VPCE Communication Driver

Driver for MPI or Industrial Ethernet
communications on VIPA HMIs

Contents

Introduction

The VPCE driver enables communication between the Studio system on WinCE (VIPA HMI) and the VIPA devices using MPI or Industrial Ethernet, according to the specifications discussed in this publication.

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

- **Introduction**: Provides an overview of the VPCE driver documentation
- **General Information**: Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the VPCE driver
- **Installing the Driver**: Explains how to install the VPCE 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 environment.
  If you are unfamiliar with Windows 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 VPCE driver and VIPA devices using MPI or Industrial Ethernet.

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**: VIPA
- **Compatible Equipment**: Any PLC that is compatible with the MPI or Industrial Ethernet protocol
- **VIPA PLC Programmer Software**: WinPLC 7

Link Characteristics

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

- **Device Communication Port**: MPI Port / Ethernet Port
- **Physical Protocol**: RS232 / Ethernet TCP/IP
- **Logic Protocol**: MPI / Industrial Ethernet
- **API / Library**: S7COMM library
- **Adapters/Converters**: None

Driver Characteristics

The VPCE driver is composed of the following files:

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

> **Notes:**
> - All of the preceding files are installed in the **`/DRV`** subdirectory of the Studio installation directory.
> - You must use the Adobe Acrobat® Reader™ (provided on the Studio installation CD-ROM) to view the **`VPCE.PDF`** document.
> - The VPCE driver requests the **`S7COMM.DLL`** and **`MPI.dll`** (VIPA files).

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

- Windows CE

The VPCE driver supports the following registers:

Register Type	Supported Data type	Write	Read	Decimal	Hex	Counter	S5Time	Real	Pointer
M (*Flags*)	Byte	•	•	•	•	–	–	–	–
	Word	•	•	•	•	•	•	–	–
	DWORD	•	•	•	•	–	–	•	•
T (*Timers*)	–	–	–	•	–	•	–	•	–	–
Z or C (*Counters*)	–	–	–	•	–	•	•	–	–	–
E or I (*Inputs*)	Byte	–	•	•	•	–	–	–	–
	Word	–	•	•	•	•	•	–	–
	DWORD	–	•	•	•	–	–	•	•
A or Q (*Outputs*)	Byte	•	•	•	•	–	–	–	–
	Word	•	•	•	•	•	•	–	–
	DWORD	•	•	•	•	–	–	•	•
DB (*Data Blocks*)	Byte	•	•	•	•	–	–	–	–
	Word	•	•	•	•	•	•	–	–
	DWORD	•	•	•	•	–	–	•	•

Conformance Testing

The following hardware/software was used for conformance testing:

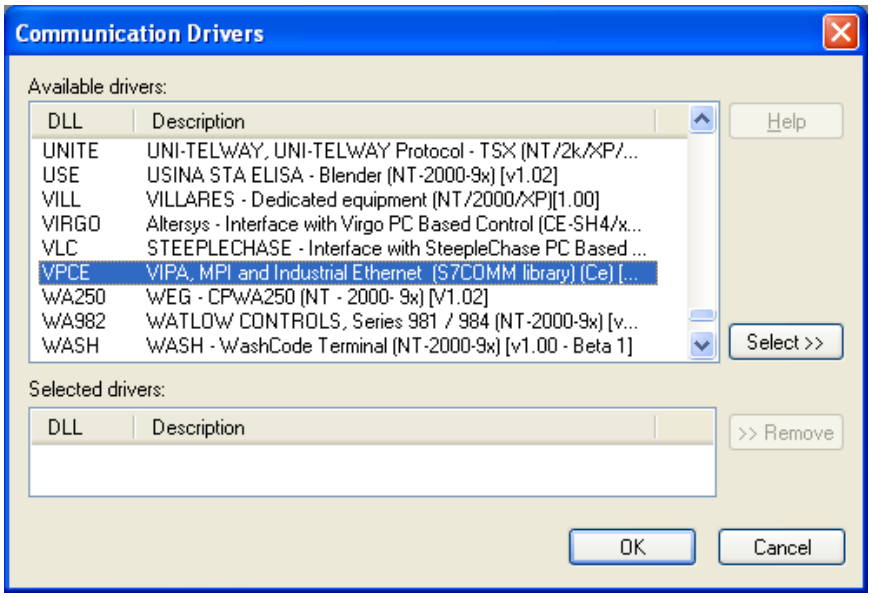
- **Equipment**:
  HMI: VIPA 608-1BC00 (ArmV4i)
  PLC: VIPA CPU 314ST (314-6CF02)
- **Driver Configuration**:
  – **PLC Program**: WinPLC7
  – **Protocol**: MPI and Industrial Ethernet
- **MPI Cable:** VIPA 950-0KB50
- **Operating System**: Windows CE 5.0
- **Studio Version**: 7.0
- **Driver Version**: 1.4

Driver Version	Studio Version	Operating System	Equipment
1.4	7.0	Windows CE 5.0	▪ **HMI**: VIPA 608-1BC00 (ArmV4i) ▪ **PLC**: VIPA CPU 314ST (314-6CF02

Selecting the Driver

When you install Studio, all of the communication drivers are automatically installed in the **\DRV** subdirectory but they remain dormant until manually selected for specific applications. To select the VPCE driver for your Studio application:

1. From the main menu bar, select **Insert** → **Driver** to open the *Communication Drivers* dialog.
2. Select the **VPCE** driver from the *Available Drivers* list, and then click the **Select** button.



Communication Drivers Dialog

3. When the **VPCE** driver is displayed in the **Selected Drivers** list, click the **OK** button to close the dialog. The driver is added to the *Drivers* folder, in the *Comm* tab of the Workspace.

> **Note:**
> It is necessary to install the S7COMM library on your WinCE device to enable communication between Studio and your target device. However, this communication can only be used by the Studio application; it cannot be used to download control logic to the device.

> **Attention:**
> For safety reasons, you must take special precautions when installing any physical hardware. Please consult the manufacturer's documentation for specific instructions.

Configuring the Device

Once the selected driver and the target device are both properly configured, it is necessary to install the S7COMM library on your WinCE device to enable communication between the host and the device. All runtime communication is handled within your Studio application project. However, programming the device itself — that is, developing control logic and downloading it to the device — still requires using the device's own programming tool.

Configuring the Driver

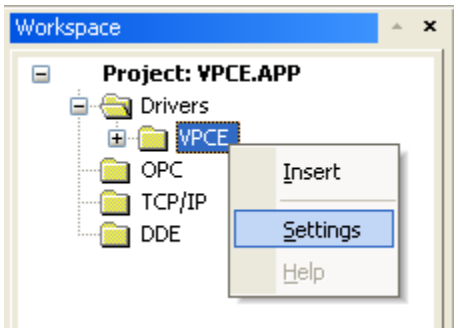
Once you have selected the VPCE driver in Studio, you must properly configure it to communicate with your target device. First, you must set the driver's communication settings to match the parameters set on the device. Then, you must build driver worksheets to associate database tags in your Studio application with the appropriate addresses (registers) on the device.

Configuring the Communication Settings

The communication settings are described in detail in the "Communication" chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

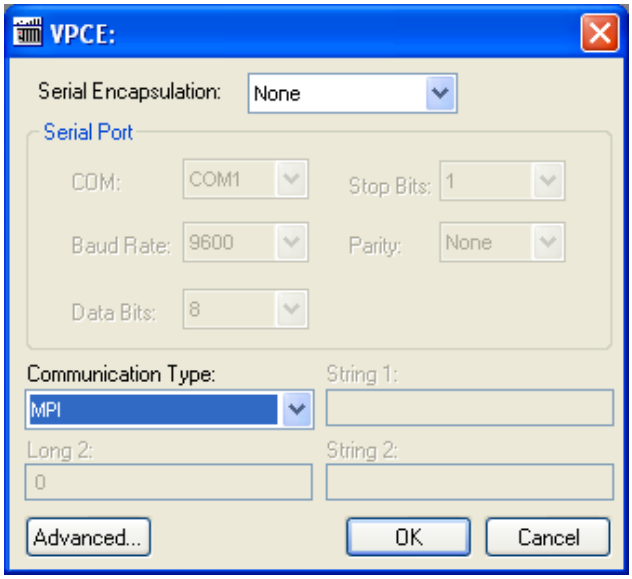
For the purposes of this document, only VPCE driver-specific settings and procedures will be discussed here. To configure the communication settings for the VPCE driver:

1. In the *Workspace* pane, select the *Comm* tab and then expand the *Drivers* folder. The VPCE driver is listed here as a subfolder.
2. Right-click on the *VPCE* subfolder and then select the **Settings** option from the pop-up menu:



Select Settings from the Pop-Up Menu

The *VPCE: Communication Settings* dialog is displayed:



VPCE: Communication Settings Dialog

3. In the *Communication Settings* dialog, configure the driver settings to enable communication with your target device. To ensure error-free communication, the driver settings must *exactly match* the corresponding settings on the device. Please consult the manufacturer's documentation for instructions how to configure the device and for complete descriptions of the settings.

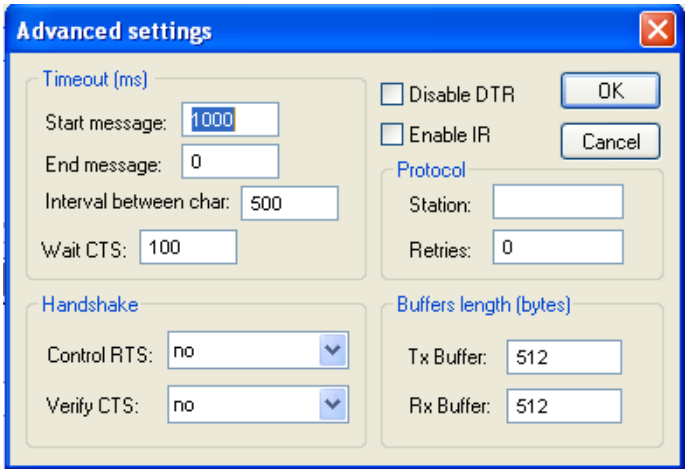
   Depending on your circumstances, you may need to configure the driver *before* you have configured your target device. If this is the case, then take note of the driver settings and have them ready when you later configure the device.

> ➲ **Attention:**
> For safety reasons, you **must** take special precautions when connecting and configuring new equipment. Please consult the manufacturer's documentation for specific instructions.

The communication settings and their possible values are described in the following table:

Parameters	Default Values	Valid Values	Description
Communication Type	Industrial Ethernet	- MPI - Industrial Ethernet	The protocols used for communication: - MPI (RS232) - Industrial Ethernet (Ethernet TCP/iP)

4. If you are using a Data Communication Equipment (DCE) converter (e.g., 232/485) between your PC and your target device, then you must also adjust the **Control RTS** (Request to Send) setting to account for the converter. In the *Communication Settings* dialog, click the **Advanced** button to open the *Advanced Settings* dialog:



Advanced Settings Dialog

When the dialog is displayed, configure the **Control RTS** setting using the following information:

Setting	Default	Values	Description
Control RTS	no	no	Do not set the RTS (Request to Send) handshake signal. IMPORTANT: If you are using Windows 95/98 or Windows CE with the correct RS232/RS485 adapter (i.e. without RTS control), then you must select this option.
		yes	Set the RTS (Request to Send) handshake signal before communication. IMPORTANT: If you are using Windows NT and the Cutler-Hammer RS232/RS485 adapter, then you must select this option.
		yes+echo	Set the RTS (Request to Send) handshake signal before communication, and echo the signal received from the target device.
		Always on	The RTS (Request to Send) handshake signal is always set

➲ **Attention**:
If you incorrectly configure the **Control RTS** setting, then runtime communication will fail and the driver will generate a –15 error. See "Troubleshooting" for more information.

You do not need to change any other advanced settings at this time. You can consult the Studio *Technical Reference Manual* later for more information about configuring these settings.

5. Click **OK** to close the *Advanced Settings* dialog, and then click **OK** to close the *Communication Settings* dialog.

Configuring the Driver Worksheets

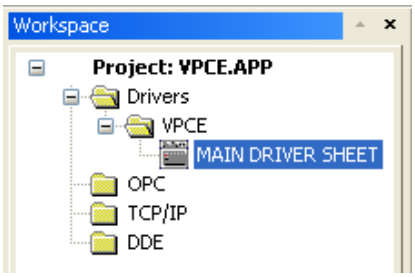
A selected driver includes one or more driver worksheets, which are used to associate database tags in Studio with registers on the target device. Each worksheet is triggered by specific application behavior, so that the tags / registers defined on that worksheet are scanned only when necessary – that is, only when the application is doing something that requires reading from or writing to those specific tags / registers. Doing this optimizes communication and improves system performance.

The configuration of these worksheets is described in detail in the "Communication" chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

Main Driver Sheet

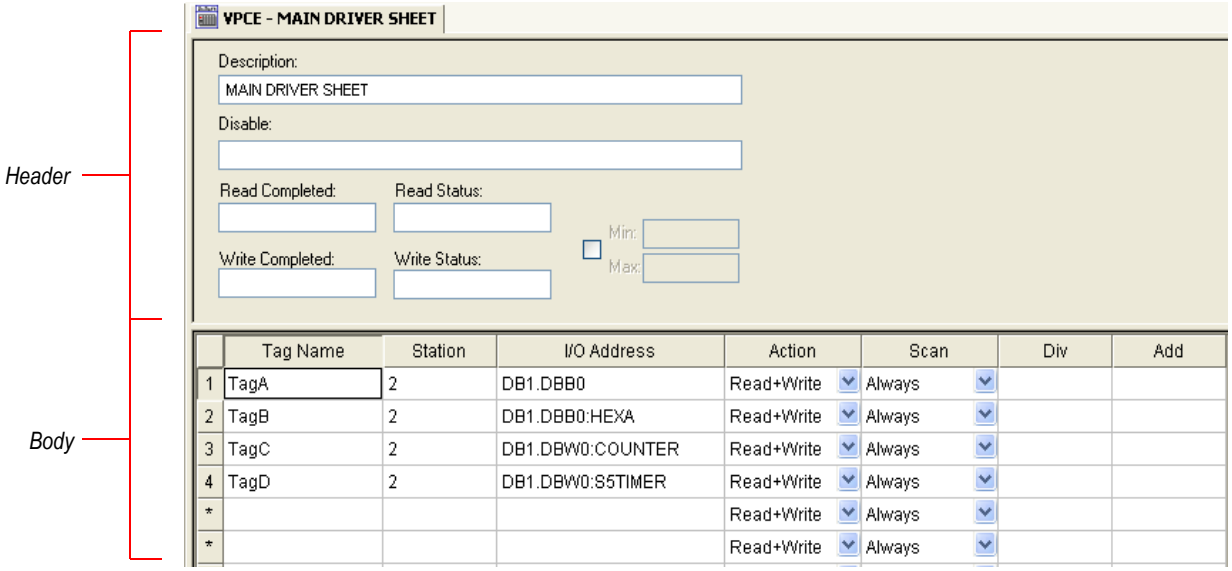
When you select the VPCE driver and add it to your application, Studio automatically inserts the *Main Driver Sheet* in the *VPCE* driver subfolder. To configure the Main Driver Sheet:

1. Select the *Comm* tab in the *Workspace* pane.
2. Open the *Drivers* folder, and then open the *VPCE* subfolder:



Main Driver Sheet in the VPCE Subfolder

3. Double-click on the **MAIN DRIVER SHEET** icon to open the following worksheet:



Opening the Main Driver Sheet

Most of the fields on this sheet are standard for all drivers; see the "Communication" chapter of the *Technical Reference Manual* for more information on configuring these fields. However, the **Station** and **I/O Address** fields use syntax that is specific to the VPCE driver.

4. For each table row (i.e., each tag/register association), configure the **Station** and **I/O Address** fields as follows:

   - **Station** field: Specify station of the device, using the following syntax:

   For MPI Communication Type:

   <PLC ID>

   Example — **2**

   Where:

   – ***<PLC ID>*** is the device's ID on the MPI network; Valid values are 1– 126.

   For Industrial Ethernet Communication Type:

   <IP Address>

   Example — **192.168.125.31**

   Where:

   – ***<IP Address>*** is the device's IP address on the TCP/IP network;

You can also specify an indirect tag (e.g. **{station}**), but the tag that is referenced must follow the same syntax and contain a valid value.

> ➲ **Attention:**
> • You must use a non-zero value in the **Station** field, and you cannot leave the field blank.

- **I/O Address:** Specify the address of the associated device register.

For all non-Strings:

❑ For Flags, Timers, Counters, Inputs and Outputs:

<Type>[Datatype]<Address>:[Format]

For example: **MW10:HEX**

❑ For Data-Blocks:

<Type><TypeGroup>*.DB*[Datatype]<Address>:[Format]

For example: **DB1.DBB5:DECIMAL**

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).

- **DataType** defines how Studio treats the value read from or written to the device.
    - B (Byte)
    - W (Word)
    - D (DWord)
- **Address** is the specific address of the register on the device.
- **Fomat** defines how Studio shows the value read from or written to the device. If this parameter is omitted so DECIMAL is assumed.
    - DECIMAL (for example: 255);
    - HEX (for example: 0x00ab);
    - COUNTER (for example: C800);
    - S5TIME (for example: 800ms);
    - REAL (for example: 1.123000e+000);
    - POINTER (for example: P31.7 (octal)).

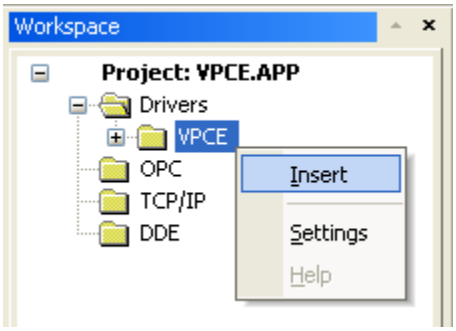
For more information about registers and addressing, please consult the manufacturer's documentation.

Standard Driver Worksheet

When you select the VPCE driver, you may insert *Standard Driver Worksheet*s to define tag/register associations that are triggered by specific application behaviors. Doing this will optimize communication and improve system performance by ensuring that tags/registers are scanned only when necessary – that is, only when the application is performing an action that requires reading or writing to those specific tags/registers.

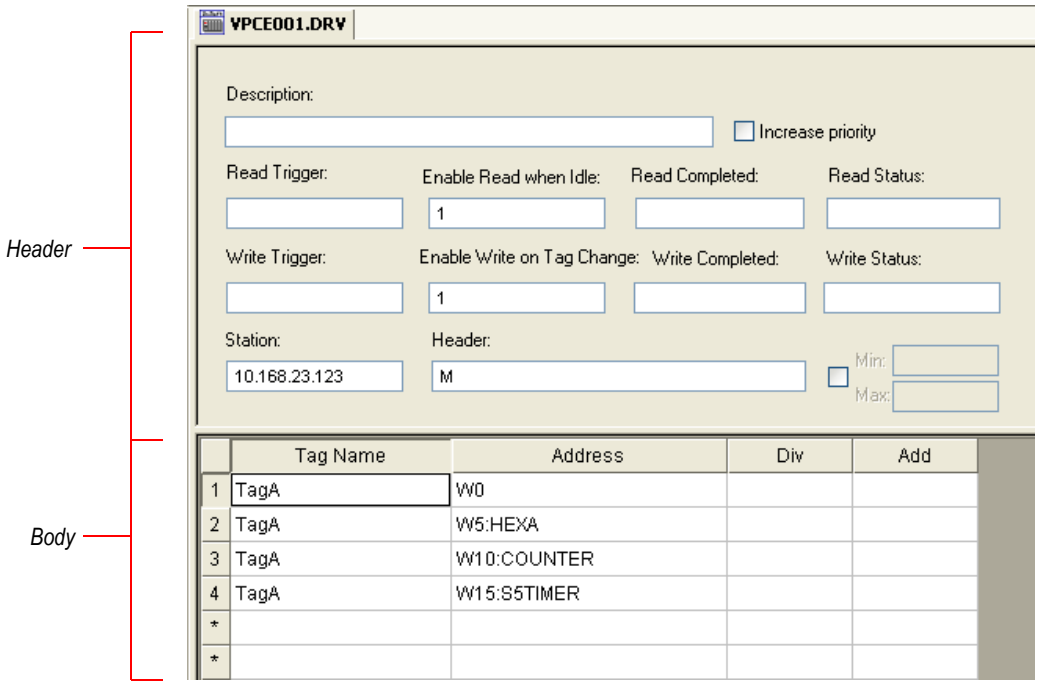
To insert a new *Standard Driver Worksheet*:

2. In the *Comm* tab, open the *Drivers* folder and locate the *VPCE* subfolder.
3. Right-click on the *VPCE* subfolder, and then select **Insert** from the pop-up menu:



Inserting a New Worksheet

A new VPCE driver worksheet is inserted into the *VPCE* subfolder, and the worksheet is opened for configuration:



VPCE Driver Worksheet

Most of the fields on this worksheet are standard for all drivers; see the "Communication" chapter of the *Technical Reference Manual* for more information on configuring these fields. However, the **Station**, **Header**, and **Address** fields use syntax that is specific to the VPCE driver.

4. Configure the **Station** and **Header** fields as follows:
   - **Station** field: Specify the station of the device, using the following syntax:

     `For MPI Communication Type:`

     `<PLC ID>`

     Example — **`2`**

     Where:

     – ***`<PLC ID>`*** is the device's ID on the MPI network; Valid values are 1– 126.

     `For Industrial Ethernet Communication Type:`

     `<IP Address>`

     Example — **`192.168.125.31`**

     Where:

     – ***`<IP Address>`*** is the device's IP address on the TCP/IP network;

You can also specify an indirect tag (e.g. **`{station}`**), but the tag that is referenced must follow the same syntax and contain a valid value.

> ➲ **Attention:**
> • You must use a non-zero value in the **Station** field, and you cannot leave the field blank.

- **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.

The **Header** field uses the following syntax:

- For Flags, Timers, Counters, Inputs and Outputs:
  `<Type>` (for example: **`M`**)

- For Data-Blocks:
  `<Type><TypeGroup>`**`.DB`** (for example: **`DB1.DB`**)

  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).

The next table lists all of the data types and address ranges that are valid for the VPCE driver:

Header Field Information			
Data Types	**Sample Syntax**	**Valid Range of Initial Addresses**	**Comments**
Flags	**M**	Varies according to the equipment	▪ Logical Flags
Timers	**T**	Varies according to the equipment	▪ Timer Values
Counters	**Z or C**	Varies according to the equipment	▪ Counter Values
Inputs	**E or I**	Varies according to the equipment	▪ Physical Input Values
Outputs	**A or Q**	Varies according to the equipment	▪ Physical Output Values
Data Blocks	**DB1.DB** **DB2.DB** **DB10.DB**	Varies according to the equipment	▪ Data block values

- **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:

[Datatype]<Address>:[Format] (for example: **W10:HEX**)

Where:

- **DataType** defines how Studio treats the value read from or written to the device.
  - B (Byte)
  - W (Word)
  - D (DWord)
- **Address** is the specific address of the register on the device.
- **Fomat** defines how Studio shows the value read from or written to the device. If this parameter is omitted so DECIMAL is assumed.
  - DECIMAL (for example: 255);
  - HEX (for example: 0x00ab);
  - COUNTER (for example: C800);
  - S5TIME (for example: 800ms);
  - REAL (for example: 1.123000e+000);
  - POINTER (for example: P31.7 (octal)).

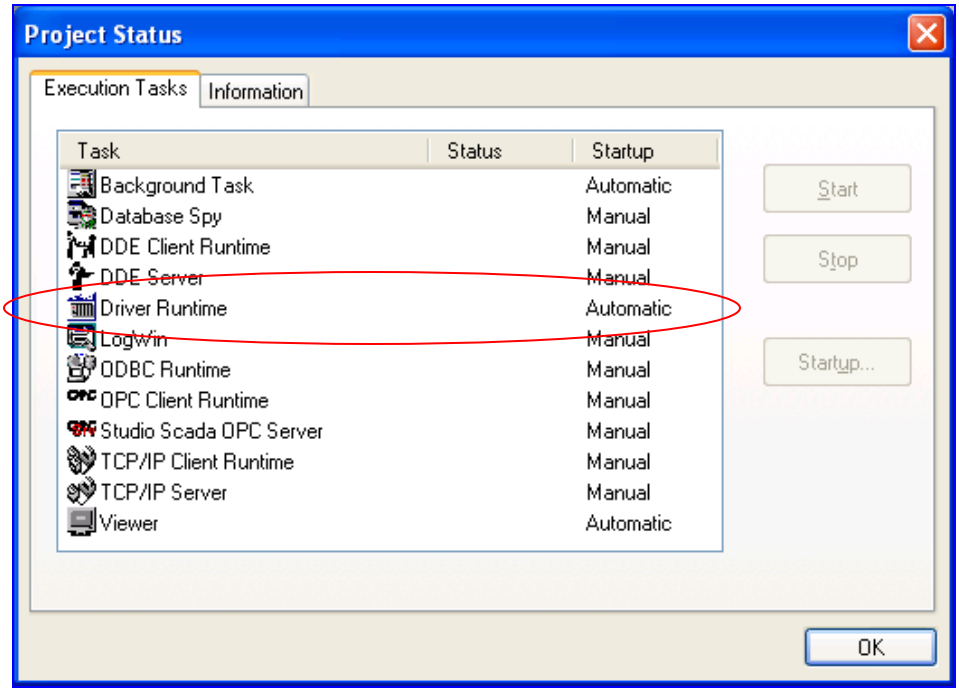
➲ **Attention:**
You must not configure a range of addresses greater than the maximum block size (data buffer length) supported by the protocol within the same worksheet. The maximum data buffer length for this driver is 128 addresses per *Standard Driver* worksheet.

Executing the Driver

By default, Studio will automatically execute your selected communication driver(s) during application runtime. However, you may verify your application's runtime execution settings by checking the *Project Status* dialog.

To verify that the communication driver(s) will execute correctly:

1. From the main menu bar, select **Project** → **Status**. The *Project Status* dialog displays:



Project Status Dialog

2. Verify that the *Driver Runtime* task is set to **Automatic**.
   - If the setting is correct, then proceed to step 3 below.
   - If the **Driver Runtime** task is set to **Manual**, then select the task and click the **Startup** button to toggle the task's *Startup* mode to **Automatic**.
3. Click **OK** to close the *Project Status* dialog.
4. Start the application to run the driver.

Troubleshooting

If the VPCE driver fails to communicate with the target device, then the database tag(s) that you configured for the **Read Status** or **Write Status** fields of the Driver Sheet 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
-34	Invalid Address	The address specified is invalid	Check if the worksheet configuration is correct. Check if the worksheet is configured according to PLC.
-39	Invalid Block Size	The maximum number of bytes per worksheet has been exceeded.	Split your driver worksheets into different driver worksheets or use the MAIN DRIVER SHEET.
1	MPI.dll was not found	The MPI.dll is not installed correctly in the HMI.	Check if with the manufacture if the S7COMM library is installed correctly in the HMI (WinCE).
2	Error in MPI functions	The MPI.dll is not installed correctly in the HMI.	Check if with the manufacture if the S7COMM library is installed correctly in the HMI (WinCE).
3	S7Comm.dll was not found	The S7COMM.dll is not installed correctly in the HMI.	Check if with the manufacture if the S7COMM library is installed correctly in the HMI (WinCE).
4	Error in S7Comm functions	The S7COMM.dll is not installed correctly in the HMI.	Check if with the manufacture if the S7COMM library is installed correctly in the HMI (WinCE).
5	Error in MPI Open function.	Error in the MPI Open function	Check cable wiring. Check the PLC state – it must be RUN.
6	Error in MPI GoOnLine function.	Error in the GoOnLine function.	Check cable wiring. Check the PLC state – it must be RUN.
7	Invalid Connection	The driver is enable to establish the connection with the PLC.	Check cable wiring. Check the PLC state – it must be RUN. Check the configuration in the Station field.
8	Fail to Read	The address configured in the driver worksheet does not exist in the PLC.	Check if the worksheet configuration is correct. Check if the worksheet is configured according to PLC.
9	Fail to Write	The address configured in the driver worksheet does not exist in the PLC.	Check if the worksheet configuration is correct. Check if the worksheet is configured according to PLC.

⇨ **Tip:**

You can monitor communication status by establishing an event log in Studio's *Output* window (*LogWin* module). To establish a log for **Field Read Commands**, **Field Write Commands** and **Serial Communication,** right-click in the *Output* window and select the desired options from the pop-up menu.

You can also use the *LogWin* module (**Tools** → **LogWin**) to establish an event log on a remote unit that runs Windows CE. The log is saved on the unit in the **`celog.txt`** file, which can be downloaded later.

If you are unable to establish communication between Studio and the target device, then try instead to establish communication using the device's own programming software. Quite often, communication is interrupted by a hardware or cable problem or by a device configuration error. If you can successfully communicate using the programming software, then recheck the driver's communication settings in Studio.

To test communication between Studio and the device, 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

There was not an official sample application available for this driver by the timer that this document was written.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of changes
A	1.01	Eric Vigiani	May/12/2008	Initial release
B	1.02	Paulo Balbino	Sep/22/2009	- Reviewed error codes section - Improved driver validation to make application debugging easier.
C	1.03	Eric Vigiani	Apr/20/2010	Fixed issue to create virtual groups when using DBs.
D	1.04	Paulo Balbino	Jul/27/2011	- Fixed issue with writing operations for Bytes and Words - Fixed issue with Time Stamp