ABBTF Communication Driver

Driver for TCP/IP Communication with ABB Totalflow devices Using TCI Library

Contents

Introduction

The ABBTF driver enables communication between the Studio system and ABB Totalflow devices that support the native DB2 serial protocol using the TCI (Totalflow Communication Interface) library over TCP/IP, according to the specifications discussed in this document.

This document will help you to select, configure and execute the ABBTF 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 ABBTF driver in the Studio system.
- **Configuring the Driver**: Explains how to configure the ABBTF driver in the Studio system, including how to associate database tags with device registers.
- **Executing the Driver**: Explains how to execute the ABBTF 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 ABBTF driver configuration
- **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 XP/Vista/7 environment. If you are not familiar with Windows, then we suggest using the **Help** feature (available from the Windows desktop **Start** menu) as you work through this guide.

General Information

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

The information is organized into the following sections:

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

Device Specifications

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

- **Manufacturer:** ABB
- **Compatible Equipment:** 6000 Series microFLO, 6000 XSeriesG4, 6000 XSeriesG3
- **Programmer Software:**

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**: DB2
- **Device Runtime Software**: None
- **Third Party Library**: TCI Toolkit Library (TCIDLL.dll) installed on the system supplied by ABB
- **Specific PC Board**: None
- **Cable Wiring Scheme**: Any TCP/IP Adapter (Ethernet Board)

Driver Characteristics

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

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

> **Note:**
> You must use Adobe Acrobat® Reader™ to view the **`ABBTF.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 ABBTF driver on the following operating systems:

- Windows XP/Vista/7/2003/2008

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

The ABBTF driver supports the following functionalities:

- reading/writing registers
- logging data from the meter (daily, hourly, alarm and event)
- reading configuration data

Conformance Testing

The following hardware/software was used for conformance testing:

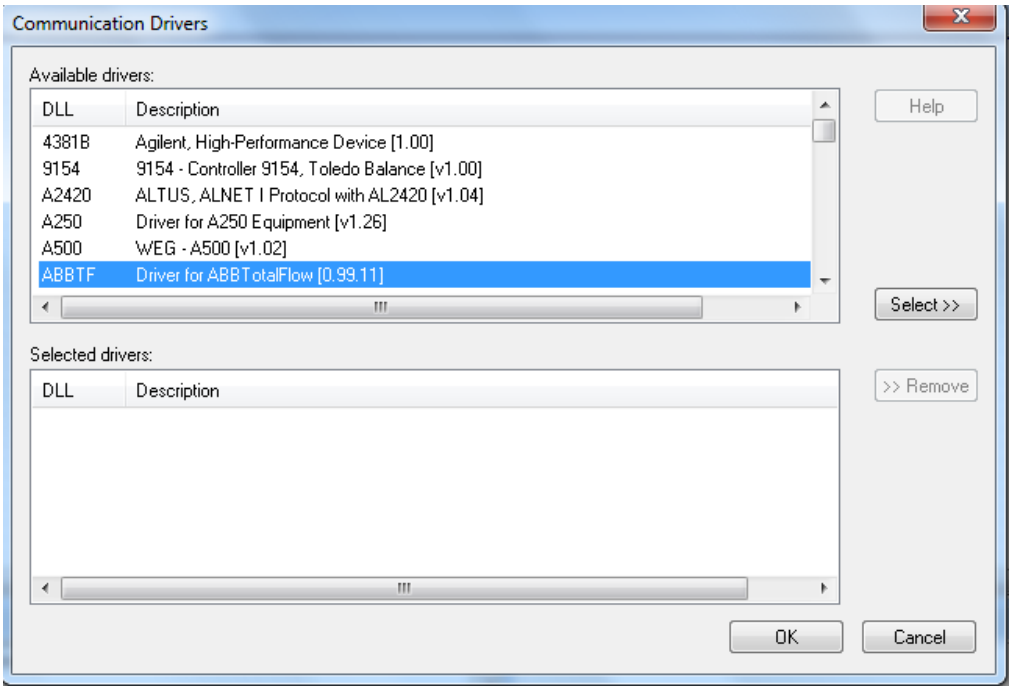
- **Driver Configuration**:
  Protocol : DB2
- **Cable**: Use specifications described in the “Network Specifications” section above.

Driver Version	Studio Version	Operating System (development)	Operating System (target)	Equipment
1.0	7.1	Windows 7	Windows XP/7/8	Flow Computer Unit

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 ABBTF driver for your Studio application:

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



Communication Drivers Dialog

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

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

Configuring the Driver

After opening Studio and selecting the ABBTF driver, you must configure the driver. Configuring the ABBTF driver is done in two parts:

- Specifying communication parameters
- Defining tags and controls in the *MAIN* and *STANDARD DRIVER SHEET*s (or Communication tables)

Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the **Station**, **Header** and **Address** fields, which are driver-specific. This document explains how to configure the **Station**, **Header** and **Address** fields only.

> **Note:**
> For a detailed description of the Studio *MAIN* and *STANDARD DRIVER SHEET*s, and information about configuring the standard fields, review the product's *Technical Reference Manual*.

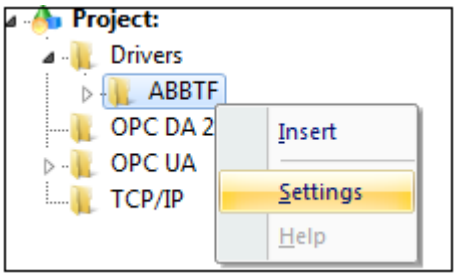
Configuring the Communication Settings

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.

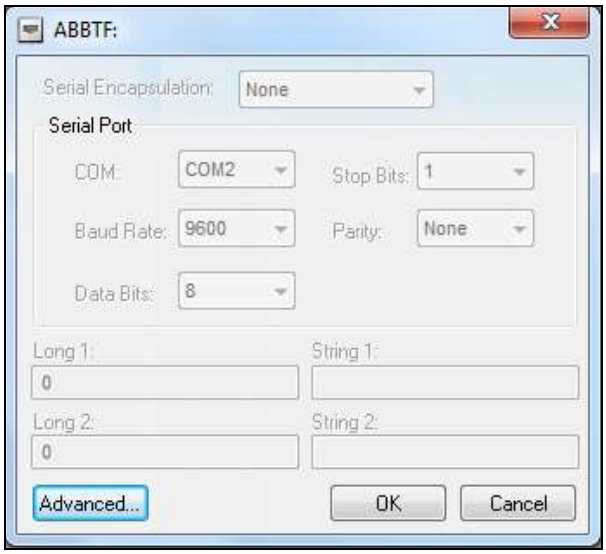
For the purposes of this document, only ABBTF driver-specific settings and procedures will be discussed here. To configure the communication settings for the ABBTF driver:

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

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

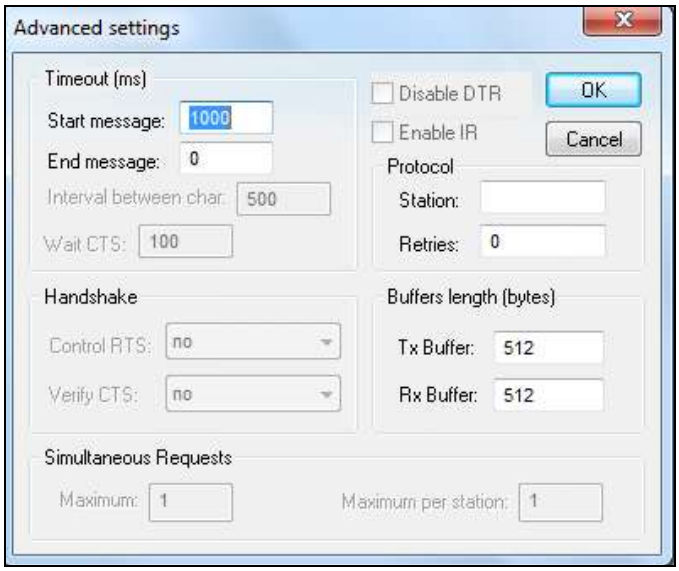


Select Settings from the Pop-Up Menu



ABBTF: Communication Parameters Dialog

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



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.

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

Configuring the Connection

The connection parameters of the target Totalflow device is specified in the file **`ABBTF.xml`**, which resides in the **Config** folder of the application. This XML file contains the following fields that are used for establishing connection.

Fields	Description	Example
id	ID number of this device, referred in the station of Main or Standard driver sheets	*2*
Ip	IP address of the target device.	*127.0.0.1*
port	Port number of the CPC. (Communication Port Controller)	*12345*
deviceId	A string to identify an ABB Totalflow device.	*EM002*
stationId	Identification number of the ABB Totalflow device.	*EM002*
deviceType	Relevent ABB device type.	*FCU*
tubeNumber	Identification number of the tube in a multi-tube device.	*2*
securityCode	Code used for access control.	*0000*

Fields	Description	Example
gatewayPortNumber	Denotes the port number of the serial-to-ethernet terminal server.	***11***
gatewayComBaudRate	When COM port is used in Ethernet encapsulation, baud rate is given. (in bits/second)	***9600***
communicationType	Type of communication with the target device.	***tcp/ip***

More than one station connection parameters can be configured in file, ABBTF.xml. Below is an example **`ABBTF.xml`** file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Stations>
   <Station id="2"
      ip="T127.0.0.1"
      port="12345"
      deviceId="EM002"
      stationId="EM002"
      deviceType="FCU"
      tubeNumber="2"
      securityCode="0000"
      gatewayPortNumber="11"
      gatewayComBaudRate="9600"
      communicationType="tcp/ip"/>
   <Station id="3"
      ip="T127.0.0.1"
      port="12345"
      deviceId="EM003"
      stationId="EM003"
      deviceType="FCU"
      tubeNumber="2"
      securityCode="0000"
```

```
            gatewayPortNumber="11"

            gatewayComBaudRate="9600"

            communicationType="tcp/ip"/>

        </Stations>
```

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.

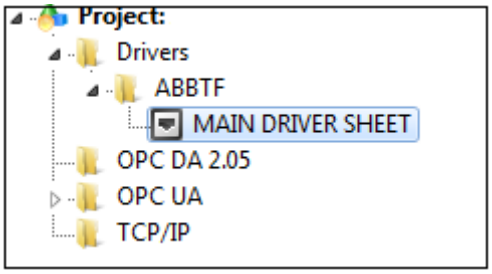
The configuration of these worksheets is described in detail in the "Communication" chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

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

MAIN DRIVER SHEET

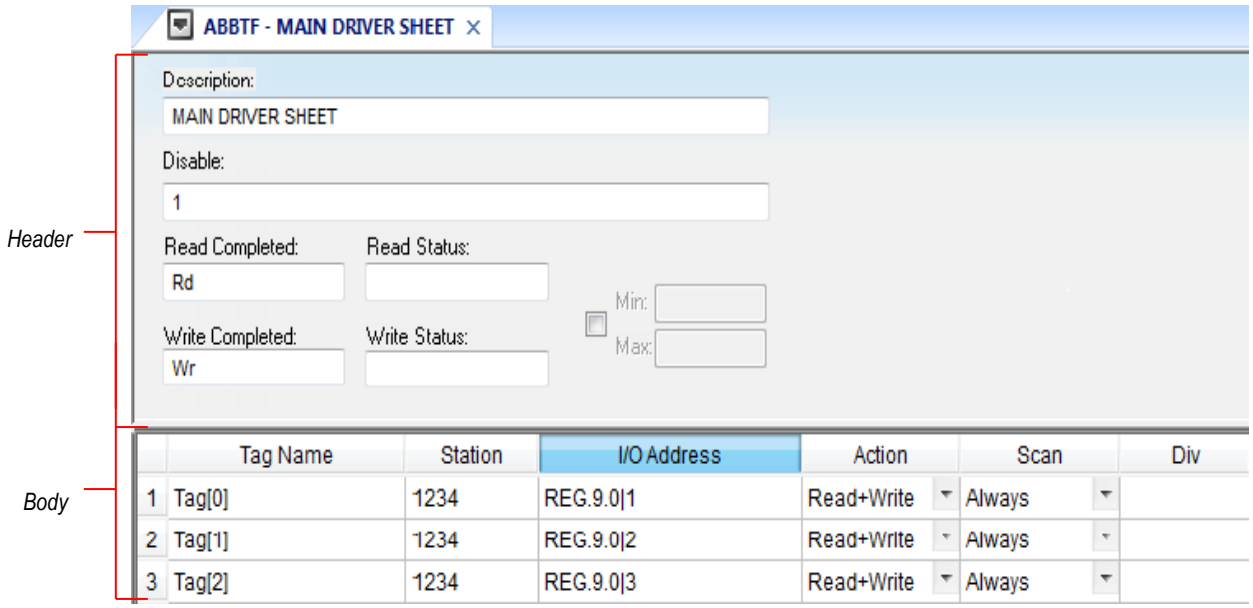
When you select the ABBTF driver and add it to your application, Studio automatically inserts the *Main Driver Sheet* in the *ABBTF* driver subfolder. To configure the Main Driver Sheet:

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



Main Driver Sheet in the ABBTF Subfolder

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



Opening the Main Driver Sheet

Most of the fields on this sheet are standard for all drivers; see the "Communication" chapter of the *Technical Reference Manual* for more information on configuring these fields. However, the **Station** and **I/O Address** fields use syntax that is specific to the ABBTF 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 ID number (node) of the target Totalflow device, using the following syntax:

     <ID Number>

     Example — **57**

     Where ***<ID Number>*** is a value between 0 and 100001.

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

     > ➲ **Attention:**
     > You cannot leave the **Station** field blank.

     For more information see **Configuring the Connection** section.

   - **I/O Address:**

     (1) Register read/write

     The address of the register consists of three values – application number, array number and register number. These three values uniquely identify any register.

     For all register types use the following syntax:

     REG.<App>.<Array>|<DataType><Register>

Examples — ***REG.9.0|B0, REG.10.0|W10, REG.8.0|F0.*** (B, W and F are data types)

Where:

– ***<DataType>***: A data type used to interpret the register being read. The default is signed byte. All of the following are accepted:

Data Type	Description
B	Signed 8 bits variable (byte)
UB	Unsigned 8 bits variable (byte)
W	Signed 16 bits variable (word)
SW	Signed 16 bits variable (word) with byte swap
UW	Unsigned 16 bits variable (word)
USW	Unsigned 16 bits variable (word) with byte swap
DW	Signed 32 bits variable (double word)
SDW	Signed 32 bits variable (double word) with byte swap
DWSW	Signed 32 bits variable (double word) with word swap
SDWSW	Signed 32 bits variable (double word) with word swap
UDW	Unsigned 32 bits variable (double word)
USDW	Unsigned 32 bits variable (double word) with byte swap
F	32 bits float points (float)
SF	32 bits float points with byte swap (float)
DF	64 bits float points (double)
SDF	64 bits float points with byte swap (double)
BCD	16 bits BCD value
BCDDW	32 bits BCD value
SBCD	16 bits BCD value with byte swap
SBCDD	32 bits BCD value with byte swap
S	String

– ***<App>***: Application. Each application is usually designed to perform a specific task, calculation or function in the ABB Totalflow device. The valid range for the application field is 0-255.

– ***<Array>***: Array Index. It is the subdivision in the application. The valid range for the array field is 0-255.

– ***<Register>***: Register Number. It denotes the register number. The valid range for the register field is 0-65535.

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

➲ **Attention:**

- For DWord registers using the unsigned option in the Address field, the associated Studio database tag must be Real Type, otherwise there may be overflow issues.

(2) Reading Configuration parameters from the device

Configuration parameters are read from the meter by giving the address as follows:

CFG|<ConfigOption>

Where:

<ConfigOption> is one of the device setup data option read from the meter.

For configuration options, see the following table:

<ConfigOption>	Type
DATA_ID	**unsigned char**
DEVICE_SETUP_REVISION	**unsigned short**
PWD_MODE	**char**
LAST_LOGIN_TAG	**unsigned short**
CHECK_SEC_CODE	**char**
PRIMARY_ELMT	**unsigned char**
PRIMARY_ELMT_MASK	**unsigned short**
VOL_CALC_METHOD	**unsigned char**
VOL_CALC_METHOD_MASK	**unsigned short**
REPORT_UNITS	**unsigned char**
REPORT_UNITS_MASK	**unsigned short**
CALC_UNITS	**unsigned char**
CALC_UNITS_MASK	**unsigned short**
Z_METHOD	**unsigned char**
Z_METHOD_MASK	**unsigned short**
VOLUME_LOG_PERIOD	**unsigned long**
VOLUME_CALC_PERIOD	**unsigned short**
CONTCT_HOUR	**unsigned char**
ORIFICE_SIZE	**float**
MONEL_ORIFICE	**char**
PIPE_SIZE	**float**
TAP_LOCATION	**unsigned char**

<ConfigOption>	Type
T_BASE	**float**
P_BASE	**float**
RSPH	**float**
VISCOSITY	**float**
ZERO_CUT_OFF	**float**
FT	**float**
FP	**float**
BAROP	**float**
FAUX	**float**
SQRT_AVGS	**char**
USE_Y	**char**
USE_FPV	**char**
USE_FW	**char**
USE_FAUX	**char**
AP_LOW_LIMIT	**float**
AP_HIGH_LIMIT	**float**
AP_LOW_CALIB	**float**
AP_MID_LOW_CALIB	**float**
AP_MID_CALIB	**float**
AP_MID_HIGH_CALIB	**float**
AP_HIGH_CALIB	**float**
DP_LOW_LIMIT	**float**
DP_HIGH_LIMIT	**float**
DP_LOW_CALIB	**float**
DP_MID_LOW_CALIB	**float**
DP_MID_CALIB	**float**
DP_MID_HIGH_CALIB	**float**
DP_HI_CALIB	**float**
RTD_INSTLLD	**char**
USE_MEAS_TEMP	**char**
FIXED_TF	**float**
TEMP_BIAS	**float**
TF_LOW_LIMIT	**float**
TF_HIGH_LIMIT	**float**
FLOW_LOW_LIMIT	**float**

<ConfigOption>	Type
FLOW_HIGH_LIMIT	**float**
LC_CONTACT1	**char**
DP_LOW_CONTACT1	**char**
DP_HIGH_CONTACT1	**char**
AP_LOW_CONTACT1	**char**
AP_HIGH_CONTACT1	**char**
REM_SENSE_CONTACT1	**char**
VOL_POINT_CONTACT1	**char**
VOLUME_SETPOINT	**float**
AUTO_RESET_CONTACT1	**char**
STREAM_ATTACHED	**char**
STREAM_ID	**unsigned long**
FIXED_AX_ON_ERROR	**char**
FIRST_ANALYSIS	**char**
TF_LOW_CONTACT1	**char**
TF_HIGH_CONTACT1	**char**
FLOW_LOW_CONTACT1	**char**
FLOW_HIGH_CONTACT1	**char**
LC_CONTACT2	**char**
DP_LOW_CONTACT2	**char**
DP_HIGH_CONTACT2	**char**
AP_LOW_CONTACT2	**char**
AP_HIGH_CONTACT2	**char**
REM_SENSE_CONTACT2	**char**
VOL_SETPOINT_CONTACT2	**char**
VOL_SETPOINT2	**float**
AUTO_RESET_CONTACT2	**char**
TF_LOW_CONTACT2	**char**
TF_HIGH_CONTACT2	**char**
FLOW_LOW_CONTACT2	**char**
FLOW_HIGH_CONTACT2	**char**
SUPPORT_PIPE_TAPS	**char**
TAP_TYPE	**unsigned char**
FB	**float**
USE_FB	**char**

<ConfigOption>	Type
USE_FR	**char**
USE_FPB	**char**
USE_FTB	**char**
USE_FTF	**char**
USE_FG	**char**
USE_FA	**char**
ORIFICE_REF_TEMP	**float**
PIPE_REF_TEMP	**float**
Z_BA	**float**
ORIFICE_EXP	**float**
PIPE_EXP	**float**
FIXED_CD	**float**
USE_CALCED_CD	**char**
BTU	**float**
GRAVITY	**float**
N2	**float**
CO2	**float**
H2S	**float**
H2O	**float**
HELIUM	**float**
METHANE	**float**
ETHANE	**float**
PROPANE	**float**
N_BUTANE	**float**
I_BUTANE	**float**
N_PENTANE	**float**
I_PENTANE	**float**
N_HEXANE	**float**
N_HEPTANE	**float**
N_OCTANE	**float**

For more information about the configuration data, please consult the manufacturer's documentation.

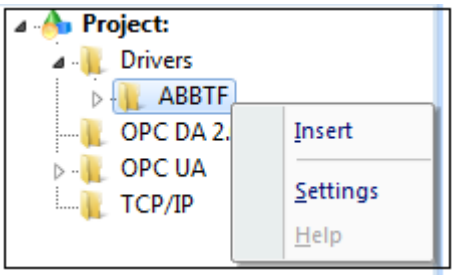
STANDARD DRIVER WORKSHEET

When you select the ABBTF 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.

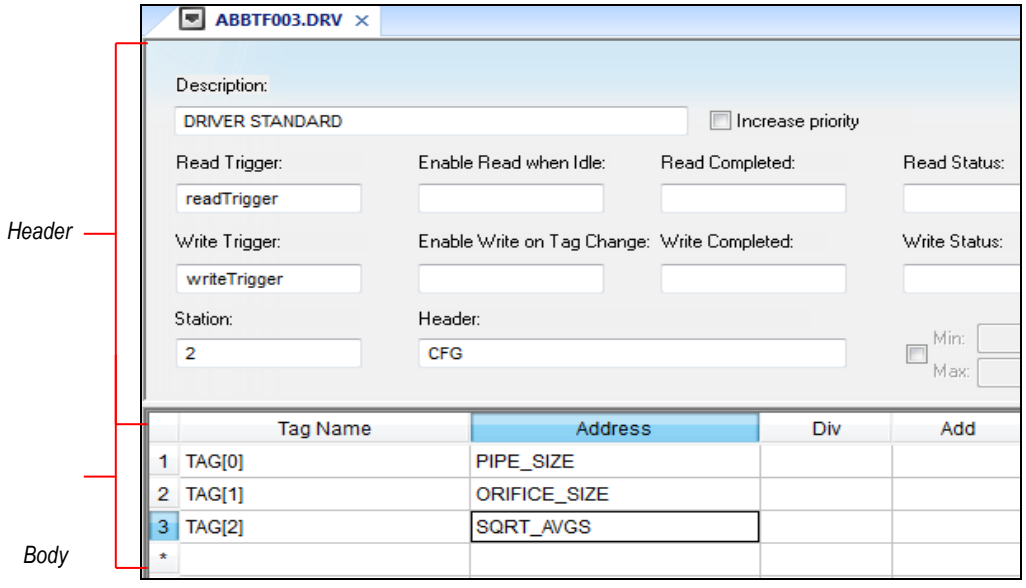
To insert a new Standard Driver Worksheet:

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



Inserting a New Worksheet

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



ABBTF Driver Worksheet

> **Note:**
> Worksheets are numbered in order of creation, so the first worksheet is **`ABBTF001.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 ABBTF driver.

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

   - **Station** field: Specify the ID number (node) of the target Totalflow device, using the following syntax:

     `<ID Number>`

     Example — **`57`**

     Where ***`<ID Number>`*** is a value between 0 and 100001.

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

     > **Attention:**
     > You cannot leave the **Station** field blank.

     For more information see **Configuring the Connection** section.

   - **Header** field:

     (a) Register read/write

     Specify the first two components of the register address on the target device. When Read/Write operations are executed for the entire worksheet (see **Read Trigger** and **Write Trigger** above), it scans the registers from the first address to the last.

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

     `REG.<App>.<Array>`

     Example — **`REG.10.0`**

     Where:

     – ***`<App>`*** is the application identifier of the register.

     – ***`<Array>`*** is the array identifier of the register.

     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 **`REG0.0`**.

     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.

     (b) Collecting historical Electronic Flow Measurement (EFM) data from the device

     The header field for logging data is as follows:

     `EFM:<FilePath>`

     Example — **`EFM:WEB\MYDAILY1234`**

Where:

- ***<FilePath>*** is an optional string. If ***<FilePath>*** is not given, then the log files are created in the root folder of the application. ***<FilePath>*** can be absolute or relative (relative to the root folder of the application). Initially, the driver checks for the relative log file path. If it is not available, absolute path is taken for file creation. File name can also be given in the ***<FilePath>.*** This file name is taken as the prefix of the log file to be created.

(c) Reading configuration parameters from the device

The header field for reading configuration parameters from the device is as follows:

CFG

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

(a) Register read/write

For all register read/write, use the following syntax in address field:

<DataType><Register>

Examples — **B1**, W**4.(**B and W are data types)

Where:

– ***<DataType>***: A data type used to interpret the register being read. The default is swapped float. All of the following are accepted:

Data Type	Description
B	Signed 8 bits variable (byte)
UB	Unsigned 8 bits variable (byte)
W	Signed 16 bits variable (word)
SW	Signed 16 bits variable (word) with byte swap
UW	Unsigned 16 bits variable (word)
USW	Unsigned 16 bits variable (word) with byte swap
DW	Signed 32 bits variable (double word)
SDW	Signed 32 bits variable (double word) with byte swap
DWSW	Signed 32 bits variable (double word) with word swap
SDWSW	Signed 32 bits variable (double word) with word swap
UDW	Unsigned 32 bits variable (double word)
USDW	Unsigned 32 bits variable (double word) with byte swap
F	32 bits float points (float)
SF	32 bits float points with byte swap (float)
DF	64 bits float points (double)
SDF	64 bits float points with byte swap (double)

Data Type	Description
BCD	16 bits BCD value
BCDDW	32 bits BCD value
SBCD	16 bits BCD value with byte swap
SBCDD	32 bits BCD value with byte swap
S	String

– ***<Register>*** is the register identifier, which is the third constituent of the register address.

(b) Collecting historical Electronic Flow Measurement (EFM) data from the device

For getting the historical data files (comma separated files), use the following syntax in address field:

<LogOption>:<HistoryDays>:<FileNameSuffix>

Examples — **DAILY:50:MYDAILY1234, ALARMS:50:MYALARMS1234, EVENTS:1000:MYEVENTS1234. (**MYDAILY1234, MYALARMS1234, MYEVENTS1234 are file name suffixes**)**

Where:

<LogOption>:

1. **ALARMS** : for logging alarms data
2. **EVENTS** : for logging events data
3. **HOURLY** : for logging hourly history data
4. **DAILY** : for logging daily history data

<HistoryDays>:

It is an optional number. The history data of ***<HistoryDays>*** preceding days are generated. This number is given for addresses - ALARMS, EVENTS, HOURLY, and DAILY. The fetched history data from the device are stored in a comma separated file.

For ALARMS and DAILY, the default value is 50, ie if this field is kept blank or 0 is entered, 50 will be taken as the default value. If a value greater than 50 is entered, data of only previous 50 days' are retrieved.

For EVENTS, the default value is 1000, ie if this field is kept blank or 0 is entered, 1000 will be taken as the default value.

For HOURLY, the default value is 0, ie if this field is kept blank, 0 will be taken as the default value. The hourly data of the immediate preceding day are retrieved if ***<HistoryDays>*** is 0, otherwise previous ***<HistoryDays>***th data are retrieved.

If no data is retrieved from the device, a blank file with the specified name is created.

<FileNameSuffix>:

It is an optional string. If ***<FileNameSuffix>*** is not given, then the file names are created with the default suffixes. The default file name suffixes are

1. “alarms” for ALARMS
2. “events” for EVENTS
3. “hourly” for HOURLY
4. “daily” for DAILY

(c) Reading configuration parameters from the device

Configuration parameters can be read from the device by using the following syntax in the address field:

<ConfigOption>

Where:

<ConfigOption> is one of the device setup configuration data option read from the device.

For configuration options, see the following table:

<ConfigOption>	Type
DATA_ID	**unsigned char**
DEVICE_SETUP_REVISION	**unsigned short**
PWD_MODE	**char**
LAST_LOGIN_TAG	**unsigned short**
CHECK_SEC_CODE	**char**
PRIMARY_ELMT	**unsigned char**
PRIMARY_ELMT_MASK	**unsigned short**
VOL_CALC_METHOD	**unsigned char**
VOL_CALC_METHOD_MASK	**unsigned short**
REPORT_UNITS	**unsigned char**
REPORT_UNITS_MASK	**unsigned short**
CALC_UNITS	**unsigned char**
CALC_UNITS_MASK	**unsigned short**
Z_METHOD	**unsigned char**
Z_METHOD_MASK	**unsigned short**
VOLUME_LOG_PERIOD	**unsigned long**
VOLUME_CALC_PERIOD	**unsigned short**
CONTCT_HOUR	**unsigned char**
ORIFICE_SIZE	**float**
MONEL_ORIFICE	**char**

<ConfigOption>	Type
PIPE_SIZE	**float**
TAP_LOCATION	**unsigned char**
T_BASE	**float**
P_BASE	**float**
RSPH	**float**
VISCOSITY	**float**
ZERO_CUT_OFF	**float**
FT	**float**
FP	**float**
BAROP	**float**
FAUX	**float**
SQRT_AVGS	**char**
USE_Y	**char**
USE_FPV	**char**
USE_FW	**char**
USE_FAUX	**char**
AP_LOW_LIMIT	**float**
AP_HIGH_LIMIT	**float**
AP_LOW_CALIB	**float**
AP_MID_LOW_CALIB	**float**
AP_MID_CALIB	**float**
AP_MID_HIGH_CALIB	**float**
AP_HIGH_CALIB	**float**
DP_LOW_LIMIT	**float**
DP_HIGH_LIMIT	**float**
DP_LOW_CALIB	**float**
DP_MID_LOW_CALIB	**float**
DP_MID_CALIB	**float**
DP_MID_HIGH_CALIB	**float**
DP_HI_CALIB	**float**
RTD_INSTLLD	**char**
USE_MEAS_TEMP	**char**
FIXED_TF	**float**
TEMP_BIAS	**float**
TF_LOW_LIMIT	**float**

<ConfigOption>	Type
TF_HIGH_LIMIT	**float**
FLOW_LOW_LIMIT	**float**
FLOW_HIGH_LIMIT	**float**
LC_CONTACT1	**char**
DP_LOW_CONTACT1	**char**
DP_HIGH_CONTACT1	**char**
AP_LOW_CONTACT1	**char**
AP_HIGH_CONTACT1	**char**
REM_SENSE_CONTACT1	**char**
VOL_POINT_CONTACT1	**char**
VOLUME_SETPOINT	**float**
AUTO_RESET_CONTACT1	**char**
STREAM_ATTACHED	**char**
STREAM_ID	**unsigned long**
FIXED_AX_ON_ERROR	**char**
FIRST_ANALYSIS	**char**
TF_LOW_CONTACT1	**char**
TF_HIGH_CONTACT1	**char**
FLOW_LOW_CONTACT1	**char**
FLOW_HIGH_CONTACT1	**char**
LC_CONTACT2	**char**
DP_LOW_CONTACT2	**char**
DP_HIGH_CONTACT2	**char**
AP_LOW_CONTACT2	**char**
AP_HIGH_CONTACT2	**char**
REM_SENSE_CONTACT2	**char**
VOL_SETPOINT_CONTACT2	**char**
VOL_SETPOINT2	**float**
AUTO_RESET_CONTACT2	**char**
TF_LOW_CONTACT2	**char**
TF_HIGH_CONTACT2	**char**
FLOW_LOW_CONTACT2	**char**
FLOW_HIGH_CONTACT2	**char**
SUPPORT_PIPE_TAPS	**char**
TAP_TYPE	**unsigned char**

<ConfigOption>	Type
FB	**float**
USE_FB	**char**
USE_FR	**char**
USE_FPB	**char**
USE_FTB	**char**
USE_FTF	**char**
USE_FG	**char**
USE_FA	**char**
ORIFICE_REF_TEMP	**float**
PIPE_REF_TEMP	**float**
Z_BA	**float**
ORIFICE_EXP	**float**
PIPE_EXP	**float**
FIXED_CD	**float**
USE_CALCED_CD	**char**
BTU	**float**
GRAVITY	**float**
N2	**float**
CO2	**float**
H2S	**float**
H2O	**float**
HELIUM	**float**
METHANE	**float**
ETHANE	**float**
PROPANE	**float**
N_BUTANE	**float**
I_BUTANE	**float**
N_PENTANE	**float**
I_PENTANE	**float**
N_HEXANE	**float**
N_HEPTANE	**float**
N_OCTANE	**float**

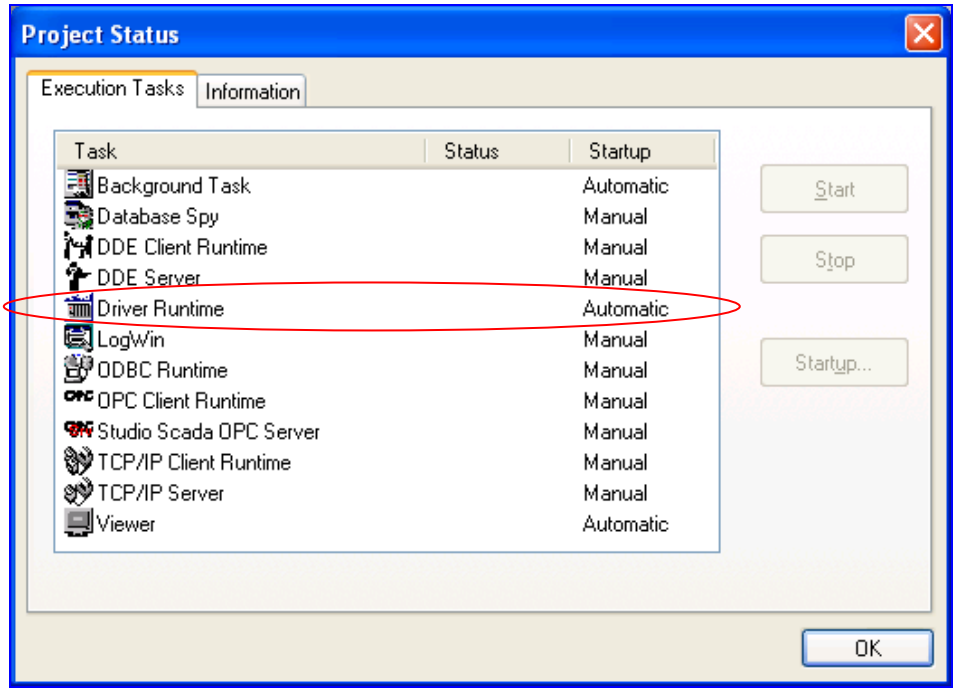
For more information about the configuration data, please consult the manufacturer's documentation.

Executing the Driver

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

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

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



Project Status Dialog

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

Troubleshooting

If 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., TCI Toolkit). 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

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

Revision History

Doc. Revision	Driver Version	Author	Date	Description of Changes
A	1.0	Anoop R.	Oct/08/2013	▪ First driver version
B	1.0	André K.	Nov/05/2013	▪ Removed incorrect support for CE