

SIPPI Communication Driver

Driver for Serial PPI Communication
Between Studio and Siemens Devices

Contents

INTRODUCTION	2
GENERAL INFORMATION.....	3
DEVICE CHARACTERISTICS	3
LINK CHARACTERISTICS	3
DRIVER CHARACTERISTICS	3
CONFORMANCE TESTING	4
INSTALLING THE DRIVER	5
CONFIGURING THE DRIVER	6
SETTING THE COMMUNICATION PARAMETERS.....	6
CONFIGURING THE DRIVER WORKSHEET	7
MAIN DRIVER SHEET (MDS)	12
EXECUTING THE DRIVER	14
TROUBLESHOOTING	15
SAMPLE APPLICATION	19
REVISION HISTORY.....	20

Introduction

The SIPPI driver enables communication between Studio system and the Siemens devices, according to the specifications discussed in this publication.

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

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

The information is organized into the following sections:

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

Device Characteristics

This driver has been tested successfully with the following devices:

- f **Manufacturer:** Siemens
- f **Compatible Equipment:** Siemens S7-200 PLC communicating via PPI interface
- f **Siemens PLC Programmer Software:** Step 7

Link Characteristics

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

- f **Device Communication Port:** PPI Port
- f **Physical Protocol:** Serial
- f **Logic Protocol:** S7 PPI

Driver Characteristics

The SIPPI driver is composed of the following files:

- f **SIPPI.INI:** Internal driver file. *You must not modify this file.*
- f **SIPPI.MSG:** Internal driver file containing error messages for each error code. *You must not modify this file.*
- f **SIPPI.PDF:** Document providing detailed information about the SIPPI driver.
- f **SIPPI.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 `SIPPI.PDF` document.
- The SIPPI driver requires the `AGLINK.DLL` into the `/BIN`.

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

- ▮ Windows NT/2K/XP
- ▮ Windows CE

The SIPPI driver supports the following registers:

Register Type	Write	Read	Bit	Byte	Word	DWord	Float
M (Flags)	•	•	•	•	•	•	•
T (Timers)	•	•	–	–	•	–	–
Z or C (Counters)	•	•	–	–	•	–	–
E or I (Inputs)	–	•	•	•	•	•	•
A or Q (Outputs)	•	•	•	•	•	•	•
DB (Data Blocks)	•	•	•	•	•	•	•

Notes:

The V Register from S7-200 devices can be accessed with the DB1 Register.

Conformance Testing

The following hardware/software was used for conformance testing:

- f **Driver Configuration:**
 - PLC Program: Step 7
 - Protocol: S7 PPI
 - PC Adaptor: TECNATRON TCD S7-200

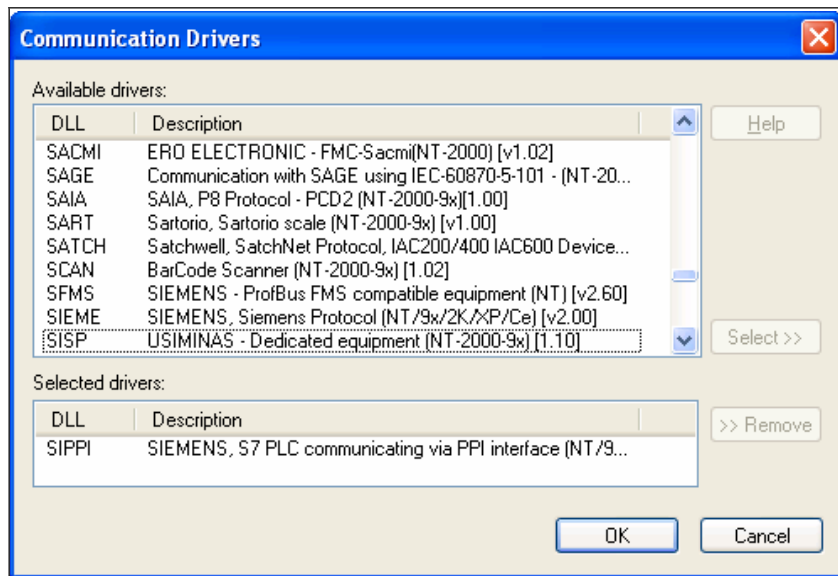
Driver Version	Studio Version	Operating System	Equipment
1.02	6.1	Windows XP	S7-200 with CPU226

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 **SIPPI** driver from the *Available Drivers* list, and then click the **Select** button:



Communication Drivers Dialog

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

⚠ Caution:
For safety reasons, you must use caution when installing the physical hardware. Consult the hardware manufacturer's documentation for specific installation instructions.

Configuring the Driver

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

- f Specifying communication parameters
- f 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.

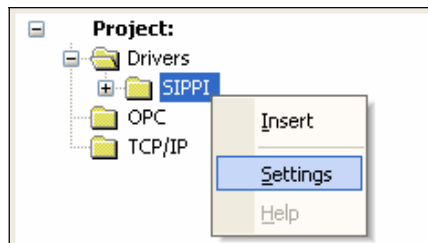
Notes:

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 *SIPPI* subfolder, and when the pop-up menu displays, select the **Settings** option:



Select Settings from the Pop-Up Menu

The *SIPPI: Communication Parameters* dialog displays:

SIPPI: Communication Parameters Dialog

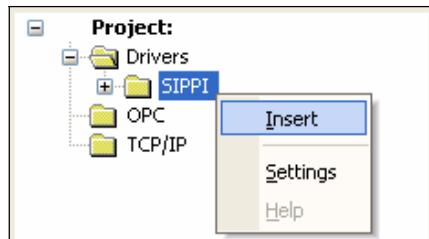
4. Verify the **COM** and **Baud Rate** settings, and change them if necessary.
5. Click **OK** to close the dialog.

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 *SIPPI* 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, we recommend configuring the communication addresses in sequential blocks to improve performance.

The *SIPPI.drv* dialog displays (similar to the following figure):

	Tag Name	Address	Div	Add
1	Tag1	B0		
2	Tag2	B1		
3	Tag3	B6		

SIPPI 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:

f **Station** field: Use this field to specify the station number.

Syntax:

<Station Number>

f **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.

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).

Notes:

The V Register from S7-200 devices can be accessed with the DB1 Register.

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

Header Address		SIPPI 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 SIPPI 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	DB1 : 1	Varies according to the equipment	Data block values, where: <i>f</i> Number following DB(1) specifies the data block number <i>f</i> Number following the colon specifies the word offset in the data block <i>f</i> V registers can be accessed with the DB1 Register.

f **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.
- ‰ **<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
DB1 (Word 2 = Byte 2 / Byte 3)	DB1:0	W2
	DB1:2	W0
	DB1:1	W1
DB1 (Byte 2)	DB1:0	B2
	DB1:2	B0
	DB1:1	B1
DB1 (Byte 3)	DB1:0	B3
	DB1:3	B0
	DB1:2	B1
DB1 (Word 7 = Byte 7 / Byte 8)	DB1:0	W7
	DB1:7	W0
	DB1:4	W3
DB1 (Byte 7)	DB1:0	B7
	DB1:7	B0
	DB1:4	B3

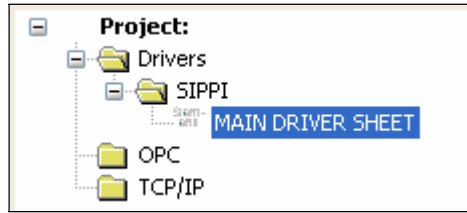
Sample Address Configuration		
Address on the Device	Header Field	Address Field
DB1 (Byte 8)	DB1:0	B8
	DB1:8	B0
	DB1:4	B4

⚠ Caution:

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. The maximum data buffer length for this driver is 1024 bytes in each *Standard Driver* worksheet.

Main Driver Sheet (MDS)

When the driver is inserted into the application, the *MAIN DRIVER SHEET* is automatically added to the driver folder.



Main Driver Sheet

The MAIN DRIVER SHEET provides a simple way to associate Studio tags to addresses in the PLC. Most of the MAIN DRIVER SHEET entries are standard for any driver. Refer to the Studio *Technical Reference Manual* about the configuration of the standard fields. The fields that require specific syntax for this driver are described below:

Description: <input type="text" value="MAIN DRIVER SHEET"/>							
Disable: <input type="text"/>							
Read Completed: <input type="text"/>		Read Status: <input type="text"/>		<input type="checkbox"/> Min: <input type="text"/>			
Write Completed: <input type="text"/>		Write Status: <input type="text"/>		<input type="checkbox"/> Max: <input type="text"/>			
	Tag Name	Station	I/O Address	Action	Scan	Div	Add
1	Tag1	2	M:B0	Read+Write	Always		
2	Tag2	2	M:B1	Read+Write	Always		
3	Tag3	2	C:1	Read+Write	Always		
4	Tag4	2	T:1	Read+Write	Always		

Main Driver Sheet

- f **Station** field: Use this field to specify the IP Address, Rack and Slot.
 Syntax:
<Station Number>
- f **I/O Address**: Address of each register from the PLC. The syntax used in this field is described below:
 - %o For Flags, Timers, Counters, Inputs and Outputs:
<Type> : <Format><Address> . <Bit> (for example: **M:1**)
 - %o For Data Blocks:
<Type><TypeGroup> : <Format><Address> . <Bit> (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).
- **<Address>** is the device address. This number always refers to the **Byte** address number.
- **<Format>** defines how Studio treats the value read or written from/to the device (**B**=Byte, **W**=Word, **DW**=Dword, **F**=Float).
- **<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**).

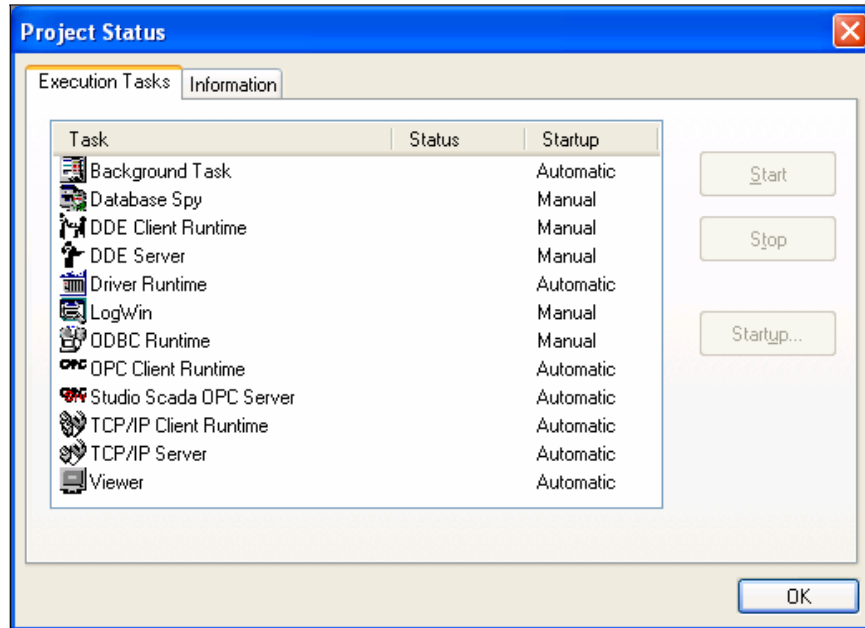
Executing the Driver

After adding the SIPPI 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:



Project Status Dialog

2. Verify that the *Driver Runtime* task is set to **Automatic**.
 - f If the setting is correct, click **OK** to close the dialog.
 - f 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 SIPPI 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
-15	Timeout Start Message
-17	Timeout between rx char.
0	No error
244	Function is not supported
245	Internal error, please check
246	Listed job number is invalid
247	At least one parameter for opening the device is invalid
248	No free space in the request queue
249	The necessary class cannot be initialized
250	The necessary memory cannot be allocated
251	Device is not open
252	Device was not found
253	Device is already in use, open, or update is not valid
254	Function is not valid
255	End of program request
256	A parameter was not in the defined range
508	Not all parameters can be changed because the adapter is already initialized
509	The program " AGLink_Config.EXE " cannot be started
510	The parameters were changed because of plausibility checks
511	Parameter length is not supported
512	Requested option is not available
752	Sending buffer too small for packet
753	Receiving buffer too small for packet
754	Timeout while waiting for DLE after sending the packet
755	Packet to be sent is not correct (length 0 or NULL pointer)
756	After STX arbitrary information was sent instead of DLE
757	After STX NAK was sent instead of DLE
758	Timeout after initialization conflict (both have high priority)
759	Timeout while waiting for DLE after sending STX

Error Code	Description
760	Initialization conflict
761	Wrong protocol status
762	Checksum error
763	Timeout while waiting for checksum
764	Information after DLE was not DLE or ETX
765	Timeout while waiting for packet information (ZVZ)
766	Timeout while waiting for beginning of packet (QVZ)
767	Wrong information received instead of STX
768	Timeout while waiting for STX at the beginning of the program
1019	Adapter is not initialized
1020	Unknown error message from adapter
1021	Wrong MPI baud rate
1022	The address code is higher than HSA
1023	Requested adapter address already exists
1024	Received packet has wrong content
1260	Type (of data) is not supported
1261	Access to object is not permitted
1262	Invalid address
1263	Context is not supported
1264	PLC sends no data
1265	Function protection level is not sufficient
1266	Context is not supported
1267	Information cannot be determined at the moment
1268	Unknown error message from PLC, please check
1269	Wrong size operands or selected range too large
1270	Wrong operating status of PLC
1271	Error while restarting the PLC
1272	Error while starting the PLC
1273	Wrong PLC operating status
1274	Internal error, please check
1275	No data available (for example, missing DB)
1276	Hardware error (for example, nonexistent peripheral equipment)
1277	Number of frame does not fit
1278	PLC was not found
1279	No additional connection possible
1280	No connection to the requested PLC
1523	DSR signal changed to 0 (modem disconnected)

Error Code	Description
1524	DCD signal changed to 0 (no carrier)
1525	No connection to remote terminal
1526	No modem found at the device
1527	Error during initialization of auto-answer
1528	Error during initialization of dial tone
1529	Error during initialization of selection procedure
1530	Error during initialization sequence 4
1531	Error during initialization sequence 3
1532	Error during initialization sequence 2
1533	Error during initialization sequence 1
1534	Error during basis initialization (AT&FE0V1)
1535	Modem cannot hang up
1536	General modem error
1786	CIF card not logged in on the logical ring (bus)
1787	A resource error exists
1788	Wrong firmware version of CIF card
1789	Wrong hardware version of CIF card
1790	Error in a device driver function
1791	Requested board not found
1792	Requested device driver not found
2038	Close received instead of ReadOK
2039	Timeout while reading IP
2040	Error while reading IP
2041	Close received instead of WriteOK
2042	Timeout while writing IP
2043	Error while writing IP
2044	Close received instead of ConnectOK (for example, wrong rack or slot number)
2045	Timeout while establishing IP connection
2046	Error while establishing IP connection
2047	Listed IP address invalid
2048	Socket cannot be opened

 **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 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**.
- f **Driver version** and **communication log**: Displays in the Studio *Output* window when the driver is running.
- f **Device model** and **boards**: Consult the hardware manufacturer's documentation for this information.

Sample Application

This driver does not have a sample application.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of Changes
A	1.00	Fabio H. Y. Komura	24 Sep 04	First version
B	1.01	Leandro Coeli	06 Sep 05	<ul style="list-style-type: none">– Implemented V type– Changes related to Windows CE 3.0 support
C	1.02	Lourenço Teodoro	28 Mar 06	Fixed problem in function GetBit
