SSTDH Communication Driver

Driver for Serial Communication
with Allen-Bradley Devices Using the DH or DH+ Protocol

Contents

Introduction

The SSTDH driver enables serial communication between the Studio system (with a SST board installed) and Allen-Bradley devices using the DH or DH+ protocol, according to the specifications discussed in this document.

This document will help you to select, configure and execute the SSTDH 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 SSTDH driver in the Studio system.
- **Configuring the Driver**: Explains how to configure the SSTDH driver in the Studio system, including how to associate database tags with device registers.
- **Executing the Driver**: Explains how to execute the SSTDH 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 XP, 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 serial communication between the SSTDH driver in Studio and an Allen-Bradley device using the DH or DH+ 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:**
  - PLC5 Series
  - PLC2 Series
  - SLC500 Series
- **Programmer Software:** RSLogix, ASP, RS6200

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

Network Specifications

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

- **Device Communication Port:** Any PLC channel configured for DH+
- **Physical Protocol:** DH+
- **Logic Protocol:** DH or DH+
- **Device Runtime Software:** SST board drivers
- **Specific PC Board:** SST 5136-SD-104 or SST-DHP-PCI
- **Cable:** 1770-CD shielded twin axial
  - Trunk line: 10,000 ft. maximum
  - Drop lines: 100 ft. maximum

Driver Characteristics

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

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

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

- Windows XP

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

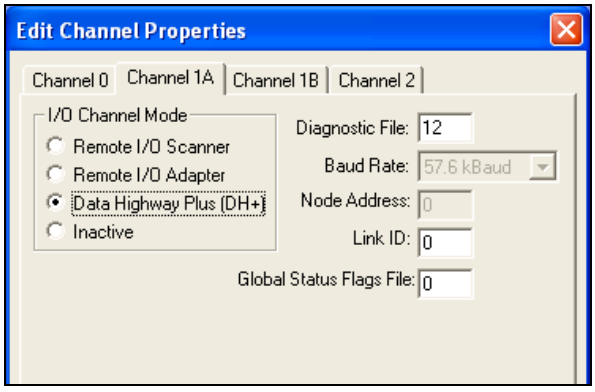
The SSTDH driver supports the following registers:

Register Type	Length	Range	Write	Read	Bit	Integer	Float	String	BCD
O (Output)	2 Bytes	-32768 to 32767	•	•	•	•	–	–	•
I (Input)	2 Bytes	-32768 to 32767	•	•	•	•	–	–	•
S (Status)	2 Bytes	-32768 to 32767	•	•	•	•	–	–	–
B (Binary)	2 Bytes	-32768 to 32767	•	•	•	•	–	–	•
T (Timer)	6 Bytes	-32768 to 32767 – ACC/PRE	•	•	–	–	–	–	–
C (Counter)	6 Bytes	-32768 to 32767 – ACC/PRE	•	•	–	–	–	–	–
R (Control)	6 Bytes	-32768 to 32767 – POS/LEN	•	•	–	–	–	–	–
F (Float)	4 Bytes	1.175494351e-038 to 3.402823466e+038	•	•	–	–	•	–	–
N (Integer File)	2 Bytes	-32768 to 32767	•	•	•	•	–	–	•
A (ASCII File)	2 Bytes	ASC values	•	•	–	•	–	•	–
ST (String File)	*n* Bytes	–	•	•	–	–	–	•	–

Conformance Testing

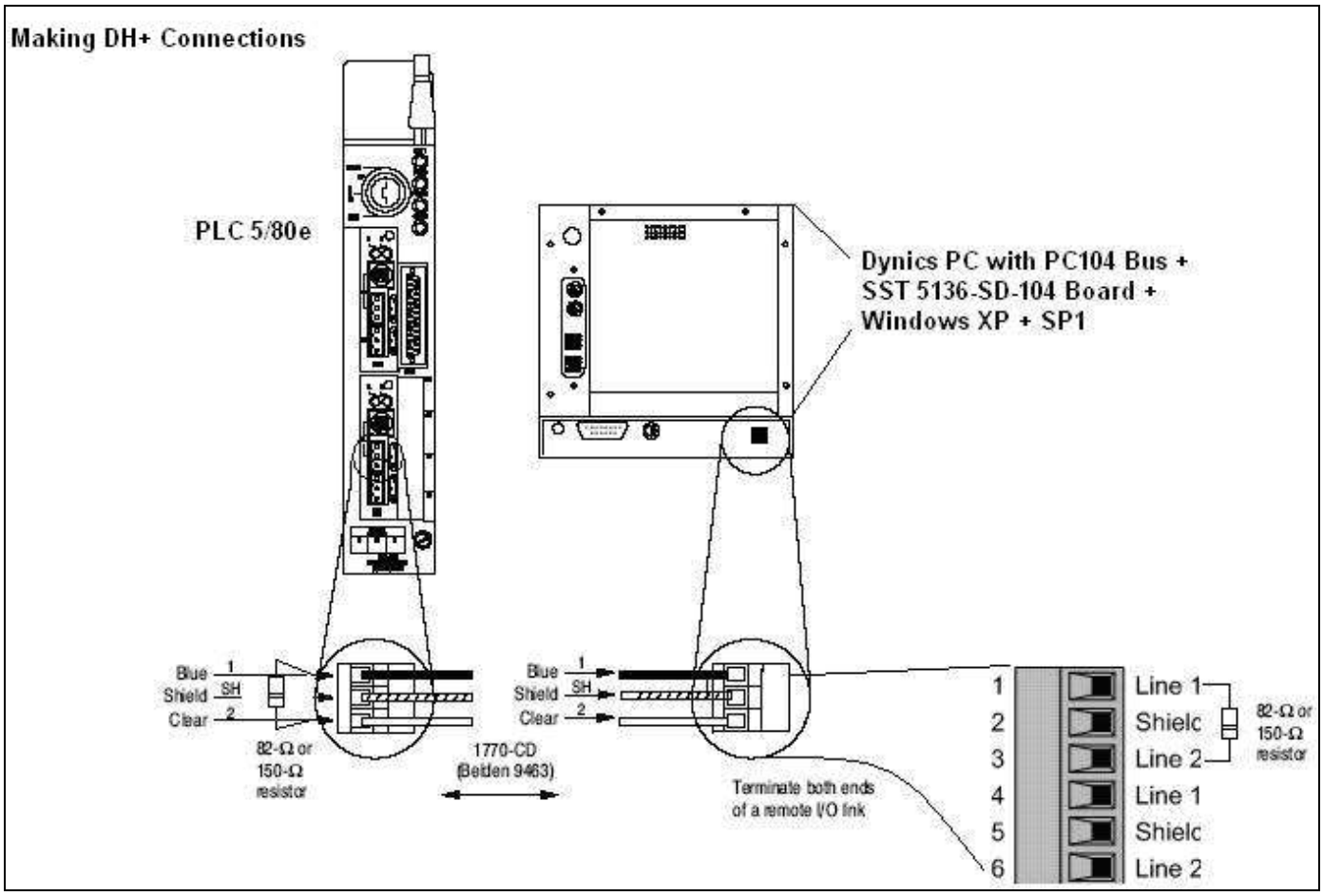
The following hardware/software was used for conformance testing:

- **Equipment:** Allen-Bradley PLC5/80e
- **Driver Configuration**:
  - **Baud Rate**: 57.6k
  - **Protocol**: Data Highway Plus (DHP or DH+)
  - **Channel**: 1A



Editing Channel Properties for the SST Board

- **Cable:** See diagram below…



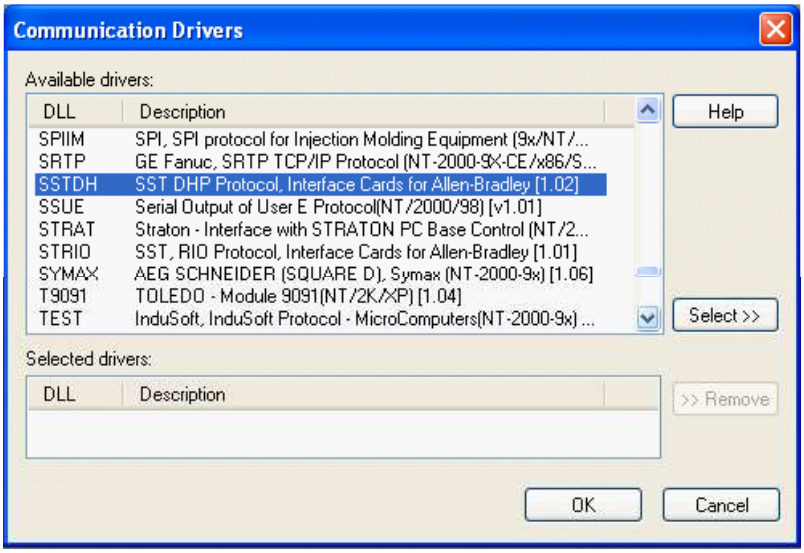
Cabling Diagram for DH+ Connection

- **Operating System** (development): Windows XP
- **Operating Systems** (target): Windows XP
- **Studio Version**: 7.1
- **Driver Version**: 1.08

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 SSTDH driver for your Studio application:

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



Communication Drivers Dialog

3. When the **SSTDH** 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:**
> Other than configuring the Device Driver and the SST Board, 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 Driver

Once you have selected the SSTDH 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.

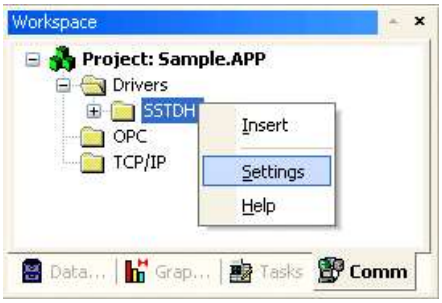
Configuring the Communication Settings

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.

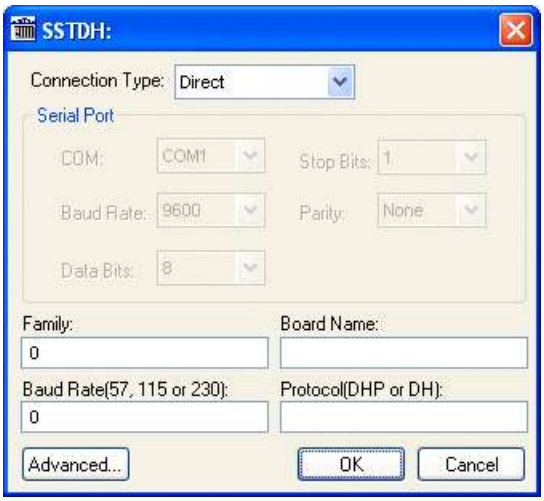
For the purposes of this document, only SSTDH driver-specific settings and procedures will be discussed here. To configure the communication settings for the SSTDH driver:

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

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



Select Settings from the Pop-Up Menu



SSTDH: Communication Settings Dialog

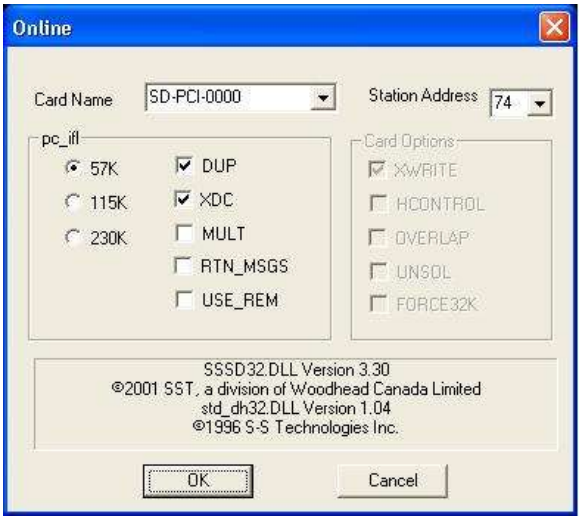
3. In the *Communication Settings* dialog, configure the driver settings to enable communication with your target device. To ensure error-free communication, the driver settings must *exactly match* the corresponding settings on the device. Please consult the manufacturer's documentation for instructions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

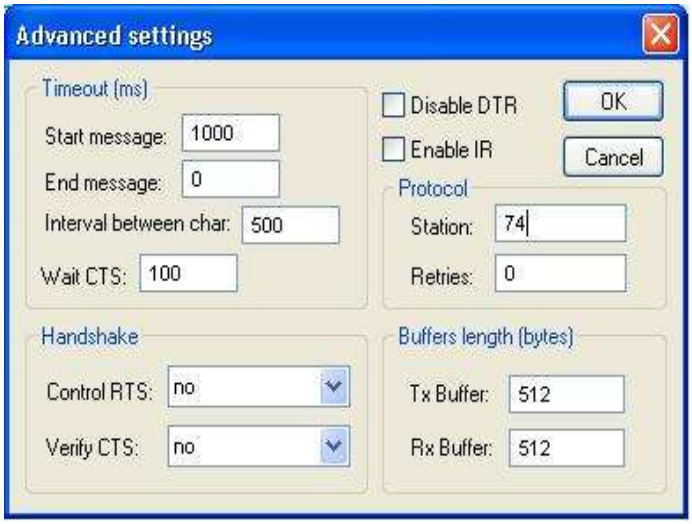
   Depending on your circumstances, you may need to configure the driver *before* you have configured your target device. If this is the case, then take note of the driver settings and have them ready when you later configure the device.

> ➲ **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:

Parameter	Default Value	Valid Values	Description
Family	–	**2**	Driver uses "unprotected Read/Write" command (in the DH+ protocol) to communicate with the PLC2 family.
		5	Driver uses "typed Read/Write" command (in the DH+ protocol) to communicate with the PLC5 family. The address for all data types is decimal.
		500	Driver uses "protected typed logical Read/Write" command (in the DH+ protocol) to communicate with the SLC500 family.
Board Name	–	any	Name of the SST board installed in the host PC.
Baud Rate	**115**	**57**	Baud rate at which the installed SST board will communicate with the target device.
		115	
		230	
Protocol	**DHP**	**DH**	Protocol that the installed SST board will use to communicate with the target device. **NOTE:** DHP is the same as DH+.
		DHP	





Setting	Default	Description
Station	__	Set the *station value* using the same number configured in the diagnostic software.

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

Note:

Additional communication settings can be accessed in the *Advanced Settings* dialog. To open this dialog, simply click the **Advanced** button in the *Communication Parameters* 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.

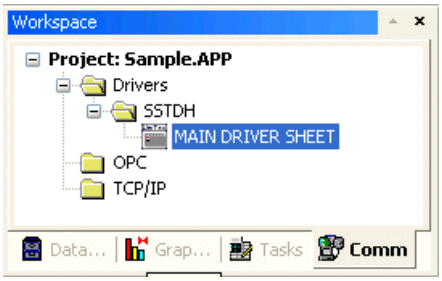
The configuration of these worksheets is described in detail in the "Communication" chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

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

MAIN DRIVER SHEET

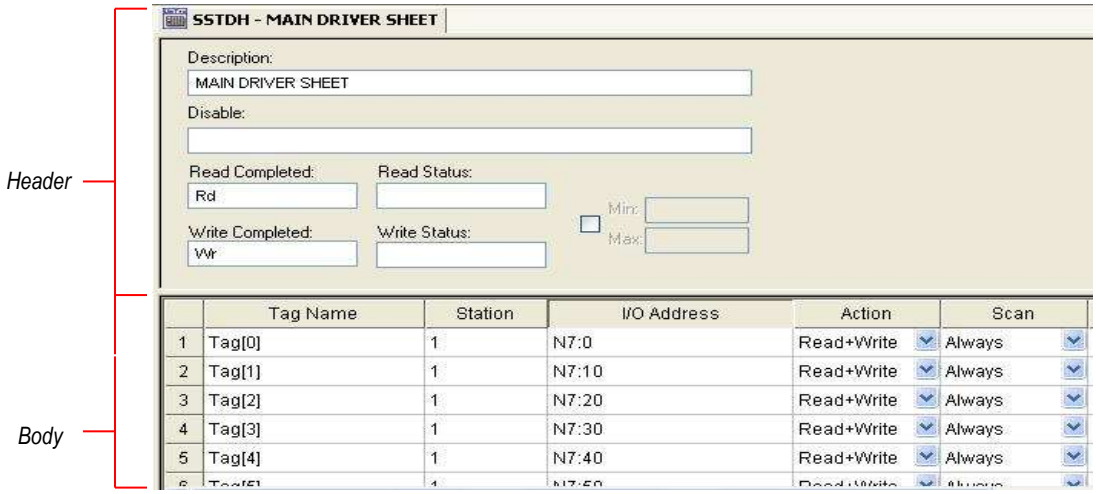
When you select the SSTDH driver and add it to your application, Studio automatically inserts the *Main Driver Sheet* in the *SSTDH* driver subfolder. To configure the Main Driver Sheet:

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



Main Driver Sheet in the SSTDH 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 SSTDH 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:

     <Device Address>

     Example — `3`

     Where ***<Device Address>*** is the device's configured address on the DH+ network. This must be a valid value between `1` and `31`.

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

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

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

  For Inputs and Outputs, use the following syntax:

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

  Example — **O:1.W2/4**

  For Status, use the following syntax:

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

  Example — **S:W0/4**

  For Binary and Integer, use the following syntax:

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

  Example — **N7:W150/2**

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

  <Type><Type Group>:<Data Format><Address>.<Element> *or*

  <Type><Type Group>:<Data Format><Address>/<Element>

  Example — **T4:W0.DN**

  For ASCII and String, use the following syntax:

  <Type><Type Group>:<Data Format><Address>.<Number of Bytes>

  Example — **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), **F** (Float), **A** (ASCII) and **ST** (String).

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

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

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

  – ***<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** (Float or Double Word), and **S** (String).

  > ➲ **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*.

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

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

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

– ***<Element>*** : Element type for Timer, Counter or Control, according to the following table:

Register	Elements																	
	DN	PRE	ACC	EN	TT	CON	UN	OV	CD	CU	FD	IN	UL	ER	EM	EU	LEN	POS
Timer	•	•	•	•	•	•	–	–	–	–	–	–	–	–	–	–	–	–
Counter	•	•	•	–	–	•	•	•	•	•	–	–	–	–	–	–	–	–
Control	•	–	–	•	–	•	–	–	–	–	•	•	•	•	•	•	•	•

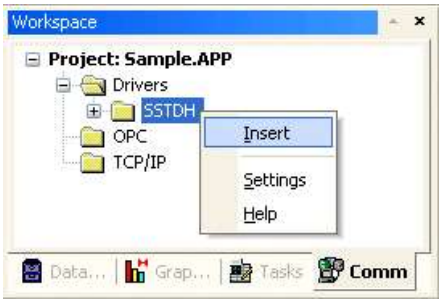
STANDARD DRIVER WORKSHEET

When you select the SSTDH 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.

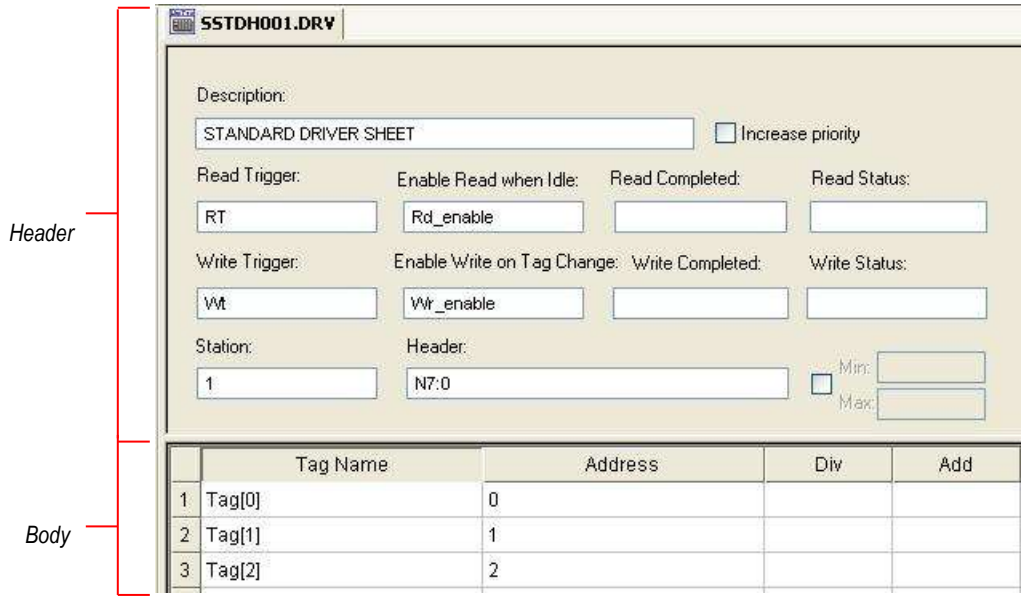
To insert a new Standard Driver Worksheet:

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



Inserting a New Worksheet

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



SSTDH Driver Worksheet

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

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

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

     <Device Address>

     Example — **3**

     Where ***<Device Address>*** is the device's configured address on the DH+ network. This must be a valid value between **1** and **31**.

     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>.<Address Reference>

     Example — **O:1.0**

     For Status, use the following syntax:

     <Type>:<Address Reference>

     Example — **S:0**

     For Binary, Integer, Timer, Counter and Control, use the following syntax:

     <Type><Type Group>:<Address Reference>

     Example — **N7:0**

     For ASCII and String, use the following syntax:

     <Type><Type Group>:<Address Reference>

     Example — **ST15:0**

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. Reads and Writes the Output addresses
Input	`I:0.0`	Varies according to the equipment	Physical inputs: Where "I" means input. Reads and Writes the Input addresses
Status	`S:0`	Varies according to the equipment	Reads and Writes the status words.
Binary	`B3:0`	0 to 255	Reads and Writes the Binary Operator.
Integer	`N7:0`	0 to 255	Reads and Writes the Integer addresses.
Timer	`T4:0`	0 to 255	Reads and Writes the Timer addresses.
Counter	`C5:0`	0 to 255	Reads and Writes the Counter addresses.
Control	`R6:0`	0 to 255	Reads and Writes the Control addresses.
Float	`F8:0`	0 to 255	Reads and Writes the Float addresses.
ASCII	`A14:0`	0 to 255	Reads and Writes the ASCII addresses: For **SLC500**, the "Address Reference" is defined in Words. For **PLC5**, the "Address Reference" is defined in Bytes.
String	`ST15:0`	0 to 255	Reads and Writes the String addresses.

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

   For Inputs and Outputs, use the following syntax:

   [Data Format]<Octet Number>/[Bit]

   Example — **0/3**

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

   [Data Format]<Address Offset>/[Bit]

   Example — **10/12**

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

   [Data Format]<Address Offset>/<Element> *or*

   [Data Format]<Address Offset>.<Element>

   Example — **2/PRE**

   For ASCII and String, use the following syntax:

   <Address Offset>.<Number of Bytes>

   Example — **1.2**

   Where:

   – ***[Data Format]*** (optional): Format of the data being read or written, which determines how Studio will handle the data. Valid values are **W** (Word), **B** (BCD), **F** (Float or Double Word), and **S** (String).

   – ***<Octet Number>*** : Number (in octal) of the desired Input or Output on the I/O card that was specified in the **Header** above.

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

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

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

   – ***<Element>*** : Element type for Timer, Counter or Control, according to the following table:

Register	Elements																	
	DN	PRE	ACC	EN	TT	CON	UN	OV	CD	CU	FD	IN	UL	ER	EM	EU	LEN	POS
Timer	•	•	•	•	•	•	–	–	–	–	–	–	–	–	–	–	–	–
Counter	•	•	•	–	–	•	•	•	•	•	–	–	–	–	–	–	–	–
Control	•	–	–	•	–	•	–	–	–	–	•	•	•	•	•	•	•	•

➲ **Attention:**

You can use the **Write trigger** field for the Bit Writing function. However it are going to work like a write Item (writing a tag per time). If you have a lot of address in the DriverSheet it can be a problem.

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	0/7
I:0/10	I:0.0	0/8
I:0/17	I:0.0	0/15
I:0/25	I:0.0	1/9
I:3/4	I:3.0	0/4
O:0/7	O:0.0	0/7
O:0/10	O:0.0	0/8
O:0/17	O:0.0	0/15
O:0/25	O:0.1	0/9
O:3/4	O:3.0	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
T4:0/ACC	T4:0	0.ACC
T4:0/PRE	T4:0	0.ACC
T15:0/EN	T15:0	0.EN
T14:0/ACC	T14:0	0.ACC
T14:2/PRE	T14:1	1.PRE
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
A14:0 (default size: 2 bytes)	A14:0	0.2
A14:1 (default size: 2 bytes)	A14:1	0.2
A14:1 (default size: 2 bytes)	A14:0	1.2

Address on the Device	Header Field	Address Field
A14:0 to A14:2	**A14:0**	**0.6**
ST15:0 (String: maximum 20 bytes)	**ST15:0**	**0.20**
ST15:1 (String: maximum 50 bytes)	**ST15:0**	**1.50**
ST15:2 (String: maximum 10 bytes)	**ST15:1**	**1.10**

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

➲ **Attention:**

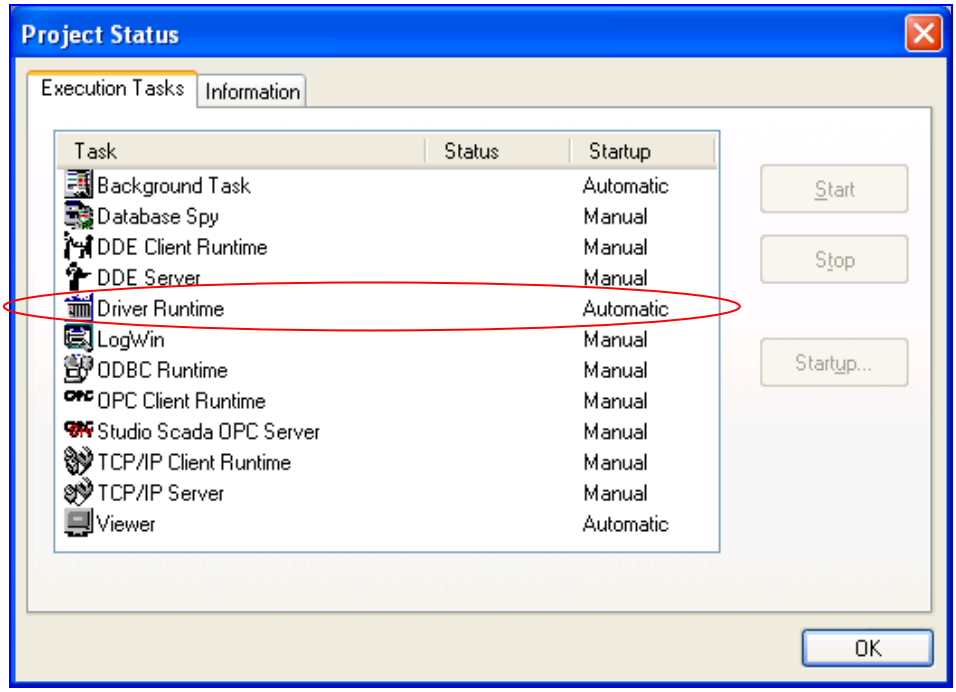
You must not configure a range of addresses greater than the maximum block size (data buffer length) supported by each device within the same worksheet. The maximum data buffer length for this driver is 242 bytes per worksheet.

Executing the Driver

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

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

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



Project Status Dialog

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

Troubleshooting

If the SSTDH driver fails to communicate with the target device, then the database tag(s) that you configured for the **Read Status** or **Write Status** fields of the Driver Sheet will receive an error code. Use this error code and the following table to identify what kind of failure occurred.

Error Code	Description	Possible Causes	Procedure to Solve
0	OK	Communication without problems	None required
3	Initialization error	Requested DLLs not found	Verify the DLL's validity.
6	Reading error	Invalid reading	Try reading again, and/or check the configuration of application parameters.
7	Writing error	Invalid writing	Try writing again, and/or check the configuration of application parameters.
12	Station configuration error	The **Station** field on the worksheet is blank.	Enter a valid value in the **Station** field.
20	Invalid address	▪ Timers, Counters, and Controls files were not specified in the correct fields of the **Address** column. ▪ Wrong address syntax was specified for the other files.	▪ Check the **Header** field. ▪ If you used a TAG, check the TAG value validity for the specified addresses. ▪ If you did not use a TAG and you changed the **Header**, check the validity of the addresses for the new **Header**.
30	Invalid header	Invalid **Header** entered in the *Driver Configuration* worksheet.	Enter a valid **Header**.
32	Block size error	Offset specified for the *Driver Configuration* worksheet exceeds the maximum, and the message cannot be framed.	Change the offsets or create a new worksheet.

⇨ **Tip:**

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

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

If you are unable to establish communication between Studio and the target device, then try instead to establish communication using the device's own programming software (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.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of Changes
A	1.00	Leandro G. Coeli	Dec/29/2004	First driver version
B	1.01	Leandro G. Coeli	Feb/21/2005	Fixed Bugs in Communication Parameters
C	1.02	Leandro G. Coeli	Mar/15/2005	Implemented MDS
D	1.03	Leandro G. Coeli	Sep/6/2005	Fixed problems on MDS
E	1.04	Eric Vigiani	Jun/16/2006	Fixed problem in the Write Item when it uses the SLC500 PLC.
F	1.04	Michael D. Hayden	Sep/7/2006	Documentation edited for language and usability.
G	1.05	Athur S. Allievi	Dec/4/2006	Fixed errors with C, T, R and ST operands.
H	1.06	Graziane C. Forti	Dec/21/2006	- Fixed problem with writing of C5, R6 and T4 operands. - Fixed problem with writing/reading of ASCII operand. - Fixed problem with configuration of BCD (limit digits – 0 to 9). - Fixed problem with Data Types ('F' and 'S'). - Fixed problem using formats in MDS - Fixed problem writing Input/Status operand - Fixed problem ASCII operand - Fixed problem reading T4, C5 and R6 (Unsigned problem) - Fixed problem configuring address T4, C5 and R6 (MDS/SDS incompatibility)
I	1.07	André Körbes	Sep/23/2010	- Fixed virtual group generation - Fixed reading/writing bugs
J	1.07	Andre Bastos	Jul/25/2011	- Documentation changes only. No changes on the driver
K	1.08	Paulo Balbino	Jan/15/2013	- Resolved issue when communicating with Input and Outputs