MELSE Communication Driver

Driver for Ethernet and Serial communication with
Mitsubishi QnA and Q Series devices
using MELSEC Protocol

Contents

Introduction

The MELSE driver enables communication between the Studio system and some Mitsubishi devices (QnA and Q Series) using the MELSEC protocol over Ethernet and Serial, according to the specifications discussed in this document.

This document will help you to select, configure and execute the MELSE driver, and it is organized as follows:

- **Introduction**: This section, which provides an overview of the document.
- **General Information**: Identifies all of the hardware and software components required to implement communication between the Studio system and the target device.
- **Selecting the Driver**: Explains how to select the MELSE driver in the Studio system.
- **Configuring the Device**: Describes how the target device must be configured to receive communication from the MELSE driver.
- **Configuring the Driver**: Explains how to configure the MELSE driver in the Studio system, including how to associate database tags with device registers.
- **Executing the Driver**: Explains how to execute the MELSE driver during application runtime.
- **Troubleshooting**: Lists the most common errors for this driver, their probable causes, and basic procedures to resolve them.
- **Sample Application**: Explains how to use a sample application to test the MELSE driver configuration
- **Revision History**: Provides a log of all changes made to the driver and this documentation.

Notes:

- This document assumes that you have read the “Development Environment” chapter in Studio’s *Technical Reference Manual*.
- This document also assumes that you are familiar with the Microsoft Windows NT/2000/XP environment. If you are not familiar with Windows, then we suggest using the **Help** feature (available from the Windows desktop **Start** menu) as you work through this guide.

General Information

This chapter identifies all of the hardware and software components required to implement Ethernet and Serial communication between the MELSE driver in Studio and a target Mitsubishi QnA and Q Series devices.

The information is organized into the following sections:

- Device Specifications
- Network Specifications
- Driver Characteristics
- Conformance Testing

Device Specifications

To establish communication, your target device must meet the following specifications:

- **Manufacturer**: Mitsubishi
- **Compatible Equipment**: - Q Series
  - QnA Series
  - L Series

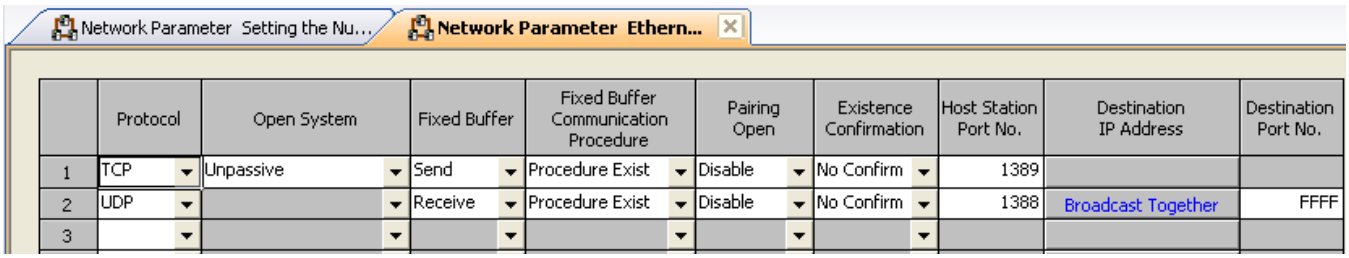
Network Specifications

To establish communication, your device network must meet the following specifications:

- **Device Communication Port**:
  - Ethernet Port (QJ71E71-100 module)
  - Serial Port (QJ71C24N module)
  - For L Series, use the Ethernet port embedded on the CPU
- **Physical Protocol**:
  - Ethernet
  - Serial
- **Logic Protocol**: MELSEC
- **Specific PC Board:**
  - For Ethernet: Any Ethernet Adapter (Ethernet board)
  - For Serial: Any Serial Port
- **Device Runtime Software**: None

Keep in mind that in order to communicate with the PLC over TCP/IP or UDP/IP, you must configure the Ethernet settings on the PLC using the Mitsubishi's programming software (GX Developer or GX Works2).

For TCP and UDP configuration on **Q-Series PLC**, it would be something like this:

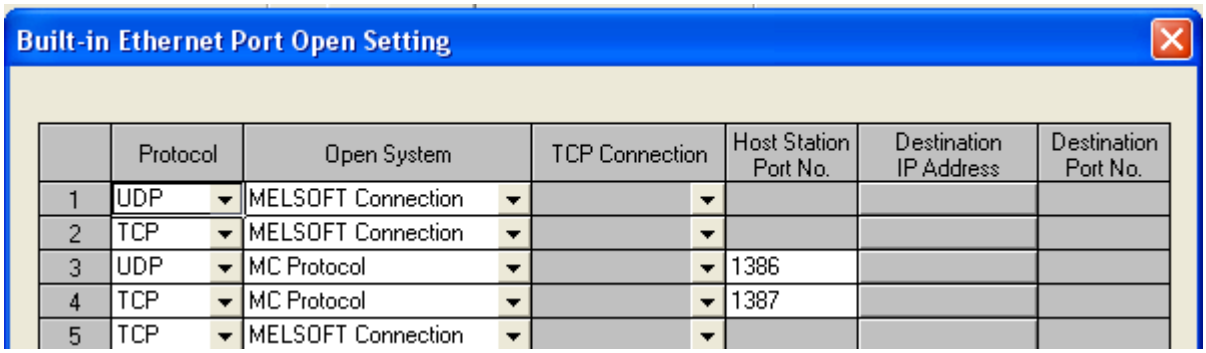


	Protocol	Open System	Fixed Buffer	Fixed Buffer Communication Procedure	Pairing Open	Existence Confirmation	Host Station Port No.	Destination IP Address	Destination Port No.
1	TCP	Unpassive	Send	Procedure Exist	Disable	No Confirm	1389		
2	UDP		Receive	Procedure Exist	Disable	No Confirm	1388	Broadcast Together	FFFF
3									

In this configuration, you will be able to communicate over TCP/IP using the port number 5001 (0x1389) and over UDP using the port number 5000.

> **Note:**
> It is necessary to configure the Destination IP Address when communicating over UDP/IP. You can configure it as 255.255.255.255 and it will communicate with any PLC that tries to communicate with it.

For the **L Series**, PLC, you need a similar configuration:



With the configuration above, you are able to communicate using UDP/IP on the port number 4998 and over TCP/IP using the port 4999

Driver Characteristics

The MELSE driver package consists of the following files, which are automatically installed in the **`\DRV`** subdirectory of Studio:

- **`MELSE.INI:`** Internal driver file. *You must not modify this file*.
- **`MELSE.MSG:`** Internal driver file containing error messages for each error code. *You must not modify this file*.
- **`MELSE.PDF:`** This document, which provides detailed information about the MELSE driver.
- **`MELSE.DLL:`** Compiled driver.

> **Note:**
> You must use Adobe Acrobat® Reader™ to view the **`MELSE.PDF`** document. You can install Acrobat Reader from the Studio installation CD, or you can download it from Adobe's Web site.

You can use the MELSE driver on the following operating systems:

- Windows XP/Vista/2003/7/2008
- Windows CE

The MELSE driver supports the following registers:

Registers Type	Length	Representation	Write	Read	Bit Access	Dword	String
SM (Special relay)	1 bit	Decimal	•	•	–	–	–
SD (Special register)	2 bytes	Decimal	•	•	•	•	•
X (Input relay)	1 bit	Hexadecimal	•	•	–	–	–
Y (Output relay)	1 bit	Hexadecimal	•	•	–	–	–
M (Internal relay)	1 bit	Decimal	•	•	–	–	–
L (Latch relay)	1 bit	Decimal	•	•	–	–	–
F (Annunciator)	1 bit	Decimal	•	•	–	–	–
V (Edge relay)	1 bit	Decimal	•	•	–	–	–
B (Link relay)	1 bit	Hexadecimal	•	•	–	–	–
D (Data register)	2 bytes	Decimal	•	•	•	•	•
W (Link register)	2 bytes	Hexadecimal	•	•	•	•	•
TS (Timer Contact)	1 bit	Decimal	•	•	–	–	–
TC (Timer Coil)	1 bit	Decimal	•	•	–	–	–
TN (Timer Current Value)	2 bytes	Decimal	•	•	•	•	•
SS (Retentive Timer Contact)	1 bit	Decimal	•	•	–	–	–
SC (Retentive Timer Coil)	1 bit	Decimal	•	•	–	–	–
SN (Retentive Timer Current Value)	2 bytes	Decimal	•	•	•	•	•
CS (Counter Contact)	1 bit	Decimal	•	•	–	–	–
CC (Counter Coil)	1 bit	Decimal	•	•	–	–	–
CN (Counter Current Value)	2 bytes	Decimal	•	•	•	•	•
SB (Special link relay)	1 bit	Hexadecimal	•	•	–	–	–
SW (Special link register)	2 bytes	Hexadecimal	•	•	•	•	•
S (Step relay)	1 bit	Decimal	•	•	–	–	–
DX (Direct input)	1 bit	Hexadecimal	•	•	–	–	–
DY (Direct Output)	1 bit	Hexadecimal	•	•	–	–	–
Z (Index register)	2 bytes	Decimal	•	•	•	•	•
R (File Register)	2 bytes	Decimal	•	•	•	•	•
ZR (File Register)	2 bytes	Hexadecimal	•	•	•	•	•

Conformance Testing

The following hardware/software was used for conformance testing:

- **Driver Configuration (Q Series PLC)**:
  - **Communication Type:** UDP/IP, TCP/IP and Serial
  - **UDP Port:** 5001
  - **TCP Port:** 5002
  - **Ethernet Cable**: Ethernet Cable
  - **Serial Communication Settings**
    - **Cable:** Null Modem Cable with Gender Adapter
    - **QJ71C24 Port:** CH1 (RS232)
    - **Comm Settings:** 9600, 8, 1, Odd
    - **Advanced Settings:** Control RTS: Always On

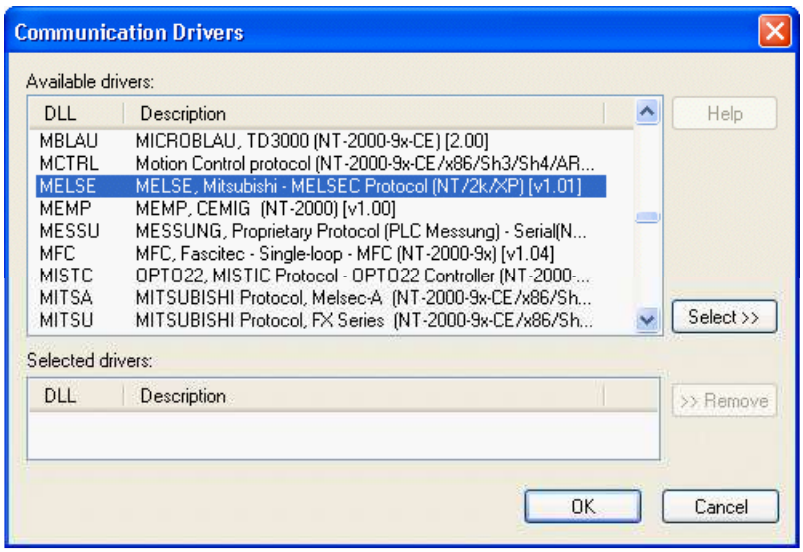
- **Driver Configuration (L Series PLC)**:
  - **Communication Type:** UDP/IP, TCP/IP
  - **UDP Port:** 4998
  - **TCP Port:** 4999
  - **Ethernet Cable**: Ethernet Cable

Driver Version	Studio Version	Operating System (development)	Operating System (runtime)	Equipment
10.3	v6.1 + SP5	WinXP + SP3	▪ WinXP + SP2 ▪ WinCEv5.00	- Q00JCPU with QJ71E71-100 module (Ethernet) - Q00JCPU with QJ71C24 module (Serial)
10.4	V7.0	WinXP + SP3	▪ WinXP + SP2 ▪ WinCEv5.00	- L26CPU-BT with Ethernet port embedded on the CPU
10.6	V8.0 + P3	Windows 8.1	▪ WinXP + SP2	- Q00JCPU with QJ71E71-100 module (Ethernet) - Q00JCPU with QJ71C24 module (Serial)
10.7	V8.0 SP2 P1	Windows 8.1	▪ Win 7	- Q00JCPU with QJ71E71-100 module (Ethernet)
10.8	V8.1 SP2	Windows 10	▪ WinXP +SP2 ▪ WinCEv5.00	- Q00JCPU with QJ71E71-100 module (Ethernet)
10.9	V8.2	Windows 10	▪ WinXP +SP2 ▪ WinCEv5.00 ▪ Raspberry Pi 3 ▪ Ubuntu 14.04	- Q00JCPU with QJ71E71-100 module (Ethernet)

Selecting the Driver

When you install Studio, all of the communication drivers are automatically installed in the **\DRV** subdirectory but they remain dormant until manually selected for specific applications. To select the MELSE driver for your Studio application:

1. From the main menu bar, select **Insert** → **Driver** to open the *Communication Drivers* dialog.
2. Select the **MELSE** driver from the *Available Drivers* list, and then click the **Select** button.



Communication Drivers Dialog

3. When the **MELSE** driver is displayed in the **Selected Drivers** list, click the **OK** button to close the dialog. The driver is added to the *Drivers* folder, in the *Comm* tab of the Workspace.

> **Note:**
> It is not necessary to install any other software on your computer to enable communication between Studio and your target device. However, this communication can only be used by the Studio application; it cannot be used to download control logic to the device. To download control logic to a Mitsubishi device, you must also install the Mitsubishi programming software, such as MELSEC GX Developer or GX Works2. For more information, please consult the documentation provided by the device manufacturer.

> **Attention:**
> For safety reasons, you must take special precautions when installing any physical hardware. Please consult the manufacturer's documentation for specific instructions.

Configuring the Driver

Once you have selected the MELSE driver in Studio, you must properly configure it to communicate with your target device. First, you must set the driver's communication settings to match the parameters set on the device. Then, you must build driver worksheets to associate database tags in your Studio application with the appropriate addresses (registers) on the device.

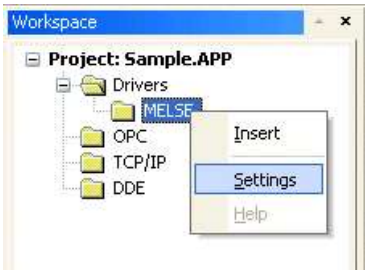
Configuring the Communication Settings

The communication settings are described in detail in the "Communication" chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

For the purposes of this document, only MELSE driver-specific settings and procedures will be discussed here. To configure the communication settings for the MELSE driver:

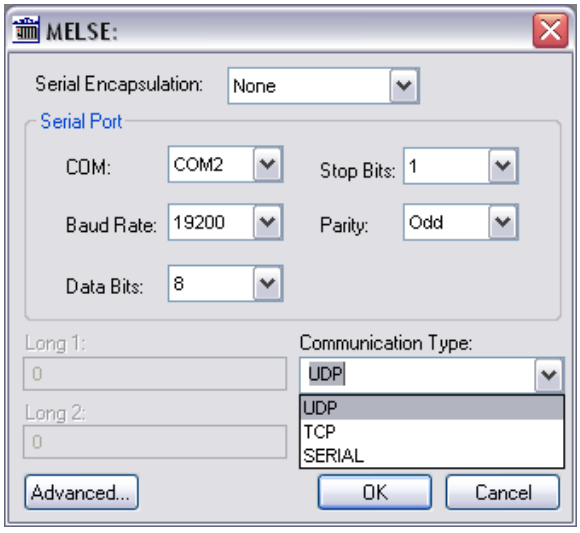
1. In the *Workspace* pane, select the *Comm* tab and then expand the *Drivers* folder. The MELSE driver is listed here as a subfolder.

2. Right-click on the *MELSE* subfolder and then select the **Settings** option from the pop-up menu:



Select Settings from the Pop-Up Menu

The *MELSE: Communication Settings* dialog is displayed:



MELSE: Communication Settings Dialog

3. In the *Communication Settings* dialog, configure the driver settings to enable communication with your target device. To ensure error-free communication, the driver settings must *exactly match* the corresponding settings on the device. Please consult the manufacturer's documentation for instructions how to configure the device and for complete descriptions of the settings.

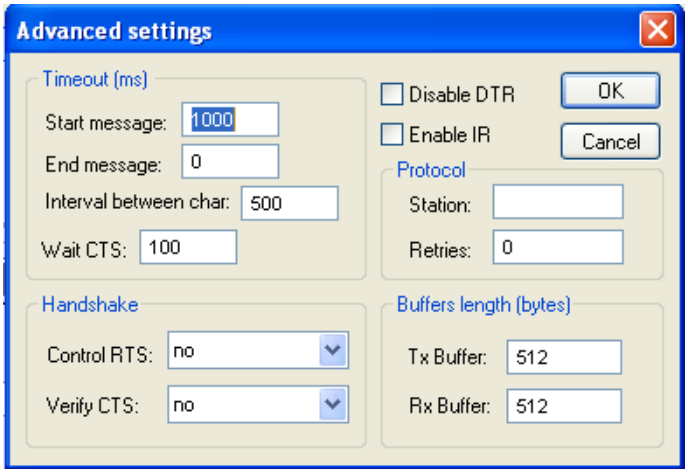
   Depending on your circumstances, you may need to configure the driver *before* you have configured your target device. If this is the case, then take note of the driver settings and have them ready when you later configure the device.

   > ➲ **Attention:**
   > For safety reasons, you **must** take special precautions when connecting and configuring new equipment. Please consult the manufacturer's documentation for specific instructions.

   The communication settings and their possible values are described in the following table:

Parameters	Default Values	Valid Values	Description
Communication Type	**UDP**	**UDP** **TCP** **SERIAL**	The protocols used for communication: ▪ **UDP** – UDP/IP type. ▪ **TCP** – TCP/IP type. ▪ **SERIAL** – Serial type.

4. If you are using a Data Communication Equipment (DCE) converter (e.g., 232/485) between your PC and your target device, then you must also adjust the **Control RTS** (Request to Send) setting to account for the converter. In the *Communication Settings* dialog, click the **Advanced** button to open the *Advanced Settings* dialog:



Advanced Settings Dialog

When the dialog is displayed, configure the **Control RTS** setting using the following information:

Setting	Default	Values	Description
Control RTS	no	no	Do not set the RTS (Request to Send) handshake signal. IMPORTANT: If you are using Windows 95/98 or Windows CE with the correct RS232/RS485 adapter (i.e. without RTS control), then you must select this option.
		Yes	Set the RTS (Request to Send) handshake signal before communication. IMPORTANT: If you are using Windows NT and the Cutler-Hammer RS232/RS485 adapter, then you must select this option.
		Yes+echo	Set the RTS (Request to Send) handshake signal before communication, and echo the signal received from the target device.
		Always on	The RTS (Request to Send) handshake signal is always set

➲ **Attention**:
If you incorrectly configure the **Control RTS** setting, then runtime communication will fail and the driver will generate a –15 error. See “Troubleshooting” for more information.

If you are communicating serially with a QJ71C24N, you may need to change the **Control RTS** to **Always On.**

Other than that, you do not need to change any other advanced settings at this time. You can consult the Studio *Technical Reference Manual* later for more information about configuring these settings.

5. Click **OK** to close the *Advanced Settings* dialog, and then click **OK** to close the *Communication Settings* dialog.

Configuring the Driver Worksheets

A selected driver includes one or more driver worksheets, which are used to associate database tags in Studio with registers on the target device. Each worksheet is triggered by specific application behavior, so that the tags / registers defined on that worksheet are scanned only when necessary – that is, only when the application is doing something that requires reading from or writing to those specific tags / registers. Doing this optimizes communication and improves system performance.

The configuration of these worksheets is described in detail in the "Communication" chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

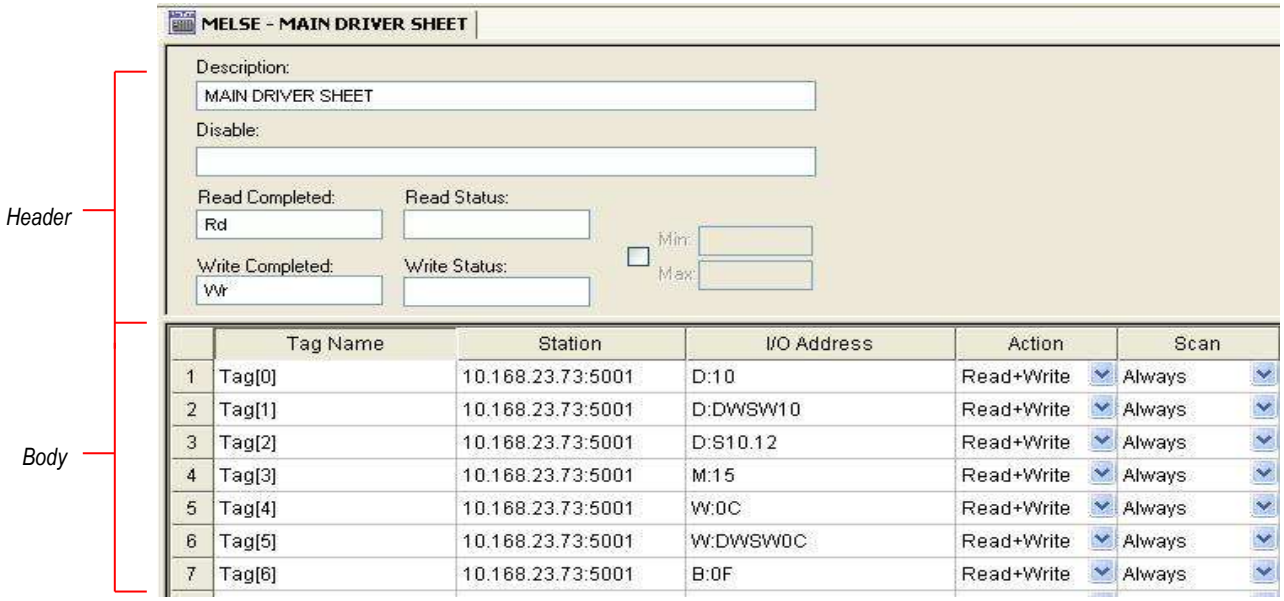
MAIN DRIVER SHEET

When you select the MELSE driver and add it to your application, Studio automatically inserts the *Main Driver Sheet* in the *MELSE* driver subfolder. To configure the Main Driver Sheet:

1. Select the *Comm* tab in the *Workspace* pane.
2. Open the *Drivers* folder, and then open the *MELSE* subfolder:

Main Driver Sheet in the MELSE Subfolder

3. Double-click on the **MAIN DRIVER SHEET** icon to open the following worksheet:



Opening the Main Driver Sheet

Most of the fields on this sheet are standard for all drivers; see the "Communication" chapter of the *Technical Reference Manual* for more information on configuring these fields. However, the **Station** and **I/O Address** fields use syntax that is specific to the MELSE driver.

4. For each table row (i.e., each tag/register association), configure the **Station** and **I/O Address** fields as follows:

   - **Station** field: Identify the target device, using the following syntax:

     - **For Ethernet :**

     <IP Address>:<Port Number>

     Example — **10.168.23.73:5001**

     - **For Serial :**

     <Station Number>

     Example — **0**

     Where:

     <IP Address> is the device's IP address on the Ethernet network.

     <Port Number> is the port number for the MELSEC protocol.

     <Station Number> the station number of the PLC.

     You can also specify an indirect tag (e.g. **{station}**), but the tag that is referenced must follow the same syntax and contain a valid value.

     > ➲ **Attention:**
     > You cannot leave the **Station** field blank.

   - **I/O Address:** Specify the address of the associated device register.

   For all non-Strings:

   <Type>:[Format]<AddressOffset>.[Bit]

   Examples — **B:5**, **W:0C**, **W:S0C.0A**, **D:DW10**

   For Strings:

   <Type>:[Format]<AddressOffset>.<Length>

   Examples — **D:S5.16**

   Where:

   - ***<Type>*** is the register type, which must be one of the types listed; and

   – ***[Format]*** is an optional parameter that defines the format of the register. If you leave this parameter blank, then Studio provides the data in Word format. Otherwise, the options for this parameter are:

     - **S**: String value: Reads and writes String values using consecutive Words.

- **UB**: Unsigned 8 bits variable (byte)
- **W**: Signed 16 bits variable (word)
- **SW**: Signed 16 bits variable (word) with byte swap
- **UW**: Unsigned 16 bits variable (word)
- **DW**: Signed 32 bits variable (double word)
- **SDW**: Signed 32 bits variable (double word) with byte swap
- **UDW:** Unsigned 32 bits variable (double word)
- **F:** 32 bits float points (float)
- **SBCD:** 16 bits BCD value with byte swap
- **SBCDD:** 32 bits BCD value with byte swap

– ***<AddressOffset>*** is the specific address of the register on the device; and

– ***[Bit]*** is an optional parameter used only for the default Word format, to indicate which bit in the Word will be read from / written to; and

– ***<Length>*** is a parameter used only for the String format, to indicate the length of the String in bytes.

> **Note:**
> - See the supported registers in the Driver Characteristics session.

For examples of how register addresses are composed using the values in the **I/O Address** field, see the following table:

Register Address in the Device	Representation	I/O Address
D10	Decimal (Word)	D:10
		D:DW10
		D:S10.12
M15	Decimal (Bit)	M:15
W0C	Hexadecimal (Word)	W:0C
		W:DW0C
		W:S0C.0A
B0F	Hexadecimal (Bit)	B:0F

For more information about registers and addressing, please consult the manufacturer's documentation.

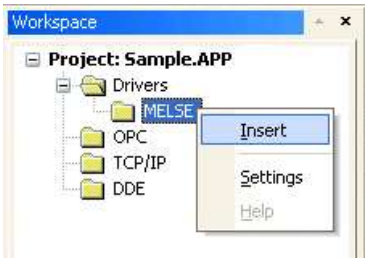
STANDARD DRIVER WORKSHEET

When you select the MELSE driver and add it to your application, it has only a Main Driver Sheet by default (see previous section). However, you may insert additional Standard Driver Worksheets to define tag/register associations that are triggered by specific application behaviors. Doing this will optimize communication and improve system performance by ensuring that tags/registers are scanned only when necessary – that is, only when the application is performing an action that requires reading or writing to those specific tags/registers.

> **Note:**
> We recommend configuring device registers in sequential blocks in order to maximize performance.

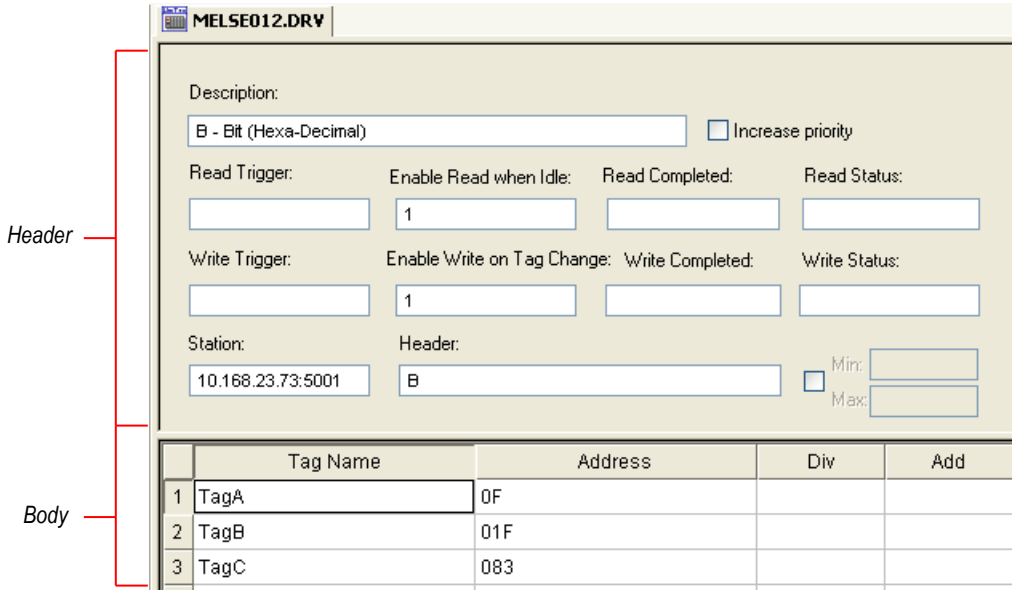
To insert a new driver worksheet:

1. In the *Comm* tab, open the *Drivers* folder and locate the *MELSE* subfolder.
2. Right-click on the *MELSE* subfolder, and then select **Insert** from the pop-up menu:



Inserting a New Worksheet

A new MELSE driver worksheet is inserted into the *MELSE* subfolder, and the worksheet is opened for configuration:



MELSE Driver Worksheet

> **Note:**
> Worksheets are numbered in order of creation, so the first worksheet is **`MELSE001.drv`**.

Most of the fields on this worksheet are standard for all drivers; see the "Communication" chapter of the *Technical Reference Manual* for more information on configuring these fields. However, the **Station**, **Header**, and **Address** fields use syntax that is specific to the MELSE driver.

3. Configure the **Station** and **Header** fields as follows:

   - **Station** field: Identify the target device, using the following syntax:

     - **For Ethernet :**

     `<IP Address>:<Port Number>`

     Example — **`10.168.23.73:5001`**

     - **For Serial :**

     `<Station Number>`

     Example — **`0`**

     Where:

     `<IP Address>` is the device's IP address on the Ethernet network.

     `<Port Number>` is the port number for the *MELSE* protocol.

     `<Station Number>` the station number of the PLC.

     You can also specify an indirect tag (e.g. **`{station}`**), but the tag that is referenced must follow the same syntax and contain a valid value.

   - **Header** field: Specify the address of the first register in a block of registers on the target device. The addresses declared in the *Body* of the worksheet are simply offsets of this **Header** address. When Read/Write operations are executed for the entire worksheet (see **Read Trigger** and **Write Trigger** above), it scans the entire block of registers from the first address to the last.

     The **Header** field uses the following syntax:

     `<Type>:<AddressReference>`

     Examples — **`B:0`**, **`D:10`**, **`M:15`**

     Where:

     – ***`<Type>`*** is the register type, which must be one of the types listed; and

     – ***`<AddressReference>`*** is the initial address (reference) of the specified register type.

After you edit the **Header** field, Studio checks the syntax to determine if it is valid. If no value is entered, then the default type is **D:0**.

You can also specify an indirect tag (e.g. **{header}**), but the tag that is referenced must follow the same syntax and contain a valid value.

Attention:

- You cannot leave the **Station** and **Header** fields blank; you must specify some values.

Note:

- See the supported registers in the Driver Characteristics session.

4. For each table row (i.e., each tag/register association), configure the **Address** field using the following syntax…

   For all non-Strings:

   [Format]<AddressOffset>.[Bit]

   Examples — **5**, **10.4**, **0C**, **DW10**

   For Strings:

   [Format]<AddressOffset>.<Length>

   Examples — **S5.16**

   Where:

   – ***[Format]*** is an optional parameter that defines the format of the register. If you leave this parameter blank, then Studio provides the data in Word format. Otherwise, the options for this parameter are:

      - **S**: String value: Reads and writes String values using consecutive Words.
      - **UB**: Unsigned 8 bits variable (byte)
      - **W**: Signed 16 bits variable (word)
      - **SW**: Signed 16 bits variable (word) with byte swap
      - **UW**: Unsigned 16 bits variable (word)
      - **DW**: Signed 32 bits variable (double word)
      - **SDW**: Signed 32 bits variable (double word) with byte swap
      - **UDW:** Unsigned 32 bits variable (double word)
      - **F:** 32 bits float points (float)
      - **SBCD:** 16 bits BCD value with byte swap
      - **SBCDD:** 32 bits BCD value with byte swap

   – ***<AddressOffset>*** is a value that is added to the ***<AddressReference>*** parameter of the **Header**, to compose the specific address of the register on the device; and

   – ***[Bit]*** is an optional parameter used only for the default Word format, to indicate which bit in the Word will be read from / written to; and

   – ***<Length>*** is a parameter used only for the String format, to indicate the length of the String in bytes.

For examples of how register addresses are composed using the values in the **Header** and **Address** fields, see the following table:

Register Address in the Device	Representation	Header Field	Address Field
D10	Decimal (Word)	D:0	10
		D:0	DW10
		D:0	S10.12
M15	Decimal (Bit)	M:0	15
W0C	Hexadecimal (Word)	W:0	0C
		W:0	DW0C
		W:0	S0C.0A
B0F	Hexadecimal (Bit)	B:0	0F

For more information about registers and addressing, please consult the manufacturer's documentation.

➲ **Attention:**

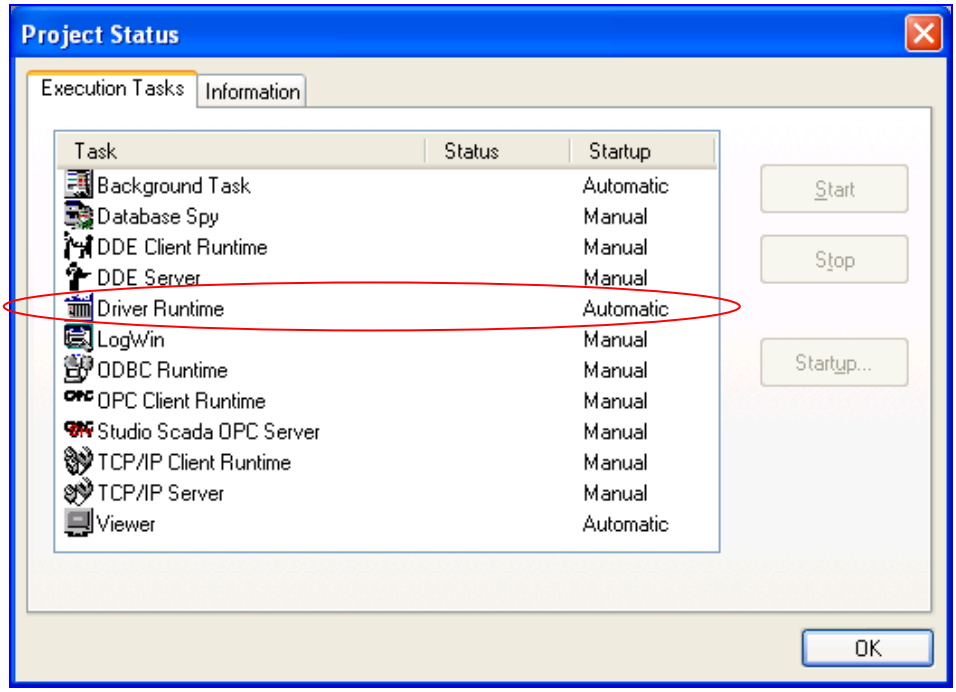
You must not configure a range of addresses greater than the maximum block size (data buffer length) supported by the protocol within the same worksheet. The maximum data buffer length for this driver is 500 bytes per *Standard Driver* worksheet.

Executing the Driver

By default, Studio will automatically execute your selected communication driver(s) during application runtime. However, you may verify your application's runtime execution settings by checking the *Project Status* dialog.

To verify that the the communication driver(s) will execute correctly:

1. From the main menu bar, select **Project** → **Status**. The *Project Status* dialog displays:



Project Status Dialog

2. Verify that the *Driver Runtime* task is set to **Automatic**.
   - If the setting is correct, then proceed to step 3 below.
   - If the **Driver Runtime** task is set to **Manual**, then select the task and click the **Startup** button to toggle the task's *Startup* mode to **Automatic**.
3. Click **OK** to close the *Project Status* dialog.
4. Start the application to run the driver.

Troubleshooting

If the MELSE driver fails to communicate with the target device, then the database tag(s) that you configured for the **Read Status** or **Write Status** fields of the Main Driver Sheet will receive an error code. Use this error code and the following table to identify what kind of failure occurred.

Error Code	Description	Possible Causes	Procedure to Solve
0	OK	Communication without problems	None
1	ERROR CONNECT	IP address or Port number invalid	Check the IP Address or the Port number
2	PLC Error or invalid memory address	You may have tried to Read a memory from the PLC (Device) that is not present on that CPU	Check if the address (Device) configured does exist in the PLC.
3	CHECK-SUM ERROR	The check sum byte received is not equal to the expected byte.	Contact your Studio technical support representative
4	PROTOCOL ERROR	The driver received an unexpected message from the device.	Contact your Studio technical support representative
-39	INVALID BLOCK SIZE	Worksheet is configured with range of addresses greater than the maximum block size.	Check what the maximum block size is and configure your worksheet right.
-37	INVALID HEADER	The header field specified in the driver worksheet is invalid	Check the supported header fields and use one of the supported headers
-39	INVALID ADDRESS	The address field specified is invalid or out of the supported range	Check the supported ranges and specify an address within the supported range.
-15	Timeout waiting start a message.	▪ Disconnected cables ▪ PLC turned off, or in Stop or error mode ▪ Wrong Station number, ▪ Wrong RTS/CTS control settings	▪ Check the cable wiring ▪ Check the Station value ▪ Check the right configuration. Review the Communication Parameters section for valid RTS/CTS configurations.
-17	Timeout between rx characters.	▪ PLC in stop or error mode ▪ Wrong station number ▪ Wrong parity ▪ Wrong RTS/CTS configuration settings	▪ Check the cable wiring ▪ Check the PLC stat e (it must be RUN) ▪ Check the station number ▪ Check the configuration. Review the Communication Parameters section for valid RTS/CTS configurations.

⇨ **Tip:**

You can monitor communication status by establishing an event log in Studio's *Output* window (*LogWin* module). To establish a log for **Field Read Commands**, **Field Write Commands** and **Serial Communication,** right-click in the *Output* window and select the desired options from the pop-up menu.

You can also use the *LogWin* module (**Tools** → **LogWin**) to establish an event log on a remote unit that runs Windows CE. The log is saved on the unit in the **`celog.txt`** file, which can be downloaded later.

If you are unable to establish communication between Studio and the target device, then try instead to establish communication using the device's own programming software. Quite often, communication is interrupted by a hardware or cable problem or by a device configuration error. If you can successfully communicate using the programming software, then recheck the driver's communication settings in Studio.

To test communication between Studio and the device, we recommend using the sample application provided rather than your new application.

If you must contact us for technical support, please have the following information available:

- **Operating System** (type and version): To find this information, select **Tools** → **System Information**.
- **Project Information**: To find this information, select **Project** → **Status.**
- **Driver Version** and **Communication Log**: Displays in the Studio *Output* window when the driver is running.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

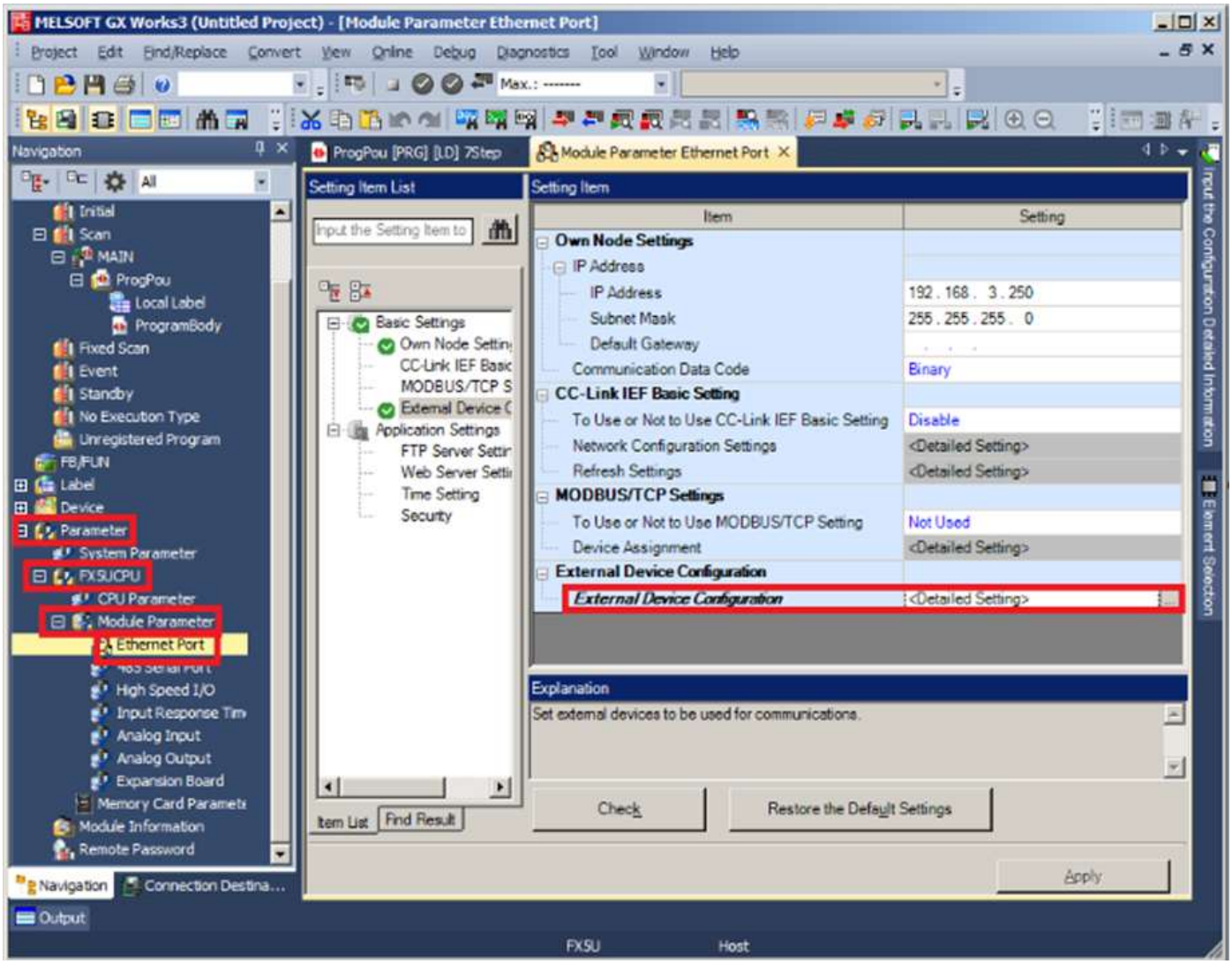
Sample Application

There was not an official sample application available for this driver by the timer that this document was written.

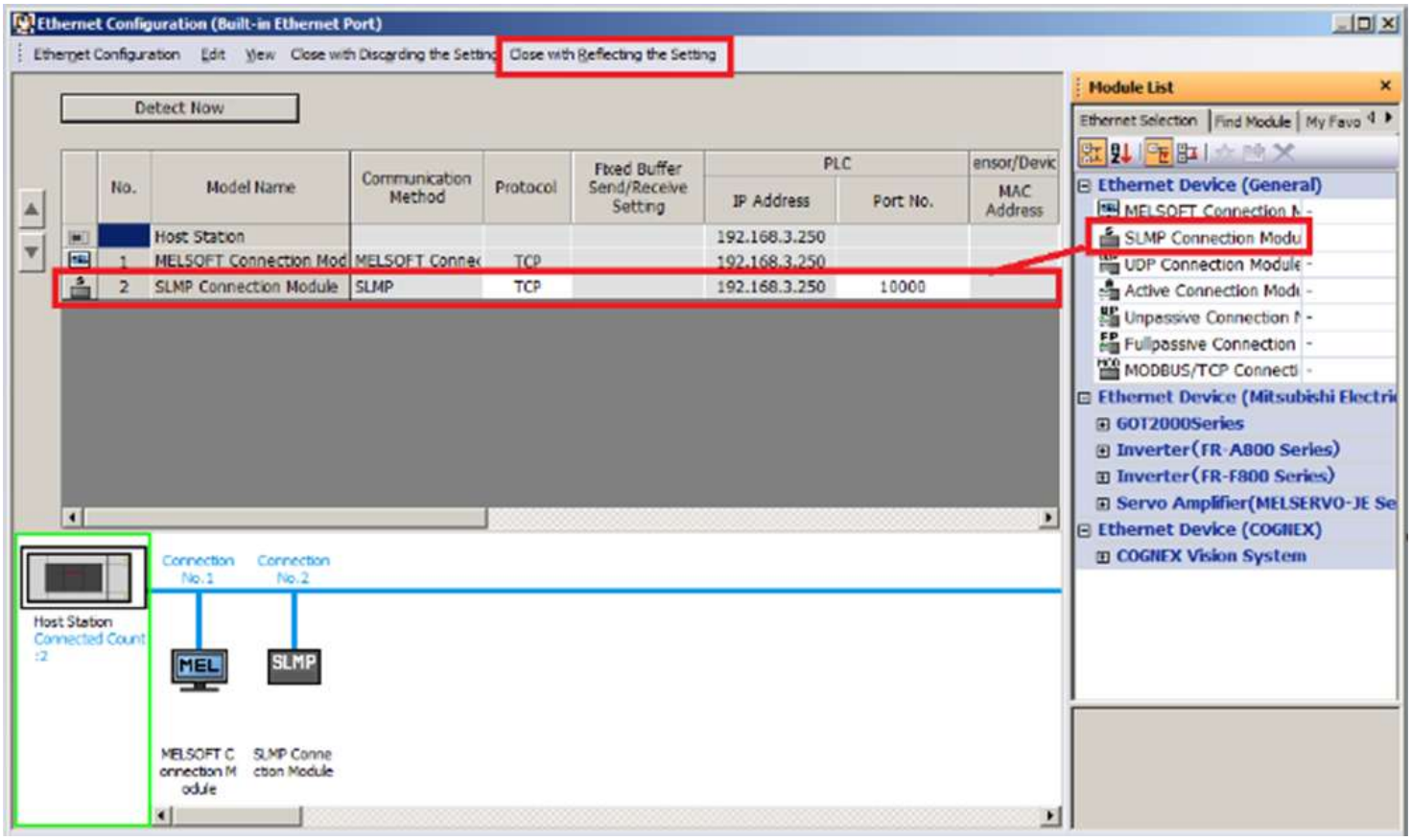
Appendix: Configuring FX5U PLCs to be able to communicate with the driver

Using the GX Works 3

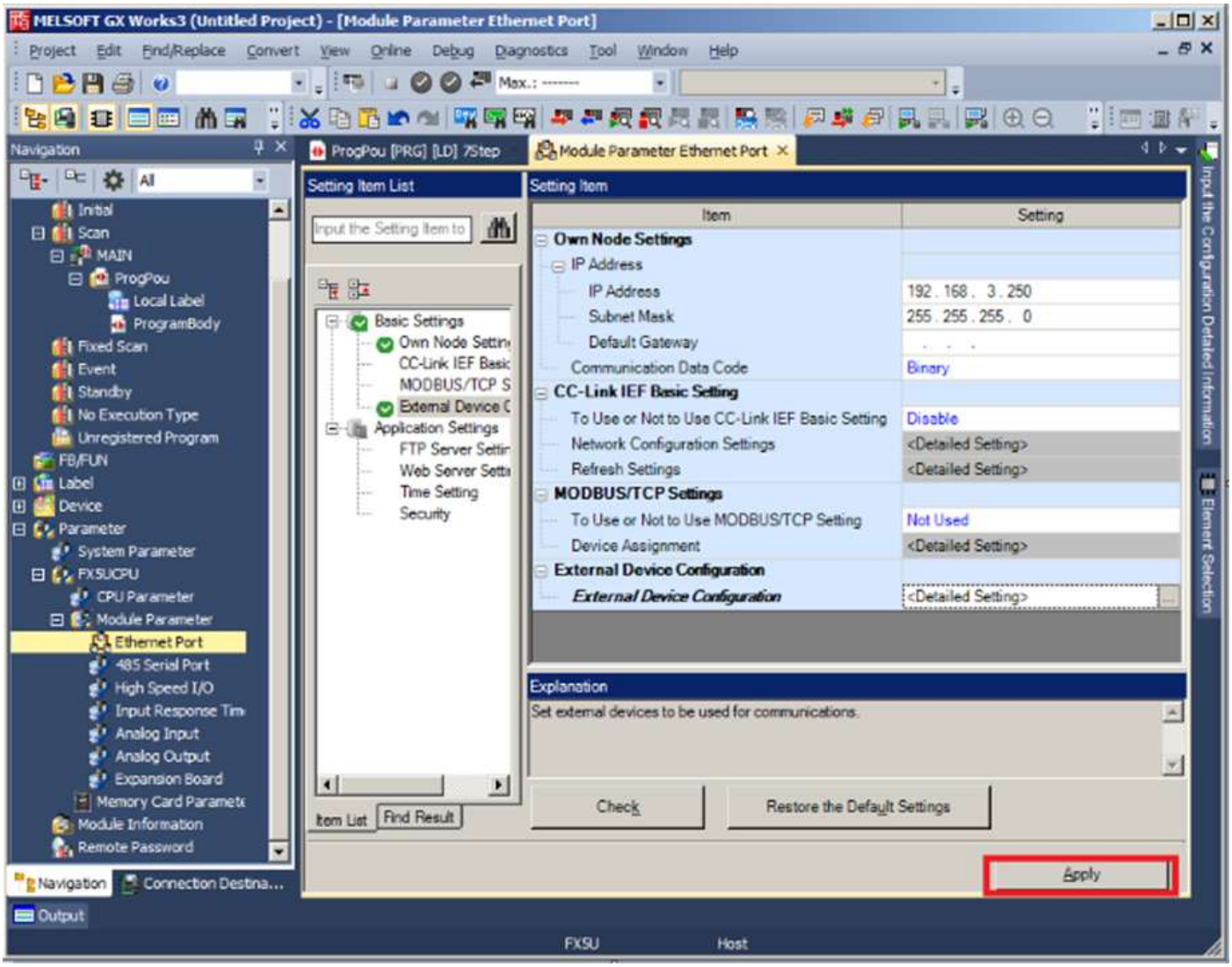
1. Go to Parameter-FX5UCPU-Module Parameter-Ethernet Port;
2. Double click External Device Configuration



3. On the upcoming screen, add a SLMP Connection with TCP Protocol and desired port number and then select “Close with Reflecting the Setting” as follows:


Ethernet Configuration (Built-in Ethernet Port)
Ethernet Configuration
Edit
View
Close with Discarding the Setting
Close with Reflecting the Setting
Detect Now
No.
Model Name
Communication Method
Protocol
Fixed Buffer Send/Receive Setting
PLC
IP Address
Port No.
MAC Address
Host Station
192.168.3.250
1
MELSOFT Connection Mod
MELSOFT Connec
TCP
192.168.3.250
2
SLMP Connection Module
SLMP
TCP
192.168.3.250
10000
Module List
Ethernet Selection
Find Module
My Favo
Ethernet Device (General)
MELSOFT Connection M
SLMP Connection Modu
UDP Connection Module
Active Connection Modu
Unpassive Connection M
Fullpassive Connection
MODBUS/TCP Connecti
Ethernet Device (Mitsubishi Electric
GOT2000Series
Inverter(FR-A800 Series)
Inverter(FR-F800 Series)
Servo Amplifier(MELSERVO-JE Se
Ethernet Device (COGNEX)
COGNEX Vision System
Connection No.1
Connection No.2
Host Station
Connected Count :2
MEL
SLMP
MELSOFT Connection Module
SLMP Connection Module

4. When the screen closes select APPLY for the Module Parameter Ethernet Port Screen configuration:



5. After this configuration transfer the FULL project and hard reset the CPU.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of changes
A	1.01	Eric Vigiani	24-Jun-2004	First driver version.
B	1.01	Fabio Carvalho	12-Aug-2005	Implemented new commands to fit Q series.
C	1.02	Eric Vigiani	06-Sep-2006	Reviewed and Tested the driver with the Q and QnA series.
D	1.02	Michael D. Hayden	09-Nov-2006	Edited for language and usability.
E	1.03	Eric Vigiani Graziane C. Forti	20-Feb-2008	▪ Fixed Block Size problem with 2 Bytes operands. ▪ Fixed problem with Hexadecimal address. ▪ Fixed problem to write bit. ▪ Implemented UB, W, DW, UDW, F, SBCD, SBCDD and S data types. ▪ Inserted Main Driver Work Sheet. ▪ Fixed problem with last string address.
F	1.04	Rafael R. Fernandes	25-Apr-2008	▪ Implemented Serial Communication option .
G	1.05	Eric Vigiani	15-Jun-2008	▪ Fixed some problem with Serial Communication
H	10.01	Lourenço Teodoro	15-Dec-2008	▪ Updated driver version, no changes in the contents.
I	10.1	Marcelo Carvalho	07-Jan-2009	▪ Updated driver version, no changes in the contents.
J	10.3	Fellipe Peternella Eric Vigiani Lourenço Teodoro	30-Jul-2009	▪ Fixed problems when reading Strings ▪ Fixed problems with DLE byte ▪ Improved the error codes ▪ Fixed problem with buffer overrun ▪ Changed the driver to assume UDP as default connection. If no selection was made the driver would simply not work.
K	10.3	Andre Bastos	6-Jun-2011	▪ Updated Documentation only to include support to the “L” series PLC. No changed in the driver functionalities
L	10.4	Anushree Phanse	3-Oct-2015	▪ Fixed error related to invalid block size when hex addresses are used.
M	10.5	Lourenço Teodoro	07-Jul-2015	▪ Fixed issue to reconnect when using TCP/IP
N	10.6	Andre Korbes Anushree Phanse	31-May-2016	▪ Fix to prevent unnecessary blocks. ▪ Fixed issue with Q-series PLC where after program download in runtime or when experiencing timeouts, the driver updates wrong values on items.
O	10.7	Eduardo Castro	29-Dec-2016	▪ Fixed issue with reading signed double words
P	10.7	Anushree Phanse	30-Nov-2017	▪ Documentation updated with information about communicating with FX5U PLCs.
Q	10.8	Anushree Phanse	07-Nov-2018	▪ Added support for group write for OI MELSEC
R	10.9	Anushree Phanse	26-Dec-2018	▪ Added support for linux runtime when using TCP/IP