EATON Communication Driver

Driver for Ethernet or Serial Communication
with EATON ELC Devices using Modbus protocol

Contents

CONTENTS **1**

INTRODUCTION **2**

GENERAL INFORMATION **3**

- DEVICE SPECIFICATIONS 3
- NETWORK SPECIFICATIONS 3
- DRIVER CHARACTERISTICS 3
- CONFORMANCE TESTING 4

SELECTING THE DRIVER **5**

CONFIGURING THE DRIVER **6**

- CONFIGURING THE COMMUNICATION SETTINGS 6
- CONFIGURING THE DRIVER WORKSHEETS 7

EXECUTING THE DRIVER **17**

TROUBLESHOOTING **18**

REVISION HISTORY **21**

Introduction

This document will help you to select, configure and execute the EATON driver, and it is organized as follows:

- **Introduction**: This section, which provides an overview of the document.
- **General Information**: Identifies all of the hardware and software components required to implement communication between the Studio system and the target device.
- **Selecting the Driver**: Explains how to select the EATON driver in the Studio system.
- **Configuring the Driver**: Explains how to configure the EATON driver in the Studio system, including how to associate database tags with device registers.
- **Executing the Driver**: Explains how to execute the EATON driver during application runtime.
- **Troubleshooting**: Lists the most common errors for this driver, their probable causes, and basic procedures to resolve them.
- **Revision History**: Provides a log of all changes made to the driver and this documentation.

Notes:

- This document assumes that you have read the "Development Environment" chapter in Studio's *Technical Reference Manual*.
- This document also assumes that you are familiar with the Microsoft Windows 7/XP environment. If you are not familiar with Windows, then we suggest using the **Help** feature (available from the Windows desktop **Start** menu) as you work through this guide.

General Information

This chapter identifies all of the hardware and software components required to implement communication between the EATON driver in Studio and remote devices.

The information is organized into the following sections:

- Device Specifications
- Network Specifications
- Driver Characteristics
- Conformance Testing

Device Specifications

You can use this driver to communicate with any device using a simple ASCII protocol. (The devices used for conformance testing are listed on the next page.)

Network Specifications

To establish communication, your device network must meet the following specifications:

- **Device Communication Port**: Ethernet or SERIAL
- **Physical Protocol**: TCP/IP, RS-232
- **Logic Protocol**: Modbus ASCII/RTU
- **Device Runtime Software**: None
- **Specific PC Board**: None
- **Adapters/Converters**: None
- **Cable Wiring Scheme**: None

Driver Characteristics

The EATON driver package consists of the following files, which are automatically installed in the **\DRV** subdirectory of Studio:

- **EATON.INI:** Internal driver file. *You must not modify this file*.
- **EATON.MSG:** Internal driver file containing error messages for each error code. *You must not modify this file*.
- **EATON.PDF:** This document, which provides detailed information about the EATON driver.
- **EATON.DLL:** Compiled driver.

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

- Windows 7/XP/ Embedded
- Windows CE 5.x, 6.x, 7.x

For a description of the operating systems used to test driver conformance, see “Conformance Testing” below.

Conformance Testing

The following hardware/software was used for conformance testing:

For Serial Communication Tests

- **Driver Configuration**:
  - **PLC Program**: Eaton ELC V1.6 – RS-232
  - **Baud Rate**: 9600
  - **Protocol**: ASCII
  - **Data Bits**: 7
  - **Stop Bits**: 1
  - **Parity**: Even
  - **Station**: 1

For Ethernet Tests

- **TCP/IP Port**: 502
- **Protocol**: RTU
- **Station**: 1
- **Cable**: Ethernet Cable

Driver Version	Studio Version	Operating System	Equipment
1.2	7.1	Windows XP SP3 Windows 7 x64 Windows CE 5.0 ARMV4i	– 2x EATON ELC-PV28NNDR with ELC-COENETM module

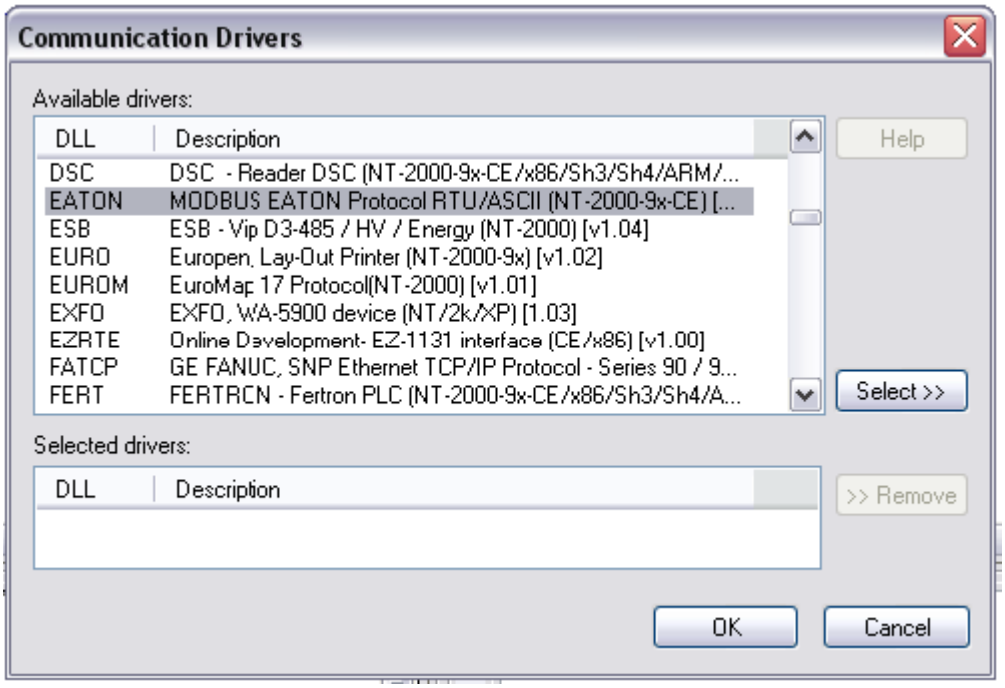
The EATON driver supports the following device types:

Device Name	Length	Write	Read	Bit	Word	Float	Double
X (Discrete Input)	1 Bit	–	•	•	–	–	–
Y (Discrete Output)	1 Bit	•	•	•	–	–	–
M (Main relay)	1 Bit	•	•	•	–	–	–
S (Step point)	1 Bit	–	•	•	–	–	–
TS (Timer Status)	1 Bit	–	•	•	–	–	–
TPV (Timer Present Value)	1 Word	•	•	–	•	–	–
CS (16-Bits Counter Status)	1 Bit	–	•	•	–	–	–
CPV (16-Bits Counter Present Value)	1 Word	•	•	–	•	–	–
CSX (32-Bits Counter Status)	1 Bit	–	•	•	–	–	–
CPVX (32-BitsConter Present Value)	1 DWord	•	•		–	–	•
D (Data Register)	1 Word	•	•	•	•	•	•

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 EATON driver for your Studio application:

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



Communication Drivers Dialog

3. When the **EATON** 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.

➲ **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 Driver

Once you have selected the EATON driver in Studio, you must properly configure it to communicate with your target device.

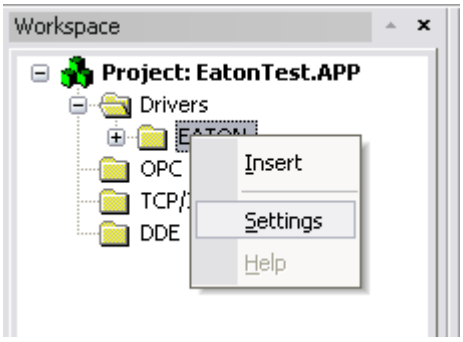
Configuring the Communication Settings

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.

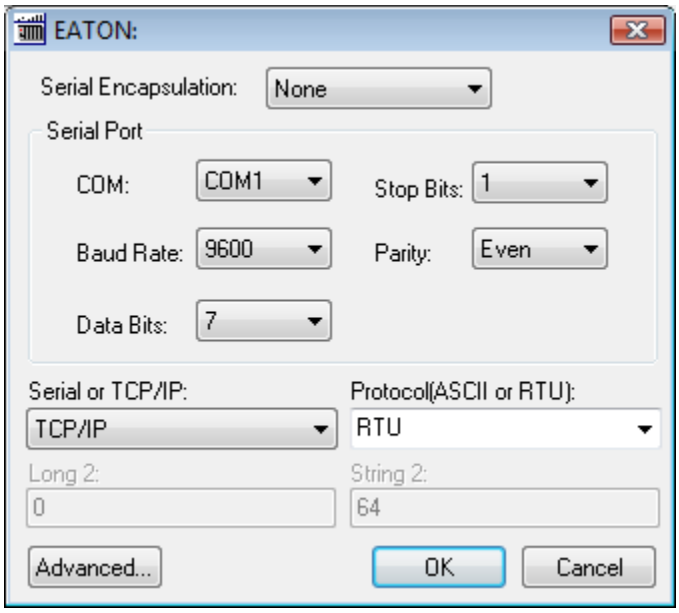
For the purposes of this document, only EATON driver-specific settings and procedures will be discussed here. To configure the communication settings for the EATON driver:

1. In the *Workspace* pane, select the *Comm* tab and then expand the *Drivers* folder. The EATON driver is listed here as a subfolder.

2. Right-click on the EATON subfolder and then select the **Settings** option from the pop-up menu. The EATON*: Communication Parameters* dialog is displayed:



Select Settings from the Pop-Up Menu



EATON: Communication Parameters Dialog

3. Configure the additional driver-specific settings, as described in the following table:

Setting	Default Value	Valid Values	Description
Serial or TCP/IP	**TcpIp**	Serial or TcpIp	Select connection type
Protocol	**RTU**	**RTU or ASCII**	Protocol type

4. Click **OK** to close the *Communication Settings* dialog.

Configuring the Driver Worksheets

Each selected driver includes a Main Driver Sheet and one or more Standard Driver Worksheets. The Main Driver Sheet is used to define tag/register associations and driver parameters that are in effect at all times, regardless of application behavior. In contrast, Standard Driver Worksheets can be inserted to define additional tag/register associations that are triggered by specific application behaviors.

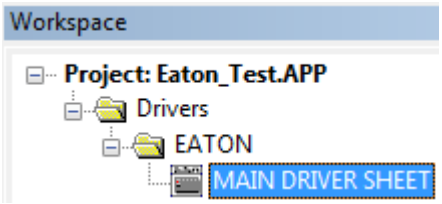
The configuration of these worksheets is described in detail in the "Communication" chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

For the purposes of this document, only EATON driver-specific parameters and procedures will be discussed here.

MAIN DRIVER SHEET

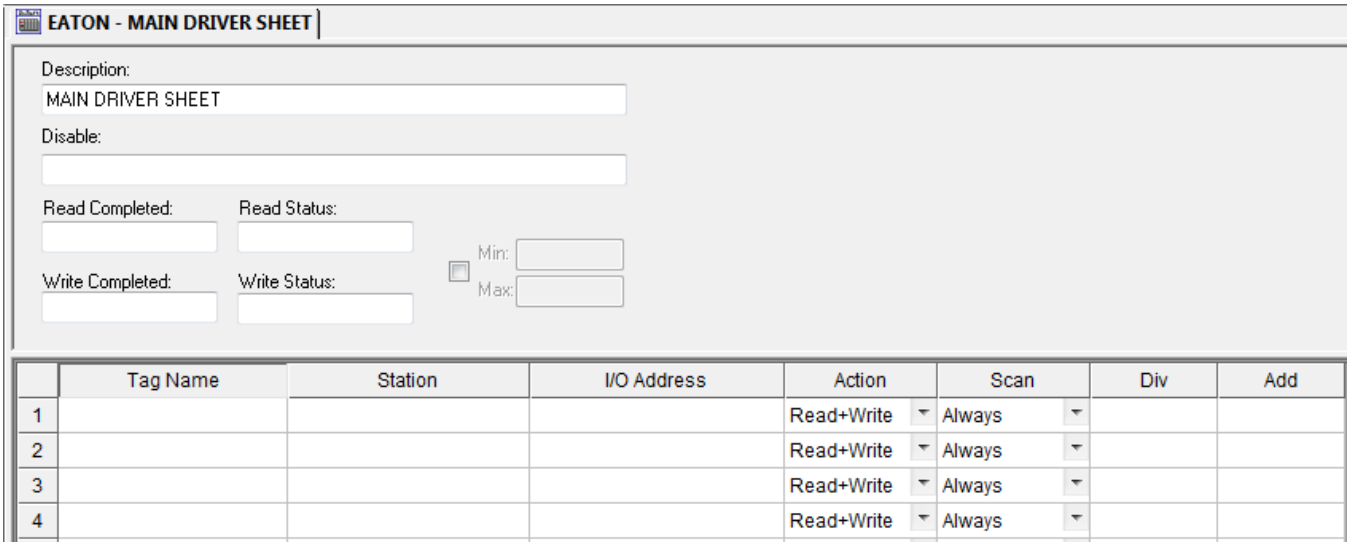
When you select the EATON driver and add it to your application, Studio automatically inserts the *Main Driver Sheet* in the *EATON* driver subfolder. To configure the Main Driver Sheet:

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



Main Driver Sheet in the EATON Subfolder

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



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 EATON driver.

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

   - **Station** field

     - For *Serial Communication*, specify here the **ELC ID** Number.
     - For *Ethernet*, use the following syntax:

       <IP Address>:<opt ID Number>:<Opt Port Number>

   Example — **`192.168.0.52`** or **`192.168.0.52:1:502`**

   Where:

   – ***<IP Address>*** is the IP address of the device on the Ethernet network.

   – ***<opt ID Number>*** optional PLC Slave ID in the Modbus network. If you do not configure this parameter, the driver will use the broadcast value **255**

   – ***<opt Port Number>*** optional TCP/IP Port number. If you do not configure this parameter, the default value of **502** will be used

   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 cannot leave the Station field **blank**

- **I/O Address** field — Specify the name or address of the associated device, using the following Syntax:

  <Type>[optFormat]<Address>.[optBit or StringLength]

  Where:

  – ***<Type>*** : Register type. Valid values are **X**, **Y**, **M**, **S**, **CS**, **CPV,TS,TPV,D**.

  – ***[Format]*** (optional): Parameter used for D address only, for formatting the value. Valid values are **D (***Double Word – 32 bits***), F** (*Floating Point*), **S** (*String*)

  – ***<Address>*** : Address of the device register.

  – ***[Bit]*** (optional): Use this parameter only for **D** (Device Register), to indicate which bit from the Word or DWord on the register will be read from and/or written to.

  – ***[StringLength]:*** Use this parameter only for **D** (Device Register) when you configure the format for string (ST), to indicate how many characters you will be reading from or writing to. Each D address is a word and it holds 2 characters.

➲ **Attention:**

- The Floating-point values are 4 bytes using 6 significant digits.
- X and Y addresses are in Octal format

STANDARD DRIVER WORKSHEET

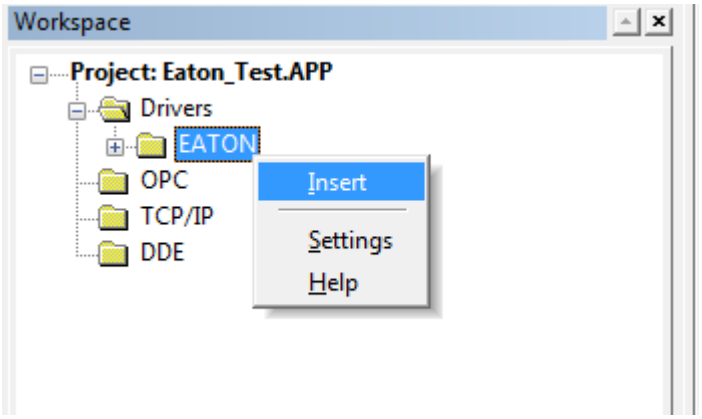
When you select the EATON driver and add it to your application, it has only a Main Driver Sheet by default (see previous section). However, you may insert additional Standard Driver Worksheets to define tag/register associations that are triggered by specific application behaviors. Doing this will optimize 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.

> **Note:**
> We recommend configuring device registers in sequential blocks in order to maximize 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.

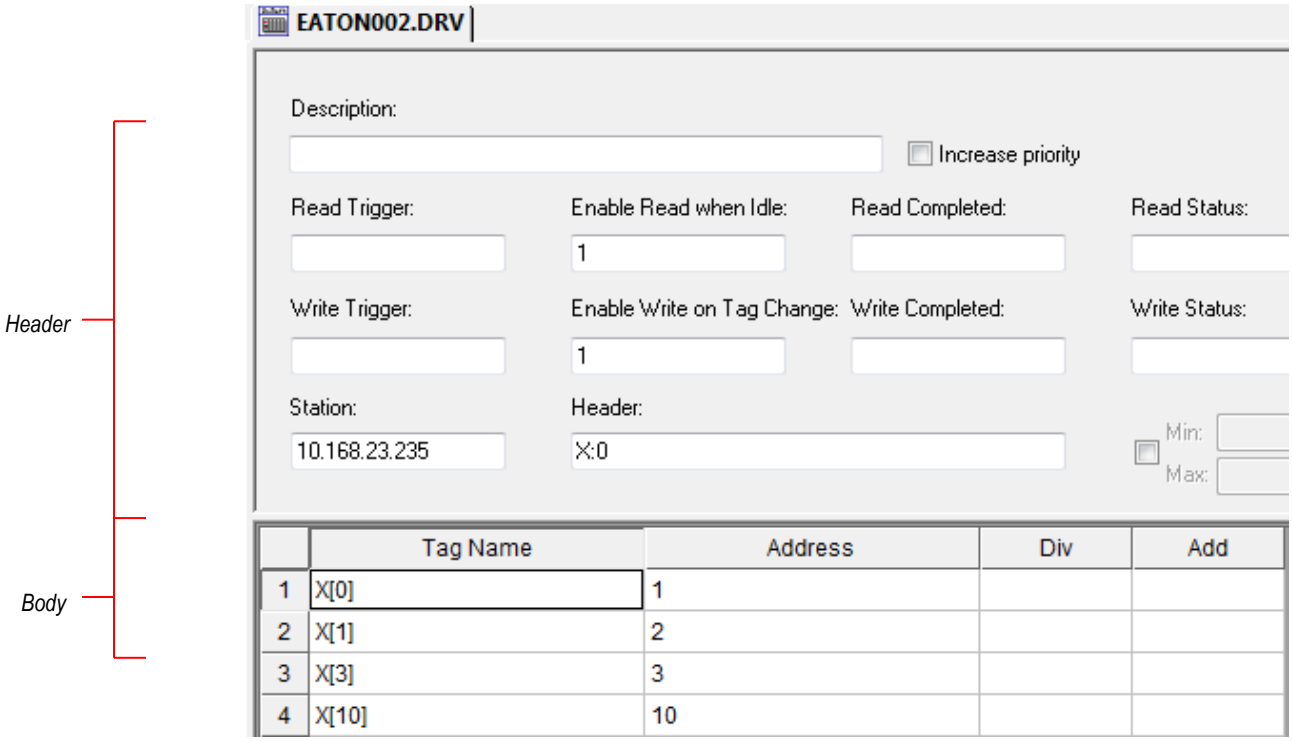
To insert a new driver worksheet:

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



Inserting a New Worksheet

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



EATON Driver Worksheet

> **Note:**
> Worksheets are numbered in order of creation, so the first worksheet is **`EATON001.drv`**.

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 **Header**, and **Address** fields use syntax that is specific to the EATON driver.

3. Configure the **Header** fields as follows:

   - **Station** field

     - For *Serial Communication*, specify here the **ELC ID** Number.
     - For *Ethernet*, use the following syntax:

       `<IP Address>:<opt ID Number>:<Opt Port Number>`

       Example — **`192.168.0.52`** or **`192.168.0.52:1:502`**

       Where:

       – ***`<IP Address>`*** is the IP address of the device on the Ethernet network.

       – ***`<opt ID Number>`*** optional PLC Slave ID in the Modbus network. If you do not configure this parameter, the driver will use the broadcast value **255**

       – ***`<opt Port Number>`*** optional TCP/IP Port number. If you do not configure this parameter, the default value of **502** will be used

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 cannot leave the Station field **blank**

- **Header** field: Specify the address of the first register of a block of registers on the target device. The addresses declared in the *Body* of the worksheet are simply offsets of this **Header** address. When Read/Write operations are executed for the entire worksheet (see **Read Trigger** and **Write Trigger** above), it scans the entire block of registers from the first address to the last.

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

  <Type>:<optAddressReference>

  Examples — **`D:0`**

  — **`X`**

  — **`TPV`**

  Where:

  – ***<Type>*** is the register type (**`X`**, **`Y`**, **`M`**, **`S`**, **`CS`**, **`CPV, CSX`**, **`CPVX, TS, TPV, D`**).

  – ***<optAddressReference>*** is the initial address (reference) of the configured type. This parameter is optional

  After you edit the **Header** field, Studio checks the syntax to determine if it is valid. If the syntax is invalid, then Studio automatically inserts a default value of **`S`**.

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

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

Type	Syntax	Valid Range	Comments
Input	**X**	0 – 377 (Octal)	Physical Digital Input – Address in octal format
Outputs	**Y**	0 – 377 (Octal)	Physical Digital Output – Address in octal format
Main Relay	**M**	0 - 4095	Internal Working Memory
Step Relay	**S**	0 - 1023	Sequential Function Chart usage
Timer Status	**TS**	0 - 255	Boolean Value = 1 when the Timer Present Value reaches the Set Value Reference
Timer Present Value	**TPV**	0 - 255	Represents the Present Value for the timers
Counter Status (16 bits)	**CS**	0 - 199	Boolean Value = 1 when the 16-bit Counter Present Value reaches the Set Value Reference
Counter Present Value (16 bits)	**CPV**	0 - 199	Represents the Present Value for the 16-bits Counters
Counter Status (32 Bits)	**CSX**	200 - 255	Boolean Value = 1 when 32-bit Counter Present Value reaches the Set Value Reference
Counter Present Value (32 bits)	**CPVX**	200 - 255	Represents the Present Value for the 32-bits Counters
Data Register	**D**	0 - 9999	General Data Register, and can be used in its native WORD format or as Double Words, Floating Point and Strings

Note:

The protocol does not support reading certain ranges of addresses by using a single command. For instance, it is not possible to read the D addresses 4095 and 4096 in a single block.

For this reason, you cannot have these addresses configured in the same standard driver sheet. If you try such configuration the driver will return error code **10**, indicating that you need to split the addresses into different driver groups.

Notice that the main driver sheet does not have this problem because it will break the addresses into different groups automatically.

he table below lists all the ranges that cannot intercalate in the same standard driver sheet:

Valid ranges for types M and D when using Standard Driver Sheet:

Header Field	Address Field
D	0000~4095
D	4096~8191
D	8192~9999
M	0000~1535
M	1536~4095

4. For each table row (i.e., each tag/register association), configure the **Address** field using the following syntax…

   For all register types other than **D**

   <AddressOffset>

   Examples — **10, F110, DW15,**

   For **D** registers *only*, use the following syntax:

   [optFormat]<AddressOffset>.[optStringLength or optBit]

   Example — **10:5**

   Where:

   – ***[optFormat]*** (optional): Parameter used for D address only, for formatting the value. Valid values are **D (***Double Word – 32 bits***), F** (*Floating Point*), **S** (*String*)

   – ***<AddressOffset>*** : Address of the device register. If you configured a <***optAddressReference>*** in the Header field, this value is added to the <opt***AddressReference>*** parameter to compose the specific address of the register in the block

   – ***[optBit]*** (optional): Use this parameter only for **D** (Device Register), to indicate which bit from the Word or DWord on the register will be read from and/or written to.

   – ***[optStringLength]:*** Use this parameter only for **D** (Device Register) when you configure the format for string (S), to indicate how many characters you will be reading from or writing to. Each D address is a word and it holds 2 characters.

Attention:

- Use Bit Write commands in the **D** Registers for the **Write on Tag Change** field only.
- The Floating-point value is stored in two consecutive **D** Registers, where the address value corresponds to the first Data Register position. You must ensure that you do not configure a non-existent address, or a conflict will occur.
- The Floating-point values are 4 bytes using 6 significant digits.
- Keep in mind that when using the **Write Trigger** feature, the driver writes to the entire block of registers from the first address through the last. If there is a register that has not been declared in the worksheet, and its address is within the block, then the register will receive a zero (0) value. Check the worksheet for holes in the address range.
- Writing bit values to the **D** registers is allowed; however there is no such function the Modbus protocol. In this case, the driver performs the *read-mask*-write operation, Before writing, the driver first reads the entire word, modifies the bit that will be written and then writes back to PLC the entire word. Even with this being a very fast operation, if in this period between reading and writing the PLC value changes, the driver will overwrite it with the value that is being processed.

For examples of how device registers are specified using **Header** and **Address**, see the following table:

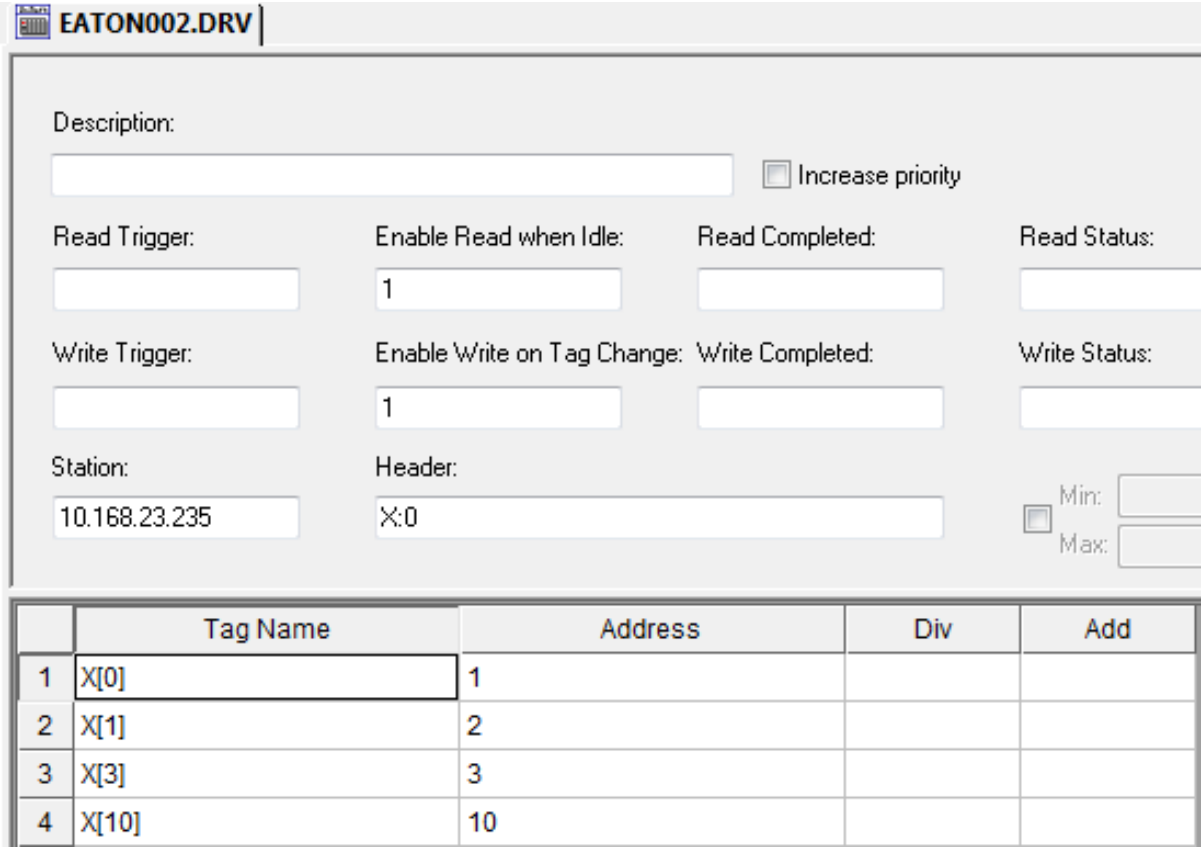
Device Register	Header	Address
X0	**X**	**0**
X107	**X**	**107**
X107	**X:100**	**7**
Y217	**Y:200**	**17**
Y217	**Y**	**217**
T5 (Status)	**TS**	**5**
T5 (Present Value)	**TPV**	**5**
C10 (Status)	**CS**	**10**
C10 (Present Value)	**CPV**	**10**
C210 (Status)	**CSX**	**210**
C210 (Present Value)	**CPVX**	**210**
D0	**D**	**0**
D100	**D**	**100**
D100	**D:100**	**0**
D110 and D111 (Floating Point)	**D**	**F110**
D110 and D111 (32-bits Double Word)	**D**	**DW110**
D200 – D229 (20 Characters String)	**D**	**S200.20**
D80.5	**D**	**80.5**
D80.15	**D**	**80.15**

➲ **Attention:**

You must not configure a range of addresses greater than the maximum block size (data buffer length) supported by the target device. The block size for the different registers are:

Header Type	Max Block Size (number of registers)
D, CPV and TPV	60
CPVX	30
M and S	1024
X and Y	255

Driver Worksheet sample configuration



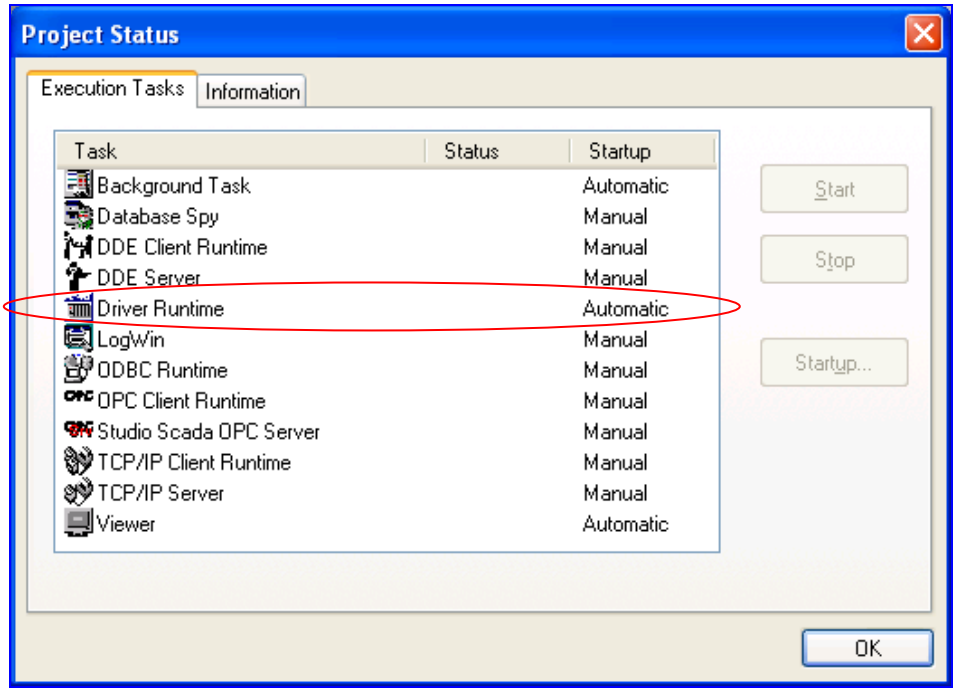
	Tag Name	Address	Div	Add
1	X[0]	1		
2	X[1]	2		
3	X[3]	3		
4	X[10]	10		

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 EATON 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 Standard 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	N/A	N/A
1	Error sending buffer	- If using TCP/IP, the connection with the slave device might have been reset - If using serial connection the serial port might have failed to send the output buffer	- Check the switches and network connection for failures - Run tests on your serial port to identify possible failures
2	Invalid station on slave response	The response sent by the slave does not match the request made by the driver	- Check for multiple masters on the network. - If you are communicating with multiple devices on a serial connection and your time out is too short you might see this error. Check the time out configuration settings (see section "Configuring the communication settings") and increase the time out.
3	Invalid Checksum	- Electrical interference on your serial connection - Serial port configuration settings are not matching the PLC configuration	Check the physical connections on your serial network and verify if it is within the recommended physical layer specifications. Check the serial port configuration settings (see section "Configuring the communication settings") and make sure that it matches the PLC Serial port settings.
4	Invalid sequence number	The time out configured in the driver settings is too short.	Check the time out configuration settings (see section "Configuring the communication settings") and increase the time out.
5	Multiple Ranges Error	Multiple ranges of types D or M used in the Standard Driver Sheet	When using header D or M in the Standard Driver Sheet you must take special precaution with the range and follow this rule: Don't use the 2 different ranges inside the same sheet. If you need more information check the table of page 9 that describes the valid ranges. Valid ranges for types M and D when using Standard Driver Sheet: **Header Field** / **Address Field** **D** 0000~4095 **D** 4096~8191 **D** 8192~9999 **M** 0000~1535 **M** 1536~4095
6	Write not supported	A write operation was attempted on a read only address	Check if you are trying to write to X registers and modify your driver configuration in case you are doing so. Writing to X registers is not allowed.

<table>
<tr><td>7</td><td>Invalid function code on slave response</td><td>The slave replied with a function code that does not match the request</td><td>- Check for multiple masters on the network.
- If you are communicating with multiple devices on a serial connection and your time out is too short you might see this error. Check the time out configuration settings (see section "Configuring the communication settings") and increase the time out.</td></tr>
<tr><td>8</td><td>Slave returned exception code</td><td>The slave could not comply with the driver request</td><td>Enable the protocol analyzer to see more details and the exception code returned, the message will help you to identify wrong settings (e.g.: unsupported address range) or list possible errors in the PLC.</td></tr>
<tr><td>-15</td><td>Timeout waiting start a message</td><td>On a serial communication, the PLC is not responding. It can be:
- Wrong communication settings, such as Baud Rate, Parity, Stop Bits, Data Bits, Protocol format (ASCII or RTU)
- Wrong Station value</td><td>- Check Serial Communication settings. It must match with the PLC settings
- Check cable connections
- Check Station field for wrong ID number or wrong IP Address</td></tr>
<tr><td>-40</td><td>Invalid bit write</td><td>Tried to write bits from a D register using Write Trigger</td><td>Bit writing on D registers is only possible using Enable Write on Tag Change. Change your Standard Driver Worksheet configuration to use it instead of Write Trigger</td></tr>
<tr><td>-38</td><td>Invalid station</td><td>Invalid configuration in the Station field, such as leaving it blank or configuring an invalid IP Address for the Ethernet communication</td><td>Check the Station field on your driver sheet. Make sure that it is not blank and, in the case of Ethernet Communication, the IP address has all the 4 octets</td></tr>
<tr><td>-39</td><td>Invalid block size</td><td>Standard Driver Sheet with an offset between the lowest and the highest address higher than the allowed amount for that device register type</td><td>Check your driver sheet to make sure you respect the block size limitations, which are:
<table><tr><th>Header Type</th><th>Max Block Size (number of registers)</th></tr><tr><td>D, CPV and TPV</td><td>60</td></tr><tr><td>CPVX</td><td>30</td></tr><tr><td>M and S</td><td>1024</td></tr><tr><td>X and Y</td><td>255</td></tr></table></td></tr>
<tr><td>-37</td><td>Invalid header</td><td>This error happens when you configure a {Tag} between curly brackets on the Header field and he value of this {Tag} does not comply with the Header syntax</td><td>Change the Tag value in order to comply with the Header Syntax.</td></tr>
<tr><td>-42</td><td>Invalid bit number</td><td>This error happens when you configure the D register to access a bit higher than 15 or, in the case of Double Word format, higher than 31</td><td>Check your driver worksheet Address configuration, to make sure that the bits from a D register go from 0 to 15 in the Word format (Standard) and from 0 to 31 on the DW format</td></tr>
<tr><td>-60</td><td>Connection Error</td><td>Error to establish a TCP/IP connection with the Slave device. Possibly wrong IP Address or Port Number in the Station field</td><td>Check the IP Address, port and ID number in the Station field
Try to ping the IP address that you configured in the Station field</td></tr>
</table>

⇨ **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 **Protocol Analyzer,** right-click in the *Output* window and select the desired options from the pop-up menu.

You can also use the *Remote LogWin* module to establish an event log on a remote unit that runs Windows CE

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 (ELC Soft). 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.

If you must contact us for technical support, please have the following information available:

- **Operating System and Project Information** (type and version): To find this information, select **Help** → **Support Information**.
- **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.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of Changes
A	1.00	Paulo R. Balbino	25 Feb 2010	Initial version
B	1.1	André Körbes	16 Nov 2010	Fixed problem with multiple stations on Main Driver Sheet Fixed problem with address validation on Main Driver Sheet
C	1.2	Paulo Balbino	23 Aug 2012	Resolved issue with writing real values when using the Serial RTU mode