

PPCBR Communication Driver

Driver for TCP/IP Communication
with PPC Devices Using SCP API

Contents

INTRODUCTION2

GENERAL INFORMATION.....3

 DEVICE CHARACTERISTICS3

 LINK CHARACTERISTICS.....3

 DRIVER CHARACTERISTICS3

 CONFORMANCE TESTING4

INSTALLING THE DRIVER5

CONFIGURING THE DRIVER6

 SETTING THE COMMUNICATION PARAMETERS6

 CONFIGURING THE DRIVER WORKSHEETS8

 DEVICE CONFIGURATION 14

EXECUTING THE DRIVER 16

TROUBLESHOOTING 17

SAMPLE APPLICATION 19

REVISION HISTORY..... 20

Introduction

The PPCBR driver enables communication between the Studio system and PPC devices using the SCP API over TCP/IP, according to the specifications discussed in this document.

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

- **Introduction:** Provides an overview of the PPCBR driver documentation.
- **General Information:** Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the PPCBR driver.
- **Installing the Driver:** Explains how to install the PPCBR driver.
- **Configuring the Driver:** Explains how to configure the PPCBR 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 PPCBR 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 PPCBR driver and the PPC 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:** Bosch RexRoth
- **Compatible Equipment:**
Bosch Rexroth Indramat PPC-R02. 2N-N-L2-T2-N N-FW
- **Device Runtime Software:** None

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

Link Characteristics

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

- **Device Communication Port:** Ethernet Port
- **Physical Protocol:** Ethernet/TCP-IP
- **Specific PC Board:** Any TCP/IP Adapter (Ethernet board)

Driver Characteristics

The PPCBR driver is composed of the following files:

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

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

- Windows 9x
- Windows 2000
- Windows NT

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

The PPCBR driver supports the following types:

Item Type	Write	Read
!I2	•	•
!I4	•	•
!R4	•	•
!R8	•	•
!BSTR	•	•
!BOOL	•	•
!DATE	•	•

Conformance Testing

The following hardware/software was used for conformance testing:

- Cable: Ethernet cable

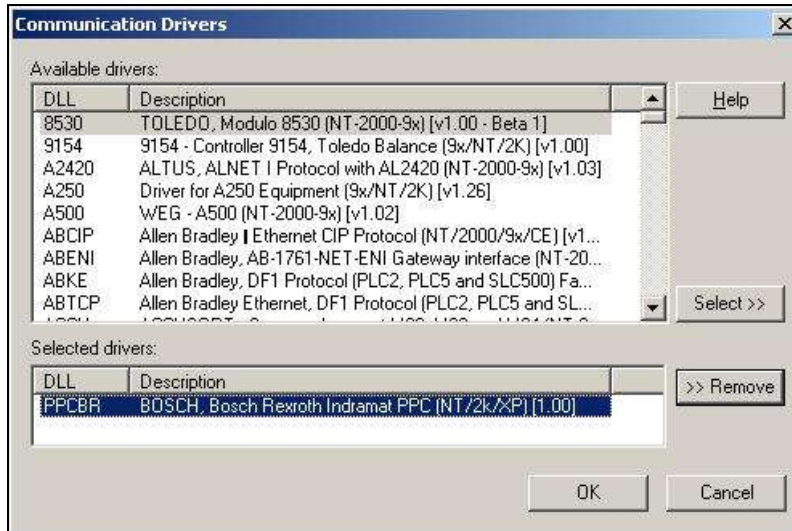
Driver Version	Studio Version	Operating System (development)	Operating System (runtime)	Equipment
1.01	6.1 + SP1	WinXP	▪ WinXP	Bosch Rexroth Indramat PPC-R02. 2N-N-L2-T2-N N-FW

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



Communication Drivers Dialog

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

Note:
It is necessary to install the SCP API 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 PPCBR driver, you must configure the driver. Configuring the PPCBR driver is done in two parts:

- Specifying communication parameters
- Defining tags and controls in the *MAIN* and *Standard Driver* worksheets (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 *MAIN* and *Standard Driver* worksheets, and information about configuring the standard fields, review the product's *Technical Reference Manual*.

Setting the Communication Parameters

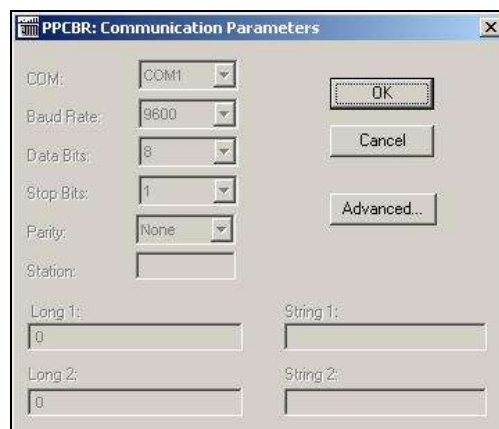
Use the following steps to configure the communication parameters, which are valid for all driver worksheets configured in the system:

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




Select Settings from the Pop-Up Menu

The *PPCBR: Communication Parameters* dialog displays:




PPCBR: Communication Parameters Dialog

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

4. Click the **Advanced** button on the *Communication Parameters* dialog to open the *Advanced Settings* dialog.
5. When the dialog displays, configure the **Control RTS** parameter using the following information:

Parameters	Default Value	Valid Values	Description
Control RTS	No	<ul style="list-style-type: none"> ▪ no ▪ yes ▪ yes+echo 	Configure this parameter if the RTS (Request to Send) handshake signal is set before communication, and if there is an echo in the communication. <ul style="list-style-type: none"> ▪ If you are using Windows 95 or CE with the correct RS232/RS485 Converter (without RTS control), choose the no option. ▪ If you are using Windows NT and the Cutler-Hammer RS232/RS485 adapter, you must use the yes option. <p>IMPORTANT! If you configure this field incorrectly, the driver will not work and will generate a <i>Timeout, waiting to start a message</i> error message.</p>

 **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, the 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 *MAIN* and *Standard Driver Worksheets* (or Communication tables) to associate application tags with the PLC addresses. You can configure multiple *Driver* worksheets — each of which is divided into a *Header* section and a *Body* section.

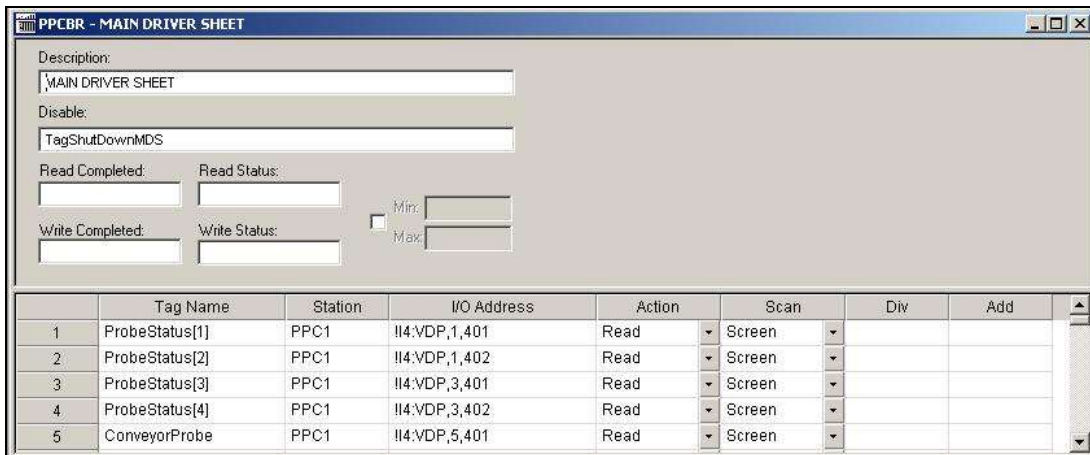
CONFIGURING THE MAIN DRIVER WORKSHEET

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



MAIN Driver Worksheet

You use this worksheet (similar to the following figure) to associate Studio tags to addresses in the PLC:



PPCBR MAIN Driver Worksheet

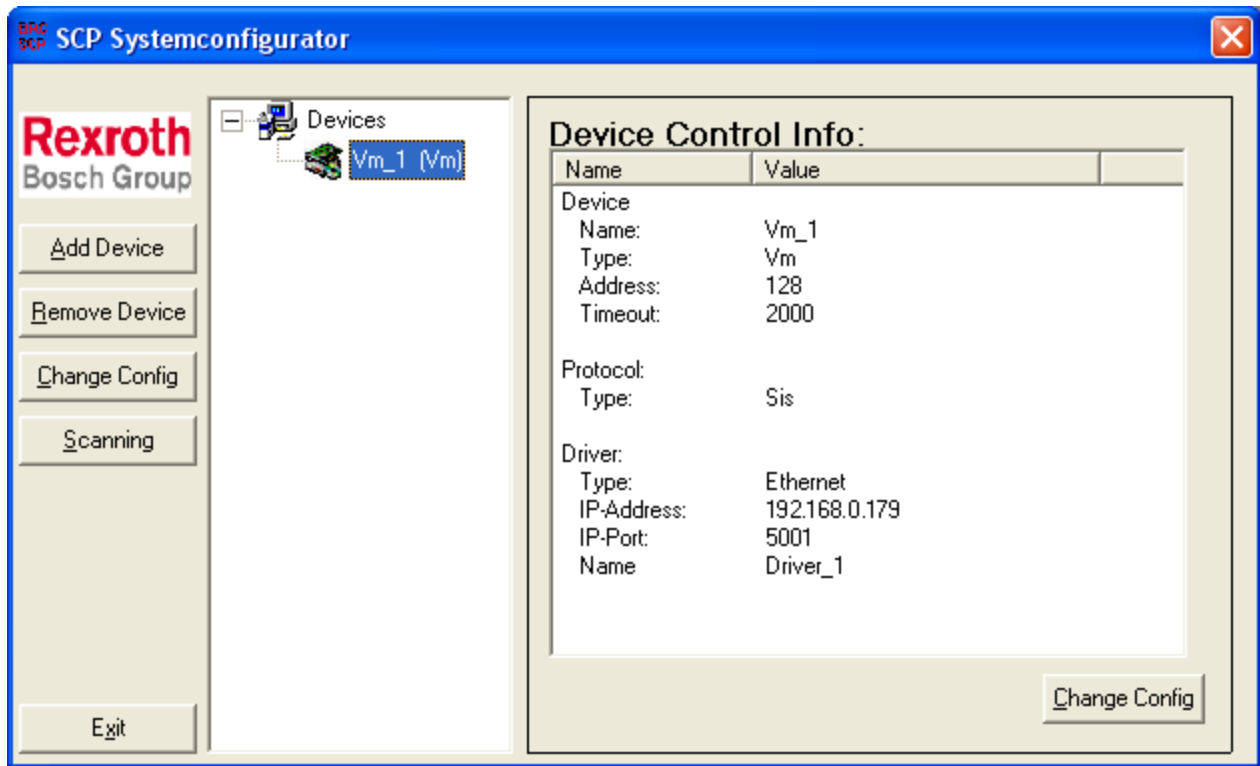
Note:

Most of the *MAIN Driver Worksheet* parameters are standard for all drivers, and are not discussed in this document. Instructions for configuring these standard parameters are provided in the Studio *Technical Reference Manual*.

Use the following information to configure the **Station** and **Address** parameters specific to this driver:

- **Station:** Name of the logical device, configured using the **SCP Configurator** Program

In the example below, you can see the screenshot of the SPC Configurator:



For the **Station** field, you would configure **Vm_1**

- **I/O Address:** Type the address of each item in the device using the following syntax.

<Type>: <Name>, <Row>, <Column> (for example, **!I2:VCP,0,0**)

Where:

- **Type:** Item data type. Valid values are: **!I2, !I4, !R4, !R8, !BSTR, !BOOL** and **!DATE**
- **Name:** The name identifies the element of the logical device.

Where:

System Data valid values are: **VCP** (Card), **VTP** (Task), **VAP** (Aixs), **VDP** (Drive) and **VRD** (I/O Registers)

Program Data valid values are: **VFP** (Fx); **VHP** (GFx); **VIP** (Ix) and **VGP** (Vlx)

- **Row:** Row position of the data element. Usually this is the first information of that address, such as Axis Number, Task number or Drive Number.
- **Column:** Column position of the element. Usually the parameter number for that address.

For example, to read the **Data Display TP 2.100** in Decimal format, you would have in the **Visual Motion I/O Box**, something like this:

Tag Edit Box Original Data Selection: [RD 0.100]

Data Type:

<u>System Data</u>	<u>Program Data</u>	<u>GPP Multi-Master Data</u>	<u>System Feature Data</u>
<input type="radio"/> Card	<input type="radio"/> Fx	<input type="radio"/> Virtual Master	<input type="radio"/> Cam Indexer
<input checked="" type="radio"/> Task	<input type="radio"/> GFx	<input type="radio"/> Group	<input type="radio"/> Registration
<input type="radio"/> Axis	<input type="radio"/> lx	<input type="radio"/> System Masters	<input type="radio"/> PID Loop
<input type="radio"/> Drive	<input type="radio"/> Glx	<input type="radio"/> Slip Monitoring	
<input type="radio"/> I/O Registers			

Data Index:

Task #:

Param #:

Data Label:

Label Name from User Program

Current Value

Label Name from IOI Profile

Name

Set Connection Mode to:

None (Not active) Manual (Request Mode) Auto (Advise mode)

On the Studio I/O Address, you would configure: !I4:VTP,2,100

CONFIGURING THE STANDARD DRIVER WORKSHEET

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

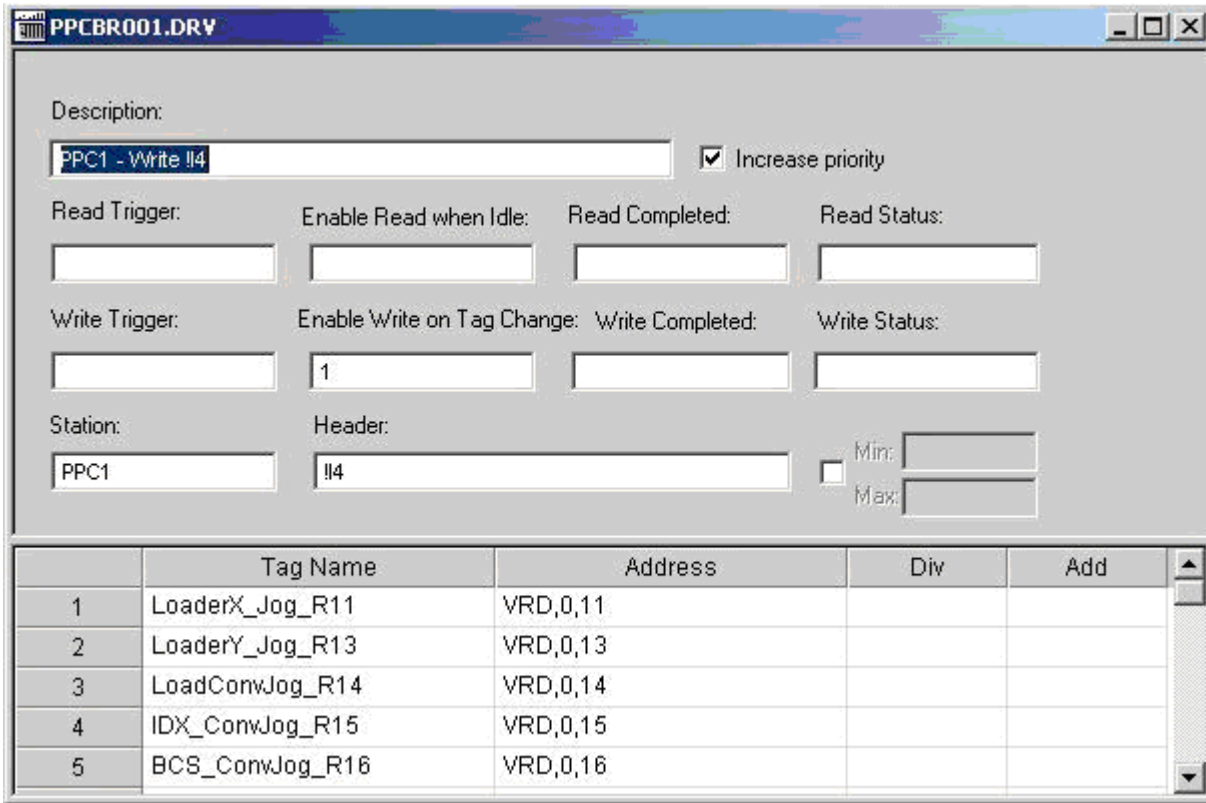


Inserting a New Worksheet

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

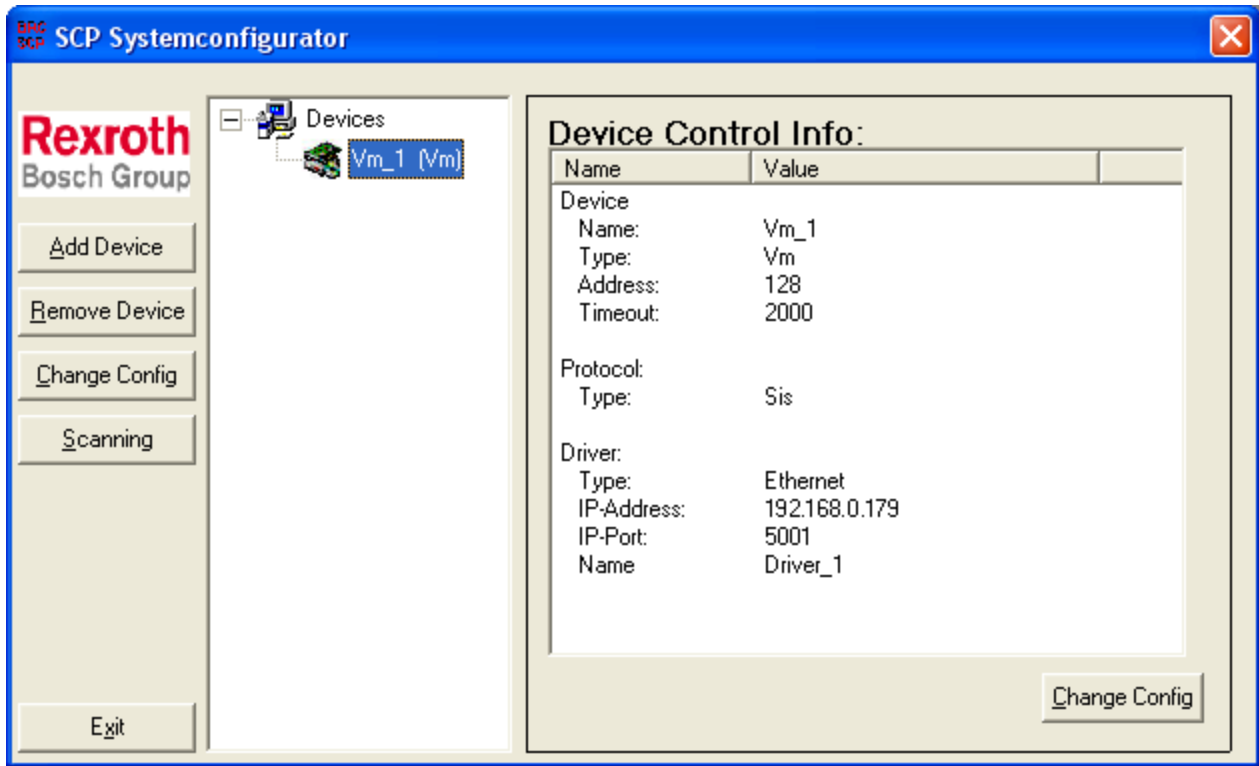


PPCBR Driver Worksheet

In general, all parameters in 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: Name of the logical device

In the example below, you can see the screenshot of the SPC Configurator:



For the **Station** field, you would configure **Vm_1**

- **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.

These variables must comply with the following syntax:

<Type>

Where:

- **Type:** Item data type. Valid values are: **!I2**, **!I4**, **!R4**, **!R8**, **!BSTR**, **!BOOL** and **!DATE**.

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 **!I2** 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 data types and address ranges that are valid for the PPCBR driver:

Data Types	Comments
!I2	Integer with 2 bytes
!I4	Integer with 4 bytes
!R4	Float Point with 4 bytes
!R8	Float Point with 8 bytes
!BSTR	String
!BOOL	Boolean

Data Types	Comments
IDATE	Date

- **Address field:** Use the information in the next 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:

<Name>, <Row>, <Column> (for example **VCP, 104, 1**)

Where:

- **Name:** The name identifies the element of the logical device.

Where:

System Data valid values are: **VCP** (Card), **VTP** (Task), **VAP** (Aixs), **VDP** (Drive) and **VRD** (I/O Registers)

Program Data valid values are: **VFP** (Fx), **VHP** (GFx), **VIP** (Ix) and **VGP** (Vlx)

- **Row:** Row position of the data element. Usually this is the first information of that address, such as Axis Number, Task number or Drive Number.
- **Column:** Column position of the element. Usually the parameter number for that address.

For example, to read the **Data Display TP 2.100** in Decimal format, you would have in the **Visual Motion I/O Box**, something like this:

The screenshot shows a 'Tag Edit Box' window titled 'Original Data Selection: [RD 0.100]'. It contains several sections:

- Data Type:** A grid of radio buttons for selecting a data type. Under 'System Data', 'Task' is selected. Other options include Card, Axis, Drive, I/O Registers, Fx, GFx, Glx, Virtual Master, Group, System Masters, Slip Monitoring, Cam Indexer, Registration, and PID Loop.
- Data Index:** Two spin boxes: 'Task #' is set to 2 and 'Param #' is set to 100. A 'View' button is located to the right.
- Data Label:** Two radio buttons. 'Label Name from User Program' is selected, with 'Target Point Number' in the text box and '0' in the 'Current Value' box. The other option is 'Label Name from IOI Profile' with 'TP 2.100' in the 'Name' box.
- Set Connection Mode to:** Three radio buttons: 'None (Not active)', 'Manual (Request Mode)', and 'Auto (Advise mode)'. 'Auto (Advise mode)' is selected.

At the bottom are buttons for 'OK', 'Cancel', 'Apply', and 'Help'.

On the Studio **I/O Address**, you would configure:

Header: **!!4**

Address: **VTP,2,100**

Device Configuration

Because there are multiple devices that use the Modbus protocol, we cannot define a standard device configuration. Consequently, we recommend using the default RTU protocol configuration.

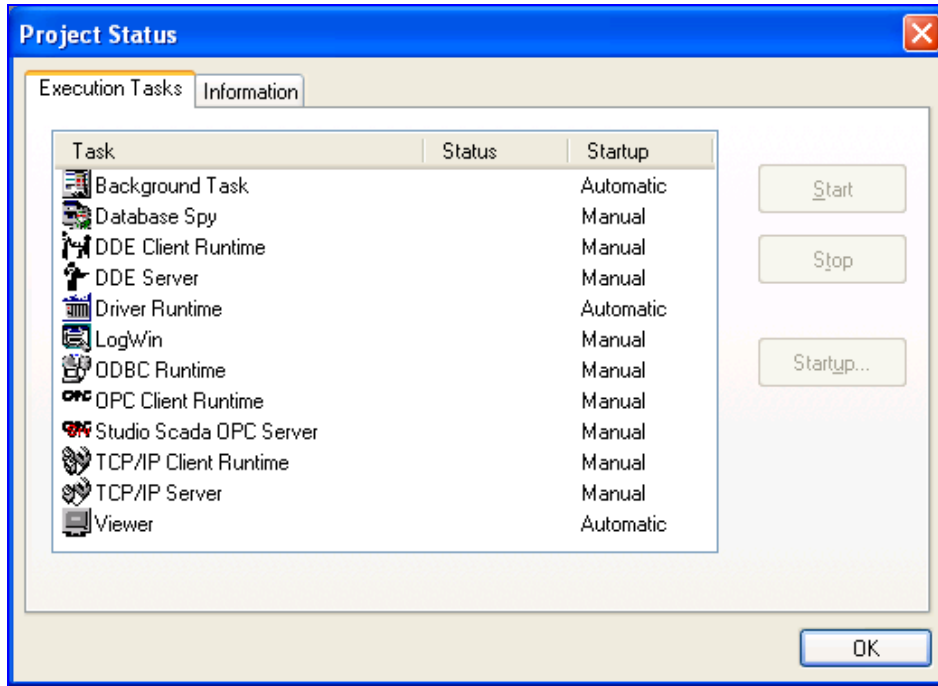
Executing the Driver

After adding the PPCBR 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.
 - 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 PPCBBR 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 CREATEINSTANCE	Error initializing SPC API	Reinstall SPC API.
2	ERROR CONNECT	Error connecting with device	Make sure the name of the logical device is correct.
4	ERROR INVALID ITEM	Error sending message	Make sure the name identifying the element of the logical device, row and column is correct.
5	ERROR INVALID GETGROUP	Internal error	Contact technical support.
6	ERROR INVALID ADDGROUP	Internal error	Contact technical support.
8	ERROR INVALID SYNCREAD	Internal error	Contact technical support.
9	ERROR INVALID VALUE	Error receiving message	Make sure the data type is correct.
-15	Timeout Start Message	<ul style="list-style-type: none"> ▪ Disconnected cables ▪ PLC is turned off, in stop mode, or in error mode ▪ Wrong station number ▪ Wrong RTS/CTS control settings 	<ul style="list-style-type: none"> ▪ Check cable wiring. ▪ Check the PLC state – it must be RUN. ▪ Check the station number. ▪ Check the configuration. See <i>Studio Technical Reference Manual</i> for information about valid RTS/CTS configurations.
-17	Timeout between rx char	<ul style="list-style-type: none"> ▪ PLC is in stop mode or in error mode ▪ Wrong station number ▪ Wrong parity ▪ Wrong RTS/CTS configuration settings 	<ul style="list-style-type: none"> ▪ Check cable wiring. ▪ Check the PLC state – it must be RUN. ▪ Check the configuration. ▪ Check the configuration. See <i>Studio Technical Reference Manual</i> 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 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

You will find a sample application in the `/COMMUNICATION EXAMPLES/PPCBR` directory. We strongly recommend that you use this sample application to test the PPCBR driver before configuring your customized application, for the following reasons:

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

 **Note:**

This application sample is not available for all drivers.

Use the following procedure to perform the test:

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

 **Tip:**

You can use the sample application screen as the maintenance screen for your custom applications.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of changes
A	1.00	Eric Vigiani	Jan/07/2005	Initial version
B	1.01	Eric Vigiani	Oct/05/2006	General Revision