Changes for the Better

MITSUBISHI ELECTRIC

Using RMC75E and RMC150E Hydraulic Motion Controllers with iQ and L CPUs' Built-In Ethernet

V1.10

DELTA COMPUTER SYSTEMS

This book applies to the Mitsubishi Electric Corporation product components and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

MITSUBISHI ELECTRIC CORPORATION PROVIDES THIS BOOK "AS IS," WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

This book could contain technical inaccuracies or typographical errors. Changes are made periodically to the information herein. Mitsubishi Electric Corporation may make improvements and changes at any time to the product(s) and/or program(s) described in this book.

Contents

FURTHER READING REFERENCE LIST

Mitsubishi – Further Reading Reference List

QnUCPU User’s Manual Communication via Built-in Ethernet Port SH(NA)080811ENG

MELSEC L CPU Module User’s Manual (Built-In Ethernet Function) SH(NA)080891ENG

Delta Computer Systems, Inc.– Further Reading Reference List

RMCTools User Manual – DMCP Protocol (http://www.deltamotion.com/pdf/rmctools.pdf)

RMC70 Startup Guide

RMC150 Startup Guide

ATTACHMENTS

Mitsubishi - Attachments

RMC_FBs_Simple_UDP_MEAU.gxw - GX Works2 File

RMC_FBs_Structured_TCP_MEAU.gxw - GX Works2 File

RMC_Ethernet_FB.qxw – GX Works2 Function Block Library File (Simple Project)

RMC_FB_Lib.sul – GX Works2 Function Block Library File (Structured Project)

Delta Computer Systems- Attachments

RMC75MitsuDemo.rmcproj – RMC75E File

Chapter 1

1 Introduction

This document describes how to set up a Mitsubishi iQ or L CPU with a built-in Ethernet port to communicate with an RMC75E or RMC150E hydraulic motion controller from Delta Computer Systems, Inc. Delta has created function blocks to make this process easy.

The iQ and L CPUs support up to 16 Ethernet connections and can therefore communicate with up to 16 RMCs, although typically, some of these connections will be needed for communications with other devices.

Other Mitsubishi modules, such as the QJ71E71-100 and QJ71MT91, also support Ethernet communication with the RMC motion controllers. However, Delta recommends using the CPU's built-in Ethernet port, along with the provided RMC_Read and RMC_Write function blocks, because this requires much less effort.

Requirements

The RMC and PLC must be connected via an Ethernet network. The RMC can communicate with iQ "Universal" CPUs or LCPUs, with built-in Ethernet. The Serial Number of the iQ "Universal" CPU hardware must start with 11011 or later.

The PLC Programming software must be GX Works2 V1.31H or greater for iQ CPUs or V1.53F or greater for LCPUs. Melsoft Navigator is not required.

It is assumed the user has basic knowledge of the RMC motion controllers and RMCTools software.

1.1 Simple and Structured Projects

This document includes instructions for a simple project with labels, and for a structured project with structured ladder/FBD. Read the corresponding section for your project. The setup procedure in the RMC setup is identical for either method.

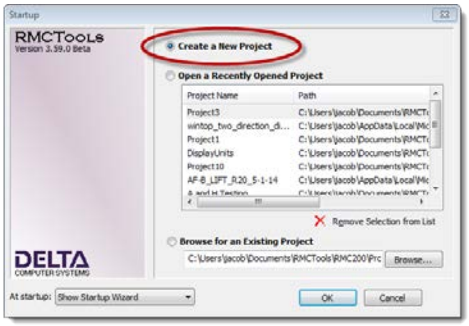
Chapter 2

2 RMC Configuration

You must set up the RMC’s IP address and Indirect Data Map.

2.1 Connecting to an RMC with RMCTools

1. Install the RMCTools software. RMCTools can be downloaded from www.deltamotion.com.
2. Connect a USB cable from your computer to the RMC.
3. Start RMCTools.
4. In the StartUp dialog, choose **Create a New Project** and click **OK**.



Startup

RMCTools
Version 3.59.0 Beta

● Create a New Project

○ Open a Recently Opened Project

Project Name	Path
Project3	C:\Users\jacob\Documents\RMCT
wintop_two_direction_di...	C:\Users\jacob\AppData\Local\Mic
Project1	C:\Users\jacob\Documents\RMCT
DisplayUnits	C:\Users\jacob\Documents\RMCT
Project10	C:\Users\jacob\Documents\RMCT
AF-8_LIFT_R20_5-1-14	C:\Users\jacob\AppData\Local\Mic
A and H Testing	C:\Users\jacob\Documents\RMCT

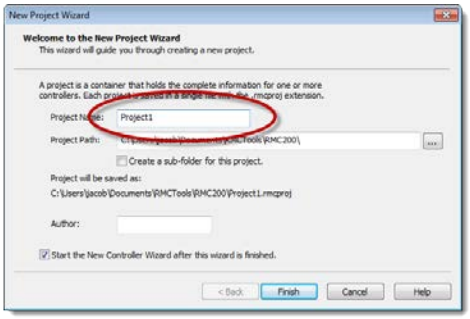
○ Browse for an Existing Project

C:\Users\jacob\Documents\RMCTools\RMC200\Prc

At startup: Show Startup Wizard

5. Enter the **Project Name** and click **Finish**.

   The New Controller Wizard will open.



New Project Wizard

Welcome to the New Project Wizard
This wizard will guide you through creating a new project.

A project is a container that holds the complete information for one or more controllers. Each project is saved in a single file with the .rmcproj extension.

Project Name: Project1

Project Path: C:\Users\jacob\Documents\RMCTools\RMC200\

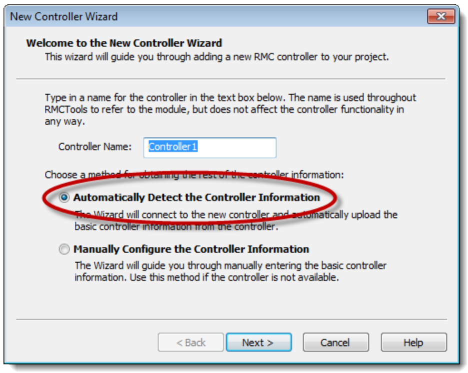
□ Create a sub-folder for this project.

Project will be saved as:
C:\Users\jacob\Documents\RMCTools\RMC200\Project1.rmcproj

Author:

☑ Start the New Controller Wizard after this wizard is finished.

6. Choose **Automatically Detect the Controller Information** and click **Next**.



New Controller Wizard

Welcome to the New Controller Wizard
This wizard will guide you through adding a new RMC controller to your project.

Type in a name for the controller in the text box below. The name is used throughout RMCTools to refer to the module, but does not affect the controller functionality in any way.

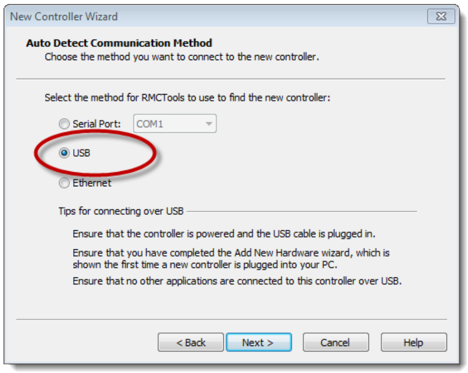
Controller Name: Controller1

Choose a method for obtaining the rest of the controller information:

● **Automatically Detect the Controller Information**
The Wizard will connect to the new controller and automatically upload the basic controller information from the controller.

○ **Manually Configure the Controller Information**
The Wizard will guide you through manually entering the basic controller information. Use this method if the controller is not available.

7. Choose **USB** and click **Next**.



New Controller Wizard

Auto Detect Communication Method
Choose the method you want to connect to the new controller.

Select the method for RMCTools to use to find the new controller:

○ Serial Port: COM1

● USB

○ Ethernet

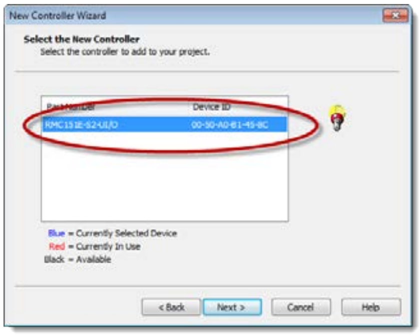
Tips for connecting over USB

Ensure that the controller is powered and the USB cable is plugged in.

Ensure that you have completed the Add New Hardware wizard, which is shown the first time a new controller is plugged into your PC.

Ensure that no other applications are connected to this controller over USB.

8. Choose your RMC from the list and click **Next**.



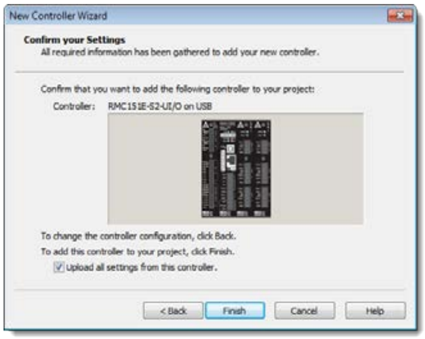
New Controller Wizard

Select the New Controller
Select the controller to add to your project.

Part Number	Device ID
RMC151E-S2-UI/O	00-50-A0-81-45-8C

Blue = Currently Selected Device
Red = Currently In Use
Black = Available

9. Verify the controller image matches yours, and click **Finish**. RMCTools will connect to the controller and upload all the setting from the controller.



New Controller Wizard

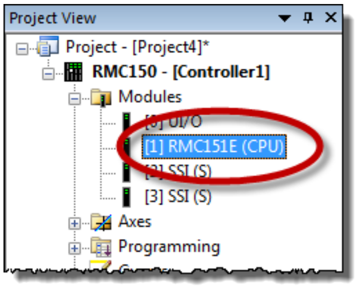
Confirm your Settings
All required information has been gathered to add your new controller.

Confirm that you want to add the following controller to your project:
Controller: RMC151E-S2-UI/O on USB

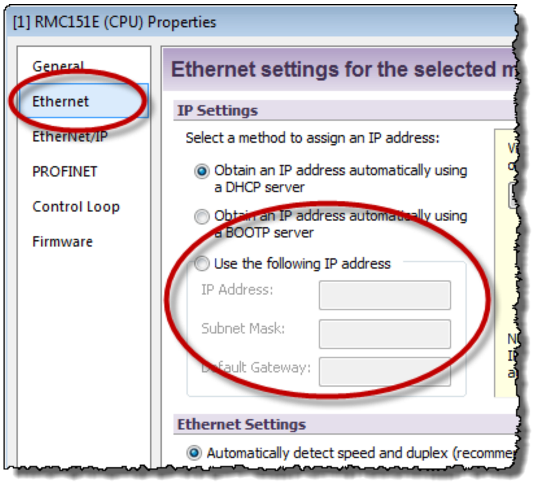
To change the controller configuration, click Back.
To add this controller to your project, click Finish.
☑ Upload all settings from this controller.

2.2 Configuring the RMC's IP Address

1. In the RMCTools Project pane, expand the **Modules** folder and double-click the RMC CPU.



2. On the **Ethernet** page, choose **Use the Following IP Address**, and set the **IP Address** and **Subnet Mask** to values that are compatible with your PLC and network.



[1] RMC151E (CPU) Properties

Ethernet settings for the selected m

IP Settings

Select a method to assign an IP address:

● Obtain an IP address automatically using a DHCP server

○ Obtain an IP address automatically using a BOOTP server

○ Use the following IP address

IP Address:

Subnet Mask:

Default Gateway:

Ethernet Settings

● Automatically detect speed and duplex (recomme

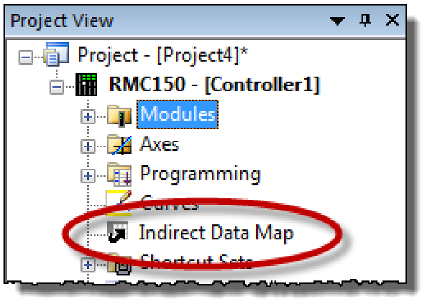
3. Click **OK**. The changes will be applied.

4. On the **Controller** menu, click **Update Flash**. This will save the changes in Flash memory so they will be saved even when power is removed.

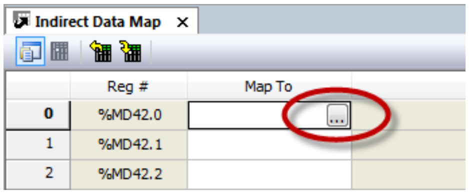
2.3 Setting Up the Indirect Data Map

The RMC's Indirect Data Map is very useful for communications with a PLC. Typically, you will want the PLC to read from and write to a number of items (32-bit data registers) in the RMC. These data registers are usually scattered in many different address locations in the RMC. You can map all these items to the Indirect Data Map. The PLC then simply reads and writes from the Indirect Data Map, instead of from many different locations.

The Indirect Data Map is very useful for frequently accessed registers. Larger reads or writes, or less frequent ones, can be done directly instead of via the Indirect Data Map.

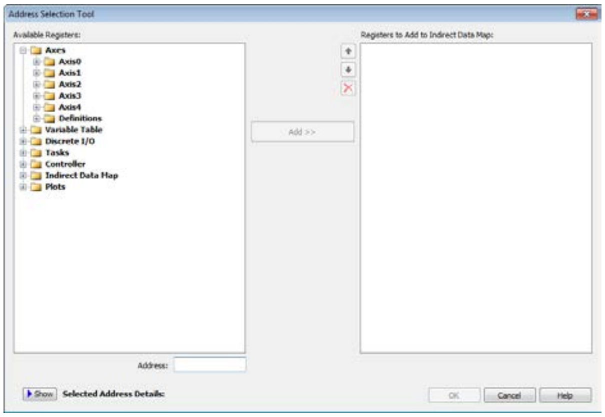


1. In the RMCTools Project pane, double-click **Indirect Data Map**.



	Reg #	Map To
0	%MD42.0	
1	%MD42.1	
2	%MD42.2	

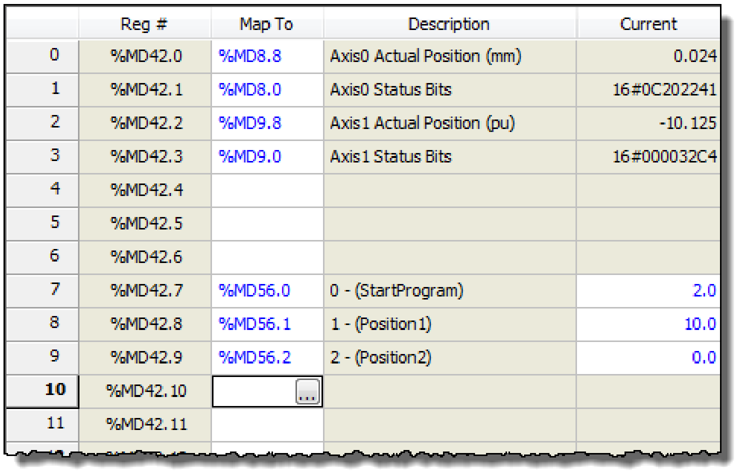
2. In the **Map To** column, click a cell, then click the ellipsis button.



3. In the Address Selection Tool, browse to a desired register in the RMC and click **Add**. Repeat for as many registers as you wish, then click **OK**.

If you plan on using the Indirect Data Map for both reading and writing, it is good practice to set one area of the Indirect Data Map for reading, and the other for writing.

For example, this Indirect Data Map was set up to include Axis 0 and Axis 1 status information in a read area, and some variables in a write area.

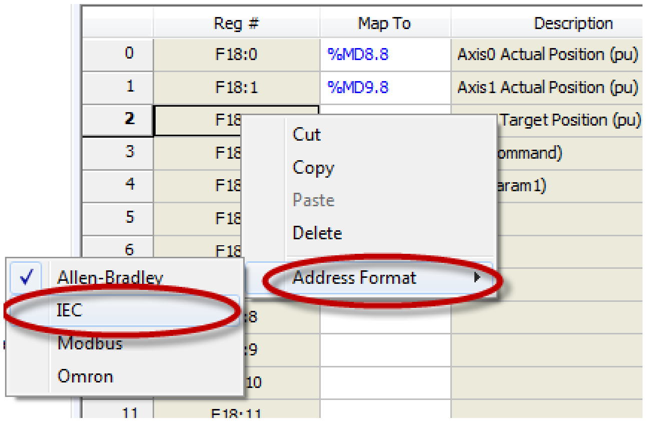


	Reg #	Map To	Description	Current
0	%MD42.0	%MD8.8	Axis0 Actual Position (mm)	0.024
1	%MD42.1	%MD8.0	Axis0 Status Bits	16#0C202241
2	%MD42.2	%MD9.8	Axis1 Actual Position (pu)	-10.125
3	%MD42.3	%MD9.0	Axis1 Status Bits	16#000032C4
4	%MD42.4			
5	%MD42.5			
6	%MD42.6			
7	%MD42.7	%MD56.0	0 - (StartProgram)	2.0
8	%MD42.8	%MD56.1	1 - (Position1)	10.0
9	%MD42.9	%MD56.2	2 - (Position2)	0.0
10	%MD42.10	...		
11	%MD42.11			

2.4 Understanding RMC Addresses

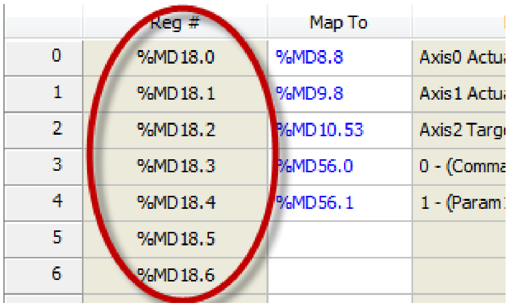
The RMC's 32-bit data registers can be addressed in several different forms. For the purposes of communications with Mitsubishi PLC's, you should use the IEC form. Wherever an address is displayed in RMCTools, you can right-click and choose the desired format.

For example, in the Indirect Data Map, in the **Reg** column, right-click a cell, point to **Address Format**, and choose **IEC**.



	Reg #	Map To	Description
0	F18:0	%MD8.8	Axis0 Actual Position (pu)
1	F18:1	%MD9.8	Axis1 Actual Position (pu)
2	F18		Target Position (pu)
3	F18		ommand)
4	F18		aram1)
5	F18		
6	F18		
	:8		
	:9		
	10		
11	F18:11		

The addresses will now be displayed in IEC format:



	Reg #	Map To	
0	%MD18.0	%MD8.8	Axis0 Actu
1	%MD18.1	%MD9.8	Axis1 Actu
2	%MD18.2	%MD10.53	Axis2 Targ
3	%MD18.3	%MD56.0	0 - (Comma
4	%MD18.4	%MD56.1	1 - (Param
5	%MD18.5		
6	%MD18.6		

Files and Elements

The IEC address of an RMC register has two levels. The first is called the **file** number, the second is called the **element** number. In the Mitsubishi PLC, you will use the file and element numbers when communicating with the RMC.

Example: The address for Indirect Data Item 0 in the RMC75 is **%MD18.0**. The file is 18. The element is 0.

Example: The address for variable 7 in the RMC75 is **%MD56.7**. The file is 56. The element is 7.

Finding Addresses in the RMCTools Help

You can use the RMCTools help to find the address for any register in the RMC:

1. On the **Help** menu, choose **Help Topics**.
2. On the **Index** tab, type **Register Map**, then choose your controller type. This topic lists the addresses of all the registers in the RMC.

Chapter 3

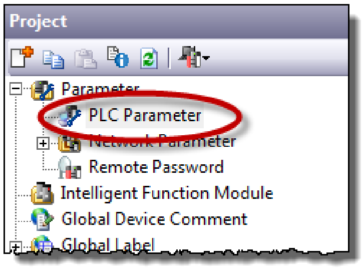
3 PLC Configuration

These are the steps to configure the iQ or L CPU to communicate with the RMC.

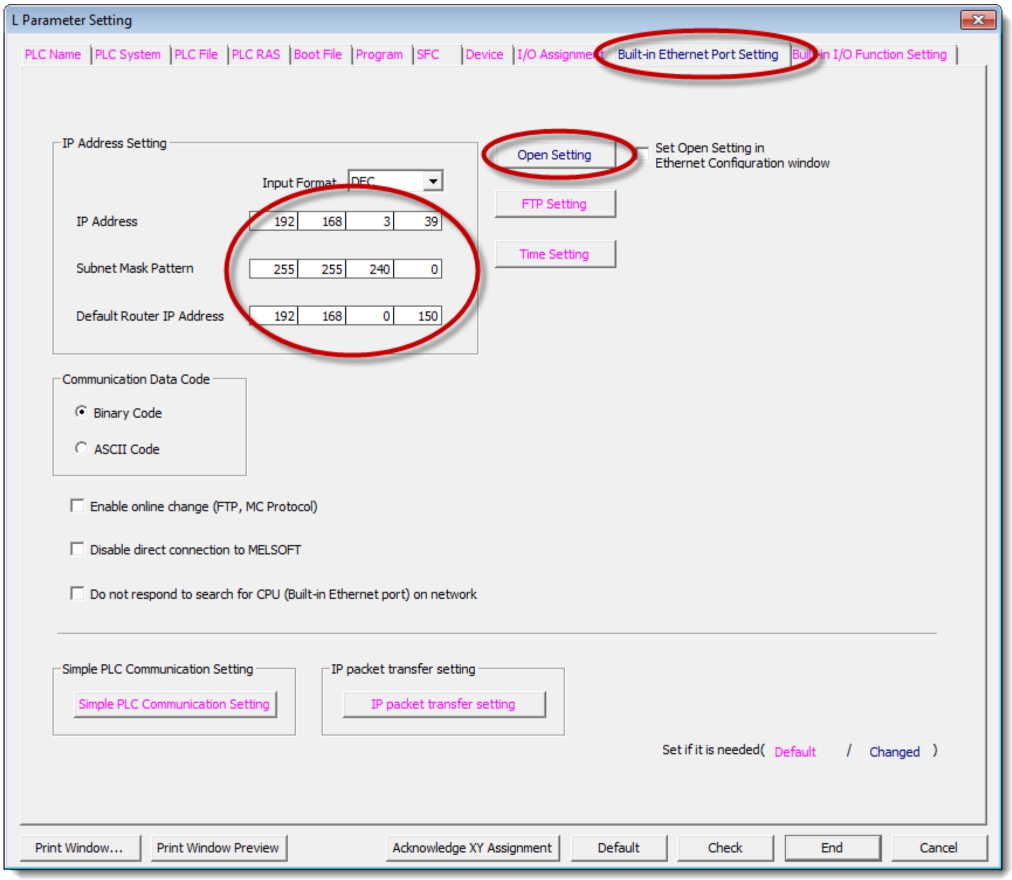
3.1 Configure the PLC Parameters

1. Open GX Works2 and create a new for the appropriate CPU module. You can choose a Simple Project with Labels, or a Structured Project with Structured Ladder/FBD.

2. In the Project tree, expand the **Parameter** node and double-click **PLC Parameter**.



3. On the **Built-on Ethernet Port Setting** tab, set the IP Address, Subnet Mask Pattern, and Default Router IP Address to your desired values.



L Parameter Setting

IP Address Setting

Input Format DEC

IP Address 192 168 3 39

Subnet Mask Pattern 255 255 240 0

Default Router IP Address 192 168 0 150

Set Open Setting in Ethernet Configuration window

Communication Data Code

● Binary Code

○ ASCII Code

□ Enable online change (FTP, MC Protocol)

□ Disable direct connection to MELSOFT

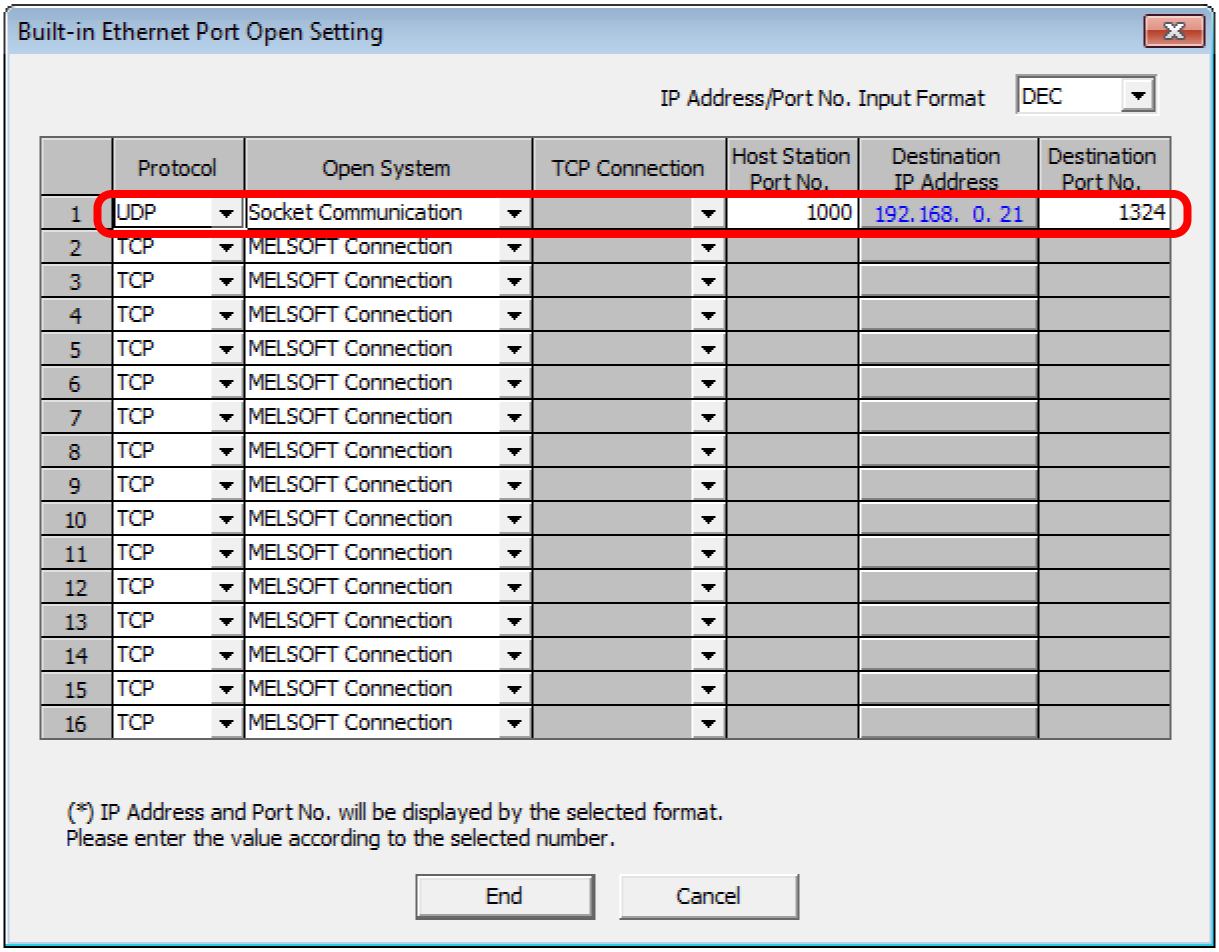
□ Do not respond to search for CPU (Built-in Ethernet port) on network

Set if it is needed(Default / Changed)

PLC Name
PLC System
PLC File
PLC RAS
Boot File
Program
SFC
Device
I/O Assignment
Built-in Ethernet Port Setting
Built-in I/O Function Setting
Open Setting
FTP Setting
Time Setting
Simple PLC Communication Setting
Simple PLC Communication Setting
IP packet transfer setting
IP packet transfer setting
Print Window...
Print Window Preview
Acknowledge XY Assignment
Default
Check
End
Cancel

4. Click the **Open Setting** button.

5. Choose a connection for the RMC. Set the following:
    a. **Protocol:** UDP
    b. **Open System:** Socket Communication
    c. **Host Station Port No**: Set to a hexadecimal value within the range 1 - 1387 or 1392 - FFFE. For example, 1000.
    d. **Destination IP Address**: The IP address of the RMC.
    e. **Destination Port No**: 1324



Built-in Ethernet Port Open Setting

IP Address/Port No. Input Format: DEC

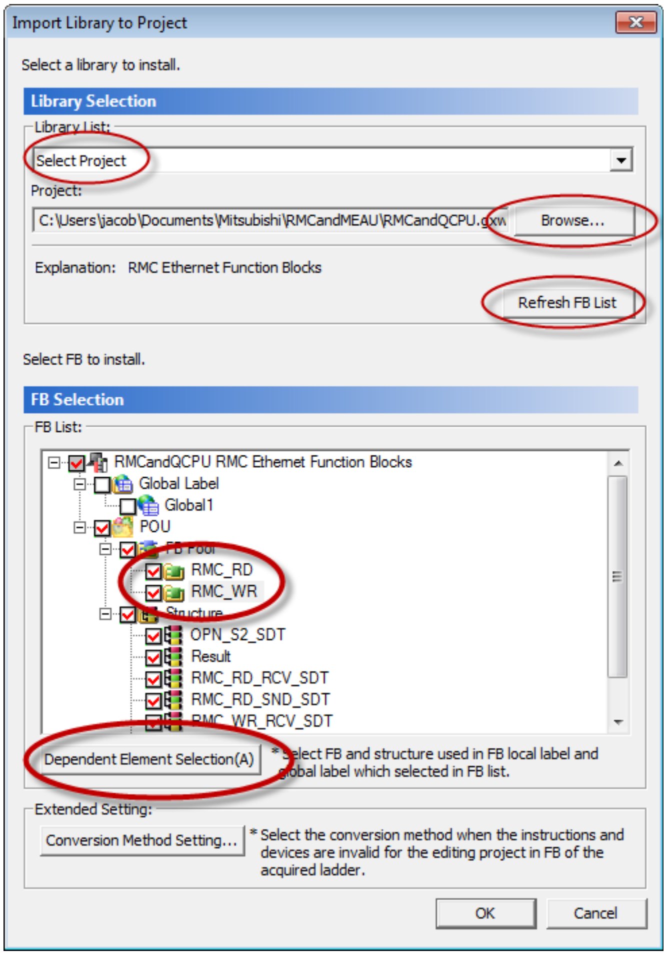
	Protocol	Open System	TCP Connection	Host Station Port No.	Destination IP Address	Destination Port No.
1	UDP	Socket Communication		1000	192.168. 0. 21	1324
2	TCP	MELSOFT Connection				
3	TCP	MELSOFT Connection				
4	TCP	MELSOFT Connection				
5	TCP	MELSOFT Connection				
6	TCP	MELSOFT Connection				
7	TCP	MELSOFT Connection				
8	TCP	MELSOFT Connection				
9	TCP	MELSOFT Connection				
10	TCP	MELSOFT Connection				
11	TCP	MELSOFT Connection				
12	TCP	MELSOFT Connection				
13	TCP	MELSOFT Connection				
14	TCP	MELSOFT Connection				
15	TCP	MELSOFT Connection				
16	TCP	MELSOFT Connection				

(*) IP Address and Port No. will be displayed by the selected format.
Please enter the value according to the selected number.

6. Click **End**, then write the parameters to the PLC and reset the PLC to apply the parameter settings.

3.2 Add the RMC Read and RMC Write Function Blocks (Simple Project)

1. On the **Project** menu, choose **Library**, then click **Install**.
    a. In the **Library List** box, choose **Select Project.**
    b. Click **Browse**.
    c. At the bottom of the **Open** dialog, if the button says **Open a Single File Format Project,** click it.
    d. Browse to Delta's **RMC_Ethernet_FB.gxw** file.
    e. Click **Refresh FB List**.
    f. In the **FB List**, check the **RMC_RD** and **RMC_WR** items.
    g. Click **Dependent Element Selection(A)** and click **OK**.
    h. Click **OK**.



Import Library to Project

Select a library to install.

Library Selection

Library List: Select Project

Project: C:\Users\jacob\Documents\Mitsubishi\RMCandMEAU\RMCandQCPU.gxw

Explanation: RMC Ethernet Function Blocks

Select FB to install.

FB Selection

FB List:

- ☑ RMCandQCPU RMC Ethemet Function Blocks
  - □ Global Label
    - □ Global1
  - ☑ POU
    - ☑ FB Pool
      - ☑ RMC_RD
      - ☑ RMC_WR
    - ☑ Structure
      - ☑ OPN_S2_SDT
      - ☑ Result
      - ☑ RMC_RD_RCV_SDT
      - ☑ RMC_RD_SND_SDT
      - ☑ RMC_WR_RCV_SDT

* Select FB and structure used in FB local label and global label which selected in FB list.

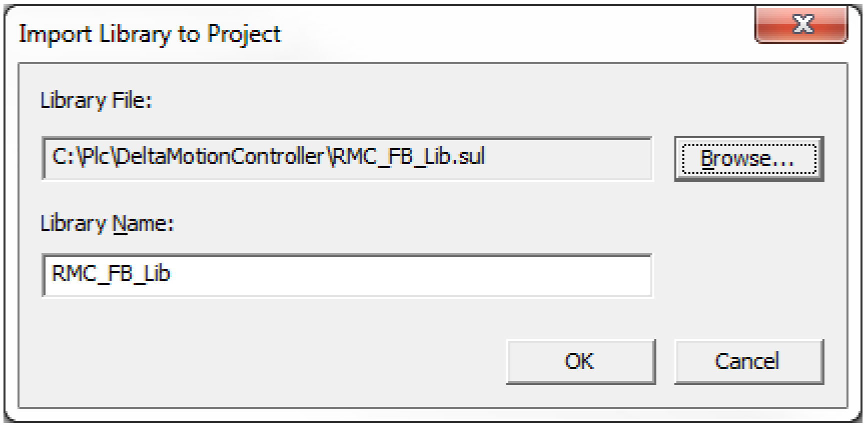
Extended Setting:

* Select the conversion method when the instructions and devices are invalid for the editing project in FB of the acquired ladder.

The RMC Read and RMC Write function blocks will now appear in the FB Pool in your project, and some associated structured data types will have been added.

3.3 Add the RMC Read and RMC Write Function Blocks (Structured Project)

1. In the **Navigation** panel, choose **User Library**.
2. On the **Project** menu, choose **Library**, then click **Install**.
   a. Browse to Delta's **RMC_FB_Lib.sul** file.
   b. Click **Open**, then **OK**.



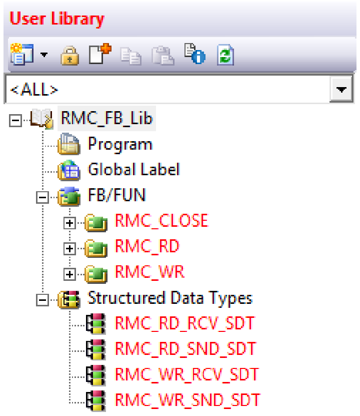
Import Library to Project

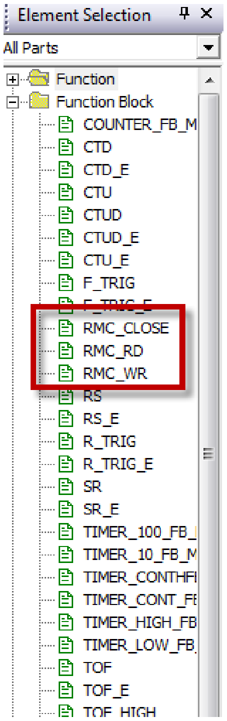
Library File: C:\Plc\DeltaMotionController\RMC_FB_Lib.sul

Library Name: RMC_FB_Lib

The RMC_FB_Lib will now appear in the User Library list.

3. In the User Library list, click on the Lock Symbol to Unlock the Library then expand **RMC_FB_Lib**, then expand **FB/FUN**. You will see the RMC Read and RMC Write function blocks and associated SDTs. The FBs are now directly available for using in your user program. Do not copy and paste them into the FB/FUN area of the users program, this will duplicate the FBs.





Chapter 4

4 Using the RMC Read and RMC Write Function Blocks (Simple Project)

The RMC Read and RMC Write function blocks are used to communicate with the RMC. In this example the RMC IP Address is 192.168.0.21, using PLC Connection #1 UDP.

4.1 Description of Function Blocks (Simple Project)

The **RMC Read** function block inputs and outputs are as follows:

START (Bit)	The function block will perform one read for each rising edge of the Start bit.
CONN (Word)	The connection number (1-16) as defined in the CPU's Open Settings.
RMC_FILE (Word) **RMC_ELEMENT** (Word)	Specify the address in the RMC to read from.
NUM_DW (Word):	The number of 32-bit registers that you wish to read from the RMC. The maximum is 256.
DST_DEV (Word)	The destination device for the data in the PLC. Typically, this is a FLOAT array.
ENABLED (Bit)	Set while the read is in progress. When the read completes, or has an error, the Enabled bit will be cleared.
Done (Bit)	Set when the read successfully completes. This bit is cleared when the function block starts.
Error (Bit)	Set if the read has an error. This bit is cleared when the function block starts.

The **RMC Write** function block inputs and outputs are as follows:

START (Bit)	The function block will perform one write for each rising edge of the Start bit.
CONN (Word)	The connection number (1-16) as defined in the CPU's Open Settings.
SRC_DEV (Word)	The source device of the data in the PLC. Typically, this is a FLOAT array.
RMC_FILE (Word) **RMC_ELEMENT** (Word)	Specify the address in the RMC to write to.
NUM_DW (Word):	The number of 32-bit registers that you wish to write to the RMC. The maximum is 256.
ENABLED (Bit)	Set while the write is in progress. When the write completes, or has an error, the Enabled bit will be cleared.
Done (Bit)	Set when the write successfully completes. This bit is cleared when the function block starts.
Error (Bit)	Set if the write has an error. This bit is cleared when the function block starts.

Important!

When using multiple function blocks with the same Ethernet connection, it is very important that only one function block is active at a time. Follow the examples below.

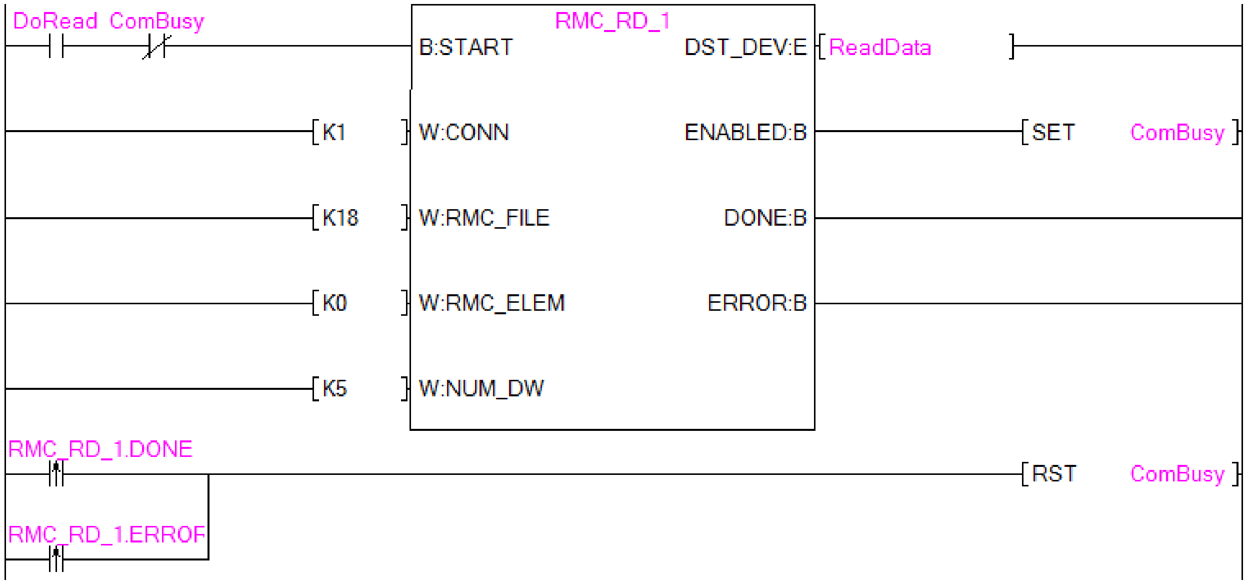
The RMC Read and RMC Write function blocks have a built-in timeout of 0.5 seconds.

4.2 RMC Read Function Block: Continuous Reads

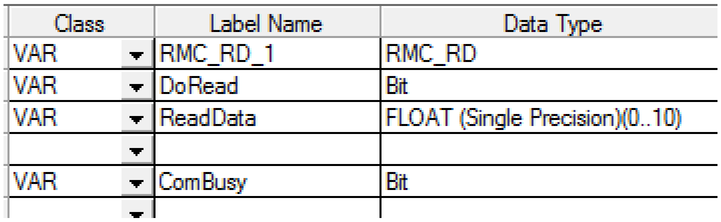
Follow these steps to use the RMC Read function block to continuously read from the RMC:

1. Open a ladder logic program.
2. In the Project tree, click the **RMC Read** function block and drag it to the ladder.
3. GX Works 2 automatically assigns an instance name (RMC_RD_1), which you can change if you wish. Click **OK**.
4. Create logic and add labels as shown below. The ReadData length should be the number of registers to read. The ComBusy bit prevents the function block from being triggered while the communication transaction is in progress, and is very important for synchronizing multiple function blocks, which will be demonstrated later.
5. Compile the code and write it to the PLC. The communications should start when the DoRead bit is set and continuously read as long as DoRead is set.

Example Ladder Logic (Continuous Reads):



The example uses these labels:



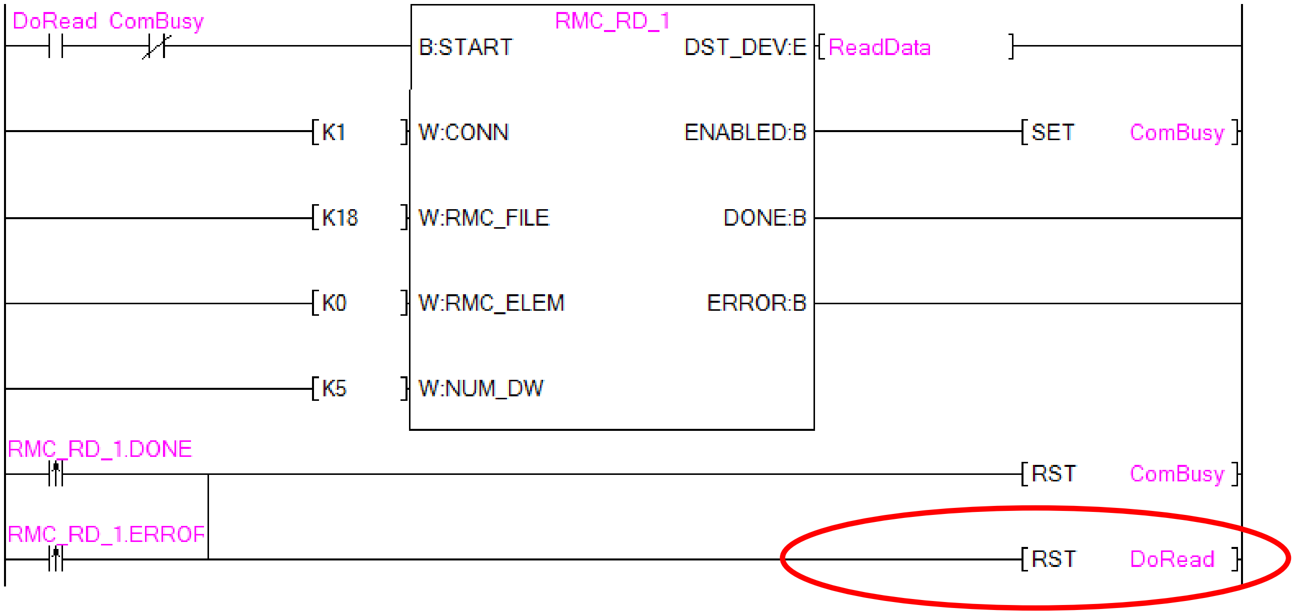
Class	Label Name	Data Type
VAR	RMC_RD_1	RMC_RD
VAR	DoRead	Bit
VAR	ReadData	FLOAT (Single Precision)(0..10)
VAR	ComBusy	Bit

Explanation:

The function block uses Ethernet connection 1, and will read 5 registers from address %MD18.0 in the RMC, placing the values in the **ReadData** label. When the **DoRead** bit is on, and communications are not in progress (**ComBusy** is off), the function block starts. While it is in progress, the **Enabled** bit is set, which will also set the **ComBusy**, indicating the communication transaction is in progress. When the transaction completes, or has an error, the **ComBusy** bit will be cleared. At this point, the function block is triggered again and the cycle will repeat continuously. The reads will stop if the **DoRead** bit is reset.

4.3 RMC Read Function Block: Single Read

Using the RMC Read function block to perform a single read is nearly identical to the continuous reads method. As compared to continuous reads, the additional code is circled below. The labels are all the same as for the continuous reads method.



Explanation:

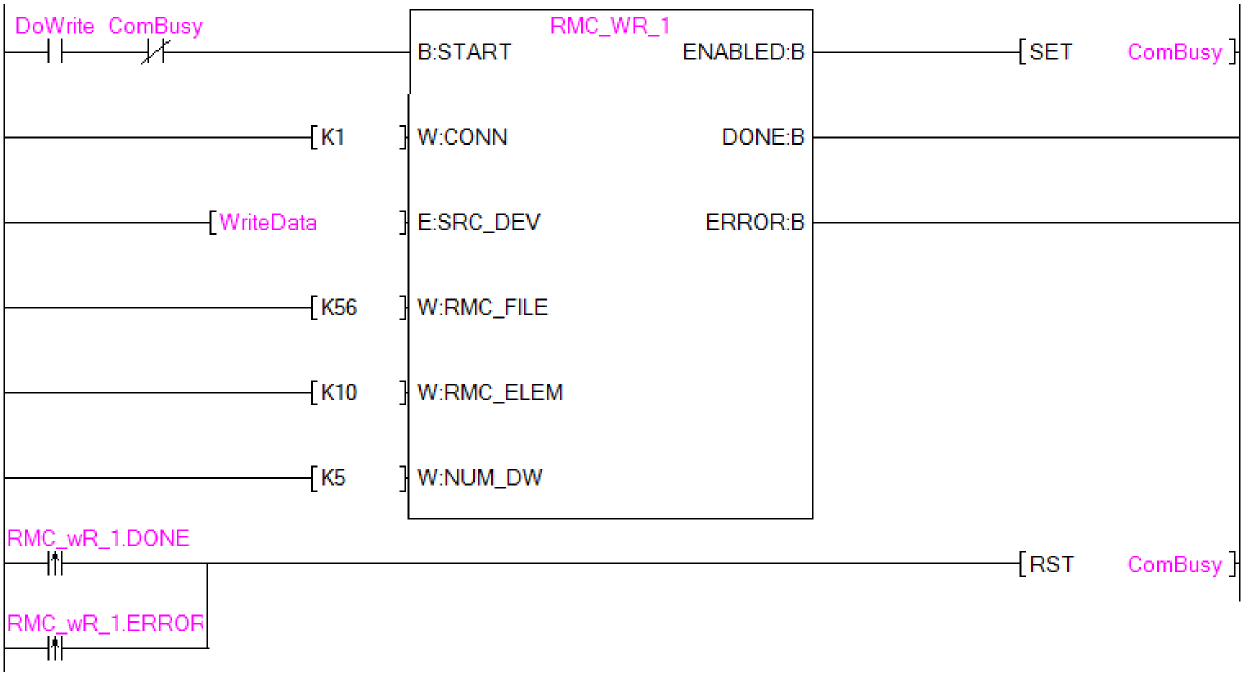
This logic is identical to the previous example, with the difference that when the when the transaction completes, or has an error, the DoRead bit is reset. To perform another read, the DoRead bit must be set.

4.4 RMC Write Function Block: Continuous Writes

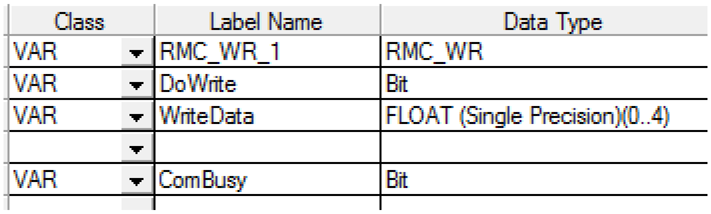
Follow these steps to use the RMC Write function block to continuously write to the RMC:

1. In the Project tree, click the **RMC Write** function block and drag it to the ladder. GX Works 2 automatically assigns it an instance name (RMC_WR_1), which you can change if you wish.
2. Create logic and add labels as shown below. The ComBusy bit prevents the function block from being triggered while the communication transaction is in progress, and is very important for synchronizing multiple function blocks, which will be demonstrated later.

Example Ladder Logic (Continuous Writes):



The example uses these labels:



Class	Label Name	Data Type
VAR	RMC_WR_1	RMC_WR
VAR	DoWrite	Bit
VAR	WriteData	FLOAT (Single Precision)(0..4)
VAR	ComBusy	Bit

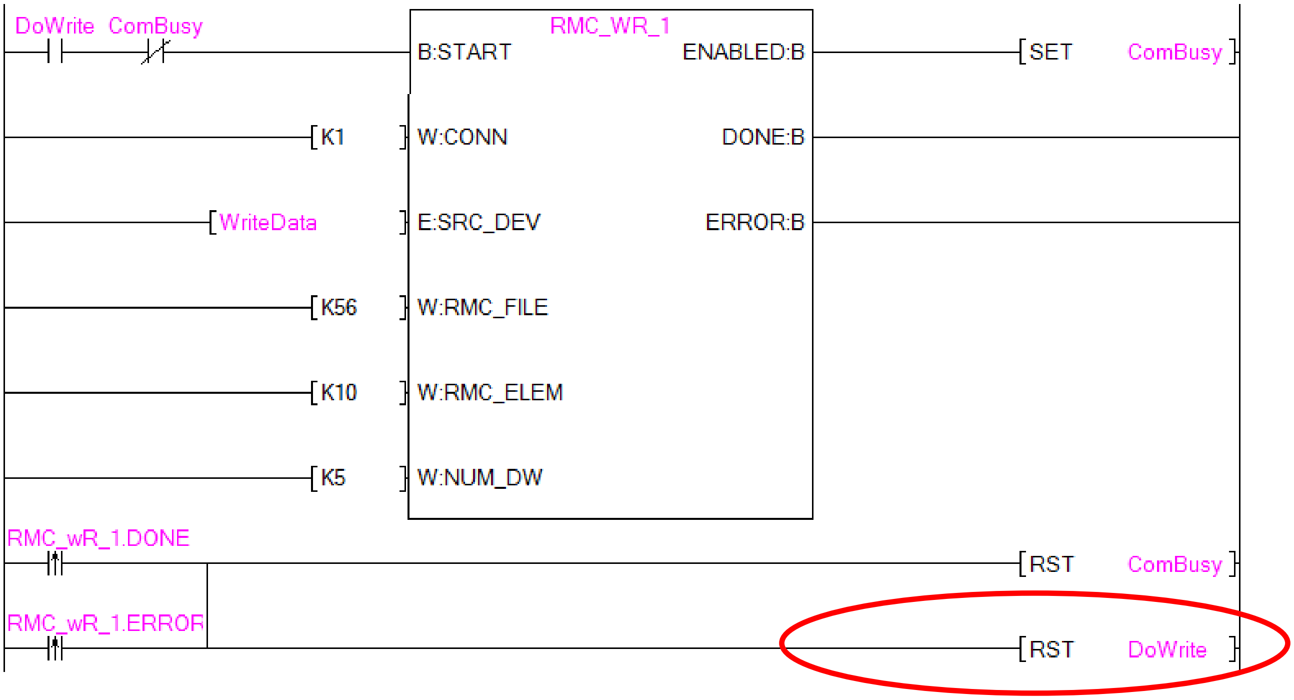
Explanation:

The function block uses Ethernet connection 1, and will write 10 registers to address %MD56.10 in the RMC. The source data is in the **WriteData** array.

When the **DoWrite** bit is on, and communications are not in progress (**ComBusy** is off), the function block starts. While it is in progress, the **Enabled** bit is set, which will also set the **ComBusy**, indicating the communication transaction is in progress. When the transaction completes, or has an error, the **ComBusy** bit will be cleared. At this point, the function block is triggered again and the cycle will repeat continuously. The reads will stop if the **DoWrite** bit is reset.

4.5 RMC Write Function Block: Single Write

Using the RMC Write function block to perform a single write is nearly identical to continuous writes. As compared to continuous writes, the additional code is circled below. The labels are all the same.



Explanation:

This logic is identical to the previous example, with the difference that when the when the transaction completes, or has an error, the DoWrite bit is reset. To perform another read, the DoWrite bit must be set.

4.6 Using Multiple RMC Read and RMC Write Function Blocks

It is very important that for all the function blocks that use the same Ethernet connection, that only one function block is active at a time. To achieve this, use a bit to indicate when the communications are busy. Such a bit can be named ComBusy. This method uses the same logic as shown in the previous examples.

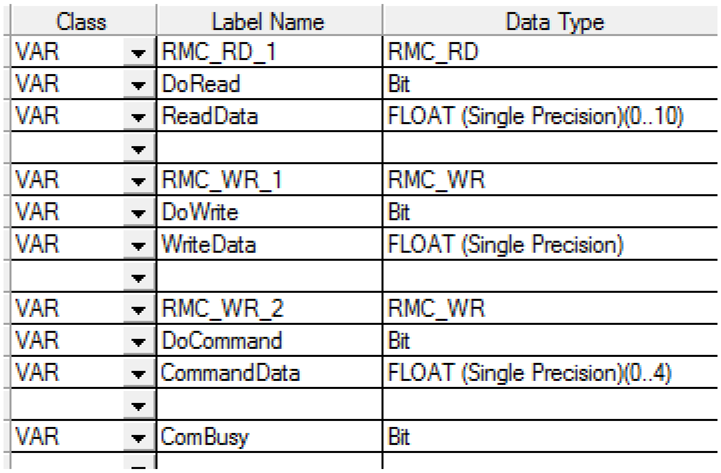
You may add as many reads and writes as you wish, and any of them can be continuous or single.

Example:

- Continuously read items 0-9 from the RMC75E's Indirect Data Map (address %MD18.0)
- Do a single write on demand to variable 10 on the RMC75E (address %MD56.10)
- Do a single write* on demand to Axis 0 Command area in the RMC75E (address %MD25.0)

*****When writing commands, always use single writes. Commands are not intended to be sent continuously.

This example uses the labels:



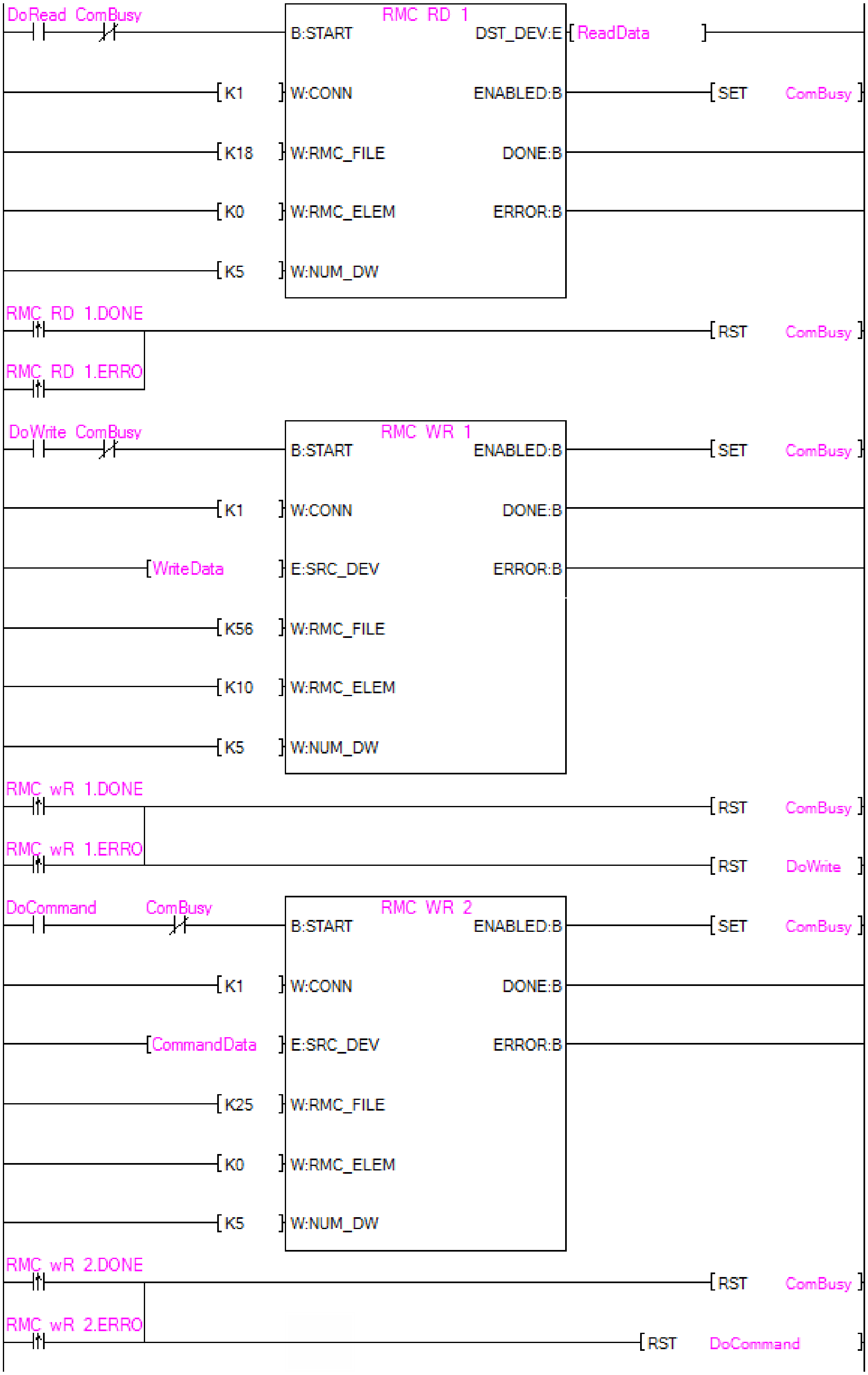
Class	Label Name	Data Type
VAR	RMC_RD_1	RMC_RD
VAR	DoRead	Bit
VAR	ReadData	FLOAT (Single Precision)(0..10)
VAR	RMC_WR_1	RMC_WR
VAR	DoWrite	Bit
VAR	WriteData	FLOAT (Single Precision)
VAR	RMC_WR_2	RMC_WR
VAR	DoCommand	Bit
VAR	CommandData	FLOAT (Single Precision)(0..4)
VAR	ComBusy	Bit

Explanation:

The RMC_RD_1 function block will continuously read when the **DoRead** bit is set. After each read completes, the **ComBusy** bit is reset, so that if the **DoWrite** or **DoCommand** bits are set, the corresponding RMC_WR_1 or RMC_WR_2 function block will start. Once the write or command is complete, the **ComBusy** is again reset, and the logic continues its cycle.

Ladder Logic on next page →

Example Ladder Logic:



4.7 Communicating With Multiple RMC's

When communicating with multiple RMC's, use a separate Ethernet connection (as defined in the Open Connection settings) for each RMC. Each connection must use a unique Host Station Port No. In the ladder logic, use a separate ComBusy bit for each connection.

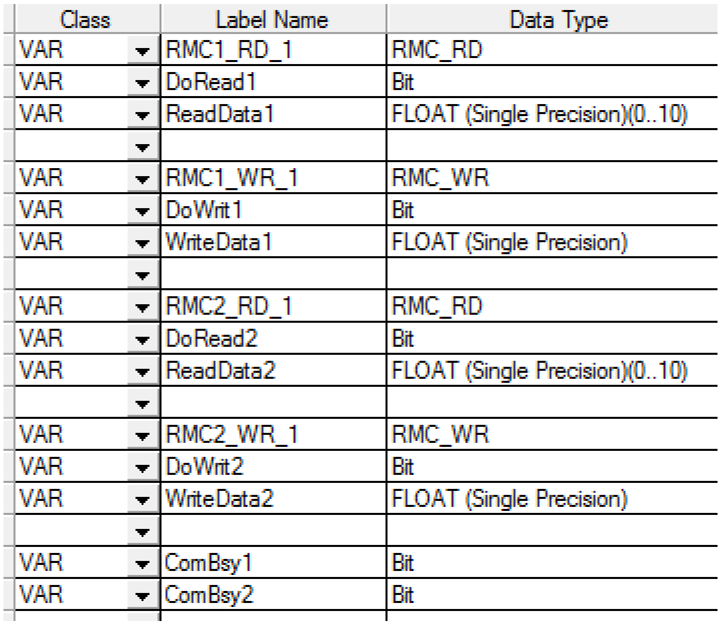
Example:

An iQ CPU communicates with two RMC75E's, called RMC1 and RMC2. RMC1 uses Ethernet connection 1, and RMC2 uses Ethernet connection 2 (defined in the CPU's Built-in Ethernet Open Connection settings). The bits ComBusy1 and ComBusy2 apply to the respective connections.

This example does the following:

- For RMC1, continuously read 8 items from item 0 of the Indirect Data Map (address %MD18.0)
- For RMC1, do a single write on demand to variable 5 (address %MD56.10)
- For RMC2, continuously read 8 items from item 0 of the Indirect Data Map (address %MD18.0)
- For RMC2, do a single write on demand to variable 5 (address %MD56.10)

This example uses the labels:



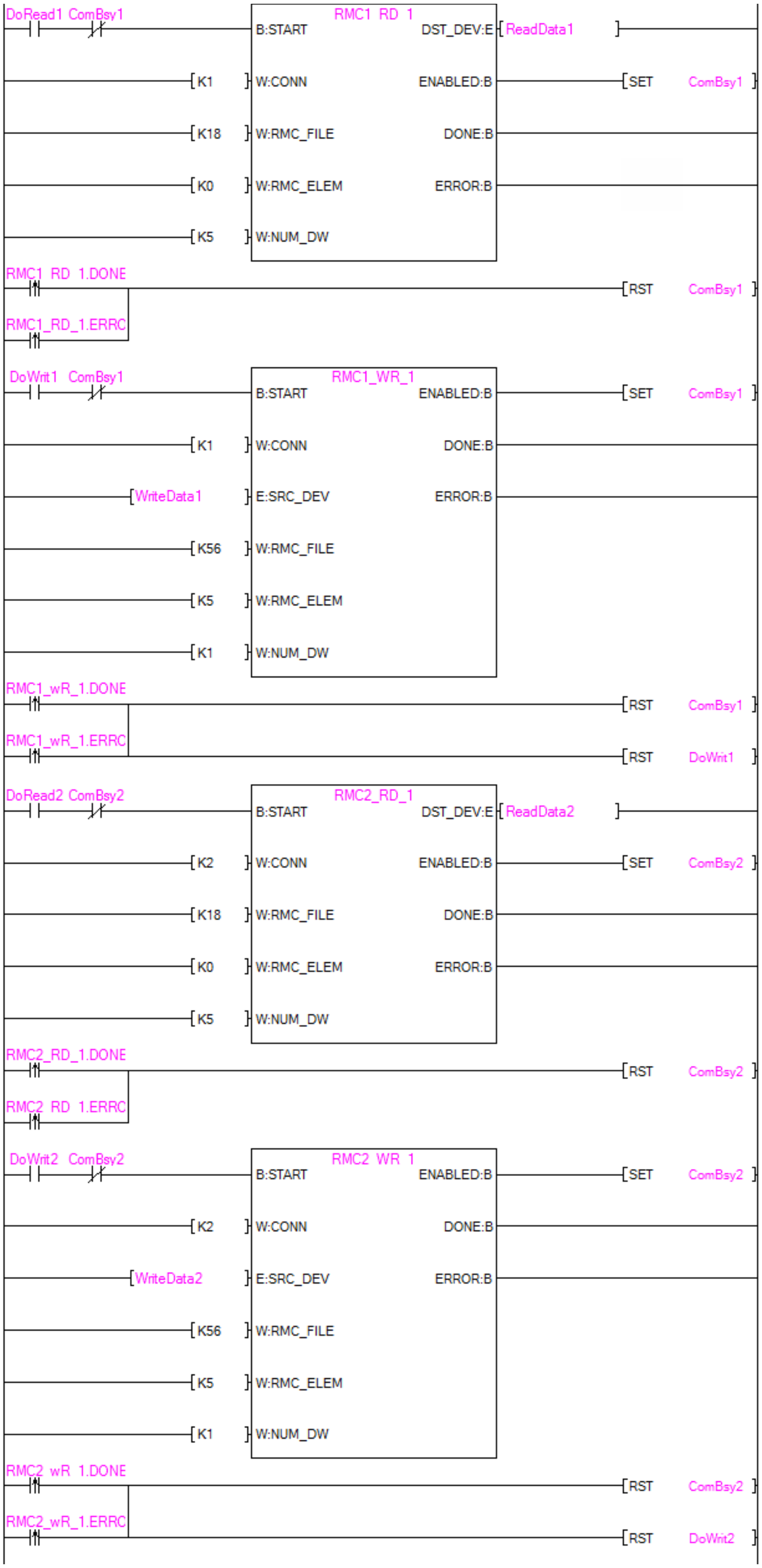
Class	Label Name	Data Type
VAR	RMC1_RD_1	RMC_RD
VAR	DoRead1	Bit
VAR	ReadData1	FLOAT (Single Precision)(0..10)
VAR	RMC1_WR_1	RMC_WR
VAR	DoWrit1	Bit
VAR	WriteData1	FLOAT (Single Precision)
VAR	RMC2_RD_1	RMC_RD
VAR	DoRead2	Bit
VAR	ReadData2	FLOAT (Single Precision)(0..10)
VAR	RMC2_WR_1	RMC_WR
VAR	DoWrit2	Bit
VAR	WriteData2	FLOAT (Single Precision)
VAR	ComBsy1	Bit
VAR	ComBsy2	Bit

Explanation:

In the ladder logic, the first two function blocks communicate with RMC1 and use the **ComBsy1** bit. The last two function blocks communicate with RMC2 and use the **ComBsy1** bit.

Ladder Logic on next page →

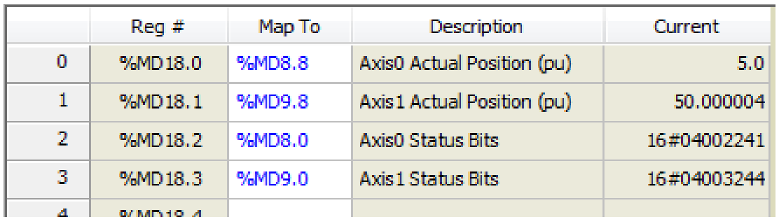
Example Ladder Logic:



4.8 Mixing Data Types

When using a Mitsubishi PLC to communicate with an RMC, the communicated data can be of type FLOAT (Single Precision) or Double Word.

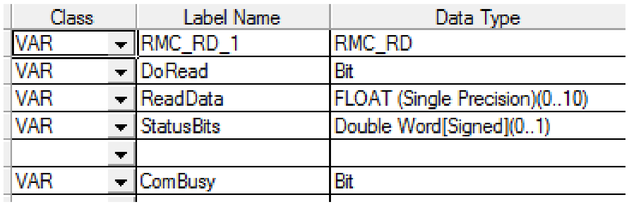
It is possible that the read or write data contains items of both data types. For example, in the following Indirect Data Map, the first two items are REALs, and the last two items are DWORDs. These RMC data types correspond to FLOAT (Single Precision) and Double Word in the Mitsubishi CPU.



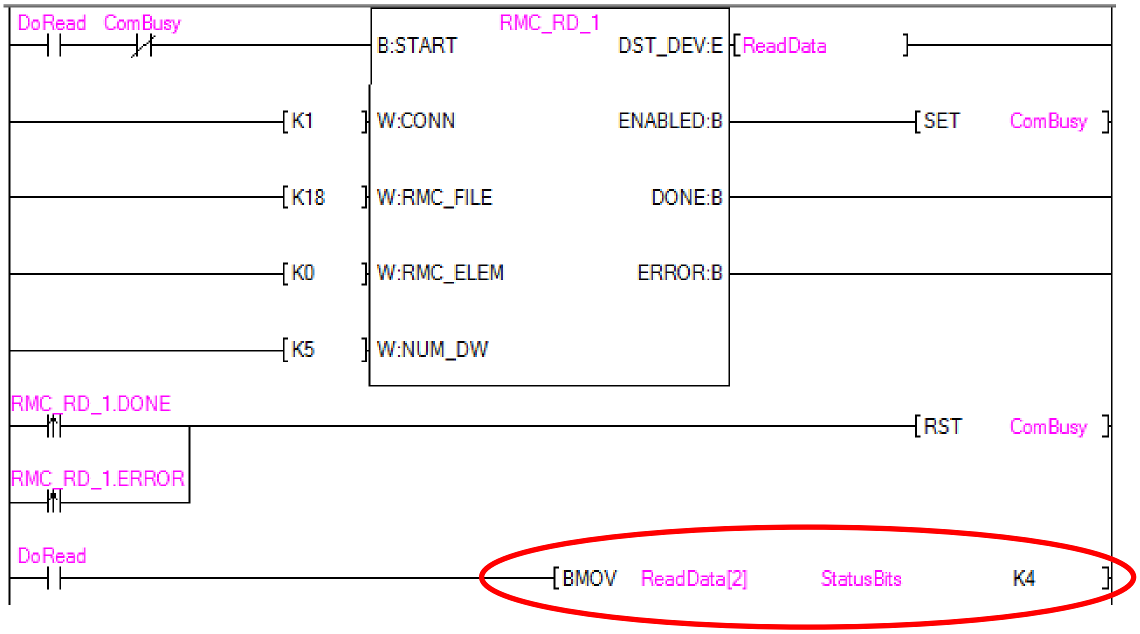
	Reg #	Map To	Description	Current
0	%MD18.0	%MD8.8	Axis0 Actual Position (pu)	5.0
1	%MD18.1	%MD9.8	Axis1 Actual Position (pu)	50.000004
2	%MD18.2	%MD8.0	Axis0 Status Bits	16#04002241
3	%MD18.3	%MD9.0	Axis1 Status Bits	16#04003244
4	%MD18.4			

Reading these four values with one function block in the PLC will result in an array containing data with different types. The data itself will be correct, but one of the types will be displayed incorrectly. To remedy this, use the BMOV instruction to copy the values to an array of the correct type, or to a label defined by a structured data type.

The example logic below shows logic that reads the shown above in the Indirect Data Map (address %MD18.0). The circled item below is the BMOV that copies the third and fourth items to the label array StatusBits, of type Double Word. Notice that the number of words to copy (K4 in the BMOV instruction below) must be twice the number of registers to copy, because the BMOV is in 16-bit units, and the RMC registers are 32 bits.



Class	Label Name	Data Type
VAR	RMC_RD_1	RMC_RD
VAR	DoRead	Bit
VAR	ReadData	FLOAT (Single Precision)(0..10)
VAR	StatusBits	Double Word[Signed](0..1)
VAR	ComBusy	Bit



Chapter 5

5 Using the RMC Read and RMC Write Function Blocks (Structured Project)

The RMC Read and RMC Write function blocks are used to communicate with the RMC.In this example the RMC IP Address is 192.168.1.102, using PLC Connection #4 TCP

5.1 Description of Function Blocks (Structured Project)

The **RMC Read** function block inputs and outputs are as follows:

START (Bit)	The function block will perform one read for each rising edge of the Start bit.
CONN (Word)	The connection number (1-16) as defined in the CPU's Open Settings.
RMC_FILE (Word) **RMC_ELEMENT** (Word)	Specify the address in the RMC to read from.
NUM_DW (Word):	The number of 32-bit registers that you wish to read from the RMC. The maximum is 1024.
DST_DEV (FLOAT)(0..1023)	The destination for the data in the PLC.
ENABLED (Bit)	Set while the read is in progress. When the read completes, or has an error, the Enabled bit will be cleared.
Done (Bit)	Set when the read successfully completes. This bit is cleared when the function block starts.
Error (Bit)	Set if the read has an error. This bit is cleared when the function block starts.

The **RMC Write** function block inputs and outputs are as follows:

START (Bit)	The function block will perform one write for each rising edge of the Start bit.
CONN (Word)	The connection number (1-16) as defined in the CPU's Open Settings.
SRC_DEV (FLOAT)(0..1023)	The source of the data in the PLC..
RMC_FILE (Word) **RMC_ELEMENT** (Word)	Specify the address in the RMC to write to.
NUM_DW (Word):	The number of 32-bit registers that you wish to write to the RMC. The maximum is 1024.
ENABLED (Bit)	Set while the write is in progress. When the write completes, or has an error, the Enabled bit will be cleared.
Done (Bit)	Set when the write successfully completes. This bit is cleared when the function block starts.
Error (Bit)	Set if the write has an error. This bit is cleared when the function block starts.

Important!

When using multiple function blocks with the same Ethernet connection, it is very important that only one function block is active at a time. Follow the examples below.

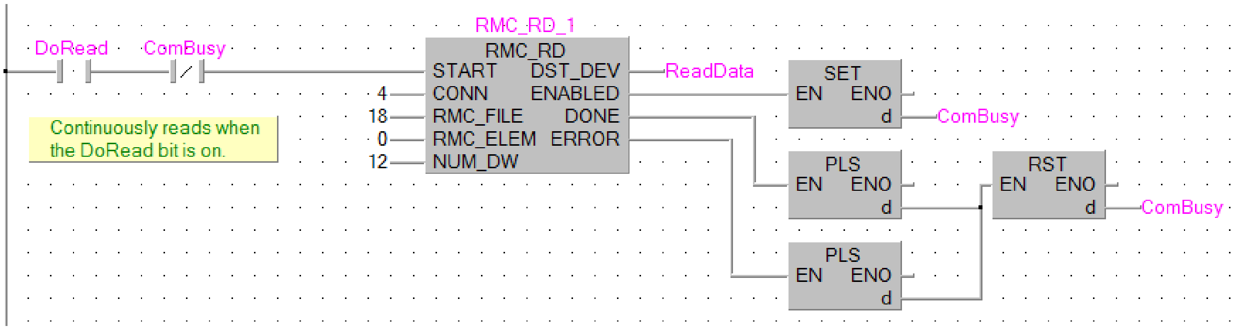
The RMC Read and RMC Write function blocks have a built-in timeout of 0.5 seconds.

5.2 RMC Read Function Block: Continuous Reads

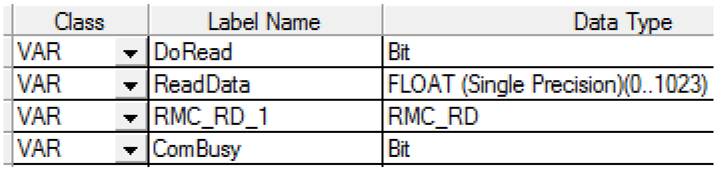
Follow these steps in a structured project to use the RMC Read function block to continuously read from the RMC:

1. Open a structured ladder program.
2. In the User Library, click the **RMC Read** function block and drag it to the ladder.
3. GX Works 2 automatically assigns an instance name (RMC_RD_1), which you can change if you wish. Click **Close**.
4. Create logic and add labels as shown below. The ComBusy bit prevents the function block from being triggered while the communication transaction is in progress, and is very important for synchronizing multiple function blocks, which will be demonstrated later.
5. Compile the code and write it to the PLC. The communications should start when the DoRead bit is set and continuously read as long as DoRead is set.

Example Ladder Logic (Continuous Reads):



The example uses these labels:



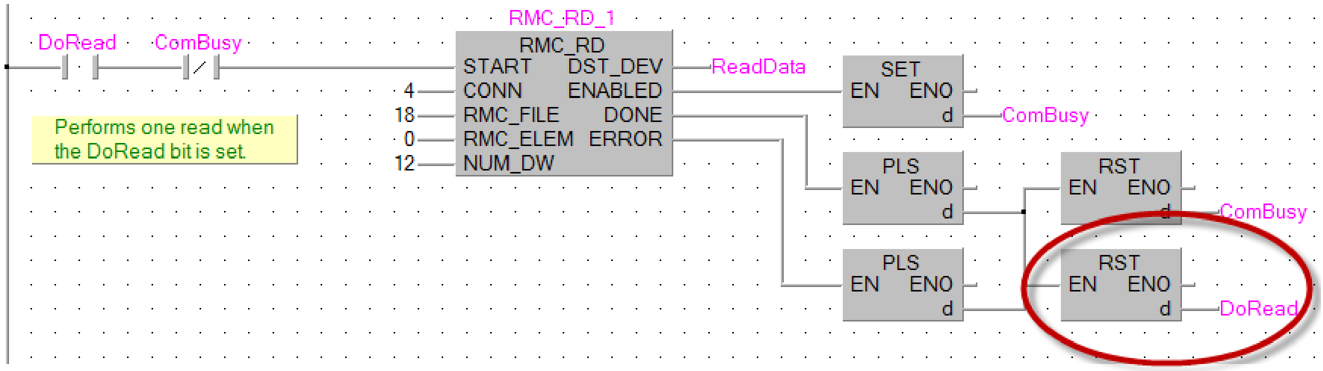
Class	Label Name	Data Type
VAR	DoRead	Bit
VAR	ReadData	FLOAT (Single Precision)(0..1023)
VAR	RMC_RD_1	RMC_RD
VAR	ComBusy	Bit

Explanation:

The example function above uses Ethernet connection 4, and will read 12 registers from address %MD18.0 in the RMC, placing the values in the **ReadData** label. When the **DoRead** bit is on, and communications are not in progress (**ComBusy** is off), the function block starts. While it is in progress, the **Enabled** bit is set, which will also set the **ComBusy**, indicating the communication transaction is in progress. When the transaction completes, or has an error, the **ComBusy** bit will be cleared. At this point, the function block is triggered again and the cycle will repeat continuously. The reads will stop if the **DoRead** bit is reset.

5.3 RMC Read Function Block: Single Read

Using the RMC Read function block to perform a single read is nearly identical to the continuous reads method. As compared to continuous reads, the additional code is circled below. The labels are all the same as for the continuous reads method.



Explanation:

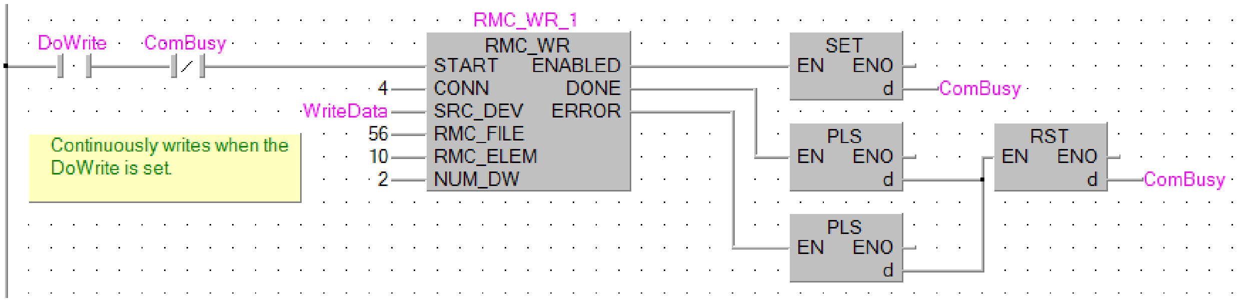
This logic is identical to the previous example, with the difference that when the when the transaction completes, or has an error, the DoRead bit is reset. To perform another read, the DoRead bit must be set.

5.4 RMC Write Function Block: Continuous Writes

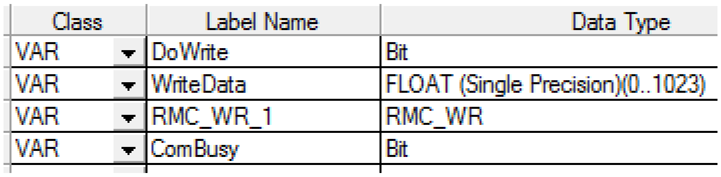
Follow these steps in a structured project to use the RMC Write function block to continuously write to the RMC:

1. In the User Library, click the **RMC Read** function block and drag it to the ladder.
2. GX Works 2 automatically assigns an instance name (RMC_RD_1), which you can change if you wish. Click **Close**.
3. Create logic and add labels as shown below. The ComBusy bit prevents the function block from being triggered while the communication transaction is in progress, and is very important for synchronizing multiple function blocks, which will be demonstrated later.

Example Ladder Logic (Continuous Writes):



The example uses these labels:



Class	Label Name	Data Type
VAR	DoWrite	Bit
VAR	WriteData	FLOAT (Single Precision)(0..1023)
VAR	RMC_WR_1	RMC_WR
VAR	ComBusy	Bit

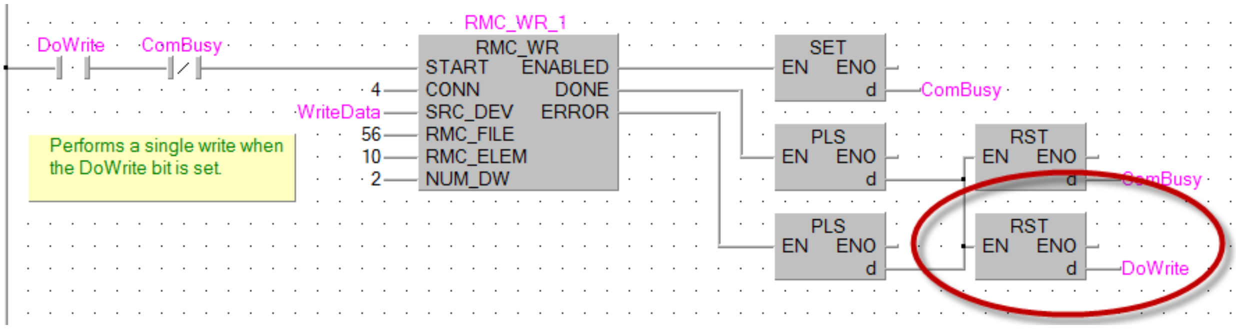
Explanation:

The example function block above uses Ethernet connection 4, and will write 2 registers to address %MD56.10 in the RMC. The source data is in the **WriteData** array.

When the **DoWrite** bit is on, and communications are not in progress (**ComBusy** is off), the function block starts. While it is in progress, the **Enabled** bit is set, which will also set the **ComBusy**, indicating the communication transaction is in progress. When the transaction completes, or has an error, the **ComBusy** bit will be cleared. At this point, the function block is triggered again and the cycle will repeat continuously. The reads will stop if the **DoWrite** bit is reset.

5.5 RMC Write Function Block: Single Write

Using the RMC Write function block to perform a single write is nearly identical to continuous writes. As compared to continuous writes, the additional code is circled below. The labels are all the same.



Explanation:

This example logic is identical to the previous example, with the difference that when the when the transaction completes, or has an error, the DoWrite bit is reset. To perform another read, the DoWrite bit must be set.

5.6 Using Multiple RMC Read and RMC Write Function Blocks

It is very important that for all the function blocks that use the same Ethernet connection, that only one function block is active at a time. To achieve this, use a bit to indicate when the communications are busy. Such a bit can be named ComBusy. This method uses the same logic as shown in the previous examples.

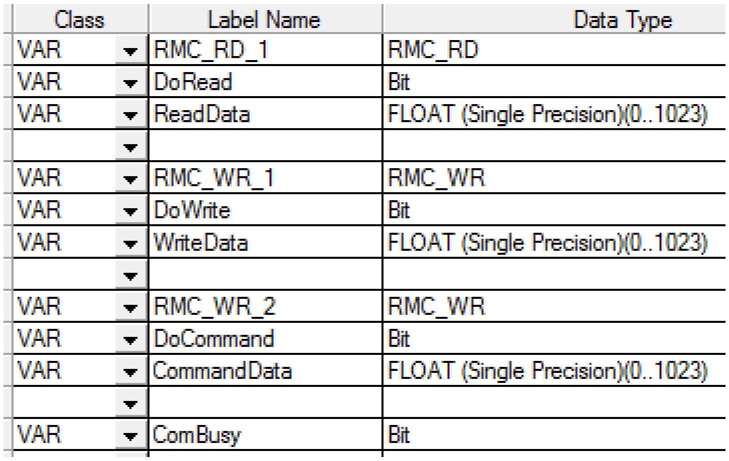
You may add as many reads and writes as you wish, and any of them can be continuous or single.

Example:

- Continuously read items 0-11 from the RMC75E's Indirect Data Map (address %MD18.0)
- Do a single write on demand to variable 10 on the RMC75E (address %MD56.10)
- Do a single write* on demand to Axis 0 Command area in the RMC75E (address %MD25.0)

*****When writing commands, always use single writes. Commands are not intended to be sent continuously.

This example uses the labels:



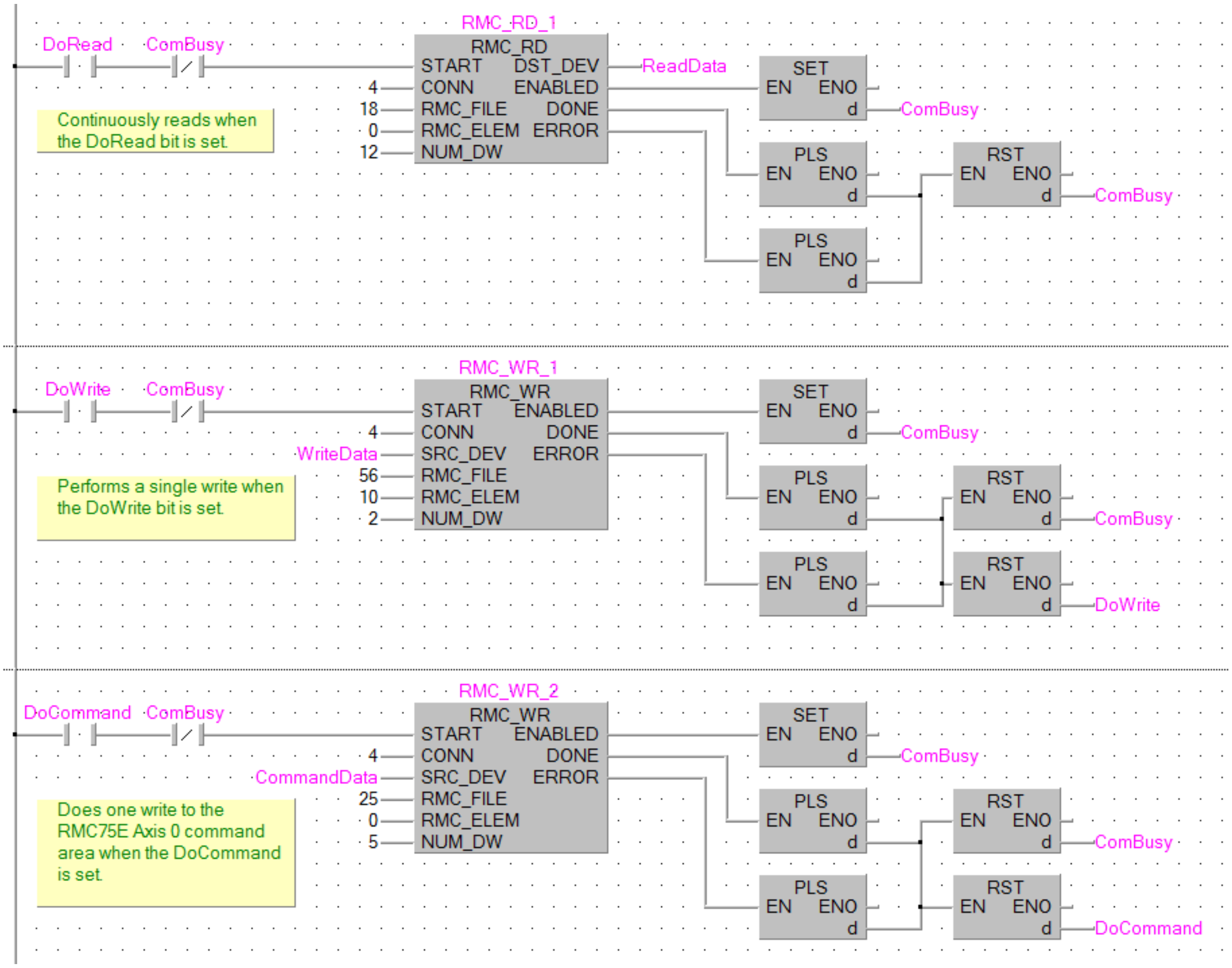
Class	Label Name	Data Type
VAR	RMC_RD_1	RMC_RD
VAR	DoRead	Bit
VAR	ReadData	FLOAT (Single Precision)(0..1023)
VAR	RMC_WR_1	RMC_WR
VAR	DoWrite	Bit
VAR	WriteData	FLOAT (Single Precision)(0..1023)
VAR	RMC_WR_2	RMC_WR
VAR	DoCommand	Bit
VAR	CommandData	FLOAT (Single Precision)(0..1023)
VAR	ComBusy	Bit

Explanation:

The RMC_RD_1 function block will continuously read when the **DoRead** bit is set. After each read completes, the **ComBusy** bit is reset, so that if the **DoWrite** or **DoCommand** bits are set, the corresponding RMC_WR_1 or RMC_WR_2 function block will start. Once the write or command is complete, the **ComBusy** is again reset, and the logic continues its cycle.

Ladder Logic on next page →

Example Ladder Logic:



Continuously reads when the DoRead bit is set

Performs a single write when the DoWrite bit is set

Does one write to the RMC75E Axis 0 command area when the DoCommand is set

5.7 Communicating With Multiple RMC's

See Section 4.7

5.8 Mixing Data Types

See Section 4.8

5.9 Device Usage when Using Multiple Function Blocks

The structured RMC READ and RMC Write function blocks may use an unnecessarily large amount of device points when compiled. This is because the ReadData output of the RMC Read and the WriteData input of the RMC Write are defined to be of arrays of length 256. The tags used with the function blocks must therefore also be of length 256.

For smaller reads and writes, Delta recommends modifying the function blocks to use smaller ReadData and WriteData arrays. If you need a combination of large and small reads and/or writes, you can duplicate the function blocks and rename them to indicate the size. Using a function block with ReadData and/or WriteData arrays that match you communications requirements will prevent wasting the device points in the PLC.

Chapter 6

6 Troubleshooting the Communications

This section provides some tips on troubleshooting the communications.

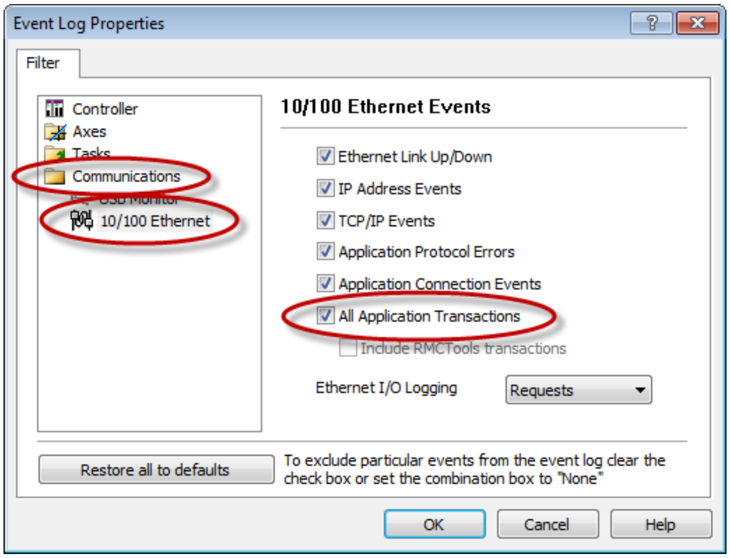
If an error occurs in the communications, check the following:

1. Are the PLC and RMC both connected to the network, with correct IP addresses?
2. Are the PLC's Built-in Ethernet Open Settings correct?
3. Is the function block's connection number (CONN) correct?
4. Are the function block's RMC_FILE, RMC_ELEMENT, and NUM_DW inputs correct?

6.1 Event Log in RMCTools

In RMCTools, the primary troubleshooting tool for communications is the Event Log. You can view basic information for each read or write that is performed via the communications. You need to first enable this functionality in the Event Log as follows:

1. In RMCTools, in the Project pane, double-click **Event Log**.
2. In the Event Log toolbar, click the Properties button.
3. Expand **Communications** and click **10/100 Ethernet**.
4. Check the **All Applications Transactions** box, then click **OK**.



10/100 Ethernet Events

☑ Ethernet Link Up/Down
☑ IP Address Events
☑ TCP/IP Events
☑ Application Protocol Errors
☑ Application Connection Events
☑ All Application Transactions
□ Include RMCTools transactions

Ethernet I/O Logging: Requests

To exclude particular events from the event log clear the check box or set the combination box to "None"

5. The Event Log will now show all the Ethernet communications with to the RMC. For each transaction, the Event Log lists the source, the address, and the number of registers.
6. To pause the Event Log, in the Event Log toolbar, click the Pause button. To resume it, click the Resume button.

6.2 Advanced Troubleshooting

6.2.1 SP.SOCSEND and SP.SOCRCV Errors

The RMC Read and RMC Write function blocks use the SP.SOCSEND and SP.SOCRCV instructions.

The SP.SOCSEND and SP.SOCRCV instructions may return an error code in the SND_S2 label or RCV_S2 label in the function blocks. For more details, see the SP.SOCSEND and SP.SOCRCV instructions in the QnUCPU User's Manual Communication via Built-in Ethernet Port - SH(NA)-080811ENG.

To see the internal errors:

1. Open the function block program.
2. Register the SND_S2 label or RCV_S2 label to the Watch window. You will be able to see the errors if there are any.

6.2.2 DMCP Errors

The RMC Read and RMC Write function blocks use the SP.SOCSEND and SP.SOCRCV instructions to implement Delta's DMCP protocol.

The DMCP protocol itself may return an error that appears in the RCV_S3.RESP_CODE label. For more details, see the DMCP protocol (UDP) in the RMCTools help.

To see the internal errors:

1. Open the function block program.
2. Register the SND_S2 label or RCV_S2 label to the Watch window. You will be able to see the errors if there are any.

Revisions

June 2014 – Document Created and Published V1.0

September 2014 – Verified and updated by MEAU V1.10