HILDP Communication Driver

Profibus DP Master Driver for Hilscher or Synergetic Boards

Contents

Introduction

The HILDP driver enables communication between the Studio system configured as a Profibus DP Master and other Profibus devices using the Hilscher board interface, according to the specifications discussed in this document.

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

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

> **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 section explains how to identify all the hardware and software components used to implement communication between the HILDP driver, Studio, and other Profibus devices using the Hilscher board interface.

The information is organized into the following sections:

- Device Characteristics
- Link Characteristics
- Driver Characteristics

Device Characteristics

This driver was tested successfully with the following devices:

- **Manufacturer**: Siemens or any other Profibus DP-compliant PLC manufacturer
- **Compatible Equipment**: Any PLC that is compatible with the Profibus DP protocol
- **PLC Programming Software**: Varies according to manufacturer

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:

- **Network Board Manufacturer**: Hilscher/Synergetic
- **Network Board Model**: Profibus DP Slave Hilscher boards
  - CIF 30-DPM
  - CIF 104-DPM
- **Network Board Software**:
  - **Software to configure the board**: PLSyCon
  - **Software to test the communication with the board**: Synergetic CIFTest

Driver Characteristics

The HILDP driver is composed of the following files:

- **HILDP.INI**: Internal driver file. *You must not modify this file*.
- **HILDP.MSG**: Internal driver file containing error messages for each error code. *You must not modify this file*.
- **HILDP.PDF**: Document providing detailed information about the HILDP driver.
- **HILDP.DLL**: Compiled driver.
- **CIF Device Driver**: Hilscher Board Libraries

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 *HILDP.PDF* document.
- Studio’s HILDP driver requires the libraries installed with the CIF Device Driver to run properly. The HILDP driver requests the **CIF32DLL.DLL** (for the Windows NT/2000 operating system) and **CIFCEDLL.DLL** (for the Windows CE operating system) APIs, which are components of the CIF Device Driver. The CIF Device Driver should be included with the board.

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

- Windows 9x
- Windows 2000
- Windows NT
- Windows CE x86 ***only***

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

Other Software Requirements

In addition to the software discussed in the preceding sections, you must install the following software:

- For **Windows NT/2000**, you must install the following software:
  - Synergetic SyCon to configure the board
  - CIFTest to test the board

  Refer to the Synergetic documentation for information about installing and using the preceding software.
- For **Windows CE**, you must install the following Synergetic or Hilscher software on your CE unit and compiled for your processor:
  - **`CifCEdll.dll`**
  - **`CiflSA.dll`**
  - **`DrvSetup.exe`**
  - **`CifTest.exe`**

  The preceding **`.dll`** files are required to run the Studio HILDP driver, and the **`.exe`** files enable you to configure and test the board. When the **`CifTest.exe`** program runs the driver with no errors (particularly in the **`DevExchangeIO()`** function), the Studio HILDP driver will run successfully.

> **Attention:**
> You must take precautions when installing the physical hardware. Consult the hardware manufacturer’s documentation for installation instructions.

Conformance Testing

The following hardware/software was used for conformance testing:

- **Master**: PC Pentium II, 166 MHz, 64MB RAM with the Synergetic Board described in the WinNT Testing section.
- **Master**: Xycom Unit x86 under Windows CE with the Synergetic Board described in the WinCE Testing section.
- **Slave Equipment:** SIEMENS PLC S7-315-2DP

Configuration:

- **PLC Project**: Siemens Step 7 – Profibus
- **Synergetic Project**: Profi_1.pb
- **Baud Rate**: 1500 k
- **Protocol**: PROFIBUS DP

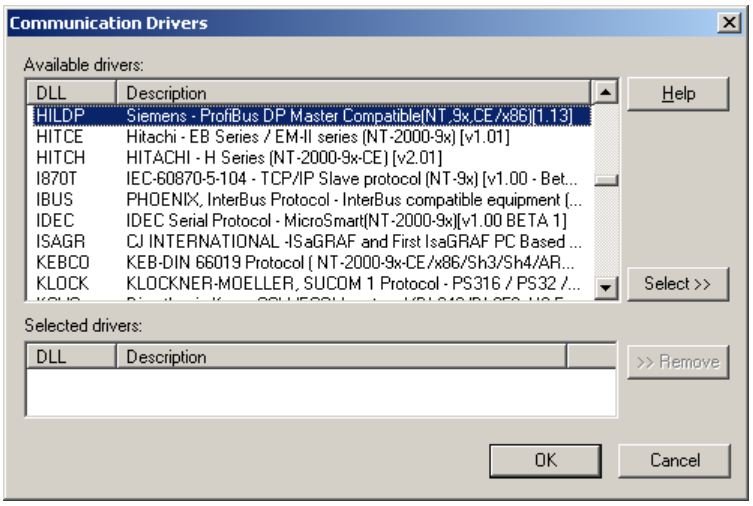
- **Hilscher/Synergetic Board Characteristics**:
  - **Model**: COM-DPM
  - **TYP**: CIF104DPSP
  - **GNR**: 9509003
  - **SNR**: 556
  - **DAT**: 11/98
- **Cable**: PROBIBUS Cable as described previously
- **Operating System** (*development*): Windows NT 4.0 + Service Pack 4, Windows 9x
- **Operating System** (*target*): Windows NT 4.0 + Service Pack 4, Windows CE v2.11, Windows 95
- **Studio Version**: 3.0
- **Driver Version**: 2.00
- **SyCon Version**:
  - **SyCon.exe** 2,1,4,0
  - **AboutDll.dll** 1,0,4,2
  - **DbAccess.dll** 1,1,0,1
  - **DBM32.dll** 2,8,0,9
  - **Cvt32.dll** 1,0,0,3
  - **DataSrv.dll** 1,1,0,6
  - **Ser32.dll** 1,0,0,5
  - **Cif32dll.dll** 2,0,2,1
  - **CifNtdll.dll** 2,0,2,1
  - **Cif95dll.dll** 2,0,2,1
  - **Funcdll.dll** 2,1,2,1
  - **StartUp.dll** 1,0,3,2
  - **Profibus.dll** 2,6,0,0
- **Step7 Version**: 5.0 + Service Pack 2 / Release k5.0 2.0
- **GSD information**:
  - **Master:**
    * **Vendor Name:** Hilscher GmbH
    * **Model Name:** COM-DPM/PKV20-DPM
    * **Identification Number:** 0x7506
    * **File Name:** Hil_7506.gsd
    * **Revision:** Version 2.002
    * **Hardware Revision:** Version 2.000
    * **Software Revision:** 1.020
    * **GSD Revision:** 1
  - **Slave:**
    * **Vendor Name:** Siemens
    * **Model Name:** S7-315-2DP
    * **Identification Number:** 0x802F
    * **File Name:** Hil_7504.gsd
    * **Device:** S7-315-2DP-AF03
    * **Revision:** Version 1.0
    * **Hardware Revision:** Version A1.0
    * **Software Revision:** Z1.0
    * **GSD Revision:** 1

Installing the Driver

When you install Studio version 3.0 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 **HILDP** driver from the *Available Drivers* list, and then click the **Select** button:



Communication Drivers Dialog

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

Configuring the Device

This section provides information about configuring Synergetic/Hilscher Master and Slave devices.

Master

To configure a Synergetic/Hilscher board you must use SyCon software. Consult the Synergetic/Hilscher SyCon software documentation for instructions.

> ⇨ **Tip:**
>
> When configuring the network, you must identify all the Slaves with which the Master will communicate and then download the configuration to the board. Sometimes, you must get the Slave GSD file to configure the Slaves. See “Conformance Testing” on page 4 for information about the master GSD file used for the conformance testing of this driver.

You can use both the SyCon and the CIF Test programs to test the board. For Windows CE,

- To test the board using the SyCon Test program, you may have to first configure the board on a NT/95 station with the SyCon software and then download the configuration to the Windows CE station.
- The CIF Test program from Synergetic is available from the manufacturer.

Slaves

The procedure for configuring Slaves for the device can vary significantly by manufacturer. Consult the device manufacturer’s documentation for instructions.

Configuring the Driver

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

- Specifying settings or communication parameters (there is only one configuration to the driver).
- Defining communication tags and controls in the *Communication* tables or *Driver* worksheet. There are two types of communication tables: **STANDARD TABLES** and the **MAIN DRIVER SHEET**.

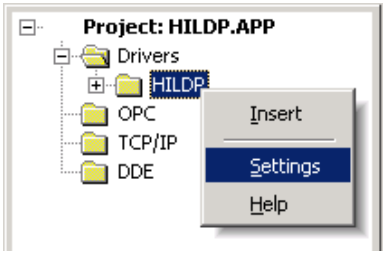
  Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the **Station**, **Header**, and **Address** fields, which are driver-specific. This document explains how to configure the **Station**, **Header**,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

> **Note:**
> For a detailed description of the Studio *Standard* and *MAIN Driver* worksheets, and information about configuring the standard fields, review the product's *Technical Reference Manual*.

Setting the Communication Parameters

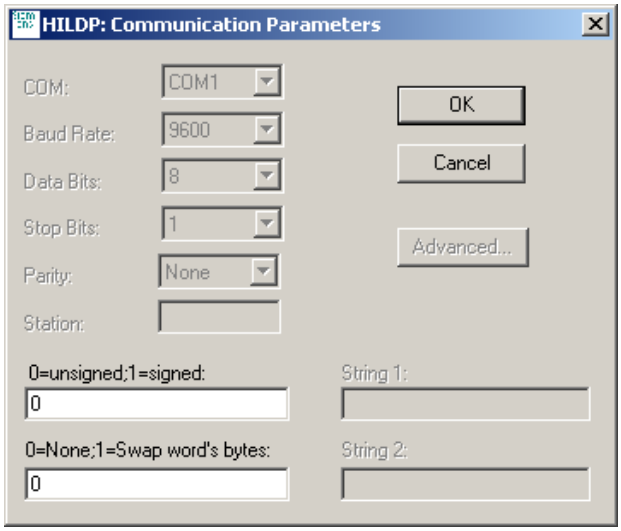
Use the following steps to configure the communication parameters, which are valid for all driver worksheets configured in the system):

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



Select Settings from the Pop-Up Menu

The *HILDP: Communications Parameters* dialog displays:



HILDP: Communication Parameters Dialog

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

Parameters	Default Values	Valid Values	Description
Station	0	0	Not used for this driver
0=None **1=Swap word's bytes**	0	0 or 1	**None:** Without Swap word's bytes **Swap word's bytes:** With Swap word's bytes
0=Unsigned **1=Signed**	0	0 or 1	**Unsigned:** From 0 to 255 byte values From 0 to 65535 word values **Signed:** From -128 to 127 byte values From -32768 to 32767 word values

> **Note:**
> No other parameters (serial settings) are required for this driver.

Configuring the Driver Worksheets

This section explains how to configure the communication tags in the *Standard* and *MAIN Driver* worksheets.

Configuring the Standard Driver Worksheet

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

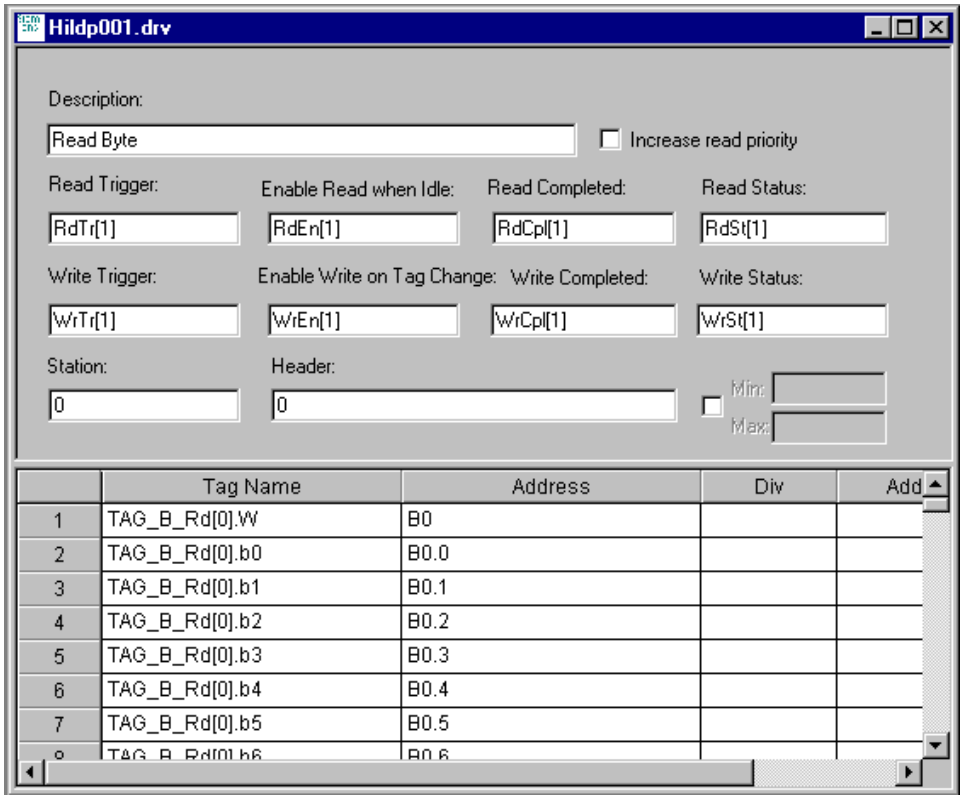
Use the following steps to create a new *Standard Driver* worksheet:

1. From the Studio development environment, select the **Comm** tab, located below the *Workspace* pane.
2. In the *Workspace* pane, expand the *Drivers* folder and right-click the *HILDP* 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 *<drivername>.drv* dialog displays (similar to the following figure):



HILDP Driver Worksheet

4. Use the following information to complete the **Station**, **Header**, and **Address** fields on this worksheet.

> **Note:**
> This publication explains how to configure the **Station**, **Header**, and **Address** fields. All of the other parameters on the driver worksheet are standard for all communication drivers. Consult the product's *Technical Reference Manual* for information about configuring these standard entries.

– **Station** field: Use this field to specify the board number. Valid values are 0 – 4 (*no default*).
– **Header** field (Default value is *0*): Use this field to define
  * The type of instructions that can be read from or written to the device
  * A reference to the initial address for inputs/outputs
  * The Network status (COMMSTATUS) and parameters (COMMPARMS).

  The following table lists all of the data types and address ranges that are valid for the Header Field:

Header Field Information			
Data Types	**Syntax**	**Valid Range of Initial Addresses**	**Comments**
Input/Output	**<*AddressReference*>** (Initial memory address (reference) to read/ write) For example, **0** or **15**	0 to 511	You must create one worksheet that reads to the device and another worksheet that writes to the device. You read from the Input Board image and write to the Output Board image, which means that although you are using the same numbers, you are using different Memory Addresses.
Communication Status	**COMMSTATUS**	–	Retrieves the communication status of this driver.
Communication Parameter	**COMMPARAM**	–	Retrieves the communication parameters for this driver.
Reset	**RESET**	–	Resets the Profibus Network. After executing the **COMMPARAM** command to change the communication parameters, you must reset the Network for your changes to take effect.

▪ **Address** field: Use this field to associate each tag in the worksheet to its address in the device. You type the tag's name into the **Tag Name** column and the tag's device address into the **Address** column, to enable the tag to read from and write to an address on the device. (See the following table for the valid Address configuration information.)

Address Configuration Sample		
Address on Master Device	**Header Field**	**Address Field**
IB 0	0	B0
IB 10	0	B10
IB 10	10	B0

Address Configuration Sample		
Address on Master Device	**Header Field**	**Address Field**
IB 10	5	B5
QB 1	0	B1
QB 217	200	B17
QW 0	0	W0
IW 100	50	W50

Entries in **Address** field must comply with the following syntax:

* **Input and Outputs: `<Format><AddressOffset>.<Bit>`**
* **COMMSTATUS: `<StatusAddress>`**
* **COMMPARAM: `<ParameterAddress>`**

Where:

* ***Format*** defines how Studio treats the value read or written from/to the device. (**B** = Byte, **W** = Word, **D** = Double Words, **F** = Float Point Words)
* ***AddressOffset*** is a parameter added to the ***AddressReference*** parameter (configured in the **Header** field) to compose the address for the memory to be read/written.
* ***Bit*** is the bit number (from 0 – 15) from the Word address. This parameter is *optional*.
* ***StatusAddress*** is the address of the status to be read from the Hilscher board.
* ***ParameterAddress*** is the address of the parameter to be read from the Hilscher board.

> ➲ **Attention:**
> - If you want to communicate with a Siemens device, be aware that these devices use an inverted L-H (Low to High) byte order within a word. This driver operates in HL (High-Low significance) byte order only.
> - This driver supports BIT reading only; it cannot execute BIT writing.

If you type **`COMMSTATUS`** in the Header field, you will be able to read the device's *Communication Status*.

You can type values from 0 to 8 into the **Address** field. The following table provides a description of these addresses:

Address	Parameter Name	Description
0.0	Global bits 0	**CONTROL-ERROR**: Parameterization error
0.1	Global bits 1	**AUTO-CLEAR-ERROR**: Device stopped communicating with all Slaves and reached the auto-clear end state
0.2	Global bits 2	**NON-EXCHANGE-ERROR**: At least one Slave has not reached the data exchange state and is not exchanging process data
0.3	Global bits 3	**FATAL-ERROR**: Because of heavy bus error, no further bus communication is possible
0.4	Global bits 4	**EVENT-ERROR**: Device detected bus short circuits. The number of detected events is fixed in the bus_error_cnt variable. Studio will set the bit when it detects the first event and will not delete any more bits.

Address	Parameter Name	Description
0.5	Global bits 5	**HOST-NOT-READY-NOTIFICATION**: Indicates whether the HOST program has set its state to operative or not If the bit is set, the HOST program is not ready to communicate
0.6	Global bits 6	**TIMEOUT-ERROR**: Device detected an overstepped timeout supervision time due to rejected PROFIBUS telegrams, which indicates bus short circuits when the Master interrupts communication. The number of detected timeouts is fixed in the bus_error_cnt variable. Studio will set the bit when it detects the first timeout is detected and will not delete any more bits.
0.7	Global bits 7	***Reserved***
1	DPM State	This variable represents the main state of the Master system. The following values are valid: ▪ 0 (00H): State OFFLINE ▪ 64 (40H): State STOP ▪ 128 (80H): State CLEAR ▪ 192 (C0H): State OPERATE
2	Err_rem_adr	Bits in the **Global_Bit** field are indicating errors in the network or in the Device itself have always a closer error description. In these cases, the **`Err_rem_adr`** variable represents the error source. The error can be detected by the Device itself (then the variable value is 255) or detected/reported by a network device (then the variable will contain the direct station address and the value can range from 0 to 125).
3	Err_event	To complete the error description, the **`Err_event`** variable delivers the corresponding error number to the error source. For a list of all possible error numbers, see the table on page 14.
4	Bus_error_cnt	This variable contains the number of heavy bus errors, such as bus short circuits.
5	Time_out_cnt	This variable contains the number PROFIBUS telegrams that were rejected due to heavy bus errors.
6.0 to 6.127	Slave cfg	This variable is a 16 byte-field containing the parameterization state of each Slave station. The Slave station number is indicated after the period (from 0 to 127). ▪ If the **`Sl_cfg`** bit of the corresponding Slave is logical ***1***, the Slave is configured in the Master, and serviced in its states. ▪ If the **`Sl_cfg`** bit of the corresponding Slave is logical ***0***, the Slave is not configured in the Master.
7.0 to 7.127	Slave state	This variable is a 16 byte-field containing the state of each Slave station. The Slave station number is indicated after the period (from 0 to 127). ▪ If the **`Sl_state`** bit of the corresponding Slave station is logical ***1***, the Slave and the Master are exchanging their I/O data. ▪ If the **`Sl_state`** bit of the corresponding Slave station is logical ***0***, the Slave and the Master are not exchanging their I/O data.
8.0 to 8.127	Slave diag	This variable is a field of 16 bytes containing the diagnostic bit of each Slave. ▪ If the **`Sl_diag`** bit of the corresponding Slave station is logical ***1***, latest Slave diagnostic data is available in the internal diagnostic buffer. ▪ If the **`Sl_diag`** bit of the corresponding Slave station is logical ***0***, because no values changed in this buffer during the last diagnostic buffer read access of the HOST. The values in the **`Sl_diag`** variable are valid only if the Master station runs the main state OPERATE.

The following table contains information about the relationship between the Slave state bit and Slave diag bit:

	sl_state = 0	sl_state = 1
sl_diag = 0	▪ No DataIOExchange between Master and Slave. ▪ Slave may not be configured or responsive.	▪ Slave is present on the bus. ▪ DataIOExchange between Master and Slave.
sl_diag = 1	▪ Master and corresponding Slave are not exchanging their I/O data. ▪ Master holds newly received diagnostic data in the internal diagnostic buffer.	▪ Slave is present on the bus. ▪ Master and corresponding Slave are exchanging their I/O data. ▪ Master holds newly received diagnostic data in the internal diagnostic buffer.

The following error numbers are valid for `Err_event`, if `Err_rem_adr` is 255:

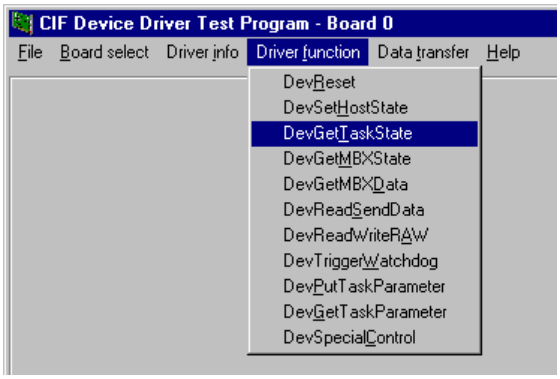
err_event	Description	Error Source	Help
No Mistakes Display	1	2	3
50	USR_INTF-Task not found	Device	Contact Synergetic technical support
51	No global data-field	Device	Contact Synergetic technical support
52	FDL-Task not found	Device	Contact Synergetic technical support
53	PLC-Task not found	Device	Contact Synergetic technical support
54	Non-existing Master parameters	Device	Execute database download again
55	Faulty parameter-value in the Master parameters	Project Planning	Contact technical support
56	Non-existing Slave parameters	Project Planning	Execute database download again
57	Faulty parameter-value in a Slave parameters data file	Project Planning	Contact technical support
58	Double Slave address	Project Planning	Check projected addresses
59	Projected send process data-offset address of a participant outside the allowable border of 0-255		
Project Planning	Check projected addresses		
60	Projected receive process data-offset address of a participant outside the allowable border of 0-255	Project Planning	Check projected addresses
61	Slave data areas are overlapping in the send process data	Project Planning	Check projected addresses
62	Slave data areas are overlapping in the receive process data	Project Planning	Check projected addresses
63	Unknown process data handshake	warmstart	Check warmstart parameters
64	Free RAM exceeded	Device	Contact Synergetic technical support
65	Faulty Slave parameter data sets	Project Planning	Contact Synergetic technical support

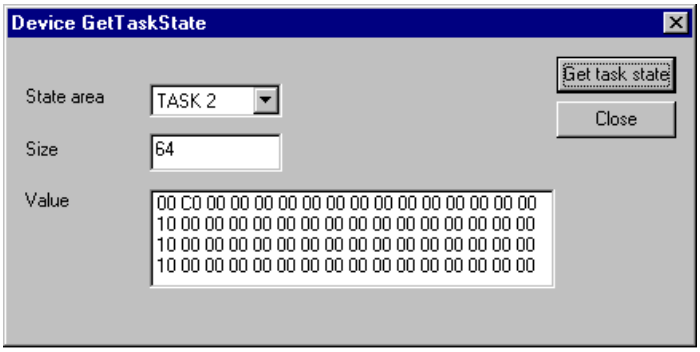
err_event	Description	Error Source	Help
202	No free segments for the treatment	Device	Contact Synergetic technical support
212	Faulty reading of a database	Device	Execute download of data base again
213	Faulty structure-surrender to operating system	Device	Contact Synergetic technical support

The following error numbers are valid for `Err_event`, if `Err_rem_adr` is unequal 255:

err_event	Description	Error Source	Help
2	Station reports overflow	Master telegram	Check length of Slave configuration or parameter data
3	Master request function is not activated in the station	Master telegram	Check Slave if PROFIBUS-DP norm compatible
9	No answer-data, although Slave must respond with data	Slave	Check station configuration data and compare it to the physical I/O data length
17	No station response	Slave	Check bus cable, check bus address of Slave
18	Master not into the logical token ring	Device	▪ Check Master FDL-Address or highest-station-Address of other Master systems ▪ Examine bus cabling to bus short circuits
21	Faulty parameter in request	Master telegram	Contact Synergetic hotline

The **COMMSTATUS** brings the values equivalent to the CIF Driver's **`DevGetTaskState`** test function:

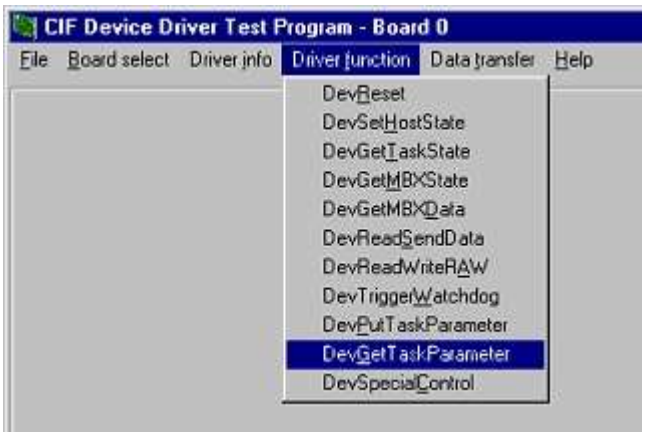


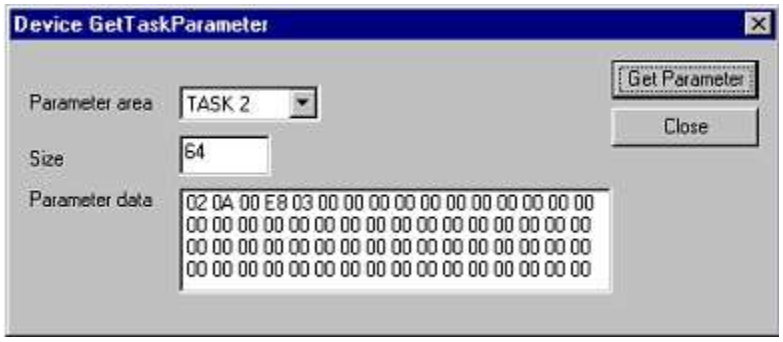


Typing **COMMPARAM** in the **Header** field and typing a value from 0 to 8 into the **Address** field enables you to read and write the Communication Parameters. The following table describes the Address parameters:

Address	Parameter Name	Description
0	Cycle Time	The **CycleTime** parameter fixes the minimum Slave interval time in multiples of 1msec. The Master must wait until it starts the next DP-process data exchange for all Slaves. If the Address value is zero, the DP data exchange cycle is done as fast a possible. (0-255ms)
1	Data format	The **Data Format** parameter changes the storage format of word-oriented process data from MSB/LSB (*Intel*) to the LSB/MSB (*Motorola*) convention and vice versa. (**`0`** = Intel and **`1`** = Motorola)
2	WatchDogTime	The **usWatchDogTime** parameter fixes the time in multiples of 1msec. The Device must supervise the HOST program if the Device started the HOST-watchdog functionality at least once.

This Header brings the values equivalents to the CIF Driver **`DevGetTaskParameter`** test function and **`DevPutTaskParameter`** test functions:

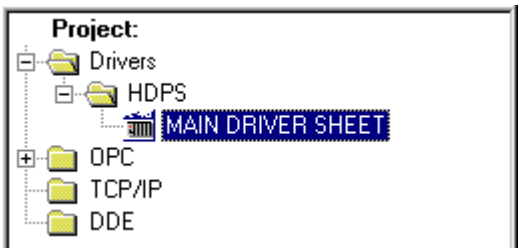




For more information, refer to the Hilscher Profibus DP Master protocol interface manual.

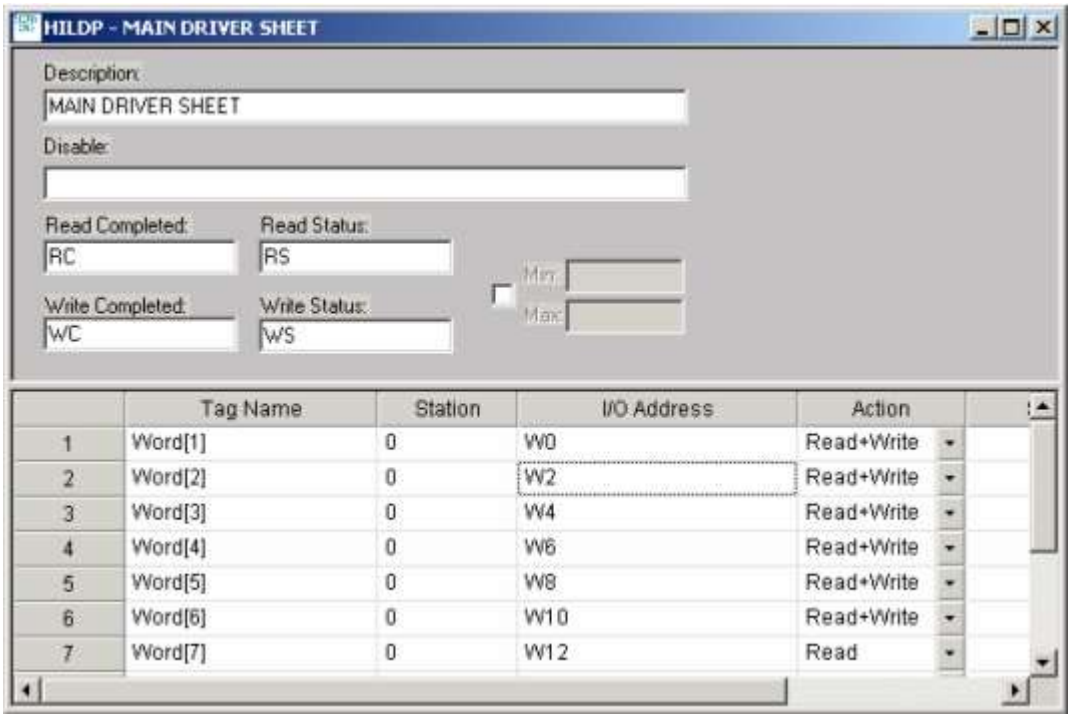
Configuring the MAIN Driver Worksheet

When you add the HILDP driver to your application, Studio automatically adds a MAIN DRIVER SHEET to the driver folder:



MAIN Driver Worksheet

The MAIN Driver Worksheet (similar to the following figure) enables you to easily associate Studio tags to addresses in the PLC.



HILDP MAIN Driver Worksheet

Note: Most of the MAIN DRIVER SHEET parameters are standard for all drivers. Instructions for configuring these standard parameters are provided in the Studio *Technical Reference Manual*. This section provides instructions for configuring the **Station** and **I/O Address** parameters, which are specific to this driver:

- **Station:** Type the Board number.
- **I/O Address:** Type the address of each register in the PLC using the following syntax (for Input and Outputs):

  <*Format*><*AddressOffset*>.<*Bit*> (for example, *W23.1*)

  Where:

  – ***Format****:* Type one of the following:

    * **W** to configure the values as words
    * **D** to configure the values as double words
    * **B** to configure the values as bytes
    * **F** to configure the values as Float Point words

  – ***AddressOffset***: Type the offset address. Studio adds this parameter to the **AddressReference** parameter (configured in the **Header** field) to compose the address of the memory to be read/written.

  – ***Bit*** (*optional*): Type the bit number (from 0 to 15) from the word address.

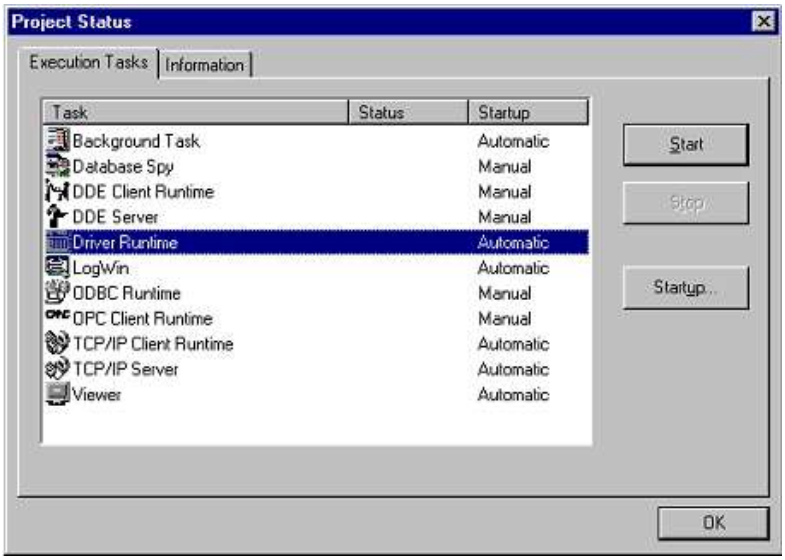
> ➲ **Attention:**
> This driver supports BIT reading only; it cannot execute BIT writing.

Executing the Driver

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

Troubleshooting

If the HILDP 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	Not Required.
1	Invalid Station	**Station** field contains a nonexistent board address	Check the Driver Worksheet related to this error. If the configured station is correct, do not change it.
2	Invalid Header	Invalid Header or tag provided in the **Header** field has an invalid configuration	Type a valid Header or tag value into the **Header** field. (See page 14 for a list of valid Header tag values.)
3	Invalid Address	Invalid address typed in the Driver Worksheet	Check the addresses configured for Driver Worksheet reporting this error. (See page 11 for a list of valid addresses for each one of the valid Headers.)
4	Block size error	**Address** field greater than 512	Maximum address offset is 511. Correct the Driver Worksheet.
5	Protocol error	Protocol error	Run the **COMMSTATUS** function to check the protocol error.
6	CheckSum error	Protocol error	Run the **COMMSTATUS** function to check the protocol error.
7	Error opening the driver	Problems initializing the board	Board might not be configured or connected. Run the *SyCon*, *DrvSetup*, or *CifTest* program to detect the board configuration error.
8	Error sending message	Protocol or bus error	Run the **COMMSTATUS** function to check the protocol error.
9	Error receiving message	Protocol or bus error	Run the **COMMSTATUS** function to check the protocol error.
10	Invalid offset	Address field greater then 512	Maximum address offset is 511. Correct the Driver Worksheet.
19	Reset error	Error during Reset operation. Can be caused by a conflict when using device interrupts or, because the timeout period can differ between Fieldbus protocols, you must be experiencing timeout problems	Contact your Technical Support representative and describe the error.
20	**COMMSTATUS** invalid address error	**Address** field outside the acceptable ranges (between 1 and 8 or 0.0 and 0.7)	Type a valid address in the **COMMSTATUS** header case.
21	**COMMPARAM** invalid address error	**Address** field outside the acceptable ranges (between 0 and 2)	Type a valid address in the **COMMPARAM** header case.
25	Bit address error in a	Bit in **Address** field outside the acceptable	Type a valid address between Wx.0 and Wx.16.

Error Code	Description	Possible Causes	Procedure to Solve
	word	range (between 0 and 15)	
26	Bit address error in a byte	Bit in **Address** field outside the acceptable range (between 0 and 7)	Type a valid address between Bx.0 and Bx.7.
27	Global bit address error	Global bit address out of the range 0.0 and 0.7, in the COMMSATUS header case	Type a valid address between 0.0 and 0.7.
28	Slave bit address error	Slave configuration, Slave state, or Slave diag address with the slave out of range 0 and 127 (6.0 – 6.127, 7.0 – 7.127, 8.0 – 8.127), in the COMMSTATUS header case	Type a valid address between 6.0 – 6.127, 7.0 – 7.127, or 8.0 – 8.127
29	**COMMSTATUS** error	Read/write operation returned an error	Run the **COMMSTATUS** command and check the error
30	Invalid bit operator error	Trying to write in a bit of a word or a byte	Driver cannot write individual bits; only words and bytes
31	Error while exchanging messages	Error reading an Input or writing an Output. These errors can be caused by: ▪ Timeouts (where device needs more time then defined by the driver) ▪ Wrong (or no) interrupt selected	*Interrupt on the device and in the driver registration must be the same!* If interrupt is already used by another PC component, contact your Technical Support representative.
32	Write parameter error	Wrong parameter in the **COMMPARAM** header writing case	▪ Check the valid parameters for this operation. ▪ Consult the device manufacturer's documentation for valid parameter values.
-15	Timeout waiting to start a message	▪ Disconnected cables ▪ PLC turned off, or in stop or error mode ▪ Wrong station number ▪ Wrong RTS/CTS configuration settings	▪ Check the cable wiring ▪ Check the PLC state--It must be RUN ▪ Check the station number. ▪ Check the configuration. See the *Studio Technical Reference Manual* for valid RTS/CTS settings.
-17	Timeout between rx chars	▪ PLC in stop or error mode ▪ Wrong station number ▪ Wrong parity ▪ Wrong RTS/CTS configuration settings	▪ Check the cable wiring ▪ Check the PLC state--It must be RUN ▪ Check the station number. ▪ Check the configuration. See the *Studio Technical Reference Manual* for valid RTS/CTS settings.

⇨ **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 cannot 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

You will find a sample application in the **/COMMUNICATION EXAMPLES/HILDP** directory. We *strongly* recommend that you use this sample application to test the HILDP 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

Version	By	Date	Description of changes
1.06	Sergio A. Poon	09-dec-1999	First driver version Driver available for Windows CE
1.07	Roberto V. Junior	04-may-2000	Fixed bug of first address value.
1.08	Roberto V. Junior	04-jul-2000	Fixed bug of Signed and Unsigned data format.
1.09	José L. Teodoro	02-Oct-2001	Inserted double words Implemented the Run-time load to Hilsher Library Inserted MAIN DRIVER SHEET functionality
1.10	Roberto V.Junior	07-dec-2001	Fixed bug of Check Task Status Included COMMSTATUS and COMMPARAM in MainDriverSheet
1.11	Roberto V.Junior	11-jan-2002	Fixed bug of COMMSTATUS error after executed a write command.
1.12	Eric Vigiani	30-sep-2002	Included support for Float Point data type
1.13	Andre Bastos	15-jan-2003	Memory Size increased from 512 to 3584 (new boards)

▪