ABCIP – *Driver Version 11.14*

Doc. Revision AU – Nov 28, 2018

ABCIP Communication Driver

Driver for TCP/IP Ethernet Communication
with Allen-Bradley Devices Using the CIP Protocol

Contents

Introduction

The ABCIP driver enables communication between the Studio system and compatible target devices — including Allen-Bradley ControlLogix, FlexLogix, CompactLogix, and MicroLogix PLCs — according to the specifications discussed in this document. The ABCIP driver communicates via the Allen-Bradley Common Industrial Protocol (CIP) over Ethernet/IP

This document will help you to select, configure and execute the ABCIP 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 ABCIP driver in the Studio system.
- **Configuring the Device**: Describes how the target device must be configured to receive communication from the ABCIP driver.
- **Configuring the Driver**: Explains how to configure the ABCIP driver in the Studio system, including how to associate database tags with device registers.
- **Routing Communication**: Describes how to communicate with additional devices using the ControlLogix Backplane as a device network router.
- **Executing the Driver**: Explains how to execute the ABCIP 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 environment. If you are not familiar with Windows, then we suggest using the **Help** feature as you work through this guide.

General Information

This chapter identifies all of the hardware and software components required to implement Ethernet communication between the ABCIP driver in Studio and a target device using the CIP protocol.

The information is organized into the following sections:

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

Device Specifications

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

- **Manufacturer**: Allen Bradley/Rockwell
- **Compatible Equipment**:
  - ControlLogix 5000 Family with 1756-ENET or 1756-ENBT module installed
  - FlexLogix
  - CompactLogix
  - MicroLogix (1100 Series A and B, 1400)
  - PLC5 and SLC500 though routing
- **Device Programming Software**: Rockwell RSLogix5000 and Rockwell RSLogix500

For a description of the device(s) used to test driver conformance, see “Conformance Testing”.

Network Specifications

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

- **Device Communication Port**:
  - for ControlLogix 5000 Family, Ethernet port on the 1756-ENET module
  - for CompactLogix and FlexLogix, the built-in Ethernet Channel
  - for MicroLogix 1100 and 1400, Ethernet Channel (Channel 1)
- **Physical Protocol**: Ethernet
- **Logic Protocol**: CIP over Ethernet/IP
- **Device Runtime Software**: None
- **Specific PC Board**: Ethernet port
- **Cable Wiring Scheme:** Regular Ethernet cable

Driver Characteristics

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

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

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

- Windows XP/7/8
- Windows CE

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

The ABCIP driver supports the following register types:

For ControlLogix, FlexLogix and CompactLogix					
Data Type	**Length**	**Write**	**Read**	**Bit**	**Comments**
BOOL	1 Bit	●	●	●	Reads and writes the BOOL data type, which consists of a bit value (0 or 1).
SINT	1 Bytes	●	●	●	Reads and writes the SINT data type, which consists of a Byte 8 bits signed Integer Value (-128 to 127).
INT	2 Bytes	●	●	●	Reads and writes the INT data type, which consists of a WORD 16 bits signed Integer Value (-32768 to 32767).
DINT	4 Bytes	●	●	●	Reads and writes the DINT data type, which consists of a DWORD 32 bits signed Integer Value (-2,147,483,648 to 2,147,483,647).
REAL	4 Bytes	●	●	●	Reads and writes the REAL data type, which consists of a 32Bits-IEEE Floating Point Value (-9.99x10^{37} to 9.99x10^{37}).
STRING	Configurabl	●	●	–	Reads and writes the STRING data type, which stores up to 82

	e				characters
LINT	64 Bytes	•	•	•	Reads and writes the LINT data type, which consists of a 64 bits signed Integer Value (-9223372036854775808 to - 9223372036854775807).

For MicroLogix 1100/1400								
Register Type	**Length**	**Write**	**Read**	**Bit**	**Integer**	**Float**	**String**	**BCD**
O (Output)	1 Byte	•	•	•	•	–	–	•
I (Input)	1 Byte	–	•	•	•	–	–	•
S (Status)	2 Bytes	–	•	•	•	–	–	–
B (Binary)	1 Byte	•	•	•	•	–	–	•
T (Timer)	6 Bytes	•	•	–	•	–	–	–
C (Counter)	6 Bytes	•	•	–	•	–	–	–
R (Control)	6 Bytes	•	•	–	•	–	–	–
F (Float)	4 Bytes	•	•	–	–	•	–	–
N (Integer File)	2 Bytes	•	•	•	•	–	–	•
ST (String File)	*n* Bytes	•	•	–	–	–	•	–
L (Long)	4 Bytes	•	•	–	•	–	–	–

Conformance Testing

The following hardware/software configuration was used to test driver conformance:

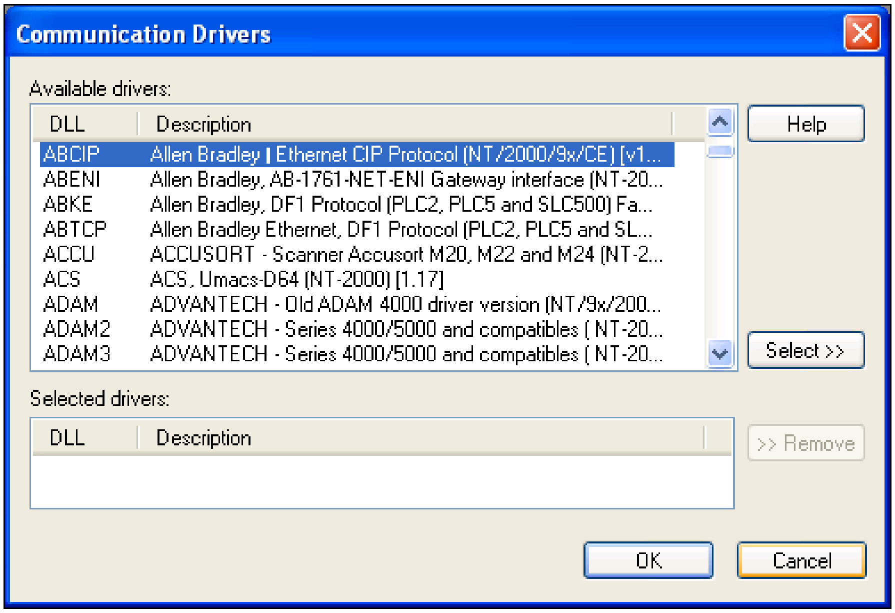
- **Driver Configuration**:
  - **Protocol**: CIP over Ethernet TCP/IP
- **Cable**: Regular Ethernet cables

Driver Version	Studio Version	Operating System (development)	Operating System (target)	Equipment
11.14	8.1 + SP1	Windows 8	▪ Windows 7 ▪ Windows 8 ▪ Windows CE	• PLC Allen Bradley 1756-L62 ControlLogix 5562 Controller + 1756EN2T Module firmware revision 20.12 • PLC Allen Bradley 1756-L75 ControlLogix 5570 Controller + 1756EN2T Module firmware revision 27.11 • PLC Allen Bradley 1756-L83 ControlLogix 5580 Controller + 1756EN2T Module firmware revision 28.11 • PLC Allen Bradley 1756-L83 ControlLogix 5580 Controller + 1756EN2T Module firmware revision 28.11 • PLC Allen Bradley MicroLogix 1100 1763-L16BWA Ser.A Rev.B. • PLC5/80 via DH+ routing • SLC5/04 via DH+ routing

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 ABCIP driver for your Studio application:

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



Available drivers:

DLL	Description
ABCIP	Allen Bradley | Ethernet CIP Protocol (NT/2000/9x/CE) [v1...
ABENI	Allen Bradley, AB-1761-NET-ENI Gateway interface (NT-20...
ABKE	Allen Bradley, DF1 Protocol (PLC2, PLC5 and SLC500) Fa...
ABTCP	Allen Bradley Ethernet, DF1 Protocol (PLC2, PLC5 and SL...
ACCU	ACCUSORT - Scanner Accusort M20, M22 and M24 (NT-2...
ACS	ACS, Umacs-D64 (NT-2000) [1.17]
ADAM	ADVANTECH - Old ADAM 4000 driver version (NT/9x/200...
ADAM2	ADVANTECH - Series 4000/5000 and compatibles (NT-20...
ADAM3	ADVANTECH - Series 4000/5000 and compatibles (NT-20...

Selected drivers:

DLL	Description

Communication Drivers Dialog

3. When the **ABCIP** 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 not necessary to install any other software on your computer 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. To download control logic to a ControlLogix device, you must also use RSLogix5000. For more information, please consult the documentation provided by the device manufacturer.

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

According to the manufacturer's documentation, configure a valid IP address on the target device and then place it in RUN mode. (For ControlLogix 5000 Series, you need to configure the 1756-ENET module. For MicroLogix 1100, you need to configure Ethernet Channel 1. If there is neither network nor connection problems, the device should now be ready to receive communication from your Studio application.

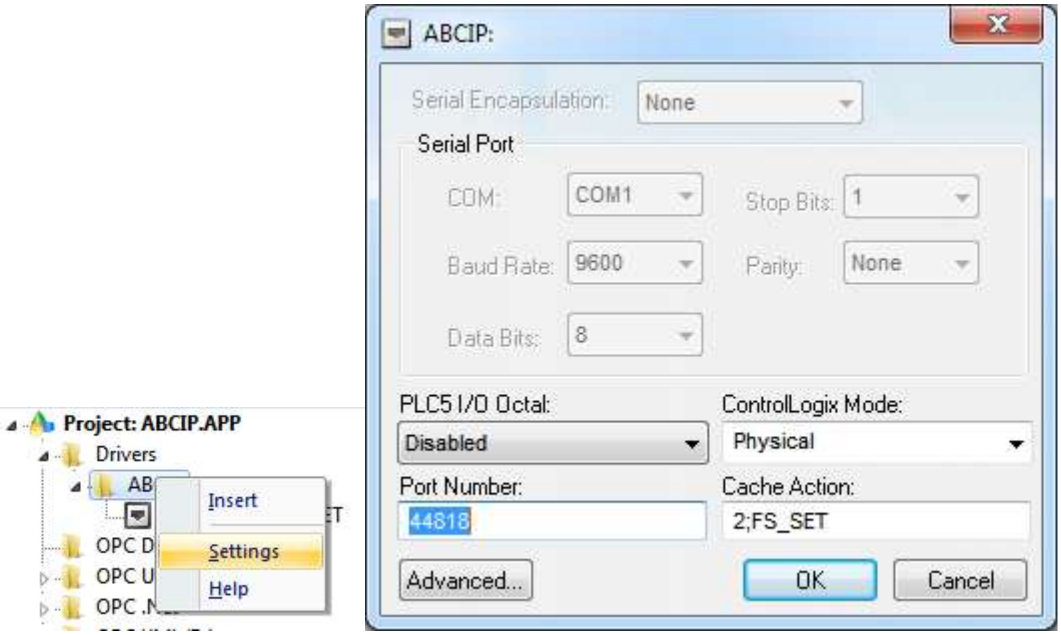
Configuring the Driver

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 ABCIP driver-specific settings and procedures will be discussed here. To configure the communication settings for the ABCIP driver:

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



Serial Encapsulation: None

COM: COM1

Stop Bits: 1

Baud Rate: 9600

Parity: None

Data Bits: 8

PLC5 I/O Octal: Disabled

ControlLogix Mode: Physical

Port Number: 44818

Cache Action: 2;FS_SET

ABCIP: 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
PLC5 I/O Octal	`Disabled`	**Disabled** or **Enabled**	This setting will affect only the PLC5 when using a network routing. See the Appendix for more information. ▪ **Disabled** – Input (I) and Output (O) in PLC5 will be treated as decimal value. ▪ **Enabled** – Input (I) and Output (O) in PLC5 will be treated as octal value.
Port Number	`44818`	**Any number**	The port number configured here affects all TCP connections created by the driver.
ControlLogix Mode	`Symbolic`	**Symbolic** or **Physical**	This setting affects how the driver communicates with PLCs of ControlLogix family. - The **Physical** mode will read the variables from the PLC using its memory address, increasing the general performance of the driver. This requires that on the first read the driver uploads the information for all variables and its data types, which will imply a performance penalty on the first read. After that, the driver creates a cache of the program variables which is stored in the application's Config folder. See also the RESETCACHE address information.

Parameters	Default Values	Valid Values	Description
			- The **Symbolic** mode will read variables using their full names. This will create a performance hit when tag names are large, and especially if UDTs are used.

Cache Action	**`Empty`**	**Empty,**
		`1,`
		`2;<PLC TAG ADDRESS>`
		or
		`3;<PLC TAG ADDRESS>`
			This setting affects the automatic control of the cache file when the Physical mode is enabled.
			- `Empty`: No action is taken
			- `1`: The cache is reset on the first read of the station after the driver starts.
			- `2;<PLC TAG ADDRESS>`: The cache is reset when the tag value is 1. The tag must be of BOOLEAN type. This operation is the safest one in the following scenario:
			• The tag must be set to 1 on the first scan of the PLC after changing from Program to Run mode, and must not be further modified by the PLC.
			• The driver will read the tag after each read operation. If the tag value is 1, the cache is invalidated, the read is cancelled with error code 509, and the tag is written back to 0.
			- 3`;<PLC TAG ADDRESS>`: Combination of 1 and 2. The cache is reset on first read and whenever the PLC Tag Address value is 1. This option is safer to use when there is a possibility of copying the project files to use with different equipment.

➲ **Attention:**

When using the **Physical** *ControlLogix Mode* option, the driver cache must be updated whenever the PLC program is changed. The default value of the ***Cache Action*** parameter is empty, therefore, if the PLC program changes you MUST either use the *RESETCACHE* command to refresh the Tag IDs, or change the Cache Action parameter to another value that would force this cache to be recreated in order to guarantee the synchronism between the Tags in the Driver Worksheets and the PLC. Otherwise the driver will not communicate properly.

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.

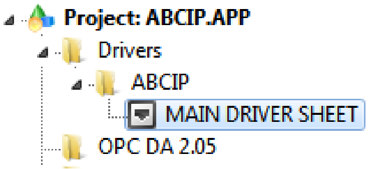
The configuration of these worksheets is described in detail in the “Communication” chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

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

Main Driver Sheet

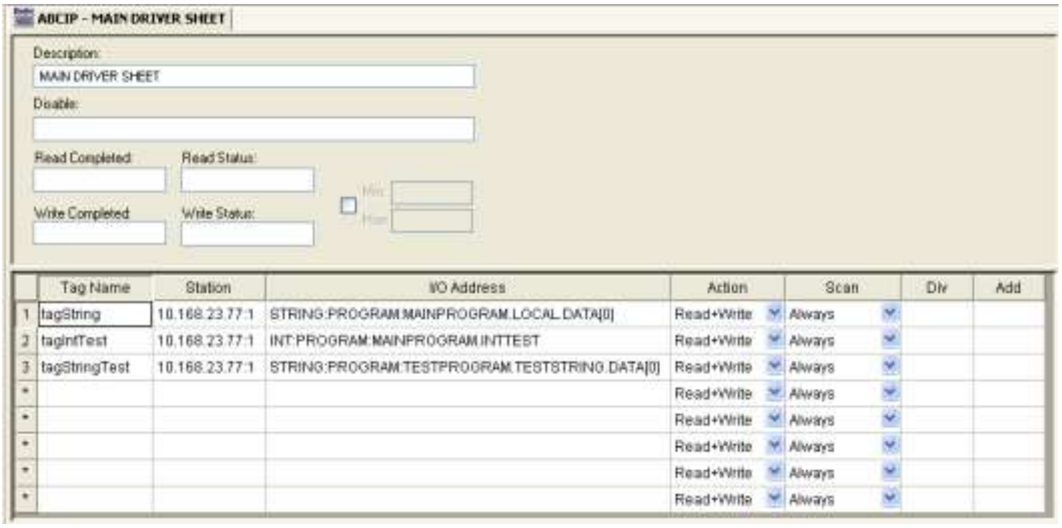
When you select the ABCIP driver and add it to your application, Studio automatically inserts the *Main Driver Sheet* in the *ABCIP* driver subfolder. To configure the Main Driver Sheet:

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



Main Driver Sheet in the ABCIP Subfolder

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



ABCIP - MAIN DRIVER SHEET

Description: MAIN DRIVER SHEET

Disable:

Read Completed:

Read Status:

Write Completed:

Write Status:

□ Min: Max:

	Tag Name	Station	I/O Address	Action	Scan	Div	Add
1	tagString	10.168.23.77:1	STRING:PROGRAM:MAINPROGRAM.LOCAL.DATA[0]	Read+Write	Always		
2	tagIntTest	10.168.23.77:1	INT:PROGRAM:MAINPROGRAM.INTTEST	Read+Write	Always		
3	tagStringTest	10.168.23.77:1	STRING:PROGRAM:TESTPROGRAM.TESTSTRING.DATA[0]	Read+Write	Always		
*				Read+Write	Always		
*				Read+Write	Always		
*				Read+Write	Always		
*				Read+Write	Always		
*				Read+Write	Always		

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 ABCIP driver.

> ✎ **Note:**
>
> The instructions below describe only how to communicate directly with ControlLogix 5000 Family and MicroLogix 1100 devices. For more information about how to communicate with PLC5 Family and SLC500 Family devices using the ControlLogix PLC as a device network router, please see the **Appendix** at the end of this document.

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

   - **Station** field — Specify the IP Address of the device and the slot number, using the following syntax:

     For Contrologix ***<IP Address>*:*<optSlotNumber>***

     For Micrologix ***<Family>*:*<IP Address>***

     Example — **`10.168.23.77:0`** or **`1100:192.168.1.53`**

     Where:

     – ***<Family>***: If you do not specify this parameter, Studio assumes it is a ControlLogix 5000 Family device. Otherwise, you can specify **`1100`** for a MicroLogix 1100 or 1400 device.

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

     – ***<optSlotNumber>*** is the number of the slot in the backplane in which the CPU module is configured. If you omit this parameter, the driver will use the Slot number ***0***

     You can also specify a string tag (e.g. **`{station}`**), but the tag value 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.

ControlLogix, FlexLogix and CompactLogix

- **I/O Address** field — Specify the name or address of the associated register.

  For **BOOL**, **SINT**, **INT**, **DINT, LINT** and **REAL** types:

  [Data Type]*:*<Logix Tag Name>* /*[Bit]

Examples — **`SINT:PLCTAG; REAL:LEVEL[10]`**

For **STRING** type:

`STRING:`*`<Logix Tag Name>`*

Examples —**`STRING:TEXT`**

For **LINT** type:

`For support for 64 bits integers, configure LINT Tag to be a string datatype instead of an integer datatype.`

For **PROGRAM** tags outside of the Controller Tags database (**BOOL**, **SINT**, **INT**, **DINT** and **REAL** types):

`[Data Type]`*`:PROGRAM:`*`<Program Name>.<Logix Tag Name>`

Example — **`DINT:PROGRAM:MAINPROGRAM.UDT1.M1`**

For **PROGRAM** tags outside of the Controller Tag database (**STRING** type):

`STRING:PROGRAM:`*`<Program Name>.<Logix Tag Name>`*

Example — **`STRING:PROGRAM:MAINPROGRAM.HMI_STRING`**

For resetting the program variables cache, use the following special address:

`RESETCACHE`

This is a special address, which can only be written to, that will cause the cache file to be deleted and recreated the next time a variable is read.

Where:

- ***`[Data Type]`*** (optional) is the data type of the Logix tag. Use one of the following: **`BOOL`**, **`INT`**, **`SINT`**, **`DINT`**, **`LINT`**, **`REAL`** or **`STRING`**. If this parameter is omitted, then the driver assumes **`DINT`** data type. Please see the table on page 4 for a complete description.
- ***`<Logix Tag Name>`*** is the tag name in RSLogix.
- ***`<Program Name>`*** is the name of the program outside of the Controller Tags database.

✍ **Notes:**

- The maximum Block Size to be sent is 544 bytes and the maximum Block Size to be received is 492 bytes, following the protocol. However, there are some many variable that influence the number of the bytes such as data type, dimension arrays, tag names and amount of the tags in one worksheet. In other words, the maximum elements, it can have in the sheets, depend of the all variables mentioned above.
- For better performance use array tags. Using array tags it is possible to have: INT (240 elements), DINT (120 elements), SINT (480 elements) and REAL (120 elements) using Standard Driver Sheet.
- The cache file is created only when the ***ControlLogix*** Mode setting is on **Physical**. It is stored in the application's Config folder, and its name is composed of the prefix `ABCIP_CACHE_`, the IP address as a 32-bit unsigned number and the CPU slot number. The RESETCACHE address should be used with the same station as the other tags being read. In the Main Driver Sheet, this line should be configured as Write only. Attempting to read this address will raise an error of "Unknown Tag", code 504. To reset the cache manually, write any value to the tag configured to that address. To reset the cache automatically, see the **Cache Action** configuration in the driver settings.
- The maximum size of a string that will be accepted by the driver is 380 characters.
- For program tags when communicating to PLC with firmware equal or greater than 21 the driver will use symbolic mode for the entire group.
- When specifying struct tags, always specify the member name, if the whole struct is specified block size error can occur Example: testStruct.speed1 (speed1 is the member name). The individual bits of type BOOL cannot be accessed.

MicroLogix 1100/1400

- **I/O Address** field — Specify the address of the associated register.

For **Output (O)** and **Input (I)** files:

`<File Type>:0.[Format]<word number> /[Bit]`

Examples — **`O:0.2/4`** or **`I:0.2/7`**

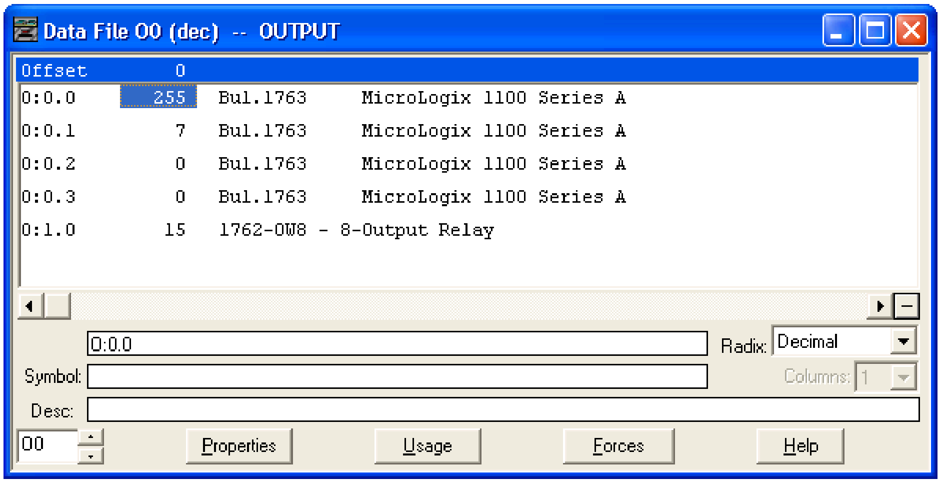
✍ **Notes:**

- In the Micrologix PLC, even though they belong to the 500 Series family, the address is not *Slot-based*, but *0-offset* based. This is why after the ":" you should always configure the number **0** (zero) when using the Micrologix PLCs
- When you use additional I/O modules on the Micrologix 1100/1400 PLCs, in order to

know exactly how to configure the ***word number***, you should check how the RSLogix500 sees these addresses.

On the screenshot below, the CPU has only six digital Outputs. They all fit on the First word. However, the CPU reserves the next 3 words as well. The first External Output Module in then seen by the PLC as the word number **4** (5th word)

So, for example, in order to read the bit **2** coming from an external 8-Output Relay module, the RSlogix500 shows it as **O:1.0/2.** On the ABCIP driver you must configure it as **O:0.4/2**



Data File O0 (dec) -- OUTPUT

```
Offset         0
O:0.0        255   Bul.1763     MicroLogix 1100 Series A
O:0.1          7   Bul.1763     MicroLogix 1100 Series A
O:0.2          0   Bul.1763     MicroLogix 1100 Series A
O:0.3          0   Bul.1763     MicroLogix 1100 Series A
O:1.0         15   1762-OW8 - 8-Output Relay
```

O:0.0

Radix: Decimal

Symbol:

Columns: 1

Desc:

O0

For **Status (S)**, **Binary (B)** and **Integer (N)** files:

<File Type><File Number>: [Format]<Address> /[Bit]

Examples — **`N7:W150/2`** or **`N7:150/2`**

For **Timer (T)**, **Counter (C)** and **Control (R)** files:

Unsigned values - ***<File Type><File Number >: [Format]<Address>.<Element>***

Signed values - ***<File Type><File Number >:S[Format]<Address>.<Element>***

Examples — **`T4:W0.PRE`**, **`T4:0.PRE`** or **`T4:S0.PRE`** ***(Signed)***

For **String (ST)** file:

ST<File Number>: [Format]<Address>.[Number of Bytes]

Examples — **`ST15:S0.50`** or **`ST15:0`**

For **Float (F)** file:

F<File Number>:[Format]<Address>

Examples — **F8:F0** or **F8:0**

For **Long (L)** file:

L<File Number>:[Format]<Address>

Examples — **L9:L0** or **L9:0**

Where:

- ***<Register Type>*** is the type of register. Use one of the following: **O**, **I**, **S**, **B**, **N**, **T**, **C**, **R**, **F,L** or **ST**. Please see the table on page 4 for a complete list.
- ***<Slot Number>*** is the number of the PLC slot in which the Output (O) or Input (I) is located.
- ***<Type Group>*** is the number of the register group in which the register is configured.
- ***[Format]*** (optional) is how the data should be handled. Use one of the following: **W** for Word, **B** for BCD, or **S** for String. If ***S*** is configured for Timer, Counter or Control it will be *signed value*.
- ***<Address>*** is the address of the register in the specified ***<Slot Number>*** or ***<Type Group>***.
- ***[Bit]*** (optional) is the bit number (from 0 to 15) of the register. Word format only.
- ***[Number of Bytes]*** (optional) is the maximum size of the ASCII/STRING data.
- ***<Element>*** is the element type for Timers (T), Counters (C) and Controls (R), according to the following table:

Register	Elements																	
	DN	PRE	ACC	EN	TT	UA	UN	OV	CD	CU	FD	IN	UL	ER	EM	EU	LEN	POS
Timer	R	R/W	R/W	R	R	–	–	–	–	–	–	–	–	–	–	–	–	–
Counter	R	R/W	R/W	–	–	R	R	R	R	R	–	–	–	–	–	–	–	–
Control	R	–	–	R	–	–	–	–	–	–	R	R	R	R	R	R	R/W	R/W

Note:

If PLC5 is used the table above is not valid. It is possible to write using any configurable sub-element.

For examples of how the **I/O Address** field should be completed in order to address specific tags and device registers, please refer to the following tables:

For ControlLogix, FlexLogix and CompactLogix Devices		
Data Type	**Tag Name on the Device**	**I/O Address Field in Studio**
BOOL	RETURN_3_LOW_TEMP_ALARM	`BOOL:RETURN_3_LOW_TEMP_ALARM`
	REC_ZONES[3]	`BOOL:REC_ZONES[3]`
	MYUDT2[1,0].BOOL	`BOOL:MYUDT2[1,0].BOOL`
	BOOLAR1[8] **(Program Tag)**	`BOOL:PROGRAM:MAINPROGRAM.BOOLAR1[8]`
INT	HOT_RM_TEMP_DEGF	`INT:HOT_RM_TEMP_DEGF`
	INT[4]	`INT:INT[4]`
	MYUDT2[1,1].INT	`INT:MYUDT2[1,1].INT`
	INT **(Program Tag)**	`INT:PROGRAM:MAINPROGRAM.INT`
SINT	Control_1	`SINT:CONTROLl_1`
	Control[1,2,0]	`SINT:CONTROL[1,2,0]`
	Device.Parameter[2]	`SINT:DEVICE.PARAMETER[2]`
	SINTAR1[4] **(Program Tag)**	`SINT:PROGRAM:MAINPROGRAM.SINTAR1[4]`
LINT	LINT11	`LINT:LINT11`
INT	INTArr[4]	`INT:INTARR[4]`
	INT_Test.0	`INT:INT_Test/0`
DINT	DINT1B[2,0]	`DINT1B[2,0]`

For ControlLogix, FlexLogix and CompactLogix Devices		
Data Type	**Tag Name on the Device**	**I/O Address Field in Studio**
	MYUDT2[1,0].MEMBER1	`MYUDT2[1,0].MEMBER1`
	CONTROLLERUDTTAG[9].DINTAR[5]	`DINT:CONTROLLERUDTTAG[9].DINTAR[5]`
	DINTAR2[1,5] **(Program Tag)**	`DINT:PROGRAM:MAINPROGRAM.DINTAR2[1,5]`
REAL	MIG380REAL	`REAL:MIG380REAL`
	REALAR2[1,3]	`REAL:REALAR2[1,3]`
	YUDT2[1,0].REAL	`REAL:YUDT2[1,0].REAL`
	MYUDT[0,8].REAL **(Program Tag)**	`REAL:PROGRAM:MAINPROGRAM.MUUDT[0,8].REAL`
STRING	STRING1	`STRING:STRING1`
	STRINGAR[1]	`STRING:MYUDT2[0,0].STRINGAR[1]`
	STRINGAR3[1,0,4] **(Program Tag)**	`STRING:PROGRAM:MAINPROGRAM.STRINGAR3[1,0,4]`
	HMI_STRING	`STRING:HMI_STRING`
HMI_STRING	HMIS[6,4,3]	`STRING:HMIS[6,4,3]`
	MYUDT2[1,1].HMI_STRINGAR[2]	`STRING:MYUDT2[1,1].HMI_STRINGAR[2]`
	HMI_STRING **(Program Tag)**	`STRING:PROGRAM:MAINPROGRAM.HMI_STRING`

For MicroLogix 1100/1400		
Register Type	**Register Address on the Device**	**I/O Address Field in Studio**
Input	I:0	`I:0.0`
	I:0/10	`I:0.0/10`
	I:0/17	`I:0.0/17`
	I:3	`I:0.3`

For MicroLogix 1100/1400		
Register Type	**Register Address on the Device**	**I/O Address Field in Studio**
	I:6/4	`I:0.6/4`
Output	O:0	`O:0.0`
	O:0/10	`O:0.0/10`
	O:0/17	`O:0.0/17`
	O:3	`O:0.3`
	O:6/4	`O:0.6/4`
Status	S:0/5	`S:0/5`
	S:10	`S:10`
	S:20/7	`S:20/7`
Binary	B3:0	`B3:0`
	B3:10	`B3:10`
	B3:10/7	`B3:10/7`
Integer	N7:0	`N7:0`
	N7:0/10	`N7:0/10`
	N7:50	`N7:50`
Timer	T4:0.ACC	`T4:0.ACC`
	T4:0.PRE	`T4:W0.PRE`
	T15:0.EN	`T15:0.EN`
	T15:0.ACC	`T15:S0.ACC (signed value)`
	T15:1.ACC	`T15:1.ACC`
Counter	C5:0.ACC	`C5:0.ACC`

For MicroLogix 1100/1400		
Register Type	**Register Address on the Device**	**I/O Address Field in Studio**
	C5:1.PRE	**C5:S1.PRE (signed value)**
	C20:15.UA	**C20:15.UA**
Control	R6:0.LEN	**R6:0.LEN**
	R6:0.POS	**R6:0.POS**
	R6:1.POS	**R6:1.POS**
Float	F8:0	**F8:0**
	F8:5	**F8:5**
	F8:10	**F8:10**
String	ST15:0	**ST15:0**
	ST15:1 (String: maximum 50 bytes)	**ST15:S1.50**
	ST15:2 (String: maximum 10 bytes)	**ST15:S2.10**
Long	L9:0	**L9:0**
	L9:5	**L9:5**
	L9:10	**L9:10**

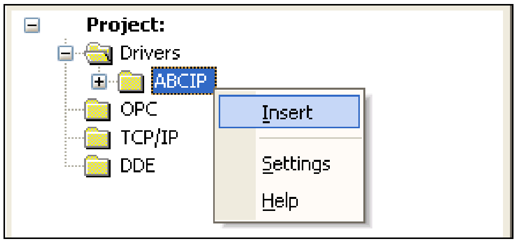
Standard Driver Worksheet

When you select the ABCIP 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.

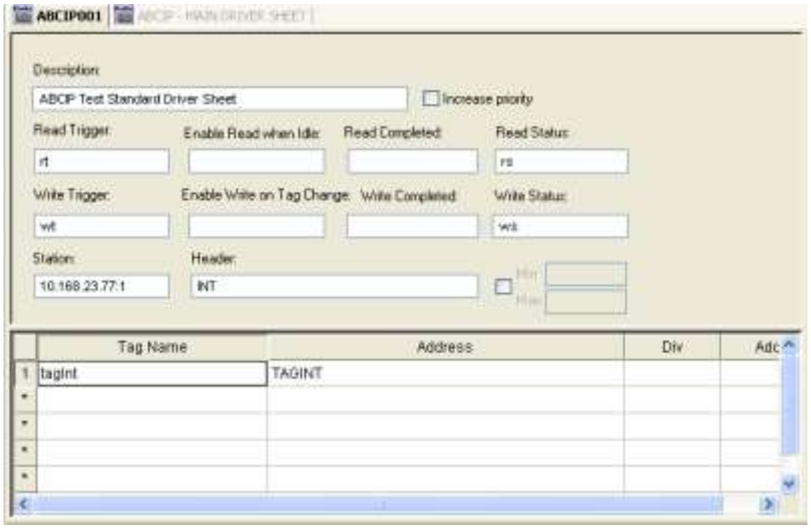
To insert a new Standard Driver Worksheet:

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



Inserting a New Worksheet

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


ABCIP001 | ABCIP - MAIN DRIVER SHEET

Description: ABCIP Test Standard Driver Sheet

□ Increase priority

Read Trigger: rt
Enable Read when Idle:
Read Completed:
Read Status: rs

Write Trigger: wt
Enable Write on Tag Change:
Write Completed:
Write Status: ws

Station: 10.168.23.77:1
Header: INT
□ Min: Max:

	Tag Name	Address	Div	Adc
1	tagInt	TAGINT		
*				
*				
*				
*				

ABCIP Driver Worksheet

> **Note:**
>
> Worksheets are numbered in order of creation, so the first worksheet is **`ABCIP001.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, the **Station**, **Header**, and **Address** fields use syntax that is specific to the ABCIP driver.

> ✍ **Note:**
>
> The instructions below describe only how to communicate directly with ControlLogix 5000 Family and MicroLogix 1100/1400 devices. For more information about how to communicate with PLC5 Family and SLC500 Family devices using the ControlLogix PLC as a device network router, please see the **Appendix** at the end of this document.

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

   - **Station** field — Specify the IP Address of the device and the slot number, using the following syntax:

     For Contrologix ***<IP Address>**:**<optSlotNumber>***

     For Micrologix ***<Family>**:**<IP Address>***

     Example — **`10.168.23.77:0, 10.168.23.77`** or **`1100:192.168.1.53:1`**

     Where:

     – ***<Family>*** : If you do not specify this parameter, Studio assumes it is a ControlLogix 5000 Family device. Otherwise, you can specify **`1100`** for a MicroLogix 1100 oe 1400 device.

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

     – ***<optSlotNumber>*** is the number of the slot in the backplane in which the CPU module is configured. If you omit this parameter, the driver will use the Slot number ***0***

     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.

For ControlLogix, FlexLogix and CompactLogix

- **Header** field for ControlLogix, FlexLogix and CompactLogix devices — Specify the base data type for the worksheet, using the following syntax:

  <Data Type>

Example — **`INT; DINT`**

Where:

– ***`<Data Type>`*** is the data type of the device register. Use one of the following: **`BOOL`**, **`SINT`**, **`INT DINT`**, **`LINT`**, **`REAL`** or **`STRING`**.

✍ **Note:**

For successful communication with tags on standard driver sheet:

For Symbolic and physical mode : The **Header** field should be ***`<Logix Datatype>`***

The **I/O Address** field should be ***`<Logix Datatype>`***:***`<Logix Tag Name>`***

In Symbolic mode: The **Header** field should be ***`<Logix Datatype>`***

The **I/O Address** field should be ***`<Logix Tag Name>`***

In Physical mode: The **Header** field should be left blank.

The **I/O Address** field should be ***`<Logix Datatype>`***:***`<Logix Tag Name>`***

For MicroLogix 1100/1400

- **Header** field — Specify the type and address of the initial register in the block.

  For **Output (O)** and **Input (I)**:

  `<File Type>`:*`0.`****`<Address Reference>`***

  Examples — **`O:0.0`** or **`I:0.4`**/5

✍ **Notes:**

- Even though the Micrologix PLCs belong to the 500 Series family, the address is not *Slot-based*, as it is on the SLC500, but *0-offset* based. This is why when configuring the Header, you should always configure the number **0** (zero) after the "**:**"

For **Status (S)**, **Binary (B)**, **Integer (N)**, **Timer (T)**, **Counter (C)**, **Control (R)**, **Float (F)**, **Long (L)** and **String (ST)**:

`<File Type><File Number>`:***`<Address Reference>`***

Examples — **`N7:0`** or **`ST15:0`**

Where:

- *<File Type>* is the type of register. Use one of the following: O, I, S, B, N, T, C, R, F, L or ST.
- ***<Slot Number>*** is the number of the PLC slot in which the Output (O) or Input (I) is located.
- ***<File Number>*** is the number of the register group in which the register is configured.
- ***<Address Reference>*** is the starting address of the block of registers covered by this worksheet. This value is combined with ***<Address Offset>*** below to get the exact address.

After you enter the **Header** parameter, Studio checks that the syntax is valid. If the syntax is invalid, then Studio automatically inserts a default value (`DINT`) into the **Header** field.

Alternatively, you can specify an indirect tag (**`{Tag})`** in the **Header** field, but you must be certain that the tag's value is correct and uses the correct syntax or you will get an invalid header error.

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

For ControlLogix, FlexLogix and CompactLogix

- **Address** field — Specify the exact Logix tag using the following syntax:

  For **BOOL**, **SINT**, **INT**, **DINT, LINT** and **REAL** types:

  <Logix Tag Name> /[Bit]

  Examples — **PLCTAG; LEVEL[10]**

  For **STRING** type:

  <Logix Tag Name>

  Examples — **STRING:TEXT**

  For **LINT** type:

  For support for 64 bits integers, configure LINT Tag to be a string datatype instead of an integer datatype.

For **PROGRAM** tags outside of the Controller Tags database (**BOOL**, **SINT**, **INT**, **DINT , LINT** and **REAL** types):

PROGRAM:*<Program Name>.<Logix Tag Name> /[Bit]*

Example — **PROGRAM:MAINPROGRAM.UDT1.M1[0]**

For **PROGRAM** tags outside of the Controller Tag database (**STRING** type):

PROGRAM:*<Program Name>.<Logix Tag Name>*

Example — **PROGRAM:MAINPROGRAM.HMI_STRING**

Where:

- ***<Logix Tag Name>*** is the name (or address) of the tag in RSLogix, plus higher array dimensions (if necessary).
- ***<Program Name>*** is the name of the program outside of the tag database.

✍ **Notes:**

- You must not configure a range of addresses greater than the maximum block size (*data buffer length*) supported by each device within the same *Driver* worksheet. The maximum Block Size to be sent is 544 bytes and the maximum Block Size to be received is 492 bytes, following the protocol. However, there are some many variables that influence the number of the bytes such as data type, array size, tag names length and total amount of the tags in one worksheet. In other words, the maximum elements, it can have in the sheets, depend of the all variables mentioned above.
- A characteristic of the protocol’s message structuring within ABCIP is the ability to greatly diminish the message size of data requests for items defined as array elements in the PLC device. When all of the tags on a standard driver sheet are associated with contiguous elements from an array in the PLC, the Read request for this group does not contain individual item references. It simply contains the array name, and the highest and lowest array position, plus a small number of bytes of overhead. We recommend that the user takes advantage of this structure by utilizing arrays wherever possible in the PLC for those tags which will be configured on a standard driver sheet in the application. This will significantly increase the number of tags that can be configured per sheet, and by extension, reduce the total number of driver sheets configured in the application. Using array tags, depending on the tag name’s length, usually, it is possible to have: INT (240 elements), DINT (120 elements), LINT (120 elements), SINT (480 elements) and REAL (120 elements) using Standard Driver Sheet.

Thus, we highly recommend using arrays or aliasing tags to arrays, using RSLogix, and communicating with these arrays in order to optimize the communications.

For MicroLogix 1100/1400

- **Address** field — Specify the exact register address using the following syntax:

  For **Output (O)**, **Input (I)**, **Status (S)**, **Binary (B)** and **Integer (N)** files:

  [optFormat]<Address Offset>/[Bit]

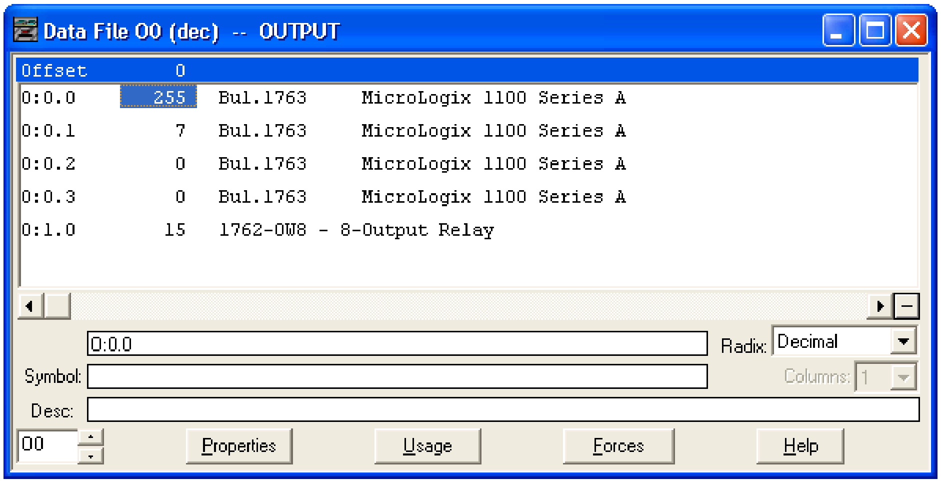
  Examples — **W2/4**, **2/4**, **15/2**, **8**

✍ **Notes:**

- Keep in mind that even though the Micrologix PLCs belong to the 500 Series family, the address is not *Slot-based*, as it is on the SLC500, but *0-offset* based.
- When you use additional I/O modules on the Micrologix 1100/1400 PLCs, in order to know exactly how to configure the ***word number***, you should check how the RSLogix500 sees these addresses.

  On the screenshot below, the CPU has only six digital Outputs. They all fit on the First word. However, the CPU reserves the next 3 words as well. The first External Output Module in then seen by the PLC as the word number **4** (5th word)

  So, for example, in order to read the bit **2** coming from an external 8-Output Relay module, the RSlogix500 shows it as **O:1.0/2.** On the ABCIP driver you must configure it as if it was **O:0.4/2**.



Data File O0 (dec) -- OUTPUT

Offset	0		
0:0.0	255	Bul.1763	MicroLogix 1100 Series A
0:0.1	7	Bul.1763	MicroLogix 1100 Series A
0:0.2	0	Bul.1763	MicroLogix 1100 Series A
0:0.3	0	Bul.1763	MicroLogix 1100 Series A
0:1.0	15	1762-OW8 - 8-Output Relay	

0:0.0 Radix: Decimal

Symbol: Columns: 1

Desc:

00

For **Timer (T)**, **Counter (C)** and **Control (R)** files:

Unsigned values – ***[Format]<Address>.<Element>***

Signed values – ***S[Format]<Address>.<Element>***

Examples — **T4:0.PRE**

For **String (ST)** file:

[Format]<Address Offset>.[Number of Bytes]

Examples — **S0.50** or **0**

For **Float (F)** file:

[Format]<Address Offset>

Examples — **F0** or **0**

Where:

– ***[Format]*** (optional) is how the data should be handled. Use one of the following: **W** for Word, **B** for BCD, **F** for Float or **S** for String.

– ***<Address Offset>*** is the value added to ***<Address Reference>*** above to get the exact address of the register.

– ***[Bit]*** (optional) is the bit number (from 0 to 15) of the register. Word format only.

- ***[Number of Bytes]*** (optional) is the maximum size of the ASCII/STRING data.
- ***<Element>*** is the element type for Timers (T), Counters (C) and Controls (R), according to the following table:

Register	Elements																	
	DN	PRE	ACC	EN	TT	UA	UN	OV	CD	CU	FD	IN	UL	ER	EM	EU	LEN	POS
Timer	R	R/W	R/W	R	R	–	–	–	–	–	–	–	–	–	–	–	–	–
Counter	R	R/W	R/W	–	–	R	R	R	R	R	–	–	–	–	–	–	–	–
Control	R	–	–	R	–	–	–	–	–	–	R	R	R	R	R	R	R/W	R/W

Attention:

You can use the Bit Writing function only with the **Write on tag change** driver tag enabled, which means that you cannot use the **Write trigger** tag for the Bit Writing function. The same rule applies to Timers, Counters and Controls.

Note:

If PLC5 is used the table above is not valid. It is possible to **Write on tag change** using any configurable sub-element.

For examples of how the **Header** and **Address** fields should be completed in order to address specific tags and device registers, please refer to the following tables:

For ControlLogix, FlexLogix and CompactLogix		
Logix Tag Name	**Header field**	**Address field**
RETURN_3_LOW_TEMP_ALARM	**BOOL**	**RETURN_3_LOW_TEMP_ALARM**
REC_ZONES[3]	**BOOL**	**REC_ZONES[3]**
MYUDT2[1,0].BOOL	**BOOL**	**MYUDT2[1,0].BOOL**
BOOLAR1[8] **(Program Tag)**	**BOOL**	**PROGRAM:MAINPROGRAM.BOOLAR1[8]**
HOT_RM_TEMP_DEGF	**INT**	**HOT_RM_TEMP_DEGF**

INT[4]	**INT**	**INT[4]**
INT_TEST.2	**INT**	**INT:INT_TEST/2**
MYUDT2[1,1].INT	**INT**	**MYUDT2[1,1].INT**
INT **(Program Tag)**	**INT**	**PROGRAM:MAINPROGRAM.INT**
Control_1	**SINT**	**CONTROL_1**
Control[1,2,0]	**SINT**	**CONTROL[1,2,0]**
Device.Parameter[2]	**SINT**	**DEVICE.PARAMETER[2]**
SINTAR1[4] **(Program Tag)**	**SINT**	**PROGRAM:MAINPROGRAM.SINTAR1[4]**
DINT1B[2,0]	**DINT**	**DINT1B[2,0]**
MYUDT2[1,0].MEMBER1	**DINT**	**MYUDT2[1,0].MEMBER1**
CONTROLLERUDTTAG[9].LINTAR[5]	**LINT**	**CONTROLLERUDTTAG[9].LINTAR[5]**
LINTAR2[1,5] **(Program Tag)**	**LINT**	**PROGRAM:MAINPROGRAM.LINTAR2[1,5]**
MYUDT2[1,0].MEMBER1.2	**DINT**	**MYUDT2[1,0].MEMBER1/2**
MIG380REAL	**REAL**	**MIG380REAL**
REALAR2[1,3]	**REAL**	**REALAR2[1,3]**
YUDT2[1,0].REAL	**REAL**	**YUDT2[1,0].REAL**
MYUDT[0,8].REAL **(Program Tag)**	**REAL**	**PROGRAM:MAINPROGRAM.MUUDT[0,8].REAL**
STRING1	**STRING**	**STRING1**
STTAG1[1,1,1]	**STRING**	**STTAG1[1,1,1]**
STRINGAR[1]	**STRING**	**MYUDT2[0,0].STRINGAR[1]**
STRINGAR3[1,0,4] **(Program Tag)**	**STRING**	**PROGRAM:MAINPROGRAM.STRINGAR3[1,0,4]**
HMI_STRING	**STRING**	**HMI_STRING**
HMIS[6,4,3]	**STRING**	**HMIS[6,4,3]**
MYUDT2[1,1].HMI_STRINGAR[2]	**STRING**	**MYUDT2[1,1].HMI_STRINGAR[2]**
HMI_STRING **(Program Tag)**	**STRING**	**PROGRAM:MAINPROGRAM.HMI_STRING**

For MicroLogix 1100/1400		
Register Address on the Device	**Header field**	**Address field**
I:0/7	`I:0.0`	`0/7`
I:0/10	`I:0.0`	`0/10`
I:0/17	`I:0.0`	`0/17`
I:0/25	`I:1.0`	`0/5`
I:3/4	`I:0.0`	`3/4`
I:3/4	`I:0.3`	`0/4`
O:0/7	`O:0.0`	`0/7`
O:0/10	`O:0.0`	`0/10`
O:0/17	`O:0.0`	`0/17`
O:0/25	`O:1.0`	`0/5`
O:3/4	`O:0.0`	`3/4`
O:3/4	`O:0.3`	`0/4`
S:0/5	`S:0`	`0/5`
S:10/7	`S:0`	`10/7`
S:10/7	`S:10`	`0/7`
B3:0/5	`B3:0`	`0/5`
B3:10/7	`B3:0`	`10/7`
B3:10/7	`B3:10`	`0/7`
N7:0	`N7:0`	`0`
N7:0/10	`N7:0`	`0/10`
N7:50	`N7:20`	`30`

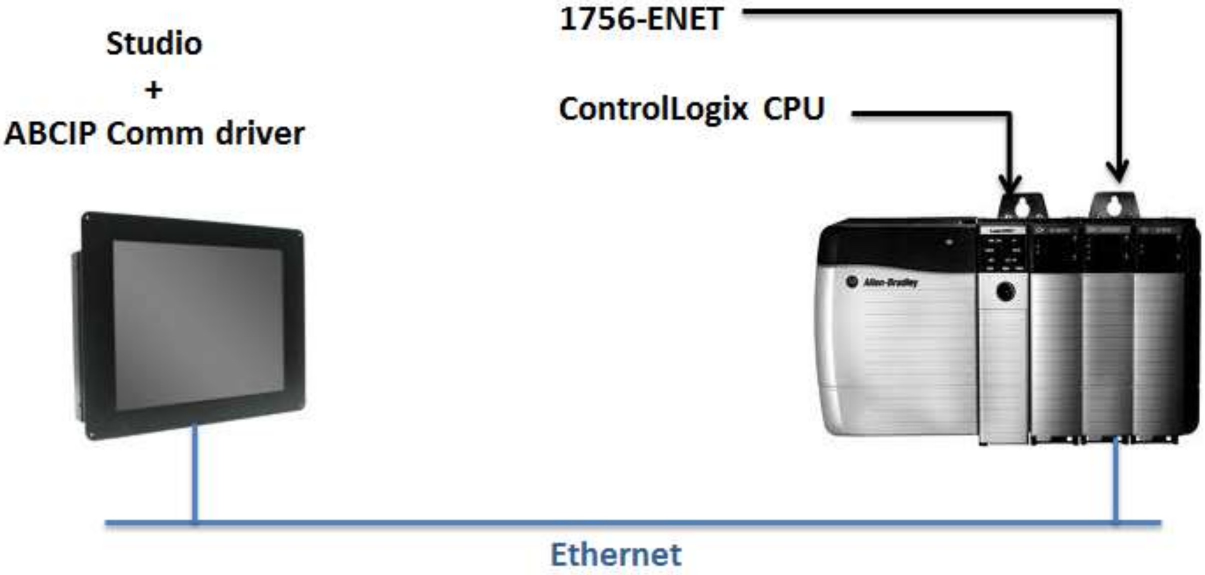
For MicroLogix 1100/1400

Register Address on the Device	Header field	Address field
T4:0.ACC	**T4:0**	0.**ACC**
T4:0.PRE	**T4:0**	0.**PRE**
T15:0.LEN	**T15:0**	0/**EN**
T15:0.ACC	**T15:0**	0.**ACC**
T15:1.ACC	**T15:0**	1.**ACC**
C5:0.ACC	C5:0	0.**ACC**
C5:1.PRE	C5:0	1.**PRE**
C20:15.UA	C20:10	5/**UA**
R6:0.LEN	**R6**:0	0.**LEN**
R6:0.POS	**R6**:0	0.**POS**
R6:1.POS	**R6**:0	1.**POS**
F8:0	**F8**:0	0
F8:5	**F8**:5	0
F8:5	**F8**:0	5
ST15:0 (String: maximum 20 bytes)	**ST15**:0	**S**0.20
ST15:1 (String: maximum 50 bytes)	**ST15**:0	**S**1.50
ST15:2 (String: maximum 10 bytes)	**ST15**:1	**S**1.10
L9:0	**L9**:0	0
L9:5	**L9**:5	5

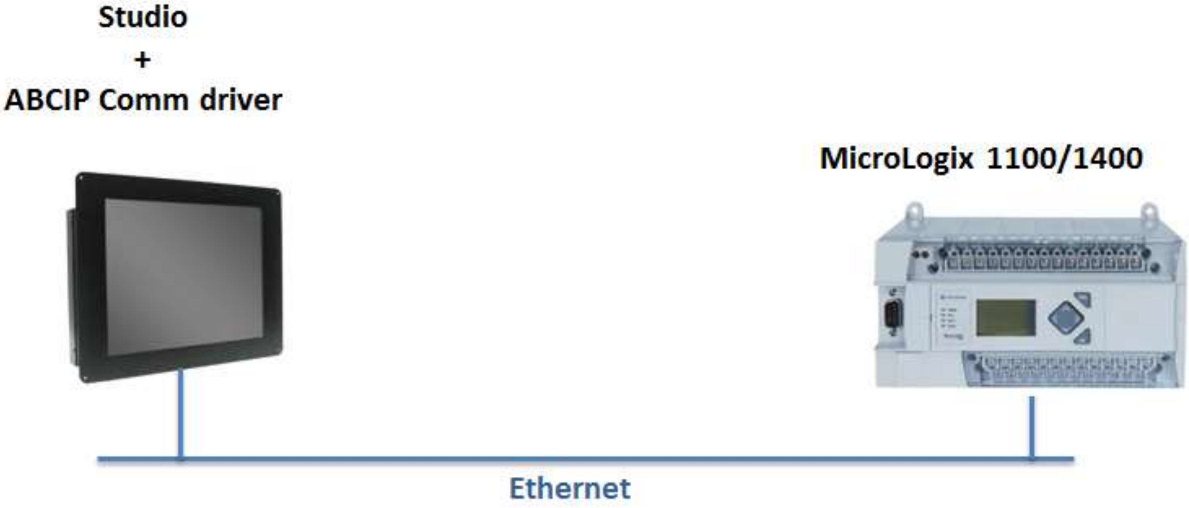
For MicroLogix 1100/1400		
Register Address on the Device	**Header field**	**Address field**
L9:10	`L9:10`	`10`

Routing Communication with Remote SLC500 and PLC5 Nodes

The ABCIP driver supports network communication on two different levels. In the typical configuration (as described earlier in this document), Studio communicates directly with ControlLogix 5000 and MicroLogix 1100 devices via Ethernet:



Communication with ControlLogix 55XX CPU

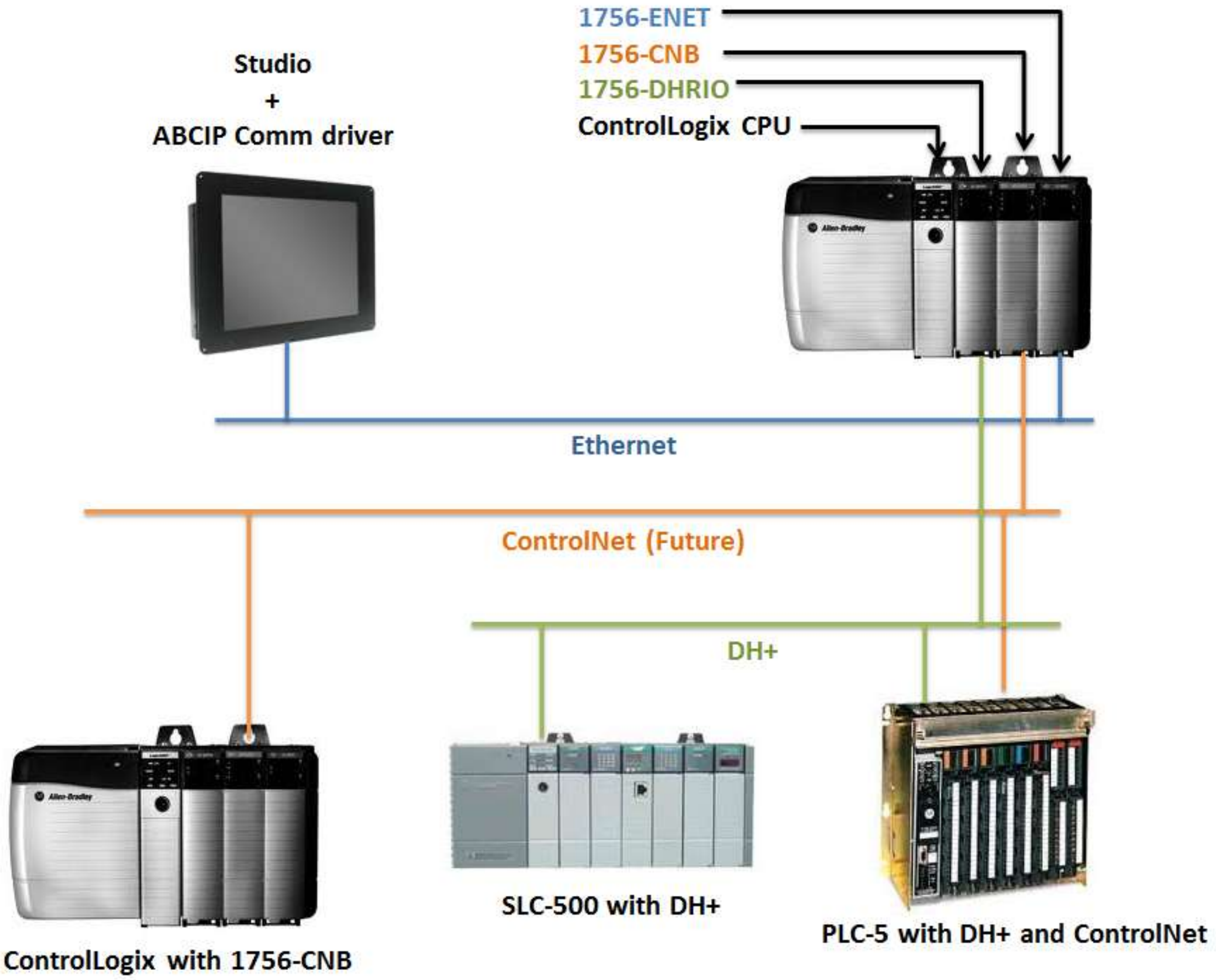


Communication with Micrologix 1100

However, the ABCIP driver also supports routed communication with remote nodes. In this configuration, the primary ControlLogix PLC also acts as a device network router provided that it has at least one of the following modules installed:

- **1756-DHRIO –** Communication interface for Data Highway Plus (DH+) or Remote I/O (RIO); or
- **1756-CNB –** Communication interface for ControlNet.

The following illustration shows how such a configuration would be set up:



Communication with Remote Nodes Using DH+ or ControlNet

Configuring the Station Field

To address these remote nodes, you must configure the **Station** field on the worksheet (for Main Driver Sheet, see page 11; for Standard Worksheet, see page 20) using the following syntax:

<Family>*:*<IP Address>*:*<Backplane>*:*<Slot>*:*<Channel>*:*<Remote Node>

Where:

- ***<Family>*** is the model family of the remote node. Use one of the following:
  - **Blank** – ControlLogix 5000
  - **2** or **500** – SLC500
  - **3** or **5** – PLC5
  - **4 or 1100 or 1400 or 1500 –** Micrologix 1100/1400/1500
- ***<IP Address>*** is the IP address of the of the ControlLogix PLC (or more specifically, its 1756-ENET module) that is acting as the device network router.
- ***<Backplane>*** is always 1.
- ***<Slot>*** is the number of the ControlLogix PLC slot where the 1756-DHRIO module is installed.
- ***<Channel>*** is the DH+ channel (**A** or **B**) to which the remote node is connected.
- ***<Remote Node>*** is DH+ address of the remote node (in decimal).

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.

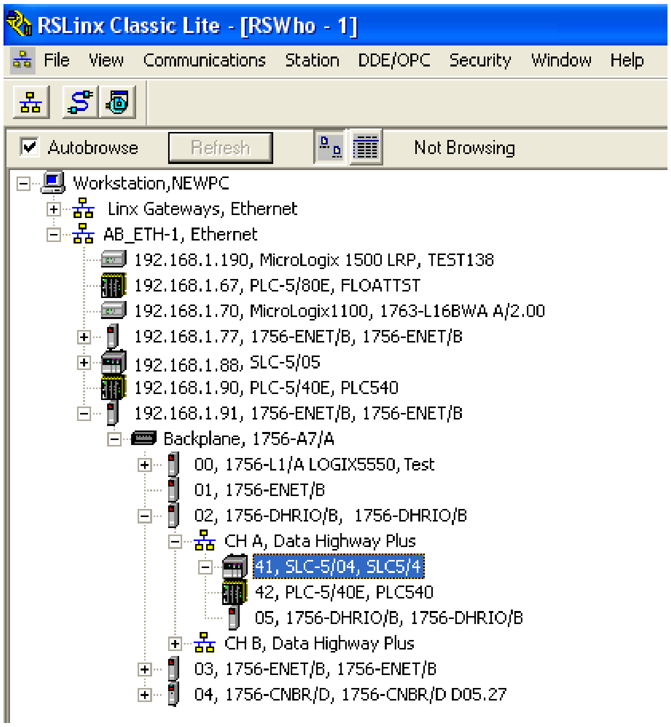
➲ **Attention:**
You must use a non-zero value in the **Station** field, and you cannot leave the field blank.

For instance, in the picture at right, we will access the highlighted PLC5/40 NSC_SYS1, DH+ address 41. The Station Address will be:

```
500:192.168.1.91:1:2:A:33
```

Where:

- **500** is SLC500 family;
- **192.168.1.91** is the IP address of the routing ControlLogix PLC;
- **1** is the backplane;
- **2** is the slot where the 1756-DHRIO/B module is installed in the routing ControlLogix PLC;



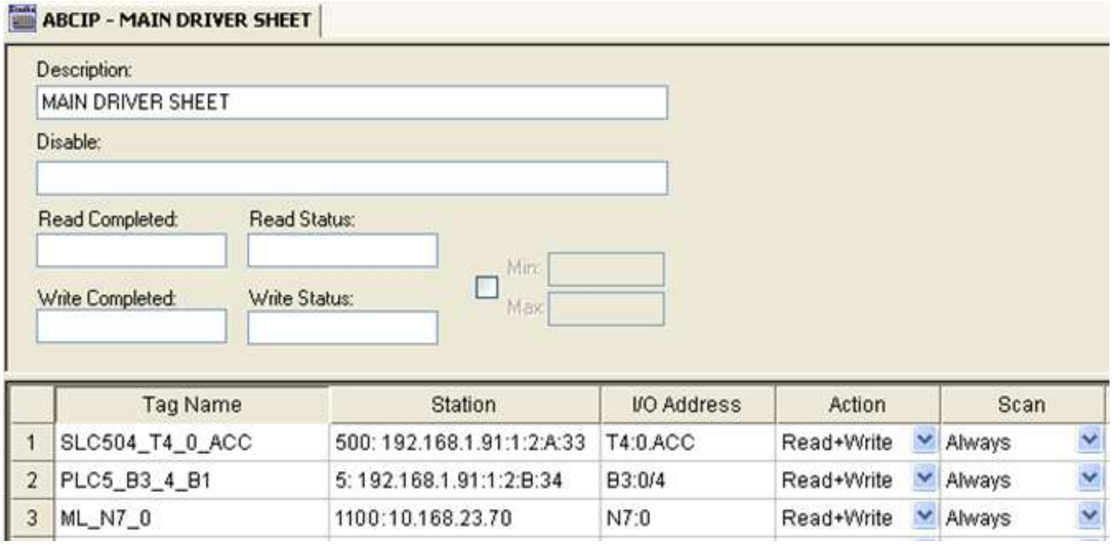
Workstation,NEWPC
Linx Gateways, Ethernet
AB_ETH-1, Ethernet
192.168.1.190, MicroLogix 1500 LRP, TEST138
192.168.1.67, PLC-5/80E, FLOATTST
192.168.1.70, MicroLogix1100, 1763-L16BWA A/2.00
192.168.1.77, 1756-ENET/B, 1756-ENET/B
192.168.1.88, SLC-5/05
192.168.1.90, PLC-5/40E, PLC540
192.168.1.91, 1756-ENET/B, 1756-ENET/B
Backplane, 1756-A7/A
00, 1756-L1/A LOGIX5550, Test
01, 1756-ENET/B
02, 1756-DHRIO/B, 1756-DHRIO/B
CH A, Data Highway Plus
41, SLC-5/04, SLC5/4
42, PLC-5/40E, PLC540
05, 1756-DHRIO/B, 1756-DHRIO/B
CH B, Data Highway Plus
03, 1756-ENET/B, 1756-ENET/B
04, 1756-CNBR/D, 1756-CNBR/D D05.27

- `A` is the DH+ channel; and
- `33` is the DH+ address (41 octal converted to decimal) for the SLC500.

Configuring the I/O Address Field

To address registers on SLC500 and PLC5 remote nodes, you can use basically the same syntax as the MicroLogix 1100 described earlier in this document. However, please refer to the documentation for the ABKE and ABTCP drivers to verify your configurations.

For an example of a finished ABCIP driver worksheet, see the following screenshot:



Description: MAIN DRIVER SHEET

Disable:

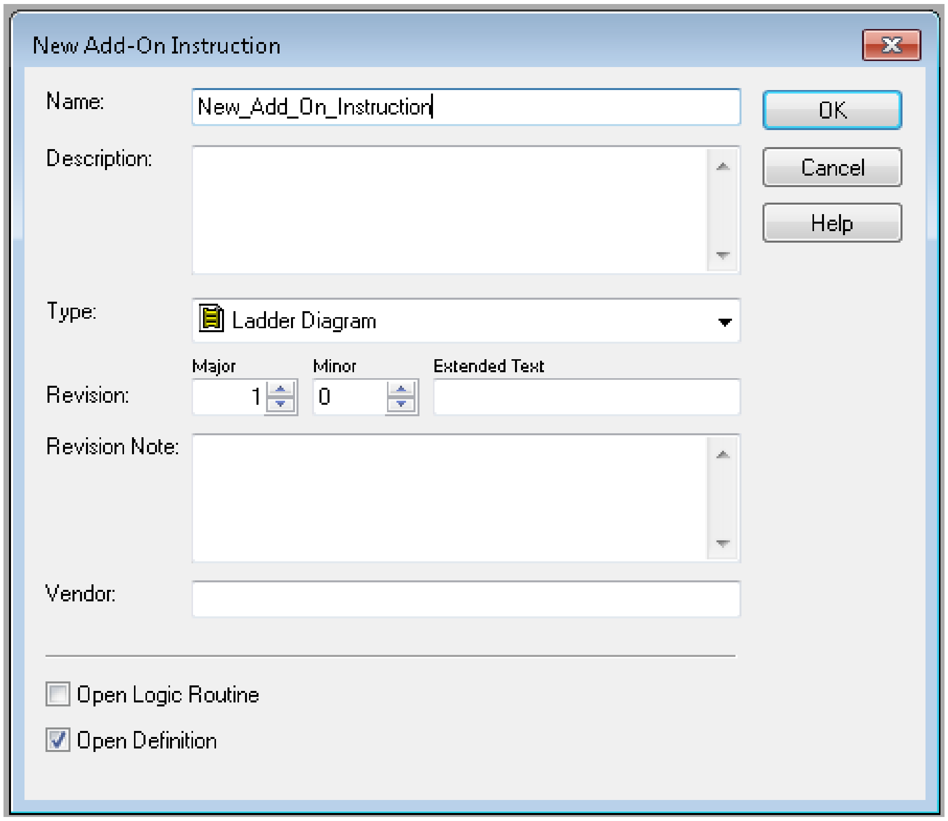
	Tag Name	Station	I/O Address	Action	Scan
1	SLC504_T4_0_ACC	500: 192.168.1.91:1:2:A:33	T4:0.ACC	Read+Write	Always
2	PLC5_B3_4_B1	5: 192.168.1.91:1:2:B:34	B3:0/4	Read+Write	Always
3	ML_N7_0	1100:10.168.23.70	N7:0	Read+Write	Always

ABCIP Main Driver Worksheet Showing Communication with Remote Nodes

In this example, there is a ControlLogix 5000 PLC at IP address 192.168.1.91 acting as the device network router. It has a 1756-DHRIO module installed in slot 2. A SLC500 is connected to the module's DH+ channel A, and a PLC5 is connected to the module's DH+ channel B. Studio is also communicating directly with a MicroLogix 1100 at IP address 10.168.23.70.

Notes - Add On Instructions – None Access Attribute - ControlLogix

Starting with ControlLogix firmware version 18.x, using the Rockwell RSLogix 5000 Programming Software, UDT tags and their elements can be configured with an External Access property setting of *Read/Write*, *ReadOnly* or *None*. The *None* setting is specifically meant to define a private tag within the processor, which is not exposed to components outside of the controller, such as the ABCIP driver. This affects Add On Instructions behavior, which make extensive use of UDTs. For these reasons, UDTs with elements having the External Access property set to None is not supported.



Name: New_Add_On_Instruction

Description:

Type: Ladder Diagram

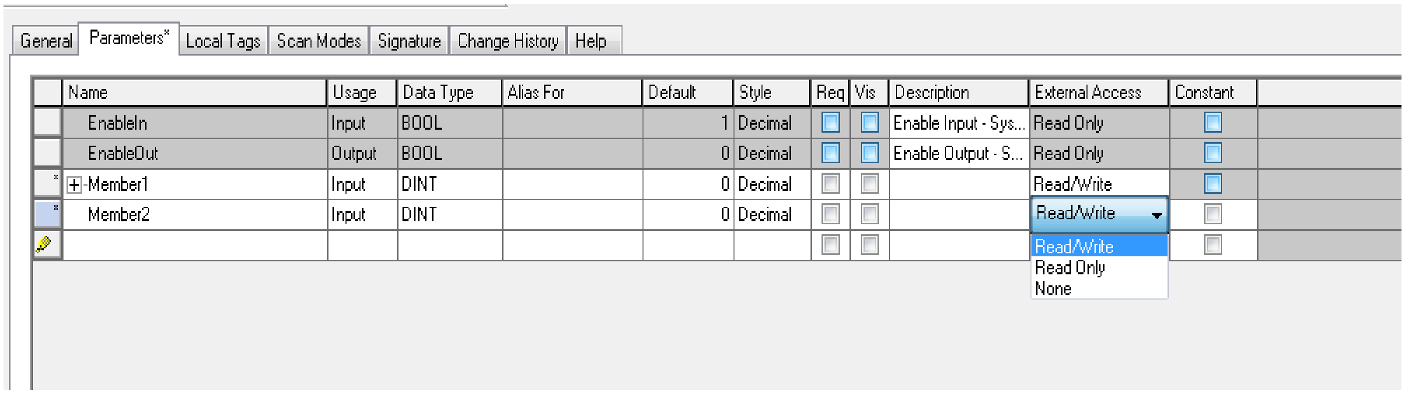
Revision: Major 1 Minor 0 Extended Text

Revision Note:

Vendor:

☐ Open Logic Routine

☑ Open Definition



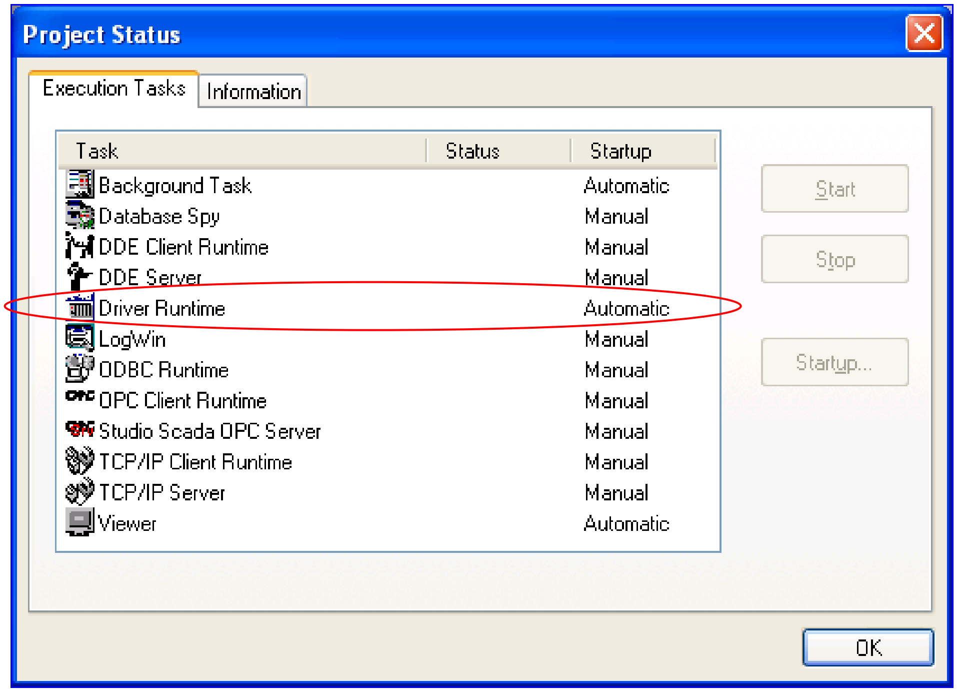
Name	Usage	Data Type	Alias For	Default	Style	Req	Vis	Description	External Access	Constant
EnableIn	Input	BOOL		1	Decimal	☐	☐	Enable Input - Sys...	Read Only	☐
EnableOut	Output	BOOL		0	Decimal	☐	☐	Enable Output - S...	Read Only	☐
+ Member1	Input	DINT		0	Decimal	☐	☐		Read/Write	☐
Member2	Input	DINT		0	Decimal	☐	☐		Read/Write	☐
						☐	☐			☐

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 the communication driver(s) will execute correctly:

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



Task	Status	Startup
Background Task		Automatic
Database Spy		Manual
DDE Client Runtime		Manual
DDE Server		Manual
Driver Runtime		Automatic
LogWin		Manual
ODBC Runtime		Manual
OPC Client Runtime		Manual
Studio Scada OPC Server		Manual
TCP/IP Client Runtime		Manual
TCP/IP Server		Manual
Viewer		Automatic

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 ABCIP 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 Main 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
3	Invalid Command	▪ Trying to write BOOL Data Type using the "Write Trigger" driver worksheet tag ▪ Wrong Data Type in the Driver Worksheet Header field	▪ The Tags of Data Type BOOL can only be written via "Enable Write on tag change" or Main Driver Sheet. ▪ Type a valid header (INT, SINT, DINT, REAL, BOOL, STRING).
7	Invalid Block Size	▪ More items than allowed in a Standard Driver Worksheet ▪ If a struct is being configured without specifying the member name block size error can occur.	▪ Split your driver worksheet into two or more. ▪ Check the struct Tags make sure all of them has the member name specified.
8	Invalid Write Command	▪ The PLC didn't recognize the tag that the application is trying to write.	▪ Check the Tag Data Type in your driver worksheet. ▪ Check the Tag Name in your driver worksheet. ▪ Check the Address Syntax in your driver worksheet.
9	Error Answer Block Size	▪ The answer length exceeded the supported limit.	▪ Split your driver worksheet into two or more.
10	Not Allocated Memory	▪ The driver is trying to remove memory that was not previously allocated.	▪ This is a driver internal error. If this error persists, please contact technical support.
11	Invalid Read Command	▪ The PLC didn't recognize the tag that the application is trying to read.	▪ Use the Output Window (LogWin), enabling the *Protocol Analyzer* option to see which PLC Tag that the driver is trying to communicate with is considered invalid ▪ Check the Tag Data Type in your driver worksheet. ▪ Check the Tag Name in your driver worksheet. ▪ Check the Address Syntax in your driver worksheet.
22	Invalid Data Type	▪ The data type specified in the Driver Worksheet Address Field is not a valid one.	▪ Type one of the valid Types (INT, DINT, SINT, REAL, BOOL, STRING).
23	Error in send_RR_data Function	▪ The driver is not getting the logical connection to the PLC.	▪ This is a driver internal error. If this error persists, please contact technical support.
24	Invalid IP	▪ The IP address is not valid.	▪ Check the valid IP address and Check the valid **station** field configuration.
25	Invalid Back Plane	▪ The Back Plane was not configured	▪ Check the valid **station** field configuration and check the Back Plane in your driver worksheet.
26	Invalid Slot	▪ The Slot was not configured	▪ Check the valid **station** field configuration and check the Slot in your driver worksheet.
27	Invalid Channel	▪ The Channel was not configured	▪ Check the valid **station** field configuration and check the Channel in your driver worksheet.
28	Invalid Remote Node Address	▪ The Remote Node Address was not configured	▪ Check the valid **station** field configuration and check the Remote Node Address in your driver

Error Code	Description	Possible Causes	Procedure to Solve
			worksheet.
31	Invalid Address	▪ The configured address is not valid.	▪ Check the valid **Address** field configuration.
32	Invalid Command	▪ The PLC does not support this command.	▪ Configure a valid command.
33	Blank Station	▪ The Station was not configured	▪ Configure the Station
35	Error Micrologix Request	▪ Request Error	▪ Check the valid IP address and Check the valid **station** field configuration. If this error persists, please contact technical support.
38	Invalid Sub-Element	▪ Invalid Timer, Counter or Control Element.	▪ Check the valid **Address** field configuration for Timer, Counter or Control.
39	Invalid Writing Sub-Element	▪ Impossible writing using Element	▪ Check Elements can be used to write.
40	Invalid BCD	▪ Invalid BCD Number	▪ Insert a valid BCD Number
41	Invalid Format	▪ Incompatible Format	▪ Check the valid configuration in the register type tables.
42	Connection Error	▪ There is a problem with the connection.	▪ Try to connect again. If this error persists, please contact technical support.
43	Invalid Octet	▪ Invalid Octet Number.	▪ Insert a valid octet Number.
44	Invalid Message ID	▪ The waited response is not the received one.	▪ Please, contact technical support.
45	Communication Problem	▪ An invalid tag was requested	▪ Check the tag names requested
46	Invalid Tag Name	▪ One of the requested tags does not exist or is not accessible in the PLC	▪ Check the tag names requested
501	Program Not Found	▪ There was a problem uploading the Program information	▪ Reset the cache file. See the RESETCACHE address. ▪ Contact your Studio technical support
502	Unknown Datatype	▪ The data type could not be read from the PLC and is not a primitive known type	▪ The driver will try to auto-recover by recreating the symbols cache. ▪ Reset the cache file. See the RESETCACHE address. ▪ Contact your Studio technical support
503	Unknown Member	▪ The member of the UDT is not known	▪ Reset the cache file. See the RESETCACHE address.
504	Unknown Tag	▪ The variable name in the address field is not known.	▪ Reset the cache file. See the RESETCACHE address. ▪ Enable the Protocol Analyzer log and look for messages containing the tag name not found in symbols cache.
505	Invalid Read Complete	▪ The read process has not completed or completed incorrectly	▪ Check the timeout configuration
506	Waiting For Another Station	▪ The driver is waiting for another instance to complete the upload of the program variables from the PLC.	▪ Wait for a few minutes ▪ Enable Protocol Analyzer log. There should be messages being transmitted and received.

Error Code	Description	Possible Causes	Procedure to Solve
		▪ This is an expected error that should happen only when Simultaneous Connections is enabled, and that should stop happening after a few minutes.	▪ If there are no messages being transmitted and received and the error persists, contact Studio technical support
507	Unexpected error	▪ Some unexpected situation happened	▪ Please, contact technical support.
508	Cache Synchronization Error	▪ There was an error reading the PLC Tag Address defined in the Cache Action parameter in driver settings	▪ Verify that the tag exists in the PLC and has the appropriate type. See the configurations section. ▪ Check the timeout configuration ▪ Check for network settings
509	Program Changed	▪ The program has changed in the PLC and the PLC Tag Address defined in the Cache Action has changed value. The values read from the PLC may not be correct and must be discarded.	▪ This is an expected error code when the program changes. After this, the cache will be recreated.
510	Invalid Cache Action settings	▪ The cache action defined in the driver settings is invalid	▪ Review the cache action settings
511	Invalid Memory Offset	▪ There was a problem while creating the symbols cache.	▪ The driver will try to auto-recover by recreating the symbols cache. ▪ Reset the cache file. See the RESETCACHE address. ▪ If the problem continues, please contact technical support
1001	Malformed Enip Header	▪ The received message was incorrectly formed	▪ Check the communication configurations
1002	Malformed Enip Data	▪ The received message was incorrectly formed	▪ Check the communication configurations
1003	Device Error	▪ The device returned an error code. The error code is shown in the log if Protocol Analyzer is enabled	▪ Check the device configuration
1004	Wrong Sequence Number	▪ The received message had a wrong sequence number	▪ Check the communication configurations, specially the timeout
1100	Invalid Session Handle	▪ The device returned this error indicating that the session is no longer valid.	▪ The driver will attempt to restore the session in the next communication
2001	Connection Failure	▪ The device returned this error indicating that the connection is no longer valid	▪ The driver will attempt to restore the connection in the next communication
2004	Path Segment Error	▪ The device returned this error indicating that the memory path used to read a variable is not valid	▪ Reset the cache file. See the RESETCACHE address.
2005	Invalid Path	▪ The device returned this error indicating that the memory path used to read a variable is not valid	▪ Reset the cache file. See the RESETCACHE address.
2601	Unrecognized Service Code	▪ The service code returned from the device is not recognized by the driver	▪ Check the device configuration ▪ Contact Studio technical support
2602	Malformed Cip Header	▪ The received message was incorrectly formed	▪ Check the communication configurations
2603	Malformed Cip Data	▪ The received message was incorrectly formed	▪ Check the communication configurations

Error Code	Description	Possible Causes	Procedure to Solve
2604	Unexpected Response	▪ The message returned was in a malformed state or in wrong order.	▪ Check the communication configurations ▪ Check the device configuration ▪ Contact Studio technical support
2605	String too big	▪ The length of the string is more than 81 or more than the length specified on the address.	▪ Change the size of the string on the address or reduce the size of the string value.
1004	Timeout	▪ IP Address may be wrong or The SLOT configuration in STATION field may be incorrect.	▪ Try to ping the IP Address. ▪ If it responds, fix the Station Field
1005	Timeout	▪ IP Address may be wrong or The SLOT configuration in STATION field may be incorrect.	▪ Try to ping the IP Address. ▪ If it responds, fix the Station Field
-15	Timeout Start Message	▪ Disconnected cables ▪ PLC is turned off, in stop mode, or in error mode ▪ Wrong station number	▪ Check cable wiring. ▪ Check the PLC state – it must be RUN. ▪ Check the station address. Try to "ping" your PLC.
-17	Timeout Between rx char	▪ PLC in stop mode or in error mode ▪ Wrong station number	▪ Check the PLC state – it must be RUN. ▪ Check the station address.

⇨ **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.

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 (e.g., RSLogix5000). 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:

- **System and Project Information**: 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	1.01	Eric Vigiani	03 Oct 2003	▪ Initial version
B	1.02	Eric Vigiani	28 Jun 2004	▪ Modified the driver to accept multiple connections ▪ Modified driver must support communication with remote PLCs (SLC, PLC5, ControlLogix) via AB-1756 gateway ▪ Fixed bug to read boolean data properly
C	1.03	Fabio H.Y Komura	19 Jul 2004	▪ Implemented read/write of bits
D	1.04	Fabio H.Y Komura	29 Oct 2004	▪ Fixed bug with STRING header ▪ Fixed bug to get Header information properly
E	1.05	Leandro Coeli	27 Jan 2005	▪ Fixed problems in MDS ▪ Implemented Multi-Array
F	1.06	Leandro Coeli	27 Mar 2005	▪ Fixed problems in Station field
G	1.07	Leandro Coeli	12 Sep 2005	▪ Fixed problems in Station field
H	1.08	Leandro Coeli	26 Jan 2006	▪ Modified driver to accept the Program Operand
I	1.09	Diego Barros	13 Apr 2006	▪ Modified driver to communicate with MicroLogix1100
J	1.10	Diego Barros	05 May 2006	▪ Fixed some problems with input/output registers.
K	1.11	Eric Vigiani	05 Sep 2006	▪ Fixed problems with MicroLogix 1100 ▪ Modified String type to accept UDP and PDT.
L	1.12	Graziane C. Forti	07 Dec 2006	▪ Implemented Message Error "invalid writing sub-elements" (C5, T4 and R6 Registers) ▪ Fixed problem with String Reading ▪ Fixed problem with conversions about Data Types ▪ Fixed problem with address configuration
M	1.12	Rafael R. Fernandes	09 Feb 2007	▪ Fixed problem with BCD reading. ▪ Fixed problem with String reading and writing. ▪ Fixed problem with Data Types on MDS.
N	1.12	Graziane C. Forti	08 May 2007	▪ Fixed reading/writing of String odd length (Micrologix) ▪ Inserted some error messages. ▪ Fixed problem reading/writing BOOL array (Contrologix) ▪ Fixed problem writing DINT (Contrologix)
O	1.12	Michael D. Hayden	05 Jun 2007	▪ Edited for language and usability
P	1.13	Graziane C. Forti	14 Jun 2007	▪ Implemented message with TagName not found
Q	1.14	Graziane C. Forti	11 Feb 2008	For Micrologix

				▪ Fixed connection problem with ARMV4i ▪ Implemented “CON” sub-type Timer, Control and Counter. ▪ Modified to work with ControlLogix at the same time. ▪ Fixed problem writing STATUS operand. ▪ Checked status writing an inexistent operand. ▪ Fixed problem reading Last String address. ▪ Fixed problem using “Address Reference” with Standard Driver WorkSheet. For Controllogix ▪ Fixed problem to create the Main Driver WorkSheet group. ▪ Inserted “Invalid Message ID”. ▪ Modified to work with MicroLogix at the same time. For PLC5 ▪ Implemented write using Timer, Control and Counter sub-elements. ▪ Implemented BCD Read/Write. ▪ Fixed problem "F" Format Float type. ▪ Implemented Octet address I/O. ▪ Implemented Signed (Timer, Control and Counter) ▪ Fixed problem writing T4 sub-elements For SLC500 ▪ Fixed problem reading I/O. ▪ Implemented Signed (Timer, Control and Counter) ▪ Fixed problem writing T4 sub-elements
R	1.14	Plínio M. Santana	Apr 01 2008	Fixed problem to use two tags with the same address in the MDS.
S	1.15	Eric Vigiani	Jun 01 2008	Modified to work properly with PLC5 through ControlLogix
T	1.15	Eric Vigiani	Sep 18 2008	▪ Fixed Problems of Block Size in the MDS ▪ Fixed limitations in some PLC file address when communicating with PLC5
U	10.01	Eric Vigiani	Dec 15 2008	▪ Modified for SLC and PLC5 show error code when receiving error message from PLC. ▪ Modified for ControlLogix shows the correct PLC Tag Name in the LogWin.
V	10.1	Marcelo Carvalho	Jan 07 2009	▪ Updated driver version, no changes in the contents.
W	10.3	Fellipe Peternella	Mar 25 2009	▪ Modified to properly handle PLC Tag Names with names longer than 45 characters
X	10.4	Lourenço Teodoro Vicente Teodoro	Jul 02 2009	▪ Fixed memory allocation problem caused on version 10.3. The driver was allocating 6MB more of RAM, which was a huge problem for Windows CE applications with low amount of RAM. ▪ Modified the driver to support connection with Backpane routing and Control Logix at the same time.

Z	10.5	Lourenço Teodoro	Jul 02 2009	▪ Improved memory allocation ▪ Fixed issues with Block Size error
AA	10.5	Andre Bastos	Mar 31 2010	▪ Changed the documentation only. No modifications in the driver
AB	10.6	André Körbes Fellipe Peternella Paulo Balbino	Jan 31 2011	▪ Fixed problems with reading and writing to SLC500 and PLC5 ▪ Fixed block size of timers of PLC5 ▪ Fixed memory usage to avoid block size errors
AC	10.6	Andre Bastos	Aug 18 2011	▪ Documentation revision only, related to I/O addressing for MicroLogix 1100/1400
AD	10.6	Lucas Caccavaro	Oct 17 2011	▪ Modified screenshots on the Appendix
AE	10.7	André Körbes	May 15, 2012	▪ The driver no longer closes a connection when a reading error happens ▪ Fixed virtual group issues related to *Screen* scan type ▪ Fixed problems of reading array members of structs, boundary elements for SLC500 and invalid elements ▪ Fixed virtual group issues when using a tag in curly brackets in the Station field for Main Driver Sheet ▪ Added TCP Port Number configuration in the settings dialog
AF	11.0	André Körbes	Apr. 1, 2013	▪ Added the new ControlLogix mode for reading variables using the memory address. ▪ Updated error codes section
AG	11.1	Caio Cerquetani	Oct. 1, 2013	▪ Added support for L register ▪ Fixed routing to SLC5 that was sending the write command with an extra zeroed byte ▪ Fixed the reading of S register on ML 1100/1400
AH	11.2	André Körbes	Mar. 5, 2014	▪ Fixed issue when communicating with multiple PLCs routed by the same ControlLogix.
AI	11.3	Paulo Balbino	July. 29, 2014	▪ Fixed issue when writing to Bool arrays ▪ Fixed issue when reading Strings in SLC
AJ	11.4	Eduardo Castro	Dec. 22, 2014	▪ Added support for communication with ControlLogix/CompactLogix CPUs with firmware 21 and 24, in physical mode
AK	11.5	Paulo Balbino	Apr. 7, 2015	▪ Fixed issue with firmware 20
AL	11.5	Andre Bastos	Jun 30, 2015	▪ Modified the documentation only, to better explain the Slot number on the ControlLogix PLCs. No changes on the driver
AM	11.6	Anushree Phanse	Aug 07, 2015	▪ Solved problem with reading of bits in Firmware21 ▪ Solved the problem when processing the UDTs ▪ Improved performance of MDS Virtual groups split when handling STRINGs for Firmware >= v21
AN	11.7	Paulo Balbino	Oct,30, 2015	▪ Increased String sizes up to 380 bytes.
AO	11.8	Anushree Phanse	Mar,01,2016	▪ Solved a problem of inability to stop runtime when the PLC is offline. ▪ Fixed support for Boolean Logix tags on the PLC which are of the type Alias. ▪ Improved the output log message when a standard driver sheet has an

				invalid block size. ▪ Added output log messages when using physical mode to show progress of symbol download. ▪ Implemented support for Logix tags of datatype LINT. ▪ Fixed the write operation for strings on SLC5/04 via DH+ routing. ▪ Implemented support for predefined structure of type MESSAGE. ▪ Fixed Read for some Logix tags of datatype DINT which are part of predefined structures.
AP	11.9	Paulo Balbino Anushree Phanse	Sept, 06,2016	▪ Increased range of LINT datatype and fixed writing issues. ▪ Fixed communication for program tags when firmware is equal or greater than 21. ▪ Added support for communication with controllers of the type L83 and L85 with firmware 28 ▪ Changed documentation to warn the user about possible block size issues when configuring complex tags (structs) without specifying the members on symbolic mode. ▪ Fixed DINT array issue in firmware 20 for some controllers. ▪ Added support for new predefined structures on firmware 28. ▪ Fixed issue of writing the maximum length of a string in physical mode which didn't work as expected.
AQ	11.10	Paulo Balbino Anushree Phanse	July, 24, 2017	▪ Fixed issue when reading F9 registers on SLC500 ▪ Fixed issue with reading from 2 different PLCs not working in Symbolic mode ▪ Fixed issue with Physical mode showing no error messages when the items in the standard driver sheet exceeded size limitations. ▪ Fixed issue of ABCIP not supporting Program tags when using firmware 28 and higher. ▪ Fixed issues when communicating with physical mode with firmware 30.12 ▪ Fixed issue with reading wrong values when using the N15 register in PLC5 ▪ Fixed issue with reading and writing to strings when using PLC5 ▪ Updated documentation with information on using Add On Instruction type tags ▪ Fixed issue with no support for using registers/files over 255 in PLC5 ▪ Fixed issue with correctly reading BOOL datatypes for predefined structs and add-on instructions ▪ Updated documentation to include additional method of using Header and Address fields on Standard driver sheets
AR	11.11	Anushree Phanse	Oct,20,2017	▪ Fixed driver documentation with the correct information about reading and writing to individual bits for addresses on ControlLogix, FlexLogix and CompactLogix. ▪ Fixed the error reading strings correctly when it's value is assigned using the Concat block in the PLC program on ControLogix ▪ Fixed the issue where the driver doesn't communicate to the right tags when the first octet of the IP address is a single digit
AS	11.12	Anushree Phanse	Dec,06,2017	▪ Ported driver to be platform agnostic and communicate with Controllogix devices and made a crash bug fix.

AT	11.13	Anushree Phanse	June,14,2018	▪ Fixed issue with block creation that showed invalid block size errors. ▪ Fixed issue with some tags communicating with a Status 2030 when using Physical mode.
AU	11.14	Anushree Phanse	Nov, 28, 2018	▪ Fixed issue with bits reading incorrectly when using physical mode in firmware 30.11 ▪ Fixed issue with group splitting for Micrologix when using type Timers. ▪ Fixed some internal security issues. ▪ Changed LINT datatype to support reading and writing of 64 bit values.