

MITSA Communication Driver

Driver for Serial Communication with
Mitsubishi A Series devices using RS-232C

Contents

INTRODUCTION2

GENERAL INFORMATION3

DEVICE CHARACTERISTICS3

LINK CHARACTERISTICS3

DRIVER CHARACTERISTICS3

CONFORMANCE TESTING5

INSTALLING THE DRIVER6

CONFIGURING THE DRIVER.....7

SETTING THE COMMUNICATION PARAMETERS.....7

CONFIGURING THE STANDARD DRIVER WORKSHEET 10

CONFIGURING THE MAIN DRIVER SHEET (MDS) 15

EXECUTING THE DRIVER 17

TROUBLESHOOTING..... 18

REVISION HISTORY 20

Introduction

The MITSA driver enables communication between the Studio system and some of the Mitsubishi devices (A-Series) using their protocol by RS-232C, according to the specifications discussed in this document.

This document was designed to help you install, configure and execute the MITSA driver to enable communication with Mitsubishi devices. The information in this document is organized as follows:

- **Introduction:** Provides an overview of the MITSA driver documentation
- **General Information:** Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the MITSA driver
- **Installing the Driver:** Explains how to install the MITSA driver
- **Configuring the Driver:** Explains how to configure the communication driver
- **Executing the Driver:** Explains how to execute the driver to verify that you installed and configured the driver correctly
- **Troubleshooting:** Lists the most common error codes for this protocol and explains how to fix these error
- **Sample Application:** Explains how to use a sample application to test the driver configuration
- **Revision History:** Provides a log of all modifications made to the driver and the documentation



Notes:

- This document assumes that you have read the “Development Environment” chapter in the product’s *Technical Reference Manual*.
- This document also assumes that you are familiar with the Windows NT/2000/XP environment. If you are unfamiliar with Windows NT/2000/XP, we suggest using the **Help** feature (available from the Windows desktop **Start** menu) as you work through this guide.

General Information

This chapter explains how to identify all the hardware and software components used to implement communication between the MITSA driver and the Mitsubishi A Series device.

The information is organized into the following sections:

- Device Characteristics
- Link Characteristics
- Driver Characteristics
- Conformance Testing

Device Characteristics

- **Manufacturer:** Mitsubishi
- **Compatible Equipment:** A Series PLCs
- **Mitsubishi PLC Programmer Software:** GX-Developer

This driver has been tested successfully with the Mitsubishi A-Series devices. (For a list of the devices used for conformance testing, see “Conformance Testing”).

Link Characteristics

To establish communication, you must use links with the following specifications:

- **Device Communication Port:** RS-232C Port on AJ71C24 module
- **Physical Protocol:** Serial RS-232C
- **Logic Protocol:** Mitsubishi Protocol
- **Device Run-Time Software:** None
- **Specific PC Board:** None
- **Adapters/Converters:** None
- **Cable Wiring:** See PLC manufacturer documentation

Driver Characteristics

The MITSA driver is composed of the following files:

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



Notes:

- All of the preceding files are installed in the /DRV subdirectory of the Studio installation directory.
- You must use Adobe Acrobat® Reader™ (provided on the Studio installation CD-ROM) to view the **MITSA.PDF** document.

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

- Windows NT/2000/Vista
- Windows CE

For a list of the operating systems used for conformance testing, see “Conformance Testing”.

The MITSA driver supports the following registers:

Register Type	Length	Write	Read	Bit	Integer
X (Input)	1 bit	–	•	•	–
Y (Output)	1 bit	•	•	•	–
M (Internal Relay)	1 bit	•	•	•	–
L (Latch Relay)	1 bit	•	•	•	–
B (Link Relay)	1 bit	•	•	•	–
F (Annunciator)	1 bit	•	•	•	–
MS (Special Relay)	1 bit	•	•	•	–
TS (Timer contact)	1 bit	•	•	•	–
TC (Timer coil)	1 bit	•	•	•	–
CS (Counter contact)	1 bit	•	•	•	–
CC (Counter coil)	1 bit	•	•	•	–
TN (Timer preset value)	2 bytes	•	•	•	•
CN (Counter preset value)	2 bytes	•	•	•	•
D (Data Register)	2 bytes	•	•	•	•
W (Link Register)	2 bytes	•	•	•	•
R (File Register)	2 bytes	•	•	•	•
DS (Special Register)	2 bytes	•	•	•	•
DW (Data Register - Double)	4 bytes	•	•	–	•

Conformance Testing

The following hardware/software was used for conformance testing:

- **Driver Configuration:**
 - **COM Port:** COM1
 - **Baud Rate:** 9600
 - **Data Bits:** 8
 - **Stop Bits:** 1
 - **Parity:** Odd

 - **Protocol Type:** FORM1
 - **Checksum:** Enabled
 - **Wait Message Time:** 0ms
 - **Word Swap:** No

 - **Control RTS:** Always On

- **Cable:** See “Link Specifications”.

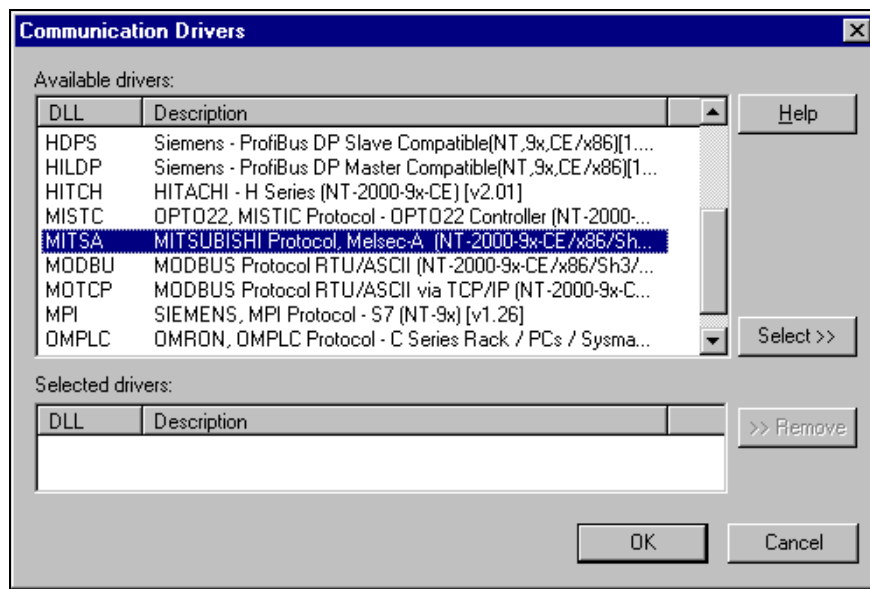
Driver Version	Studio Version	Operating System (development)	Operating System (target)	Equipment
10.3	6.1 + SP5	Windows XP + SP3	Windows XP + SP3	Mitsubishi A1SHCPU (A1SJ71C24-R2)

Installing the Driver

When you install Studio version 5.1 or higher, all of the communication drivers are installed automatically. You must select the driver that is appropriate for the application you are using.

Perform the following steps to select the driver from within the application:

1. Open Studio from the **Start** menu.
2. From the Studio main menu bar, select **File** → **Open Project** to open your application.
3. Select **Insert** → **Driver** from the main menu bar to open the *Communication drivers* dialog.
4. Select the **MITSA** driver from the *Available Drivers* list (as shown in the following figure), and then click the **Select** button.



Communication Drivers Dialog Box

5. When the **MITSA** driver displays in the **Selected Drivers** list, click the **OK** button to close the dialog.

Note:

It is not necessary to install any other software on your PC to enable the communication between the host and the device. However, to download a custom program to the device, you must install one of the Mitsubishi programmer software packages, such as Melsec Medoc. Please see the Melsec Medoc documentation for the software installation procedure.

Attention:

Special precautions must be taken when installing the physical hardware. Refer to the hardware manufacturer's documentation for specific instructions in this area.

Configuring the Driver

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

- Specifying communication parameters
- Defining tags and controls in the *MAIN* and *STANDARD DRIVER SHEETS* (or Communication tables)

The 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.

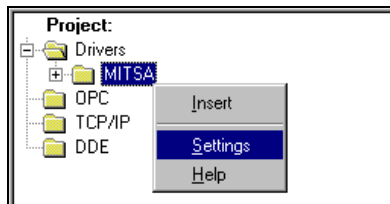
Note:

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*.

Setting the Communication Parameters

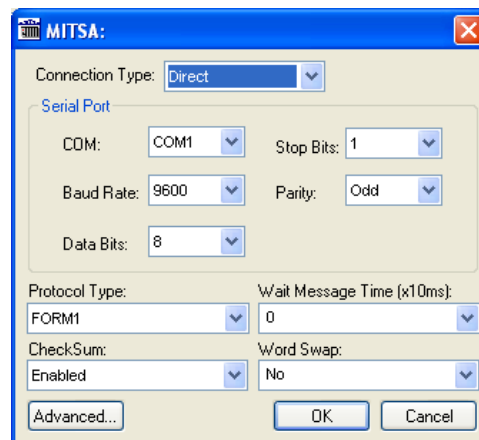
Use the following steps to configure the communication parameters, which are valid for all driver worksheets configured in the system:

1. From the Studio development environment, select the *Comm* tab located below the *Workspace*.
2. Click on the *Drivers* folder in the *Workspace* to expand the folder.
3. Right-click on the *MITSA* subfolder and when the pop-up menu displays (as shown in the following figure), select the **Settings** option.



Select Settings from the Pop-Up Menu

The *MITSA: Communications Parameters* dialog displays (as follows).



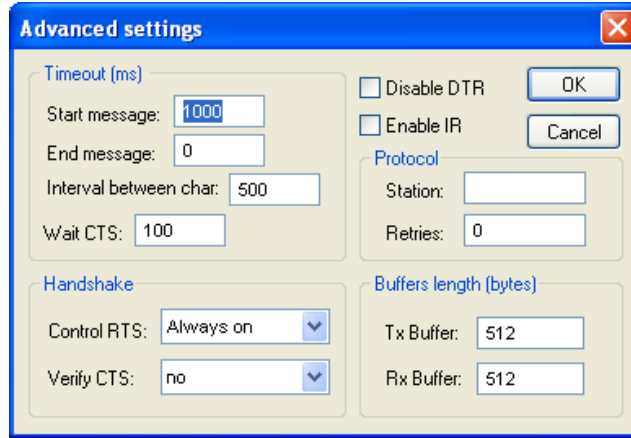
MITSA: Communication Parameters Dialog

4. Specify the parameters as noted in the following table:

Parameters	Default Values	Valid Values	Description
Protocol Type	FORM1	FORM1 FORM2 FORM3 FORM4	<p>Four formats of the control procedures (structure, transmission/reception Procedures of command messages and response messages) are available for an external device to access the PLCs.</p> <p>The differences between the four formats in relation to Format 1 are as follows:</p> <p>Format 2 : A block number is added to each message Format 3 : Each message is enclosed between STX and ETX Format 4 : CR and LF are added to each message</p>
Check Sum	Disabled	Disabled Enabled	<p>When "check sum is disabled" Studio does not attach the checksum code to the transmission message.</p> <p>When "sum check is enabled," Studio attaches it to the transmission message.</p>
Wait Message Time (x10ms)	0	0 ..15	<p>Wait Message time is a data value for generating delay time in response transmission. Some external devices require a certain time to go into the receiving status after sending a command. This value designates the minimum time that the Q series C24 must wait before sending a result after receiving a command from an external device. The wait time should be designated according to the specifications of the external device.</p> <p>The wait time is designated in 10 ms units in the range from 0 to 150 ms, where every 10 ms is converted to 1 H in order to obtain a 1-digit ASCII code (hexadecimal) from 0H to FH (0 to 15). E.g.: 0=0ms, 1=10ms, ... , 15=150ms).</p>
Word Swap	No	No Yes	<p>Option to change the order of the word as they are processed. This option works only for the DW data type (Data Register - Double):</p> <p>No – Word Swap OFF; word are not swapped (Lo Hi) Yes – Word Swap ON; registers are swapped (Hi Lo)</p>

Note:
 These MITSA driver Communication Parameters must be configured exactly the same as the parameters configured for the device.

You can click the **Advanced** button in the *Communication Parameters* dialog to access additional communication parameters.



Advanced settings Dialog

The *Advanced settings* parameters are explained in the *Studio Technical Reference Manual*. You should not change any of the default values for these fields, except the **Control RTS** field. Configure this field as described in the following table.

Parameter	Default Value	Valid Values	Description
Control RTS	No	<ul style="list-style-type: none"> ▪ no ▪ yes ▪ yes + echo ▪ Always on 	Define if the RTS (<i>Request to Send</i>) handshake signal is set before communication and if there is an echo in the communication. If you are using Windows 95 or CE with the correct RS 232 – RS 485 Converter (without RTS Control), select the “no” option. If you are using Windows NT and the Cutler Hammer RS232 – 485 adapter, you must select the “yes” option. Important: Using the wrong settings in this field will prevent the driver from working correctly and will cause Timeout error messages.

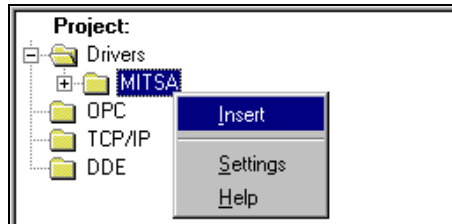
Tip:
 Generally, you must change the *Advanced* parameter settings if you are using a DCE (Data Communication Equipment) converter (232/485, for example), modem, and so forth between the PC, the driver and the host. You must be familiar with the DCE specifications before adjusting these configuration parameters.

Configuring the Standard Driver Worksheet

This section explains how to configure a *Standard Driver Worksheet* (or communication table) to associate application tags with the PLC addresses. You can configure multiple Driver Worksheets — each of which is divided into a *Header* section and *Body* section.

Use the following steps to create a new Standard Driver Worksheet:

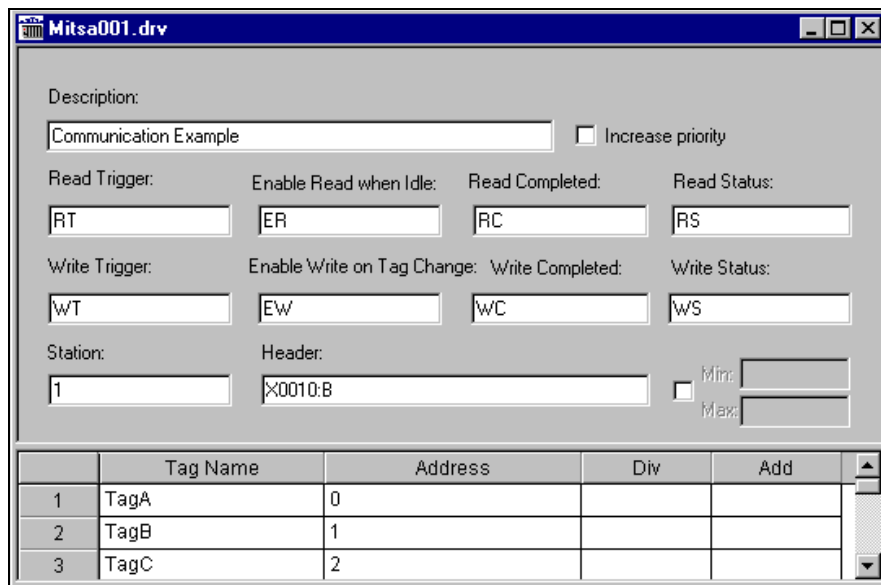
1. From the Studio development environment, select the *Comm* tab, located below the *Workspace* pane.
2. In the *Workspace* pane, expand the *Drivers* folder and right-click the *MITSA* subfolder.
3. When the pop-up menu displays (as shown in the following figure), select the **Insert** option.



Inserting a New Worksheet

Note:
 To optimize communication and ensure better system performance, you must tie the tags in different driver worksheets to the events that trigger communication between each tag group and the period in which each tag group must be read or written. Also, we recommend configuring the communication addresses in sequential blocks to improve performance.

The *Mitsa001.drv* dialog box displays (similar to the following figure).



MITSA Driver Worksheet

4. Use the following information to complete the **Station**, **Header** and **Address** fields on this worksheet.
- **Station** field: Use this field to specify the Station number of the target device. Valid values are 0– 31 (no default).
 - **Header** field: Use the information in the following table to define the type of variables that will be read from or written to the device, and a reference to the initial address. Default value is **X0000 : B**.
 - These variables must comply with the following syntax:

<Type> : <AddressReference> : <Format> (For example: **X0000 : B**)

Where:

- **<Type>** is the register type. Valid values: **X** = Input; **Y** = Output; **M** = Internal Relay; **L** = Latch Relay; **B** = Link Relay; **F** = Annunciator; **MS** = Special Relay; **TS** =Timer contact; **TC** =Timer coil; **CS** = Counter contact; **CC** = Counter coil; **TN** =Timer preset value; **CN** =Counter preset value; **D** =Data Register; **W** =Link Register; **R** = File Register; **DS** = Special Register; **DW** = Data Register – double-word
- **<AddressReference>** is the initial address (reference) of the configured group. This number *always* refers to the Byte.
- **<Format>** specifies whether to treat the values as words (**w**) or as bits (**B**).

 **Notes:**

- Only format **w** is supported by the word devices (**TN**, **CD**, **D**, **w**, **R**, **DS**, and **DW**).
- After editing the **Header** field, the system will check the validity of the entry. If the syntax is not correct, the system will automatically insert the default value (**X0000 : B**) in this field.

You can type a tag in curly brackets into this field, but be sure that the tag's value and syntax are correct, or you will get an Invalid Header error. The following table contains the correct tag syntax and values.

Header Parameter Information			
Type	Syntax Sample	Valid Initial Address Ranges	Comments
X (Input)	X0000:W or X0000:B	<ul style="list-style-type: none"> ▪ X0000 to X07FF(AnACPU) ▪ X0000 to X1FFF(AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device (Read only) ▪ Address Reference Format: Hexadecimal
Y (Output)	Y0000:W or Y0000:B	<ul style="list-style-type: none"> ▪ Y0000 to Y07FF(AnACPU) ▪ Y0000 to Y1FFF(AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device ▪ Address Reference Format: Hexadecimal
M (Internal Relay)	M0000:W or M0000:B	<ul style="list-style-type: none"> ▪ M0000 to M2047(AnACPU) ▪ M0000 to M8191(AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device ▪ Address Reference Format: Decimal
L (Latch Relay)	L0000:W or L0000:B	<ul style="list-style-type: none"> ▪ L0000 to L2047(AnACPU) ▪ L0000 to L8191(AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device ▪ Address Reference Format: Decimal
B (Link Relay)	B0000:W or B0000:B	<ul style="list-style-type: none"> ▪ B0000 to B03FF(AnACPU) ▪ B0000 to B1FFF(AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device ▪ Address Reference Format: Hexadecimal
F (Annunciator)	F0000:W or F0000:B	<ul style="list-style-type: none"> ▪ F0000 to F0255(AnACPU) ▪ F0000 to F2047(AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device ▪ Address Reference Format: Decimal
MS (Special Relay)	MS9000:B	<ul style="list-style-type: none"> ▪ MS9000 to MS9255 (AnA/AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device ▪ Address Reference Format: Decimal
TS (Timer contact)	TS000:W or TS000:B	<ul style="list-style-type: none"> ▪ TS000 to TS255(AnACPU) ▪ TS000 to TS2047(AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device ▪ Address Reference Format: Decimal
TC (Timer coil)	TC000:W or TC000:B	<ul style="list-style-type: none"> ▪ TC000 to TC255(AnACPU) ▪ TC000 to TC2047(AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device ▪ Address Reference Format: Decimal
CS (Counter contact)	CS000:W or CS000:B	<ul style="list-style-type: none"> ▪ CS000 to CS255(AnACPU) ▪ CS000 to CS1023(AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device ▪ Address Reference Format: Decimal
CC (Counter coil)	CC000:W or CC000:B	<ul style="list-style-type: none"> ▪ CC000 to CC255(AnACPU) ▪ CC000 to C1023(AnUCPU) 	<ul style="list-style-type: none"> ▪ Bit Device ▪ Address Reference Format: Decimal
TN (Timer preset)	TN000:W	<ul style="list-style-type: none"> ▪ TN000 to TN255(AnACPU) ▪ TN000 to TN2047(AnUCPU) 	<ul style="list-style-type: none"> ▪ Word Device ▪ Address Reference Format: Decimal
CN (Counter preset)	CN000:W	<ul style="list-style-type: none"> ▪ CN000 to CN255(AnACPU) ▪ CN000 to N1023(AnUCPU) 	<ul style="list-style-type: none"> ▪ Word Device ▪ Address Reference Format: Decimal
D (Data Register)	D0000:W	<ul style="list-style-type: none"> ▪ D0000 to D1023(AnACPU) ▪ D0000 to D8191(AnUCPU) 	<ul style="list-style-type: none"> ▪ Word Device ▪ Address Reference Format: Decimal
W (Link Register)	W0000:W	<ul style="list-style-type: none"> ▪ W0000 to W03FF(AnACPU) ▪ W0000 to W1FFF(AnUCPU) 	<ul style="list-style-type: none"> ▪ Word Device ▪ Address Reference Format: Hexadecimal
R (File Register)	R0000:W	<ul style="list-style-type: none"> ▪ R0000 to R8191 ▪ (AnA/AnUCPU) 	<ul style="list-style-type: none"> ▪ Word Device ▪ Address Reference Format: Decimal
DS (Special Register)	DS9000:W	<ul style="list-style-type: none"> ▪ DS9000 to DS9255 ▪ (AnA/AnUCPU) 	<ul style="list-style-type: none"> ▪ Word Device ▪ Address Reference Format: Decimal
DW (Data Register)	DW0000:W	<ul style="list-style-type: none"> ▪ DW0000 to DW1022(AnACPU) ▪ DW0000 to DW8190(AnUCPU) 	<ul style="list-style-type: none"> ▪ Double-Word Device ▪ Address Reference Format: Decimal


- **Address** field: Use the information provided in the following table to associate each tag to its respective device address.

Type the tag from your application database into the **Tag Name** column. This tag will receive values from or send values to an address on the device. The address must comply with the following syntax:

<AddressOffset>.<Bit> (For example: 10.2)

Where:

- **<AddressOffset>** is a parameter added to the **AddressReference** parameter (configured in the **Header** field) to compose the group address configured in the **Header** field. The **AddressNumberType** configured in the **Header** field defines whether the **AddressOffset** is a **Byte** offset or a **Word** offset.
- **<Bit>** is the bit number (from 0 – 15) from the **Word** address. This parameter is *optional*.

 **Notes:**

- When configuring a bit device in the **Header** field with the **B** format (for example, **x0000:B**), the **Address** column syntax is **<AddressOffset>**.
- **Important!** The **DW** type does not support bit reading/writing.

Address Configuration Sample		
Address on the Device	Header	Address
X0001 (Hexadecimal)	X0000:B	1
	X0001:B	0
	X0000:W	0.1
X000E (Hexadecimal)	X0000:B	14
	X000E:B	0
	X0000:W	0.14
X0010 (Hexadecimal)	X0000:B	16
	X0010:B	0
	X0000:W	1.0
M0001 (Decimal)	M0000:B	1
	M0001:B	0
	M0000:W	0.1
M0014 (Decimal)	M0000:B	14
	M0014:B	0
	M0000:W	0.14
M0016 (Decimal)	M0000:B	16
	M0016:B	0
	M0000:W	1.0

W0001 (Hexadecimal)	W0000:W	1
	W0001:W	0
W000E (Hexadecimal)	W0000:W	14
	W000E:W	0
W0010 (Hexadecimal)	W0000:W	16
	W0016:W	0
W0001 – Bit 2 (Hexadecimal)	W0000:W	1.2
	W0001:W	0.2
W0001 – Bit F (Hexadecimal)	W0000:W	1.15
	W0001:W	0.15
W000E – Bit F (Hexadecimal)	W0000:W	14.15
	W000E:W	0.15
TN0001 (Decimal)	TN0000:W	1
	TN0001:W	0
TN0014 (Decimal)	TN0000:W	14
	TN0014:W	0
TN0016 (Decimal)	TN0000:W	16
	TN0016:W	0
TN0001 – Bit 2 (Decimal)	TN0000:W	1.2
	TN0001:W	0.2
TN0001 – Bit F (Decimal)	TN0000:W	1.15
	TN0001:W	0.15
TN0014 – Bit F (Decimal)	TN0000:W	14.15
	TN0014:W	0.15

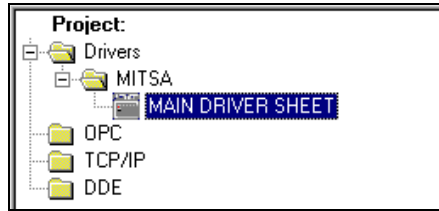
➔ Attention:

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 **X, Y, M, L, B, F, MS, TS, TC, CS** and **CC** bit devices, the maximum data buffer length is 256 bytes.
- For word devices (**TN, CN, D, W, R** and **D**), the maximum data buffer length is 63 words.
- For double-word devices (**DW**), the maximum data buffer length is 30 double-words.

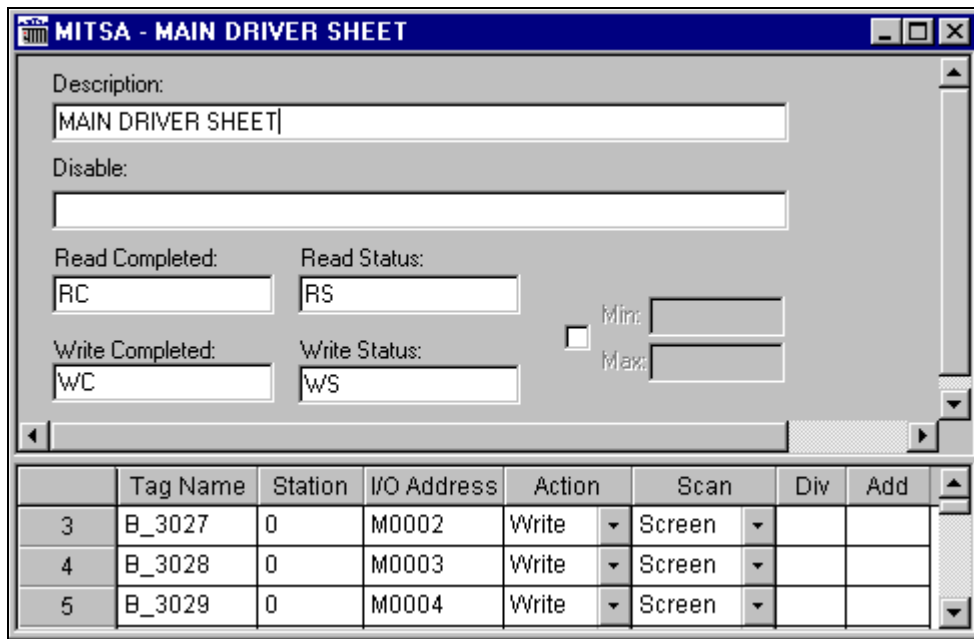
Configuring the Main Driver Sheet (MDS)

When you insert the driver into your application, the MAIN DRIVER SHEET is automatically added to the driver folder (as shown in the following figure).



Main Driver Sheet

Double-click on the *MAIN DRIVER SHEET* folder to open the following worksheet.



Main Driver Sheet

- **Station:** PLCs address (ID number)
- **I/O Address:** Address of each register in the PLC. Use the following syntax:
 - For bit devices (**X, Y, M, L, B, F, MS, TS, TC, CS** and **CC**):
 - <Type><Address> (Read 1 point. For example: **x0000**)
 - <Type><Address>:**W** (Read 16 point. For example: **x0000 :W** – Read from **x0000** to **x000F**)
 - For word devices (**TN, CN, D, W, R, DS** and **DW**):
 - <Type><Address> (Read 1 word. For example: **TN0000**)
 - <Type><Address>.**<Bit>** (Read 1 bit from the word point. For example: **TN0000 .2** – Read bit 2 from **TN0000**)

Where:

- **<Type>** is the Register type. Valid values: **X** = Input, **Y** = Output, **M** = Internal Relay, **L** = Latch Relay, **B** = Link Relay, **F** = Annunciator, **MS** = Special Relay, **TS** =Timer contact, **TC** =Timer coil, **CS** = Counter contact, **CC** = Counter coil, **TN** =Timer preset value, **CN** =Counter preset value, **D** =Data Register, **W** =Link Register, **R** = File Register, **DS** = Special Register, **DW** = Data Register – double-word
- **<Address>** is the Device register address.
- **<Bit>** is the Bit number (from 0 to 15) from the word address. This is an *optional* parameter.



Attention:

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*.

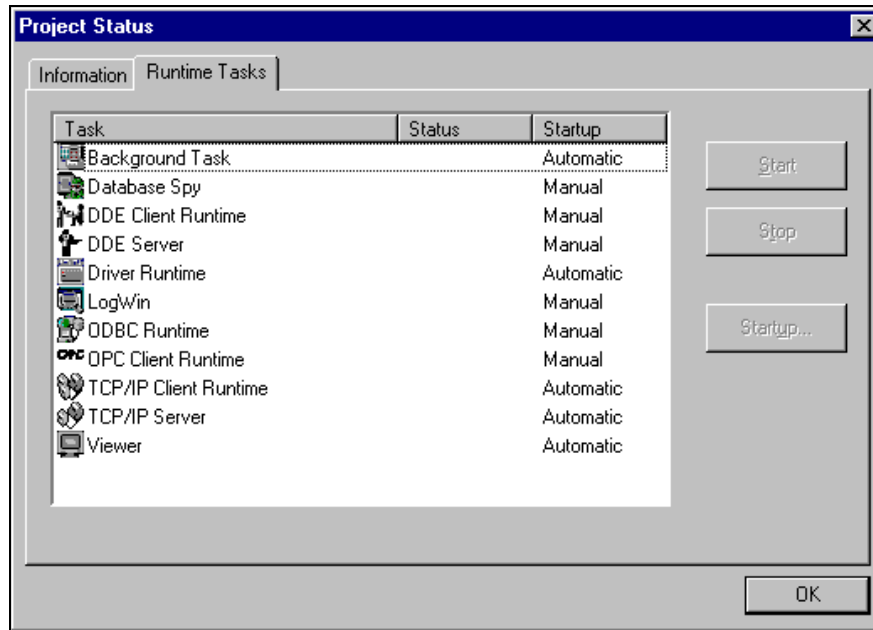
Executing the Driver

After adding the MITSA driver to a project, Studio sets the project to execute the driver automatically when you start the run-time environment.

To verify that the driver run-time task is enabled and will start correctly, perform the following steps:

1. Select **Project** → **Status** from the main menu bar.

The *Project Status* dialog box displays, as follows.



Project Status Dialog Box

2. Verify that the *Driver Runtime* task is set to **Automatic**.
 - If the setting is correct, click **OK** to close the dialog box.
 - If the **Driver Runtime** task is set to **Manual**, select the **Driver Runtime** line. When the **Startup** button becomes active, click the button to toggle the *Startup* mode to **Automatic**.
3. Click **OK** to close the *Project Status* dialog.
4. Start the application to run the driver.

Troubleshooting

If the MITSA driver fails to communicate with the device, the tag you configured for the **Read Status** or **Write Status** fields will receive an error code. Use this error code and the following table to identify the failure that occurred.

Error Code	Description	Possible Causes	Procedure to Solve
0	OK	Communication without problems	None
2	Block Size Error	<ul style="list-style-type: none"> Address offset greater than 255 bit command Address offset greater than 64 word operands 	Check the driver's configuration worksheet
5	Invalid Command	<ul style="list-style-type: none"> Bit command with Word Devices Write bit with "Write Trigger" 	<ul style="list-style-type: none"> If the command is a Word Device, the command must be a word command. If write bit with "Write Trigger," change configuration to "Write on Tag Change"
7	Invalid Response	NACK received in response	<ul style="list-style-type: none"> Check the serial communication configuration. Check the operands in the device.
20	Invalid Address	An invalid Address has been typed to write bit	Type a valid address in the address field or for the tag value.
30	Invalid Header	An invalid Header has been typed or the tag that is inside this field has an invalid configuration	Type a valid Header in the Header field or for the tag value. A list of valid Headers is shown on the Header Parameter Information table.
-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 right configuration. Review the <i>Communication Parameters</i> section for valid RTS/CTS configurations.
-17	Timeout between rx characters	<ul style="list-style-type: none"> PLC in stop or error mode Wrong station number Wrong parity Wrong RTS/CTS configuration 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 configuration. Review the <i>Communication Parameters</i> section for valid RTS/CTS configurations.

➔ **Tip:**
 You can verify communication status using the Studio development environment *Output* window (*LogWin* module). To establish an event log for **Field Read Commands**, **Field Write Commands** and **Protocol Analyzer** right-click in the *Output* window. When the pop-up menu displays, select the option to set the log events. If you are testing a Windows CE target, you can use the **Remote LogWin** (*Tools->Remote LogWin*)

If you are unable to establish communication with the PLC, try to establish communication between the PLC Programming Tool and the PLC. Quite frequently, communication is not possible because you have a hardware or cable problem, or a PLC configuration error. After successfully establishing communication between the device's Programming Tool and the PLC, you can retest the supervisory driver.

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 time that this document was written.

Revision History

Revision	Driver Version	Author	Date	Description of Changes
A	1.00	Roberto V. Junior	8-May-2000	First driver version
B	1.01	Roberto V. Junior	26-May-2000	Fixed bug with MS and DS operand
C	1.02	Roberto V. Junior	02-Ago-2000	<ul style="list-style-type: none"> ▪ Included command to read/write high address to AnUCPU ▪ Modified "Protocol Type" of communication parameter
D	1.03	Lourenço Teodoro	02-Sep-2000	Implemented for CE operation system
E	1.04	Lourenço Teodoro	30-Oct-2000	Included MAIN DRIVER SHEET feature
F	1.05	Lourenço Teodoro	02-Apr-2001	Implemented DW register type
G	1.06	Roberto V. Junior	20-Jul-2001	Fixed CheckSum bug
H	1.09	Fábio Komura	18-Dec-2001	Fixed Block Size bug
I	1.10	Leandro G. Coeli	18-Dec-2001	Implemented SwapRegister to DW datatype
J	1.11	Leandro G. Coeli	06-Sep-2005	Implemented Unsigned/Signed values
K	1.12	Rafael R. Fernandes / Eric Vigiani	25-Apr-2008	Fixed bug with bit operations Enabled the combo-boxes in the Communication parameters.
L	10.1	Marcelo Carvalho	07-Jan-2009	Updated driver version, no changes in the contents.
M	10.3	Fellipe Peternella	1-May-2009	Modified Main Driver Sheet to properly create groups