. This provides sufficient protection for the RMC and no additional fusing is required.

For Class I, Division 2 compliance (only available for RMC150E with the Class I, Division 2 designation)

- The RMC150E USB port is intended for configuration, programming, and troubleshooting purposes only. Do not leave it connected during normal machine operation, as it is sensitive to electrical noise.

- Conductors must be copper only.

Terminal Block Wires and Clamp Screw Torque

Tighten the wire clamp screws on the terminal blocks to:

Controller	Module	AWG	Torque
RMC75	all modules	26-12	7 lb-in (0.8 Nm)
RMC150	RMC150E, MDT (M), SSI (S), ANLG (H), ANLG INPUTS (A), ANLG2 (G), RES (R)	26-12	4.5 lb-in (0.51 Nm)
	RMC150E DI/O, UI/O	28-16	2.2 lb-in (0.25 Nm)
RMC200	PS4D, PS6D	24-12	n/a (spring connector)
	CPU40L (DI/O), CPU20L (power input and DI/O), A8, S8, LC8, Q4, CA4, CV8, D24, U14	24-16	n/a (spring connector)

Using Spring-Cage Connectors (RMC200)

Spring-cage connectors may be used for stranded wire or stranded wire with ferrules. Wire ferrules provide easy insertion. Recommended wire ferrule spec are as follows:

	PS4D, PS6D	CPU20L, CPU40, CA4, CV8, S8, A8, Q4, D24, U14
Wire Gauge, stranded	24 -12 AWG 0.2mm^2 - 1.5mm^2	24 -16 AWG 0.2mm^2 - 1.5mm^2
Wire Gauge, ferrule no plastic sleeve	0.25mm^2 - 2.5mm^2	0.25mm^2 - 1.5mm^2
Wire Gauge, ferrule with plastic sleeve	0.25mm^2 - 2.5mm^2	0.25mm^2 - 0.75mm^2
Stripped length	10mm	10mm
Ferrule Length	10mm - 12mm	10mm - 12mm

Inserting stranded wire:

1. Press and hold the spring clamp actuator.
2. Insert wire.
3. Release spring clamp actuator.

Inserting wire with ferrule:

1. Insert wire (may require some force).

Removing wire (stranded or ferrule):

1. Press and hold the spring-cage actuator.

2. Remove wire.
3. Release spring-cage actuator.

See Also

RMC75 Mounting Instructions | RMC150 Mounting Instructions | RMC200 Mounting Instructions | Agency Compliance

10.2. RMC75

10.2.1. RMC75 Mounting Instructions

Mounting Options:

- Symmetrical DIN 3
- Panel-mount

Orientation:

The RMC should be mounted upright on a vertical surface, such that the air holes are on top and bottom.

Clearance above and below:

The amount of clearance required depends on the maximum ambient temperature:

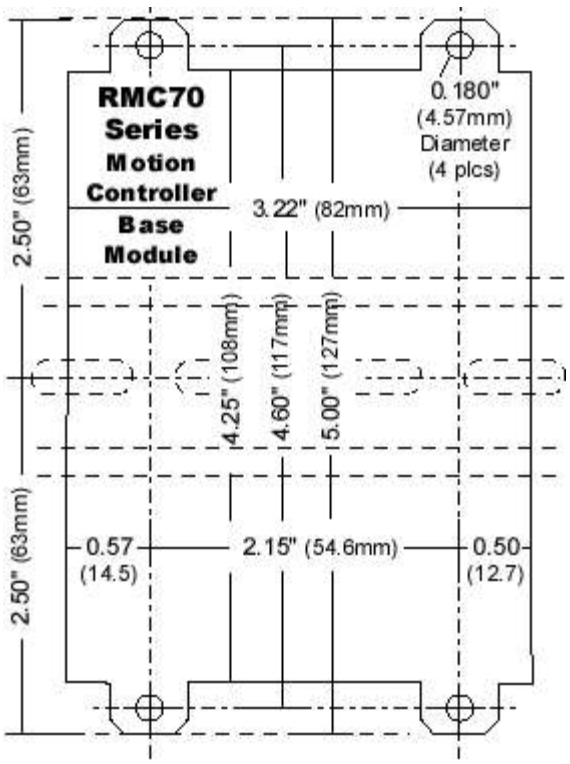
Ambient Temperature	Clearance
122 - 140°F (50-60°C)	3 in (7.6 cm)
86 to 122°F (30 to 50°C)	2 in (5.1 cm)
Less than 86°F (30°C)	1 in (2.5 cm)

Mounting Dimensions

Base Module:

3.22 in. x 5.0 in.

Protrudes 2.0 in. Make sure to leave room for the front connectors.

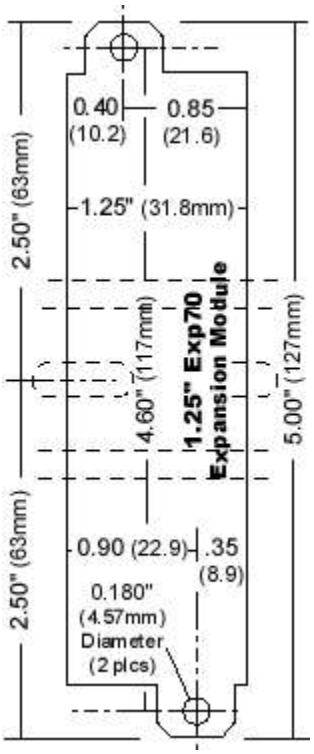


Expansion Modules

Protrude 2.0 in. Make sure to leave room for the front connectors.

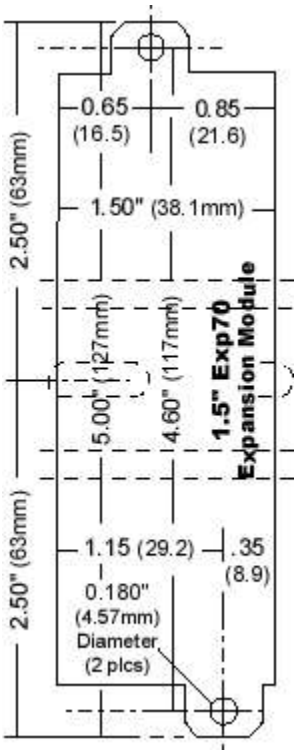
D8 module

1.25 in. x 5.0 in.



AP2, A2, Q1 modules

1.50 in. x 5.0 in.

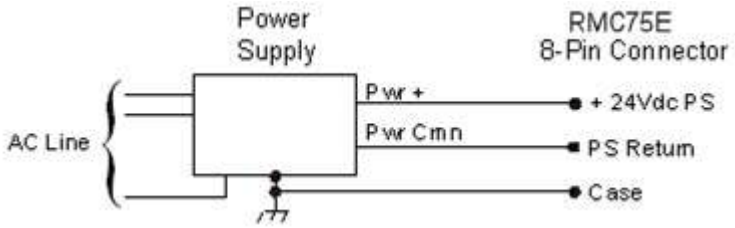


See Also
Wiring Guidelines

10.2.2. RMC75E Wiring

The RMC75E CPU module contains a connector for power, and two connectors for communications.

Wiring Power



The connector screws and wire clamp screws must be tightened to max 7 b-in (0.8Nm).

Wiring the Communications

USB Monitor Port

Use a standard USB cable to connect to the USB port. USB connector shielding is grounded with case ground.

RJ-45 Ethernet connector

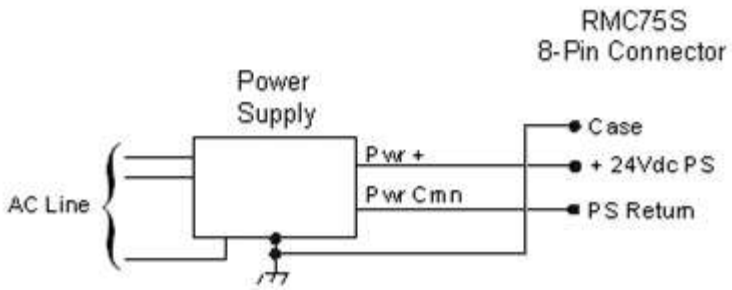
Use a standard Category 5, 5e, or 6 cable with an RJ-45 connector to connect to the 10/100 Ethernet port.

See Also
RMC75E Module | Ethernet Overview

10.2.3. RMC75S Wiring

The RMC75S CPU module contains a connector for power and RS-485 communications, and two connectors for RS-232 communications.

Wiring Power



Wire clamp screws must be tightened to max 7 b-in (0.8Nm).

Wiring the Communications

For wiring RS-232 communications and the RS-232 Monitor Port, see the RS-232 Wiring for the RMC75 topic. For wiring RS-485, see the RS-485 Wiring topic.

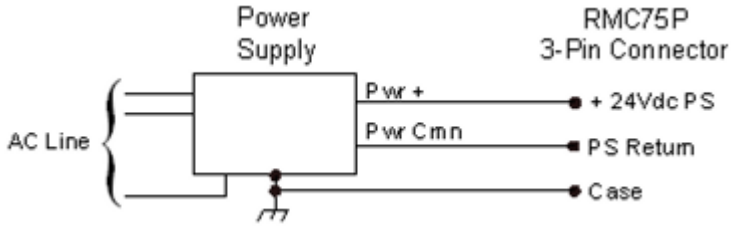
See Also

RMC75S Module

10.2.4. RMC75P Wiring

The RMC75P CPU module contains a connector for power, and two connectors for communications.

Wiring Power



The connector screws and wire clamp screws must be tightened to max 7 b-in (0.8Nm).

Wiring the Communications

RS-232 Monitor Port

For wiring the RS-232 Monitor Port, see the RS-232 Wiring for the RMC75 topic.

9-Pin PROFIBUS-DP Connector

This connector is the standard PROFIBUS connector. It is used for connecting the RMC75 to other PROFIBUS-DP devices via a standard PROFIBUS-DP cable. The pin assignment is defined by the PROFIBUS specification.

See Also

RMC75P Module

Show All

10.2.5. AA Wiring

The AA axis module can be wired to voltage or current feedback transducers. The AA module also a Fault input, Enable output, and a Control Output. If the AA module has two axes, each axis' pin-out is identical.

Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers. See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 7 b-in (0.8Nm).

NOTE:
The example schematics do not include transducer pin numbers, color codes, or power supply requirements, since these vary between different transducers. To determine your power supply needs and connector pin-outs or cable color codes, consult your transducer manufacturer's documentation.

NOTE:
If the input is disconnected, input voltage will be pulled down ≤ -10V.

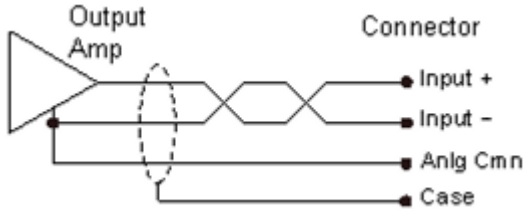
Pin-Out

Pin #	AA1 Label	AA2 Label	Function
1	+Fault In	Flt In+	Fault Input
2	-Fault In	Flt In-	
3	+Enable Out	En Out+	Enable Output
4	-Enable Out	En Out-	
5	Control Out	Ctrl Out	Control Output
6	Common	Cmn	Common
7	+Analog In	An In+	+Analog Input
8	Jumper for 4-20mA	Jmpr for 4-20mA	Jumper for 4-20mAfeedback
9	-Analog In	An In	-Analog In
10	Common	Cmn	Common
11	+10Vdc Exciter Out	+10Vdc Exciter	+10Vdc output for potentiometers
12	Case	Case	Connected to

The commons are internally connected.

Voltage Feedback Transducers

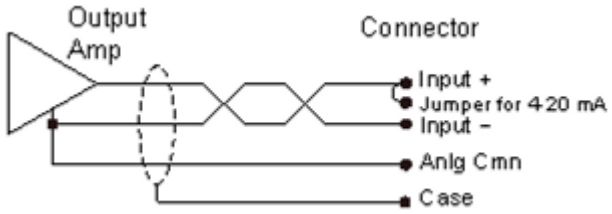
Voltage feedback transducers can be connected directly to the **Input +** and **Input -** connections of the desired axis. The Cmn *must* be connected to the transducer, or the signal will not be read correctly! The following configuration is recommended:



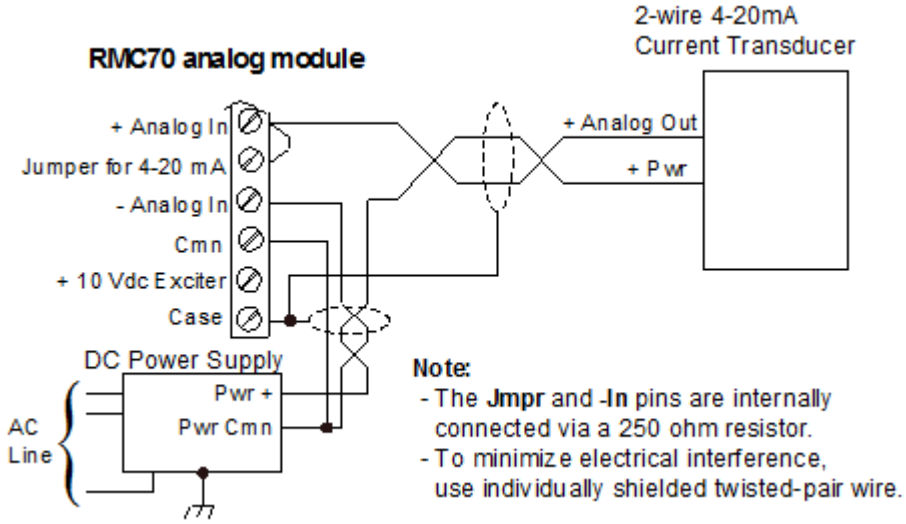
Current Feedback Transducers

Current feedback transducers are connected in the same way as voltage transducers except that a jumper must be inserted between the **Input+** and **Jumper for 4-20 mA** pins. The label indicates where this jumper should be connected. This places a resistor internal to the RMC across the two inputs, thus converting the current to a voltage input. The wiring diagram below shows a suggested configuration.

The Cmn *must* be connected to the transducer, or the signal will not be read correctly!



2-Wire Current Transducer



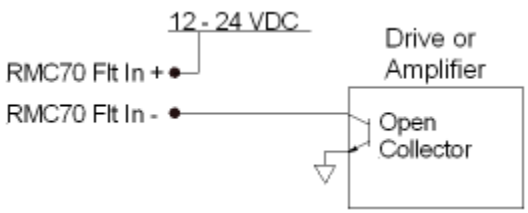
Fault Input Wiring

The Fault input is compatible with signal levels ranging from 12V to 24V. The Fault Input draws 2.7mA maximum and turns on at 6V. The Fault input turns on when a current flows. The polarity of the voltage is unimportant. See the Fault Input topic for more details.

Fault input wiring diagrams:

Generic:

From Open Collector Output:



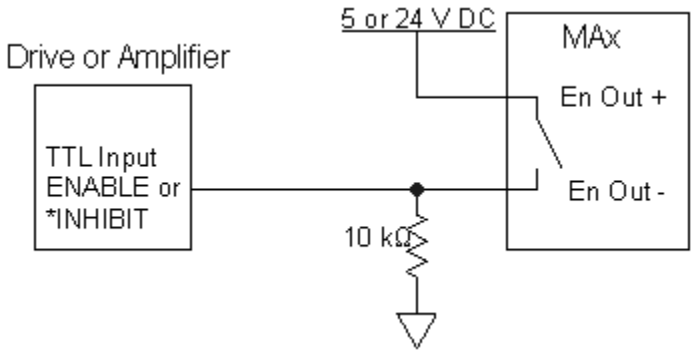
Enable Output Wiring

The Enable output is a solid state relay (SSR). When it is off, it has high impedance, and when on it has low impedance (10 Ω maximum). Because the Enable output is isolated, the user must power it externally. The maximum current and voltage for the Enable output is 100 mA and 30 V.

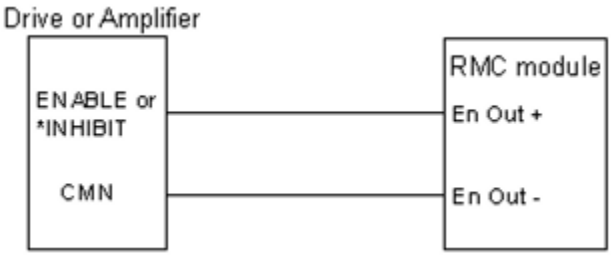
The Enable output has a + and - connection. Both lines must be connected for the output to function. Because both sides of the output are provided, the switches may be independently connected in a high side or low side configuration (that is, with the load (input) on the source or sinking side of the output). See the wiring diagrams below.

See the Enable Output topic for more details on the Enable Output.

To TTL input (high = enable):



To active low Enable input:



Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an "ON" to an "OFF" state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR.

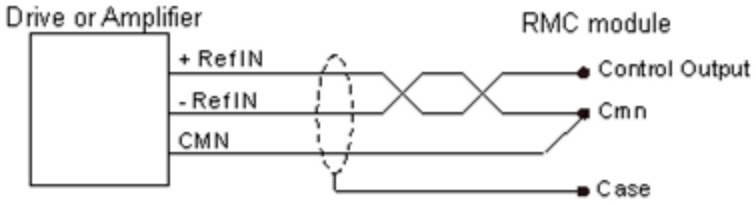
Control Output Wiring

The valve or motor drive connects to the following MA*x* pins:

Pin	Function
Control Output	Control Output
Cmn	Control Output Common. Each axis has two Cmn pins. Either of the 2 pins may be used.

See the Control Output topic for more details on the Control Output.

Note:
The Control Output polarity can be set with the Invert Output Polarity parameter.



See Also

AA module (RMC75)

10.2.6. MAx Wiring

The MA*x* module can be wired to MDT and SSI transducers. Each axis on the MAx also has a Fault input, Enable output, and a Control output. If the MA module has two axes, each axis' pinout is identical.

Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers. See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 7 b-in (0.8Nm).

NOTE:
The following example schematics do not include transducer pin numbers, color codes, or power supply requirements, since these vary between different transducers. To determine your power supply needs and connector pin-outs or cable color codes, consult your transducer manufacturer's documentation.

Pin-Out

Pin #	MA1 Label	MA2 Label	Function
1	+Fault In	Flt In+	Fault Input
2	-Fault In	Flt In-	
3	+Enable Out	En Out+	Enable Output
4	-Enable Out	En Out-	
5	Control Out	Ctrl Out	Control Output
6	Common	Cmn	Common
7	+Int/Clock	Int/Clk+	MDT Interrogation or SSI Clock
8	-Int/Clock	Int/Clk-	
9	Common	Cmn	Common
10	+Ret/Data	Ret/Dat+	MDT Return or SSI Data
11	-Ret/Data	Ret/Dat-	
12	Case	Case	Connected to RMC Chassis

The commons are internally connected.

MDT Wiring (see below for SSI)

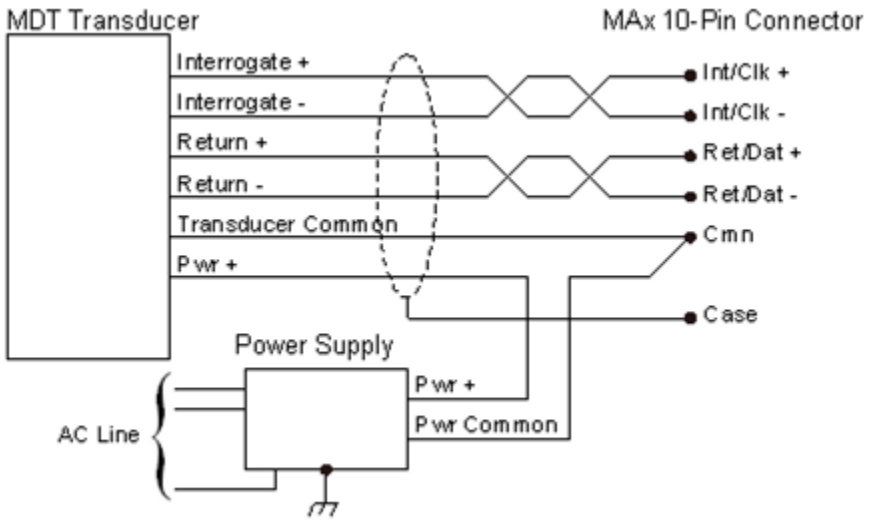
The MAx modules interface only to transducers with Differential Line Driver (RS422) signals. Single-ended (TTL) transducers are *not* supported due to low noise immunity. MDT transducers connect to the following MA*x* pins:

Pin	Function	Additional Possible Manufacturer Designation
Int/Clk +	MDT + Interrogation	Interrogate + Input
Int/Clk -	MDT - Interrogation	Interrogate - Input
Cmn	MDT Common	
Ret/Dat +	MDT + Return	Pulse + Output
Ret/Dat -	MDT - Return	Pulse - Output
Case	RMC75 Chassis	

Wiring instructions:

- Connect the '+Int' and '-Int' between the transducer and the MA module for the interrogation signal.
- Connect the '+Ret' and '-Ret' between the transducer and the MA module for the return signal.
- Connect the transducer DC ground to Cmn. Each axis has two Cmn pins. Either of the 2 pins may be used. The Cmn *must* be connected to the transducer, or the signals will not be read correctly!
- The transducer power supply is not provided by the RMC. The user must supply a power supply. The power supply common may be connected to the same common that is connected to the RMC75 CPU Cmn.

Diagram



SSI Wiring

SSI uses differential line driver (RS422) clock and data signals. Connect the transducer DC ground to SSI Cmn. SSI transducers connect to the following MA*x* pins:

Pin	Function
Int/Clk +	SSI + Clock
Int/Clk -	SSI - Clock
Cmn	Common
Ret/Dat +	SSI + Data
Ret/Dat -	SSI - Data

Case	RMC75 Chassis

Max Cable Length

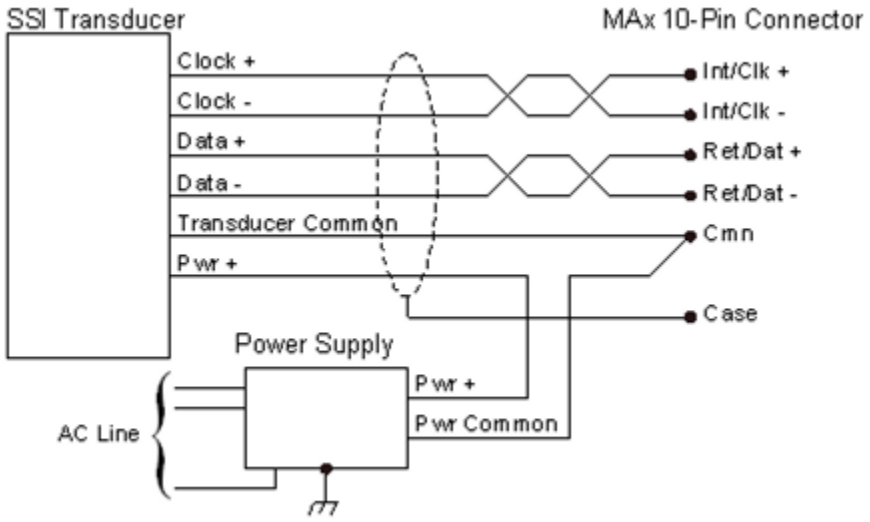
SSI Clock Rate	Maximum Cable Length*
150 kHz	1360 ft (415 m)
250 kHz	770 ft (235 m)
375 kHz	475 ft (145 m)

* The cable lengths are approximate, and may be affected by the type of wire and transducer.

Wiring Instructions

- Connect both the +Clock and -Clock between the MA module and transducer for the clock signal.
- Connect both the +Data and -Data between the MA module and transducer for the data signal.
- Connect the transducer DC ground to Cmn. Each axis has two Cmn pins. Either of the 2 pins may be used. The Cmn *must* be connected to the transducer, or the signals will not be read correctly!
- The transducer power supply is not provided by the RMC. The user must supply a power supply. The power supply common may be connected to the same common that is connected to the RMC75 CPU Cmn.

Diagram



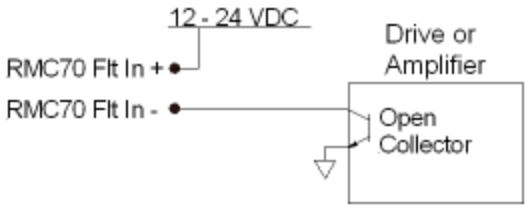
Fault Input Wiring

The Fault input is compatible with signal levels ranging from 12V to 24V. The Fault Input draws 2.7mA maximum and turns on at 6V. The Fault input turns on when a current flows. The polarity of the voltage is unimportant. See the Fault Input topic for more details.

Fault input wiring diagrams:

Generic:

From Open Collector Output:



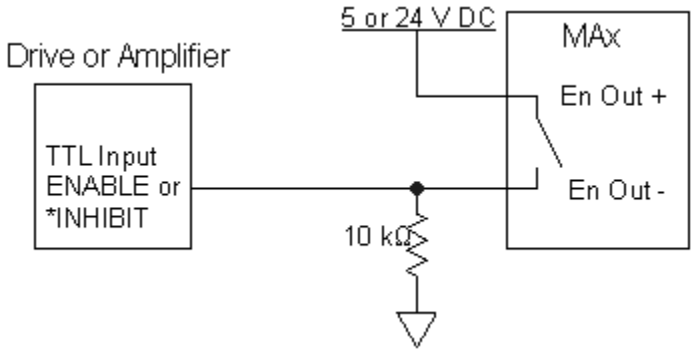
Enable Output Wiring

The Enable output is a solid state relay (SSR). When it is off, it has high impedance, and when on it has low impedance (10 Ω maximum). Because the Enable output is isolated, the user must power it externally. The maximum current and voltage for the Enable output is 100 mA and 30 V.

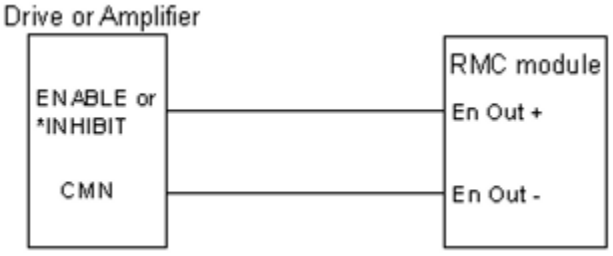
The Enable output has a + and - connection. Both lines must be connected for the output to function. Because both sides of the output are provided, the switches may be independently connected in a high side or low side configuration (that is, with the load (input) on the source or sinking side of the output). See the wiring diagrams below.

See the Enable Output topic for more details on the Enable Output.

To TTL input (high = enable):



To active low Enable input:



Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an "ON" to an "OFF" state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR.

Control Output Wiring

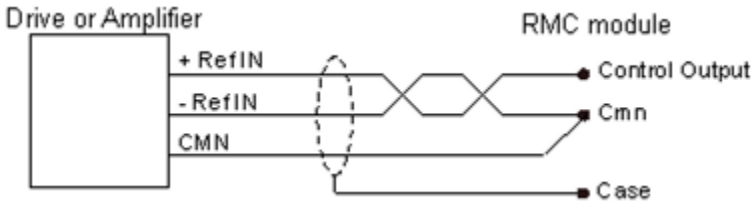
The valve or motor drive connects to the following MA*x* pins:

Pin	Function
Control Output	Control Output

Cmn	Control Output Common. Each axis has two Cmn pins. Either of the 2 pins may be used.

See the Control Output topic for more details on the Control Output.

Note:
The Control Output polarity can be set with the Invert Output Polarity parameter.



See Also

MA module (RMC75)

Show All

10.2.7. QAx Wiring

The QA*x* module can be wired to quadrature encoders. Each axis on the QAx also has a Fault input, Enable output, Control output, two high-speed registration inputs, and a high-speed home input. The pinout of each axis is identical.

Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers. See Wiring Guidelines for general wiring information.

NOTE:
The following example schematics do not include transducer pin numbers, color codes, or power supply requirements, since these vary between different transducers. To determine your power supply needs and connector pin-outs or cable color codes, consult your transducer manufacturer's documentation.

Pin-Out

The RMC-CB-QUAD-01-xx cable is an optional accessory for the QA module.

Pin #	QA1 Label	QA2 Label	Function	RMC-CB-QUAD-01-xx Wire Color
1	A-	A-	A- from encoder (5 V)	Enc: white/blue
2	A+	A+	A+ from encoder (5 V)	Enc: blue/white
3	B-	B-	B- from encoder (5 V)	Enc: white/orange
4	B+	B+	B+ from encoder (5 V)	Enc: orange/white
5	n/c	n/c	No connection	
6	Reg Y/NegLim-	RY/NL-	Registration Y or Negative Limit (12-24 VDC)	Lim: white/orange
7	Reg Y/NegLim+	RY/NL+		Lim: orange/white

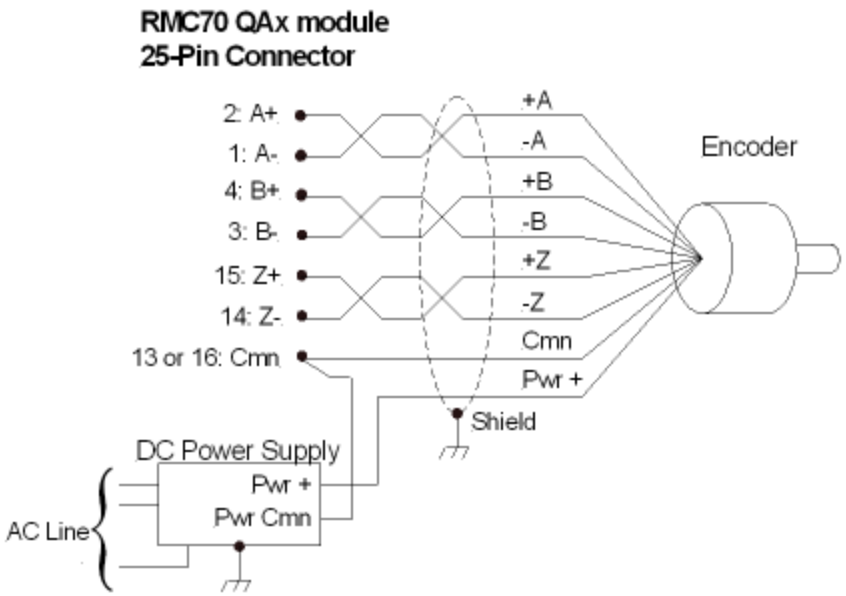
8	Reg X/PosLim-	RX/PL-	Registration X or Positive Limit (12-24 VDC)	Lim: white/blue
9	Reg X/PosLim+	RX/PL+		Lim: blue/white
10	n/c	n/c	No connection	
11	n/c	n/c	No connection	
12	Control Out	CtrlOut	Control Output	Drv: blue/white
13	Cmn	Cmn	Common	Drv: white/blue
14	Z-	Z-	Index pulse from encoder (5 V)	Enc: white/green
15	Z+	Z+		Enc: green/white
16	Cmn	Cmn	Common	Enc: white/brown Enc: brown/white
17	n/c	n/c	No connection	
18	Home-	Home-	Home Input (12-24 VDC)	Lim: white/green
19	Home+	Home+		Lim: green/white
20	FltIn-	FltIn-	Fault Input (12-24 VDC)	Drv: white/green
21	FltIn+	FltIn+		Drv: green/white
22	n/c	n/c	No connection	
23	n/c	n/c	No connection	
24	EnOut-	EnOut-	Enable Output (12-24 VDC)	Drv: white/orange
25	EnOut+	Enout+		Drv: orange/white

Encoder Wiring

Quadrature Encoders

The QAx module A, B, and Z inputs accept only Differential Line Driver (RS-422) signals. Single-ended line drivers (TTL) are *not* supported due to low noise immunity. The user must supply power for the encoder.

The Cmn on the QA module *must* be connected to the encoder, or the signals will not be read correctly!

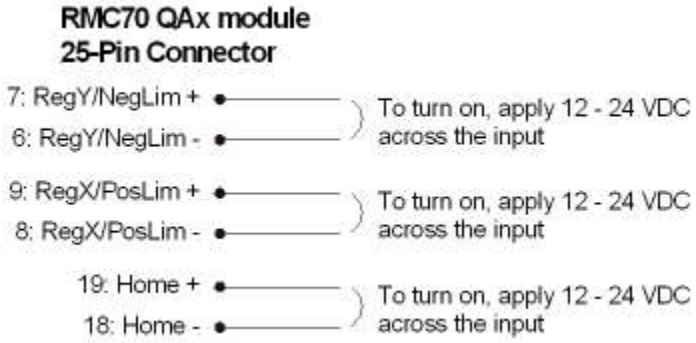


Daisy-Chaining A and B Quadrature Inputs

One quadrature encoder can typically output its signals to multiple QA modules. Use a daisy-chain topology, and add termination only to the last input. Incorrect termination will result in distorted signals and will cause incorrect transducer readings. Use the Input Termination Axis Parameter to set the termination. Termination is selectable on the A and B signals. The Z signal is always terminated with 120 Ω.

RegX/PosLim, RegY/NegLim, and Home Inputs

The Reg/Lim and Home inputs are compatible with signal levels from 12 to 24 V. They draw 2.7 mA max and turn on when the voltage across the input is greater than 6 V. The polarity is unimportant.



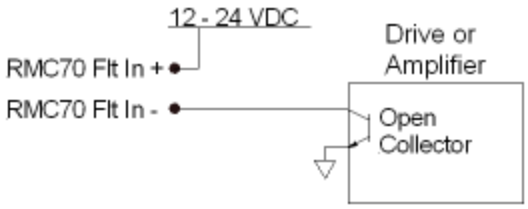
Fault Input Wiring

The Fault input is compatible with signal levels ranging from 12V to 24V. The Fault Input draws 2.7mA maximum and turns on at 6V. The Fault input turns on when a current flows. The polarity of the voltage is unimportant. See the Fault Input topic for more details.

Fault input wiring diagrams:

Generic:

From Open Collector Output:



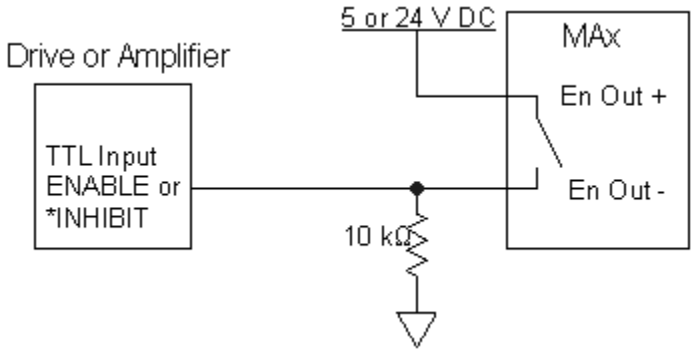
Enable Output Wiring

The Enable output is a solid state relay (SSR). When it is off, it has high impedance, and when on it has low impedance (10 Ω maximum). Because the Enable output is isolated, the user must power it externally. The maximum current and voltage for the Enable output is 100 mA and 30 V.

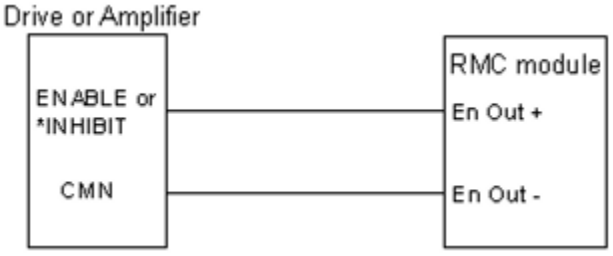
The Enable output has a + and - connection. Both lines must be connected for the output to function. Because both sides of the output are provided, the switches may be independently connected in a high side or low side configuration (that is, with the load (input) on the source or sinking side of the output). See the wiring diagrams below.

See the Enable Output topic for more details on the Enable Output.

To TTL input (high = enable):



To active low Enable input:



Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an "ON" to an "OFF" state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR.

Control Output Wiring

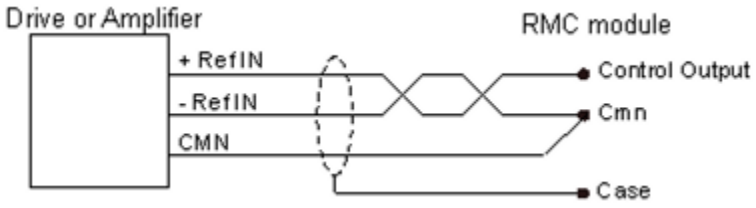
The valve or motor drive connects to the following MA*x* pins:

Pin	Function
Control Output	Control Output

Cmn	Control Output Common. Each axis has two Cmn pins. Either of the 2 pins may be used.

See the Control Output topic for more details on the Control Output.

Note:
The Control Output polarity can be set with the Invert Output Polarity parameter.



See Also
QA module (RMC75)

Show All

10.2.8. A2 Wiring

The A2 expansion module can be wired to voltage or current feedback transducers.

Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers. See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 7 lb-in (0.8Nm).

The commons are internally connected.

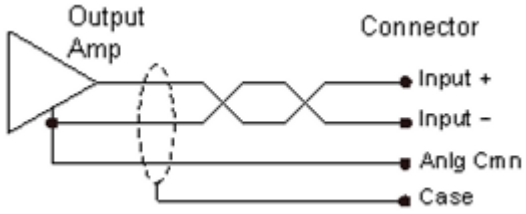
NOTE:
The example schematics do not include transducer pin numbers, color codes, or power supply requirements, since these vary between different transducers. To determine your power supply needs and connector pin-outs or cable color codes, consult your transducer manufacturers documentation.

NOTE:
If the input is disconnected, input voltage will be pulled down ≤ -10V.

Wiring Diagrams and Instructions

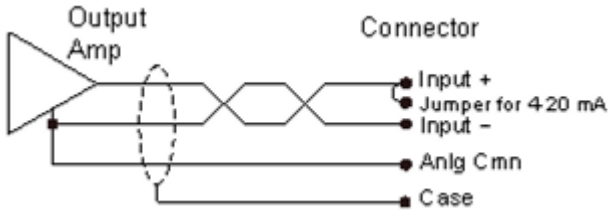
Voltage Feedback Transducers

Voltage feedback transducers can be connected directly to the **Input +** and **Input -** connections of the desired axis. The **Anlg Cmn** and **Case** pins may be shared by the axes. The following configuration is recommended:

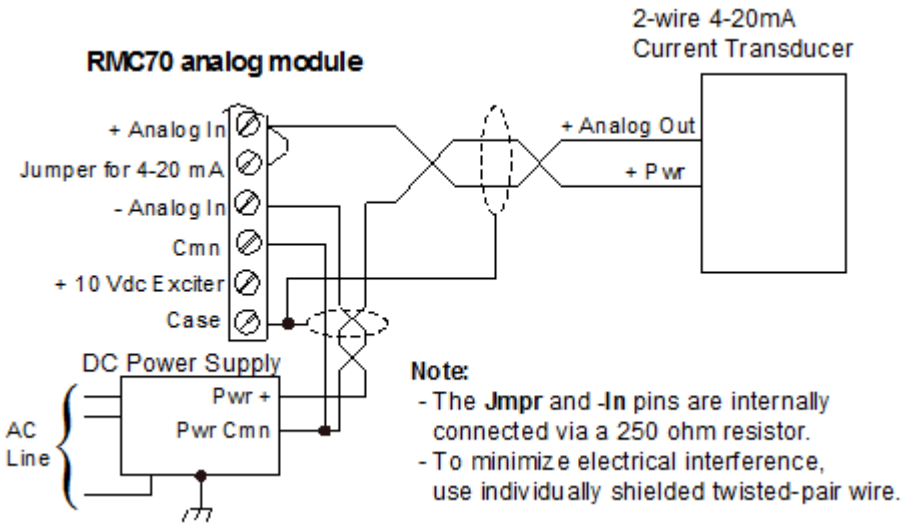


Current Feedback Transducers

Current feedback transducers are connected in the same way as voltage transducers except that a jumper must be inserted between the **Input+** and **Jumper for 4-20 mA** pins. The label indicates where this jumper should be connected. This places a resistor internal to the RMC across the two inputs, thus converting the current to a voltage input. The following wiring diagram shows a suggested configuration:



2-Wire Current Transducer



See Also

A2 module (RMC7)

10.2.9. AP2 Wiring

The AP2 expansion module can be wired to voltage or current feedback transducers.

Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers. See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 7 lb-in (0.8Nm).

All commons are internally connected.

NOTE:
The example schematics do not include transducer pin numbers, color codes, or power supply

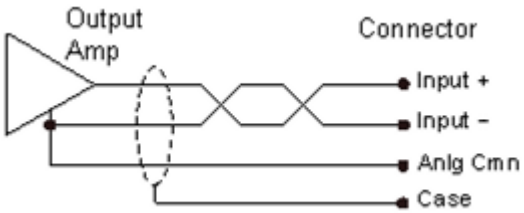
requirements, since these vary between different transducers. To determine your power supply needs and connector pin-outs or cable color codes, consult your transducer manufacturers documentation.

NOTE:
If the input is disconnected, input voltage will be pulled down ≤ -10V.

Wiring Diagrams and Instructions

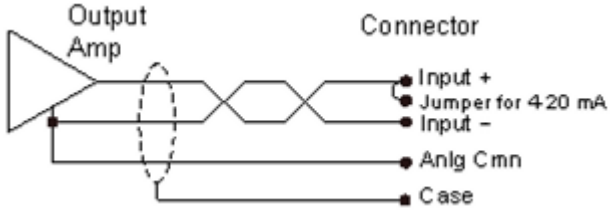
Voltage Feedback Transducers

Voltage feedback transducers can be connected directly to the **Input +** and **Input -** connections of the desired axis. The **Anlg Cmn** and **Case** pins may be shared by the axes. The following configuration is recommended:

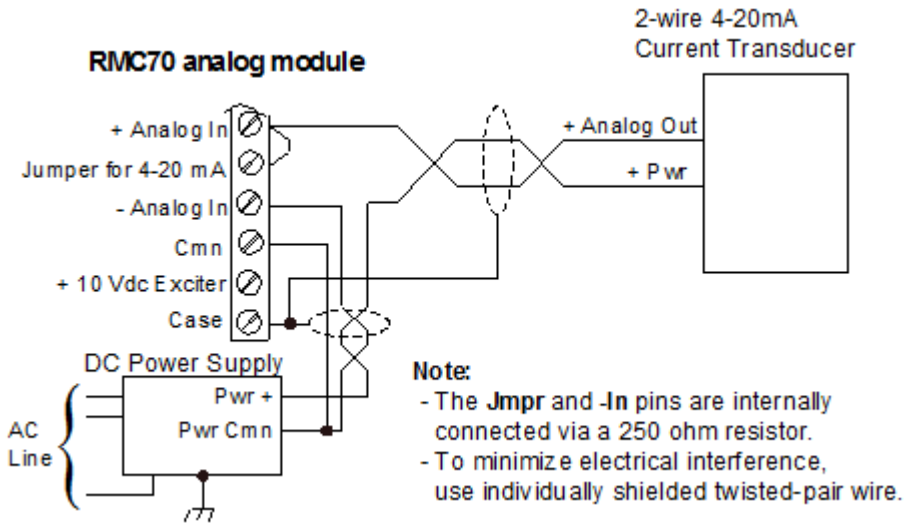


Current Feedback Transducers

Current feedback transducers are connected in the same way as voltage transducers except that a jumper must be inserted between the **Input+** and **Jumper for 4-20 mA** pins. The label indicates where this jumper should be connected. This places a resistor internal to the RMC across the two inputs, thus converting the current to a voltage input. The following wiring diagram shows a suggested configuration:



2-Wire Current Transducer



See Also

AP2 module (RMC75)

10.2.10. D8 Wiring

Each discrete I/O point on the D8 expansion module is individually configurable in software as an input or output. Since there is just one input common and one output common, all inputs must be the same polarity, and all outputs must be the same polarity, but inputs need not be the same polarity as outputs, that is, outputs can switch high side or low side, and inputs can be active high or low.

Inputs and outputs are 24Vdc rated, and optically isolated from controller. Since all inputs share a common connection, there is no isolation between input points. By the same token, all outputs share a common pin and therefore do not have isolation between outputs.

Wire clamp screws must be tightened to max 7 b-in (0.8Nm).

Pin-Out

Pin	Function
Output Cmn	Common to one side of all outputs
I/O 0	Input or Output
I/O 1	Input or Output
I/O 2	Input or Output
I/O 3	Input or Output
I/O 4	Input or Output
I/O 5	Input or Output
I/O 6	Input or Output
I/O 7	Input or Output
Input Cmn	Common to one side of all inputs

Discrete Outputs

Each discrete output is a solid state relay (SSR). When it is "OFF", it has high impedance, and when "ON" it has low impedance (50 Ω maximum, 25 Ω typical). Because the output is isolated, the user must power it externally. The maximum current and voltage for the output is 75 mA (50 mA for Class I, Div 2) and 30 V.

Using Outputs with Resistive Loads

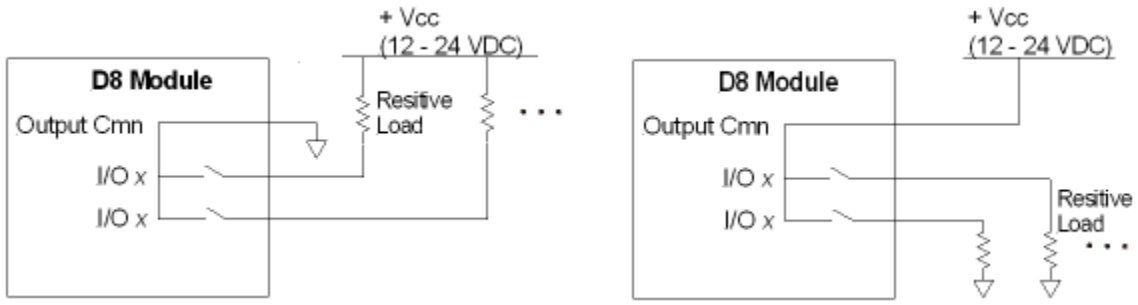


Figure 2: SSR switching resistive load: low-side configuration.

Figure 3: SSR switching resistive load: high-side configuration.

Example: Calculating maximum current for resitive load.

To calculate the maximum current through the SSR in the above diagram, we assume zero SSR resistance:

Max. current = 24V / 480Ω = 50mA

Max. current = 12V / 480Ω = 25mA

In the 24V case, the maximum current is right at the maximum allowed by the SSRs. The outputs may be overpowered if the resistance is reduced further. To calculate the typical current through the SSR, we use the typical SSR resistance of 25W:

Typical current = 24V / (480Ω + 25Ω) = 47.5mA

Typical current = 12V / (480Ω + 25Ω) = 23.8mA

Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an ”ON” to an ”OFF” state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR. See figures below for details.

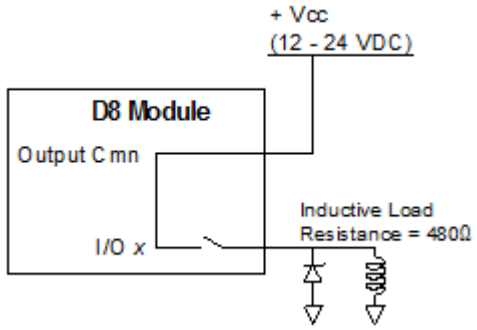


Figure 1: SSR switching inductive inductive load: high-side configuration.

Example: Calculating maximum current for inductive load.

To calculate the maximum current through the SSR in the above diagram, we assume zero SSR resistance:

Max. current = 24V / 480Ω = 50mA

Max. current = 12V / 480Ω = 25mA

In the 24V case, the maximum current is right at the maximum allowed by the SSRs. The outputs may be overpowered if the coil resistance is reduced further. To calculate the typical current through the SSR, we use the typical SSR resistance of 25W:

Typical current = 24V / (480Ω + 25Ω) = 47.5mA

Typical current = 12V / (480Ω + 25Ω) = 23.8mA

Discrete Inputs

The discrete inputs are compatible with signal levels ranging from 12V to 24V. Each input draws a maximum of 2.6mA with a 12V or 24V input. Most PLC outputs can be connected to the DI/O inputs directly. The exception is open collector (sinking) outputs. See the discussion below for using sinking outputs.

Wiring Diagrams

The following are some input wiring diagrams:

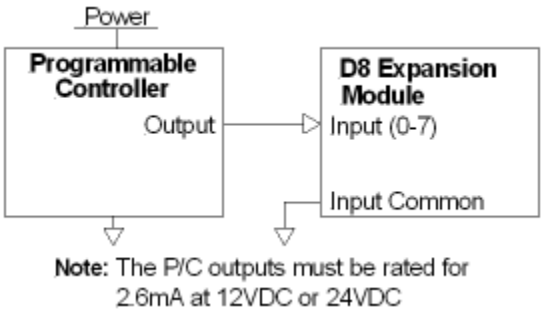


Figure 4: Direct connection to Programmable Controller.

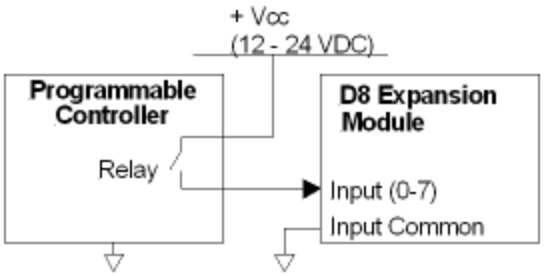


Figure 5: Relay Connection from Programmable Controller.

Using D8 Inputs with Open Collector Outputs

The D8's discrete inputs can be used with open collector (sinking) outputs. There are three configurations that can be used to drive the D8 inputs from an open-collector output:

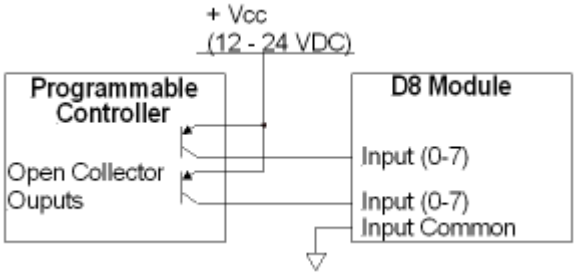


Figure 6: PNP Configuration: This configuration is the most popular for open collector PNP outputs.

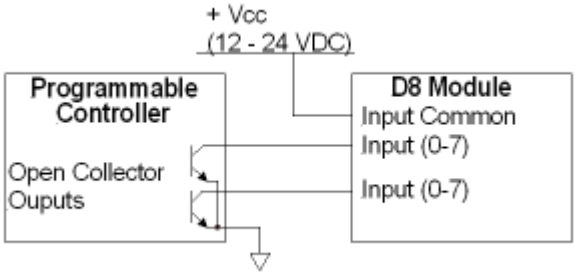


Figure 7: Open Collector Outputs to D8 Inputs with Input Common Connected to Vcc.

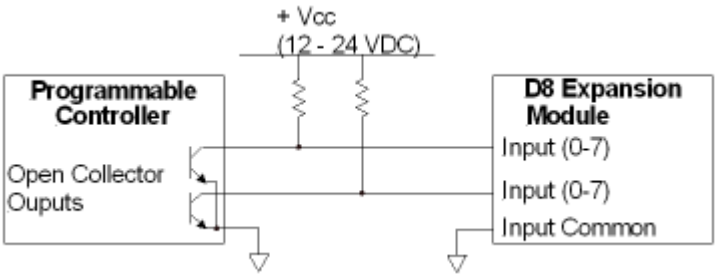


Figure 8: Open Collector Outputs to D8 Inputs with Input Common Connected to ground.

For 24VDC power, the pull-up resistor should be a 4.7kΩ, ¼ watt resister. The output must be capable of switching 2.6mA when closed. When the open collector output is open, the DI/O input sees 11.8V @ 2.6mA.

For 12VDC power, the pull-up resistor should be a 1 kΩ, ¼ watt resister. The output must be capable of switching 2.6mA when closed. When the open collector output is open, the DI/O input sees 9.4V @ 2.6mA.

See Also

D8 module (RMC75)

10.2.11. Q1 Wiring

The Q1 expansion module can be wired to quadrature encoders.

Use shielded twisted pairs for all connections to inputs and outputs. Route the encoder wiring separate from other wiring. You must provide the power supplies needed by your encoders. See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 7 b-in (0.8Nm).

NOTE:
The example schematics do not include transducer pin numbers, color codes, or power supply requirements, since these vary between different transducers. To determine your power supply needs and connector pin-outs or cable color codes, consult your transducer manufacturers documentation.

Pin-Out

Pin	Function
Reg In+	High-Speed Registration or Home Input (12 - 24 VDC)
Reg In-	
A+	Encoder A Input (5 V differential)
Jumper for Termination	
A-	
Quad Cmn	Encoder Common
B+	Encoder B Input (5 V differential)
Jumper for Termination	
B-	
Case	Connected to RMC Chassis

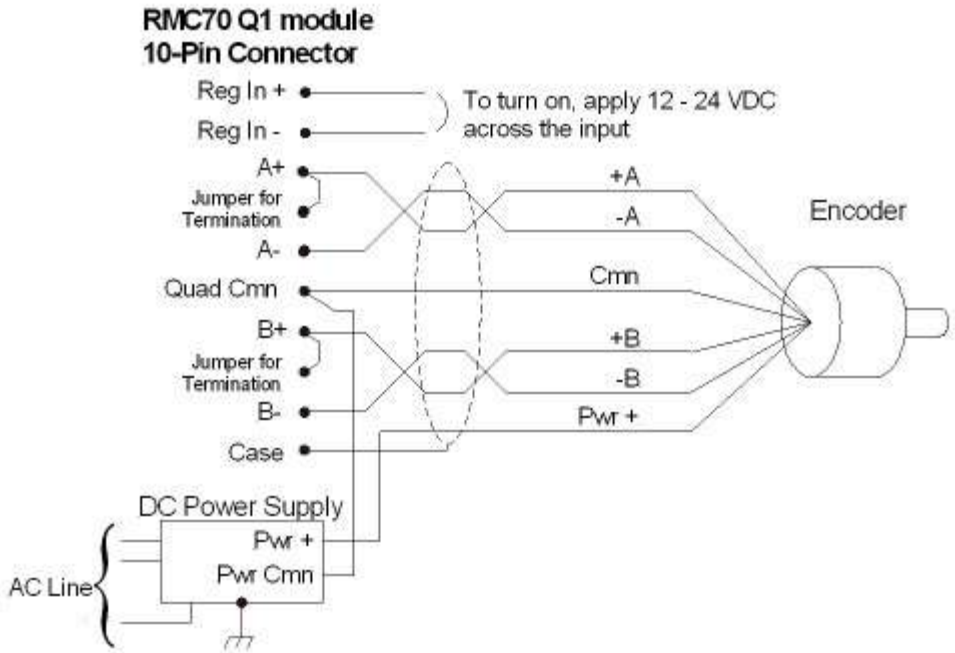
Encoder Wiring

The Q1 module interfaces only to quadrature encoders with 5 V Differential Line Driver (RS-422) signals. Single-ended line drivers (TTL) are *not* supported due to low noise immunity.

Use twisted pairs and shielded cables for the A and B signals. Always install the termination jumper to maintain signal integrity. The termination jumper adds a 249Ω resistor between the + and - inputs.

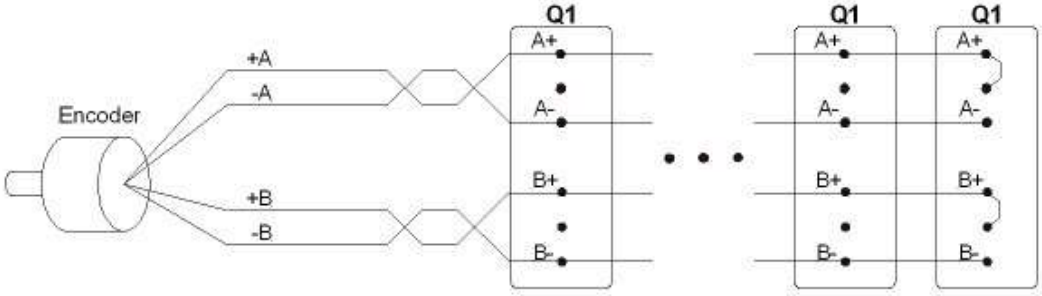
The **Reg** input is compatible with signal levels from 12 to 24 V. It draws 2.7 mA max and turns on when the voltage across the input is greater than 6 V. The polarity is unimportant.

Diagram



Daisy-Chaining Quadrature Inputs

One quadrature encoder can typically output its A and B signals to thirty-two (32) RMC75 Q1 modules. Use a daisy-chain topology as shown below. Do not use a star topology. Add termination only to the last input, as illustrated. Incorrect termination will result in distorted signals and will cause incorrect transducer readings.



See Also

Q1 module (RMC75)

10.3. RMC150

10.3.1. RMC150 Mounting Instructions

Mounting Options:

- Symmetrical DIN 3
- Panel-mount

Orientation:

The RMC should be mounted upright on a vertical surface, such that the air holes are on top and bottom.

Clearance above and below:

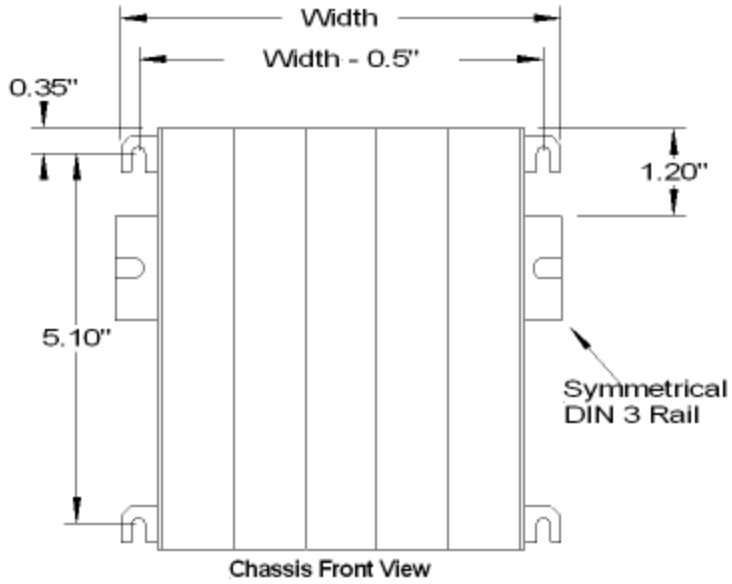
The amount of clearance required depends on the maximum ambient temperature:

Ambient Temperature	Clearance
122 - 140°F (50-60°C)	3 in (7.6 cm)
86 to 122°F (30 to 50°C)	2 in (5.1 cm)
Less than 86°F (30°C)	1 in (2.5 cm)

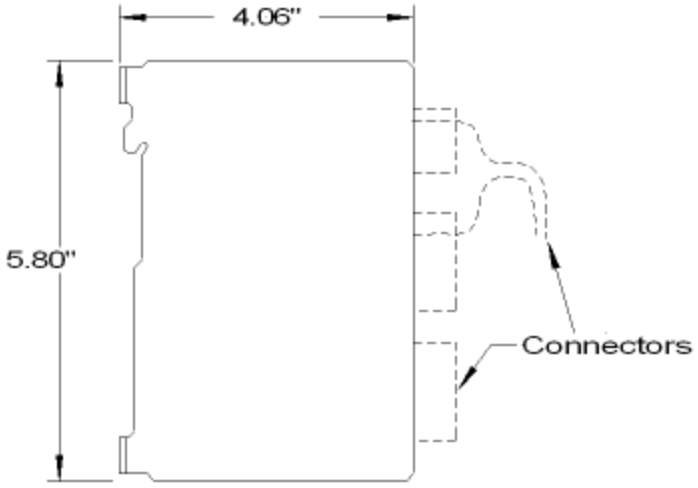
Mounting Dimensions

The width depends on the number of slots in the backplane.

	Width
3 Slots	4.10 in.
4 Slots	5.10 in.
5 Slots	6.10 in.
6 Slots	7.10 in.



Chassis Front View



Chassis Side View

Note:
Allow space for the connectors on the front of the RMC.

See Also
Wiring Guidelines

10.3.2. RMC150E CPU Module Wiring

This topic covers the wiring of the RMC150E CPU Module, including the power and the discrete inputs and outputs. For wiring the discrete inputs and outputs on the DI/O UI/O modules, see the Discrete I/O Wiring and UI/O Wiring topics. See Wiring Guidelines for general wiring information.

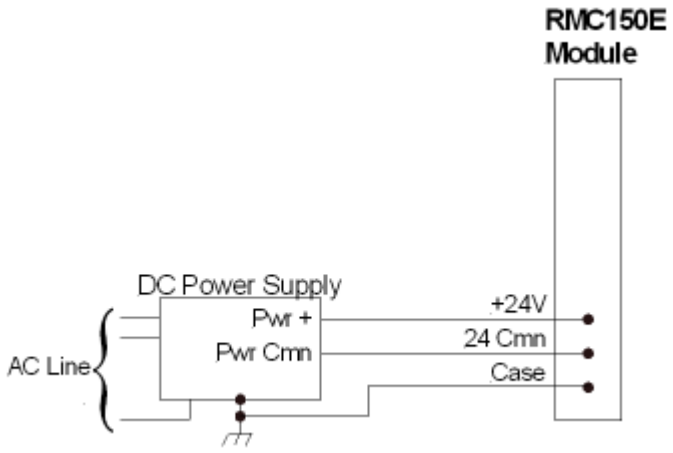
Wire clamp screws must be tightened to max 4.5 lb-in (0.51 Nm).

Power

The RMC150E CPU requires 24VDC (20.4 – 27.6VDC).

The current rating depends on the size of the backplane:

Backplane Slots	Maximum Current
3	375mA
4	500mA
5	625mA
6	750mA



The **Case** pin is electrically connected to the RMC100 case.

Wiring the Communications

USB Monitor Port

Use a standard USB cable to connect to the USB port. USB connector shielding is grounded with case ground.

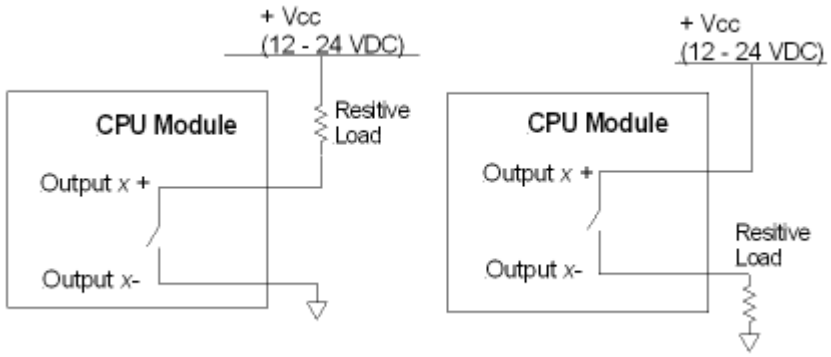
RJ-45 Ethernet connector

Use a standard Category 5, 5e, or 6 cable with an RJ-45 connector to connect to the 10/100 Ethernet port.

Discrete Outputs

The RMC150E CPU module discrete outputs are solid state relays. When they are ”off” they have high impedance, and when ”on” they have low impedance (50Ω maximum, 25Ω typical). The user must power the outputs externally. The maximum current and voltage for the outputs is 75 mA (50 mA for Class I, Div 2) and 30 V.

Each output has a ”+” and ”-” connection. Outputs can be wired in either a high-side or low-side configuration.



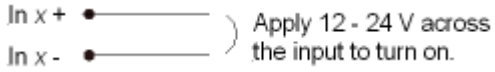
When switching inductive loads, place a diode or tranzorb across the load to protect the switch when transitioning from an "on" to an "off" state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the Solid State Relay.

See the RMC150 DI/O Module Wiring topic for a complete discussion of input wiring.

Discrete Inputs

The RMC150E module inputs are compatible with signal levels ranging from 12 to 24VDC. The discrete inputs draw 3mA maximum.

Note:
The RMC150E CPU inputs are 12-24V, whereas the DI/O module inputs are 5-24V.



For a complete discussion of input wiring, refer to the Discrete I/O Module Wiring topic. Notice, however, that the RMC150E CPU inputs are 12-24VDC and are not compatible with 5VDC signals.

See Also

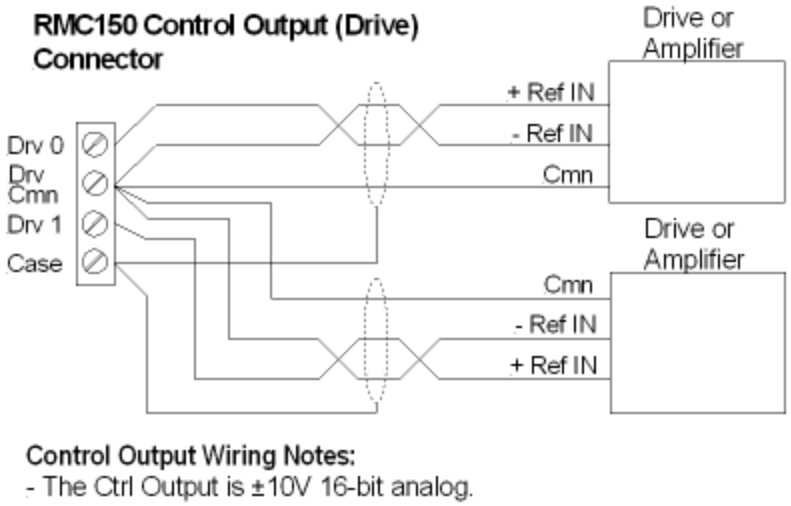
Wiring Guidelines

10.3.3. RMC150 Control Output (Drive) Wiring

The following RMC150 modules have two Control Outputs, one for each axis: Analog (G), Analog (H), MDT (M), Quadrature (Q), SSI (S), Resolver (R). The Control Output is labeled Drive, except on the Quadrature module, where it is pin 12.

Use shielded twisted pairs for all connections to inputs and outputs. See Wiring Guidelines for general wiring information.

On modules with wire clamp screws, the screws must be tightened to max 4.5 lb-in (0.51 Nm).



See Also

Analog (G) Module (RMC150) | Analog (H) Module (RMC150) | MDT (M) Module (RMC150) | Quadrature (Q) Module (RMC150) | SSI (S) Module (RMC150)

10.3.4. RMC150 MDT Wiring

This topic covers the wiring of the Start/Stop and PWM inputs on the RMC150 MDT (M) module. For the Control Output (Drive) wiring, see the RMC150 Control Output (Drive) Wiring topic.

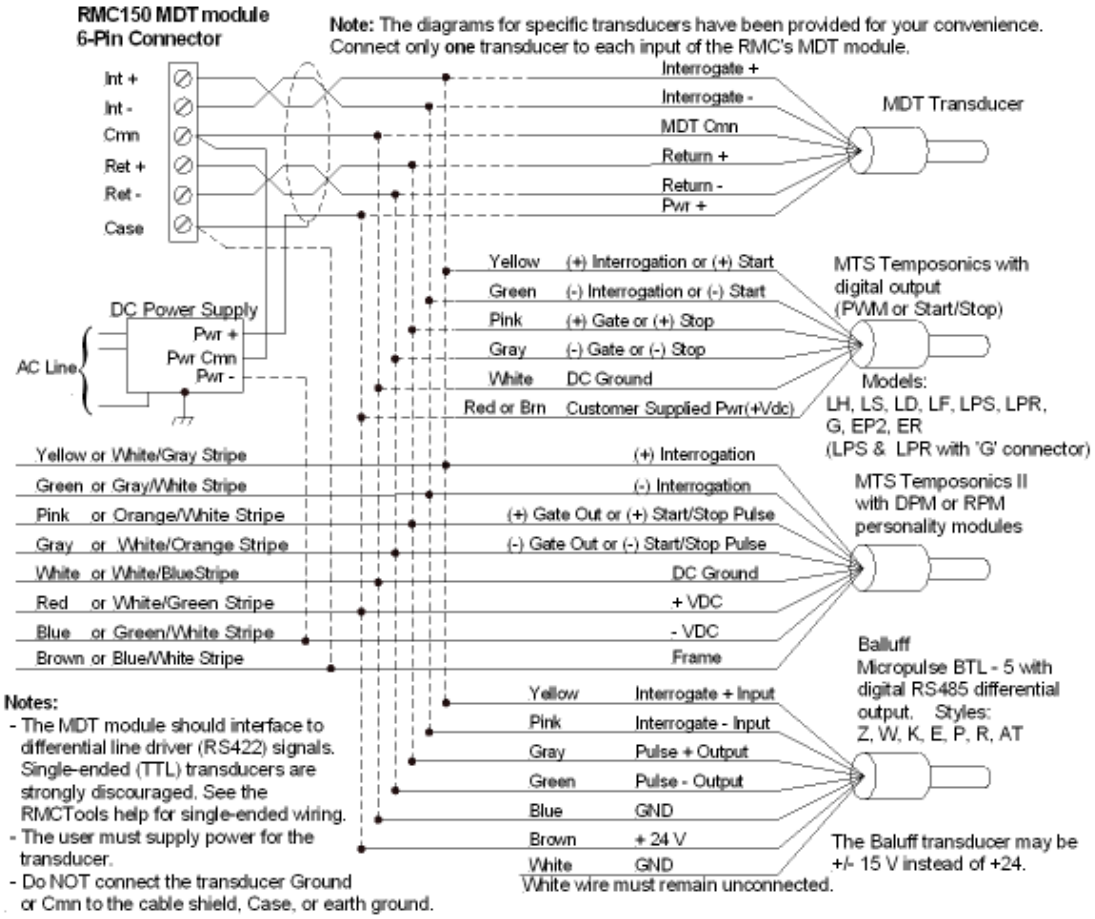
Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers. See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 4.5 lb-in (0.51 Nm).

The commons are internally connected.

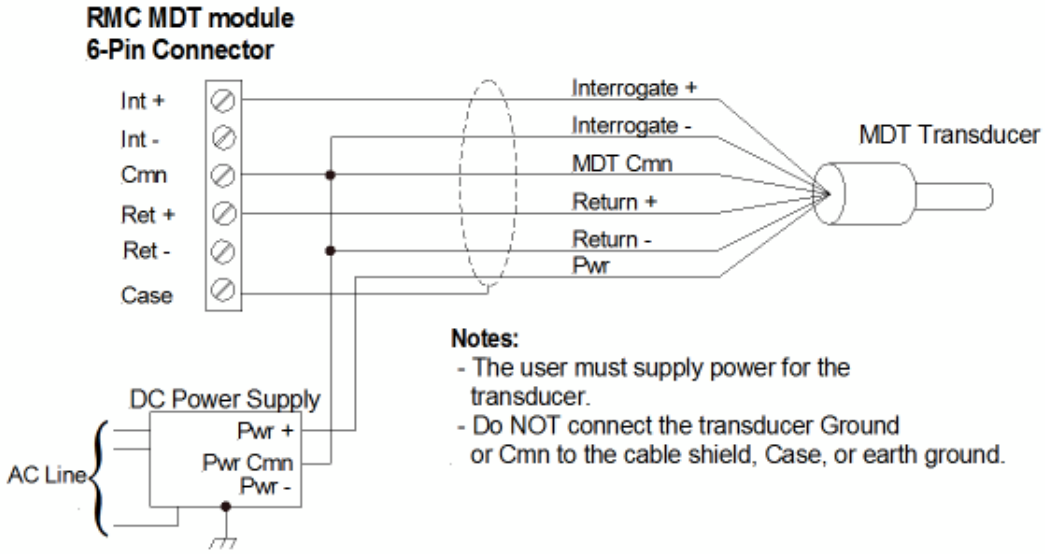
Note:
The MDT input **Cmn** pin must be connected! A disconnected **Cmn** pin can cause noise and inaccurate readings.

Start/Stop and PWM Transducers with Differential Line Driver (RS422) Signals



Start/Stop and PWM Transducers with Single-Ended Signals

This diagram applies to older transducers, such as the Temposonics I and II transducers with neuter outputs. Single-ended transducers are not recommended due to poor noise immunity.



See Also

MDT (M) Module (RMC150) | RMC150 Control Output (Drive) Wiring

10.3.5. RMC150 SSI Wiring

This topic covers the wiring of the SSI inputs on the RMC150 SSI (S) module. For the Control Output (Drive) wiring, see the RMC150 Control Output (Drive) Wiring topic. For wiring SSI inputs on the Universal I/O Module, see the RMC150 UI/O Wiring.

Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers. See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 4.5 lb-in (0.51 Nm).

The commons are internally connected.

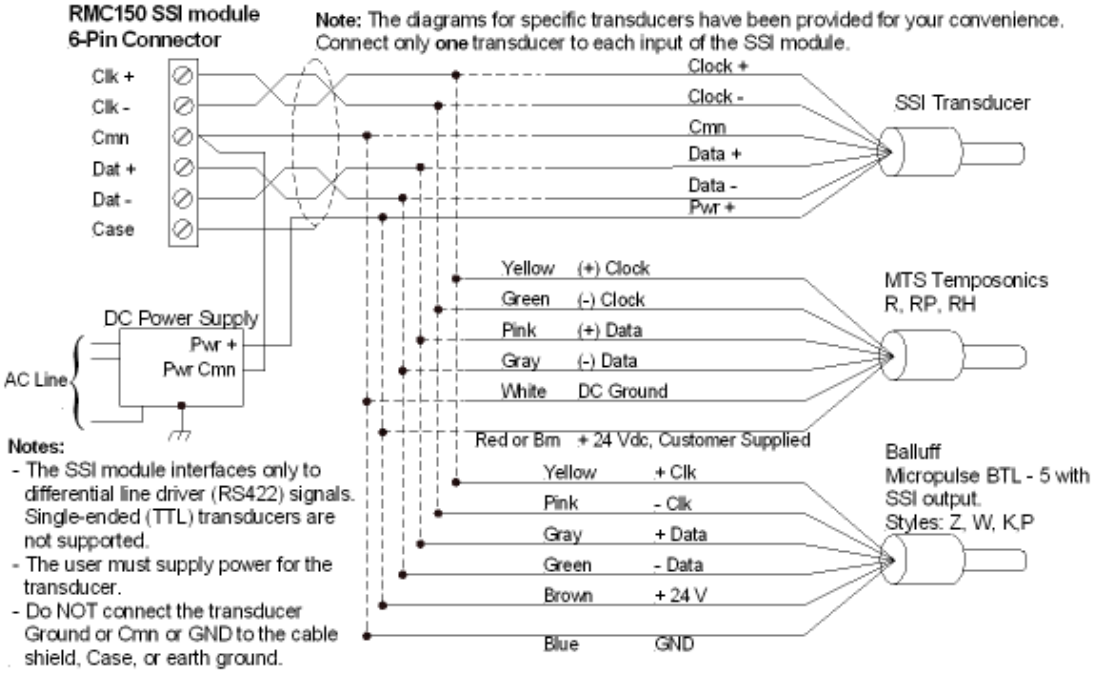
Maximum SSI Cable Length

SSI Clock Rate	Maximum Cable Length*
230 kHz	850 ft (255 m)
921 kHz	120 ft (37 m)

* The cable lengths are approximate, and may be affected by the type of wire and transducer.

Wiring Diagram

> **Note:**
> The SSI input **Cmn** pin must be connected! A disconnected **Cmn** pin can cause noise and inaccurate readings.



See Also

SSI (S) Module (RMC150) | RMC150 Control Output (Drive) Wiring

10.3.6. RMC150 Quadrature Wiring

This topic covers the wiring of the quadrature inputs on the RMC150 Quadrature (Q) module. For the Control Output (Drive) wiring, see the RMC150 Control Output (Drive) Wiring topic. For wiring quadrature inputs on the Universal I/O Module, see the RMC150 UI/O Wiring.

Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers. See Wiring Guidelines for general wiring information.

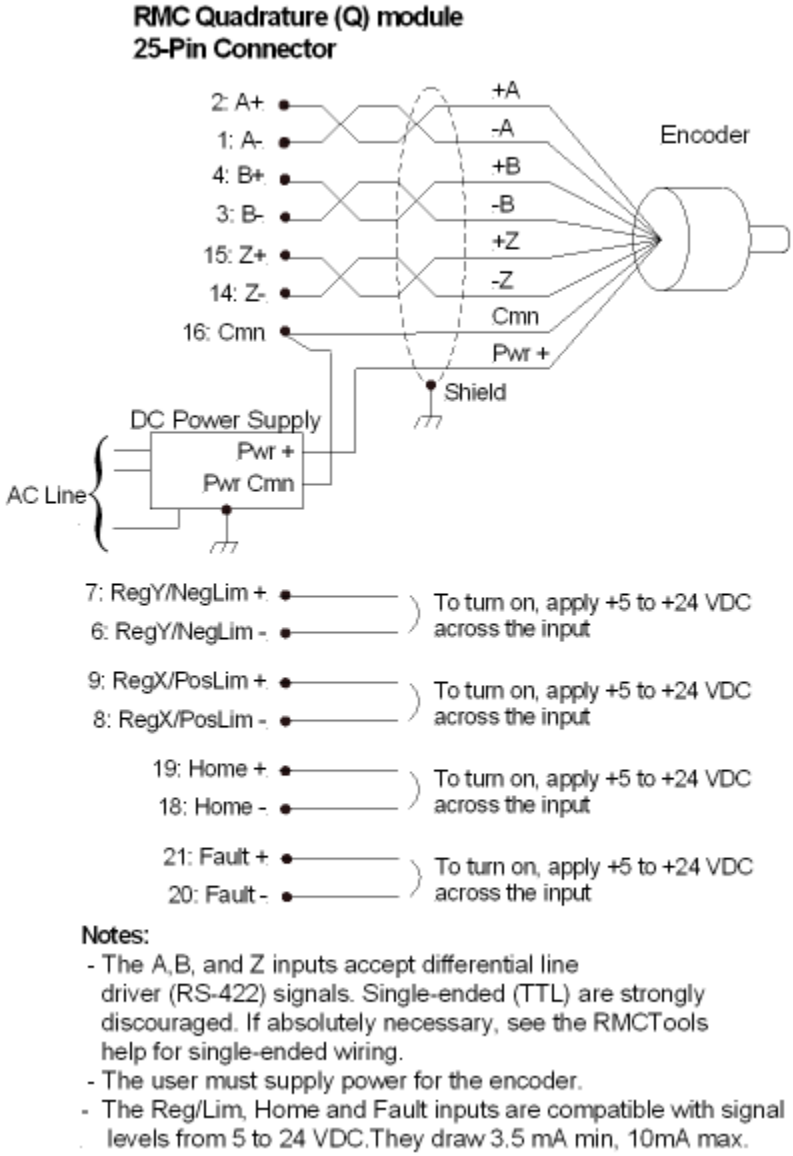
Pin-Out

The RMC-CB-QUAD-01-xx cable is an optional accessory for the QA module.

Pin #	Function	RMC-CB-QUAD-01-xx Wire Color
1	A- from encoder (5 V)	Enc: white/blue
2	A+ from encoder (5 V)	Enc: blue/white
3	B- from encoder (5 V)	Enc: white/orange
4	B+ from encoder (5 V)	Enc: orange/white
5	No connection	
6	Registration Y / Neg Limit - (5-24 VDC)	Lim: white/orange
7	Registration Y / Neg Limit + (5-24 VDC)	Lim: orange/white
8	Registration X / Pos Limit - (5-24 VDC)	Lim: white/blue
9	Registration X / Pos Limit + (5-24 VDC)	Lim: blue/white
10	No connection	
11	No connection	
12	Control Output	Drv: blue/white
13	Control Output Common	Drv: white/blue
14	Z- Index from encoder (5 V)	Enc: white/green
15	Z+ Index from encoder (5 V)	Enc: green/white
16	Encoder Common	Enc: white/brown Enc: brown/white
17	No connection	
18	Home Input - (5-24 VDC)	Lim: white/green
19	Home Input + (5-24 VDC)	Lim: green/white
20	Fault Input - (5-24 VDC)	Drv: white/green
21	Fault Input + (5-24 VDC)	Drv: green/white
22	No connection	
23	No connection	
24	Enable Output	Drv: white/orange
25		Drv: orange/white

Note:
The example schematics do not include transducer pin numbers, color codes, or power supply requirements, since these vary between different transducers. To determine your power supply needs and connector pin-outs or cable color codes, consult your transducer manufacturer's documentation.

Note:
The Quadrature input **Cmn** pin must be connected! A disconnected **Cmn** pin can cause noise and inaccurate readings.



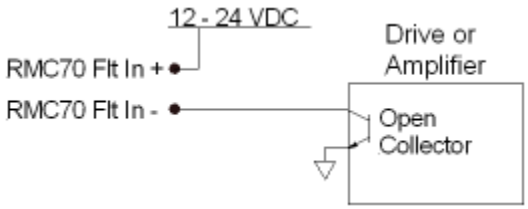
Fault Input Wiring

The Fault input is compatible with signal levels ranging from 12V to 24V. The Fault Input draws 2.7mA maximum and turns on at 6V. The Fault input turns on when a current flows. The polarity of the voltage is unimportant. See the Fault Input topic for more details.

Fault input wiring diagrams:

Generic:

From Open Collector Output:



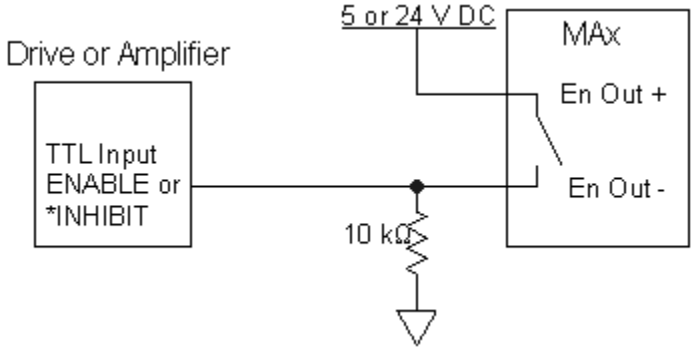
Enable Output Wiring

The Enable output is a solid state relay (SSR). When it is off, it has high impedance, and when on it has low impedance (10 Ω maximum). Because the Enable output is isolated, the user must power it externally. The maximum current and voltage for the Enable output is 100 mA and 30 V.

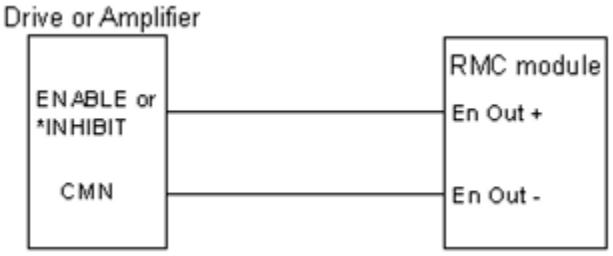
The Enable output has a + and - connection. Both lines must be connected for the output to function. Because both sides of the output are provided, the switches may be independently connected in a high side or low side configuration (that is, with the load (input) on the source or sinking side of the output). See the wiring diagrams below.

See the Enable Output topic for more details on the Enable Output.

To TTL input (high = enable):



To active low Enable input:



Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an "ON" to an "OFF" state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR.

Control Output Wiring

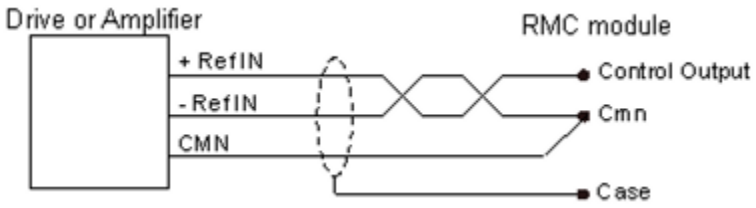
The valve or motor drive connects to the following MA*x* pins:

Pin	Function

Control Output	Control Output
Cmn	Control Output Common. Each axis has two Cmn pins. Either of the 2 pins may be used.

See the Control Output topic for more details on the Control Output.

Note:
The Control Output polarity can be set with the Invert Output Polarity parameter.



See Also

Quadrature (Q) Module (RMC150) | RMC150 Control Output (Drive) Wiring

10.3.7. RMC150 Analog Input Wiring

This topic covers the wiring of the analog inputs on the RMC150 Analog (A), Analog (G), Analog (H), and Universal I/O modules. For the Control Output (Drive) wiring, see the RMC150 Control Output (Drive) Wiring topic.

Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers. See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 4.5 lb-in (0.51 Nm).

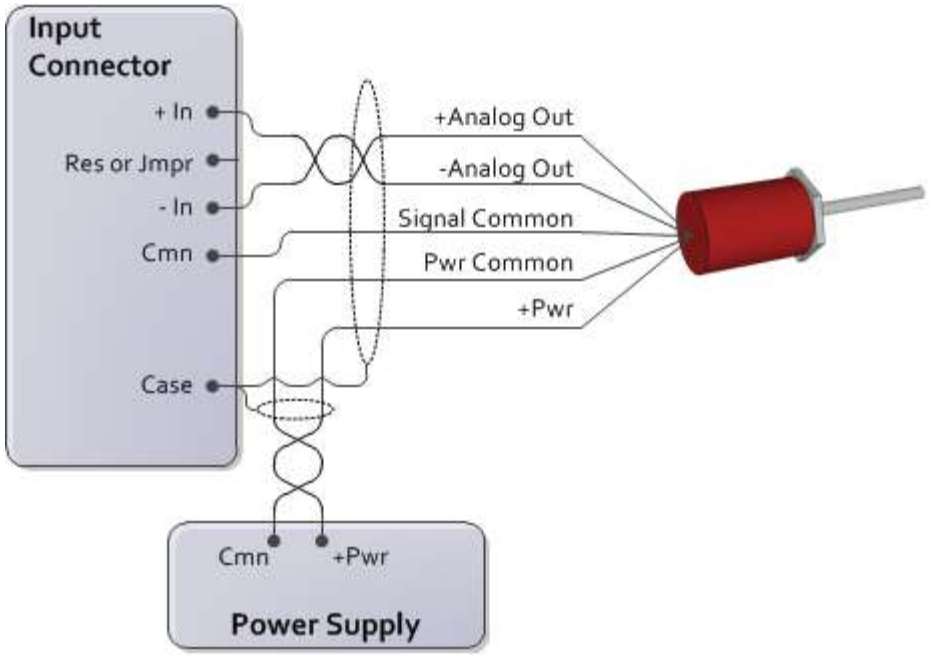
The commons are internally connected.

Note:
The example schematics do not include transducer pin numbers, color codes, or power supply requirements, since these vary between different transducers. To determine your power supply needs and connector pin-outs or cable color codes, consult your transducer manufacturer's documentation.

Note:
The analog input **Cmn** pin must be connected! A disconnected **Cmn** pin can cause noise and inaccurate readings.

Note:
If the input is disconnected, input voltage will be pulled down ≤ -10V.

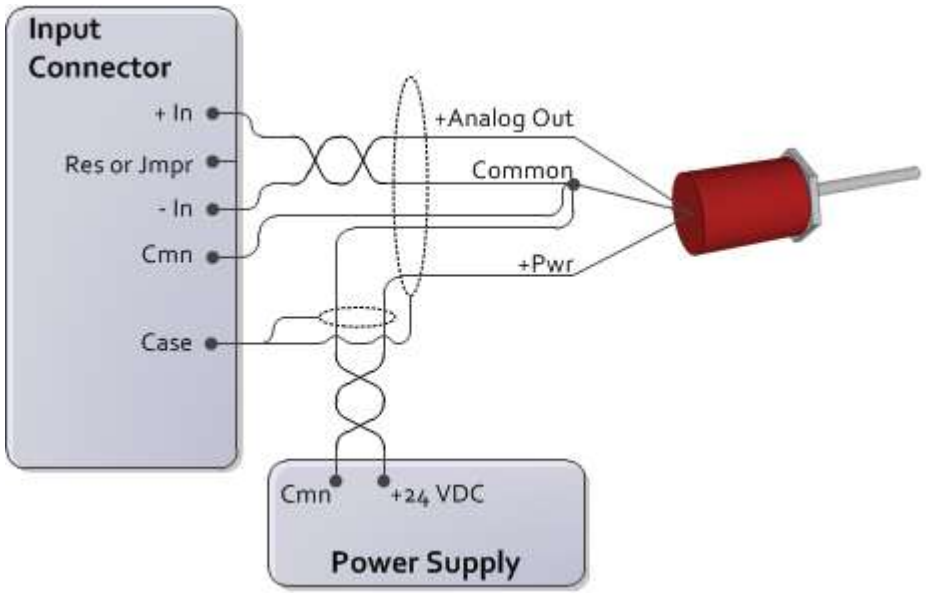
Voltage Input, 4- or 5-wire



To minimize electrical interference:

- **-In** and **Cmn** must be connected. This connected should be made as close to the transducer as possible.
- Use individually shielded twisted-pair wire.
- Typically, the cable shield should be connected to earth on one end only.
- If the transducer has only one common, connect the power supply and the RMC Cmn to it. For best results, make this connection at the transducer.

Voltage Input, 3-wire



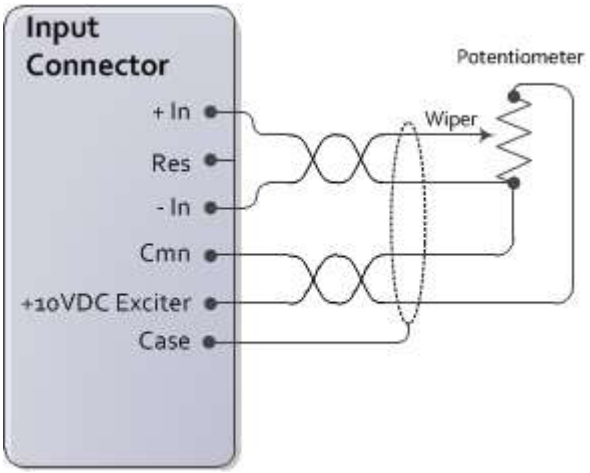
To minimize electrical interference:

- **-In** and **Cmn** must be connected. This connected should be made as close to the transducer as possible.
- Use individually shielded twisted-pair wire.

- Typically, the cable shield should be connected to earth on one end only.

Potentiometer

When using a potentiometer, use the **Exciter** pin, if the module has one, to increase the accuracy of the analog to digital conversion. Do NOT use the Exciter pin to power a transducer! The Exciter pin can source up to 8mA.

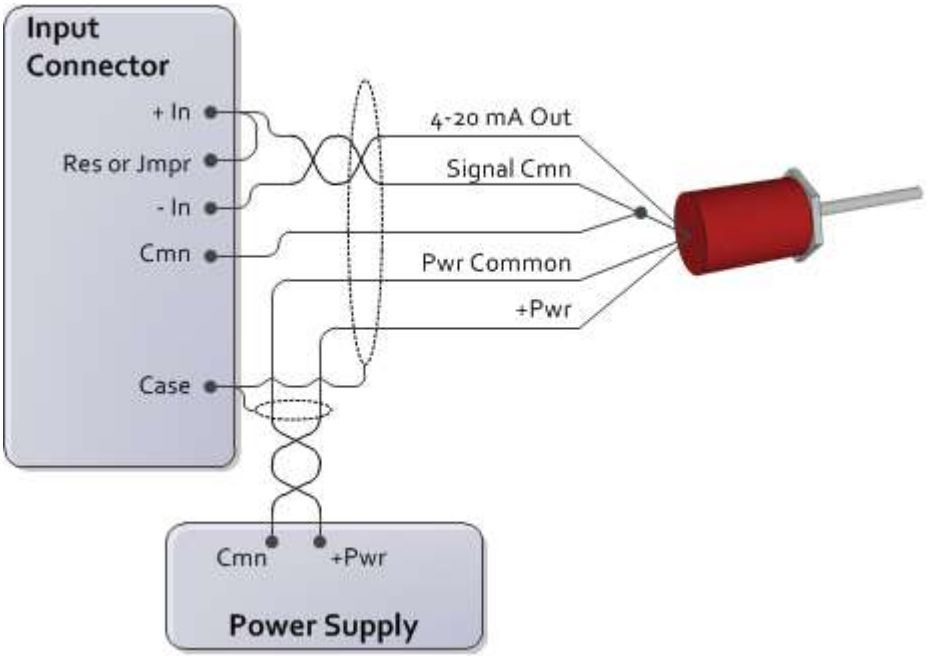


To minimize electrical interference:

- **-In** and **Cmn** must be connected. This connected should be made as close to the potentiometer as possible.
- Use individually shielded twisted-pair wire.
- Typically, the cable shield should be connected to earth on one end only.

Current 4-20mA, 4 wire

Note:
The Analog (G) Module does not support current feedback.

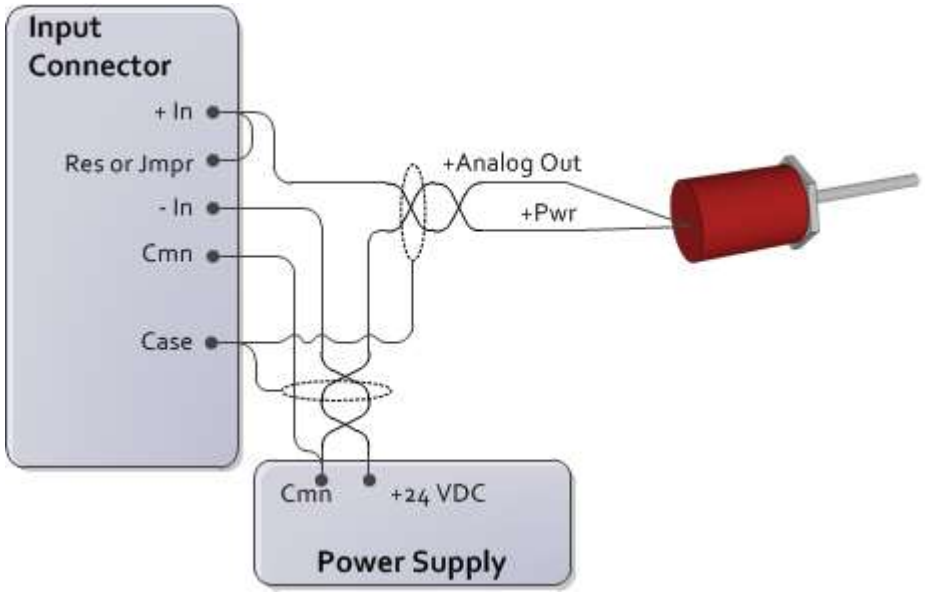


To minimize electrical interference:

- **-In** and **Cmn** must be connected. This connected should be made as close to the transducer as possible.
- Use individually shielded twisted-pair wire.
- Typically, the cable shield should be connected to earth on one end only.

Current 4-20mA, 2-Wire

Note:
The Analog (G) Module does not support current feedback.



Notes:

- The **Res** and **-In** pins are internally connected via a 250 Ohm resistor.
- To minimize electrical interference, use individually shielded twisted-pair wire.

See Also

Wiring Guidelines | Analog (A) Module (RMC150) | Analog (G) Module (RMC150) | Analog (H) Module (RMC150) | Universal I/O (UI/O) Module (RMC150) | RMC150 Control Output (Drive) Wiring

10.3.8. RMC150 Resolver Wiring

This topic covers the wiring of the Resolver inputs on the RMC150 Resolver (R) and Resolver (RW) modules. For the Control Output (Drive) wiring, see the RMC150 Control Output (Drive) Wiring topic.

Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 4.5 lb-in (0.51 Nm).

Pin-Out

Resolver (R) Input 0

Pin	Function

Resolver (RW) Input 0

Pin	Function

R1 0	Reference Output +
Ref In 0	Reference In (normally not used)
R3 0	Reference Output -
S1 0	Sine Input +
S3 0	Sine Input -
S2 0	Cosine Input +
S4 0	Cosine Input -
Case	Controller Chassis ground (shield)

Resolver (R) Input 1

Pin	Function
R1 1	Reference Output +
Ref In 1	Reference In (normally not used)
R3 1	Reference Output -
S1 1	Sine Input +
S3 1	Sine Input -
S2 1	Cosine Input +
S4 1	Cosine Input -
Case	Controller Chassis ground (shield)

Ref In+	Reference Input +
Ref In-	Reference Input -
NC	No connection
S1 0	Sine Input +
S3 0	Sine Input -
S2 0	Cosine Input +
S4 0	Cosine Input -
Case	Controller Chassis ground (shield)

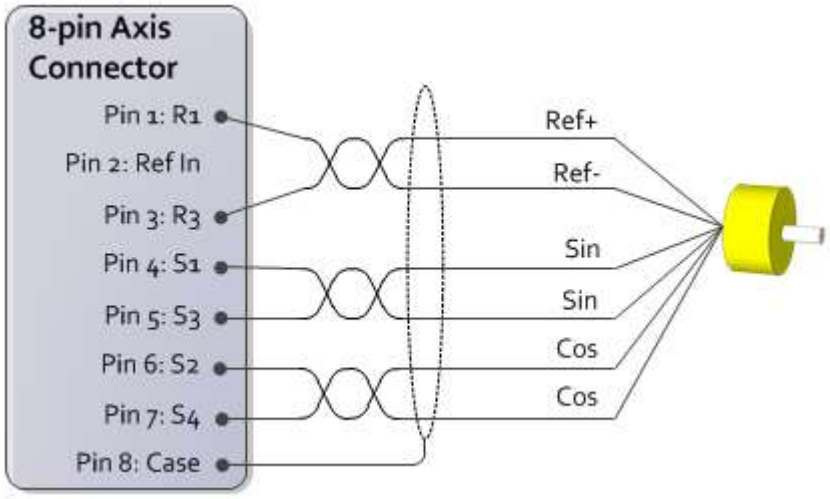
Resolver (RW) Input 1

Pin	Function
Ref In+	Reference Input +
Ref In-	Reference Input -
NC	No Connection
S1 1	Sine Input +
S3 1	Sine Input -
S2 1	Cosine Input +
S4 1	Cosine Input -
Case	Controller Chassis ground (shield)

Typical Wiring Diagram

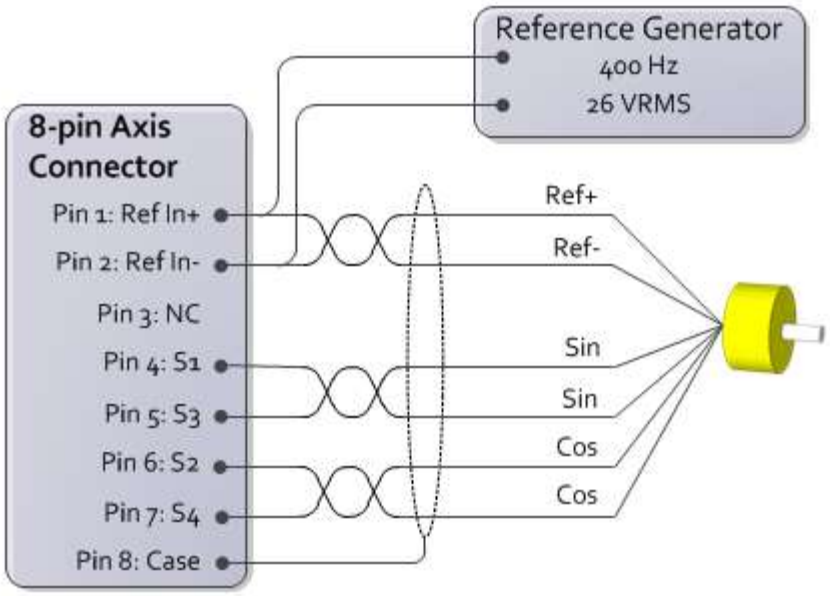
Resolver (R) Module

Below is a typical wiring diagram for resolvers that fall within the Resolver (R) module's reference signal specifications (800Hz to 5kHz and 1.42 to 4.8V RMS). The **Ref In** input is only used for reference signals outside of these specifications, and requires contacting Delta for assistance.



Resolver (RW) Module

Below is a typical wiring diagram for the Resolver (RW) module.



See Also

Resolver (R) Module (RMC150) | RMC150 Control Output (Drive) Wiring | Wiring Guidelines

Show All

10.3.9. RMC150 Discrete I/O Wiring

This topic covers the wiring of the discrete inputs and outputs on the RMC150 DI/O module (**DI/O** or **D**). For wiring the discrete I/O on the RMC150E CPU, see the RMC150E Wiring. For wiring the UI/O module, see the RMC150 UI/O Wiring. See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 2.2 lb-in (0.25 Nm).

DI/O Module Outputs

The DI/O module outputs are solid state relays. When they are ”off” they have high impedance, and when ”on” they have low impedance (50Ω maximum, 25Ω typical). The user must power the outputs externally. The maximum current and voltage for the outputs is 75 mA (50 mA for Class I, Div 2) and 30 V.

Outputs can be wired in either a low-side configuration only.

Using Outputs with Resistive Loads

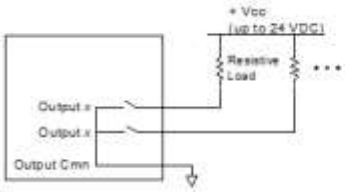


Figure 1: SSR switching resistive load: low-side configuration.

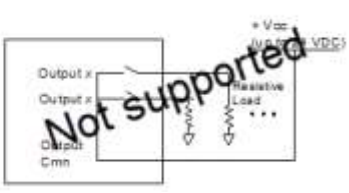


Figure 2: High-side configuration not supported.

The load resistance must be sized such that the maximum current through the SSR does not exceed 50mA. The maximum current is calculated with the following equation:

Current = V_{cc} / R_{Load}

For example, if the supply voltage V_{cc} is 24V, and the load resistance is 480Ω, the current will be:

Current = 24Ω / 480Ω = 50mA

which is right at the 50mA limit. The load resistance should not be decreased any further with a 24V supply voltage.

Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an "ON" to an "OFF" state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR. See figures below for details.

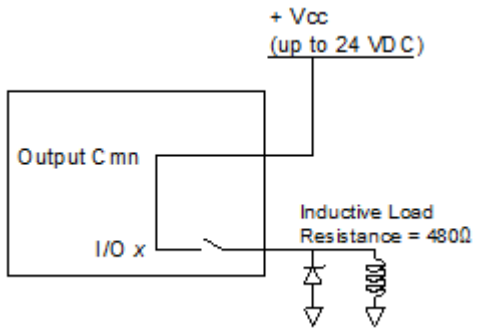


Figure 3: SSR switching inductive inductive load: high-side configuration.

Example: Calculating maximum current for inductive load.

To calculate the maximum current through the SSR in the above diagram, we assume zero SSR resistance:

Max. current = 24V / 480Ω = 50mA

Max. current = 12V / 480Ω = 25mA

In the 24V case, the maximum current is right at the maximum allowed by the SSRs. The outputs may be overpowered if the coil resistance is reduced further. To calculate the typical current through the SSR, we use the typical SSR resistance of 25Ω:

Typical current = 24V / (480Ω + 25Ω) = 47.5mA

Typical current = 12V / (480Ω + 25Ω) = 23.7mA

DI/O Module Inputs

The DI/O module inputs consist of a single common and eighteen individual input connections and are compatible with signal levels ranging from 5V to 24V. Each input draws a maximum of 10mA with a 24V input. Most P/C outputs can be connected to the DI/O inputs directly. The exception is open collector (sinking) outputs. See the discussion below for using open collector outputs.

Note:
The numbered inputs must be positive relative to the **Input Cmn** pin. That is, these are sinking inputs.

The following are some input wiring diagrams:

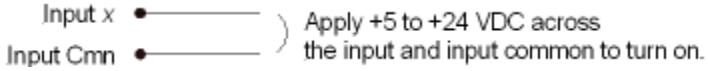


Figure 4: Direct connection to Programmable Controller.

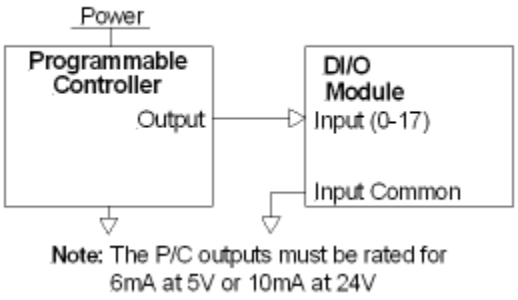


Figure 5: Direct connection to Programmable Controller.

Using DI/O Module Inputs with Open Collector Outputs

The DI/O module inputs can be used with open collector (sinking) outputs. There are two configurations that can be used to drive the discrete inputs from an open-collector output:

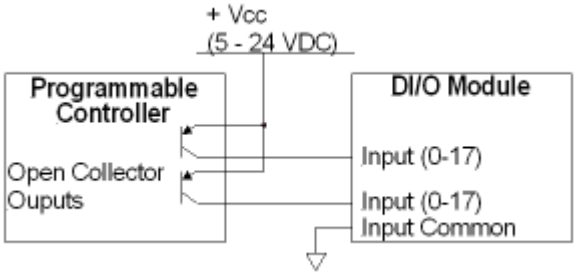


Figure 6: PNP Configuration: This configuration is the most popular for open collector PNP outputs.

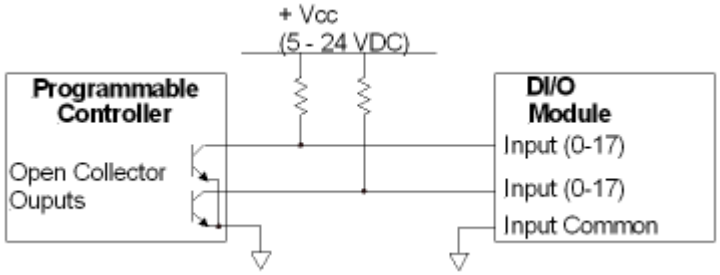


Figure 7: Open Collector Outputs to the DI/O Module Inputs with Input Common Connected to Ground.

For 24VDC power, the pull-up resistor should be a 3.3 kΩ, 1/2 watt resister. The output must be capable of switching 7.5mA when closed. When the open collector output is open, the DI/O input sees 7V @ 5.1mA.

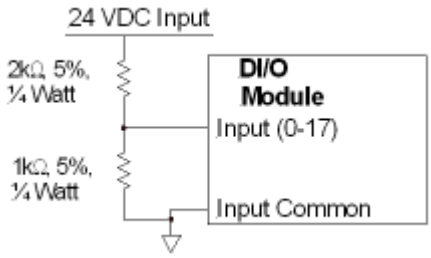
For 12VDC power, the pull-up resistor should be a 560 Ω, 1/8 watt resister. The output must be capable of switching 9.0mA when closed. When the open collector output is open, the DI/O input sees 3.1V @ 3.4mA.

Dividing the Input Voltage

Because the DI/O module inputs are designed for use with 5V outputs, the threshold is 2.75VDC. This is a small percentage of the 24V outputs. As a result, it is important that the inputs have very little noise. This section describes using a voltage divider to raise the apparent input threshold.

Note: Before resorting to dividing the input voltage, take all possible measures to eliminate noise from the inputs.

To divide the inputs, attach resisters to *each* input as shown in the following diagram:



This configuration will reduce noise susceptibility by a factor of about five.

See Also

DI/O Module (RMC150)

Show All

10.3.10. RMC150 UI/O Wiring

This topic covers the wiring of the discrete inputs and outputs on the RMC150 UI/O module (**UI/O** or **U**). See Wiring Guidelines for general wiring information.

Wire clamp screws must be tightened to max 2.2 lb-in (0.25 Nm).

UI/O Analog Inputs

Wire the analog inputs according to the RMC150 Analog Wiring topic.

UI/O Discrete Outputs

The UI/O points configured as outputs are solid state relays. When they are ”off” they have high impedance, and when ”on” they have low impedance (50Ω maximum, 25Ω typical). The user must power the outputs externally. The maximum current and voltage for the outputs is 75 mA (50 mA for Class I, Div 2) and 30 V.

Outputs can be wired in either a high-side or low-side configuration. Because all the outputs share the Output Common, all outputs on the same module must be wired the same.

Using Outputs with Resistive Loads

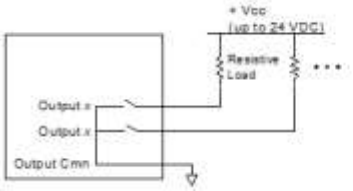


Figure 1: SSR switching resistive load: low-side configuration.

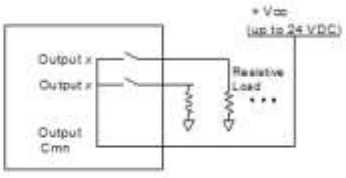


Figure 2: SSR switching resistive load: high-side configuration.

The load resistance must be sized such that the maximum current through the SSR does not exceed 50mA. The maximum current is calculated with the following equation:

$$\text{Current} = V_{cc} / R_{Load}$$

For example, if the supply voltage V_{cc} is 24V, and the load resistance is 480Ω, the current will be:

Current = 24Ω / 480Ω = 50mA

which is right at the 50mA limit. The load resistance should not be decreased any further with a 24V supply voltage.

Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an "ON" to an "OFF" state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR. See figures below for details.

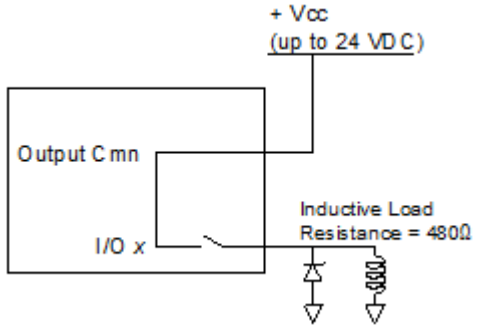


Figure 3: SSR switching inductive inductive load: high-side configuration.

Example: Calculating maximum current for inductive load.

To calculate the maximum current through the SSR in the above diagram, we assume zero SSR resistance:

Max. current = 24V / 480Ω = 50mA

Max. current = 12V / 480Ω = 25mA

In the 24V case, the maximum current is right at the maximum allowed by the SSRs. The outputs may be overpowered if the coil resistance is reduced further. To calculate the typical current through the SSR, we use the typical SSR resistance of 25Ω:

Typical current = 24V / (480Ω + 25Ω) = 47.5mA

Typical current = 12V / (480Ω + 25Ω) = 23.7mA

UI/O Discrete Inputs

The UI/O points configured as inputs use the Input Common and are compatible with signal levels ranging from 12V to 24V. Each input draws a maximum of 2.6mA with a 12V or 24V input. Most PLC outputs can be connected to the UI/O inputs directly.

Because the I/O points share a single common, the inputs must all be connected to the same type of output (sinking or sourcing). If your application has various types of outputs, see figure 9 to convert a sinking output to a sourcing output.

Wiring Diagrams

The following are some input wiring diagrams:

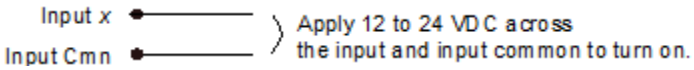


Figure 4: Direct connection to the input.

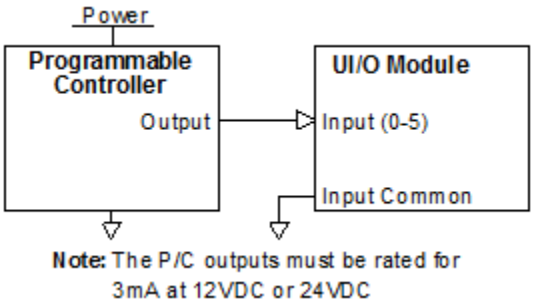


Figure 5: Direct connection to Programmable Controller.

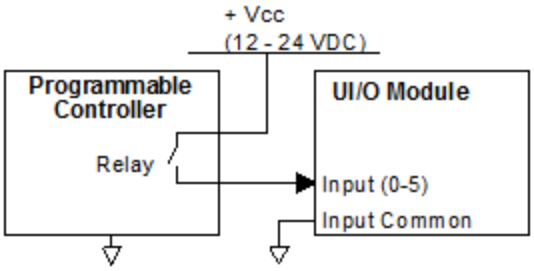


Figure 6: Relay Connection from Programmable Controller.

Using UI/O Inputs with Open Collector Outputs

The UI/O's discrete inputs can be used with open collector (sinking) outputs. There are three configurations that can be used to drive the UI/O inputs from an open-collector output:

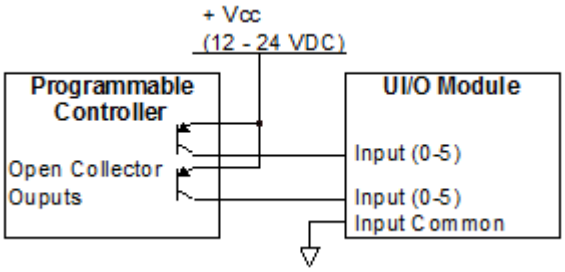


Figure 7: PNP Configuration: This configuration is the most popular for open collector PNP outputs.

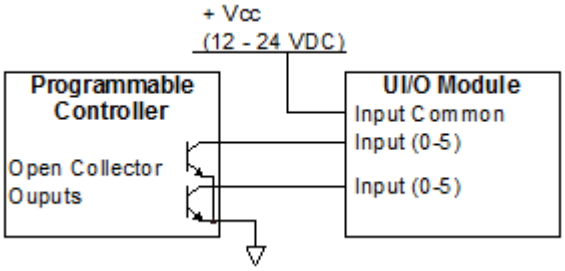


Figure 8: Open Collector Outputs to UI/O Inputs with Input Common Connected to Vcc.

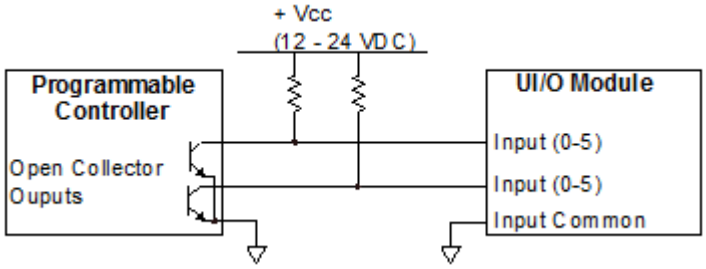


Figure 9: Open Collector Outputs to UI/O inputs with input common connected to ground. Typically not recommended due to the required external resistors and a continuous current draw,

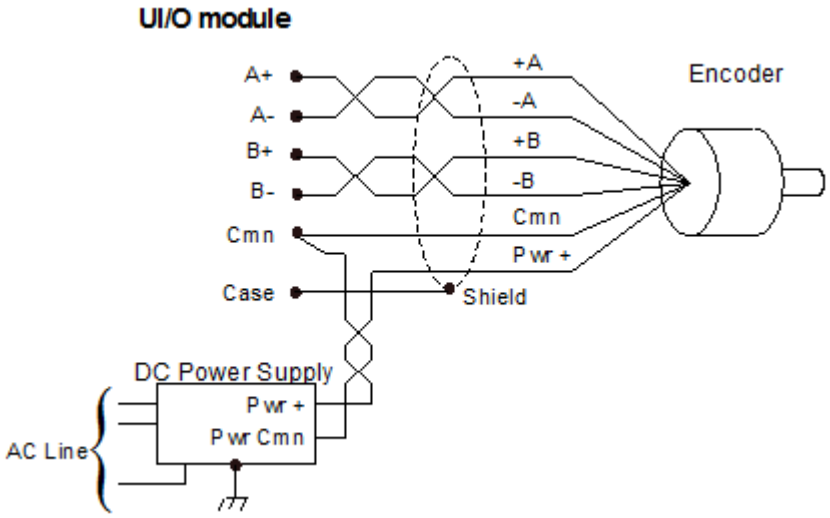
and the fact that the logic becomes inverted. May be necessary for mixing sourcing and sinking outputs.

For 24VDC power, the pull-up resistor should be a 4.7kΩ, ¼ watt resister. The output must be capable of switching 2.6mA when closed. When the open collector output is open, the DI/O input sees 11.8V @ 2.6mA.

For 12VDC power, the pull-up resistor should be a 1 kΩ, ¼ watt resister. The output must be capable of switching 2.6mA when closed. When the open collector output is open, the DI/O input sees 9.4V @ 2.6mA.

UI/O Quadrature Channel

Channels configured as Quadrature inputs should be wired as follows.



The A and B inputs interfaces only to 5V differential line driver (RS-422) signals. Higher voltages will require a signal converter, available from 3rd party suppliers. Single-ended (TTL) transducers are not supported due to low noise immunity. Always use twisted-pairs for the A and B signals. Do NOT connect the transducer Ground or Cmn to the cable shield, Case, or earth ground. The user must supply power for the transducer or encoder.

Termination

Termination is software selectable and should always be used. If quadrature inputs are daisy-chained, apply termination only to the last input.

After setting up the UI/O input as quadrature, and assigning the input to an axis, the **Input Termination** parameter will be available on the **All** tab in the Axis Parameters Pane, in the **Feedback** section. Choose **±A and ± B** to apply termination.

UI/O SSI Channels

The SSI Clock and Data pins interface only to 5V differential line driver (RS-422) signals. Always use twisted-pairs for the Clock and Data signals. Do NOT connect the transducer Ground or Cmn to the cable shield, Case, or earth ground. The user must supply power for transducers or encoders.

Maximum SSI Cable Length

For SSI inputs on the UI/O module, wire delay compensation is available. If your cable length exceeds the length for your clock rate given in the table below, you should use the SSI Wire Delay parameter to account for the time delay of the SSI signal:

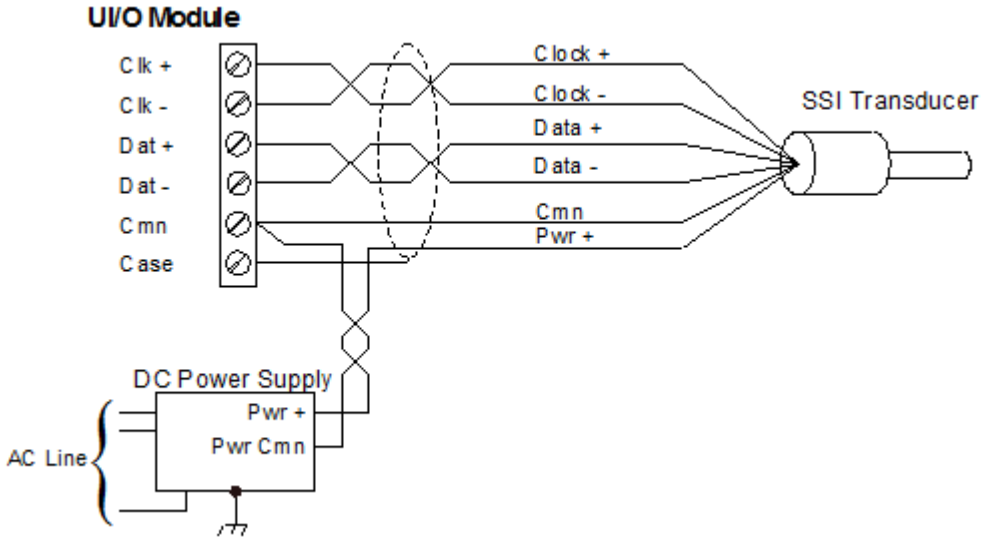
SSI Clock Rate	Maximum Cable Length*
250 kHz	770 ft (235 m)
500 kHz	325 ft (99 m)

971 kHz	110 ft (34 m)

* The cable lengths are approximate, and may be affected by the type of wire and transducer.

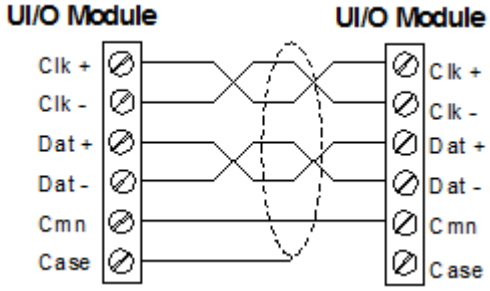
SSI Inputs

Channels configured as SSI inputs and interfacing to transducers or encoders should be wired as follows.



SSI, UI/O to UI/O

SSI channels configured for communicating between RMCs should be wired as follows.



SSI Monitor Mode

SSI Register Input mode can be used with Monitor Mode to monitor the communication between another RMC and an SSI transducer or encoder, thereby synchronizing multiple RMCs to one SSI device.

When wiring a daisy-chained SSI system, the SSI master (the UI/O) should be on one end of the daisy chain with the SSI device on the other end, and any monitoring UI/O modules in the middle of the daisy chain.

Wire the SSI device to the first monitoring RMC as illustrated in the **SSI Inputs** section above, then daisy chain the first RMC to the other RMCs as illustrated in the **SSI, UI/O to UI/O** section above. Apply termination only to the SSI master.

Termination

Termination is software selectable and should always be used. If SSI inputs are daisy-chained, apply termination only to the last input.

For channels configured as SSI Axis Input, after assigning the input to an axis, the **Input Termination** parameter will be available on the **All** tab in the Axis Parameters Pane, in the **Feedback** section. Choose **±Data** to apply termination.

For all other SSI channel configurations, the **SSI Termination** option is provided in the UI/O Properties dialog when configuring the Quad/SSI channels.

See Also

UI/O Module (RMC150)

10.4. RMC200

10.4.1. RMC200 Mounting Instructions

The RMC200 provides for panel-mounting.

Orientation:

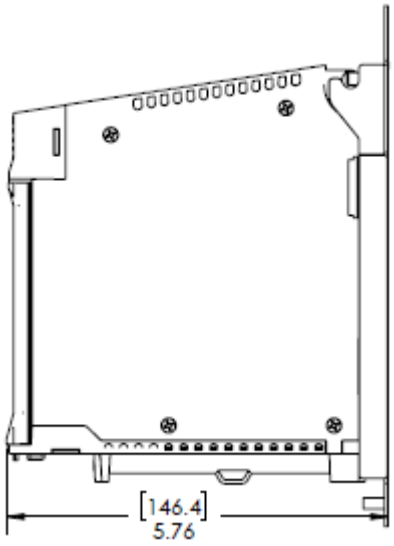
The RMC should be mounted upright on a vertical surface, such that the air holes are on top and bottom.

Clearance above and below:

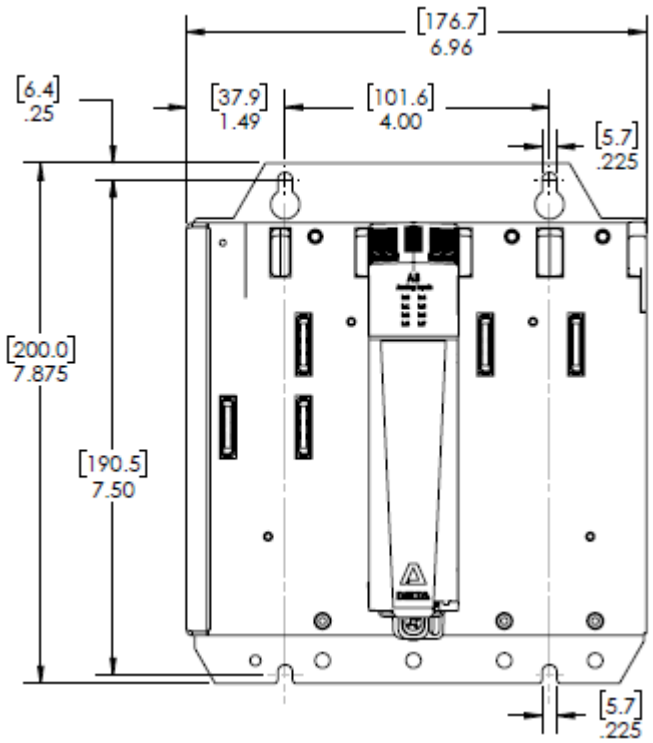
The amount of clearance required depends on the maximum ambient temperature:

Ambient Temperature	Clearance
122 - 140°F (50-60°C)	3 in (7.6 cm)
86 to 122°F (30 to 50°C)	2 in (5.1 cm)
Less than 86°F (30°C)	1 in (2.5 cm)

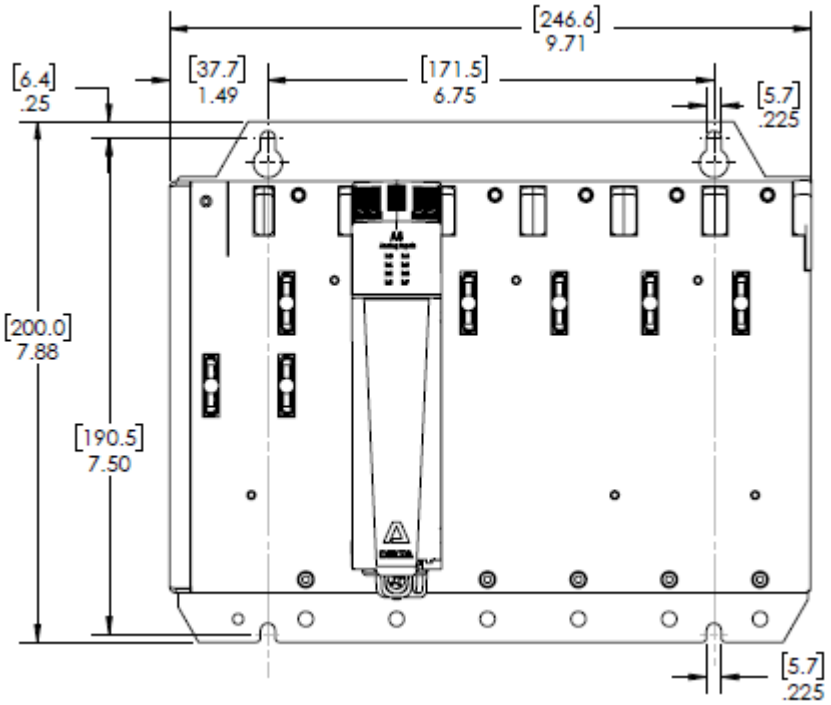
Side Dimensions, All Bases



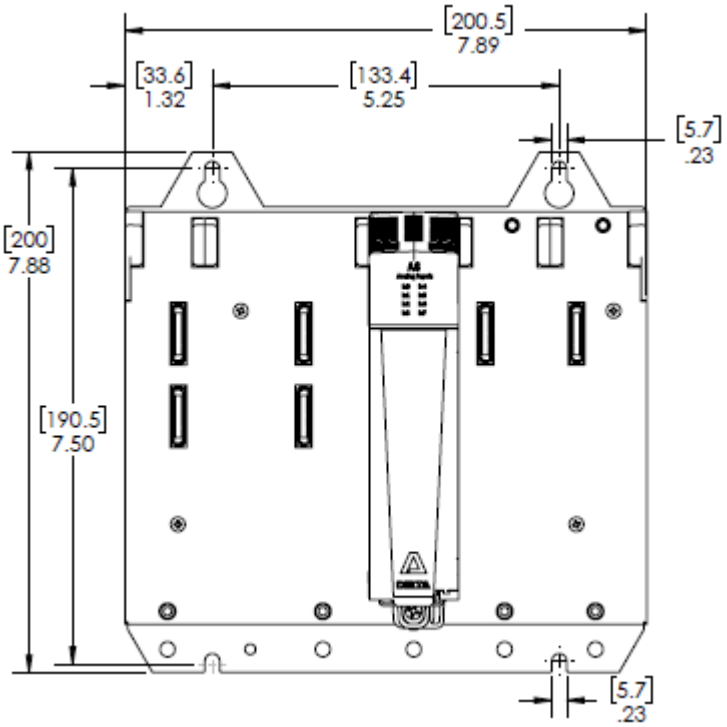
B5L Mounting Dimensions



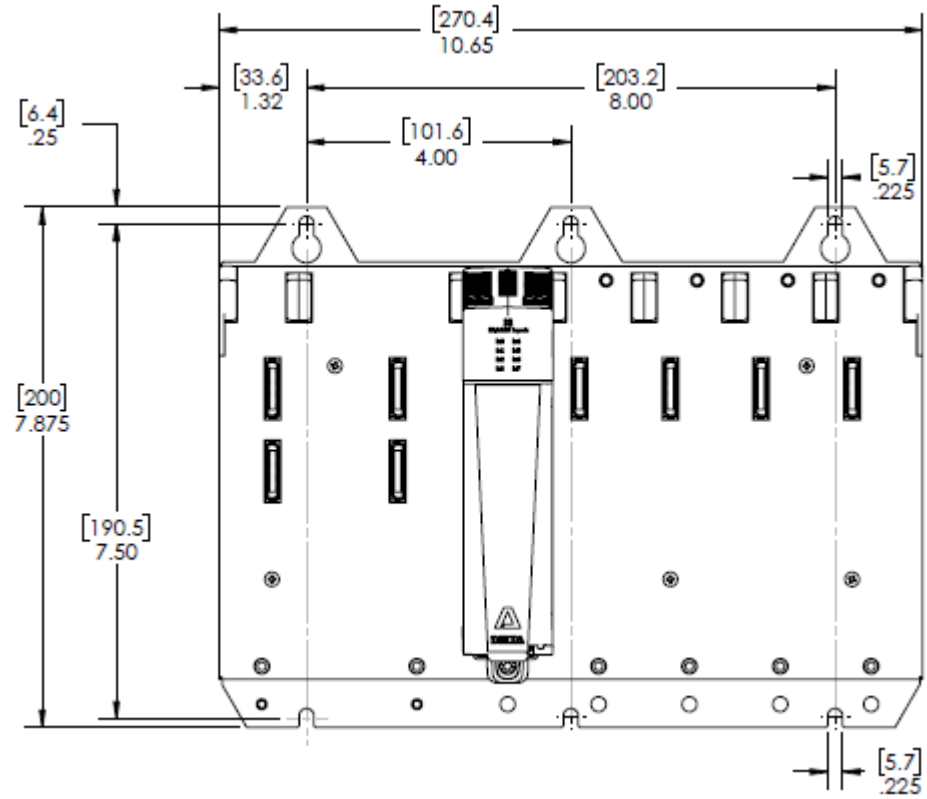
B7L Mounting Dimensions



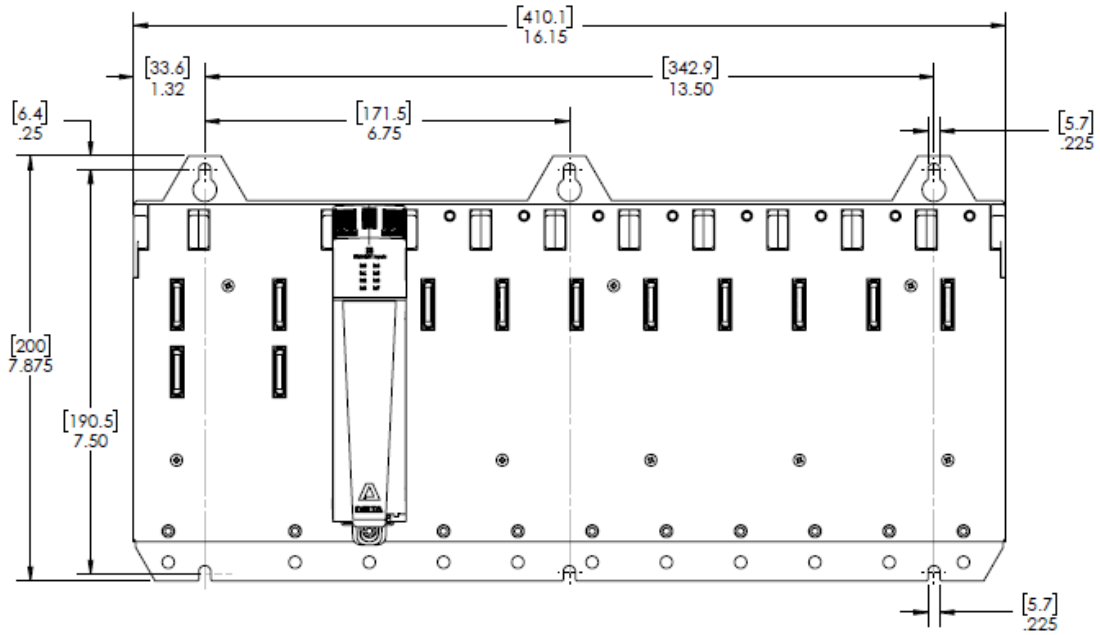
B5 Mounting Dimensions



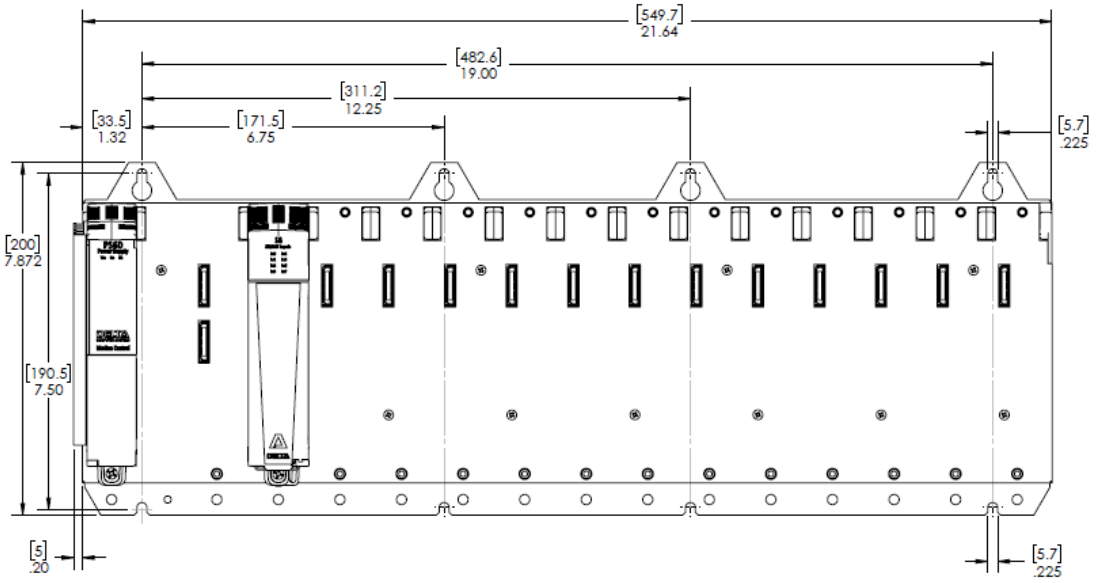
B7 Mounting Dimensions



B11 Mounting Dimensions



B15 Mounting Dimensions



See Also

Wiring Guidelines

10.4.2. CPU20L Wiring

This topic covers the wiring of the CPU20L Module (RMC200). See Wiring Guidelines for general wiring information.

Ethernet

The 2 RJ45 Ethernet ports act as a 3-way switch with the RMC on one port with a single IP address. Accepts an RJ45 Ethernet cable.

USB

Accepts a USB Type B plug. USB connector shielding is grounded with case ground.

Power Pin-out

Pin	Label	Description
1	+24V	24 VDC power
2	24 Cmn	24 VDC return
3	Cased	Shield connection

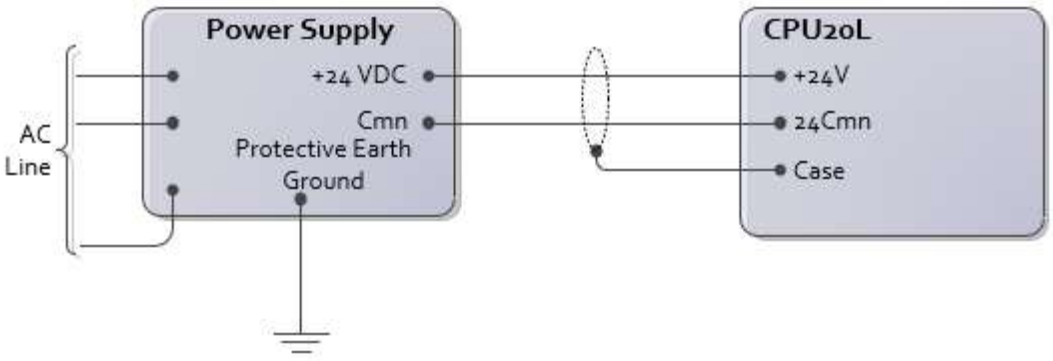
Input Power

Power	
Input Power	28 W max (1.2A at 24Vdc)
Input Voltage	Recommended 24Vdc ± 15% (20.4 – 27.6 V), 30 V max. Overvoltage shutdown at 36 V.

Power Fusing

Fusing is not required for protecting the CPU20L power input. The CPU20L has an internal, non-user-accessible, fast-acting 5 A built-in fuse. If a fuse is installed, it should be a fast-acting fuse less than 5 A, such as 4 A. Fusing may also optionally be installed to protect the wiring.

Power Wiring Diagram



The **Case** pin is electrically connected to the RMC200 case.

Discrete I/O Pin-out

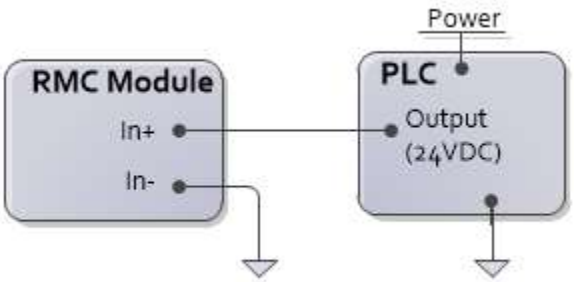
Pin	Label	Description
1	DIn0+	General-purpose input 0, 12 -24 Vdc
2	DIn0-	
3	DIn1+	General-purpose input 1, 12 -24 Vdc
4	DIn1-	
5	DOut0+	General-purpose output 0, Solid State Relay up to 30 Vdc or peak AC
6	DOut0-	
7	DOut1+	General-purpose output 1, Solid State Relay up to 30 Vdc or peak AC

8	DOut1-

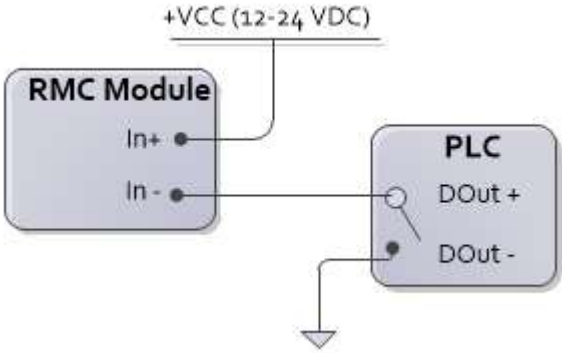
Discrete Inputs

The CPU20L discrete inputs are compatible with signal levels ranging from 12 to 24VDC. The discrete inputs draw 3mA maximum. Each input is individually isolated.

Used with a sourcing output

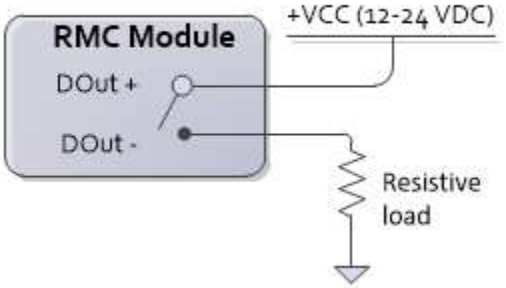


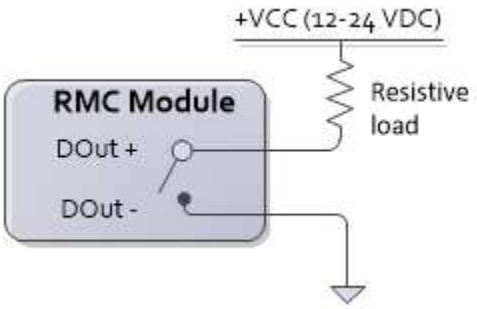
Used with a sinking output



Discrete Outputs

The CPU20L discrete outputs are solid state relays. The off state is high impedance, the on state is low impedance (8Ω max, 5Ω typical). Each output is individually isolated. Outputs can be wired in either a high-side or low-side configuration. Outputs can be wired in either a high-side or low-side configuration. Max current 75 mA. Max voltage 30 V.





See Also

Wiring Guidelines | CPU20L

10.4.3. PS4D and PS6D Wiring

This topic covers the wiring of the PS4D and PS6D power supply modules. See Wiring Guidelines for general wiring information.

Pin-out

Pin	Label	Description
1	+24V	24 VDC power

2	24 Cmn	24 VDC return
3	Cased	Shield connection

Input Power

Power	
Input Power	PS4D: 42 W max (1.8A at 24Vdc) PS6D: 60 W max (2.5A at 24Vdc)
Input Voltage	Recommended 24Vdc ± 15% (20.4 – 27.6 V), 30 V max. Overvoltage shutdown at 36 V.

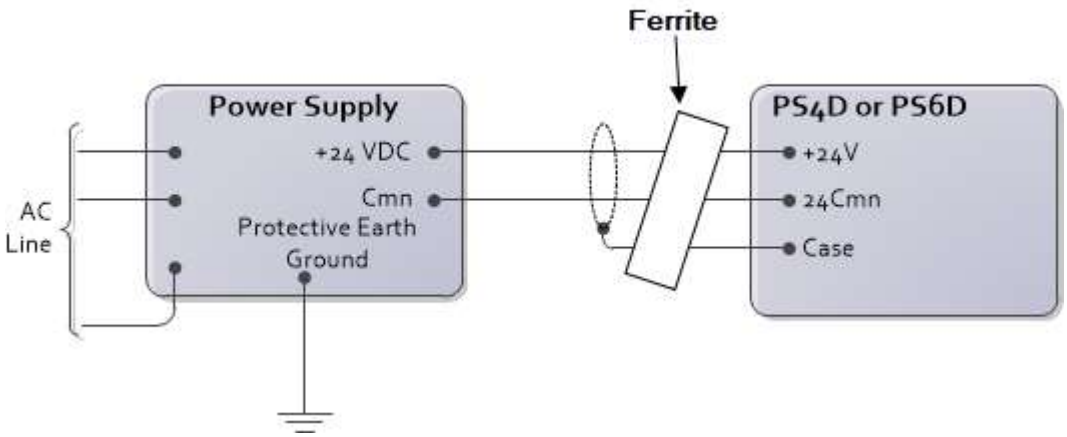
Ferrite for Conducted Emissions Compliance

A ferrite ring is required on the power wires for Conducted Emissions compliance. Recommended ferrite is Fair-Rite 0431167281.

Fusing

Fusing is not required for protecting the PS4D. The PS4D has an internal, non-user-accessible, fast-acting 5 A built-in fuse. If a fuse is installed, it should be a fast-acting fuse less than 5 A, such as 4 A. Fusing may also optionally be installed to protect the wiring.

Diagram



The **Case** pin is electrically connected to the RMC200 case.

See Also

Wiring Guidelines | PS4D | PS6D

10.4.4. CPU40 Wiring

This topic covers the wiring of the RMC200 CPU40 Module. See Wiring Guidelines for general wiring information.

Ethernet

The 2 RJ45 Ethernet ports act as a 3-way switch with the RMC on one port with a single IP address. Accepts an RJ45 Ethernet cable.

USB

Accepts a USB Type B plug. USB connector shielding is grounded with case ground.

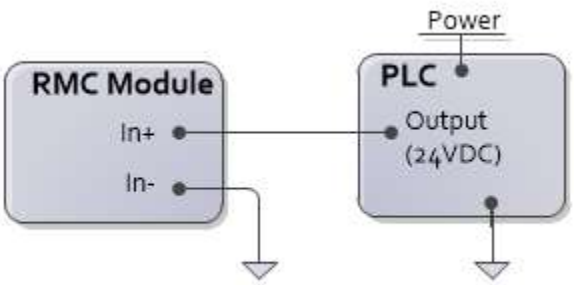
Discrete I/O Pin-out

Pin	Label	Description
1	DIn0+	General-purpose input 0, 12 -24 Vdc
2	DIn0-	
3	DIn1+	General-purpose input 1, 12 -24 Vdc
4	DIn1-	
5	DOut0+	General-purpose output 0, Solid State Relay up to 30 Vdc or peak AC
6	DOut0-	
7	DOut1+	General-purpose output 1, Solid State Relay up to 30 Vdc or peak AC
8	DOut1-	

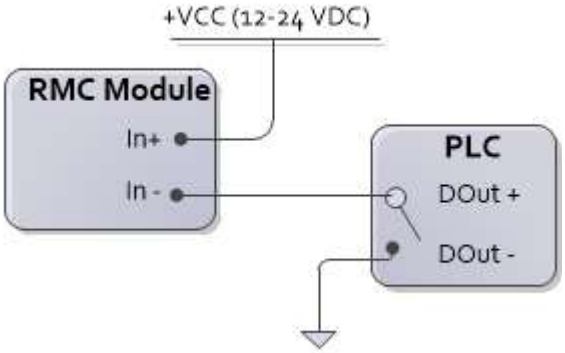
Discrete Inputs

The CPU40 discrete inputs are compatible with signal levels ranging from 12 to 24VDC. The discrete inputs draw 3mA maximum. Each input is individually isolated.

Used with a sourcing output

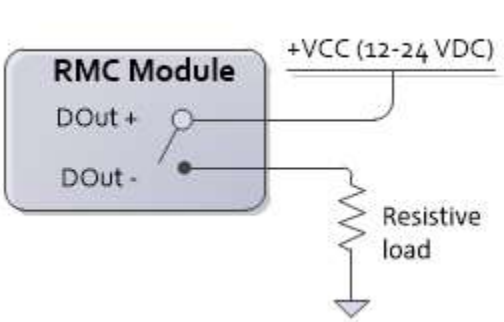


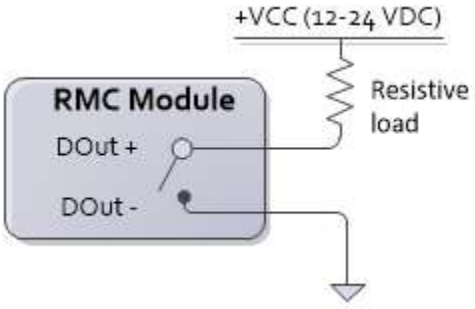
Used with a sinking output



Discrete Outputs

The CPU40 discrete outputs are solid state relays. The off state is high impedance, the on state is low impedance (8Ω max, 5Ω typical). Each output is individually isolated. Outputs can be wired in either a high-side or low-side configuration. Outputs can be wired in either a high-side or low-side configuration. Max current 75 mA. Max voltage 30 V.





See Also

Wiring Guidelines

10.4.5. CA4 Wiring

This topic covers the wiring of the RMC200 CA4 Module. The CA4 has 4 analog outputs, individually software selectable as 0-10V, ±10 V, 4-20 mA, or ±20 mA, with unipolar or bipolar operation, or a custom range within the ±10 V or ±20 mA range.

See Wiring Guidelines for general wiring information.

Pin-Out

Terminal Block 1 (top)

Description		Pin			Description
Voltage or current output 0	CtrlOut0	**1**	**2**	CtrlOut1	Output 1
Output common	Cmn	**3**	**4**	Cmn	
Output common	Cmn	**5**	**6**	Cmn	
Shield connection	Case	**7**	**8**	Case	
Enable Output 0+	EnOut0+	**9**	**10**	EnOut1+	
Enable Output 0-	EnOut0-	**11**	**12**	EnOut1-	
Fault Input 0+	FltIn0+	**13**	**14**	FltIn1+	
Fault Input 0-	FltIn0-	**15**	**16**	FltIn1-	
Shield connection	Case	**17**	**18**	Case	

All the Cmn pins in Terminal Blocks 1 and 2 are internally connected.

Terminal Block 2 (bottom)

Description		Pin			Description
Output 2	CtrlOut2	**1**	**2**	CtrlOut3	Output 3
	Cmn	**3**	**4**	Cmn	
	Cmn	**5**	**6**	Cmn	
	Case	**7**	**8**	Case	
	EnOut2+	**9**	**10**	EnOut3+	
	EnOut2-	**11**	**12**	EnOut3-	
	FltIn2+	**13**	**14**	FltIn3+	
	FltIn2-	**15**	**16**	FltIn3-	
	Case	**17**	**18**	Case	

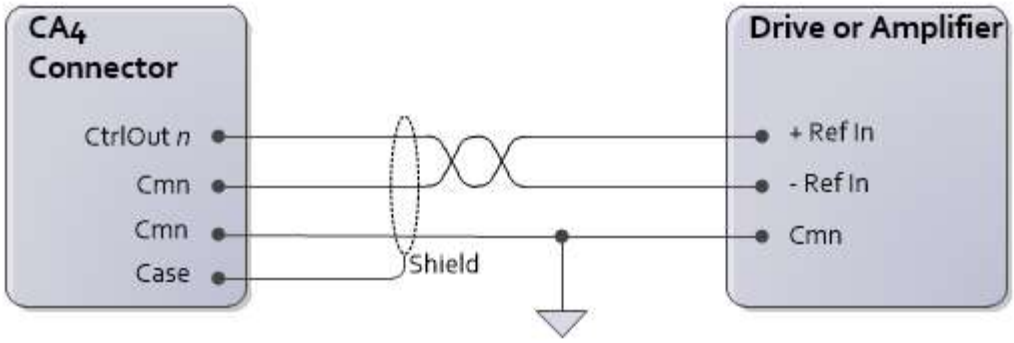
All the Cmn pins in Terminal Blocks 1 and 2 are internally connected.

Wiring to Differential Inputs

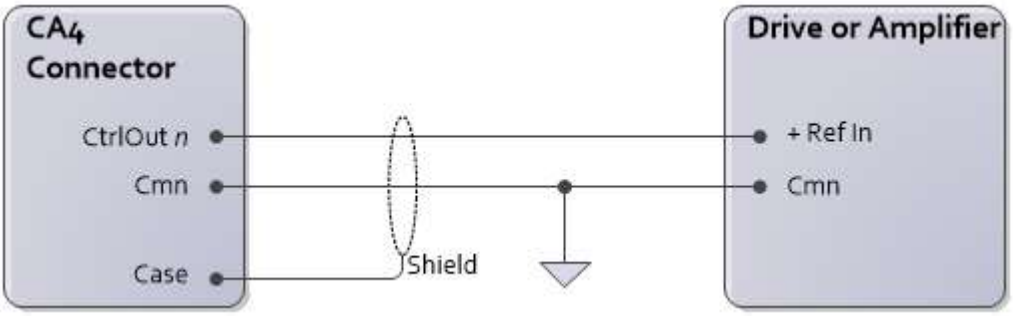
The CA4 analog outputs provide ±10 V, 4-20 mA, or ±20 mA.

Wiring the CA4 Analog Output to Differential Inputs

Differential inputs provide the best noise immunity. This is indicated by individual +, -, and cmn inputs on the drive or amplifier. The CA4 provides two Analog Output Common pins per analog output to make analog differential wiring easy.



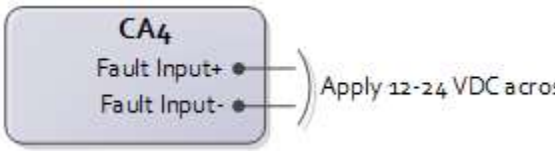
Wiring to Single-ended Inputs



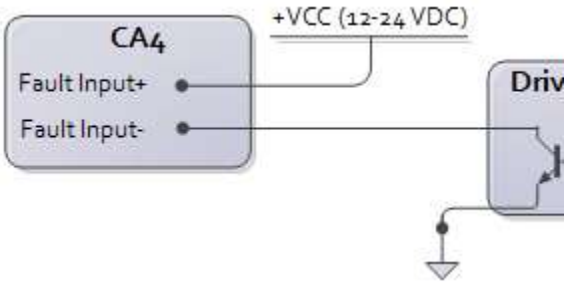
Fault Input Wiring

The Fault input is compatible with signal levels ranging from 12V to 24V. The Fault Input draws 3mA maximum and turns on at 9V. The Fault input turns on when a current flows. The polarity of the voltage is unimportant. See the Fault Input topic for more details.

Generic:



From Open Collector Output:



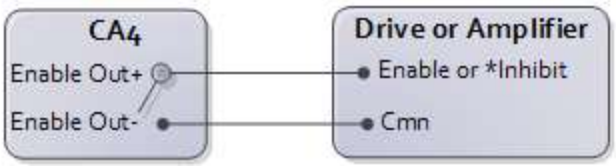
Enable Output Wiring

The Enable output is a solid state relay (SSR). When it is off, it has high impedance, and when on it has low impedance (12 Ω maximum). Because the Enable output is isolated, the user must power it externally. The maximum current and voltage for the Enable output is 75 mA and 30 V.

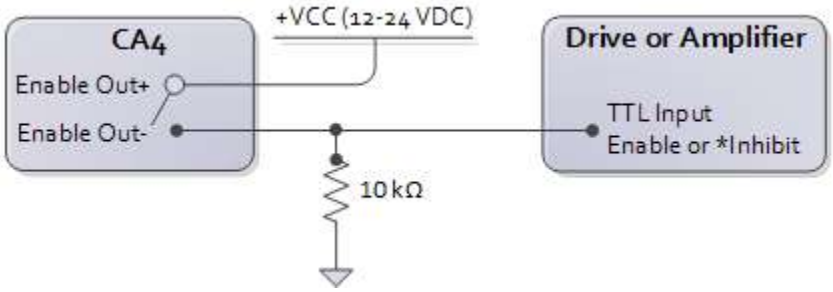
The Enable output has a + and - connection. Both lines must be connected for the output to function. Because both sides of the output are provided, the switches may be independently connected in a high side or low side configuration (that is, with the load (input) on the source or sinking side of the output). See the wiring diagrams below.

See the Enable Output topic for more details on the Enable Output.

To active low Enable input:



To TTL input (high = enable):



Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an "ON" to an "OFF" state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR.

See Also

Wiring Guidelines

10.4.6. CV8 Wiring

This topic covers the wiring of the RMC200 CV8 Module. The CV8 has 8 voltage outputs and also has 8 discrete I/O that can be individually configured as general-purpose inputs, general-purpose outputs, Fault Inputs, or Enable Outputs.

See Wiring Guidelines for general wiring information.

Pin-Out

Terminal Block 1 (top)

Description		Pin			Description
Voltage output 0	CtrlOut0	**1**	**2**	CtrlOut1	Output 1
Output common	Cmn	**3**	**4**	Cmn	
Output common	Cmn	**5**	**6**	Cmn	
Connection to case	Case	**7**	**8**	Case	
Output 2	CtrlOut2	**9**	**10**	CtrlOut3	Output 3
	Cmn	**11**	**12**	Cmn	
	Cmn	**13**	**14**	Cmn	
	Case	**15**	**16**	Case	
Discrete I/O 0+	D0+	**17**	**18**	D1+	Discrete I/O 1

Discrete I/O 0-	D0-	**19**	**20**	D1-	
Discrete I/O 2	D2+	**21**	**22**	D3+	Discrete I/O 3
	D2-	**23**	**24**	D3-	

All the Cmn pins in Terminal Blocks 1 and 2 are internally connected.

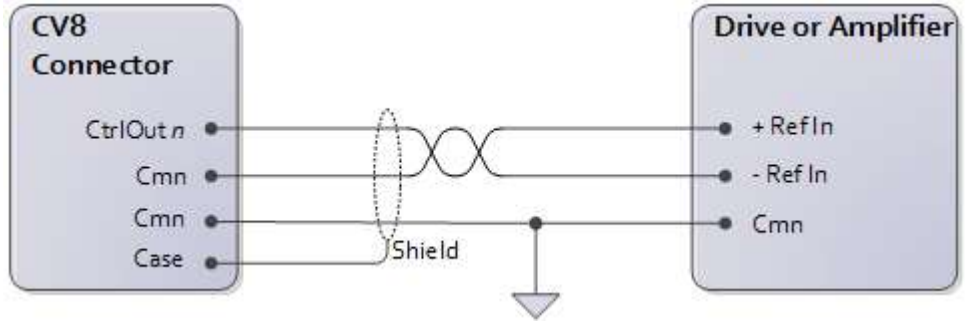
Terminal Block 2 (bottom)

Description		Pin			Description
Output 4	CtrlOut4	**1**	**2**	CtrlOut5	Output 5
	Cmn	**3**	**4**	Cmn	
	Cmn	**5**	**6**	Cmn	
	Case	**7**	**8**	Case	
Output 6	CtrlOut6	**9**	**10**	CtrlOut7	Output 7
	Cmn	**11**	**12**	Cmn	
	Cmn	**13**	**14**	Cmn	
	Case	**15**	**16**	Case	
Discrete I/O 4	D4+	**17**	**18**	D5+	Discrete I/O 5
	D4-	**19**	**20**	D5-	
Discrete I/O 6	D6+	**21**	**22**	D7+	Discrete I/O 7
	D6-	**23**	**24**	D7-	

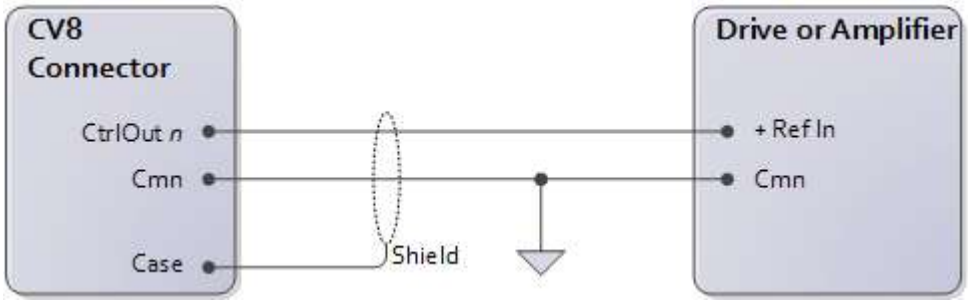
All the Cmn pins in Terminal Blocks 1 and 2 are internally connected.

Wiring to Differential Inputs

Differential inputs provide the best noise immunity. This is indicated by individual +, -, and cmn inputs on the drive or amplifier.



Wiring to Single-ended Inputs

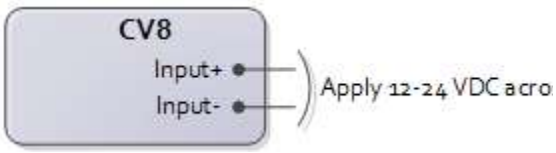


Discrete Input Wiring

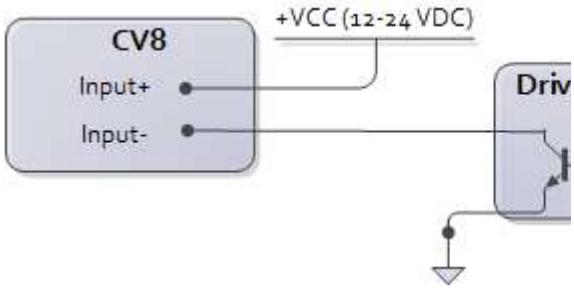
Each discrete input is compatible with signal levels ranging from 12V to 24V. The discrete input draws 3mA maximum and turns on at 9V. The discrete input turns on when a current flows. The

polarity of the voltage is unimportant. See the Fault Input topic for details on using this input as a Fault Input.

Generic:



From Open Collector Output:



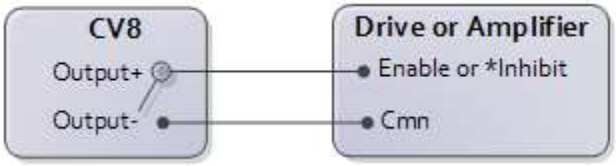
Discrete Output Wiring

Each discrete output is a solid state relay (SSR). When it is off, it has high impedance, and when on it has low impedance (12 Ω maximum). Because the discrete output is isolated, the user must power it externally. The maximum current and voltage for the discrete output is 75 mA and 30 V.

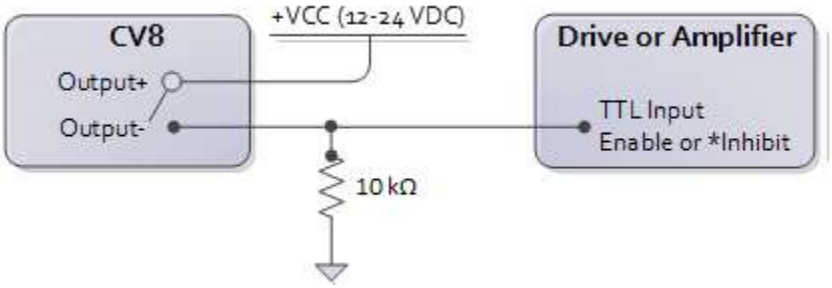
The discrete output has a + and - connection. Both lines must be connected for the output to function. Because both sides of the output are provided, the switches may be independently connected in a high side or low side configuration (that is, with the load (input) on the source or sinking side of the output).

See the Enable Output topic for details on using this output as an Enable Output.

To active low Enable input:



To TTL input (high = enable):



Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an "ON" to an "OFF" state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR.

See Also

Wiring Guidelines

10.4.7. S8 Wiring

The S8 Module can be wired to transducers with SSI interfaces or the magnetostrictive (MDT) Start/Stop and PWM interfaces. Use shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers.

See Wiring Guidelines for general wiring information.

Pin-Out

Terminal Block 1 (top)

Description		Pin			Description
Input 0 Interrogate+ or Clock +	Int/Clk0+	**1**	**2**	Ret/Dat0+	Input 0 Return+ or Data +
Input 0 Interrogate- or Clock -	Int/Clk0-	**3**	**4**	Ret/Dat0-	Input 0 Return- or Data -
Input 0 Shield Connection	Case	**5**	**6**	Cmn	Input 0 Common
Input 1	Int/Clk1+	**7**	**8**	Ret/Dat1+	Input 1
	Int/Clk1-	**9**	**10**	Ret/Dat1-	
	Case	**11**	**12**	Cmn	
Input 2	Int/Clk2+	**13**	**14**	Ret/Dat2+	Input 2
	Int/Clk2-	**15**	**16**	Ret/Dat2-	
	Case	**17**	**18**	Cmn	
Input 3	Int/Clk3+	**19**	**20**	Ret/Dat3+	Input 3
	Int/Clk3-	**21**	**22**	Ret/Dat3-	
	Case	**23**	**24**	Cmn	

All the Cmn pins in Terminal Blocks 1 and 2 are internally connected.

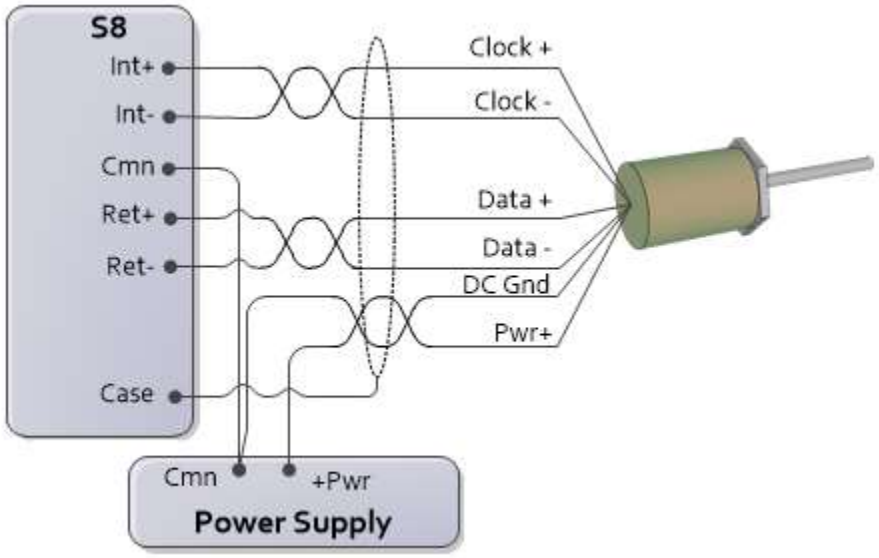
Terminal Block 2 (bottom)

Description		Pin			Description
Input 4	Int/Clk4+	**1**	**2**	Ret/Dat4+	Input 4
	Int/Clk4-	**3**	**4**	Ret/Dat4-	
	Case	**5**	**6**	Cmn	
Input 5	Int/Clk5+	**7**	**8**	Ret/Dat5+	Input 5
	Int/Clk5-	**9**	**10**	Ret/Dat5-	
	Case	**11**	**12**	Cmn	
Input 6	Int/Clk6+	**13**	**14**	Ret/Dat6+	Input 6
	Int/Clk6-	**15**	**16**	Ret/Dat6-	
	Case	**17**	**18**	Cmn	
Input 7	Int/Clk7+	**19**	**20**	Ret/Dat7+	Input 7
	Int/Clk7-	**21**	**22**	Ret/Dat7-	
	Case	**23**	**24**	Cmn	

All the Cmn pins in Terminal Blocks 1 and 2 are internally connected.

SSI Wiring

SSI uses differential line driver (RS422) clock and data signals.



Wiring Instructions

- Connect the transducer DC ground to the S8 Cmn. The Cmn *must* be connected to the transducer, or the signals will not be read correctly!
- The transducer power supply is not provided by the RMC. The user must supply a power supply. The power supply common may be connected to the same common that is connected to the RMC power supply Cmn.

Max Cable Length

SSI Clock Rate **Clock Rate**	**Maximum Cable Length***
100 kHz	2100 ft (640 m)
150 kHz	1360 ft (415 m)
230 kHz	850 ft (255 m)
250 kHz	770 ft (235 m)
375 kHz	475 ft (145 m)
500 kHz	325 ft (99 m)
921 kHz	120 ft (37 m)
971 kHz	110 ft (34 m)
1000 kHz	100 ft (30 m)
1500 kHz	25 ft (7.5 m)
2500 kHz	3 ft (1 m)

* The cable lengths are approximate, and may be affected by the type of wire and transducer.

SSI Monitor Mode

SSI Monitor Mode normally requires daisy-chaining the SSI wiring. When wiring a daisy-chained SSI system, the SSI master should be on one end of the daisy chain with the SSI device on the other end, and any monitoring modules in the middle of the daisy chain.

Wire the SSI device to the first monitoring RMC as illustrated in the diagram above, then daisy chain the first RMC to the other RMCs. Apply termination only to the SSI master.

Termination

SSI signals need to be terminated at the input end. For a typical RMC SSI input that is connected directly to an SSI device (the transducer or encoder), the ±Data signals are an input to the RMC, and need to be terminated. The ±Clock signals are an outputs from the RMC, so ±Clock termination on the RMC side doesn't apply.

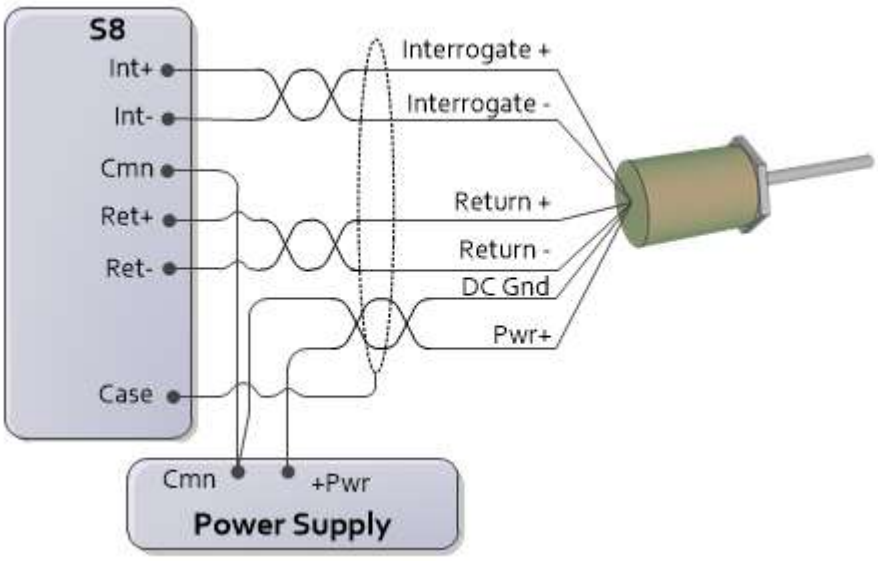
For SSI Monitor, both the ±Data and ±Clock are inputs, so any termination applies to both ±Data and ±Clock. If the SSI Monitor is in the middle of a daisy-chained SSI configuration, it should not have termination, and only the last input in the chain should have termination. If the SSI Monitor is the endpoint of the wiring, then termination should be applied.

For channels configured as SSI input, after assigning the input to an axis, the SSI Termination parameter will be available on the **All** tab in the Axis Parameters Pane, in the **Feedback** section.

Magnetostrictive (MDT) Start/Stop and PWM Wiring

The S8 module interfaces only to transducers with Differential Line Driver (RS422) signals. The transducer pin names may vary by manufacturer.

Pin	Function	Possible Manufacturer Designation
Int/Clk+	MDT Interrogation+	Interrogate+ Input
Int/Clk-	MDT Interrogation-	Interrogate- Input
Cmn	MDT Common	
Ret/Dat+	MDT Return+	Pulse+ Output
Ret/Dat-	MDT Return-	Pulse- Output
Case	Shield Connection	



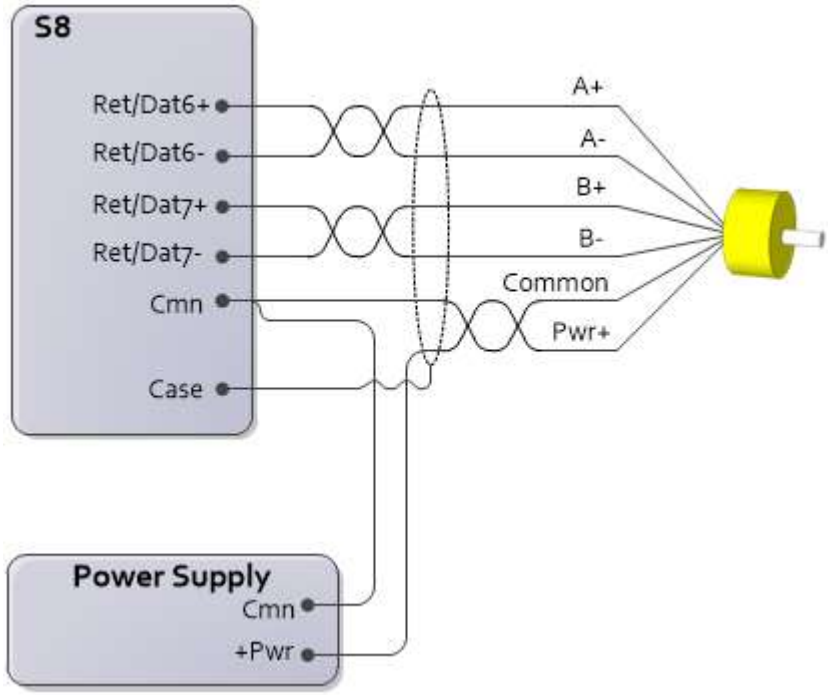
Wiring instructions:

- Connect the transducer DC ground to Cmn. Each axis has two Cmn pins. Either of the 2 pins may be used. The Cmn *must* be connected to the transducer, or the signals will not be read correctly!
- The transducer power supply is not provided by the RMC. The user must supply a power supply. The power supply common may be connected to the same common that is connected to the RMC power supply Cmn.

Quadrature Wiring

The S8 module interfaces only to encoders with 5V Differential Line Driver (RS-422) signals.

Pin	Function
Ret/Dat 6+	A+
Ret/Dat 6-	A-
Int/Clk 6+	unused
Int/Clk 6-	unused
Ret/Dat 7+	B+
Ret/Dat 7-	B-
Int/Clk 7+	unused
Int/Clk 7-	unused
Cmn	common



Wiring instructions:

- Connect the encoder DC ground to any Cmn pin on the S8 module. The Cmn *must* be connected to the encoder, or the signals may not be read correctly!
- The encoder power supply is not provided by the RMC. The user must supply a power supply. The power supply common may be connected to the same common that is connected to the RMC power supply Cmn.

See Also

Wiring Guidelines | S8 Module

10.4.8. LC8 Wiring

This topic covers the wiring of the RMC200 LC8 Module. See Wiring Guidelines for general wiring information.

Pin-Out

Terminal Block 1 (top)

Description		Pin			Description
6.75 V Exciter output	+ExcA	**1**	**2**	In0+	Input 0 +
Input 0 Exciter -	-ExcA	**3**	**4**	In0-	Input 0 -
Input 0 Shield connection	Case	**5**	**6**	S0	Sense -
Input 1	+ExcA	**7**	**8**	In1+	Input 1
	-ExcA	**9**	**10**	In1-	
	Case	**11**	**12**	S1	
Input 2	+ExcA	**13**	**14**	In2+	Input 2
	-ExcA	**15**	**16**	In2-	
	Case	**17**	**18**	S2	
Input 3	+ExcA	**19**	**20**	In3+	Input 3
	-ExcA	**21**	**22**	In3-	
	Case	**23**	**24**	S3	

Exciter Outputs: +6.75Vdc, 80mA per terminal block.

Terminal Block 2 (bottom)

Description		Pin			Description
Input 4	+ExcB	**1**	**2**	In4+	Input 4
	-ExcB	**3**	**4**	In4-	
	Case	**5**	**6**	S4	
Input 5	+ExcB	**7**	**8**	In5+	Input 5
	-ExcB	**9**	**10**	In5-	
	Case	**11**	**12**	S5	
Input 6	+ExcB	**13**	**14**	In6+	Input 6
	-ExcB	**15**	**16**	In6-	
	Case	**17**	**18**	S6	
Input 7	+ExcB	**19**	**20**	In7+	Input 7
	-ExcB	**21**	**22**	In7-	
	Case	**23**	**24**	S7	

Exciter Outputs: +6.75Vdc, 80mA per terminal block.

General LC8 Wiring Guidelines

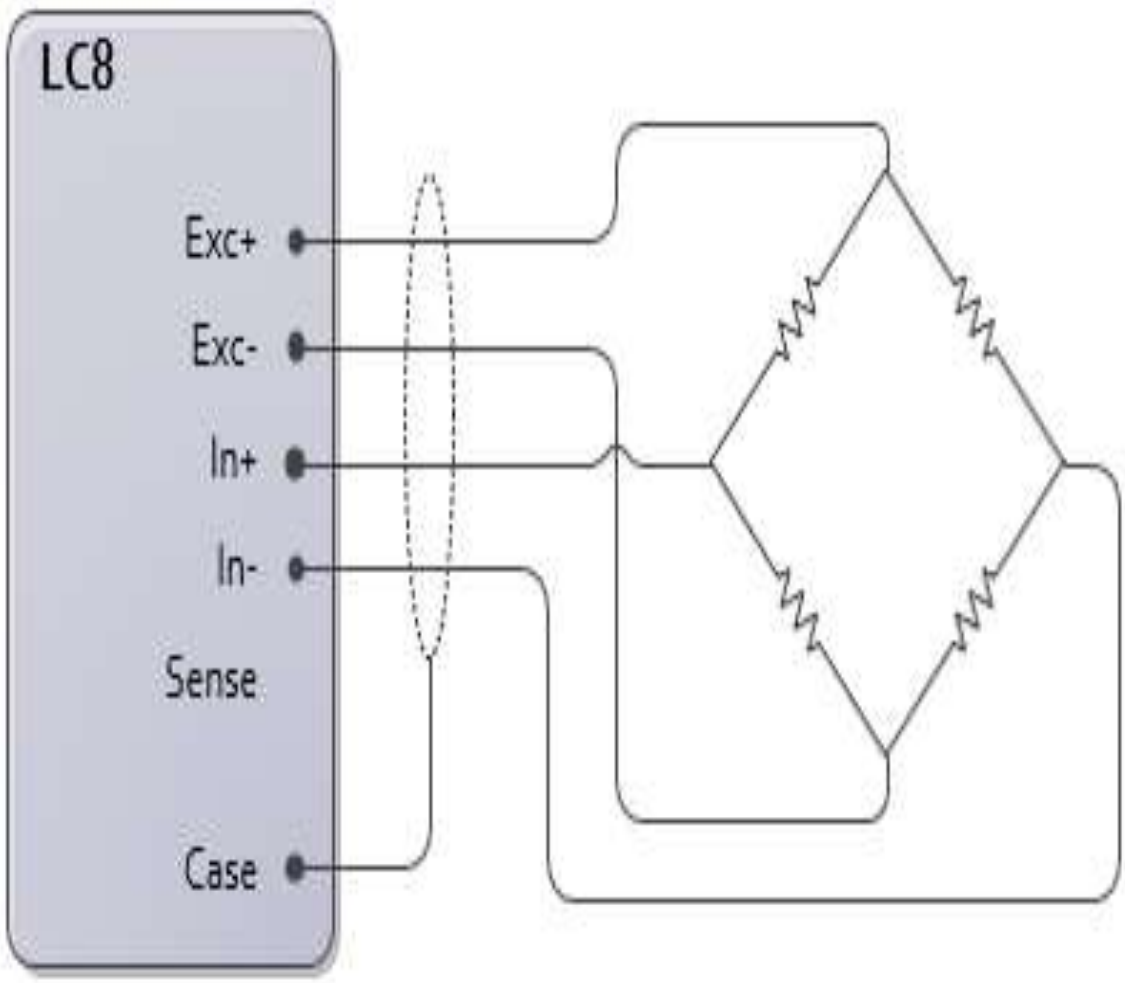
Isolation
Inputs 0-7 are isolated as a single group. There is no isolation between inputs.

Maximum Excitation Current
The total excitation current per terminal block does not exceed 80 mA. The 6.75 V excitation is intended to work with 350 ohm load cells. Load cells with lower resistance are supported, as long as the total excitation current per terminal block does not exceed 80 mA.

Shielding
Wire shielding is very important. The millivolt signal from the load cells is very susceptible to electrical noise. Make sure the shield is connected to the Case pin.

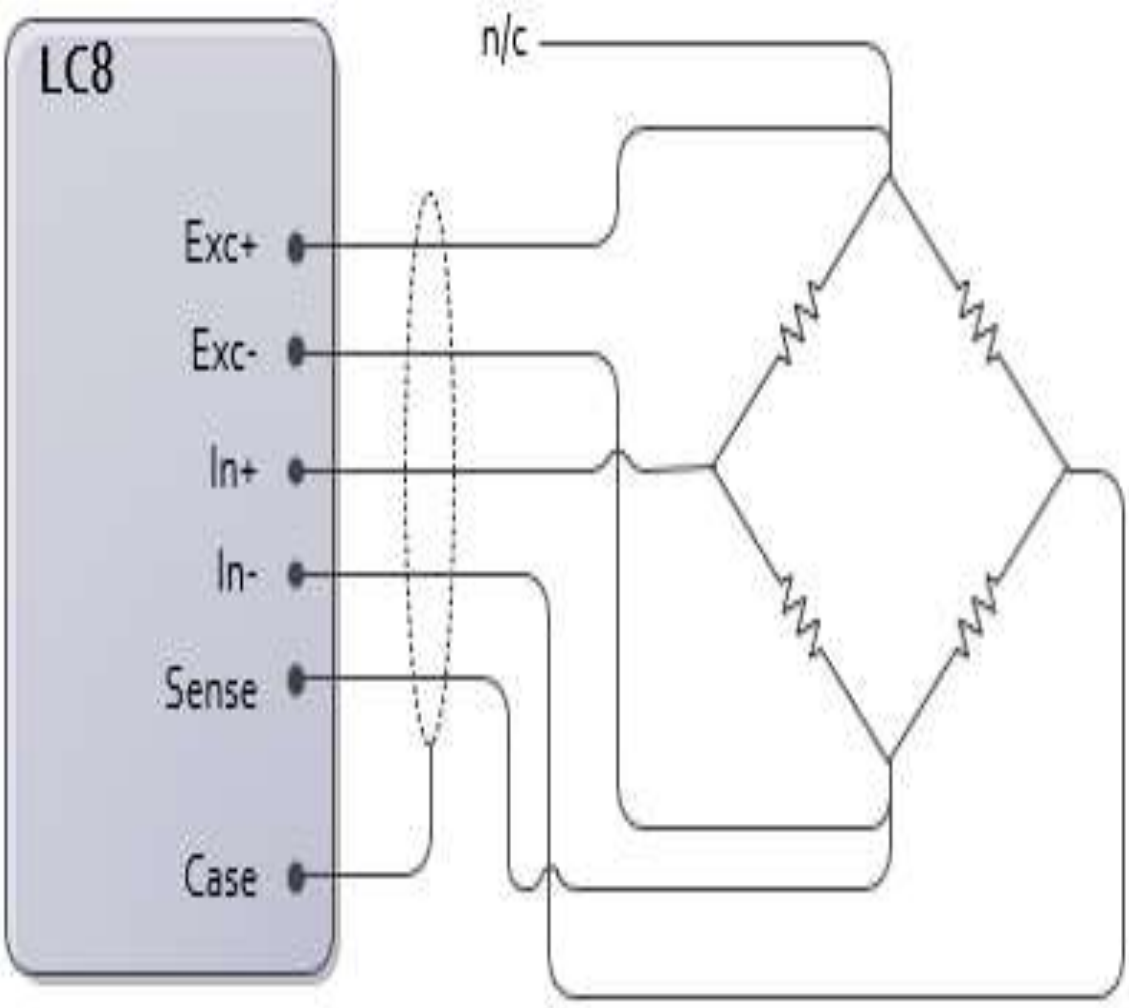
4-Wire Load Cells

Connect 4-wire load cells as follows:



6-Wire Load Cells

Connect 6-wire load cells as follows. To use the continuous wire sense feature, the Exc+ and Exc- wires must be of the same length and gauge. This is because the RMC assumes that the voltage drop from the supply voltage output (Exc+) to the load cell is identical to the voltage drop from the load cell to Exc-.



Using External Excitation

The LC8 module excitation voltage is 6.75 V. However, an external excitation may be applied to the load cell instead. Therefore, if the 6.75 V excitation will cause the load cell signal to exceed the LC8 voltage input, a lower external excitation may be used to reduce the load cell signal to within the limit of the RMC's input. Conversely, a higher excitation voltage may be used (up to 10V) to increase the output from the load cell as long as the the Max Differential Input (±34.25 mV) is not exceeded and the input voltage at In+ or In- relative to Exc- is within the Input Voltage Range (0.6 V to 6.15 V typical). This can improve the signal to noise ratio in particularly sensitive applications.

4-Wire Load Cell with External Excitation

The external Exc- pin must be connected to the LC8 Exc- pin.

6-Wire Load Cell with External Excitation

The external Exc- pin must be connected to the LC8 Exc- pin.

Strain Gauges

Wiring strain gauges to the LC8 module requires a bridge completion circuit. The individual resistances of the Wheatstone bridge completion circuit must be matched to the strain gauge resistance. Common resistance values are 120 or 350 Ohms. Commercially available bridge completion modules typically offer an adjustment to zero the bridge output. See the **Third Party Recommendations** section of Delta's forum for commercially available bridge completion modules.

A single load strain gauge requires three resistors to complete the bridge. The strain gauge may be placed in any location of the bridge. This is called a quarter bridge:

The Sense wire may be omitted if the Adaptive Exctier Mode is not used.

Two strain gauges require two resistors to complete the bridge. This is called a half bridge. The strain gauges form one side of the bridge, and the completion circuit forms the other side:

The Sense wire may be omitted if the Adaptive Exciter Mode is not used.

See Also

Wiring Guidelines | LC8 Module | Load Cell Fundamentals

10.4.9. A8 Wiring

This topic covers the wiring of the RMC200 A8 Module. See Wiring Guidelines for general wiring information.

Pin-Out

Terminal Block 1 (top)

Description		Pin			Description
Input 0 10V Exciter output	10V Exc	**1**	**2**	In0+V	Input 0+ Voltage
Input 0 Common	Cmn	**3**	**4**	In0+mA	Input 0+ Current
Input 0 shield connection	Case	**5**	**6**	In0-	Input 0-
Input 1	10V Exc	**7**	**8**	In1+V	
	Cmn	**9**	**10**	In1+mA	Input 1
	Case	**11**	**12**	In1-	
Input 2	10V Exc	**13**	**14**	In2+V	
	Cmn	**15**	**16**	In2+mA	Input 2
	Case	**17**	**18**	In2-	
Input 3	10V Exc	**19**	**20**	In3+V	
	Cmn	**21**	**22**	In3+mA	Input 3
	Case	**23**	**24**	In3-	

All the Cmn pins in Terminal Blocks 1 and 2 are internally connected.

Terminal Block 2 (bottom)

Description		Pin			Description
Input 4	10V Exc	**1**	**2**	In4+V	Input 4
	Cmn	**3**	**4**	In4+mA	
	Case	**5**	**6**	In4-	
Input 5	10V Exc	**7**	**8**	In5+V	Input 5
	Cmn	**9**	**10**	In5+mA	
	Case	**11**	**12**	In5-	
Input 6	10V Exc	**13**	**14**	In6+V	Input 6
	Cmn	**15**	**16**	In6+mA	
	Case	**17**	**18**	In6-	
Input 7	10V Exc	**19**	**20**	In7+V	Input 7
	Cmn	**21**	**22**	In7+mA	
	Case	**23**	**24**	In7-	

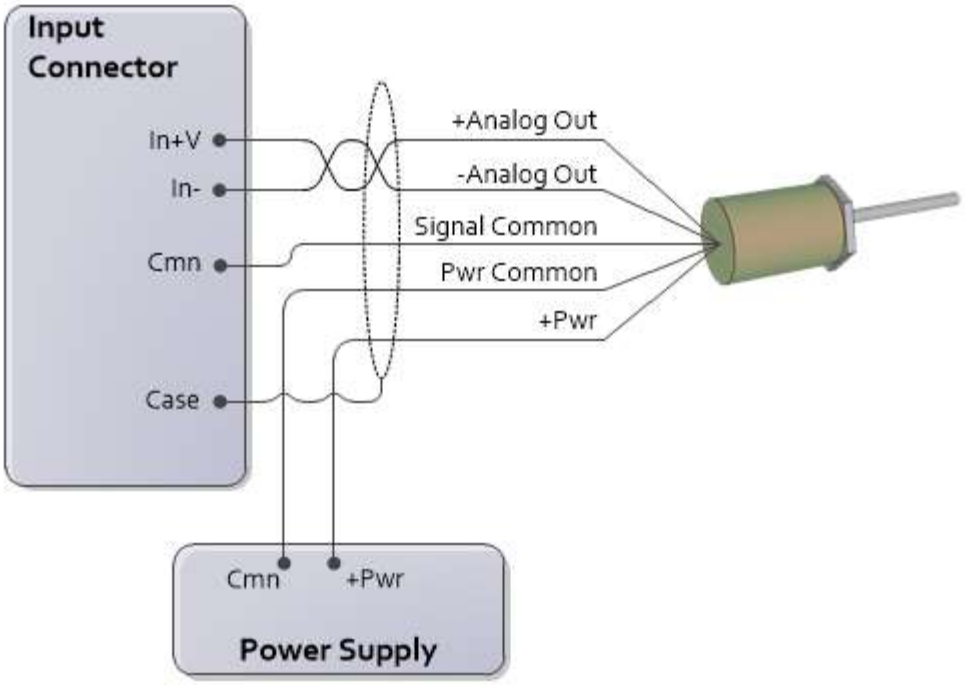
All the Cmn pins in Terminal Blocks 1 and 2 are internally connected.

Voltage Transducers

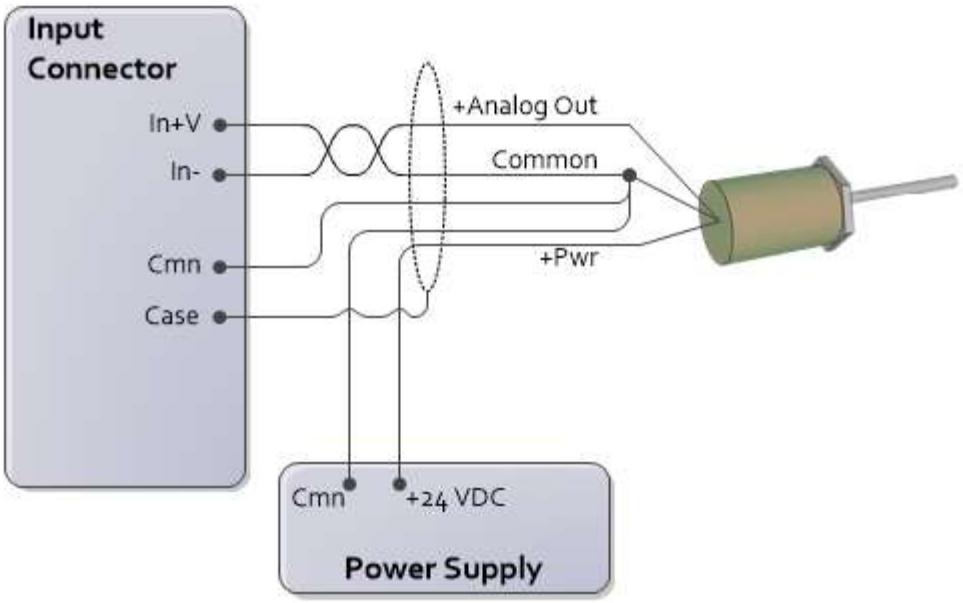
To reduce electrical interference:

- In- and Cmn must be connected, either internal to the transducer or externally as close as possible to the transducer.
- Use individually shielded twisted-pair wire.
- Connect cable shield to earth ground on one end only.
- If transducer has only one common, connect Pwr Supply Common and RMC Cmn to it. For best results, make this connection at the transducer.
- If the input is disconnected, input voltage will be pulled down ≤ -10V.

4- or 5-wire

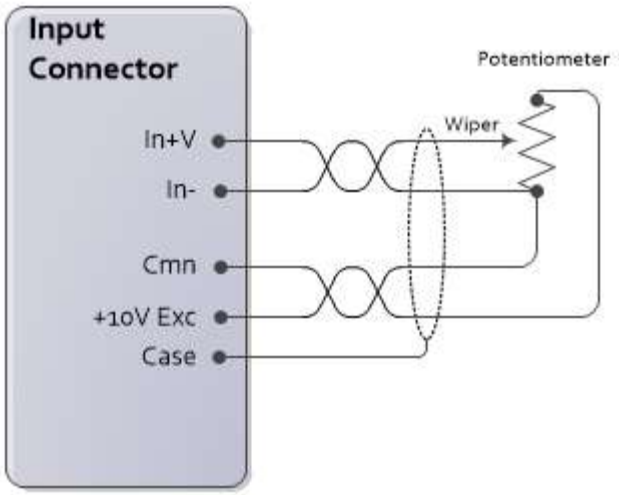


3-wire



Potentiometer

Use the Exciter pin to increase the measurement accuracy of the potentiometer.

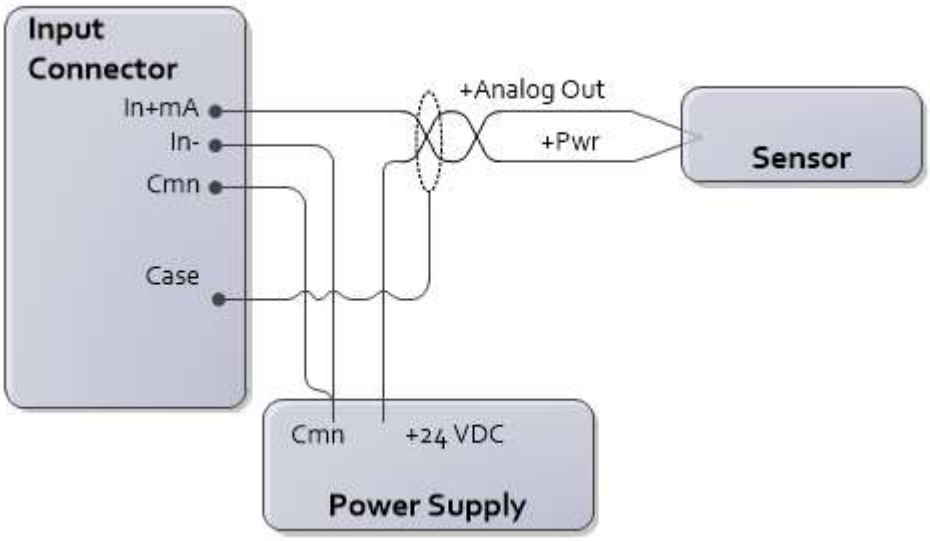


Current Transducers

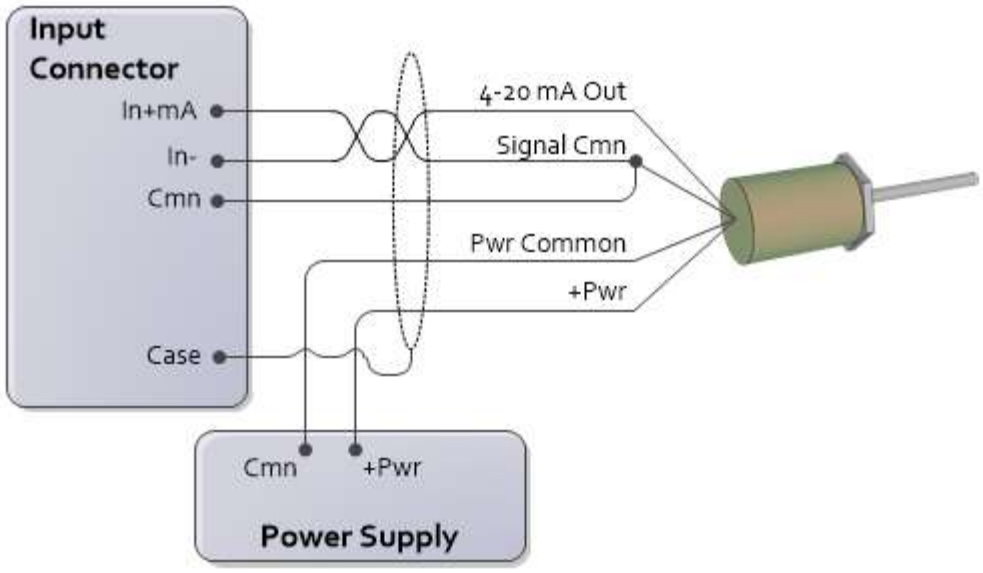
To reduce electrical interference:

- Use individually shielded twisted-pair wire.
- Connect cable shield to ground on one end only.

2-wire



4-wire



The connection of In- to Cmn should be made as close as possible to the transducer.

See Also

Wiring Guidelines

10.4.10. Q4 Wiring

The Q4 Module can be wired to quadrature encoder signals. For best noise immunity and speed performance, Delta recommends using RS-422 encoders, with shielded twisted pairs for all connections to inputs and outputs. Route the transducer wiring separate from other wiring. You must provide the power supplies needed by your transducers.

See Wiring Guidelines for general wiring information.

Pin-Out

Terminal Block 1 (top)

Description		Pin			Description
Input 0 Z+	Z0+	**1**	**2**	A0+	Input 0 A+
Input 0 Z-	Z0-	**3**	**4**	A0-	Input 0 A-
Input 0 Home	Hm0	**5**	**6**	B0+	Input 0 B+
Encoder and Home Common	Cmn	**7**	**8**	B0-	Input 0 B-
Shield Connection	Case	**9**	**10**	Cmn	Encoder and Home Common
Input 1 Z+	Z1+	**11**	**12**	A1+	Input 1 A+
Input 1 Z-	Z1-	**13**	**14**	A1-	Input 1 A-
Input 1 Home	Hm1	**15**	**16**	B1+	Input 1 B+
Encoder and Home Common	Cmn	**17**	**18**	B1-	Input 1 B-
Shield Connection	Case	**19**	**20**	Cmn	Encoder and Home Common

Registration 0 Input	Reg0+	**21**	**22**	Reg1+	Registration 1 Input
	Reg0-	**23**	**24**	Reg1-	

All the Cmn pins in Terminal Blocks 1 and 2 are internally connected.

Terminal Block 2 (bottom)

Description	Pin				Description
Input 2 Z+	Z2+	**1**	**2**	A2+	Input 2 A+
Input 2 Z-	Z2-	**3**	**4**	A2-	Input 2 A-
Input 2 Home	Hm2	**5**	**6**	B2+	Input 2 B+
Encoder and Home Common	Cmn	**7**	**8**	B2-	Input 2 B-
Shield Connection	Case	**9**	**10**	Cmn	Encoder and Home Common
Input 3 Z+	Z3+	**11**	**12**	A3+	Input 3 A+
Input 3 Z-	Z3-	**13**	**14**	A3-	Input 3 A-
Input 3 Home	Hm3	**15**	**16**	B3+	Input 3 B+
Encoder and Home Common	Cmn	**17**	**18**	B3-	Input 3 B-
Shield Connection	Case	**19**	**20**	Cmn	Encoder and Home Common
Registration 2 Input	Reg2+	**21**	**22**	Reg3+	Registration 3 Input
	Reg2-	**23**	**24**	Reg3-	

All the Cmn pins in Terminal Blocks 1 and 2 are internally connected.

Differential Quadrature Wiring

Use this wiring diagram for encoders with the following outputs:

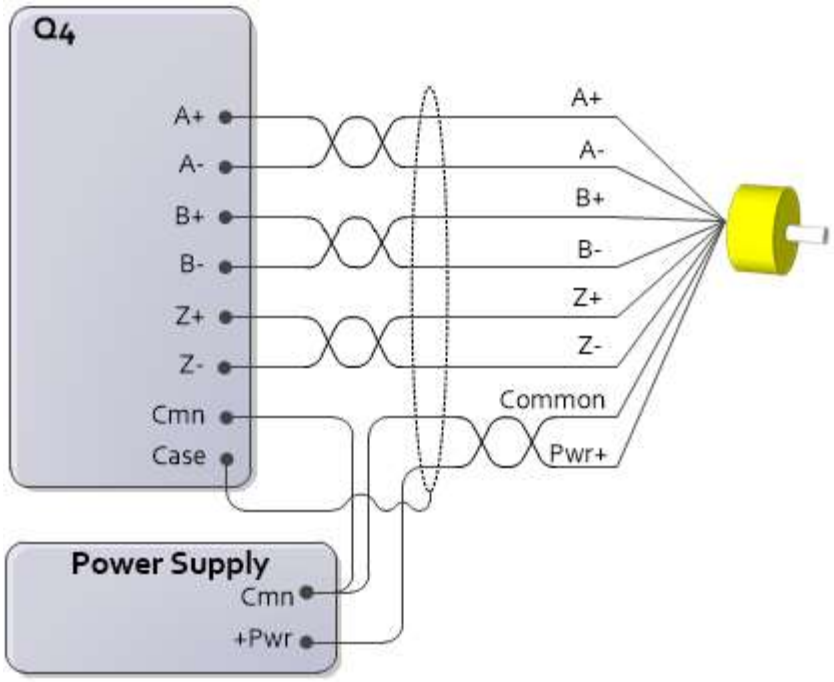
- RS-422
- 5V Differential
- Differential HTL (High Threshold Logic) for signals from 12V to 24 V
- Push-pull with complements

Axis Parameters

For this wiring configuration, set the AB Input Type and Z Input Type axis parameters as follows:

Encoder Signal	AB Input Type or Z Input Type
RS-422 5V Differential Push-pull with complements, 5V	RS-422
Differential HTL, 12-24V Push-pull with complements, 12-24V	Differential HTL

Diagram



Single-ended Quadrature Wiring (Not Recommended)

Delta does not recommend single-ended quadrature encoders due to poor noise immunity and a low maximum count frequency.

Use this wiring diagram for encoders with the following outputs:

- TTL*
- Push-pull without complements, 5-24V

*If these signals have complements, the complements are not used in this wiring configuration.

Axis Parameters

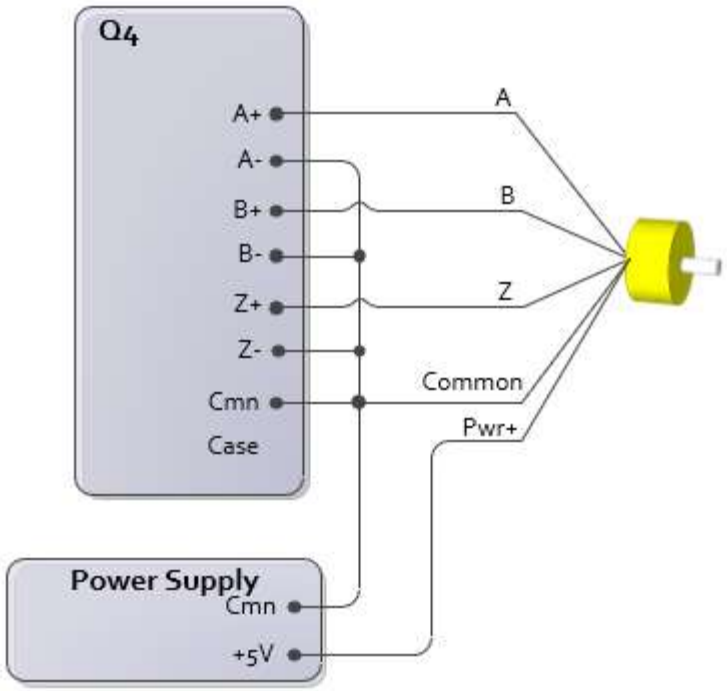
For this wiring configuration, set the AB Input Type, Z Input Type, and HTL Threshold axis parameters as follows:

Encoder Signal	AB Input Type or Z Input Type	HTL Threshold
TTL* Push-pull, 5V*	TTL	n/a
Push-pull without complements, 12V	Single-ended HTL	7V
Push-pull without complements, 24V	Single-ended HTL	12V

*Some encoders may not have adequate drive capability to support termination on the input. If termination is desired, termination can be enabled in software and A-, B-, or Z- need to be connected to Cmn.

Diagram

For single-ended connection, A-, B- and Z- must be connected to Cmn.



Home Input Wiring

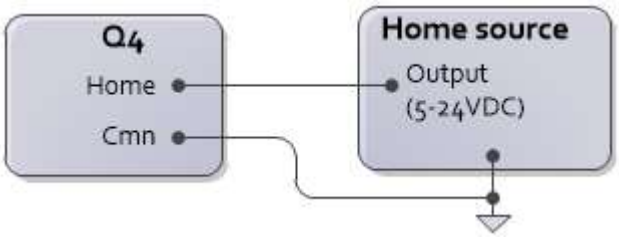
Use this wiring diagram for the home inputs on the Q4 module. The home input supports 5-24V levels.

Axis Parameters

For this wiring configuration, set the H Input Type axis parameter as follows:

Home Signal	H Input Type
TTL	TTL
12-24V	DI

Diagram



Registration Input Wiring

Use this wiring diagram for the registration inputs on the Q4 module. The registration inputs are individually isolated and support 5-24V levels.

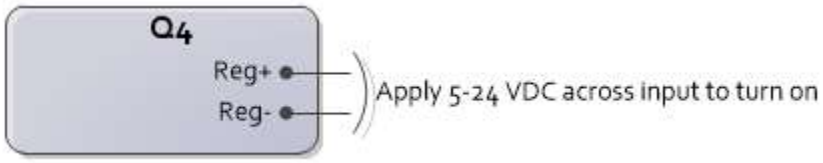
Axis Parameters

For this wiring configuration, set the R Input Type axis parameter as follows:

Home Signal	R Input Type
TTL	5V

12-24V	12-24V

Diagram



See Also

Wiring Guidelines | Q4 Module

10.4.11. U14 Wiring

This topic covers the wiring of the RMC200 U14 Module. See Wiring Guidelines for general wiring information.

Pin-Out

Terminal Block 1 (top)

Description		Pin			Description
Analog Input Common	AInCmn	**1**	**2**	AIn0+	Analog Input 0+
Shield Connection	Case	**3**	**4**	AIn0-	Analog Input 0-
Analog Input 1	AInCmn	**5**	**6**	AIn1+	Analog Input 1
	Case	**7**	**8**	AIn1-	
Analog Input 2	AInCmn	**9**	**10**	AIn2+	Analog Input 2
	Case	**11**	**12**	AIn2-	
Analog Input 3	AInCmn	**13**	**14**	AIn3+	Analog Input 3
	Case	**15**	**16**	AIn3-	
Analog Output Common	AOutCmn	**17**	**18**	AOut0	Analog Output 0
Shield connection	Case	**19**	**20**	AOutCmn	Analog Output Common
Analog Output 1	AOutCmn	**21**	**22**	AOut1	Analog Output 1
	Case	**23**	**24**	AOutCmn	

The AInCmn pins are internally connected. The AOutCmn pins are internally connected.

Terminal Block 2 (bottom)

Description		Pin			Description
Registration/Quad Z 0+	Reg/Z0+	**1**	**2**	Clk/A0+	SSI Clk/MDT Int/Quad A 0+
Registration/Quad Z 0-	Reg/Z0-	**3**	**4**	Clk/A0-	SSI Clk/MDT Int/Quad A 0-
SSI/MDT/Quad Common	S/QCmn	**5**	**6**	Dat/B0+	SSI Data/MDT Ret/Quad B 0+
Shield Connection	Case	**7**	**8**	Dat/B0-	SSI Data/MDT Ret/Quad B 0-
Registration/Quad Z 1+	Reg/Z1+	**9**	**10**	Clk/A1+	High-speed Channel 1
Registration/Quad Z 1-	Reg/Z1-	**11**	**12**	Clk/A1-	
SSI/MDT/Quad Common	S/QCmn	**13**	**14**	Dat/B1+	

Shield Connection	Case	**15**	**16**	Dat/B1-	
Discrete I/O 0+	D0+	**17**	**18**	D1+	Discrete I/O 1
Discrete I/O 0-	D0-	**19**	**20**	D1-	
Discrete I/O 2	D2+	**21**	**22**	D3+	Discrete I/O 3
	D2-	**23**	**24**	D3-	

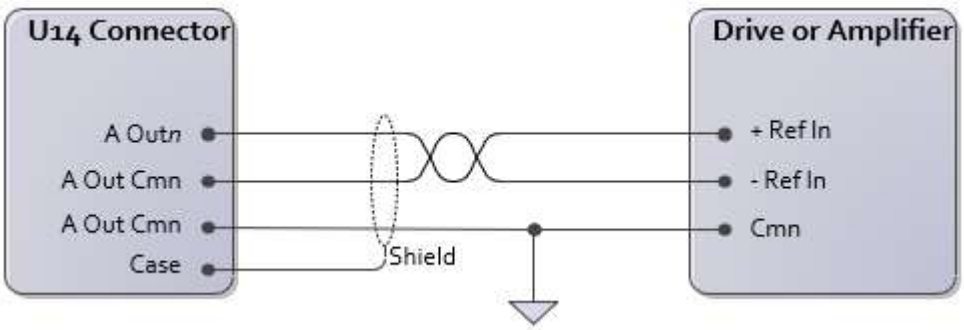
The S/QCmn pins are internally connected.

U14 Analog Outputs

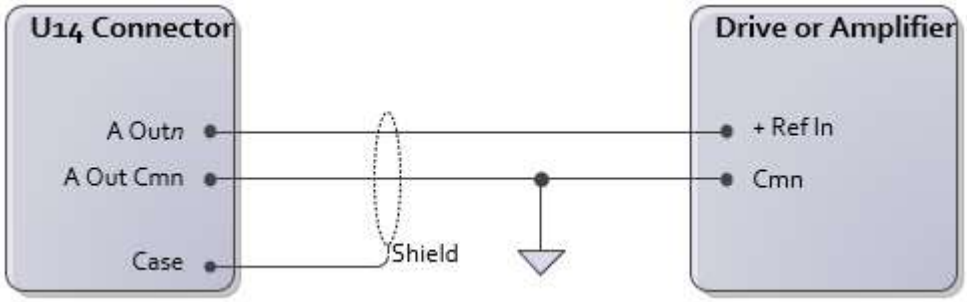
The U14 analog outputs provide ±10 V, 4-20 mA, or ±20 mA.

Wiring the U14 Analog Output to Differential Inputs

Differential inputs provide the best noise immunity. This is indicated by individual +, -, and cmn inputs on the drive or amplifier. The U14 provides two Analog Output Common pins per analog output to make analog differential wiring easy.



Wiring the U14 Analog Output to Single-ended Inputs



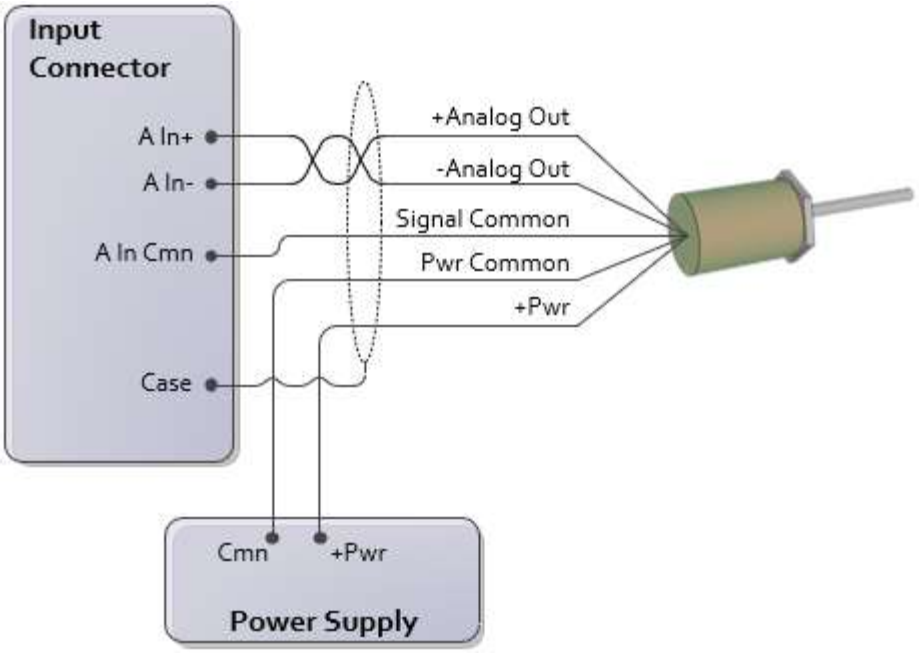
U14 Analog Inputs

Voltage Transducers

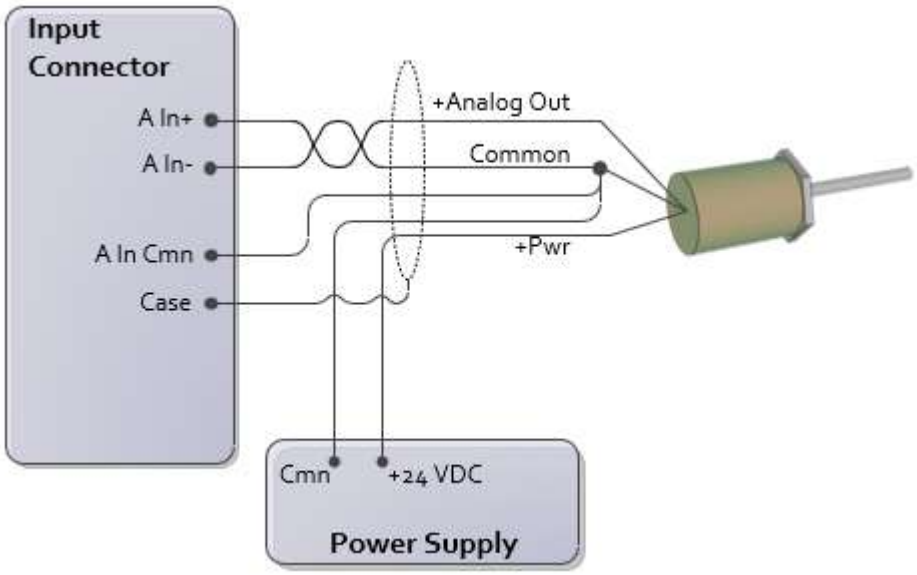
To reduce electrical interference:

- In- and Cmn must be connected, either internal to the transducer or externally as close as possible to the transducer.
- Use individually shielded twisted-pair wire.
- Connect cable shield to earth ground on one end only.
- If transducer has only one common, connect Pwr Supply Common and RMC Cmn to it. For best results, make this connection at the transducer.

4- or 5-wire Voltage



3-wire Voltage

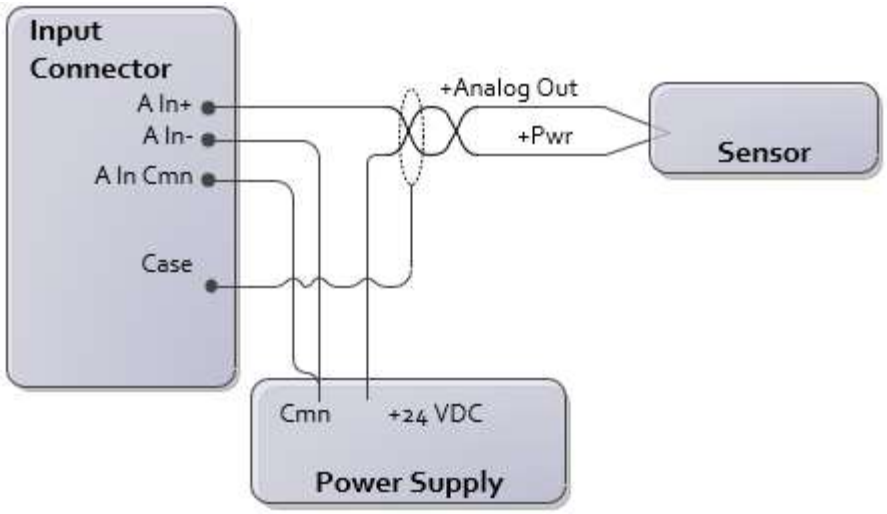


Current Transducers

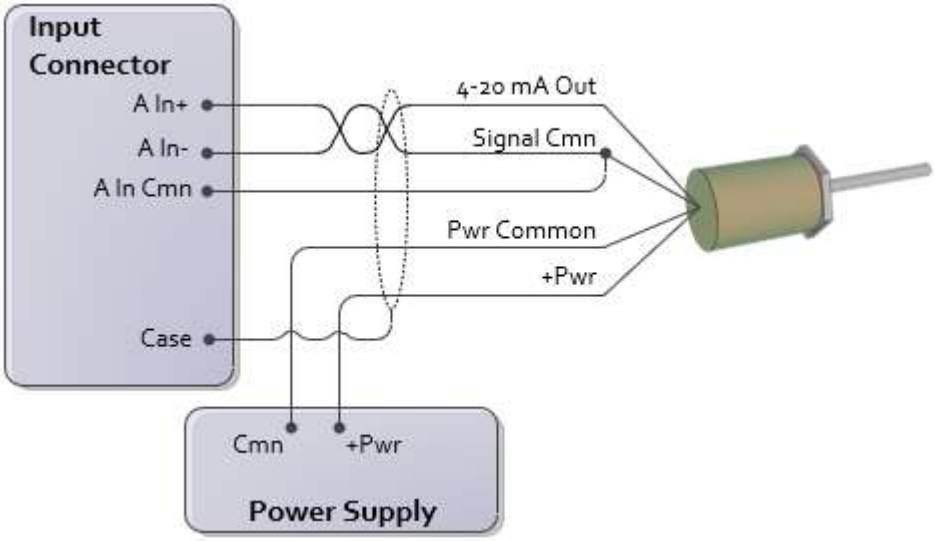
To reduce electrical interference:

- Use individually shielded twisted-pair wire.
- Connect cable shield to ground on one end only.

2-wire Current



4-wire Current

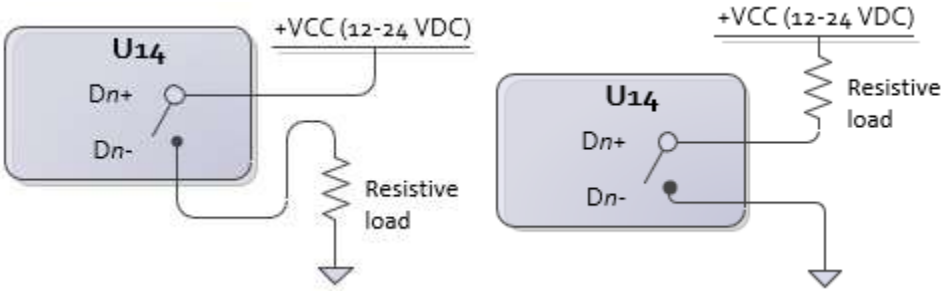


The connection of In- to Cmn should be made as close as possible to the transducer.

U14 Discrete Outputs D0-D3

The DI/O point must first be configured in RMCTools to be an output. The U14 discrete outputs are solid state relays. The off state is high impedance, the on state is low impedance (8Ω max, 5Ω typical). Max current 75 mA. Max voltage 30 V. Each DI/O point is individually isolated.

Outputs can be wired in either a high-side or low-side configuration.



The load resistance must be sized such that the maximum current through the SSR does not exceed 75 mA. The maximum current is calculated with the following equation:

Current = V_{cc} / R_{Load}

For example, if the supply voltage V_{cc} is 24V, and the load resistance is 480Ω, the current will be:

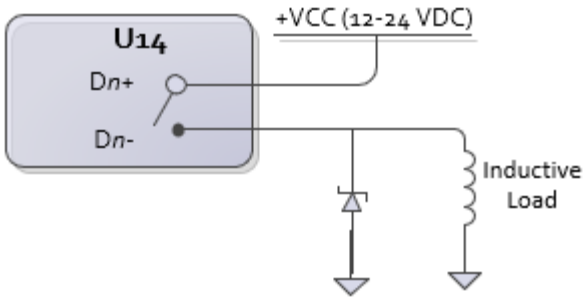
Current = 24V / 480Ω = 50mA

which is under the 75 mA limit.

Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an "ON" to an "OFF" state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR.

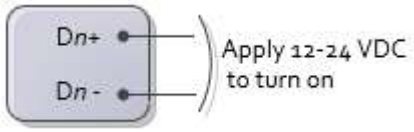
SSR switching inductive inductive load with high-side configuration:



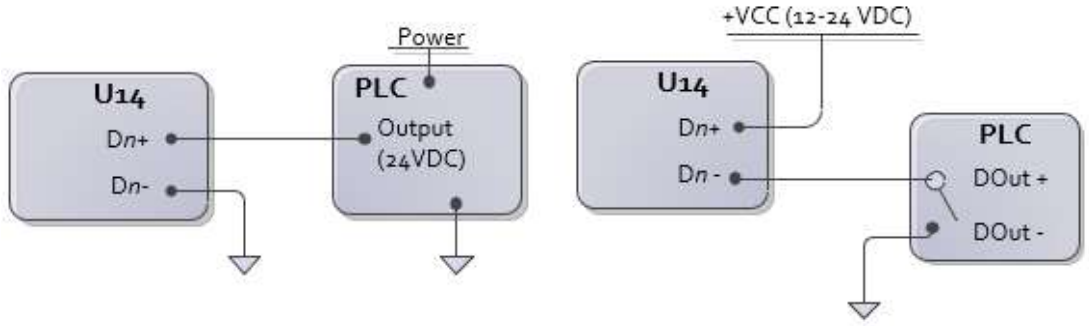
When calculating the current through the SSR, use the same type of calculation as above, with the coil resistance as the load.

U14 Discrete Inputs D0-D3

The DI/O point must first be configured in RMCTools to be an input. Each input is individually isolated and has **D*n*+** and **D*n*-** connections. To turn on a discrete input, apply a 12-24 Vdc voltage. Max current draw is 7 mA.



Examples:



U14 Reg/Z Inputs

When using the Reg/Z Inputs as an Index (Z) input from the encoder together with the A and B quadrature signals, refer to the **U14 Quadrature Channel** section below instead.

The Reg/Z inputs are individually isolated. When the high-speed channel associated with the Reg/Z input is in quadrature mode and is assigned to an axis, the Reg/Z input can be configured as single-ended or

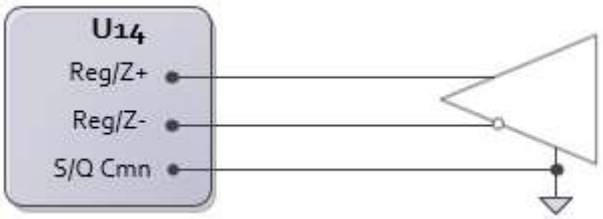
differential, and supports 5-24V levels. When the high-speed channel associated with the Reg/Z input is *not* in quadrature mode, or is *not* assigned to an axis, the Reg/Z input will be set as a single-ended HTL input with a 12V threshold.

Differential

For this wiring configuration, set the Z Input Type axis parameter as follows:

Signal	Z Input Type
RS-422 5V Differential Push-pull with complements, 5V	RS-422
Differential HTL, 12-24V Push-pull with complements, 12-24V	Differential HTL

Note: Before connecting HTL signals to the input, set the Z Input Type parameter to HTL.
Note: S/Q Cmn must be connected to the common used by the Reg/Z input signal source.
Note: The Reg/Z input shares the same common (S/Q Cmn) as the A and B inputs.



Single-Ended

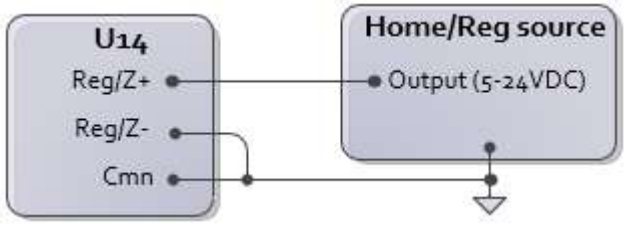
For this wiring configuration, set the Z Input Type and HTL Threshold axis parameters as follows:

Signal	Z Input Type	HTL Threshold
TTL	TTL	n/a
PNP prox switch	DI (better noise immunity than HTL)	n/a
Push-pull without complements, 12V*	Single-ended HTL	7V
Push-pull without complements, 24V*	Single-ended HTL	12V

*These signal input types are available, but not recommended when using the Reg/Z as a home or registration input. These are typically only used as a Z signal together with the A and B signal from the encoder.

Note: Before connecting HTL signals to the input, set the Z Input Type parameter to HTL.

Note: For single-ended connection, Reg/Z- must be connected to Cmn.



U14 Quadrature Channel

Differential Quadrature Wiring

Use this wiring diagram for encoders with the following outputs:

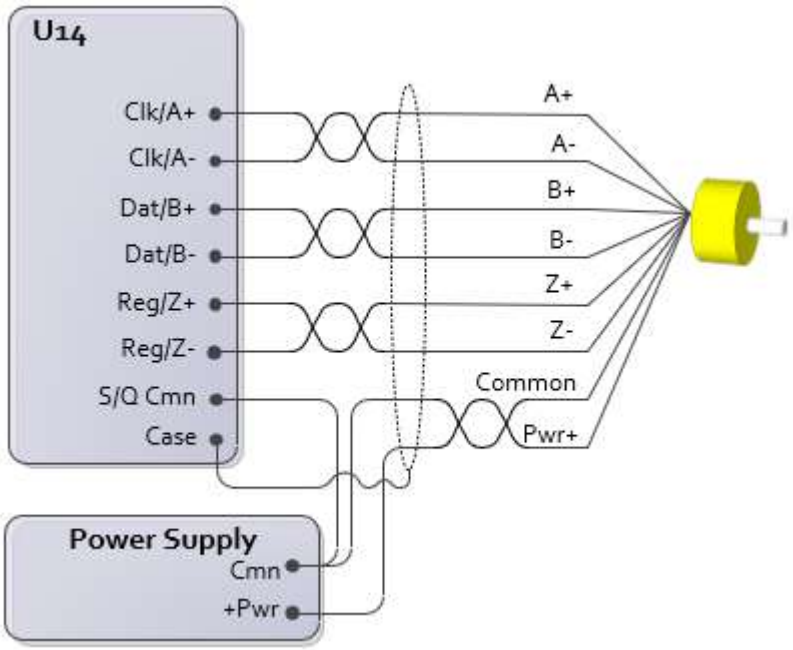
- RS-422
- 5V Differential

- Differential HTL (High Threshold Logic) for signals from 12V to 24 V
- Push-pull with complements

For this wiring configuration, set the AB Input Type and Z Input Type axis parameters as follows:

Encoder Signal	AB Input Type or Z Input Type
RS-422 5V Differential Push-pull with complements, 5V	RS-422
Differential HTL, 12-24V Push-pull with complements, 12-24V	Differential HTL

Note: Before connecting HTL signals to the input, set the AB Input Type and Z Input Type parameters to HTL!



Single-ended Quadrature Wiring (Not Recommended)

Delta does not recommend single-ended quadrature encoders due to poor noise immunity and a low maximum count frequency.

Use this wiring diagram for encoders with the following outputs:

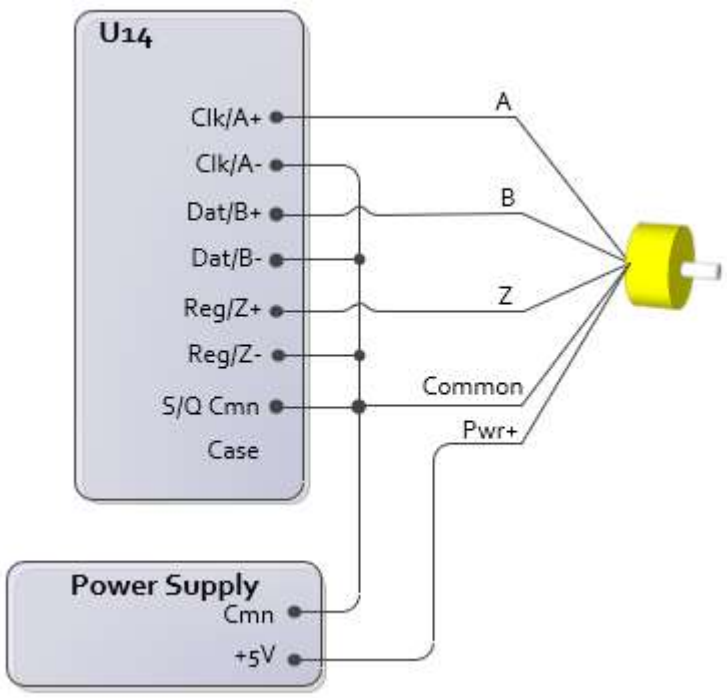
- TTL*
- Push-pull without complements, 5-24V

*If these signals have complements, the complements are not used in this wiring configuration.

For this wiring configuration, set the AB Input Type, Z Input Type, and HTL Threshold axis parameters as follows:

Encoder Signal	AB Input Type or Z Input Type	HTL Threshold
TTL* Push-pull, 5V*	TTL	n/a
Push-pull without complements, 12V	Single-ended HTL	7V
Push-pull without complements, 24V	Single-ended HTL	12V

*Some encoders may not have adequate drive capability to support termination on the input. If termination is desired, termination can be enabled in software and Clk/A-, Dat/B-, or Reg/Z- need to be connected to S/Q Cmn.



Termination

Termination is software selectable via an axis parameter. Input termination should always be applied when connecting a point-to-point RS-422 encoder signal. For daisy-chained quadrature feedback (connecting one encoder to multiple inputs), only the last input in the chain should have termination. The termination parameter is available only if the input is RS-422 or TTL.

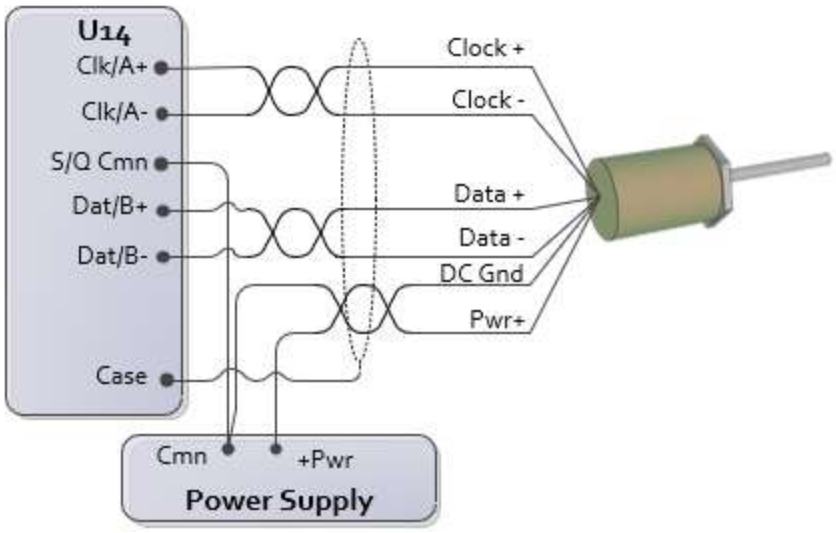
After configuring the U14 channel as quadrature, and assigning the input to an axis, the AB Termination and Z Termination parameters will be available on the **All** tab in the Axis Parameters Pane, in the **Feedback** section. Choose **Enabled** to apply termination.

U14 SSI Channel

SSI uses differential line driver (RS422) clock and data signals.

SSI Input

When using the high-speed channel as an SSI input, wire it to the transducer or encoder as follows:



Wiring Instructions

- Connect the transducer DC ground to the U14 S/Q Cmn. The S/Q Cmn *must* be connected to the transducer, or the signals will not be read correctly!
- The transducer power supply is not provided by the RMC. The user must supply a power supply. The power supply common may be connected to the same common that is connected to the RMC power supply Cmn.

Max Cable Length

SSI Clock Rate Clock Rate	Maximum Cable Length*
100 kHz	2100 ft (640 m)
150 kHz	1360 ft (415 m)
230 kHz	850 ft (255 m)
250 kHz	770 ft (235 m)
375 kHz	475 ft (145 m)
500 kHz	325 ft (99 m)
921 kHz	120 ft (37 m)
971 kHz	110 ft (34 m)
1000 kHz	100 ft (30 m)
1500 kHz	25 ft (7.5 m)
2500 kHz	3 ft (1 m)

* The cable lengths are approximate, and may be affected by the type of wire and transducer.

Termination

SSI signals need to be terminated at the input end. For a typical RMC SSI input that is connected directly to an SSI device (the transducer or encoder), the ±Data signals are an input to the RMC, and need to be terminated. The ±Clock signals are an outputs from the RMC, so ±Clock termination on the RMC side doesn't apply.

Termination is on by default, as indicated by the SSI Termination axis parameter in the associated axis.

SSI Monitor Mode

SSI Monitor Mode normally requires daisy-chaining the SSI wiring. When wiring a daisy-chained SSI system, the SSI master should be on one end of the daisy chain with the SSI device on the other end, and any monitoring modules in the middle of the daisy chain.

Wire the SSI device to the first monitoring RMC as illustrated in the **SSI Inputs** section above, then daisy chain the first RMC to the other RMCs. Apply termination only to the SSI master.

Termination

SSI signals need to be terminated at the input end. For SSI Monitor, both the ±Data and ±Clock on the RMC are inputs, so any termination applies to both ±Data and ±Clock. If the SSI Monitor is in the middle of a daisy-chained SSI configuration, it should not have termination, and only the last input in the chain should have termination. If the SSI Monitor is the endpoint of the wiring, then termination should be applied.

For U14 channels configured as SSI Monitor, after assigning the input to an axis, the SSI Termination parameter will be available on the **All** tab in the Axis Parameters Pane, in the **Feedback** section.

SSI Echo Mode

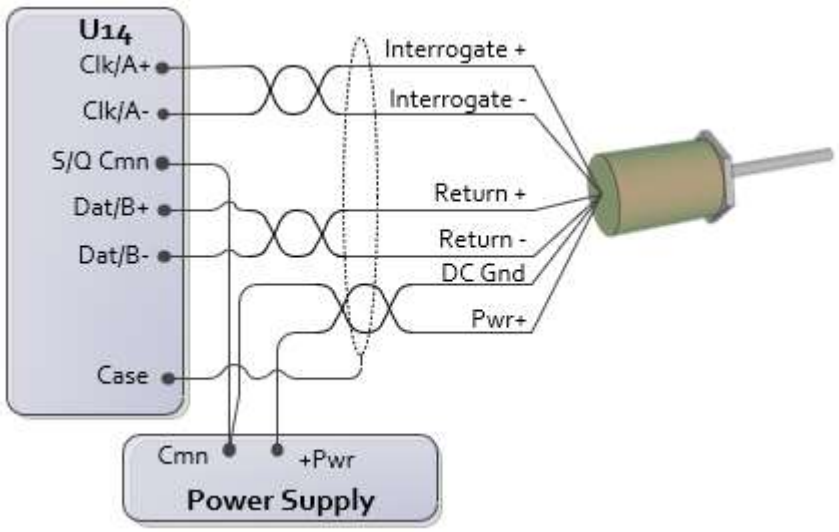
For SSI Echo Mode, the U14 channel acts like an SSI encoder. A controller sends the ±Clock signal to the U14 channel, and the U14 channel returns the ±Data signal.

Wire the U14 channel 1 as if it is the SSI encoder, with the remote controller being the SSI master. This is simply the opposite of the **SSI Input** wiring diagram above.

U14 MDT Channel

The U14 module interfaces only to transducers with Differential Line Driver (RS422) signals. The transducer pin names may vary by manufacturer.

Pin	Function	Possible Manufacturer Designation
Clk/A+	MDT Interrogation+	Interrogate+ Input
Clk/A-	MDT Interrogation-	Interrogate- Input
S/Q Cmn	MDT Common	
Dat/B+	MDT Return+	Pulse+ Output
Dat/B-	MDT Return-	Pulse- Output
Case	Shield Connection	



Wiring instructions:

- Connect the transducer DC ground to the U14 S/Q Cmn. The S/Q Cmn *must* be connected to the transducer, or the signals will not be read correctly!
- The transducer power supply is not provided by the RMC. The user must supply a power supply. The power supply common may be connected to the same common that is connected to the RMC power supply Cmn.

See Also

Wiring Guidelines | U14 Module

10.4.12. D24 Wiring

This topic covers the wiring of the RMC200 D24 Module. See Wiring Guidelines for general wiring information.

The D24 discrete I/O are organized into 4 sections:

	DI/O #	Features
Group A	**0-7**	**Input or output** – nominal 24V. Each I/O points is individually software-configurable as an input or output. Each group is isolated. Within each group, all inputs share the same input common, and all outputs share the same output common.
Group B	**8-15**	
Group C	**16-19**	
Fast Inputs	**20-23**	**Inputs only** – 5V or 24V. High-speed, supports quadrature encoder inputs and pulse train inputs. Each input is individually isolated.

Pin-Out

Terminal Block 1 (top)

Description		Pin			Description
Shield connection	Case	**1**	**2**	Case	Shield connection
Common for group A outputs	OutCmnA	**3**	**4**	OutCmnB	Common for group B outputs
Group A, DI/O point 0	D0	**5**	**6**	D8	Group B, DI/O point 8
Group A, DI/O point 1	D1	**7**	**8**	D9	Group B, DI/O point 9
Group A, DI/O point 2	D2	**9**	**10**	D10	Group B, DI/O point 10
Group A, DI/O point 3	D3	**11**	**12**	D11	Group B, DI/O point 11
Group A, DI/O point 4	D4	**13**	**14**	D12	Group B, DI/O point 12
Group A, DI/O point 5	D5	**15**	**16**	D13	Group B, DI/O point 13
Group A, DI/O point 6	D6	**17**	**18**	D14	Group B, DI/O point 14
Group A, DI/O point 7	D7	**19**	**20**	D15	Group B, DI/O point 15
Common for group A inputs	InCmnA	**21**	**22**	InCmnB	Common for group B inputs
Shield connection	Case	**23**	**24**	Case	Shield connection

Terminal Block 2 (bottom)

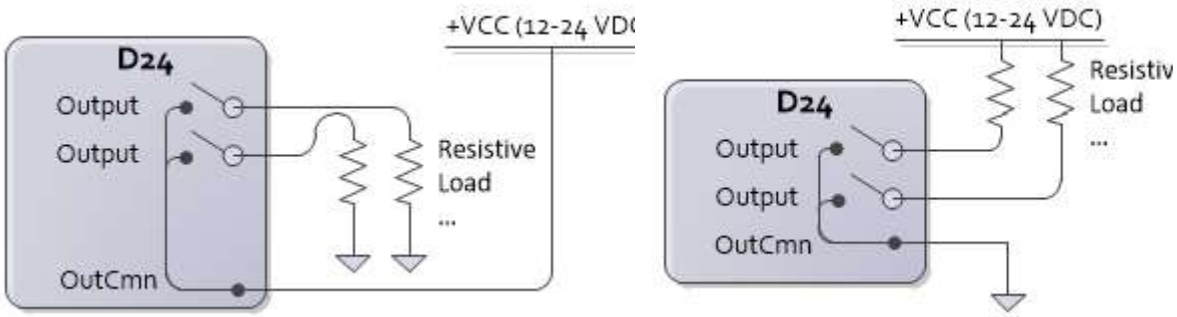
Description		Pin			Description
Shield connection	Case	**1**	**2**	Din20+	Input 20 + for 12-24 Vdc signals
Shield connection	Case	**3**	**4**	Din20+5V	Input 20 + for 5 Vdc signals

Shield connection	Case	**5**	**6**	Din20-	Input 20 – for all signals
Common for group C outputs	OutCmnC	**7**	**8**	Din21+	
Group C, DI/O point 16	D16	**9**	**10**	Din21+5V	Input 21
Group C, DI/O point 17	D17	**11**	**12**	Din21-	
Group C, DI/O point 18	D17	**13**	**14**	Din22+	
Group C, DI/O point 19	D19	**15**	**16**	Din22+5V	Input 22
Common for group C inputs	InCmnC	**17**	**18**	Din22-	
Shield connection	Case	**19**	**20**	Din23+	
Shield connection	Case	**21**	**22**	Din23+5V	Input 23
Shield connection	Case	**23**	**24**	Din23-	

Discrete Outputs

The D24 discrete outputs are solid state relays. The off state is high impedance, the on state is low impedance (8Ω max, 5Ω typical). Max current 75 mA. Max voltage 30 V.

Outputs can be wired in either a high-side or low-side configuration. Because all the outputs in a group share a common, all outputs in the same group must be wired the same.



The load resistance must be sized such that the maximum current through the SSR does not exceed 75 mA. The maximum current is calculated with the following equation:

$$\text{Current} = V_{cc} / R_{Load}$$

For example, if the supply voltage V_{cc} is 24V, and the load resistance is 480Ω, the current will be:

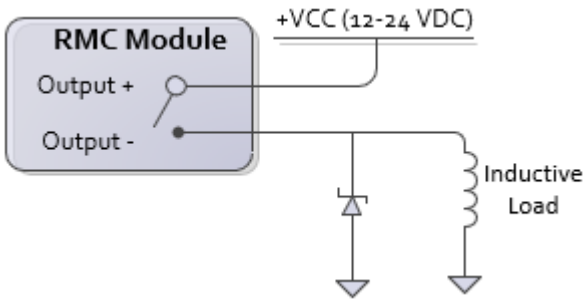
$$\text{Current} = 24\Omega / 480\Omega = 50\text{mA}$$

which is under the 75 mA limit.

Using Outputs with Inductive Loads

External fuses should be used to protect the SSRs if there is a possibility of over-current. When switching inductive loads, it is important to place a diode or tranzorb across the load to protect the switch when transitioning from an ”ON” to an ”OFF” state. Otherwise, the collapsing magnetic field can cause a reverse voltage spike in excess of the 30 V rating of the SSR.

SSR switching inductive inductive load with high-side configuration:



When calculating the current through the SSR, use the same type of calculation as above, with the coil resistance as the load.

Discrete Inputs

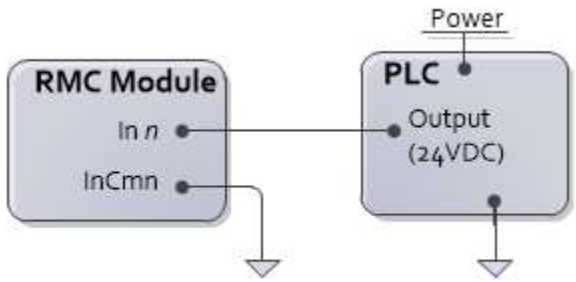
To turn on a discrete input, apply a voltage of the correct level. The polarity is unimportant for inputs 0-19, and is important for 20-23.

	Inputs 0-19	Inputs 20-23
Signal Levels	12-26.4 VDC	5-26.4 VDC
Max Current Draw	3 mA	7 mA

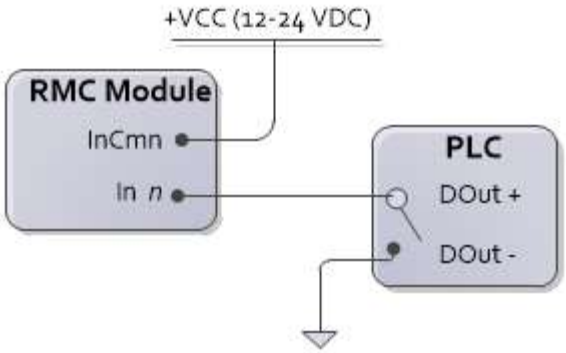
Inputs 0-19

The inputs 0-19 are polarity-independent and can be sinking or sourcing. Inputs 0-19 are divided in three groups: 0-7, 8-15, 16-19. Each group shares a common, so all inputs in a group must be wired the same.

Used with a sourcing output

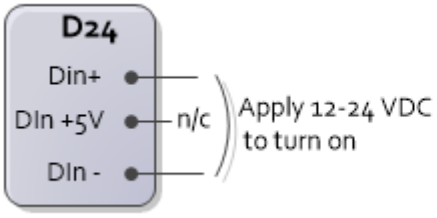


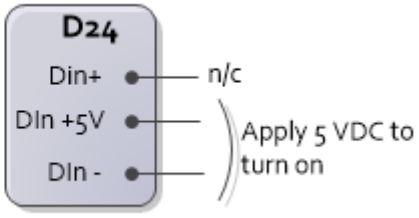
Used with a sinking output



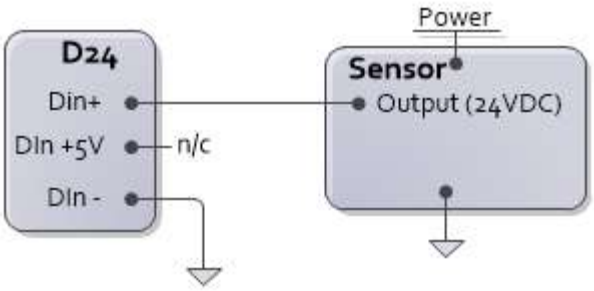
Inputs 20-23

Each input is individually isolated and has **Din+**, **Din +5V**, and **Din–** connections. Use only **Din+** or **Din+5V**, not both.





Example:



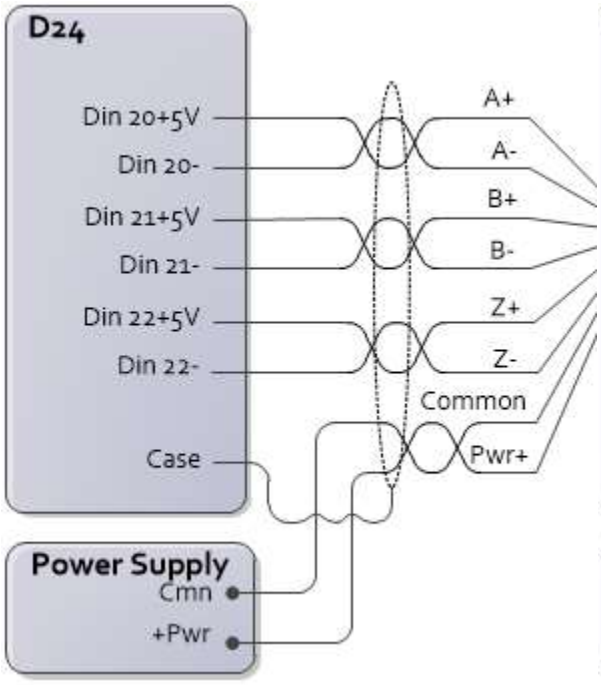
Differential Quadrature Wiring without Wire Break Detection

Use these wiring diagrams for encoders with the outputs listed below. The D24 will not be able to detect a broken wire with this wiring configuration.

- 5V Differential
- Differential HTL (High Threshold Logic) for signals from 12V to 24 V
- RS-422*

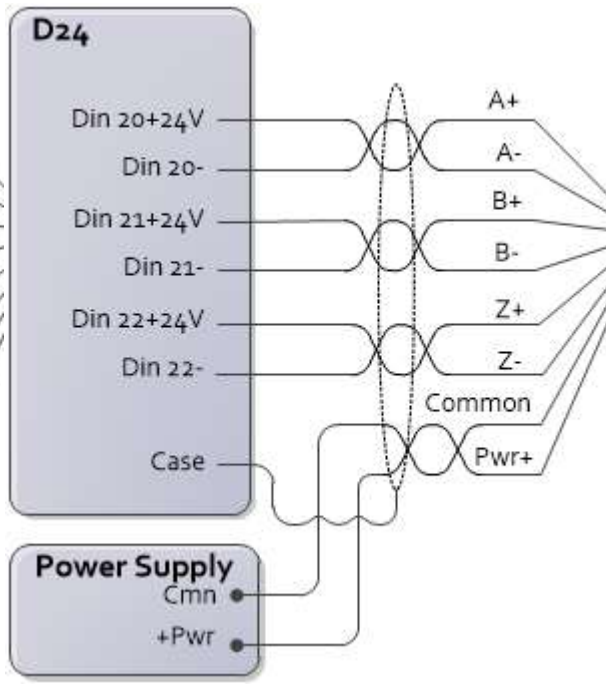
*RS-422 will only work in this configuration if the differential output is greater than 3.5V. Some RS-422 drivers may not provide sufficient voltage.

With Z (Index)
5V Differential or RS-422 with differential output > 3.5V

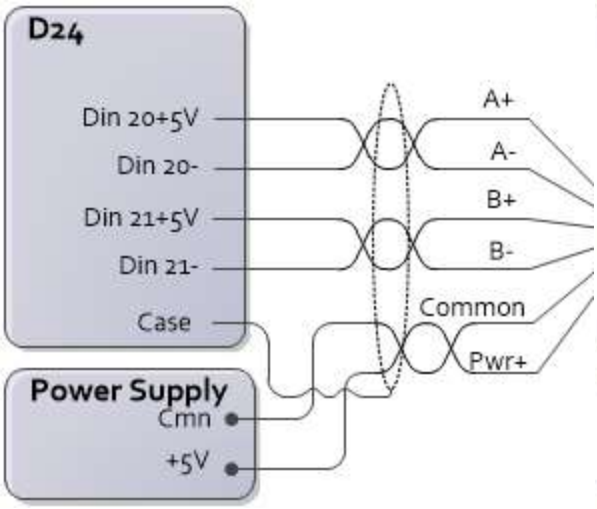


First quadrature input without Z (Index)
5V Differential or RS-422 with differential output > 3.5V

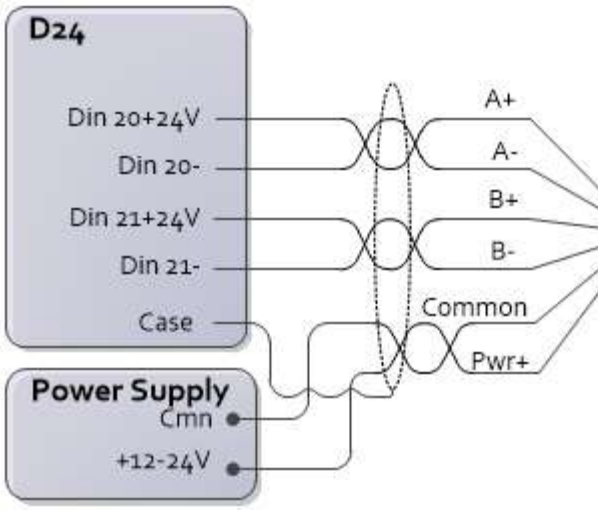
With Z (Index)
12-24V Differential or Push-Pull 12-24V with complements



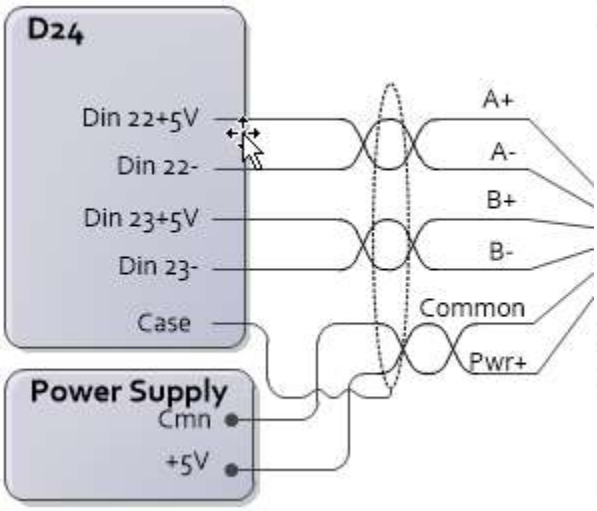
First quadrature input without Z (Index)
12-24V Differential or Push-Pull 12-24V with complements

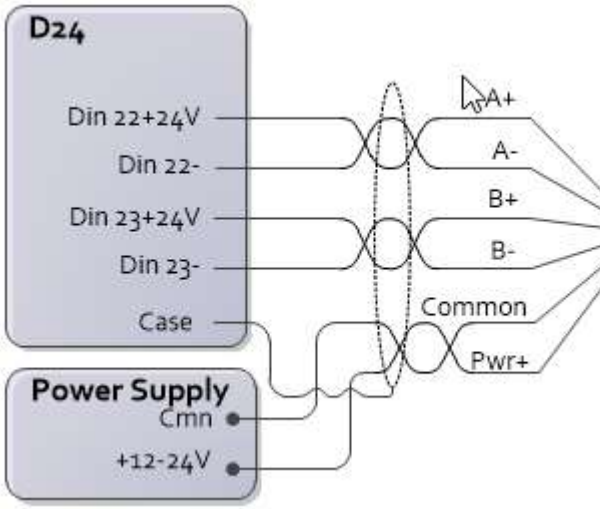


Second quadrature input without Z (Index) 5V Differential or RS-422 with differential output > 3.5V



Second quadrature input without Z (Index) 12-24V Differential or Push-Pull 12-24V with complements





Differential Quadrature Wiring with Wire Break Detection

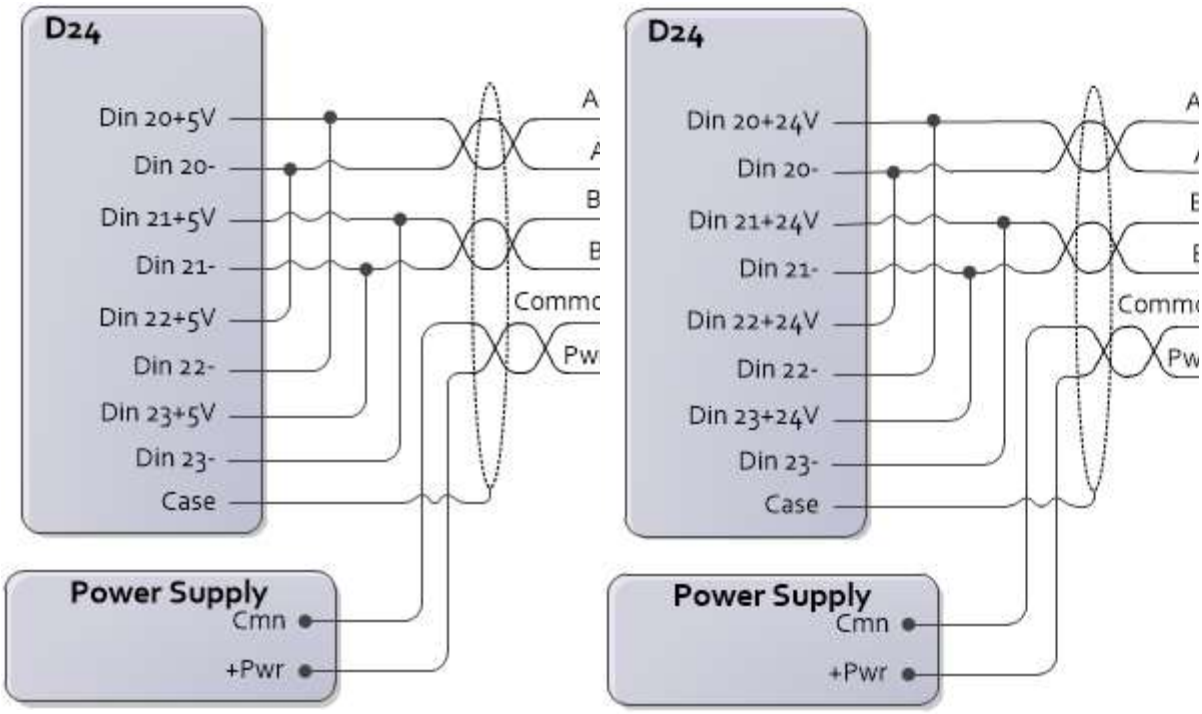
Use these wiring diagrams for encoders with the outputs listed below. The D24 will be able to detect a broken wire with this wiring configuration.

- 5V Differential
- Differential HTL (High Threshold Logic) for signals from 12V to 24 V
- RS-422*

*RS-422 will only work in this configuration if the differential output is greater than 3.5V. Some RS-422 drivers may not provide sufficient voltage.

5V Differential or RS-422 with differential output greater than 3.5V

12-24V Differential or Push-Pull 12-24V with complements



Single-ended Quadrature Wiring

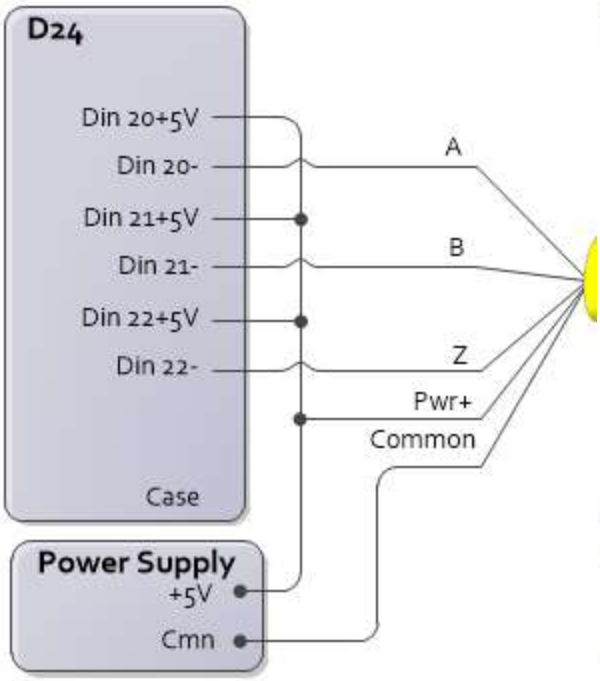
Use these wiring diagrams for encoders with the outputs listed below. The D24 will be able to detect a broken wire with this wiring configuration.

- TTL
- Push-pull 5V-24V without complements
- Open collector from 5V to 24 V
- RS-422 (3V)* - Connect A+ and B+ in place of A and B in the diagrams. Encoder wires A- and B- remain unconnected.

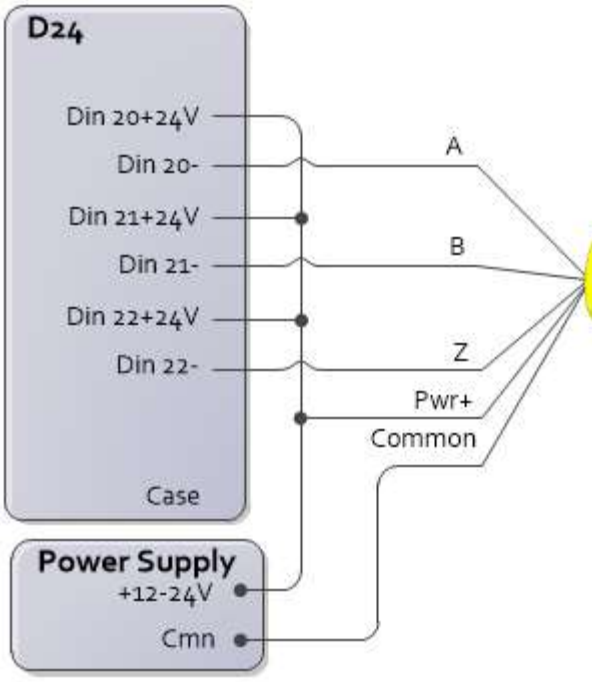
*Use RS-422 in this configuration only if the differential output is less than 3.5V. Otherwise, use the differential wiring.

Quadrature input with Z (Index)
TTL or 5V signal levels

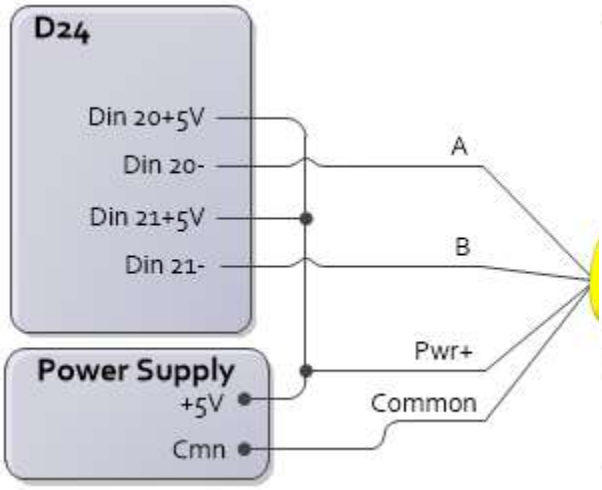
Quadrature input with Z (Index)
12-24V signal levels



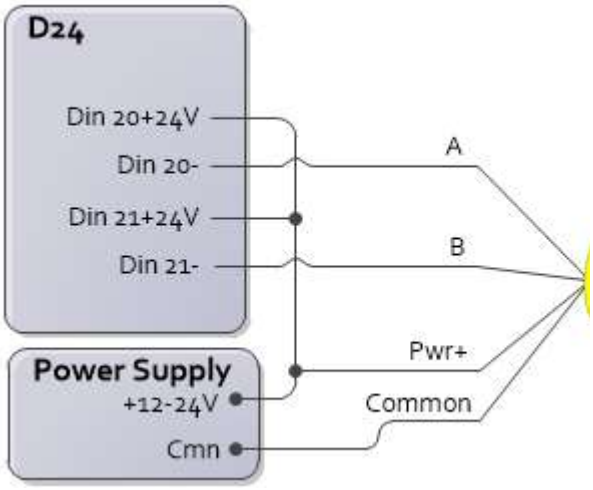
First quadrature input without Z (Index)
TTL or 5V signal levels



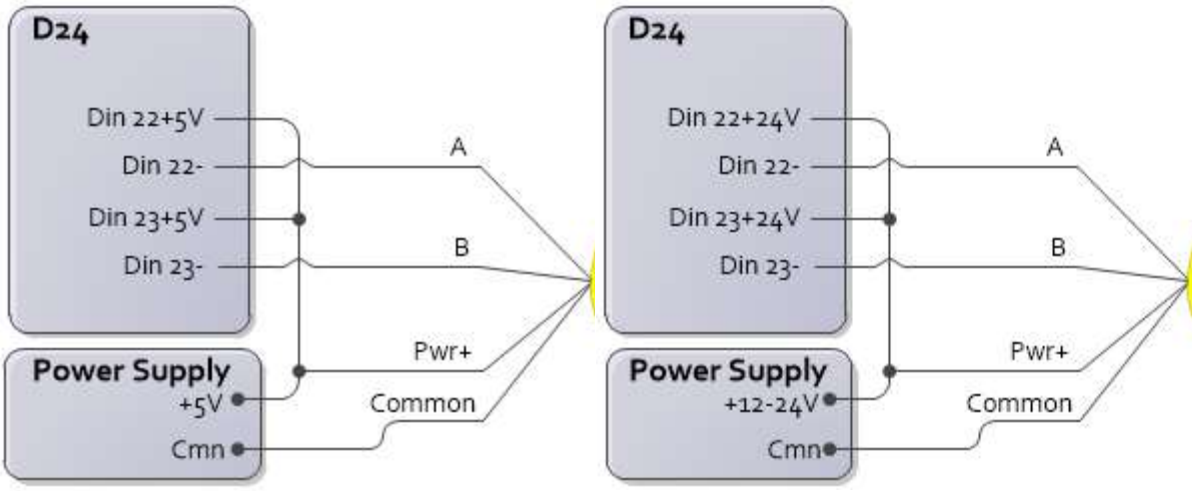
First quadrature input without Z (Index)
12-24V signal levels



Second quadrature input without Z (Index)
TTL or 5V signal levels



Second quadrature input without Z (Index)
12-24V signal levels



See Also

Wiring Guidelines | D24 Module

11.Troubleshooting

Show All

11.1. Troubleshooting Overview

[TODO] - add more to this topic

Tip:
USE THE EVENT LOG! If an error occurred, or something unexpected happens, open the Event Log to see if the command caused any errors.

How to Troubleshoot

RMCTools provides several tools that help you troubleshoot the RMC:

Tools	Description
Event Log	Logs most events that occur on the RMC, including commands, errors, and communications transactions. You can change the Event Log Filtering to log only certain events, or to add events, such as communication transactions.
Plots	Plot any register in the RMC. Track motion and see what happened at each point.
Status Bits	Provides an overview of the state of each axis.
Error Bits	Provides an overview of the errors that have occurred on the axis.

Tips

- Use the Event Log Monitor to view the commands you have issued and any errors they may have caused.
- Use the Event Log Filtering to log only certain events, or to add events, such as communication transactions.
- Add the Status Bits register and Error Bits register to the plot data.

Plots

I can't trigger a plot:

- Make sure the plot trigger is enabled. Use the Enable/Disable Plot Trigger (104) command.
- Make sure the plot is rearmed. Use the Rearm Plot (103) command.

RUN Mode

The RMC unexpectedly exits RUN mode:

After the RMC exits RUN mode, look at the Event Log. It will likely tell you why it exited RUN mode. Possible reasons are:

- A discrete output was set to put the RMC into and out of RUN mode. See the Programming Properties topic for details.
- The RMC received a Fault Controller (8) command.

Support

If you are unable to solve a problem, contact Delta's Technical Support.

See Also

Error Bits | Event Log Monitor | Technical Support

Show All

Note To Help Editor:

To add an Error code:

- **Add the text**
- **Add a bookmark next to it. Use the same format as the other bookmarks.**
- **In the HTML CODE, change the id of the dropdown text to "id = xerrCodeN", where N is the number of the error code. Look at the other IDs to make sure it is right.**
- **Add the error code name to the index, linking to newly created bookmark**

11.2. Error Codes

When any error occurs in the RMC, an error code is reported. The error codes, along with all controller events, are recorded in the Event Log. This log can be viewed in the Event Log Monitor in RMCTools.

Errors codes that occur on a specific axis will set the corresponding Error Bit on the axis. The most recent error code that occurred on the axis will be stored in the Last Error Number status register.

Error codes that do not occur on a specific axis will not set error bits on any axes.

Error Code Descriptions

Command Errors

These errors set the Command Error bit, if the error code occurred on a specific axis.

Note:
The Command Error bit is cleared when any valid command is issued.

No.	Name
1	Invalid command The command number itself is not supported by the axis it was issued to. Either that command number is reserved for future commands, or it is a defined command that is not supported by this axis type.
2	Command not implemented This error indicates that the command has not been implemented in the firmware. This error should never occur in production firmware. Please report this to Delta.
3	Invalid command parameter 0 The first command parameter had an invalid value. Refer to the commands documentation for details on valid ranges for each command parameter.
4	Invalid command parameter 1

	The second command parameter had an invalid value. Refer to the commands documentation for details on valid ranges for each command parameter.
5	Invalid command parameter 2 The third command parameter had an invalid value. Refer to the commands documentation for details on valid ranges for each command parameter.
6	Invalid command parameter 3 The fourth command parameter had an invalid value. Refer to the commands documentation for details on valid ranges for each command parameter.
7	Invalid command parameter 4 The fifth command parameter had an invalid value. Refer to the commands documentation for details on valid ranges for each command parameter.
8	Invalid time calculated This error occurs when the Speed At Position (36) command is issued with starting conditions and command parameters that lead to the RMC computing a negative time to make the move in. This indicates an impossible request. See the Speed at Position (36) topic for details on conditions and parameter values that cause this error.
9	Unable to clear Halt condition A motion command was issued, but the attempt to clear the error bits before processing the command was not successful. In order to issue a motion command, you must first clear all error conditions that cause a halt.
10	Unable to clear transducer errors A motion command that requires transducer feedback (e.g. closed loop moves) was issued, but transducer errors that cause a halt remained on the axis.
11	Axis is not enabled A motion command was issued, but the axis has not been enabled. Use the Enable Controller (7) or Enable/Disable Axis (97) commands to enable the axis.
12	Pressure/Force Limit not allowed in Direct Output A Set Pressure/Force Limit Mode (40) command was issued to enable Pressure/Force limit, but the axis was in Direct Output. The axis cannot be in Direct Output while limiting Pressure/Force.
13	Unspecified error trying to start at the requested program An internal error prevented the Start Task (90) command from executing correctly. Please report this to Delta.
14	User Programs are stopped or not loaded The Start Task command was not taken because either no User Programs were loaded, or the User Programs were not in Run mode.
15	Target pressure must be set before enabling pressure limit You must issue a command that sets the Target Pressure or Force before enabling pressure with the Set Pressure/Force Limit Mode (40) command. For example, the Ramp Pressure/Force (S-Curve) or Ramp Pressure/Force (Linear) commands will set the desired target pressure or force.
16	This axis cannot be enabled because of parameter inconsistencies The combination of Cycles, Starting Location, and Frequency in the command is invalid. See the Change Target Parameter (80) for details on valid combinations.

17	Requires axis to be set up as incremental SSI Set the axis to Incremental SSI to use this feature, or do not use this feature.
18	Command not supported by current control mode To use this command, change the control mode. See the Control Modes topic for details.
19	Invalid command parameter 5 The sixth command parameter had an invalid value. Refer to the commands documentation for details on valid ranges for each command parameter.
20	Invalid command parameter 6 The seventh command parameter had an invalid value. Refer to the commands documentation for details on valid ranges for each command parameter.
21	Invalid command parameter 7 The eighth command parameter had an invalid value. Refer to the commands documentation for details on valid ranges for each command parameter.
22	Invalid command parameter 8 The ninth command parameter had an invalid value. Refer to the commands documentation for details on valid ranges for each command parameter.
23	Command can only be issued when axis is stopped or geared This command can only be issued when the axis is stopped or geared. See the help topic for the command for details.
24	Master position already past Master Sync Position This position of the master was already past the Master Sync Position. See the help topic of the command for details.
25	This command is only supported by certain gear states This command is only supported by certain gear states. For example, the Phasing (34) and Geared Slave Offset (35) commands are only supported in incremental gear states. If the axis is gearing due to a Gear Absolute (25) command, the Phasing (34) and Geared Slave Offset (35) commands will not work.
26	Command requires that the axis be geared to a position This command requires that the axis is geared to a position.
27	Command not allowed while gear ratio is changing This command is not allowed while the gear ratio is changing.
28	Master Sync Position cannot equal the current position. The Master Sync Position cannot equal the current position. This error can also be caused by a zero Master Distance, which will result in a Master Sync Position that equals the current position. See the command for more details.
29	Command requires that the axis be running a sine move. The command can only be issued when a sine move is running.
30	Discontiguous target not allowed without selecting a transition behavior. The command that was issued requires either that the Target Position be at the starting point as defined by the command, or that a transition command be issued before the command. For curves based on time, both the current position and velocity must match the curves starting position and velocity. See the help topic for the respective command for details. Sine Start (72)

	Curve Start (86) Curve Start Advanced (88) Gear Absolute (25)
31	Unsupported combination of Cycles, Starting Location, and Frequency. Check the parameters and make sure they are correct.
32	The requested Curve ID is not valid. The requested Curve ID is not valid, because no curve with that ID exists. Use the Curve Add (82) command to add a curve with a specified ID.
33	The requested Curve ID does not support multiple cycles. The requested Curve ID does not support multiple cycles. If you want the curve to repeat, the Interpolation Options in the Curve Data must have the Endpoint Behavior set to Cyclic (+2) or Zero-Velocity (+0).
34	Curve Add failed because the curve queue is full. Curve Add failed because the curve queue is full. The curve queue contains the curves that are currently being added to the curve store. The curve queue can handle up to 16 curves simultaneously. To prevent this error, wait for a curve to be completely added before adding another curve.
35	Command is not supported by this RMC75S/RMC75P hardware revision. The hardware revision of this RMC75S or RMC75P does not support this command. You will need to obtain a newer RMC75S or RMC75P.
36	Cannot join a synchronized move in progress. This error occurs when you issued a synchronized move command with a group number that is already being used by another sync move in progress. You must use another group number.
37	Cannot do a synchronized stop on a non-synchronized axis. You issued a motion command with a Requested Position that was outside the Positive or Negative Overtravel Limits. The requested position was truncated at the limit or the current position, whichever is farther from the limits. Check the Positive and Negative Overtravel limits. **Note:** If you want the axis to halt instead of moving to a truncated position, make sure the Auto Stop on the Command Adjusted error bit is set to halt.
38	Cannot change the final position of a synchronized move in progress. When a synchronized move is in progress, a sync move command with the same group number can only be issued if the resulting final position will still be the same.
39	Cannot start a synchronized move when the axis is moving. An axis must be stopped when a sync move command is issued to it. In open loop, this means the Stopped Status bit must be on. In closed loop control, this means the Target Velocity must be zero.
40	Motion command ignored due to grouped halt. This command error appears whenever, within the same command block, a command in the same sync or halt group as the current axis had a command error.
41	Program cannot run on selected task This can occur either when a program has been set up to not run on this particular task, or if it has been set to run on only one task at a time and is already running on another task.

	Refer to the User Program Properties dialog for each individual program to view or change which tasks the user program is allowed to run on.
42	Position is too large relative to Position Unwind This will occur when a rotary axis is commanded with a position that is on the order of 16 million times the Position Unwind value. Beyond this point it is not possible to properly unwind the value due to the maximum dynamic range of floating point (REAL) values.
43	Command requires that the axis be following a curve move This error occurs if the Change Master (79) command is sent to an axis on which a curve move is not in progress (whether started with Curve Start (86) or Curve Start Advanced (88)). The Change Master (79) command should only be sent to an axis on which a curve move is in progress.
44	Command not supported in current Gain Set mode The Select Gain Set (75) command can only be sent when the Gain Sets parameter is set to **Dual**.
45	Command not allowed when Controller is Disabled The following commands are not allowed when the controller is in the Disabled state: • Enable/Disable Axis (97) • Set Discrete Output (60) • Clear Discrete Output (61) • Toggle Discrete Output (62)
46	Command not allowed when RUN/Disabled input is low The following commands are not allowed when the RUN/Disabled input is low: • Enable Controller (7) • RUN Mode (98) • PROGRAM Mode (99)
47	This axis cannot be enabled because the axis is not authorized The axis cannot be enabled because the number of Control Loops allowed by Feature Key has been exceeded.
48	This timer cannot be used because it is already armed This Event Timer cannot be used because it is already armed. If necessary, it can be disarmed with the Disarm Event Timer (106) command.
49	Unsupported combination of Trigger Type and Home Input The combination of the Trigger Type and Home Input command parameters for the Arm Home (50) command are not supported. In particular, the "Z and H" and "Z and Not H" trigger types cannot be used when the Home Input is set to the Z input for this axis.
50	The selected Home or Registration Input is configured as an output The selected Home or Registration input is configured as an output. It must be configured as an input to be used for homing or registration.
51	Command not allowed with invalid valve linearization curve selected for this axis If the Valve Linearization Type axis parameter is set to **Curve**, then the curve specified by the Valve Linearization Curve ID axis parameter must be a valid valve linearization curve in order for closed-loop commands to be issued to the axis. For details, see Valve Linearization.

Command Modified Errors

These errors set the Command Modified error bit, if the error code occurred on a specific axis.

No.	Name
81	Requested position truncated at limit A motion command was sent to the axis in which the requested position was beyond the positive or negative travel limit. The requested position was truncated to the positive or negative travel limit. If the Command Error Autostop is set to Status Only, the command will be processed normally with the truncated requested position. If the Command Error autostop is set to halt, then the axis will halt.
82	Requested pressure/force truncated at limit The Ramp Pressure/Force (41) command requested a pressure/force outside the Positive or Negative Limits. The requested pressure/force was truncated at the limit or the current pressure/force, whichever is farther from the limits. Check the Overtravel limits. **Note:** If you want the axis to halt instead of moving to a truncated pressure or force, make sure the Auto Stop on the Command Adjusted error bit is set to halt.
83	Master already in clutch area. Master Start Distance truncated The gear master is already in clutch area. The Master Start Distance was truncated.

Configuration Errors

These errors set the Configuration Error bit, if the error code occurred on a specific axis.

Note:
The Configuration Error bit is cleared when any valid parameter write occurs.

No.	Name
101	Value out of range The value written was out of range for that register. The write was ignored.
102	Non-zero write to a reserved register A reserved register was written to with a non-zero value. This has no affect, but because it was unlikely that this was intentional, we record this as an error.
103	Invalid data field A bit-field in a register was incorrect. This applies to registers that are made up of several bit fields. One or more of its components were incorrect.
104	Write to a read-only register A value was written to a read-only register. The write was ignored.
105	Write to an un-implemented register A value was written to a register that does not currently support writes, but should support writes. The write is ignored. This error should never occur in production firmware. Please contact Delta technical support if this occurs.
106	The value was truncated to fit in range The value written was out of range for that register. The value was truncated to fit into range and then used. Compare with error 101.
107	The value was rounded to the nearest acceptable value The value was not an acceptable value, but was within range. It was rounded to the nearest acceptable value and then used. For example, the

	sample periods on a plot must be a whole number of control loop times. In this case, the value will be rounded to the nearest whole number of loop times.
108	One or more Auto Stop settings were invalid One or more of the Auto Stop settings in this register were out of range. This means that either a reserved value was used or a Auto Stop setting that is not allowed for a particular error. For example, status only cannot be selected for No Transducer.
109	This value cannot be changed while in Run mode This error indicates that an attempt was made to change the number of available Tasks or enable/disable the Program Triggers while the controller is in RUN mode. These registers can currently only be set through RMCTools, which will automatically put the controller temporarily in STOP mode.
110	This value can only be changed in Direct Output or with the axis disabled To prevent dangerous motion, certain parameters can only be changed when the axis is disabled or in Direct Output. Use the Enable/Disable Axis (97) or Direct Output (9) command before attempting to change the parameter. If you disable the axis, remember to enable it after changing the parameter. When changing parameters from RMCTools, the disabling and enabling is automatically.
111	Count Unwind is invalid for Absolute Rotary SSI On axes configured as Rotary Absolute, the Count Unwind parameter must be a power of 2, such as 1024, 8192, etc.
112	An Indirect Data Map entry cannot point to Indirect Data Change the Indirect Data Map entry so that it does not contain any Indirect Data Map address. On the RMC75, these are %MD18 addresses. On the RMC150, these are %MD42 addresses. On the RMC200, these are %MD8 addresses.
113	The IP settings are locked and can only be changed via USB The IP address can be locked so that it cannot be changed over Ethernet. This prevents accidentally changing the address when connected remotely. To remove the lock, go to the Ethernet Settings Page. In the **Additional Options** section, uncheck the **Lock IP settings** check box.
114	Non-zero value written to an unconfigured Indirect Data Map entry An attempt was made to write to an Indirect Data Map entry that has not been configured. Make sure to write only to Indirect Data Map entries that have been configured.
115	The selected SSI features require a version 6 or newer SSI module This occurs on the RMC150 only when the SSI Clock Rate is set to 921 kHz or Wire Break Detection is disabled, and the SSI module version is less than 6.
116	Both axes using this resolver module must be in Direct Output or disabled When changing the Reference Amplitude or Reference Frequency, both axes of the module must be disabled or be in direct output. Changing these parameters on one axis will affect both axes.
117	Prs/Frc Orientation cannot be changed while Prs/Frc Control or Limit is enabled The Pressure/Force Orientation axis parameter cannot be changed while the axis in Pressure/Force Control or while Pressure/Force Limit is enabled.
119	Enable Output already used by another axis

	Any Enable Output can only be used by one axis. This error indicates that more than one axis are set to use the same Enable Output.
120	This value cannot be changed when Controller is in disabled state The following registers cannot be changed when the RMC is in the Disabled state: • Discrete I/O Register: Outputs for Slot *n* • Discrete I/O Register: Force Off for Outputs in Slot *n* • Discrete I/O Register: Force On for Outputs in Slot *n*

Runtime Errors

These errors set the Runtime Error bit, if the error code occurred on a specific axis.

No.	Name
151	Incorrect module detected The module in the given slot does not match the module that is expected in that slot.
152	Unsupported module detected A module was detected in the controller backplane that is not supported by this version of firmware. Consider updating to the latest CPU firmware or contacting Delta technical support.
153	Active I/O module removed An I/O module was removed from the backplane.
154	Axis is missing a required I/O module An I/O module that is part of an axis is missing from the backplane. This will be reported on startup or after a module is removed.
155	Unexpected failure saving to flash An error occurred while updating the flash memory. The operation was not successful.
156	Task exception register contents This error lists the register contents when a task exception occurred. This indicates a problem with the controller firmware and should be reported to Delta technical support.
157	Error saving controller image to the SD card An error occurred while saving the controller image to the SD card. The image was not saved.
158	Error restoring controller image from the SD card An error occurred while restoring the controller image from the SD card. The image was not restored to the controller.
159	Valve linearization curve is no longer valid If the Valve Linearization Type axis parameter is set to **Curve**, then the curve specified by the Valve Linearization Curve ID axis parameter must be a valid valve linearization curve in order for the axis to operate in closed-loop control. For details, see Valve Linearization.
160	Maximum loop time carryover exceeded The Loop Time of the RMC is the specified interval at which it updates all the inputs, calculates the control algorithms, and updates the outputs. This error indicates that the amount of time required to process all these calculations exceeded the set loop time by more than 80%. If the loop time is exceeded by a lesser amount, the RMC normally makes up that time in

	the following loop times, and there is no impact on control. However, if a loop time is exceeded by 100%, then that loop time is lost, delaying by 1 loop time the RMC's calculations, for example target generation. If this error occurs, you should take one or more of the following actions: - Reduce the amount of expressions calculations the user programs - Reduce the number of axes - Reduce the number of simultaneous motion commands - Reduce the amount of communications. - Select a larger loop time For more details, see Loop Time.
161	Output DMA is unexpectedly busy. Outputs not written this control loop. This is an unexpected condition that prevented the outputs from being written to the I/O modules for the indicated control loop. Please report this unexpected condition to Delta technical support.
162	Real time clock has an internal fault. The Real Time Clock component on the RMC200 CPU is not behaving as expected and cannot be used. See the RMC200 Real Time Clock topic for details on this condition.
163	Real time clock battery is low. The real time clock on the RMC200 CPU reports a low battery condition. See the RMC200 Real Time Clock topic for details on this condition.
201	Command block dropped A command or set of commands that was issued to the controller was dropped because too many sets were received simultaneously. The RMC75 allows up to 6 command sets (up to one command per axis in each set) to be queued up before losing a command set. The RMC150 allows up to 12 command sets (up to one command per axis in each set) to be queued up before losing a command set. One command set is processed (and thus, removed from the queue) per control loop. It is possible, although unlikely, to overflow the queue by simultaneously issuing commands from multiple sources for consecutive control loops.
202	Flash update failed An attempt to save the RMC settings to Flash failed. This indicates a failure of the RMC hardware. Please contact Delta technical support if this occurs.
203	Unable to initialize the PROFIBUS sub-system The PROFIBUS sub-system of the hardware could not be started. This indicates a failure of the RMC hardware. Please contact Delta technical support if this occurs.
204	Target Position went out of limits The Target Position of an axis went out of limits. Notice that this does not occur because the axis is commanded to go outside of limits, as that is caught by Command Adjusted error 81, but it indicates that a move was given a low deceleration ramp that caused it to move outside the limits.
205	Unsupported Flash command received This is an internal error. Please contact Delta technical support if this occurs.
206	Save of IP Setting to Flash failed The save to Flash failed.
207	Command overwritten within a Task step

	A command was overwritten in a user program. Check to make sure the step does not issue more than one command to an axis.
208	Drive disabled due to Watchdog timeout This indicates an internal error occurred that caused the control outputs to be disabled. Please contact Delta technical support if this occurs.
209	Commands overwritten in the PROFIBUS command area Commands were left in the PROFIBUS command area when either the first deferred command or a singular command was issued. All commands that had been in the command area are discarded without further processing.
210	Command overwritten in the PROFIBUS command area A command was issued over the top of a deferred command in the PROFIBUS command area. The deferred command is discarded without further processing.
211	PROFIBUS read/write length invalid The length field in the PROFIBUS Enhanced Data Channel had an invalid value.
212	No follow-up command after Adv. Gear Move command No follow-up command was issued after the Advanced Gear Move command completed. See the Advanced Gear Move (33) for details.
213	PROFIBUS command received but no axes were selected. Commands must always be issued to an axis, even if they aren't specific to any axis. Make sure to set the axis select bits.
214	Simulate Mode Model Invalid. The simulator parameter settings resulted in an invalid simulator model. The simulator will not function without a valid model. See the Simulating Motion topic for details.
215	Model-based Filter Model Invalid, The feedback model settings resulted in an invalid model. The feedback model will not function without a valid model. See the modeling topic for details.
216	Internal Target Generator fault An internal target generator fault occurred. This error will always halt the axis, regardless of the Auto Stop setting, as is required because the current target generator cannot be used when faulted.
217	Count Unwind is invalid for Absolute Rotary SSI. On axes configured as Rotary Absolute, the Count Unwind parameter must be a power of 2, such as 1024, 8192, etc.
218	No follow-up command after Adv. Time Move command. The Advanced Time Move Absolute (26) and Advanced Time Move Relative (27) commands require that another command be issued to the axis within 10 ms after the Done bit turns on, or the axis will halt.
219	Invalid time calculated by Speed at Position command. The Speed at Position (36) calculated an invalid time. This is typically caused by commanding the axis to move to a position that requires it to change direction of motion. The Speed at Position (36) does not support changing directions.
220	Axis Home Failed. Encoder A or B wire break when triggered. The Homing failed due to a detected A Wire Break or B Wire Break. Check the wiring.

221	Positive Limit Input's I/O point is configured as an Output. The input specified for the Positive Limit Input has been configured as an output and will not be usable as a limit input. Use the Discrete I/O Configuration dialog to configure the I-O point as an input.
222	Negative Limit Input's I/O point is configured as an Output. The input specified for the Negative Limit Input has been configured as an output and will not be usable as a limit input. Use the Discrete I/O Configuration dialog to configure the I-O point as an input.
223	Axis Home Failed. Index (Z) wire break. The Homing failed due to a detected Index (Z) Wire Break. Check the wiring.
224	Control Fault. I-PD requires a non-zero Integral Gain. The Position I-PD and Velocity I-PD require a non-zero Integral Gain. Set the Integral Gain to some non-zero value to use the I-PD control modes.
225	Control Fault. Control algorithm is not implemented. Please contact Delta technical support. An incorrect control mode was selected. Try a different mode. The position and velocity control modes can be set via the Default Pos/Vel Control Mode parameter and the Set Pos/Vel Ctrl Mode (68) command.
226	Master Sync Pos cannot equal the current position The Master Sync Position cannot equal the current position. This error can also be caused by a zero Master Distance, which will result in a Master Sync Position that equals the current position. See the command for more details.
227	Task Fault on Task *n* This will occur if any of the following occur: an array index is out of range, a command is overwritten (which can occur if a step has commands with both explicitly-specified commanded axes and default commanded axis), or if an invalid Word Code is encountered (internal error).
228	Curve Add failed Curve Add failed. This can be due to several reasons, such as the Data Format is not correct, or the Curve Store is full.
229	No follow-up command after non-terminated Curve profile This error occurs when a curve has completed, but the ending velocity is non-zero. See the Curve Start (86) command for more details.
230	Unable to auto-delete curve due to a full curve processing queue The curve processing task is too busy to auto-delete the curve. You may need to manually delete the curve.
231	Unable to allocate memory from Curve Store or unable to allocate buffer for assembling curve data The curve store was too full to allocate a new curve or buffer. Delete unused curves to make room for the curve being added.
232	Master Register out of range specified by Gear Absolute command The Gear Absolute (25) command specified the range which the master must be in, but the master was not in this range. Either move the master into the range and re-issue the Gear Absolute command, or choose a different Endpoint Behavior for the Gear Absolute command.
233	Master Register out of range specified by Curve The Curve Start Advanced (88) command specified the range which the master must be in, but the master was not in this range. Either move the master into the range and re-issue the Curve Start Advanced command, or

	choose a different Endpoint Behavior for the Curve Start Advanced command.
234	Failure to save retentive variables to NVRAM A failure occurred while saving retentive variables to non-volatile memory.
235	Not all variables marked retentive can fit in the NVRAM Too many variables are set to Retain, and do not fit in the non-volatile memory. To fix this problem, use the Variable Table Editor to reduce the number of retentive variables. See the Variables topic for more details.
236	Loop Time Exceeded This error will occur each time the motion loop exceeds the allotted time. This is a serious error and may adversely affect the motion control. If this error occurs, either increase the loop time or contact Delta technical support for help on how to keep from exceeding the loop time.
237	Flash image too large This error will occur if the Flash update fails because the image is too large. If this problem occurs, decrease the user programming size. For the RMC75E, make sure the firmware is 3.32.0 or newer, as indicated in Program Capacity and Time Usage.
238	Image Area is busy. Unable to process new command The **Image Area Command** was written to while the Image Area was busy building or applying an image. The new command was ignored. Monitor the **Image Area State** register to ensure that the Image Area is not busy before issuing your command.
239	Image Area: Invalid command received The value written to the **Image Area Command** register is not supported. Verify the value written to this register.
240	Image Area: Image downloaded out of sequence The image was downloaded improperly. This can happen for the following reasons: • The **Current Index** register, if written at the start of each write, may have the wrong value. • The **Image Data** registers were not written to in sequential order. That is, each consecutive write to the **Image Data** registers must either start at the first **Image Data** register, or must continue where the last write completed. • An individual write extended beyond the end of the **Image Area** file (4096 total registers). • The total re-assembled downloaded image exceeds 65,536 registers. To recover from this error, reset the download by writing a zero to the **Current Index** register or by writing **Reset Image Area (3)** to the **Image Area Command** register, and then resend the image starting at the beginning.
241	Image Area: Image uploaded out of sequence The image was uploaded improperly. This can happen for the following reasons: • The **Image Data** registers were not read in sequential order. That is, each consecutive read from the **Image Data** registers must either start at the first **Image Data** register, or must continue where the last read completed.

	• An individual read extended beyond the end of the **Image Area** file (4096 total registers). • More than 65,536 total registers have been uploaded. To recover from this error, write a zero to the **Current Index** register and re-upload the image starting at the beginning.
242	Image Area: Invalid Current Index value written A non-zero value was written to the **Current Index** register in the **Image Area**. This is only allowed when downloading an image. This register should not be written to while uploading or at other times.
243	Image Area: Download not allowed in this state The **Image Data** registers cannot be written to while the Image Area is in this state. Before downloading an image to the Image Area, you should reset the image area by writing **Reset Image Area (3)** to the **Image Area Command** register. This error can also occur if writes are made after an Image Area: Image downloaded out of sequence error has occurred.
244	Image Area: Unable to build upload image This error will occur if the entire image in the controller is larger than the maximum size of 65,536 registers and therefore could not be created.
245	Image Area: Unable to apply downloaded image The controller was unable to apply the downloaded image. The state of the controller has not been changed. There are several reasons why the RMC may be unable to apply a downloaded image. The reason will be listed in the event log entry: • **Invalid Image Format** The image downloaded appears to be corrupt or built by a later firmware edition. Verify that the image was downloaded properly and has not been corrupted in the PLC. • **Restart Required to Apply this Image** The **Do Not Restart Controller (+16)** option was used, but the image would change the axis definitions or loop time, and therefore requires restarting the controller in order to apply. Apply the image without using the **Do Not Restart Controller (+16)** option in order to change these settings. • **Failure writing to Flash** A hardware error occurred while trying to save the image to flash memory. Please contact Delta technical support to report this problem. • **Must be in PROGRAM mode to apply without restart** The **Do Not Restart Controller (+16)** option requires that the controller be in PROGRAM mode before applying changes. Set the controller to PROGRAM mode and re-apply the downloaded image. • **Image supports a different hardware configuration** The downloaded image was built for a different hardware configuration. The controller must have the same modules installed, including expansion modules (on the RMC75). The hardware revisions need not match.
246	Program Error This is an internal error. Please contact Delta technical support if this occurs.
247	Program Error This is an internal error. Please contact Delta technical support if this occurs.

248	Curve master is moving too fast relative to the curve length When following a curve, the master register cannot move by more than one curve length in a single control loop. For example, if a curve has X values ranging from 0 to 10.0, then the master register cannot move by more than 10.0 in a single control loop.
249	EtherNet/IP I/O production(s) missed This occurs when the RMC controller missed one or more I/O scheduled productions over EtherNet/IP or PROFINET. This can occur if the Ethernet system is very busy.
250	Image Area: Copy protection is enabled. Unable to build upload image This occurs when the user tries to build an upload image using the image upload/download area when the programming has copy protection enabled. Allowing the user to upload the image would violate the copy protection.
251	Position is too large relative to Position Unwind
252	Unable to return to RUN mode Unable to return to RUN mode because the RUN/Disabled input is low.
253	Unable to enable axis because axis is not authorized Unable to enable the axis because the number of Control Loops allowed by Feature Key has been exceeded.
254	System task exception detected A system task exception occurred. This exception indicates a problem with the controller firmware and should be reported to Delta technical support.
255	Task exception memory contents Lists the memory contents related to the system task exception. This exception indicates a problem with the controller firmware and should be reported to Delta technical support.

Startup Errors

These errors occur when the RMC powers up, or after it restarts.

No.	Name
1	Unable to create axis. Unsupported axis type. The Axis Type field in an axis definition was invalid. Verify that you are using the most recent version of RMCTools and firmware and re-download the project to the controller. See Axis Definition Registers for more information.
2	Unable to create axis. Unsupported input type. The specified Input Type field in an axis definition was invalid. Verify that you are using the most recent version of RMCTools and firmware and re-download the project to the controller. See Axis Definition Registers for more information.
3	Unable to create axis. Unsupported output type. The specified Output Type field in an axis definition was invalid. Verify that you are using the most recent version of RMCTools and firmware and re-download the project to the controller. See Axis Definition Registers for more information.
4	Unable to create axis. Unsupported transducer type. The specified Transducer Type field in an axis definition was invalid. Verify that you are using the most recent version of RMCTools and firmware and re-download the project to the controller. See Axis Definition Registers for more information.
5	Unsupported axis module.

	The specified axis module is not supported. Possible reasons include: - The hardware does not support that module type. - The firmware does not support that module type. - There is an error in the module so that its type has been corrupted.
6	Out of memory. This error should not occur. Contact Delta for assistance.
7	Unable to create axis. Invalid axis definition. An axis definition was invalid. Verify that you are using the most recent version of RMCTools and firmware and re-download the project to the controller. See Axis Definition Registers for more information.
8	Unable to create axis. Unsupported axis simulator type. This error should not occur. Contact Delta for assistance.
9	Unable to create axis. Unsupported axis Target Manager. This error should not occur. Contact Delta for assistance.
10	Unable to create axis. Unsupported axis Control Manager. This error should not occur. Contact Delta for assistance.
11	Unable to load User Programs from Flash. This error should not occur. Contact Delta for assistance.
12	Unrecognized serial protocol selected on Port 0. The protocol selected for port 0 on the RMC75S is unrecognized. This error should not occur. Contact Delta for assistance.
13	Unrecognized serial protocol selected on Port 1. The selected serial protocol selected for port 1 on the RMC75S is unrecognized. See RMC75 Register Map - File 21 Comm Configuration for details on the registers that specify the serial communication.
14	Unexpected error loading files from Flash. This error should not occur. Contact Delta for assistance.
15	Beta User Program Image saved in Flash was discarded. Old, unsupported user programs were discarded. These user programs were never available in official released product.
16	Unsupported module. The specified module is not supported. Possible reasons include: - The hardware does not support that module type. - The firmware does not support that module type. - There is an error in the module so that its type has been corrupted.
17	Unable to create all axes. Reverting to defaults. Because all axes were not able to be created, the axis definitions were set to the default settings.
18	Cancelled startup in RUN mode because Axis Defs were reverted to defaults. The controller has been set to start in Run mode. However, because the axis definitions were set to default values, the controller was not put into Run mode after startup.
19	No MAC address has been assigned. Ethernet cannot be started. This error should not occur. Contact Delta for assistance.
20	Internal TCP/IP stack startup error. This error should not occur. Contact Delta for assistance.

21	Internal Ethernet Driver startup error. This error should not occur. Contact Delta for assistance.
22	Internal TCP/IP interface startup error. This error should not occur. Contact Delta for assistance.
23	FPGA revision mismatch. This error should not occur. Contact Delta for assistance.
24	The RMC150 requires Quad modules to be version 7 or higher. The quadrature (-Q) module in the specified slot has a version 6 or older. The RMC150 only supports quadrature (-Q) modules version 7 or higher.
25	The requested loop time is not supported by this controller. The requested loop time is not supported by this controller. See the Loop Time topic for details.
26	Unable to start the Omron/FINS ethernet service. This error should not occur. Contact Delta for assistance.
27	Unable to start the DF1 over Ethernet (CSP) ethernet service. This error should not occur. Contact Delta for assistance.
28	Unable to start the EtherNet/IP ethernet service. This error should not occur. Contact Delta for assistance.
29	Unable to start the DMCP ethernet service. This error should not occur. Contact Delta for assistance.
30	Unable to start the Mitsubishi ethernet service. This error should not occur. Contact Delta for assistance.
31	Unable to start the Modbus/TCP ethernet service. This error should not occur. Contact Delta for assistance.
32	Unable to create axis. Out of memory. This error should not occur. Contact Delta for assistance.
33	Unable to create axis. Unable to locate feedback interface. The axis definition refers to an input channel that is not currently installed in the controller. Verify that the hardware configuration of the RMC matches the configuration in the project file.
34	Unable to create axis. Unable to locate output interface. The axis definition refers to an output channel that is not currently installed in the controller. Verify that the hardware configuration of the RMC matches the configuration in the project file.
35	Unable to create axis. Feedback channel not found. The axis definition refers to an input channel that is not currently installed in the controller. Verify that the hardware configuration of the RMC matches the configuration in the project file.
36	Unable to create axis. Control output channel not found. The axis definition refers to an output channel that is not currently installed in the controller. Verify that the hardware configuration of the RMC matches the configuration in the project file.
37	The following axis could not be created. The specified axis could not be created. Refer to other startup errors that were reported for information on the reason. Verify that the controller hardware configuration has not changed, and re-download the axis definitions to the controller.

38	Non-volatile RAM hardware error. There was an error with the non-volatile RAM hardware on the controller. Contact Delta for assistance.
39	Error encountered loading Retained variables. An error occurred when loading the Retained variables from non-volatile memory. The variables may not have been loaded correctly.
40	Saved current value for non-retentive variable ignored. The specified variable is not configured as a retained variable, but had a retained value stored for it. The retained value is ignored and the variable will instead assume its Initial Value. This can occur if the variable table setup was not saved to flash.
41	Upgrades to one or more modules are available to improve performance at the 4ms loop time. For the 4 ms loop time on the RMC150 controller, some I/O modules could be upgraded for better performance. Recommended module versions are Analog (-A, -H, -G) rev 7 or newer and SSI (-S) rev 7 or newer.
42	The UI/O channel is not configured for use by this axis. UI/O channels must be configured for Quadrature or SSI feedback before being used as that feedback type by an axis. See Configuring UI/O High-Speed Channels.
43	Unable to load field FPGA image into UI/O module. There was an error loading the FPGA image in the UI/O module. Contact Delta for assistance.
44	Unexpected error loading curves from flash. Unexpected error loading curves from flash. Contact Delta for assistance.
45	Unable to start the PROFINET service. This error should not occur. Contact Delta for assistance.
46	Controller restarted due to an internal Watchdog Timeout. This error should not occur. Contact Delta for assistance.
47	Controller restarted due to an internal Unhandled Exception. This error should not occur. Contact Delta for assistance.
48	Unable to load some settings from flash. There was an error loading some settings from flash memory. This can occur when downgrading firmware to an older version. Otherwise contact Delta for assistance.
49	Stopped creating axes due to prior failures. Due to prior errors with axis definitions, the firmware stopped creating the remaining axes. Contact Delta for assistance.
50	Error loading File Image from memory. There was an error loading some settings from flash memory. This can occur when downgrading firmware to an older version. Otherwise contact Delta for assistance.
51	Error loading memory region from kernel. This error should not occur. Contact Delta for assistance.
52	Controller restarted due to an internal System Fault. This error should not occur. Contact Delta for assistance.
53	Module could not be identified. The specified module could not be identified. Contact Delta for assistance.

54	Controller restarted due to an internal CPU power supply fault. The controller restarted due to an internal CPU power supply fault. Contact Delta for assistance.
55	No valid feature key was detected. Either no Feature Key was detected or the Feature Key contents could not be loaded. Possible reasons: - No feature key is installed. - The feature key has not been properly set up by Delta. - The RMC CPU or Feature Key has a hardware problem. Verify that a Feature Key is installed in the back of the CPU. Otherwise contact Delta for assistance.
56	Module is missing Soft ID information. The internal module identification (Soft ID) could not be read from the specified module. Contact Delta for assistance.
57	CPU is missing FPGA firmware. The CPU is missing FPGA firmware. Contact Delta for assistance.
58	Axis is not authorized and cannot be enabled. The available Control Loops on the Feature Key has been exceeded and the specified axis cannot be enabled.
59	One or more axes is not authorized by Feature Key. The total number control loops required by all defined axes exceeds the number of control loops installed on the Feature Key. One or more axes is not authorized and cannot be enabled.
60	I/O module missing on startup. An expected I/O module was missing from the controller on startup.
61	Controller Watchdog is disabled. This should only occur on development modules. Contact Delta for assistance.
62	Axis resource is missing from controller configuration. An expected module used by the specified axis is missing from the controller.
63	Too many physical control axes defined. The number of defined physical control axes exceeds the limits for the controller. One or more axes is not authorized and cannot be enabled. See Axis Types Overview for a list of maximum numbers of axes for each controller.
64	Axis is not authorized and cannot be enabled. The maximum number of physical control axes for this controller has been exceeded, so the specified axis cannot be enabled.

See Also

Troubleshooting Overview | Error Bits

11.3. Technical Support

Delta Technical Support Contact Information

Phone: +1-360-254-8688 (24-hour emergency support available)

Fax: +1-360-254-5435

Email: support@deltamotion.com

Website: https://deltamotion.com

RMC Return for Repair

If you need to return the RMC for repair, contact Delta prior to shipment for an RMA number. Returned RMCs must be packaged in static protection material and have the RMA number clearly marked on the outside of the package.

Please include a note explaining the problem!

Send the controller to:

Delta Computer Systems, Inc.

1818 SE 17th Street

Battle Ground, WA 98604

See Also

Troubleshooting Overview

12.Index

F

T

U