ABTCP – *Driver Version 10.9*
*Doc. Revision AC– May 05, 2016*

# ABTCP Communication Driver

Driver for Ethernet Communication with
Allen-Bradley PLCs using DF1 protocol

## Contents

# Introduction

The ABTCP driver enables communication between the Studio system and Allen-Bradley devices using the DF1 protocol, according to the specifications discussed in this document.

This document will help you to select, configure and execute the ABTCP 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 ABTCP driver in the Studio system.
- **Configuring the Device**: Describes how the target device must be configured to receive communication from the ABTCP driver.
- **Configuring the Driver**: Explains how to configure the ABTCP driver in the Studio system, including how to associate database tags with device registers.
- **Executing the Driver**: Explains how to execute the ABTCP driver during application runtime.
- **Troubleshooting**: Lists the most common errors for this driver, their probable causes, and basic procedures to resolve them.
- **Sample Application**: Explains how to use a sample application to test the ABTCP driver configuration
- **Revision History**: Provides a log of all changes made to the driver and this documentation.

**Notes:**

1. This document assumes that you have read the “Development Environment” chapter in Studio’s *Technical Reference Manual*.
2. This document also assumes that you are familiar with the Microsoft Windows NT/2000/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 Ethernet communication between the ABTCP driver in Studio and an Allen Bradley PLC using DF1 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
- **Compatible Equipment:** PLC2 series, PLC5 series and SLC500
- **Programmer Software:** RSLogix5, RSLogix500, RSLinx

For a description of the device(s) used to test driver conformance, see "Conformance Testing" on the next page.

## ***Network Specifications***

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

- **Device Communication Port:** Ethernet Port
- **Physical Protocol:** Ethernet TCP/IP
- **Logic Protocol:** DF1
- **Device Runtime Software:** None
- **Specific PC Board:** Any TCP/IP adapter (Ethernet card)

## ***Driver Characteristics***

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

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

**Note:**
You must use Adobe Acrobat® Reader™ to view the **`ABTCP.PDF`** document. You can install Acrobat Reader from the Studio installation CD, or you can download it from Adobe's Web site.

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

- Windows NT/2000/XP/Vista
- Windows CE

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

The ABTCP driver supports the following register types:

| Register Type | Length (in Bytes) | Default Format | Write | Read | Bit | Integer | Float | String | BCD |
|---|---|---|---|---|---|---|---|---|---|
| O (Output) | 2 | Word | ● | ● | ● | ● | – | – | ● |
| I (Input) | 2 | Word | – | ● | ● | ● | – | – | ● |
| S (Status) | 2 | Word | ● | ● | ● | ● | – | – | – |
| B (Binary) | 2 | Word | ● | ● | ● | ● | ● | – | ● |
| T (Timer) | 6 | Word | ● | ● | – | ● | – | – | – |
| C (Counter) | 6 | Word | ● | ● | – | ● | – | – | – |
| R (Control) | 6 | Word | ● | ● | – | ● | – | – | – |
| PD (PID) | 100 | Word | ● | ● | – | – | ● | – | – |
| F (Float) | 4 | Float | ● | ● | – | – | ● | – | – |
| N (Integer File) | 2 | Word | ● | ● | ● | ● | ● | – | ● |
| A (ASCII File) | 2 | String | ● | ● | – | ● | ● | ● | – |
| ST (String File) | *n* | String | ● | ● | – | – | – | ● | – |

➲ **Attention:**

- BCD format, only the first 12 bits of the register are transcribed to the associated tag. The last 4 bits are transcribed to the tag’s **Quality** property. For more information about tag properties, please refer to the *Technical Reference Manual*.
- Float format uses 4 bytes (2 Words). When using the Float format with a register type that has the default size in 2 bytes (1 Word). Then, it you use two consecutives addresses. This happens with Binary, Integer and ASCII registers.

## Conformance Testing

The following hardware/software was used for conformance testing:

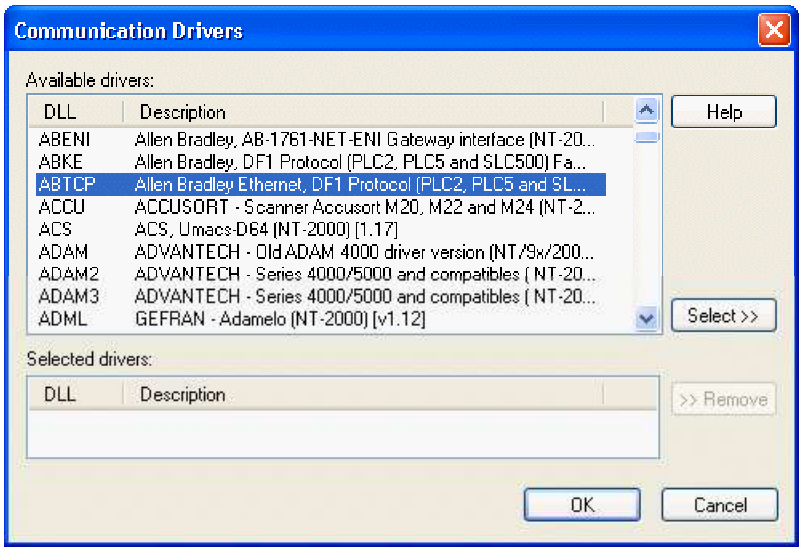
- **Equipment**: SLC5/05, PLC5/80
- **Driver Configuration**:
  - **PLCs**: SLC5/05 and PLC5/40 , PLC5 as SoftPLC
  - **Protocol**: DF1 over TCP/IP
- **Cable:** Ethernet Cable
- **Operating System** (development): Windows 8 x64
- **Operating System** (target): Windows 8 x64 and Windows CE v5.0 ARMV4i
- **Studio Version**: 8.0 + P3
- **Driver Version**: 10.9

## 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 ABTCP driver for your Studio application:

1. From the main menu bar, select **Insert** → **Driver** to open the *Communication Drivers* dialog.

2. Select the **ABTCP** driver from the *Available Drivers* list, and then click the **Select** button.



| DLL | Description |
|---|---|
| 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... |
| ADML | GEFRAN - Adamelo (NT-2000) [v1.12] |

*Communication Drivers Dialog*

3. When the **ABTCP** 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 an Allen-Bradley or Rockwell device, you must also install the Rockwell programming software (e.g., RSLogix). 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

Use the Rockwell configuration software to configure the device's Ethernet TCP/IP settings.

## Configuring the Driver

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

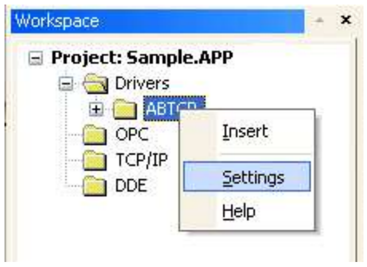
### ***Configuring the Communication Settings***

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

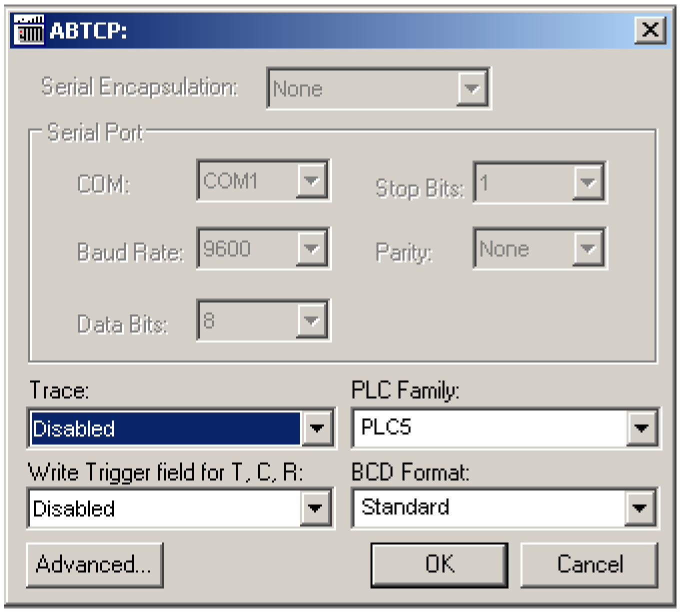
For the purposes of this document, only ABTCP driver-specific settings and procedures will be discussed here. To configure the communication settings for the ABTCP driver:

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

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



**Select Settings from the Pop-Up Menu**


ABTCP:

Serial Encapsulation: None

Serial Port

COM: COM1

Stop Bits: 1

Baud Rate: 9600

Parity: None

Data Bits: 8

Trace: Disabled

PLC Family: PLC5

Write Trigger field for T, C, R: Disabled

BCD Format: Standard

**ABTCP: Communication Parameters 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:

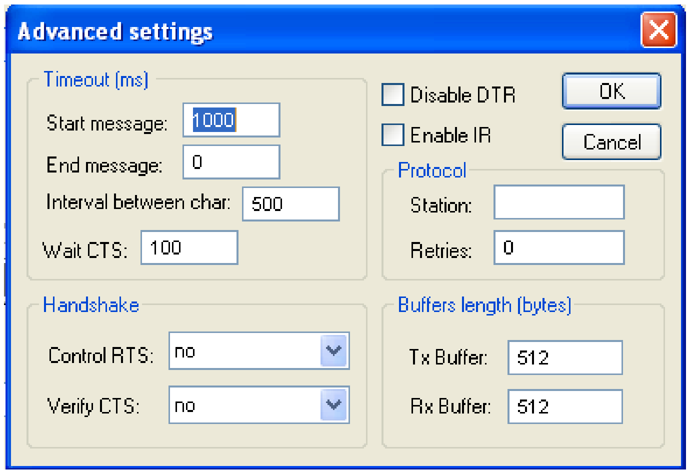
| Parameter | Default Value | Valid Values | Description |
|---|---|---|---|
| **Trace** | `Disabled` | `Disabled` or `Enabled` | When the trace is enabled, the *LogWin* module will display more detailed information about the communication. If you are generating a log file for technical support, then we recommend enabling this option. |
| **PLC Family** | `SLC500` | `PLC2`, `PLC5`, `PLC5 with I/O Octal`, `PLC5 as SoftPLC` or `SLC500` | Select the PLC Family that connects to the driver.<br><br>**PLC 2:** Driver uses "unprotected Read/Write" command (in the DF1 protocol) to communicate with the PLC2 family or compatible.<br><br>**PLC 5:** Driver uses "typed Read/Write" command (in the DF1 protocol) to communicate with the PLC5 family or compatible. The address for all data types is decimal.<br><br>**PLC 5 with I/O Octal:** Driver uses "typed Read/Write" command (in the DF1 protocol) to communicate with the PLC5 family or compatible. The address for I and O data types is octal. The address for all remaining data types is decimal.<br><br>**PLC 5 as SoftPLC:** Driver uses "typed Read/Write" command (in the DF1 protocol) to communicate with SoftPLC emulating the PLC5 family.<br><br>**SLC 500:** Driver uses "protected typed logical Read/Write" command (in the DF1 protocol) to communicate with the SLC500 family or compatible. |
| **Write Trigger field for T, C, R, PD** | `Disabled` | `Disabled` or `Enabled` | When this option is enabled, the driver is able to write using the Write Trigger field in the Standard Driver Worksheet. This option has effect only for Timers, Counter, Controls and PIDs. For all remaining data types the Write Trigger field is always enabled. |
| **BCD Format** | `Legacy` | `Legacy or Standard` | **Legacy:** Support for previous versions, where the 2 least significant digits are stored in the tag value and the most significant digit is stored in the tag **Quality** field. This parameter is used to keep compatibility with old 16-bit versions of the product.<br>**Standard:** Stores all the BCD digits in the tag value. |

| PLC Family field - Legacy Compatible | |
|---|---|
| **New** | **Old** |
| **PLC2** | `2` |
| **PLC5** | `5 or 5:0` |
| **PLC5 with I/O Octal** | `5:1` |
| **PLC5 as SoftPLC** | `5x` |
| **SLC500** | `500` |

**Attention:**
- When using **Write Trigger field** for the Timers (**T**), Counters (**C**) and Controls(**R**) registers, the driver reads the whole worksheet before writing**,** and then if at the same time the addresses that are not configured in the associated worksheet change the value, they may be overwritten.

**Note:**
To communicate with PLC 5 family, we strongly recommend selecting the option **`PLC5 with I/O Octal`** instead of **`PLC5`** in the Family field. The option **`with I/O Octal`** allows access to the I/Os in Octal, matching the PLC addressing mode, as well as direct access to the **ASCII (A)** file type by its word address.

4. In the *Communication Settings* dialog, click the **Advanced** button to open the *Advanced Settings* dialog:



Start message: 1000
End message: 0
Interval between char: 500
Wait CTS: 100
Disable DTR: ☐
Enable IR: ☐
Station:
Retries: 0
Control RTS: no
Verify CTS: no
Tx Buffer: 512
Rx Buffer: 512

***Advanced Settings Dialog***

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

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

## ***Configuring the Driver Worksheets***

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.

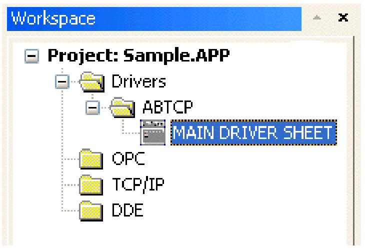
The configuration of these worksheets is described in detail in the “Communication” chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

For the purposes of this document, only ABTCP driver-specific parameters and procedures are discussed here.

## MAIN DRIVER SHEET

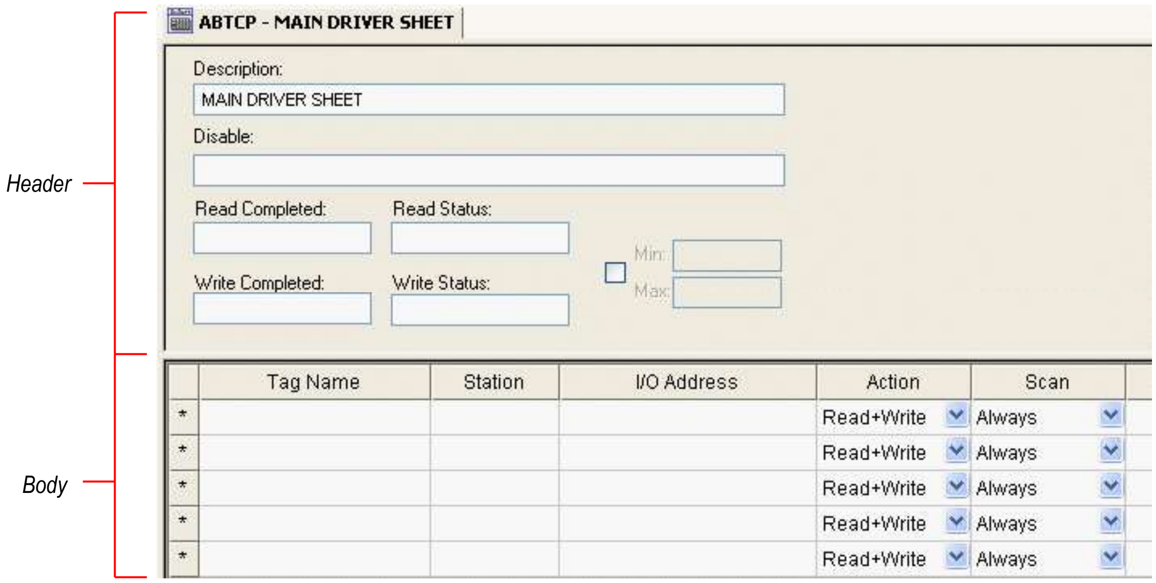
When you select the ABTCP driver and add it to your application, Studio automatically inserts the *Main Driver Sheet* in the *ABTCP* driver subfolder. To configure the Main Driver Sheet:

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



***Main Driver Sheet in the ABTCP Subfolder***

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



Description: MAIN DRIVER SHEET
Disable:
Read Completed:
Read Status:
Write Completed:
Write Status:

| | Tag Name | Station | I/O Address | Action | Scan |
|---|---|---|---|---|---|
| * | | | | Read+Write | Always |
| * | | | | Read+Write | Always |
| * | | | | Read+Write | Always |
| * | | | | Read+Write | Always |
| * | | | | Read+Write | Always |

***Opening the Main Driver Sheet***

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

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

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

     ***<IP Address>*:*[optional Port Number]*:*[optional PLC Family]***

     Examples — **`192.168.2.9`**

     **`192.168.125.31:2222`**

     **`192.168.125.31:2222:PLC5`**

     Where:

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

     – ***[Port Number]*** is the port number for the DF1 protocol (usually **2222**). This parameter is optional; you need to specify it only if the device cannot auto detect incoming DF1 communication.

     – ***[optional PLC Family]*** is the PLC Family Type (PLC2/PLC5/PLC5S/ SLC500). This parameter is optional; if it is not present the PLC family assumed is the one configured in the driver settings***.***

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

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

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

     For Inputs and Outputs, use the following syntax:

     ***<Type>*:*<Slot Number>*.*[Data Format]<Octet Number>*/*[Bit]***

     or

     ***<Type>*:*<Slot Number>*.*[Data Format]<Octet Number>*.*[Bit]***

     Example — **`O:1.W2/4`** or **`O:1.W2.4`**

> **Note:**
> If you are communicating with a PLC 5 family, usually you do not need to specify the Slot number. Simply type **0** on it. Example: for the PLC Address **O001/3**, type **O:0.W1/3**

For ASCII, Status, Binary and Integer, use the following syntax:

***<Type><Type Group>*:*[Data Format]<Address>*/*[Bit]***

or

***<Type><Type Group>*:*[Data Format]<Address>*.*[Bit]***

Example — **`N7:W150/2`** or **`N7:W150.2`**

For Timers, Counters, Controls and PIDs, use the following syntax:

***<Type><Type Group>*:*[Data Format]<Address>*.*<Element>***

Example — **`T4:W0.DN`**

For Float, use the following syntax:

***<Type><Type Group>:[F]<Address>.<Number of Bytes>***

Example — **F8:3 or F8:F3**

For String, use the following syntax:

***<Type><Type Group>:[S]<Address>.<Number of Bytes>***

Example — **ST15:0.50 or ST15:S0.50**

Where:

– ***<Type>*** : Device register type. Valid values are **O** (Output), **I** (Input), **S** (Status), **B** (Binary), **N** (Integer), **T** (Timer), **C** (Counter), **R** (Control), **PD** (PID), **F** (Float), **A** (ASCII), and **ST** (String).

– ***<Type Group>*** : Group number of the specified register type.

– ***<Slot Number>*** : I/O slot number on the device.

– ***[Data Format]*** : Format of the data being read or written, which determines how Studio will handle the data. Valid values are **W** (Word), **B** (BCD), **F** (Floating Point), and **S** (String). This parameter is *optional*; it can be left out of the address if the default format for the register type (as described on page 4) is acceptable.

> ➲ **Attention:**
> When using the BCD format, only the first 12 bits of the register are transcribed to the associated tag. The last 4 bits are transcribed to the tag's **Quality** property. For more information about tag properties, please refer to the *Technical Reference Manual*.
>
> Float format uses 4 bytes (2 words). When using the Float format with a register type that has the default size in 2 bytes. Then, it will use two consecutives addresses. This happens with Output, Input, Status, Binary and Integer.

– ***<Octet Number>*** : Number (in octal) of the desired Input or Output.

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

– ***[Bit]*** (optional): The bit number (from 0 to 15) of the address.

– ***<Number of Bytes>*** : Maximum size (in bytes) of the ASCII or String.

– ***<Element>*** : Element type for Timer, Counter, Control or PID, 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 | ● | ● | ● | ● | ● | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Counter | ● | ● | ● | – | – | ● | ● | ● | ● | ● | – | – | – | – | – | – | – | – |
| Control | ● | – | – | ● | – | – | – | – | – | – | ● | ● | ● | ● | ● | ● | ● | ● |
| PID | EN | CT | CL | PVT | DO | SWM | CA | MO | PE | INI | SPOR | OLL | OLH | EWD | | | | |
| | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | | | | |
| | DVNA | DVPA | PVLA | PVHA | SP | KP | KI | KD | BIAS | MAXS | MINS | DB | SO | MAXO | | | | |
| | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | | | | |
| | MINO | UPD | PV | ERR | OUT | PVH | PVL | DVP | DVN | PVDB | DVDB | MAXI | MINI | TIE | | | | |
| | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | | | | |

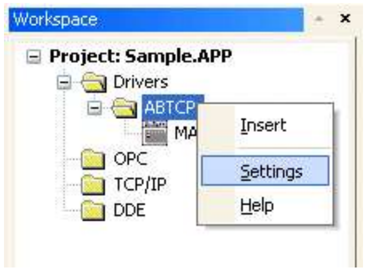
## STANDARD DRIVER WORKSHEET

When you select the ABTCP 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.

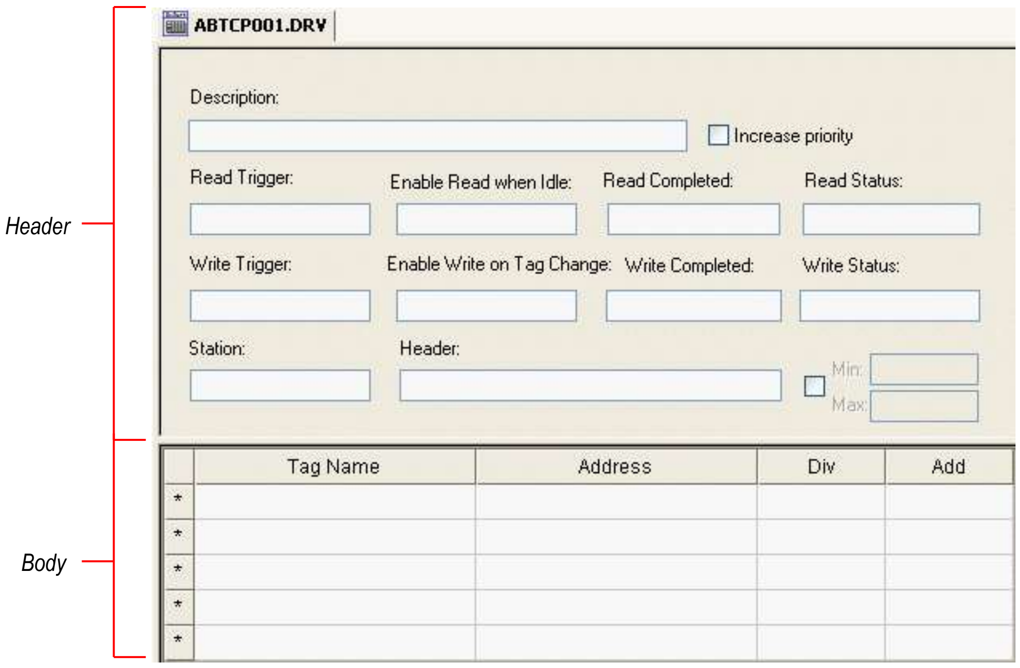
To insert a new Standard Driver Worksheet:

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



***Inserting a New Worksheet***

A new ABTCP driver worksheet is inserted into the *ABTCP* subfolder, and the worksheet is opened:



***ABTCP Driver Worksheet***

> **Note:**
> Worksheets are numbered in order of creation, so the first worksheet is **`ABTCP001.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 ABTCP driver.

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

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

     ***`<IP Address>`*****`:`*****`[optional Port Number]:[optional PLC Family]`***

     Examples — **`192.168.2.9`**

     **`192.168.125.31:2222`**

     **`192.168.125.31:2222:PLC5`**

     Where:

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

     – ***`[Port Number]`*** is the port number for the DF1 protocol (usually 2222). This parameter is optional; you need to specify it only if the device cannot autodetect incoming DF1 communication.

     – ***`[optional PLC Family]`*** is the PLC Family Type(PLC2/PLC5/PLC5S/ SLC500). This parameter is optional; if it is not present the PLC family assumed is the one configured in the driver settings*.*

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

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

   - **Header** field: 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.

     For Inputs and Outputs, use the following syntax:

     ***`<Type>`*****`:`*****`<Slot Number>`*****`.0`**

     Example — **`O:1.0`**

> **Note:**
> If you are communicating with a PLC 5 family, usually you do not need to specify the Slot number. Simply type **0** on it. Example: **O:0.0**

For Status, Binary, Integer, Timer, Counter, Control, PID, ASCII and String, use the following syntax:

***`<Type><Type Group>`*****`:`*****`<Address Reference>`***

Example — **`N7:0`** or **`ST15:0`**

For Unsolicited Message, use the following syntax:

**U:*<Message File>***

Example — **U:3**

Where:

– ***<Type>*** : Device register type. Valid values are **O** (Output), **I** (Input), **S** (Status), **B** (Binary), **N** (Integer), **T** (Timer), **C** (Counter), **R** (Control), **PD** (PID), **F** (Float), **A** (ASCII), and **ST** (String).

– ***<Type Group>*** : Group number of the specified register type.

– ***<Slot Number>*** : I/O slot number on the device.

– ***<Address Reference>*** : The initial address (reference) of the block of registers configured on this worksheet.

– ***<Message File>*** : The number of the unsolicited message file being generated by the device and then sent to Studio. These message files should already be programmed on the device before you try to access them using the Driver Worksheet.

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 **N7:0**.

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

| Information about the Header Parameter | | | |
|---|---|---|---|
| **Register Type** | **Example of Syntax** | **Valid Range of Initial Address** | **Comments** |
| Output | **O:0.0** | Varies according to the equipment | Physical outputs: Where "O" means output.<br>The first digit after the colon defines the word number if there is more then one digit in the same slot and the first digit following the dot is the output address. |
| Input | **I:0.0** | Varies according to the equipment | Physical inputs: Where "I" means input.<br>The first digit after the colon defines the word's number if there is more than one digit in the same slot and the digit following the dot is the output address. |
| Status | **S:0** | Varies according to the equipment | Reads the status words. |
| Binary | **B3:0** | Varies according to the equipment | Reads the Binary Operator. |
| Integer | **N7:0** | Varies according to the equipment | Reads and Writes the Integer addresses. |
| Timer | **T4:0** | Varies according to the equipment | Reads and Writes the Timer addresses. |
| Counter | **C5:0** | Varies according to the equipment | Reads and Writes the Counter addresses. |
| Control | **R6:0** | Varies according to the equipment | Reads and Writes the Control addresses. |
| PID | **PD:0** | Varies according to the equipment | Reads and Writes the PID addresses. |
| Float | **F8:0** | Varies according to the equipment | Reads and Writes the Float addresses. |
| ASCII | **A14:0** | Varies according to the equipment | Reads and Writes the ASCII addresses.<br>The "Address Reference" is defined in words. |
| String | **ST15:0** | Varies according to the equipment | Reads and Writes the String addresses. |

| | | | |
|---|---|---|---|
| Unsolicited Messages | `U:0` | 0 to 999 | If you program the device to send unsolicited messages to Studio, then type U: into the Header field followed by the name of the file being sent. **Important**: The Unsolicited Messages field type has been tested with the PLC2 only. |

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

| Register Type | Syntax | Accepted DataType | Examples |
|---|---|---|---|
| Output Input | ***[Data Format]<Octet Number>/[Bit]*** or ***[Data Format]<Octet Number>.[Bit]*** | W | W0/3 W0.3 |
| Status Binary Integer | ***[DataFormat]<AddressOffset>/[Bit]*** or ***[Data Format]<Address Offset>.[Bit]*** | B, W and F | W10/12 W10.12 |
| Timer Counters Controls PIDs | ***[Data Format]<Address Offset>/<Element>*** or ***[Data Format]<Address Offset>.<Element>*** | W | W2/PRE W2.PRE W2/OUT W2.OUT |
| Float | ***[F]<Address Offset>*** | F | F1.2 |
| ASCII | For Family 500 devices ***[Data Format]<Address Offset>.<Number of BYTES>*** For Family 5:1 devices ***[Data Format]<Address Offset>.<Number of WORDS>*** | S | S1.2 |
| String | For Family 500 devices ***S<Address Offset>.<Number of BYTES>*** For Family 5:1 devices ***S<Address Offset>.<Number of WORDS>*** | S | S1.2 |
| Unsolicited Message | ***[Data Format]<Address>*** | - | W2 |

Where:

– ***[Data Format]*** : Format of the data being read or written, which determines how Studio will handle the data. Valid values are **`W`** (Word), **`B`** (BCD), **`F`** (Floating Point), and **`S`** (String). This parameter is *optional*; it can be left out of the address if the default format for the register type (as described on page 4) is acceptable.

➲ **Attention:**
When using the BCD format, only the first 12 bits of the register are transcribed to the associated tag. The last 4 bits are transcribed to the tag's **Quality** property. For more information about tag properties, please refer to the *Technical Reference Manual*.

– ***<Octet Number>*** : Number (in octal) of the desired Input or Output.

– ***<Address Offset>*** : Value added to the ***<Address Reference>*** parameter (configured in the **Header** field above) to produce complete register address.

– ***[Bit]*** (optional): The bit number (from 0 to 15) of the address.

– ***<Number of Bytes>*** : Maximum size (in bytes) of the ASCII or String.

– ***<Element>*** : Element type for Timer, Counter, Control or PID, 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 | ● | ● | ● | ● | ● | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Counter | ● | ● | ● | – | – | ● | ● | ● | ● | ● | – | – | – | – | – | – | – | – |
| Control | ● | – | – | ● | – | – | – | – | – | – | ● | ● | ● | ● | ● | ● | ● | ● |
| PID | EN | CT | CL | PVT | DO | SWM | CA | MO | PE | INI | SPOR | OLL | OLH | EWD | | | | |
| | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | | | | |
| | DVNA | DVPA | PVLA | PVHA | SP | KP | KI | KD | BIAS | MAXS | MINS | DB | SO | MAXO | | | | |
| | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | | | | |
| | MINO | UPD | PV | ERR | OUT | PVH | PVL | DVP | DVN | PVDB | DVDB | MAXI | MINI | TIE | | | | |
| | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | | | | |

➲ **Attention:**

- You can use the Bit Writing function only when the **Write on tag change** field is enabled; you cannot use the **Write trigger** field for the Bit Writing function.
- When reading from and writing to ASCII registers, `\0` is used as the NULL character. Furthermore, if a string contains an odd number of characters, then a NULL character is added to the end.
- When addressing Input and Output registers on Family 5 devices, only the bit number is considered for determining the address on the device. The octet number is not considered. For example, `I:0.22`, `I:1.22` and `I:2.22` all correspond to the address `I:022` for that device family.
- When addressing Input and Output registers on Family 5:1 devices, the value in the **Address** field will be converted to an octal.

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

| Address on the Device | Header Field | Address Field |
|---|---|---|
| I:0/7 | `I:0.0` | `W0/7` or `0/7` |
| I:0/10 | `I:0.0` | `W0/10` or `0/10` |
| I:0/17 | `I:0.0` | `W0/17` or `0/17` |
| I:0/25 | `I:0.1` | `W0/9` or `0/9` |
| I:3/4 | `I:3.0` | `W0/4` or `0/4` |
| I:0/4 | `I:0.0` | `W0/4` or `0/4` |
| O:0/7 | `O:0.0` | `W0/7` or `0/7` |
| O:0/10 | `O:0.0` | `W0/10` or `0/10` |
| O:0/17 | `O:0.0` | `W0/17` or `0/17` |
| O:0/25 | `O:0.1` | `W0/9` or `0/9` |
| O:3/4 | `O:3.0` | `W0/4` or `0/4` |
| O:0/4 | `O:0.0` | `W0/4` or `0/4` |
| S:0/5 | `S:0` | `W0/5` or `0/5` |
| S:10/7 | `S:0` | `W10/7` or `10/7` |
| S:10/7 | `S:10` | `W0/7` or `0/7` |
| B3:0/5 | `B3:0` | `W0/5` or `0/5` |
| B3:10/7 | `B3:0` | `W10/7` or `10/7` |
| B3:10/7 | `B3:10` | `W0/7` or `0/7` |
| N7:0 | `N7:0` | `W0` or `0` |
| N7:0/10 | `N7:0` | `W0/10` or `0/10` |
| N7:50 | `N7:20` | `W30` or `30` |
| T4:0/ACC | `T4:0` | `W0/ACC` or `0/ACC` |
| T4:0/PRE | `T4:0` | `W0/PRE` or `0/PRE` |
| T15:0/EN | `T15:0` | `W0/EN` or `0/EN` |
| T15:0/ACC | `T15:0` | `W0/ACC` or `0/ACC` |
| T15:1/ACC | `T15:0` | `W1/ACC` or `1/ACC` |
| C5:0/ACC | `C5:0` | `W0/ACC` or `0/ACC` |
| C5:1/PRE | `C5:0` | `W1/PRE` or `1/PRE` |
| C20:15/UA | `C20:10` | `W5/UA` or `5/UA` |
| R6:0/LEN | `R6:0` | `W0/LEN` or `0/LEN` |
| R6:0/POS | `R6:0` | `W0/POS` or `0/POS` |

| Address on the Device | Header Field | Address Field |
|---|---|---|
| R6:1/POS | **R6:0** | **W1/POS** or **1/POS** |
| F8:0 | **F8:0** | **F0** or **0** |
| F8:5 | **F8:5** | **F0** or **0** |
| F8:5 | **F8:0** | **F5** or **5** |
| A14:0 (default size: 2 bytes) | **A14:0** | **S0.2** or **0.2** |
| A14:1 (default size: 2 bytes) | **A14:1** | **S0.2** or **0.2** |
| A14:1 (default size: 2 bytes) | **A14:0** | **S1.2** or **1.2** |
| A14:0 to A14:2 | **A14:0** | **S0.6** or **0.6** |
| ST15:0 (String: maximum 20 bytes) | **ST15:0** | **S0.20** or **0.20** |
| ST15:1 (String: maximum 50 bytes) | **ST15:0** | **S1.50** or **1.50** |
| ST15:2 (String: maximum 10 bytes) | **ST15:1** | **S1.10** or **1.10** |
| Unsolicited message N3:10 | **U:3** | **W10** |
| Unsolicited message N3:20 | **U:3** | **W20** |

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

➲ **Attention:**
You must not configure a range of addresses greater than the maximum block size (data buffer length) supported by each device within the same worksheet. The maximum data buffer length varies depending on the specific device being used:

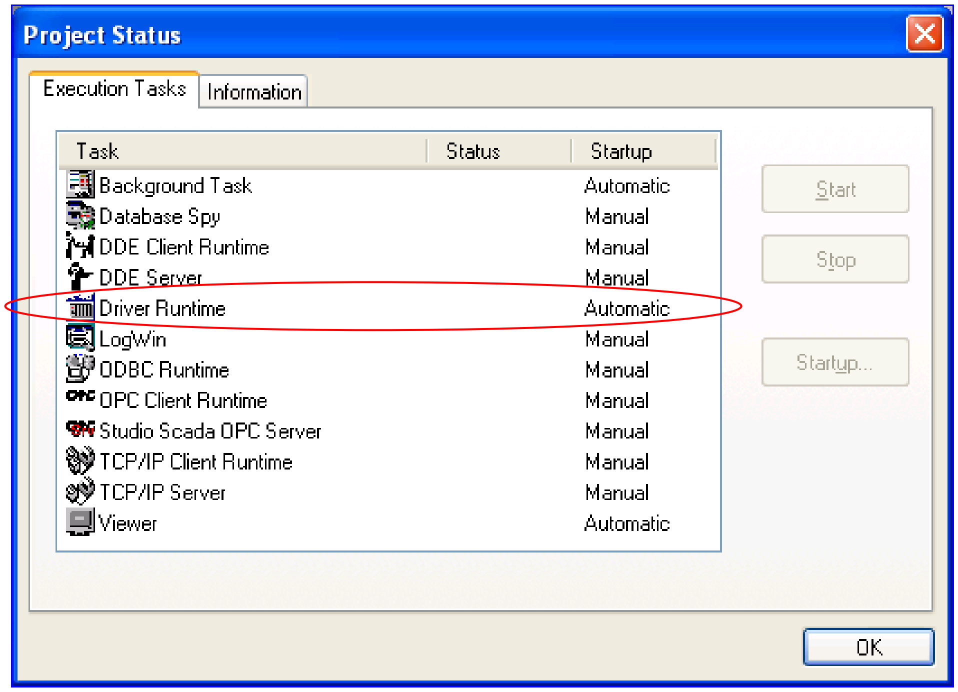
- For Family 2 and Family 5 devices, the maximum length is 244 bytes.
- For SLC/03 and SLC/04 devices, the maximum length is 236 bytes for Read commands and 234 bytes for Write commands.
- For SLC5/01 and SLC5/02 devices, the maximum length is 82 bytes.

## 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 ABTCP 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. |
| 6 | Address doesn't point to something usable | ▪ PLC5 does not have the address configure in the driver worksheet | ▪ Properly configure the Address to point to a valid PLC5 Address |
| 7 | File is wrong size, check if the address requested is in the range configured in the PLC | PLC5 Address File does not have the specified address | The item address needs to be available on the PLC. Modify either the Item address or the PLC File |
| 11 | Access denied, improper privilege, PLC is likely to being downloaded a new program | The PLC5 is receiving a new program | Wait until the new program is being downloaded to the PLC and then start communicating again |
| 17 | Illegal data type, check if the file that you are reading exists in the PLC and if the type matches. | Family 500: The file does not exist in the PLC or the file type does not match the type configured. | The item address needs to be available on the PLC. Modify either the Item address or the PLC File |
| 257 | Write or Read bit on BCD or Float Type is not allowed. | It is not possible to access bits of Float or integer addresses configured as BCD | Read the entire Float address<br>In case of Integer, properly configure the I/O address to read from the WORD format |
| 260 | PLC family is misconfigured | ▪ Wrong PLC family configured on the driver parameters | ▪ Properly configure the PLC family on the Driver Communication Settings |
| 261 | Protocol Error | SLC500: Trying to Read an Invalid Address from Valid File | Properly configure the Item Address to point to a valid SLC500 Address<br>You can use the Log file enabling the Protocol Analyzer and it will return the STS byte error code. Then, you can check the DF1 errors for the correct explanation of them |
| -34 | Invalid Address | Invalid Item address - the specified file type requires a suffix, such as T4:0.ACC | Properly configure the item Address including the suffix |
| -35 | Timeout | Invalid PLC IP Address | ▪ Check the IP Address in the Station fieldTry to Ping It, to make sure that it is reachable |
| -37 | Invalid Header | Invalid Header configured in the Standard Driver Sheet | Properly configure the Header with one of the valid ones |
| 65535 | Error Sending DF1 message. | The Driver was unable to send the protocol message due to an internal error | ▪ Please generate Log File and contact Tech Support |

⇨ **Tip:**

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

You can also use the *LogWin* module (**Tools** → **Remote LogWin**) to establish a connection with 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 (e.g., RSLogix). 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** (type and version): To find this information, select **Tools** → **System Information**.
- **Project Information**: To find this information, select **Project** → **Status.**
- **Driver Version** and **Communication Log**: Displays in the Studio *Output* window when the driver is running.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

## Sample Application

A sample application that employs the ABTCP driver is provided on the Studio installation CD. We strongly recommend that you use this sample application to test the driver *before* you develop your own applications, for the following reasons:

- To better understand the information and instructions provided in this document;
- To verify that your driver configuration is working satisfactorily with the target device; and
- To ensure that the all of hardware used in the test (i.e. the device, adapter, cable, and PC) is functioning safely and correctly.

**Note:**
The following instructions assume that you are familiar with developing project applications in Studio. If you are not, then please review the relevant chapters of the Studio *Technical Reference Manual* before proceeding.

To use the sample application:

1. Configure the device's communication settings according to the manufacturer's documentation.
2. Run Studio.
3. From the main menu bar, select **File** → **Open Project**.
4. Insert the Studio installation CD and browse it to find the sample application. It should be located in the directory **`\COMMUNICATION EXAMPLES\ABTCP`**.
5. Select and open the sample application.
6. Configure and test the driver, as described in the rest of this document.

When you have thoroughly tested the driver with your target device, you may proceed with developing your own Studio application projects.

**Tip:**
You can use the sample application screen as the maintenance screen for your own applications.

## Revision History

| Doc. Revision | Driver Version | Author | Date | Description of Changes |
|---|---|---|---|---|
| A | 1.00 | Roberto V. Junior | 21 Nov 2001 | First driver version |
| B | 1.01 | Roberto V. Junior | 21 Feb 2002 | Included "Typed Read/Write" commands to PLC5 family |
| C | 1.03 | Roberto V. Junior | 12 Apr 2002 | ▪ Modified to increase performance<br>▪ When 5 is configured in the Family field the "Typed Read/Write" commands is used with PLC5.<br>▪ When 5x is configured in the Family field the "Typed Read/Write" commands is used with SoftPLC working like a PLC5 |
| D | 1.05 | Roberto V. Junior | 09 May 2002 | ▪ Close device connection after an error communication.<br>▪ Included support to the UNITCPIP library. |
| E | 1.06 | Eric Vigiani | 18 Jun 2002 | Modified internal algorithm to accept initial addresses higher than 255. |
| F | 1.07 | Eric Vigiani | 03 Oct 2002 | ▪ Included TCP/IP Port set (optional) in the Station Field.<br>▪ Modified internal algorithm to avoid connection error (error code 90) |
| G | 1.08 | Lourenço Teodoro | 11 Aug 2003 | Fixed GPF when the address configured in the Main Driver Sheet is invalid. |
| H | 1.09 | Eric Vigiani | 14 Apr 2004 | Included the option to choose the address for data type as octal or decimal to PLC5. |
| I | 1.11 | Fabio H.Y.Komura | 15 Jul 2004 | ▪ Fixed problem with Octal (PLC5)<br>▪ Fixed problem when "Write with Header" offset is different from 0 (zero) - family 5<br>▪ Fixed problem with ARMV4 processor |
| J | 1.11 | Lourenço Teodoro | 23 Nov 2004 | Updated the communication parameters window |
| K | 1.12 | Eric Vigiani | 13 Apr 2005 | Fixed problem with reading the ST header using SoftPLC |
| L | 1.13 | Fabio H.Y.Komura | 15 Mar 2006 | ▪ Fixed problem with ASCII<br>▪ Fixed problem with BCD Values<br>▪ Fixed problem with octal addresses for I/O registers<br>▪ Changed PLC5 Address to Logical Binary Address<br>▪ Fix possible problem with ARMV4 processor |
| — | 1.13 | Michael D. Hayden | 16 Jun 2006 | Edited for language and usability. |
| M | 1.14 | Arthur S. Allievi | 15 Sep 2006 | ▪ Address syntax now accepts values without type.<br>▪ Address syntax now accepts values with either slash or dot for T, C and R headers.<br>▪ Address syntax now accepts values without type also on the Main Driver Sheet. |
| N | 1.15 | Plínio M. Santana | 08 Dec 2006 | ▪ Fixed problem about don't writing  and reading bit on BCD types.<br>▪ Fixed problem about don't writing and reading bit on FLOAT types.<br>▪ Fixed problem about write '\0' on String types.<br>▪ Fixed problem about don't writing and reading the NULL character ('\0').<br>▪ Fixed problem about reading incorrect values on Counter, Control and Timer types. |
| O | 1.15 | Plínio M. Santana | 09 Jan 2007 | ▪ Document corrections. |
| P | 1.15 | Michael D. Hayden | 29 Jan 2007 | ▪ Additional editing to merge previously forked documents. |
| Q | 1.16 | Graziane C. Forti / Eric Vigiani | 02 Aug 2007 | ▪ Fixed problem using ASCII (family 5)<br>▪ Getting connection back after commit error.<br>▪ Implemented Timer,  Control  and Counter Write Group. |
| R | 1.17 | Rafael R. Fernandes / Eric Vigiani | 08 Feb  2008 | ▪ Modified the Family PLC's names.<br>▪ Modified the driver to don't accept invalid bit (lower than 0 and higher than 15).<br>▪ Modified the Timer, Counter and Control to support bits. |

| Doc. Revision | Driver Version | Author | Date | Description of Changes |
|---|---|---|---|---|
| S | 10.01 | Eric Vigiani | 03 Dec 2008 | ▪ Modified the connecting algorithm. |
| T | 10.1 | Marcelo Carvalho | 07 Jan 2009 | ▪ Updated driver version, no changes in the contents. |
| U | 10.2 | Joel Nascimento | 01 Jun 2009 | ▪ Implemented Standard/Legacy BCD options |
| V | 10.3 | Paulo Balbino<br>Fellipe Peternela | 13 Aug 2009 | ▪ Fixed bug when using BCD with PRE Timers registers<br>▪ Modified Read operation during a Write operation<br>▪ Modified Error Codes<br>▪ Implemented family detection allowing to generate errors if family is misconfigured<br>▪ Modified Address Parser to allow Bits configuration up to 17 |
| X | 10.4 | Fellipe Peternella | 03 Mar 2010 | ▪ Fixed bug when reading and writing to address 255 of files of type N. |
| Y | 10.5 | André Körbes | 07 Jan 2013 | ▪ Included optional PLC family parameter in the station. |
| Z | 10.6 | Caio Cerquetani | 19 Apr 2013 | ▪ Fixed validation that was not accepting valid addresses in octal format (addresses with bit 16 or 17) |
| AA | 10.7 | Paulo Balbino | 15 Apr 2014 | ▪ Fixed the issue with writing values to the latest address of a F file type |
| AB | 10.8 | Anushree Phanse | 21 Oct 2015 | ▪ Added support for PID<br>▪ Solved the problem of incorrect groups created for PID |
| AC | 10.9 | Anushree Phanse | 05 May 2016 | ▪ Fixed issue of header N:12 reading as invalid header. |