

MITSU Communication Driver

Driver for Serial and TCP/IP Communication
with Mitsubishi FX Series Devices

Contents

INTRODUCTION2

GENERAL INFORMATION3

 DEVICE SPECIFICATIONS3

 NETWORK SPECIFICATIONS3

 DRIVER CHARACTERISTICS3

 CONFORMANCE TESTING5

SELECTING THE DRIVER6

CONFIGURING THE SERIAL DEVICE7

CONFIGURING THE DRIVER7

 CONFIGURING THE COMMUNICATION SETTINGS7

 CONFIGURING THE DRIVER WORKSHEETS 10

EXECUTING THE DRIVER 22

TROUBLESHOOTING 23

SAMPLE APPLICATION 26

REVISION HISTORY 27

Introduction

The MITSU driver enables communication between the Studio system and some Mitsubishi FX Series devices, according to the specifications discussed in this document.

This document will help you to select, configure and execute the MITSU 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 MITSU driver in the Studio system.
- **Configuring the Device:** Describes how the target device must be configured to receive communication from the MITSU driver.
- **Configuring the Driver:** Explains how to configure the MITSU driver in the Studio system, including how to associate database tags with device registers.
- **Executing the Driver:** Explains how to execute the MITSU 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 MITSU 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 communication between the MITSU driver in Studio and a target device.

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:**
 - FX Series programmable controller with Monitor Interface FX-232AW
 - FX3G/3GE programmable controller with Ethernet Adapter FX3U-ENET-ADP
- **Programmer Software:**
 - For FX-232AW - MELSOFT - GX Developer.
 - For FX3U-ENET-ADP – MELSOFT - GX Works2

For a description of the device(s) used to test driver conformance, see “Conformance Testing” below.

Network Specifications

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

- **Physical Protocol:** RS232/422 Serial or Ethernet TCP/IP
- **Logic Protocol:** Host protocol
- **Device Runtime Software:** None
- **Specific Interface:** FX-232AW & FX3U-ENET-ADP interfaces
- **Cable:** Mitsubishi RS20P-CADP/RS422AW

Driver Characteristics

The MITSU driver package consists of the following files, which are automatically installed in the **/DRV** subdirectory of Studio:

- **MITSU.INI:** Internal driver file. *You must not modify this file.*
- **MITSU.MSG:** Error messages for each error code. *You must not modify this file.*
- **MITSU.PDF:** Document providing detailed information about the MITSU driver.
- **MITSU.DLL:** Compiled driver.

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

- Windows NT/2000/XP/7/8
- Windows CE

For a description of the operating systems used to test driver conformance, see “Conformance Testing” below.

The MITSU driver supports the following registers for FX Series PLC with Monitor Interface FX-232AW:

Register Type	Length	Read / Write	Accept Bit	Byte	Word	Double Word	Float	String
X (Inputs)	1 Bit	R / W	-	-	-	-	-	-
Y (Outputs)	1 Bit	R / W	-	-	-	-	-	-
M (Auxiliary Relays)	1 Bit	R / W	-	-	-	-	-	-
SM (Special Auxiliary Relays)	1 Bit	R / W	-	-	-	-	-	-
S (States)	1 Bit	R / W	-	-	-	-	-	-
TS (Timer Contacts)	1 Bit	R / W	-	-	-	-	-	-
CS (Counter Contacts)	1 Bit	R / W	-	-	-	-	-	-
TO (Timer Output)	1 Bit	R	-	-	-	-	-	-
CO (Counter Output)	1 Bit	R	-	-	-	-	-	-
TR (Timer Reset)	1 Bit	R	-	-	-	-	-	-
CR (Counter Reset)	1 Bit	R	-	-	-	-	-	-
T (Timer Value)	2 Bytes	R / W	R	R / W	R / W	R / W	R / W	R / W
C16 (Counter Value)	2 Bytes	R / W	R	R / W	R / W	R / W	R / W	R / W
D (Data Registers)	2 Bytes	R / W	R	R / W	R / W	R / W	R / W	R / W
SD (Special Data Registers)	2 Bytes	R / W	R	R / W	R / W	R / W	R / W	R / W
C32 (32 Bit Counter Value)	4 Bytes	R / W	R	R / W	R / W	R / W	R / W	R / W

R = Read only
 R / W = Read and Write

The MITSU driver supports the following registers for FX3U Series PLC with Ethernet Adapter FX3U-ENET-ADP:

Register Type	Length	Read / Write	Accept Bit	Word	Double Word	Float	String
X (Inputs)	1 Bit	R / W	-	-	-	-	-
Y (Outputs)	1 Bit	R / W	-	-	-	-	-
M (Auxiliary Relays)	1 Bit	R / W	-	-	-	-	-
S (States)	1 Bit	R / W	-	-	-	-	-
TS (Timer Contacts)	1 Bit	R / W	-	-	-	-	-
CS (Counter Contacts)	1 Bit	R / W	-	-	-	-	-
TN (Timer Current Value)	2 Bytes	R / W	R	R / W	R / W	-	R / W
CN (Counter Current Value)	2 Bytes	R / W	R	R / W	R / W	-	R / W
D (Data Registers)	2 Bytes	R / W	R	R / W	R / W	R / W	R / W
R (Extension Registers)	4 Bytes	R / W	R	R / W	R / W	R / W	R / W

R = Read only
 R / W = Read and Write

Conformance Testing

The following hardware/software were used for conformance testing for the FX-232AW and FX3U-ENET-ADP:

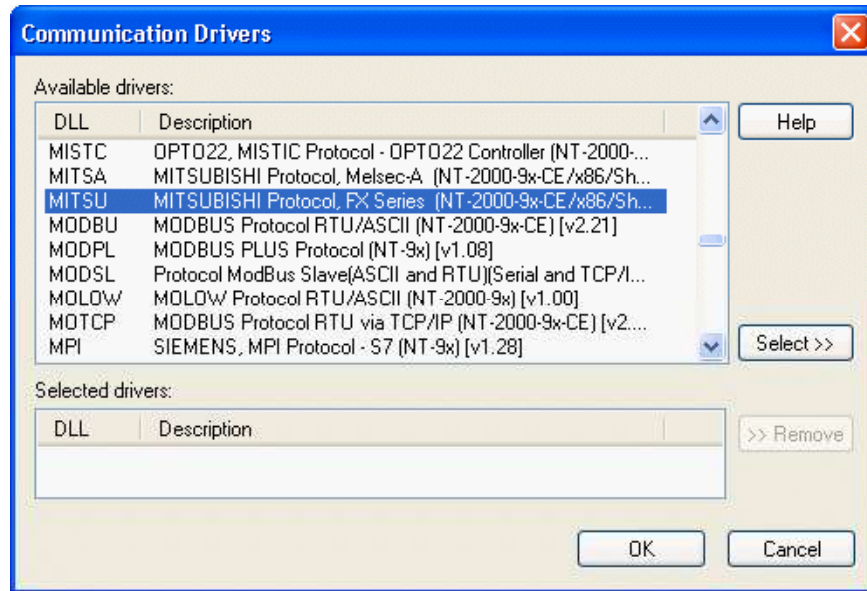
- **Configuration:**
 Serial:
 - **Baud Rate:** 9600
 - **Data Bits:** 7
 - **Stop Bits:** 1
 - **Parity:** Even
 - **COM Port:** COM1
 Ethernet
 - **Cable:** See “Network Specifications” above.

Driver Version	Studio Version	Operating System (development)	Operating System (target)	Equipment
10.7	7.1 + SP3	Windows XP/7/8	Windows XP/7/8 Windows CE 6.00	FX2N – 16MR FX3U – 16M FX3G/FX3GC

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 MITSU driver for your Studio application:

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



Communication Drivers Dialog

3. When the **MITSU** 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 FX Series device, you must also install the programming software (e.g., GX-Developer/GX Works 2 for the FX3U protocol). 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 Serial Device

We suggest using the following default configuration:

- **Baud Rate:** 9600
- **Data Bits:** 7
- **Stop Bits:** 1
- **Parity:** Even

Configuring the Driver

After opening Studio and selecting the **MITSU** driver, you must configure the driver. Configuring the MITSU driver is done in two parts:

- Specifying communication parameters (only one configuration needed).
- Defining communication tags and controls in the Communication tables or *Driver* worksheets (Standard and Main Driver Worksheets).

Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the **Station**, **Header** and **Address** fields, which are driver-specific. This document explains how to configure the **Station**, **Header** and **Address** fields only.

 **Note:**

For a detailed description of the Studio Standard and Main Driver Worksheets, and information about configuring the standard fields, review the product's *Technical Reference Manual*.

Once you have selected the MITSU 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.

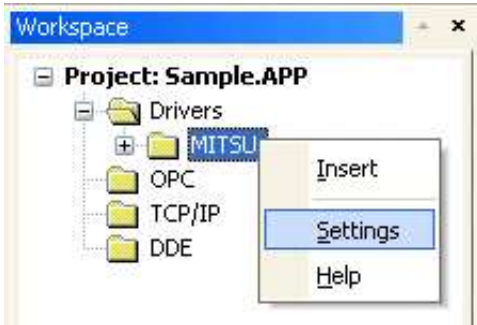
Configuring the Communication Settings

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.

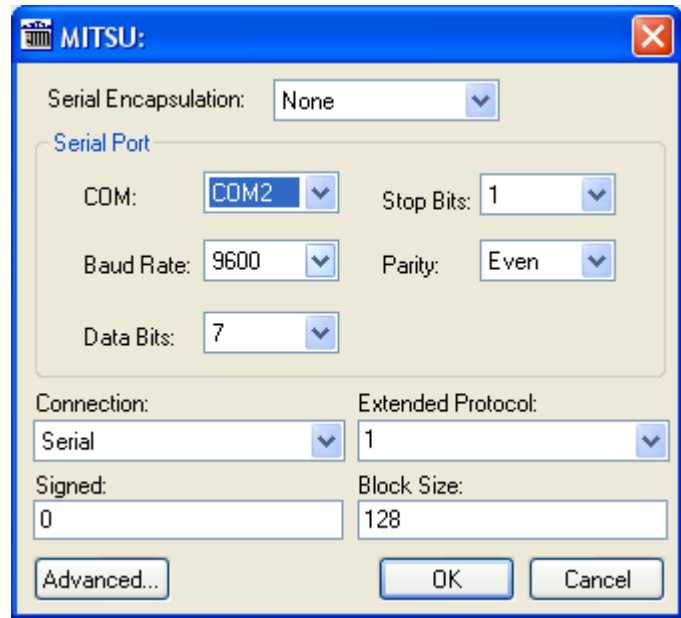
For the purposes of this document, only MITSU driver-specific settings and procedures will be discussed here. To configure the communication settings for the MITSU driver:

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

- Right-click on the *MITSU* subfolder and then select the **Settings** option from the pop-up menu. The *MITSU: Communication Settings* dialog is displayed:



Select Settings from the Pop-Up Menu



MITSU: Communication Parameters Dialog

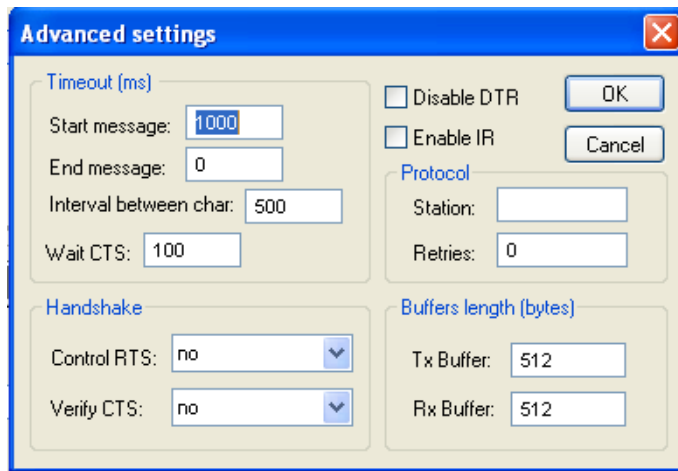
- Specify the parameters as noted in the following table:

Parameter	Default Value	Valid Values	Description
Connection	Serial	Serial or TCP/IP	Type of connection. If you select Serial , then you must also configure the Serial Port settings to match your target device. If you select TCP/IP , then Studio will automatically use the default network settings on your host computer. Note: To enable the Studio to work with the new protocol for FX3U PLCs, select the TCP/IP or TCP/IP (ASCII) option only. For now TCP/IP (ASCII) mode is supported with only FX3U PLCs.
Extended Protocol	0	0 or 1	Enables (1) or disables (0) the use of Extended Protocol. Some devices have problems communicating with high addresses in the D and CR registers (e.g. D:1023). If you have such problems, then set this parameter to 1. Note: This option is not required for FX3U PLCs.
Signed	0	0 or 1	Formats addresses as either Signed (1) or Unsigned (0). Studio will use this parameter by default if you do not specify the format for each address.
Block Size	128	1–128	Block Size in Bytes for a Read operation. Depending on the CPU, this parameter must be adjusted to solve communication problems with addresses like Timers and Counters.

- Proceed to configure the remaining 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.

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

- 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.
		yes	Set the RTS (Request to Send) handshake signal before communication. IMPORTANT.
		yes+echo	Set the RTS (Request to Send) handshake signal before communication, and echo the signal received from the target device.

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.

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.

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

Configuring the Driver Worksheets

Each selected driver includes a Main Driver Sheet and one or more Standard Driver Worksheets. The Main Driver Sheet is used to define tag/register associations and driver parameters that are in effect at all times, regardless of application behavior. In contrast, Standard Driver Worksheets can be inserted to define additional tag/register associations that are triggered by specific application behaviors.

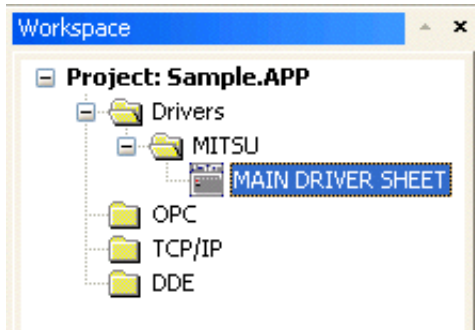
The configuration of these worksheets is described in detail in the “Communication” chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

For the purposes of this document, only MITSU driver-specific parameters and procedures are discussed here.

MAIN DRIVER SHEET

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



Main Driver Sheet in the MITSU 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 MITSU driver.

4. For each table row (i.e., each tag/register association), configure the **Station** and **I/O Address** fields as follows:
 - **Station** field: This field is used only if the driver is configured for TCP/IP connection. (See “Configuring the Communication Settings”) Identify the target device, using the following syntax:
`<IP Address>: [Port Number]`
Example — `192.168.1.53:5551`

Station Field for FX3U PLC:

`<IP Address>: <Port Number>: FX3G`
Example — `192.168.1.53:5551:FX3G`

Attention:

For Ethernet communication, if you do not specify the port number, then the driver assumes your default port number is 5551.

For FX3U PLC, specifying the port number is mandatory. If port number is not given, it might return an Invalid Station.

For FX3U PLC, the String **FX3G** must be specified as the last parameter in the Station field and the Connection on Settings dialog must be TCP/IP. Refer to “*Configuring the Communication Settings*” for details on how to set the Connection. If this is not done, the new protocol will not work as desired.

If the driver is configured for Serial connection, then leave this field blank. The driver will communicate directly with whatever device is connected to the serial port, according to the communication settings.

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.

- **I/O Address:** Specify the address of the associated device register, using the following syntax:
`[Signed/Unsigned]<Type>: [Format]<Address>. [Bit]`
For String Format:
`[Signed/Unsigned]<Type>: [Format]<Address>:<StringSize>`
Examples — `DW0.2`, `ST6:2`

Where:

- `[Signed/Unsigned](optional)`: If you do not specify this parameter, Studio uses the *Communication Parameters* settings to configure integers. Valid values for this parameter are **S** (Signed) or **U** (Unsigned).

- **<Type>** is the register type. Valid values: **X** = Input; **Y** = Output; **M** = Auxiliary Relay; **SM** = Special Auxiliary Relay; **S** = States; **TS** = Timer contact; **CS** = Counter contact; **TO** = Timer Output; **CO** = Counter Output; **TR** = Timer Reset; **CR** = Counter Reset; **T** = Timer Value; **SD** = Special Data Registers; **C16** = Counter Value; **C32** = 32 Bit Counter Value; **D** = Data Registers
- **<Type> for FX3U PLC:** Device register type. Valid values: **X** = Input; **Y** = Output; **M** = Auxiliary Relay; **S** = States; **TS** = Timer contact; **CS** = Counter contact; **TN** = Timer Current Value; **CN** = Counter Current Value; **D** = Data Registers; **R** = Extension Registers
- **[Format]**(optional) is the format of the address values. Valid values: **DW** = Double-Word; **W** = Word; **B** = Byte; **F** = Float; **ST** = String. No **<Format>** character indicates that the address value is treated as the default data type.
- **<Address>** is the address of the register configured in the **<Type>** parameter.
- **[Bit]**(optional) is the bit number. String format doesn't use this parameter. Read only.
- **<StringSize>**(optional) is the size of the string. Only String format use this parameter.

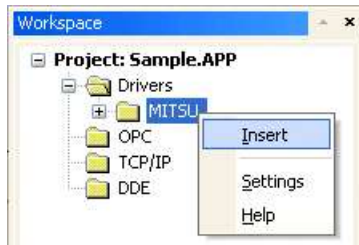
STANDARD DRIVER WORKSHEET

When you select the MITSU 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.

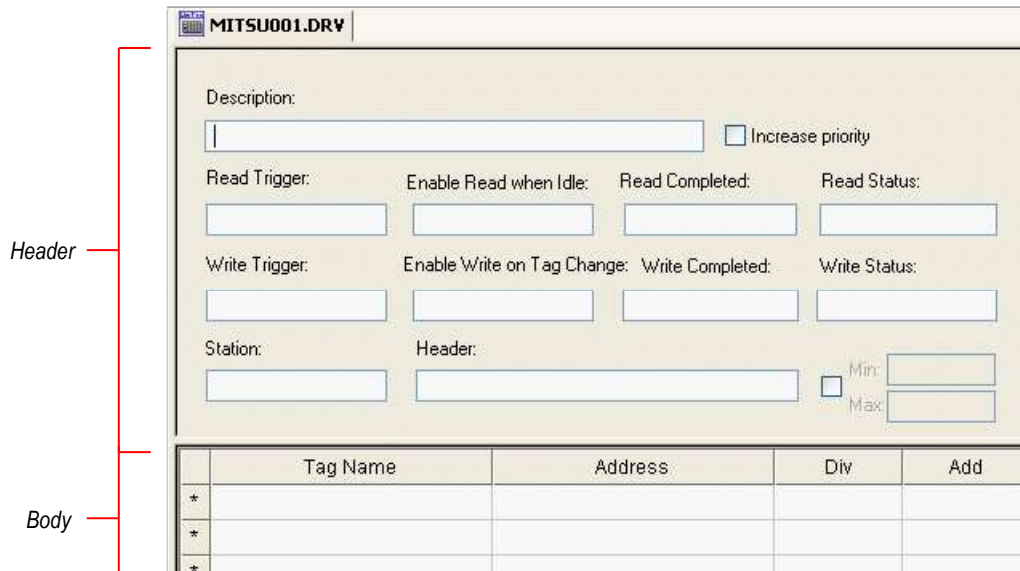
To insert a new Standard Driver Worksheet:

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




Inserting a New Worksheet

A new MITSU driver worksheet is inserted into the *MITSU* subfolder, and the worksheet is opened:



MITSU001.DRV				
Description: <input type="text"/> <input type="checkbox"/> Increase priority				
Read Trigger:	Enable Read when Idle:	Read Completed:	Read Status:	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
Write Trigger:	Enable Write on Tag Change:	Write Completed:	Write Status:	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
Station:	Header:		<input type="checkbox"/> Min:	<input type="text"/>
<input type="text"/>	<input type="text"/>		<input type="checkbox"/> Max:	<input type="text"/>
Tag Name	Address	Div	Add	
*				
*				
*				

MITSU Driver Worksheet

 **Note:**
Worksheets are numbered in order of creation, so the first worksheet is **MITSU001 . 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 MITSU driver.

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

- **Station** field: This field is used only if the diver is configured for TCP/IP connection. (See “Configuring the Communication Settings”) Identify the target device, using the following syntax:

<IP Address>: [Port Number]


Example — **192.168.1.53:5551**

Station Field for FX3U PLC:

<IP Address>: <Port Number>: FX3G

Example — **192.168.1.53:5551:FX3G**

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:**

For Ethernet communication, if you do not specify the port number, then the driver assumes your default port number is 5551.

For FX3U PLC, specifying the port number is mandatory. If port number is not given, it might return an Invalid Station.

For FX3U PLC, the String **FX3G** must be specified as the last parameter in the Station field and the Connection on Settings dialog must be TCP/IP. Refer to “*Configuring the Communication Settings*” for details on how to set the Connection. If this is not done, the new protocol will not work as desired.

If the driver is configured for Serial connection, then leave this field blank. The driver will communicate directly with whatever device is connected to the serial port, according to the communication settings.

- **Header** field: Specify the address of the first register of 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.

For all register types, use the following syntax:

<Type>: <Address Reference>

Example — **x:0**

Where:

- **<Type>** : Device register type. Valid values: **X** = Input; **Y** = Output; **M** = Auxiliary Relay; **SM** = Special Auxiliary Relay; **S** = States; **TS** = Timer contact; **CS** = Counter contact; **TO** = Timer Output; **CO** = Counter Output; **TR** = Timer Reset; **CR** = Counter Reset; **T** = Timer Value; **SD** = Special Data Registers; **C16** = Counter Value; **C32** = 32 Bit Counter Value; **D** = Data Registers
- **<Type> for FX3U PLC**: Device register type. Valid values: **X** = Input; **Y** = Output; **M** = Auxiliary Relay; **S** = States; **TS** = Timer contact; **CS** = Counter contact; **TN** = Timer Current Value; **CN** = Counter Current Value; **D** = Data Registers; **R** = Extension Registers
- **<Address Reference>** : The initial address (reference) of the block of registers configured on this worksheet.

After you edit the **Header** field, Studio checks the syntax to determine if it is valid. If the syntax is invalid, then Studio automatically inserts a default value of **X: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.

The following table describes the possible values of the **Header** field for the FX Series with FX-232AW:

Type	Sample of Syntax	Valid Range of Initial Address	Comment
Input	X:0	0 to 377 (oct)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Octal
Output	Y:0	0 to 377 (oct)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Octal
Auxiliary Relays	M:0	0 to 7999 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Decimal
Special Auxiliary Relays	SM:8000	8000 to 8255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Decimal
States	S:0	0 to 999 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Decimal
Timer Contacts	TS:0	0 to 255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Decimal

Type	Sample of Syntax	Valid Range of Initial Address	Comment
Counter Contacts	CS : 0	0 to 255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Decimal
Timer Output	TO : 0	0 to 255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read ▪ Numeric base: Decimal
Counter Output	CO : 0	0 to 255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read ▪ Numeric base: Decimal
Timer Reset	TR : 0	0 to 255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read ▪ Numeric base: Decimal
Counter Reset	CR : 0	0 to 255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read ▪ Numeric base: Decimal
Timer Value	T : 0	0 to 255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Word ▪ Possible data type: Bit, Byte, Word, Double Word, Float and String ▪ Enabled commands: <ul style="list-style-type: none"> Read Bit, Byte, Word, Double Word, Float and String Write Byte, Word, Double Word, Float and String ▪ Numeric base: Decimal
Counter Value	C16 : 0	0 to 199 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Word ▪ Possible data type: Bit, Byte, Word, Double Word, Float and String ▪ Enabled commands: <ul style="list-style-type: none"> Read Bit, Byte, Word, Double Word, Float and String Write Byte, Word, Double Word, Float and String ▪ Numeric base: Decimal
32 Bit Counter Value	C32 : 200	200 to 255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Double Word ▪ Possible data type: Bit, Byte, Word, Double Word, Float and String ▪ Enabled commands: <ul style="list-style-type: none"> Read Bit, Byte, Word, Double Word, Float and String Write Byte, Word, Double Word, Float and String ▪ Numeric base: Decimal
Data Registers	D : 0	0 to 7999 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Word ▪ Possible data type: Bit, Byte, Word, Double Word, Float and String ▪ Enabled commands: <ul style="list-style-type: none"> Read Bit, Byte, Word, Double Word, Float and String Write Byte, Word, Double Word, Float and String ▪ Numeric base: Decimal

Type	Sample of Syntax	Valid Range of Initial Address	Comment
Special Data Registers	SD : 8000	8000 to 8255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Word ▪ Possible data type Bit, Byte, Word, Double Word, Float and String ▪ Enabled commands: <ul style="list-style-type: none"> Read Bit, Byte, Word, Double Word, Float and String Write Byte, Word, Double Word, Float and String ▪ Numeric base: Decimal

The following table describes the possible values of the **Header** field for the new FX3G Series:

Type	Sample of Syntax	Valid Range of Initial Address	Comment
Input	X : 0	0 to 177 (oct)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Octal
Output	Y : 0	0 to 177 (oct)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Octal
Auxiliary Relays	M : 0	0 to 7679 (dec) 8000 to 8511 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Decimal
States	S : 0	0 to 4095 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Decimal
Timer Contacts	TS : 0	0 to 319 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Decimal
Counter Contacts	CS : 0	0 to 255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Bit ▪ Possible data type: Bit ▪ Enabled commands: Read / Write ▪ Numeric base: Decimal
Timer Current Values	TN : 0	0 to 319 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Word ▪ Possible data type: Bit, Word, Double Word ▪ Enabled commands: <ul style="list-style-type: none"> Read Bit, Word, Double Word Write Word, Double Word ▪ Numeric base: Decimal

Type	Sample of Syntax	Valid Range of Initial Address	Comment
Counter Current Values	CN : 0	0 to 199 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Word ▪ Possible data type: Bit, Word, Double Word ▪ Enabled commands: Read Bit, Word, Double Word Write Word, Double Word ▪ Numeric base: Decimal
32 Bit Counter Value	CN : 200	200 to 255 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Double Word ▪ Possible data type: Double Word ▪ Enabled commands: Read / Write ▪ Numeric base: Decimal
Data Registers	D : 0	0 to 7999 (dec) 8000 to 8511	<ul style="list-style-type: none"> ▪ Default data type: Word ▪ Possible data type: Word, Double Word, Float, Unsigned Word (Bit-String), Unsigned Double Word (bit String) ▪ Enabled commands: Read Bit, Word, Double Word, Unsigned Word and DWord (bit strings), Float Write Word, Double Word, Float, Unsigned Word and DWord, Strings ▪ Numeric base: Decimal
Extension Register	R : 0	0 to 23998 (dec)	<ul style="list-style-type: none"> ▪ Default data type: Float ▪ Possible data type: Word, Double Word, Unsigned Word and DWord (bit strings), Float, Strings ▪ Enabled commands: Read Bit, Word, Double Word, Float, Strings Write Word, Double Word, Float, Strings ▪ Numeric base: Decimal

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

[Signed/Unsigned] [Format] <AddressOffset> . [Bit]

Example — **w0.2**

For String Format:

[Signed/Unsigned] [Format] <AddressOffset> : <StringSize>

Examples — **st6:2**

Example — **w10.12**

Where:

- ***[Signed/Unsigned] (optional)***: If you do not specify this parameter, Studio uses the *Communication Parameters* settings to configure integers. Valid values for this parameter are **s** (*Signed*) or **u** (*Unsigned*).

- **[Format](optional)**: is the format of the address values. Valid values: **DW** = Double-Word; **w** = Word; **B** = Byte; **F** = Float; **ST** = String. No **<Format>** character indicates that the address value is treated as the default data type.
- **<AddressOffset>** is a parameter added to the **<AddressReference>** parameter (configured in the **Header** field) to compose the group address configured in the **Header** field.
- **[Bit](optional)** is the bit number . String format doesn't use this parameter. Read only.
- **<StringSize>(optional)** is the size of the string. Only String format use this parameter.

⊖ Attention:
 This driver does not have group writing. Use the **Write on tag change** parameter to have a good performance.

For examples of how individual device registers are specified using **Header** and **Address**, see the following table:

Address on the Device	Header Field	Address Field	Valid for PLC Type
X0	X: 0	0	FX & FX3U
X1	X: 0	1	FX & FX3U
X7	X: 7	0	FX & FX3U
X10	X: 10	0	FX & FX3U
Y0	Y: 0	0	FX & FX3U
Y1	Y: 0	1	FX & FX3U
Y6	Y: 6	0	FX & FX3U
Y10	Y: 7	1	FX & FX3U
M0	M: 0	0	FX & FX3U
M1	M: 0	1	FX & FX3U
M8	M: 0	8	FX & FX3U
M8	M: 8	0	FX & FX3U
SM8000	SM: 8000	0	Only FX
SM8001	SM: 8000	1	Only FX
SM8009	SM: 8000	9	Only FX
SM8009	SM: 8009	0	Only FX
S0	S: 0	0	FX & FX3U
S1	S: 0	1	FX & FX3U
S4	S: 4	0	FX & FX3U
TS0	TS: 0	0	FX & FX3U
TS1	TS: 0	1	FX & FX3U

Address on the Device	Header Field	Address Field	Valid for PLC Type
TS7	TS : 7	0	FX & FX3U
CS0	CS : 0	0	FX & FX3U
CS1	CS : 0	1	FX & FX3U
CS5	CS : 5	0	FX & FX3U
TO0	TO : 0	0	Only FX
TO1	TO : 1	0	Only FX
TO4	TO : 0	4	Only FX
CO0	CO : 0	0	Only FX
CO1	CO : 0	1	Only FX
CO3	CO : 0	3	Only FX
TR0	TR : 0	0	Only FX
TR1	TR : 0	1	Only FX
TR4	TR : 0	4	Only FX
CR0	CR : 0	0	Only FX
CR1	CR : 0	1	Only FX
CR4	CR : 4	0	Only FX
T0 (word)	T : 0	0	Only FX
T5 (byte)	T : 5	B0	Only FX
T1 (double word)	T : 0	DW1	Only FX
T10 (float point)	T : 0	F10	Only FX
T8 (string)	T : 8	ST0 : 4	FX & FX3U
T0 (word and bit 5)	T : 0	0.5	Only FX
C0 (word)	C16 : 0	0	Only FX
C4 (byte)	C16 : 2	B2	Only FX
C1 (double word)	C16 : 0	DW1	Only FX
C12 (float point)	C16 : 0	F12	Only FX
C10 (string)	C16 : 0	ST10 : 6	Only FX
C0 (word and bit 4)	C16 : 0	0.4	Only FX
C200 (double word)	C32 : 200	0	Only FX
C201 (byte)	C32 : 200	B1	Only FX
C204 (word)	C32 : 0	W204	Only FX
C204 (float point)	C32 : 204	F0	Only FX
C210 (string)	C32 : 200	ST10 : 4	FX & FX3U

Address on the Device	Header Field	Address Field	Valid for PLC Type
C215 (byte and bit 3)	C32 : 210	B5 . 3	Only FX
D0 (word)	D : 0	0	FX & FX3U
D1 (byte)	D : 0	B1	Only FX
D5 (double word)	D : 0	DW5	Only FX
D10 (float point)	D : 10	F0	Only FX
D10 (string)	D : 5	ST5 : 2	FX & FX3U
D2 (word and bit 0)	D : 0	2 . 0	FX & FX3U
SD8000 (word)	SD : 8000	0	Only FX
SD8001 (byte)	SD : 8000	B1	Only FX
SD8002 (double word)	SD : 8001	DW1	Only FX
SD8003 (float point)	SD : 8003	F0	Only FX
SD8004 (string)	SD : 8004	ST4 : 10	Only FX
SD8010 (double word and bit 10)	SD : 8000	DW10 . 10	Only FX

➡ Attention:

You cannot configure a range of addresses that are greater than the maximum block size (data buffer length) supported by each PLC in the same worksheet:

- For the Bit data type, the maximum data buffer length is 512 operands.
- For the Word data type, the maximum data buffer length is 32 operands.
- On the new FX3U PLC, it is not possible to specify C199 or lower (16 bits) and C200 or higher (32 bits) at the same time.
- On the new FX3U PLC, it is not possible to specify D7999 or lower and D8000 or higher at the same time.

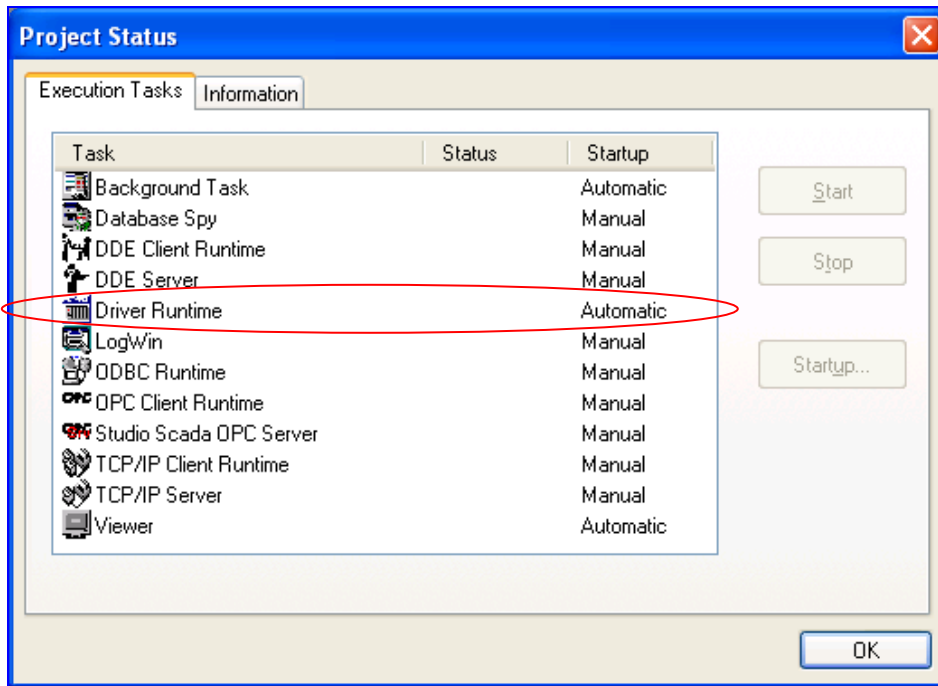
For more information about device registers and addressing, please consult the manufacturer's documentation.

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 MITSU 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	<ul style="list-style-type: none"> Communication without problems 	<ul style="list-style-type: none"> None
1	Invalid Header	<ul style="list-style-type: none"> Wrong Header typed in the driver configuration worksheet. 	<ul style="list-style-type: none"> See samples of valid Headers in the Header Parameter Information table.
3	Invalid Station	<ul style="list-style-type: none"> Wrong configuration in the Station field. There is no connection available. 	<ul style="list-style-type: none"> Check the page 10 (for MDS) or page 12 (for SDS) for configure the station field correctly. Check the TCP/IP connection.
20	Invalid Command	<ul style="list-style-type: none"> Command not allowed for the current register. 	<ul style="list-style-type: none"> Check the registers table (page 4) to verify the validation of the operation.
21	Block Size Error	<ul style="list-style-type: none"> The offset on the Driver Configuration's worksheet is too big, and the message can not be framed. 	<ul style="list-style-type: none"> Change the offsets, or create a new worksheet. Check the Block Size parameter on the Driver Communication Settings
30	Protocol Error	<ul style="list-style-type: none"> The answer from the equipment is not a valid answer. 	<ul style="list-style-type: none"> Check the equipment model and compatibility.
31	Received NACK answer	<ul style="list-style-type: none"> Invalid address typed in the header or address field Invalid command to configured register 	<ul style="list-style-type: none"> Enter the correct address register. Verify enabled command to the respective register.
32	Invalid CheckSum	<ul style="list-style-type: none"> Wrong communication parameters Wrong RTS/CTS configuration Invalid command to configured register. 	<ul style="list-style-type: none"> Check the Communication Parameters for valid configurations. Verify enabled command to the respective register.
33	Writing on bits in WORD and DWORD is not allowed	<ul style="list-style-type: none"> Attempted to write bits on a WORD or DWORD 	<ul style="list-style-type: none"> Write of bits in a WORD or DWORD is not allowed.
-15	Timeout start message	<ul style="list-style-type: none"> Disconnected cables PLC turned off, or in Stop or error mode Wrong Station number Wrong RTS/CTS control settings 	<ul style="list-style-type: none"> Check the cable wiring. Check the PLC state. It must be RUN. Check the station number. Check the Communication Parameters for valid RTS/CTS configurations.
-17	Timeout between rx char.	<ul style="list-style-type: none"> Block Size not supported by this CPU PLC in stop or error mode Wrong station number Wrong parity Wrong RTS/CTS configuration settings 	<ul style="list-style-type: none"> Change the Block Size on the Communication Settings to a new value smaller than the current one Check the cable wiring Check the PLC state. It must be RUN Check the station number. Check the Communication Parameters for valid RTS/CTS configurations.
80	Unknown Subheader	<ul style="list-style-type: none"> Codes for command/response type of 	<ul style="list-style-type: none"> Check and correct command/response type

(50H)	code	subheader are not within the specifications (00 to 05H, 13 to 16H)	set by an external device. (The Ethernet adapter automatically adds command/response type; the user does not need to set these.) <ul style="list-style-type: none"> Check and correct the data length.
84 (54H)	Communication mode mismatch between PLC and Studio. Set both in the same mode - ASCII/Binary	<ul style="list-style-type: none"> When "ASCII code communication" is selected in the [Communication data code settings] of operational setting parameters of GX Works2, ASCII code data that cannot be converted to binary code was received from an external device. 	<ul style="list-style-type: none"> Check and correct the send data of the external device.
86 (56H)	Incorrect Device specified. Check the device code	<ul style="list-style-type: none"> Device designation from the external side is incorrect. 	<ul style="list-style-type: none"> Correct the device designated
87 (57H)	Device points requested exceeds the maximum range.	<ul style="list-style-type: none"> The number of points for a command designated by an external device exceeds the maximum number of processing points for each processing (number of processes that can be executed per communication). Head device number to the designated points exceeds the maximum addresses (device number). When performing batch read/write operations on C200 to C255, the number of device points was designated with an odd number. 	<ul style="list-style-type: none"> Correct the designated points or device number.
		<ul style="list-style-type: none"> Byte length of a command does not conform to the specifications. When writing data, the set number of points for data to be written is different from the number of points specified. 	<ul style="list-style-type: none"> Check the data length of the command and adjust the data setting.
88 (58H)	Head Device number is outside the range for the device requested	<ul style="list-style-type: none"> Head device number of a command designated by an external device is set outside the allowable range. 	<ul style="list-style-type: none"> Designate the appropriate values within the range that are allowed for each processing
		<ul style="list-style-type: none"> A word device is designated in a command for bit devices. The head number of bit devices is designated by a value other than a multiple of 16 in a command for word devices. 	<ul style="list-style-type: none"> Correct the command or the designated device.
96 (60H)	Communication time exceeded the PLC monitoring timer value sent	<ul style="list-style-type: none"> Communication time between the Ethernet adapter and the PLC exceeded PLC monitoring timer value. 	<ul style="list-style-type: none"> Increase the monitoring timer value.

⇒ **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 `ce1log.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 (e.g., Melsec Medoc). 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

A sample application that employs the MITSU driver is provided on the Studio installation CD. We strongly recommend that you use this sample application to test the driver *before* you develop your own applications, for the following reasons:

- To better understand the information and instructions provided in this document;
- To verify that your driver configuration is working satisfactorily with the target device; and
- To ensure that the all of hardware used in the test (i.e. the device, adapter, cable, and PC) is functioning safely and correctly.

 **Note:**

The following instructions assume that you are familiar with developing project applications in Studio. If you are not, then please review the relevant chapters of the Studio *Technical Reference Manual* before proceeding.

To use the sample application:

1. Configure the device's communication settings according to the manufacturer's documentation.
2. Run Studio.
3. From the main menu bar, select **File → Open Project**.
4. Insert the Studio installation CD and browse it to find the sample application. It should be located in the directory `\COMMUNICATION EXAMPLES\MITSU`.
5. Select and open the sample application.
6. Configure and test the driver, as described in the rest of this document.

When you have thoroughly tested the driver with your target device, you may proceed with developing your own Studio application projects.

 **Tip:**

You can use the sample application screen as the maintenance screen for your own applications.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of Changes
A	1.02	Roberto V. Junior	10 Jul 2001	<ul style="list-style-type: none"> First driver version Driver available for Windows CE
B	1.03	Roberto V. Junior	02 Jan 2002	<ul style="list-style-type: none"> Improved communication performance Revision to conform to documentation standards
C	1.04	Lourenço Teodoro	09 Mar 2004	<ul style="list-style-type: none"> Fixed problem when communicating with D registers with address higher than 1023
D	1.05	Leandro G. Coeli	06 Sep 2005	<ul style="list-style-type: none"> Implemented Unsigned/Signed values Implemented Double Words
E	1.06	Rafael R. Fernandes	27 Mar 2007	<ul style="list-style-type: none"> Implemented TCP/IP Communication Included Unsigned/Signed options to the manual
F	1.06	Rafael R. Fernandes	22 May 2007	<ul style="list-style-type: none"> Fixed bugs with TCP/IP Communication Added a new Comm Parameter, Extended Protocol, to solve the problem with addresses higher than 1023 for D register.
G	1.06	Michael D. Hayde	25 May 2007	<ul style="list-style-type: none"> Edited for language and usability.
H	1.07	Rafael R. Fernandes	27 Sep 2007	<ul style="list-style-type: none"> Changed default TCP/IP port to 5551.
I	1.08	Plínio M. Santana Eric Vigiani	Mar 20, 2008	<ul style="list-style-type: none"> Included ST and F addresses formats. Included String length parameter in the Address field. Fixed the Float and String data types. Fixed issue with the virtual groups. Modified the block size for C32 headers. Modified the driver to work with SH4 processor. Removed the bit for the Float data type Modified the driver to accept bit with default data types.
J	10.1	Marcelo Carvalho	Jan 07, 2009	<ul style="list-style-type: none"> Updated driver version, no changes in the contents.
K	10.3	Fellipe Peternella	Jul 14, 2009	<ul style="list-style-type: none"> Added new parameter Block Size to the Communication Settings Fixed problems when writing using Serial communications Fixed problems with Initial Address + Address Offset on Standard Driver Sheet
L	10.4	Fellipe Peternella André Körbes	Jun 01, 2010	<ul style="list-style-type: none"> Fixed problems when reading or writing floats and dwords for headers T, C16, D, SD and C32.
M	10.5	André Körbes	Jan 16, 2013	<ul style="list-style-type: none"> Fixed problem of requesting too many addresses and causing block size errors Fixed detection of NACK messages
N	10.6	Priya Yennam Vijay Kankanala Felipe Andrade	Oct 13, 2014	<ul style="list-style-type: none"> Included details for new protocol implemented, FX3G PLC. Updated information related to Communication Settings. Fixed documentation.
O	10.7	Priya Yennam	Jan 13, 2015	<ul style="list-style-type: none"> Changed the maximum limit for M register to 7999 from 1023 for the FX series.
P	10.8	Priya Yennam	Jul 13, 2015	<ul style="list-style-type: none"> Fixed float issues on the new FX3G protocol

				<ul style="list-style-type: none">▪ Added String Support to the new protocol for the Word Registers.
--	--	--	--	--