SIPPI – *Driver Version 10.8*
Doc. Revision L – Apr 17, 2015

SIPPI Communication Driver

Driver for Serial PPI Communication
Between Studio and Siemens S7-200 Devices

Contents

Introduction

The SIPPI driver enables communication between the Studio system and the Siemens devices using PPI interfaces, 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:

- **Introduction**: Provides an overview of the SIPPI driver documentation
- **General Information**: Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the SIPPI driver
- **Installing the Driver**: Explains how to install the SIPPI 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 XP/7/8 environment.
  If you are unfamiliar with Windows XP/7/8, 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 using PPI interfaces.

The information is organized into the following sections:

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

Device Characteristics

This driver has been tested successfully with the following devices:

- **Manufacturer**: Siemens
- **Compatible Equipment**: Siemens S7-200 PLC Family
- **Siemens PLC Programmer Software**: Step7 MicroWin

Link Characteristics

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

- **Device Communication Port**: Port 0 or Port 1, configured for PPI
- **Physical Protocol**: Serial – RS-485
- **Logic Protocol**: S7 PPI

Driver Characteristics

The SIPPI driver is composed of the following files:

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

✍ **Notes:**

- All of the preceding files are installed in the **`/DRV`** subdirectory of the Studio installation directory.
- The SIPPI driver requests the **`AGLINK.DLL`** into the **`/BIN`**. This file is installed by default with the product

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

- Windows XP/7/8/2008/2012
- Windows Embedded Standard and Compact (WinCE 5/6/7)

The SIPPI driver supports the following registers:

Register Type	Write	Read	Bit	Byte	Word	DWord	Float	String
M (*Flags*)	●	●	●	●	●	●	●	●
T (*Timers*)	–	●	–	–	●	–	–	●
Z or C (*Counters*)	–	●	–	–	●	–	–	●
E or I (*Inputs*)	–	●	●	●	●	●	●	●
A or Q (*Outputs*)	●	●	●	●	●	●	●	●
V (*Variable Memory*)	●	●	●	●	●	●	●	●

Note:
The V Register from S7-200 devices can be accessed also with the DB1 Register

Conformance Testing

The following hardware/software was used for conformance testing:

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

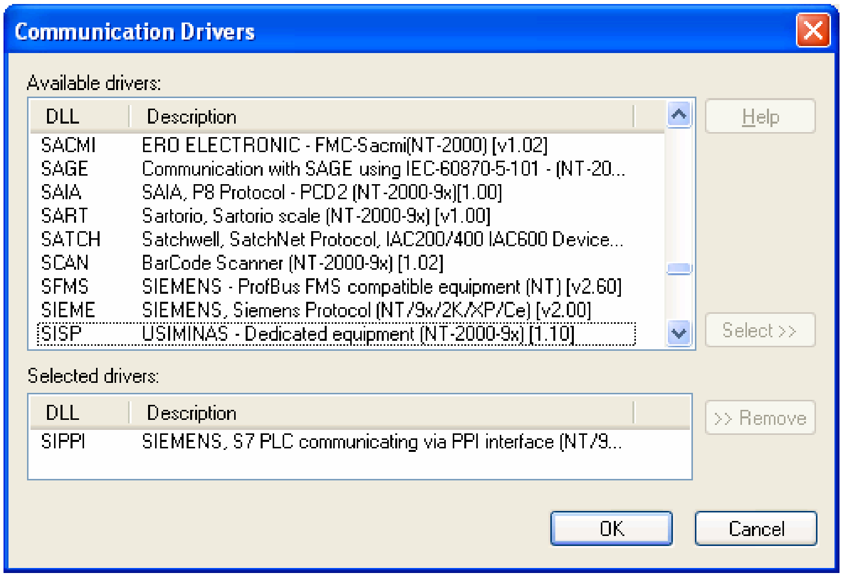
Driver Version	Studio Version	Operating System	Equipment
10.8	7.1 + SP2	Windows 7 Windows CE 5.0 ArmV4i	S7-200 with CPU226

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:



Available drivers:

DLL	Description
SACMI	ERO ELECTRONIC - FMC-Sacmi(NT-2000) [v1.02]
SAGE	Communication with SAGE using IEC-60870-5-101 - (NT-20...
SAIA	SAIA, P8 Protocol - PCD2 (NT-2000-9x)[1.00]
SART	Sartorio, Sartorio scale (NT-2000-9x) [v1.00]
SATCH	Satchwell, SatchNet Protocol, IAC200/400 IAC600 Device...
SCAN	BarCode Scanner (NT-2000-9x) [1.02]
SFMS	SIEMENS - ProfBus FMS compatible equipment (NT) [v2.60]
SIEME	SIEMENS, Siemens Protocol (NT/9x/2K/XP/Ce) [v2.00]
SISP	USIMINAS - Dedicated equipment (NT-2000-9x) [1.10]

Selected drivers:

DLL	Description
SIPPI	SIEMENS, S7 PLC communicating via PPI interface (NT/9...

Communication Drivers Dialog

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

➲ **Attention:**

For safety reasons, you must be careful 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:

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

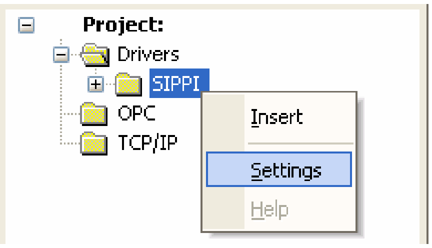
Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the **Station**, **Header** and **Address** fields, which are driver-specific. This document explains how to configure the **Station**, **Header**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

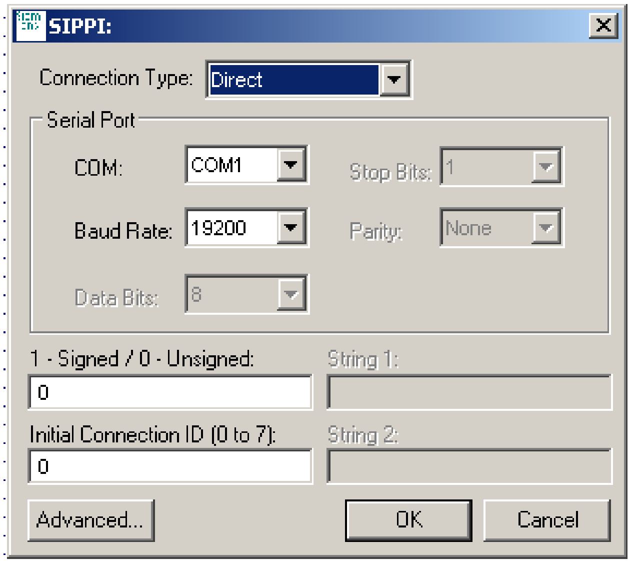
Use the following steps to configure the communication parameters, which are valid for all driver worksheets configured in the system:

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 *SIPPI* subfolder. When the pop-up menu displays, select the **Settings** option:



Select Settings from the Pop-Up Menu

The *SIPPI: Communication Parameters* dialog displays:



Connection Type: Direct

COM: COM1

Stop Bits: 1

Baud Rate: 19200

Parity: None

Data Bits: 8

1 - Signed / 0 - Unsigned: 0

String 1:

Initial Connection ID (0 to 7): 0

String 2:

SIPPI: Communication Parameters Dialog

Parameters	Default Values	Valid Values	Description
1-Signed **0-Unsigned Value**	0	0 or 1	This value will be default if you do not specify the Unsigned or Signed parameter for each format address. 0: Unsigned values 1: Signed values
Initial ID Connection	0	0 to 7	This value will be the initial ID Connection. This field must be configured only if you are going to use more than one instance of this driver. For each different instance,use different IDs.

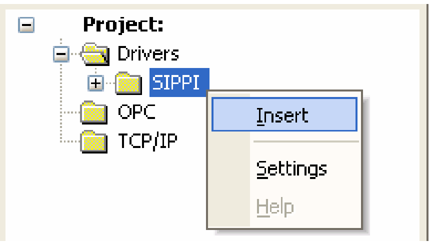
4. Verify the **COM** and **Baud Rate** settings, and change them if necessary.
5. Click **OK** to close the dialog.

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:

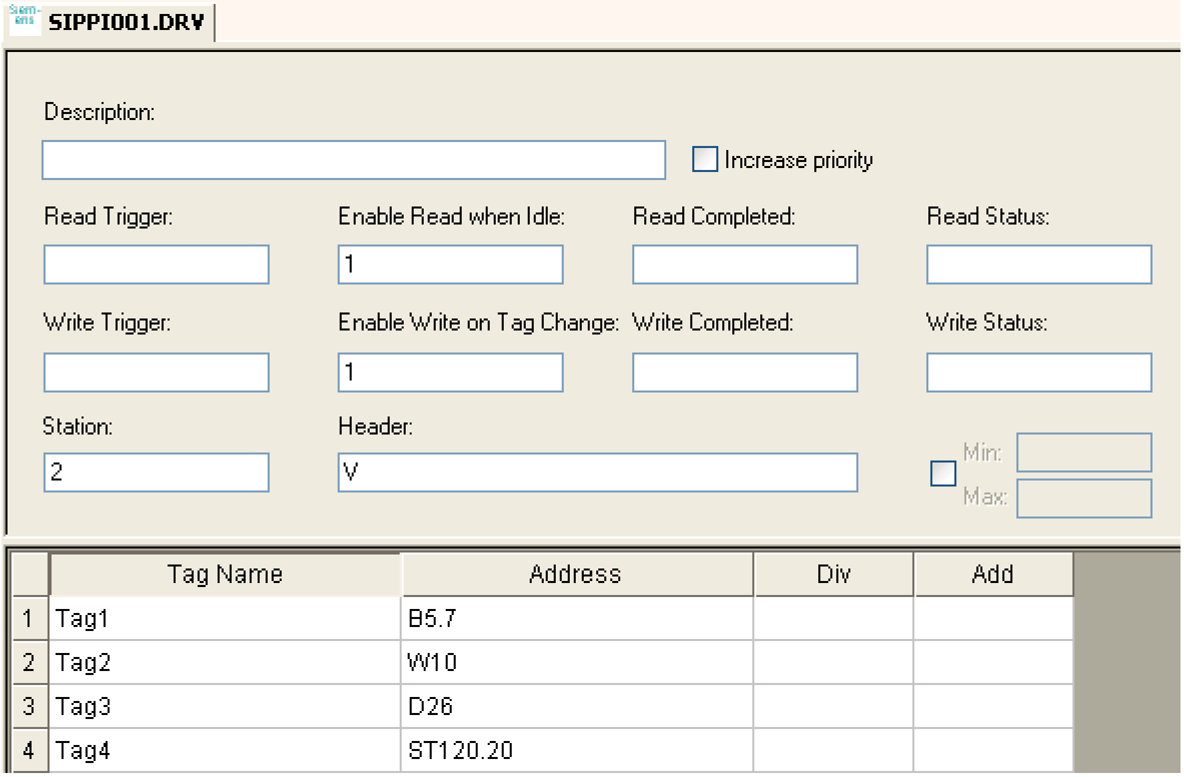
1. From the Studio development environment, select the **Comm** tab, located below the *Workspace* pane.
2. In the *Workspace* pane, expand the *Drivers* folder and right-click the *SIPPI* subfolder.
3. When the pop-up menu displays, select the **Insert** option:



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):



SIPPI001.DRV

Description:

□ Increase priority

Read Trigger:

Enable Read when Idle: 1

Read Completed:

Read Status:

Write Trigger:

Enable Write on Tag Change: 1

Write Completed:

Write Status:

Station: 2

Header: V

□ Min: Max:

	Tag Name	Address	Div	Add
1	Tag1	B5.7		
2	Tag2	W10		
3	Tag3	D26		
4	Tag4	ST120.20		

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:
   - **Station** field: Use this field to specify the PLC address (*PLC ID*). Valid values are 1– 255.

- **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: **`V:1`**) or

  <Type><TypeGroup>*.DB :*<AddressReference> (for example: **`DB1.DB:0`**)

  Where:

  – **Type** is the register type. (**M**=Flags, **T**=Timers, **Z** or **C**=Counters, **E** or I=Inputs, **A** or **Q**=Outputs, **DB**=Data Blocks and **V**=Variable)

  – **TypeGroup** is the group number of the configured register type (for Data-Block types only).

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

> **Note:**
> The V Register from S7-200 devices can be accessed also with the DB1 Register

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

Header Field Information			
Data Types	**Sample Syntax**	**Valid Range of Initial Addresses**	**Comments**
Flags	**M:1** or **M**	Varies according to the equipment	Logical Flags
Timers	**T:2** or **T**	Varies according to the equipment	Timer Values
Counters	**Z:10**, **C:10 or Z,C**	Varies according to the equipment	Counter Values
Inputs	**E:5**, **I:5** or **E,I**	Varies according to the equipment	Physical Input Values
Outputs	**A:8**, **Q:8** or **A,Q**	Varies according to the equipment	Physical Output Values
Variables	**V:1**	Varies according to the equipment	Variable Memory, equivalent to DB1

- **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:

  [Signed/Unsigned]*<Format><AddressOffset>.<Bit>* (for example: **X10.2**)

  <Format><AddressOffset>.<Len> (for example: **ST2.10, SST2.10**) – String format only

  Where:

  - ❑ **[*Signed / Unsigned*]** (*optional parameter used for integer values only*): If you do not specify this parameter, Studio inserts an integer value based on the parameters you set in the *Communication Parameters* dialog. Valid values are **S** (signed) and **U** (unsigned). Dword does not use **U** (unsigned).
  - ❑ **Format** defines how Studio treats the value read from or written to the device. (**X=**Bit, **B**=Byte or Bit, **W**=Word, **D** or **DW**=Dword, **F**=Float, **ST**=String, , **SST**= S7 String format).
  - ❑ **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 or Bit (**B** or **X**).
  - ❑ ***<Len>*** is the length to read or to write. It is in bytes. String format only.

Sample Address Configuration		
Address on the Device	**Header Field**	**Address Field**
M (Word 5 = Byte 5 / Byte 6 / String 7, Length 10)	M	W5
	M:5	W0
	M:3	W2
M (Byte 5)	M	B5
	M:5	B0
	M:1	B4
M (Byte 6)	M	B6
	M:6	B0
	M:3	B3
M (String 7, Length 10)	M	ST7.10
	M:7	ST0.10
	M:4	ST3.10
T (33)	T	33
	T:30	3
	T33	0
T(35)	T	35
	T:35	0
	T31	4
C (3)	C	3
	C:3	0
	C:2	1
C (4)	C	4
	C:4	0
	C:2	2
V (Word 2 = Byte 2 / Byte 3 / String 4 length 10)	V:0	W2
	V:2	W0
	V:1	W1
V (Byte 2)	V:2	B0
	V:1	B1
V (Byte 3)	V:3	B0
	V:2	B1
V (String 4 length 10)	V:0	ST4.10
	V:4	ST0.10
	V:2	ST2.10
V (Word 7 = Byte 7 / Byte 8 / Double Word 3)	V:0	W7
	V:7	W0
	V:4	W3
V (Byte 7)	V:0	B7
	V:7	B0
	V:4	B3

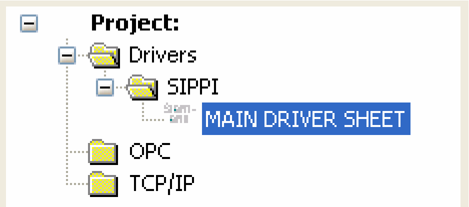
Sample Address Configuration		
Address on the Device	**Header Field**	**Address Field**
V (Byte 8)	V:0	B8
	V:8	B0
	V:4	B4
V (Double Word 3)	V:0	DW3
Input (Address 1, bit 4)	I:0 or E:0	X1.4
	I:1 or E:1	X0.4
	I:0 or E:0	B1.4
Input (Byte 1)	I:0 or E:0	B1
	I:1 or E:1	B0
Output(Address 1, bit 4)	Q:0 or A:0	X1.4
	Q:1 or A:1	X0.4
	Q:0 or A:0	B1.4
Output(Byte 1)	Q:0 or A:0	B1
	Q:1 or A:1	B0

➲ **Attention:**

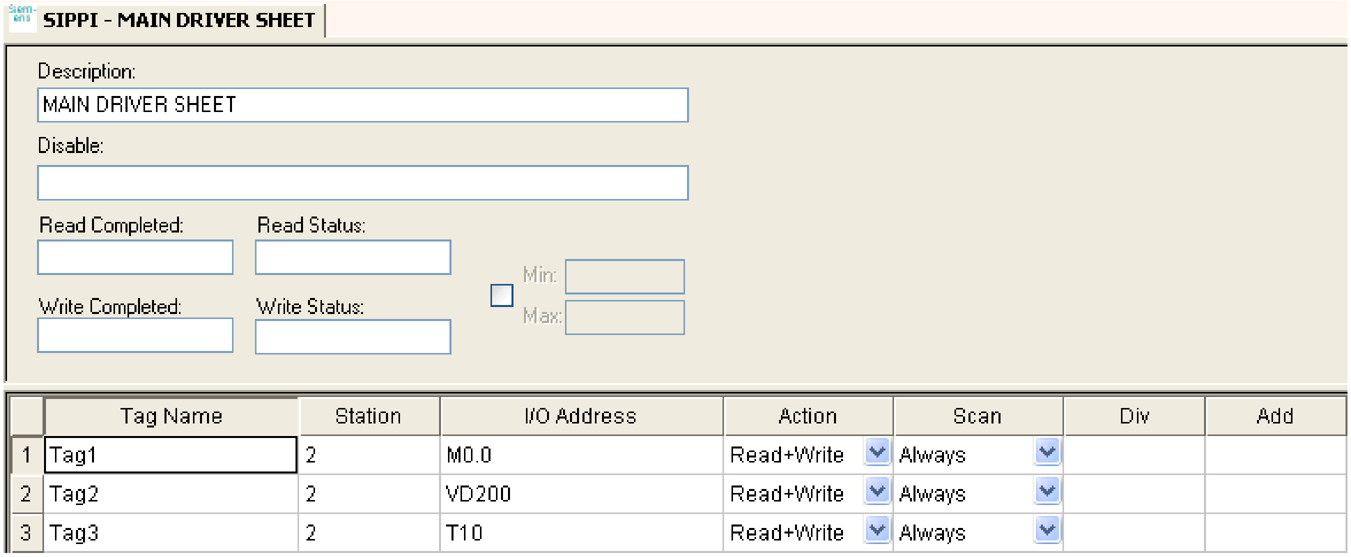
You must not configure a range of addresses greater than the maximum block size (data buffer length) supported by the protocol within the same worksheet. The maximum data buffer length for this driver is 1023 bytes per *Standard Driver* worksheet.

Main Driver Sheet (MDS)

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



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* for information about the configuration of the standard fields. The fields that require specific syntax for this driver are described below:



SIPPI - MAIN DRIVER SHEET

Description: MAIN DRIVER SHEET

Disable:

Read Completed: Read Status:

Write Completed: Write Status:

□ Min: Max:

	Tag Name	Station	I/O Address	Action	Scan	Div	Add
1	Tag1	2	M0.0	Read+Write	Always		
2	Tag2	2	VD200	Read+Write	Always		
3	Tag3	2	T10	Read+Write	Always		

- **Station** field: Use this field to specify the PLC address (*PLC ID*). Valid values are 1– 255.
- **I/O Address:** Address of each register from the PLC. The syntax used in this field is described below:
  - For Flags, Timers, Counters, Inputs, and Outputs:

    <Type>[Signed / Unsigned]<Format><Address>.<Bit> (for example: **MSW1**) or

    <Type>*[Signed/Unsigned]*<Format><Address>*.*<Bit> (for example: **MX10.2**)

    <Type><Format><Address>.<Len> (for example: **MST2.10**) – String format only

  - For Data-Blocks:

    <Type><TypeGroup>*[Signed/Unsigned]*<Format><Address>*.*<Bit> (for example: **VW1**)

    or ***<Type><TypeGroup>*.DB[Signed/Unsigned]*<Format><Address>*.*<Bit>*** (for example: **DB1.DBUB1**)

    <Type><<Format><Address>.<Len> (for example: **VST2.10**) - String format only

Where:

- **[*Signed / Unsigned*]** (*optional parameter used for integer values only*): If you do not specify this parameter, Studio inserts an integer value based on the parameters you set in the *Communication Parameters* dialog. Valid values are **S** (signed) and **U** (unsigned).. Dword does not use **U** (Unsigned).
- **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, **ST**=String)
- **TypeGroup** is the group number of the configured register type (for Data-Block types only). For the S7-200 CPUs, there is only 1 DB, DB1, which is equivalent to the **V** memory area
- **Address** is the device I/O address. This number always refers to the **Byte** address number except for Timers and Counter, in which this refers to the Timer or Counter number.
- **Format** defines how Studio treats the value read from or written to the device (**X**=Bit**, B**=Byte or Bit, **W**=Word, **D** or **DW**=Dword, **F**=Float, **ST**=String).
- **Bit** is the bit number (from 0 – 7) from the **Byte** address. This parameter is optional, and it is supported only when the format Byte or Bit (**B** or **X**).
- ***<Len>*** is the length in number of characters that will be read or written to. String format only.

✍ **Note:**

The syntax of Main Driver Sheet used on previous versions of the SIPPI driver required the colon on the address. This syntax is no longer accepted, however, existing applications will continue to work as expected.

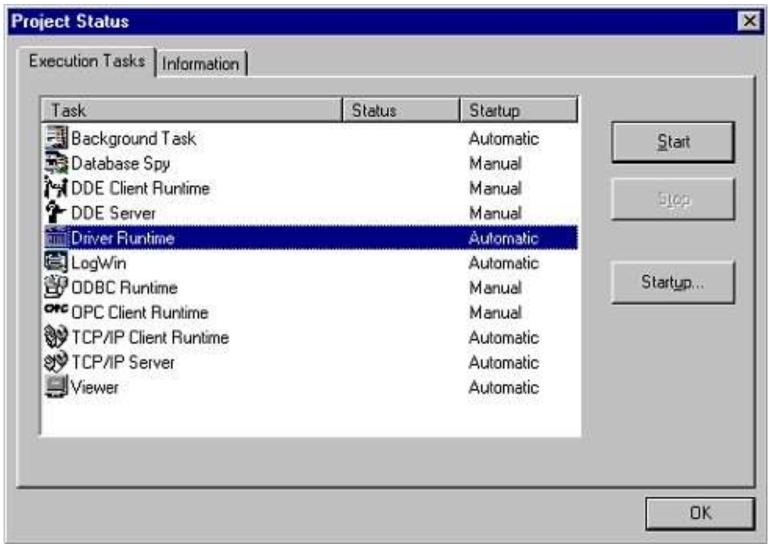
Executing the Driver

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:



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 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 the failure that occurred.

Error Code	Description
-33	Invalid INI file
-34	Invalid Address
-36	Invalid Data Type
-38	Invalid Station
-39	Invalid Block Size
0	OK
1	Error to initialize driver
2	Maximum number of connections was exceeded.
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 or 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

Error Code	Description
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
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 HAS
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

Error Code	Description
1279	No additional connection possible
1280	No connection to the requested PLC
1523	DSR signal changed to 0 (modem disconnected)
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.

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**.
- **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/SIPPI** directory.
We strongly recommend that you use this sample application to test the SIPPI driver before configuring your own customized application, for the following reasons:

- To better understand the information provided in 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	Fabio H. Y. Komura	24 Sep 04	First version
B	1.01	Leandro Coeli	01 Apr 03	Implemented V type Changes related to Windows CE 3.0 support
C	1.02	Lourenço Teodoro	15 Sep 03	Fixed problem in function GetBit
D	3.01	Rafael R. Fernandes	26 Feb 08	Implemented the Signed and Unsigned options Implemented String format Implemented "Initial ID Connection" field in Communication Parameters (to use more than one driver) Data types X and D (equivalent to Bit and DWord) were created. Driver was changed for Colon (:) to be optional in MAIN DRIVER SHEET New syntaxes to Data Blocks Operand were added.
E	10.1	Marcelo Carvalho	07 Jan 09	Updated driver version, no changes in the contents.
F	10.3	Lourenço Teodoro	30 Jun 09	Modified the driver to support SST format
G	10.4	Lourenço Teodoro	02 Dec 09	- Modified the driver to create less communication groups and optimize the communication - Fixed connection problem introduced with version 10.3
H	10.5	Fellipe Peternella	01 Apr 10	No changes specific for this driver
I	10.6	André Körbes	21 Jul 10	- Siemens drivers are allowed to coexist on the same application - Fixed problem with a specific CPU firmware that caused errors on timers and counters - Fixed virtual group separation
J	10.7	André Körbes	1 Jul 11	- Fixes on block size check
K	10.8	Ajay Anumalla	12 Mar 13	Updated driver version, no changes in the contents.
L	10.8	Andre Bastos	17 Apr 2015	Updated documentation only. No changes in the driver