MPMPI Communication Driver

Driver for Serial Communication Between
MP370 HMIs and Siemen's Devices using MPI Interface

Contents

Introduction

The MPMPI driver enables communication between HMI MP370 with CEView™ system and the Siemens S7-200, S7-300, and S7-400 devices using the Open Platform Program Application Development Kit, according to the specifications discussed in this publication.

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

- **Introduction**: Provides an overview of the MPMPI driver documentation.
- **General Information**: Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the MPMPI driver.
- **Installing the Driver**: Explains how to install the MPMPI 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 errors.
- **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 Studio MPMPI driver and Siemens S7-200, S7-300, and S7-400 devices.

The information is organized into the following sections:

- Device Characteristics
- Link Characteristics
- Driver Characteristics

Device Characteristics

This driver has been tested successfully with the following devices:

- **Manufacturer**: Siemens
- **Compatible Equipment**: Any Siemens PLC that is compatible with the MPI protocol
- **Siemens PLC Programmer Software**: Step 7, Step 7 Micro Win

For a list of the devices used for conformance testing, see "Conformance Testing" on page 4.

Link Characteristics

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

- **Device Communication Port**: 1F 1B/RS 422/485
- **Physical Protocol**: MPI Cable
- **Logic Protocol**: MPI Protocol
- **Adapters/Converters**: MPI Cable (If you are not using MIP-ISA.)

Driver Characteristics

The MPMPI driver is composed of the following files:

- In the **/DRV** subdirectory:
  - **MPMPI.INI**: Internal driver file. *You must not modify this file*.
  - **MPMPI.MSG**: Internal driver file containing error messages for each error code. *You must not modify this file*.
  - **MPMPI.PDF**: Document providing detailed information about the MPMPI driver.
  - **MPMPI.DLL**: Compiled driver.
  - **MPMPIExe.exe**: Executable file responsible for the communication.
- In the **FLASH\ADDONS** subdirectory:
  - **ATLCE211MP370.DLL**: Internal driver file. *You must not modify this file*.
  - **FWMEMMGR.DLL**: Internal driver file. *You must not modify this file*.
  - **FWOS.DLL**: Internal driver file. *You must not modify this file*.
  - **FWUTILBASE.DLL**: Internal driver file. *You must not modify this file*.
- In the **FLASH\ADDONS\MPI** subdirectory:
  - **FWCHNS7.DLL**: Internal driver file. *You must not modify this file*.
  - **MPIABSTRACTIONLAYER.DLL**: Internal driver file. *You must not modify this file*.

✍ **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 **`MPMPI.PDF`** document.

You can use the MPMPI driver on the **Windows CE** operating system only.

The MPMPI driver supports the following registers:

Register Type	Write	Read	Bit	Integer	Float	String
M (*Flags*)	●	●	●	●	●	●
T (*Timers*)	–	●	–	●	–	–
Z or C (*Counters*)	–	●	–	●	–	–
E or I (*Inputs*)	–	●	●	●	–	–
A or Q (*Outputs*)	●	●	●	●	–	–
DB (*Data Blocks*)	●	●	●	●	●	●

Conformance Testing

The following hardware/software was used for conformance testing:

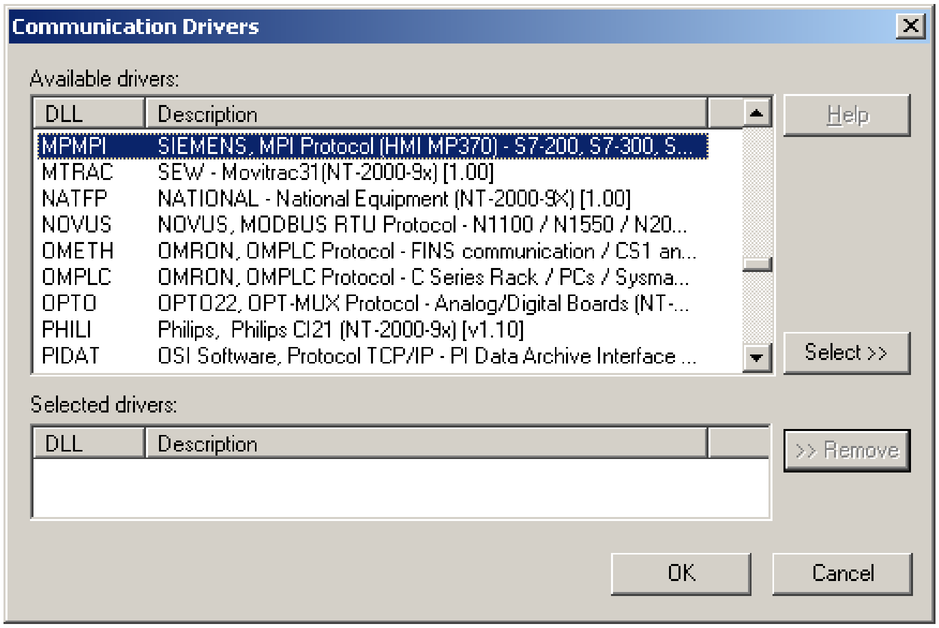
- **Equipment**: Siemens PLC S7-200 CPU226 01.01, Siemens PLC S7-300 CPU315-2DP 1.2.1, SIMATIC C7-621
- **Driver Configuration**:
  - **PLC Program**: Step 7, Step 7 Micro Win
  - **Protocol**: MPI Protocol
  - **Communication**: Asynchronous Half-Duplex
  - **COM Port**: COM1
- **Operating System (*development*)**: Windows XP
- **Operating System (*runtime*)**: Windows CE 3.0
- **Studio Version**: 5.1 + SP4
- **Driver Version**: 1.04

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 **MPMPI** driver from the *Available Drivers* list, and then click the **Select** button.



Available drivers:

DLL	Description
MPMPI	SIEMENS, MPI Protocol (HMI MP370) - S7-200, S7-300, S...
MTRAC	SEW - Movitrac31(NT-2000-9x) [1.00]
NATFP	NATIONAL - National Equipment (NT-2000-9X) [1.00]
NOVUS	NOVUS, MODBUS RTU Protocol - N1100 / N1550 / N20...
OMETH	OMRON, OMPLC Protocol - FINS communication / CS1 an...
OMPLC	OMRON, OMPLC Protocol - C Series Rack / PCs / Sysma...
OPTO	OPTO22, OPT-MUX Protocol - Analog/Digital Boards (NT-...
PHILI	Philips, Philips CI21 (NT-2000-9x) [v1.10]
PIDAT	OSI Software, Protocol TCP/IP - PI Data Archive Interface ...

Selected drivers:

DLL	Description

Communication Drivers Dialog

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

Note:

You must install Step 7 or Step 7 Micro Win software on your computer to enable communication between the host and the device, and to download the custom program to the device. Please consult the Step 7 and Step7 Micro Win documentation for information about installing their software.

Attention:

For safety reasons, you must take precautions when you install the physical hardware. Consult the hardware manufacturer's documentation for specific installation instructions.

Configuring the Driver

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

- Specifying communication parameters
- Defining communication tags and controls in the Communication tables or *Driver* worksheet

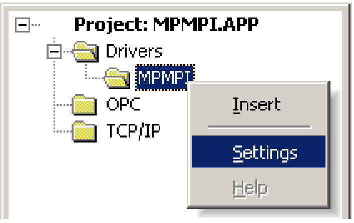
Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the Station, Header, and Address fields, which are driver-specific. This document explains how to configure the Station, Header, and Address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

> **Notes:**
> 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

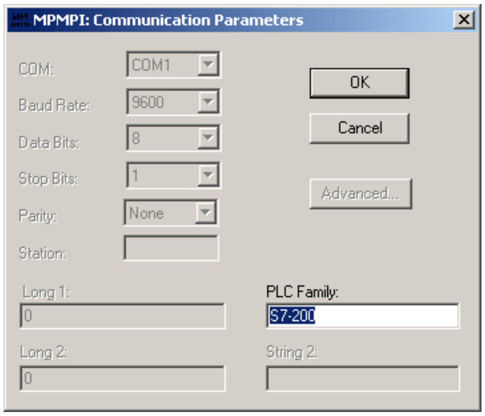
Use the following steps to configure the communication parameters, which are valid for all driver worksheets configured in the system):

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 *MPMPI* 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 *MPMPI: Communications Parameters* dialog displays:



COM: COM1
Baud Rate: 9600
Data Bits: 8
Stop Bits: 1
Parity: None
Station:
Long 1: 0
PLC Family: S7-200
Long 2: 0
String 2:

MPMPI: Communication Parameters Dialog

4. Specify the **PLC Family** parameter as noted in the following table:

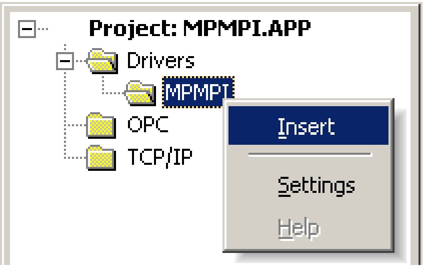
Parameter	Default Values	Valid Values	Description
PLC Family	S7-200	S7-200, S7-300, S7-400	Specify the PLC Family

Configuring the 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:

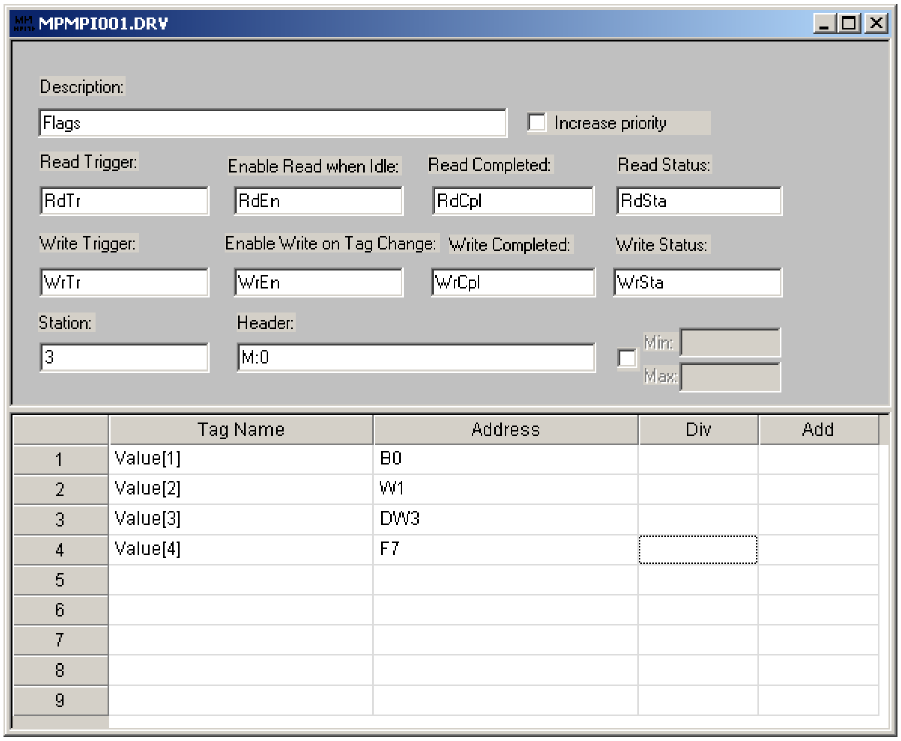
1. From the Studio development environment, select the *Comm* tab, located below the *Workspace* pane.
2. In the *Workspace* pane, expand the *Drivers* folder and right-click the *MPMPI* subfolder.
3. When the pop-up menu displays, select the **Insert** option.



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, it is recommended that you configure the communication addresses in sequential blocks to improve performance.

The *<drivername>.drv* dialog displays (similar to the following figure):



MPMPI001.DRV

Description: Flags

☐ Increase priority

Read Trigger: RdTr
Enable Read when Idle: RdEn
Read Completed: RdCpl
Read Status: RdSta

Write Trigger: WrTr
Enable Write on Tag Change: WrEn
Write Completed: WrCpl
Write Status: WrSta

Station: 3
Header: M:0

☐ Min: Max:

	Tag Name	Address	Div	Add
1	Value[1]	B0		
2	Value[2]	W1		
3	Value[3]	DW3		
4	Value[4]	F7		
5				
6				
7				
8				
9				

MPMPI Driver Worksheet

In general, all parameters on the *Driver* worksheet (except the **Station**, **Header**, and **Address** fields) are standard for all communication drivers, and they will not be discussed in this publication. For detailed information about configuring the standard parameters, consult the *Studio Technical Reference Manual*.

4. Use the following information to complete the **Station**, **Header**, and **Address** fields on this worksheet.
   - **Station** field: Use this field to specify the PLC address (*PLC ID*). Valid values are 1– 255 (*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 *DB2*.)

These variables must comply with the following syntax:

For Flags, Timers, Counters, Inputs, and Outputs:

<Type>:<AddressReference> (For example: **M:1**)

For Data-Blocks:

<Type><TypeGroup>:<AddressReference> (For example: **DB2:1**)

Where:

- **Type** is the register type. (M=Flags, T=Timers, Z or C=Counters, E or I=Inputs, A or Q=Outputs, and DB=Data Blocks)
- **TypeGroup** is the group number of the configured register type (for Data-Block types only).

- AddressReference is the initial address (reference) of the configured group. This number always refers to the Byte address number. (See the following table.)

  W indicates that this type of offset will receive a word at the address indicated by adding the offset number to the address in the header (**AddressReference**).

  B indicates the number of bytes (offset) will be added be added to the **AddressReference.** This number *always* refers to the *Byte address number*.

The following table lists all of the valid initial address (reference) values for the MPMPI driver:

Header Address	Siemens Address		
Byte Address Number	Byte Address Number	Word Address Number	
Byte 0	Byte 0	W0	
Byte 1	Byte 1		W1
Byte 2	Byte 2	W2	
Byte 3	Byte 3		W3
Byte 4	Byte 4	W4	
Byte 5	Byte 5		W5
Byte 6	Byte 6	W6	
Byte 7	Byte 7		W7
Byte 8	Byte 8	W8	
Byte 9	Byte 9		W9
Byte 10	Byte 10	W10	
Byte 11	Byte 11		

The next table lists all of the data types and address ranges that are valid for the MPMPI driver:

Header Field Information			
Data Types	**Sample Syntax**	**Valid Range of Initial Addresses**	**Comments**
Flags	M:1	Varies according to the equipment	Logical Flags
Timers	T:2	Varies according to the equipment	Timer Values
Counters	Z:10 or C:10	Varies according to the equipment	Counter Values
Inputs	E:5 or I:5	Varies according to the equipment	Physical Input Values
Outputs	A:8 or Q:8	Varies according to the equipment	Physical Output Values
Data Blocks	DB2:1	Varies according to the equipment	Data Block Values, where: ▪ Number following DB(2) specifies the data block number ▪ Number following the colon specifies the word offset in the data block

- **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:

  <Format><AddressOffset>.<Bit> (for example: W10.2)

  Where:

  - **Format** defines how Studio treats the value read or written from/to the device. (**B**=Byte, **W**=Word, **DW = Dword, F**=Float)
  - **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 – 7) from the Byte address. This parameter is *optional*, and it is supported only when the format is Byte (B).

Sample Address Configuration		
Address on the Device	**Header Field**	**Address Field**
M (Word 5 = Byte 5 / Byte 6)	M:0	W5
	M:5	W0
	M:3	W2
M (Byte 5)	M:0	B5
	M:5	B0
	M:1	B4
M (Byte 6)	M:0	B6
	M:6	B0
	M:3	B3
T (33)	T:0	33
	T:30	3
	T33	0
C (3)	C:0	3
	C:3	0
	C:2	1
M (String 3, size 10)	M:0	S3:10
	M:3	S0:10
	M:1	S2:10
DB5 (Word 2 = Byte 2 / Byte 3)	DB5:0	W2
	DB5:2	W0
	DB5:1	W1
DB5 (Byte 2)	DB5:0	B2
	DB5:2	B0
	DB5:1	B1
DB5 (Byte 3)	DB5:0	B3
	DB5:3	B0
	DB5:2	B1
DB5 (Word 7 = Byte 7 / Byte 8)	DB5:0	W7
	DB5:7	W0
	DB5:4	W3
DB5 (Byte 7)	DB5:0	B7
	DB5:7	B0
	DB5:4	B3
DB5 (Byte 8)	DB5:0	B8
	DB5:8	B0
	DB5:4	B4

➲ **Attention:**

You must not configure a range of addresses in the same worksheet that is greater than the maximum block size (data buffer length) supported by the protocol: up to 1024 bytes in each *Standard Driver* Worksheet.

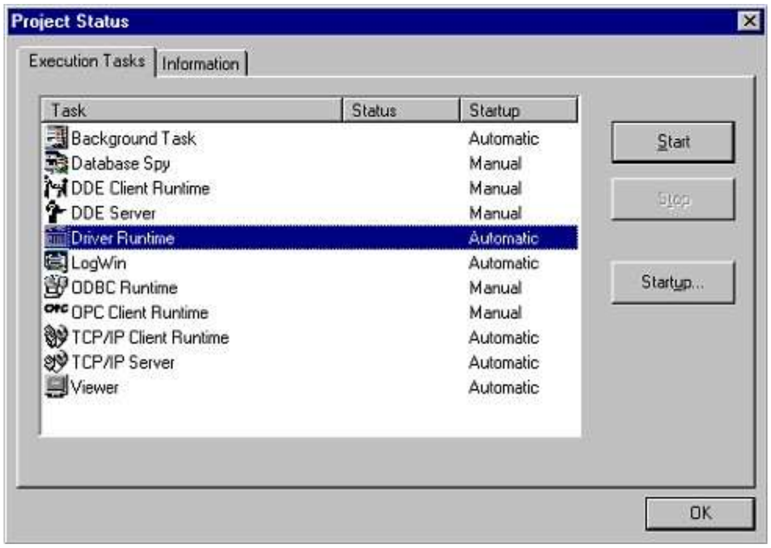
Executing the Driver

After adding the MPMPI 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 displays:



Task	Status	Startup
Background Task		Automatic
Database Spy		Manual
DDE Client Runtime		Manual
DDE Server		Manual
Driver Runtime		Automatic
LogWin		Automatic
ODBC Runtime		Manual
OPC Client Runtime		Manual
TCP/IP Client Runtime		Automatic
TCP/IP Server		Automatic
Viewer		Automatic

Project Status Dialog

2. Verify that the Driver Runtime task is set to **Automatic.**
   - If the setting is correct, click **OK** to close the dialog.
   - 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 MPMPIS 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 what kind of failure occurred.

Error Code	Description	Possible Causes	Procedure to Solve
10	Invalid PLC family	Wrong PLC model	The PLC model must be S7-200, S7-300 or S7-400. Note that a hyphen (-) is required.
11	Function not valid for this platform	Trying to run driver on a Windows NT/2000/XP platform	This driver runs only under Windows CE. The runtime can be started only in a Windows CE device with the MPI Abstraction Layer installed.
12	Could not create MPI Layer	Siemens SDK MPI Abstraction Layer not installed	Install the Siemens MPI Abstraction Layer AddOn. The abstraction layer is installed when you use Studio installation flash card. If you have not used the flash card, you will need to use a Siemens® tool such as ProSave® to install this AddOn.
13	Internal error	Function from MPI Abstraction Layer returned unrecognized value—probably the AddOn that is installed is newer than the one used for the driver development.	Contact Technical support.
14	Connection error	▪ Disconnected cables ▪ PLC turned off, in stop mode, or in error mode ▪ Wrong Station number	▪ Check the cable wiring. ▪ Check the PLC state—it must be RUN. ▪ Check the station number.
15	Address error	Wrong address for this data type	Check the PLC program, and be sure that you are trying to read-from/write-to a valid address.
16	Out of memory	The device does not have enough RAM memory for the driver internal buffer.	Add more memory to the device.
17	Fail to create request block	The function used to create a request block returned an error.	Contact Technical support.
18	Fail to add request	The function used to add a request block returned an error.	Contact Technical support.
19	Fail to send request	The function used to send a request block returned an error.	Contact Technical support.
20	Fail to get request	The function used to get a request block returned an error.	Contact Technical support.
21	Fail to remove request	The function used to remove a request block returned an error.	Contact Technical support.

Error Code	Description	Possible Causes	Procedure to Solve
22	Invalid block size	Worksheet exceeded 1024 operands. Note that the amount of operands is calculated based on the difference between the highest address and the lowest and not in the amount of tags in the worksheet.	Split your worksheet into smaller ones.
23	Exe Window could not be found	The executable file that communicates with the PLC was closed.	Restart the application. If the problem persists, try to restart the device.
24	Fail to create file map	Windows API function that creates a map file returned error.	Verify if you have enough RAM memory available. The file MAP requires 10KB.
25	Fail to send message to window.	The executable file that communicates with the PLC was closed.	Restart the application. If the problem persists, try to restart the device.
26	Executable file was not found	The file MPMPIExe.exe is not in the Drv folder.	Use studio tool to send the MPMPIExe.exe file to it.

⇨ **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 **Serial Communication** 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 enable the log at the unit (**Tools** → **LogWin**) and verify the **`celog.txt`** file created at the target unit.

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 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.

To test communication with Studio, it is recommended that you use the sample application provided rather than your new application.

If you must contact 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 is provided in the **`/COMMUNICATION EXAMPLES/<`*`Driver Name`*`>`** directory. We strongly recommend that you use this sample application to test the MPMPI driver before configuring your own customized application, for the following reasons:

- To better understand the information provided in the section of this document.
- To verify that your configuration is working satisfactorily.
- To certify that the hardware used in the test (device, adapter, cable, and PC) is working satisfactorily before you start configuring your own customized applications.

> ✍ **Note:**
> This application sample is not available for all drivers.

Use the following procedure to perform the test:

1. Configure the device’s communication parameters using the manufacturer’s documentation.
2. Open and execute the sample application.
3. Execute the *Viewer* module in Studio to display information about the driver communication.

> ⇨ **Tip:**
> You can use the sample application screen as the maintenance screen for your custom applications.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of Changes
A	1.00	Lidiane A. Moreira	14 Feb 03	First version
B	1.01	Lidiane A. Moreira	18 Mar 03	Modified the driver algorithm to improve application performance when the driver reports a communication error.
C	1.02	Lidiane A. Moreira	13 May 03	Changed to be able to read a group of bytes, instead of reading each tag.
D	1.03	Lidiane A. Moreira	11 Jun 03	Add Sleep between SendRequest and GetRequest (the driver was locking up the HMI).
E	1.04	Lourenço Teodoro	11 Sep 03	▪ Removed all the sleeps ▪ Split the driver into an executable file and a `.dll`. This change was done because the MPI Abstraction Layer needs to be started in the Main Thread. ▪ Changed the structure to read and send data in order to speed up the communication.