SPIIM Communication Driver

Driver for Serial Communication with
Devices Using SPI Protocol

Contents

Introduction

The SPIIM driver enables communication between the Studio system and devices using the SPI protocol communicating over Serial, according to the specifications discussed in this document.

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

- **Introduction**: Provides an overview of the SPIIM driver documentation.
- **General Information**: Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the SPIIM driver.
- **Installing the Driver**: Explains how to install the SPIIM driver.
- **Configuring the Driver**: Explains how to configure the SPIIM 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 SPIIM 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 Studio *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 SPIIM driver and the device.

The information is organized into the following sections:

- Device Characteristics
- Link Characteristics
- Driver Characteristics

Device Characteristics

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

- **Manufacturer**: Any device using the SPI (Society of the Plastics Industry) protocol communicating over Serial
- **Compatible Equipment**: Injection Molding Equipment

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

Driver Characteristics

The SPIIM driver is composed of the following files:

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

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

- Windows XP
- Windows Vista
- Windows 7

For a list of the operating systems used for conformance testing, see “Conformance Testing” on page 4.

Conformance Testing

The following hardware/software was used for conformance testing:

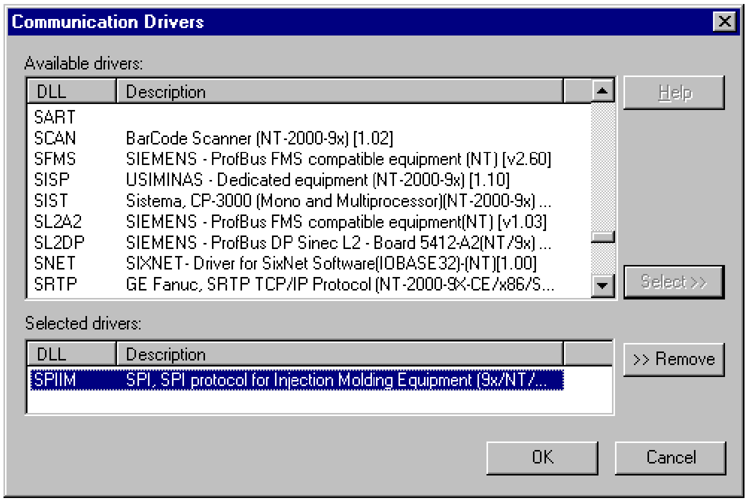
- **Equipment**: Communication Protocol Simulator (SPI Machinery Division)
- **Driver Configuration**:
  - **Baud Rate**: 9600
  - **Protocol**: RTU
  - **Data Bits**: 8
  - **Stop Bits**: 1
  - **Parity**: None
  - **COM Port**: COM1
- **Operating System (*development*):** Windows XP Professional SP3
- **Operating System (*runtime*):** Windows XP Professional SP3
- **Studio Version**: 6.1 + Service Pack 6
- **Driver Version**: 1.01

Installing the Driver

When you install Studio version 6.1 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 **SPIIM** driver from the *Available Drivers* list, and then click the **Select** button:



Available drivers:

DLL	Description
SART	
SCAN	BarCode Scanner (NT-2000-9x) [1.02]
SFMS	SIEMENS - ProfBus FMS compatible equipment (NT) [v2.60]
SISP	USIMINAS - Dedicated equipment (NT-2000-9x) [1.10]
SIST	Sistema, CP-3000 (Mono and Multiprocessor)(NT-2000-9x) ...
SL2A2	SIEMENS - ProfBus FMS compatible equipment(NT) [v1.03]
SL2DP	SIEMENS - ProfBus DP Sinec L2 - Board 5412-A2(NT/9x) ...
SNET	SIXNET- Driver for SixNet Software(IOBASE32)-(NT)[1.00]
SRTP	GE Fanuc, SRTP TCP/IP Protocol (NT-2000-9X-CE/x86/S...

Selected drivers:

DLL	Description
SPIIM	SPI, SPI protocol for Injection Molding Equipment (9x/NT/...

Communication Drivers Dialog

5. When the **SPIIM** 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 computer to enable communication between the host and the device.

> **Attention:**
> For safety reasons, you must use special precautions when installing the physical hardware. Consult the hardware manufacturer's documentation for specific instructions in this area.

Configuring the Driver

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

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

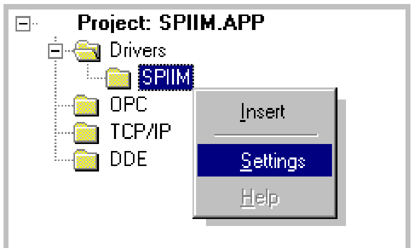
  Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the **Station**, **Header**, and **Address** fields, which are driver-specific. This document explains how to configure the **Station**, **Header**,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

> **Note:**
> For a detailed description of the Studio *STANDARD DRIVER SHEET*s, and information about configuring the standard fields, review the product's *Technical Reference Manual*.

Setting the Communication Parameters

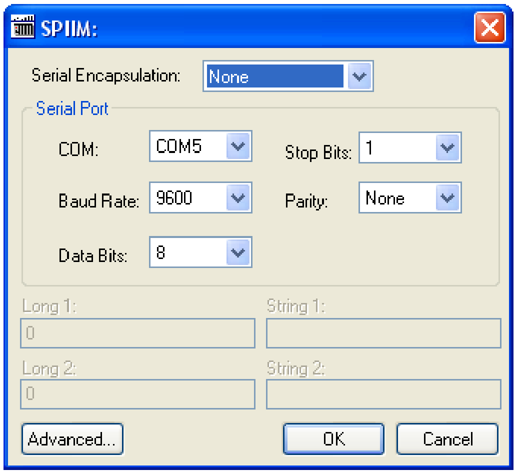
Use the following steps to configure the communication parameters, which are valid for all driver worksheets configured in the system:

1. From the Studio development environment, select the **Comm** tab located below the *Workspace* pane.
2. Click on the *Drivers* folder in the *Workspace* pane to expand the folder.
3. Right-click on the *SPIIM* subfolder and when the pop-up menu displays, select the **Settings** option:



Select Settings from the Pop-Up Menu

The *SPIIM: Communications Parameters* dialog displays:



Serial Encapsulation: None

COM: COM5

Stop Bits: 1

Baud Rate: 9600

Parity: None

Data Bits: 8

Long 1: 0

String 1:

Long 2: 0

String 2:

Communication Parameters Dialog

✍ **Note:**

The device must be configured with *exactly the same* parameters that you configured in the *SPIIM Communication Parameters* dialog.

4. Click the **Advanced** button on the *Communication Parameters* dialog to open the *Advanced Settings* dialog and configure the settings that are necessary.

✍ **Notes:**

- Do not change any of the other *Advanced* parameters at this time. You can consult the Studio *Technical Reference Manual* for information about configuring these parameters for future reference.
- 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, driver, and the host. You must be familiar with the DCE specifications before adjusting these configuration parameters.

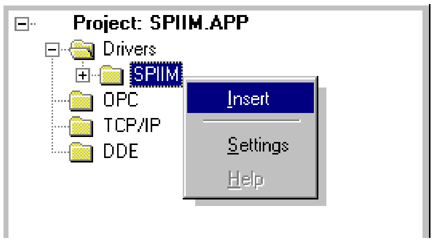
Configuring the Driver Worksheets

This section explains how to configure the *STANDARD DRIVER SHEETs* (or Communication tables) to associate application tags with the device addresses. You can configure multiple *Driver* worksheets — each of which is divided into a *Header* section and *Body* section.

Configuring the *STANDARD DRIVER SHEET*

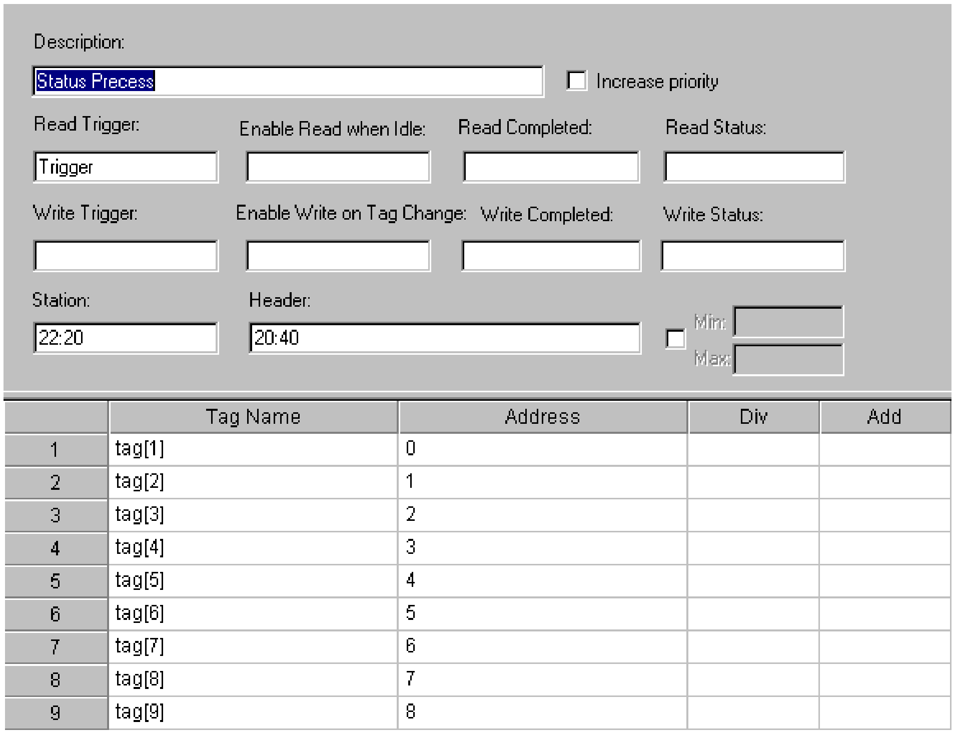
Use the following steps to create a new *STANDARD DRIVER SHEET*:

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 *SPIIM* subfolder.
3. When the pop-up menu displays, select the **Insert** option:



Inserting a New Worksheet

The STANDARD *DRIVER SHEET* displays (similar to the following figure):



Description:
Status Process
□ Increase priority

Read Trigger: Trigger
Enable Read when Idle:
Read Completed:
Read Status:

Write Trigger:
Enable Write on Tag Change:
Write Completed:
Write Status:

Station: 22:20
Header: 20:40
□ Min:
Max:

	Tag Name	Address	Div	Add
1	tag[1]	0		
2	tag[2]	1		
3	tag[3]	2		
4	tag[4]	3		
5	tag[5]	4		
6	tag[6]	5		
7	tag[7]	6		
8	tag[8]	7		
9	tag[9]	8		

STANDARD DRIVER SHEET

In general, all parameters on the *Driver* worksheet (except the **Station**, **Header**, and **Address** fields) are standard for all communication drivers, but they will not be discussed in this document. 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: Specify the device using the following syntax:
     <DEVID>:<ADD>

     Where:

     * **DEVID** is the Device Identification (values are in the range of **0x20–0xFF**)

     * **ADD** is the Device Address (values are in the range of **0x20–0xFF**)
   - **Header** field: Use the information in the following table to define the type of variables that will be read from the device These variables must comply with the following syntax:
     <CMD1>:<CMD2> (for example: **20:70**)

     Where:

     * **CMD1** is the Command #1 (values are in the range of **0x20–0xFF**)

     * **CMD2** is the Command #2 (values are in the range of **0x20–0xFF**)

     They are used to indicate a specific command to be executed by a specific device.

     After you edit the **Header** field, Studio checks the syntax to determine if it is valid. If the syntax is incorrect, Studio automatically inserts the default value in the **Header** field.

     Also, you can type a tag string in brackets **{Tag}** into the **Header** field, but you must be certain that the tag's value is correct and that you are using the correct syntax or you will get an **invalid Header** error.

     The following table lists all of the commands that are valid for the SPIIM driver:

CMD1	CMD2	Type	Description	Write
20	30	Numeric (F°)	Temperature at which the process is to be maintained	●
20	32	Numeric (F°)	This value in conjunction with the process set point, determine what temperature the high alarm will occur. It must always be positive.	●
20	40	16bit	Status Process	
20	48	16bit	Machine Mode	●
20	70	Numeric (F°)	Return the To Process Temperature	
20	72	Numeric (F°)	Return the From Process Temperature	
20	7C	Units F°	Return the to Process Dew Point Temperature	
20	80	Numeric (F°)	Value at which High Dewpoint Alarm will occur.	●

The commands shown on the table should be used on the Header field and match the POLL commands of the protocol. When attempting to write, the associated SELECT command is used internally, if available. In case there is no SELECT command associated, a write attempt will raise an error.

Address field: Use this field 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.

Header Field	Address Field	Description
20:30	0	Process Temperature Set Point
20:32	0	High Temperature Deviation Set Point
20:40	0	Processing
	1	Alarm, System
	2	Alarm, Process
	3	Alarm, Machines
	4	Alarm, High Temp.
	5	Alarm. Low Temp
	6	Alarm, Clogged Filter
	7	Alarm, Low Flow
	8	Alarm, High Dew Point
20:48	0	Machine On/Off
	1	Alarm Acknowledge
20:70	0	To Process Temperature
20:72	0	From Process Temperature
20:7C	0	To Process Dew Point Temperature
20:80	0	Dew Point Alarm Trigger

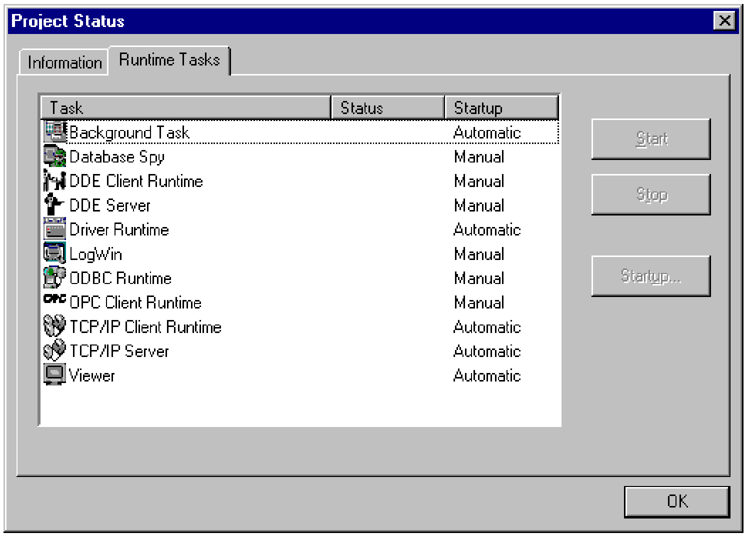
Executing the Driver

After adding the SPIIM 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		Manual
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 SPIIM 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
0	OK	▪ Communication without problems.	▪ None required.
1	INVALID COMMAND	▪ The Command configured in the Header field was not implemented in driver.	▪ Type a valid address.
2	INVALID ADD	▪ The value configured is out of range.	▪ Check if the value is inside of range.
3	INVALID DEVID	▪ The value configured is out of range.	▪ Check if the value is inside of range.
4	INVALID CMD1	▪ The value configured is out of range.	▪ Check if the value is inside of range.
5	INVALID CMD2	▪ The value configured is out of range.	▪ Check if the value is inside of range.
6	INVALID WRITE	▪ Attempted to write a tag associated with a header that does not support writing	▪ Check the worksheets ▪ Check the triggers and status tags associated with headers that do not support writing ▪ Check the table of headers for supported operations
-15	Timeout Start Message	▪ Disconnected cables. ▪ Device is turned off, in stop or in error mode. ▪ Wrong station number. ▪ Wrong RTS/CTS control settings.	▪ Check cable wiring. ▪ Check the PLC state – it must be RUN. ▪ Check the station number. ▪ Check the configuration. See *Studio Technical Reference Manual* for information about valid RTS/CTS configurations.
-17	Timeout between rx char	▪ PLC in stop mode or in error mode. ▪ Wrong station number. ▪ Wrong parity. ▪ Wrong RTS/CTS configuration settings.	▪ Check cable wiring. ▪ Check the PLC state – it must be RUN. ▪ Check the station number. ▪ Check the configuration. See *Studio Technical Reference Manual* for information about 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 **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 use the *Remote LogWin* of Studio (**Tools** → **Remote LogWin**) to get the log events from the target unit remotely.

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.

To test communication with Studio, 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**.
- **Studio Version**: To find this information, select **Help** → **About.**
- **Driver Version**: To find this information, read the full description of the driver on the *Communication Drivers* dialog.

- **Communication Log**: Displays in the Studio *Output* window (or *LogWin* window) when the driver is running. Be sure to enable the **Field Read Commands**, **Field Write Commands**, and **Serial Communication** for the *LogWin* window.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

Sample Application

You will find a sample application for drivers in the **`/COMMUNICATION EXAMPLES/<`*`Driver Name`*`>`** directory. We strongly recommend that you check if there is a sample application for this driver and use it to test the 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.

> ⇨ **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	Eric Vigiani	Aug/28/2003	Initial version
B	1.01	André Körbes	Aug/25/2010	Implemented new commands and support for write operations