SPI Storm Studio

User's Guide

Byte Paradigm – info@byteparadigm.com

Table of Contents

References

[]

History

Version	Date	Description
0.90	29 July 2011	Preliminary revision – to be completed.
1.00	August 2012	Completed missing sections
1.01	August, 22nd, 2012	Added file formats description
1.02	Sept., 20th, 2012	Added bit ordering for data representation option description + minor changes
1.03	Sept. 30th, 2013	Added description of custom segments using SS2 and SS3 to apply constant values. Completed usage description of WE line.

1 About this guide

This user's guide describes SPI Storm Studio software, used to control Byte Paradigm's SPI Storm device.

2 Installing SPI Storm Studio

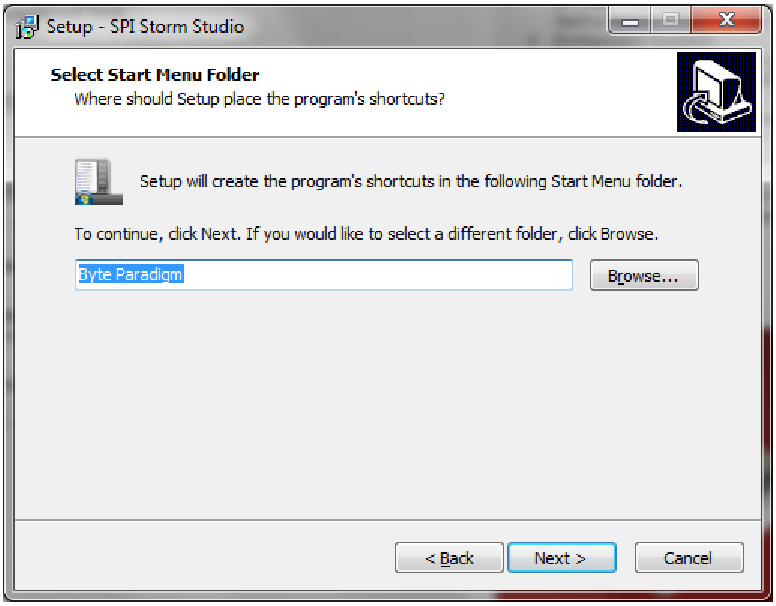
2.1 System requirements

- PC installed with Microsoft Windows XP, Windows VISTA or Windows 7 - 32-bit or 64-bit versions.
- 20 MB of free space.
- One free USB 2.0 port.
- Microsoft .NET Framework 4 Client Profile runtime installed.

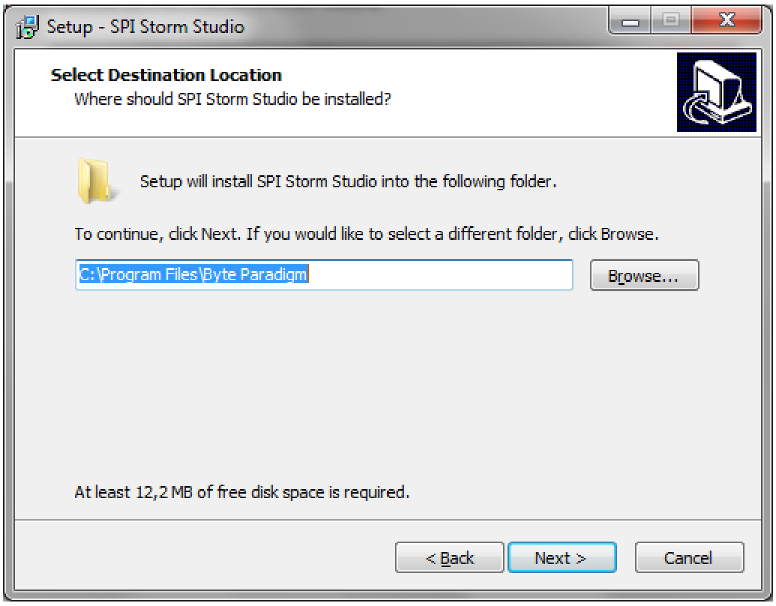
2.2 Installation Wizard

- Download SPI Storm Studio from http://www.byteparadigm.com/download-16.html
- Double-click on archive to start the installation wizard.
- At the wizard welcome screen, click on **Next>**

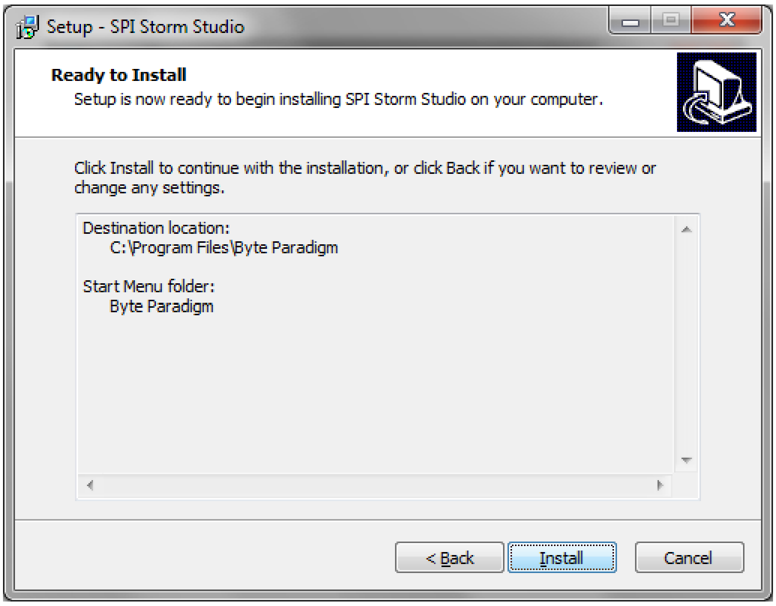
- The next screen lets you choose the 'Start Menu' folder where the SPI Storm Studio shortcuts will be installed. The folder **'Byte Paradigm'** is chosen by default.
  Select the destination folder and click on **Next>**



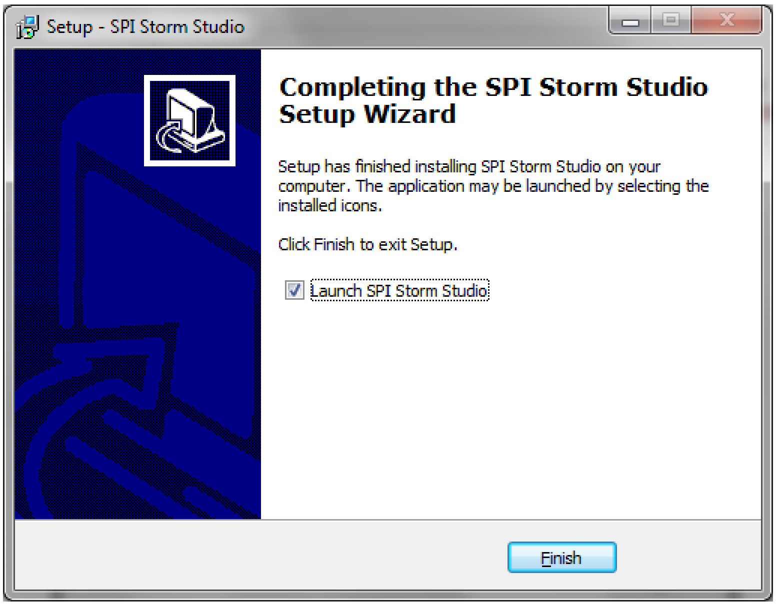
– The next screen lets you choose the destination directory on your PC hard drive. Default is: **'c:\Program Files\Byte Paradigm'**.
Select the destination directory and click on **Next>**



– Finally, click in **'Install'** at the **'Ready to install'** screen.



- Once setup is complete, the final screen offers to launch SPI Storm Studio. Select the appropriate option and click on '**Finish'** to finish the software installation.

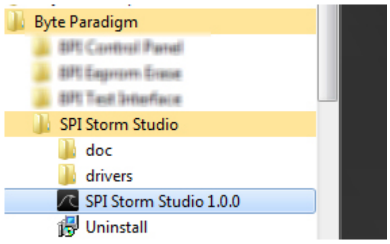


2.3 Installing the USB driver

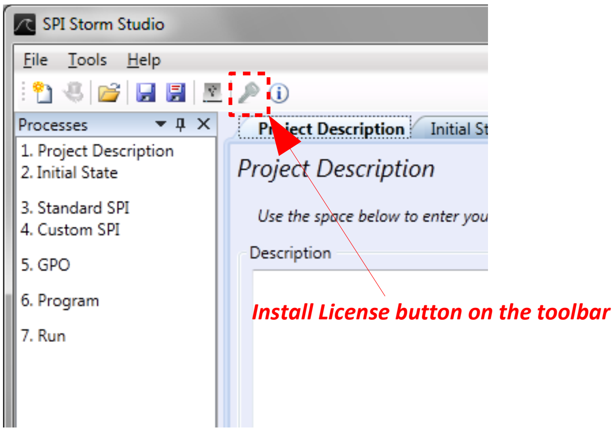
- Connect your SPI Storm device to one free USB port of your PC with the provided cable.
- When prompted, locate and install the USB driver:
  - 32-bit operating system, the driver is located in: <Installation root>\SPIStormStudio\Drivers\x86
  - 64-bit operating system, the driver is located in: <Installation root>\SPIStormStudio\Drivers\x64

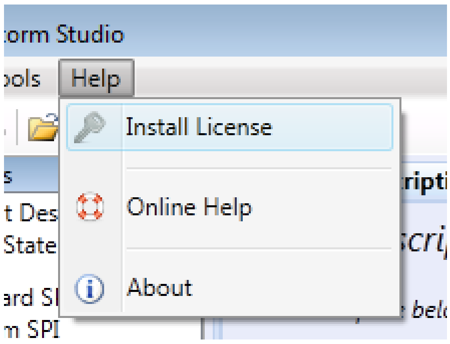
2.4 Installing the License File

- Start **SPI Storm Studio** →
- In the SPI Storm Studio main window, click on the 'install license' button from the toolbar or select **Help > Install license**



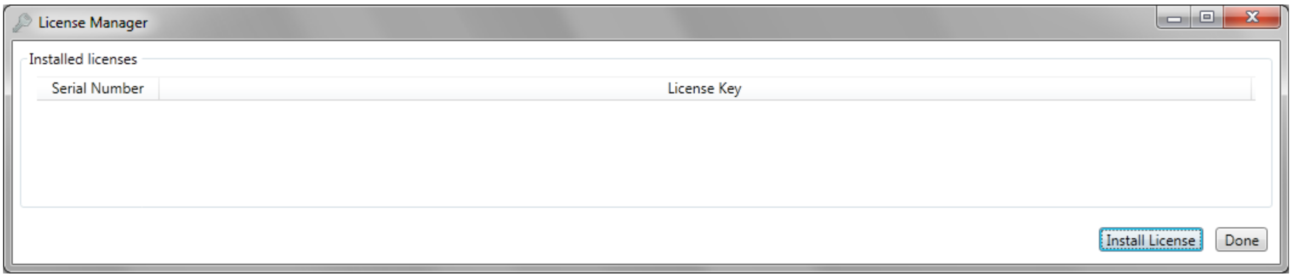
Start SPI Storm Studio from the Windows Start Menu



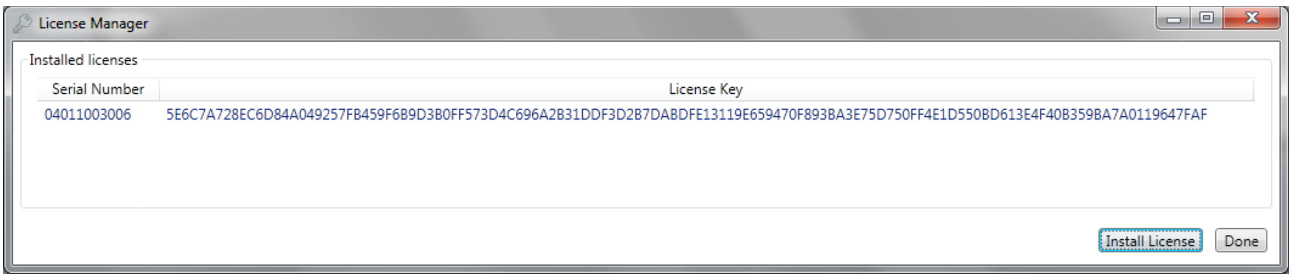


Install License from the 'Help' menu

The 'License Manager' window opens.



- Click on **'Install License'** button. A browser window opens.
- If you have not received your license file, please go to http://www.byteparadigm.com/download-16.html and follow the instructions about how to receive your license file.
- Select the received license file and click on 'Open'.
  The License Manager now lists the installed devices and the corresponding license strings.



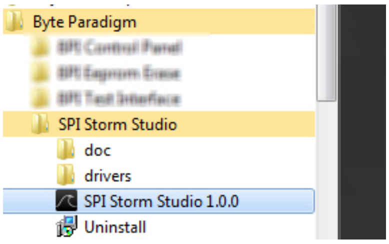
<u>Please note:</u>

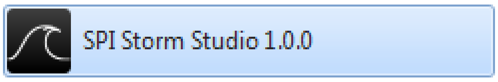
- Each installed device is designated with its serial number, in the **'Serial Number'** column of the License Manager.
- You can find the device serial number written on a label at the back of your SPI Storm device.
- You can install more than one device.
- When upgrading your version of SPI Storm Studio software, you need not to install the license file again.

3 Your first SPI Storm Studio project – example

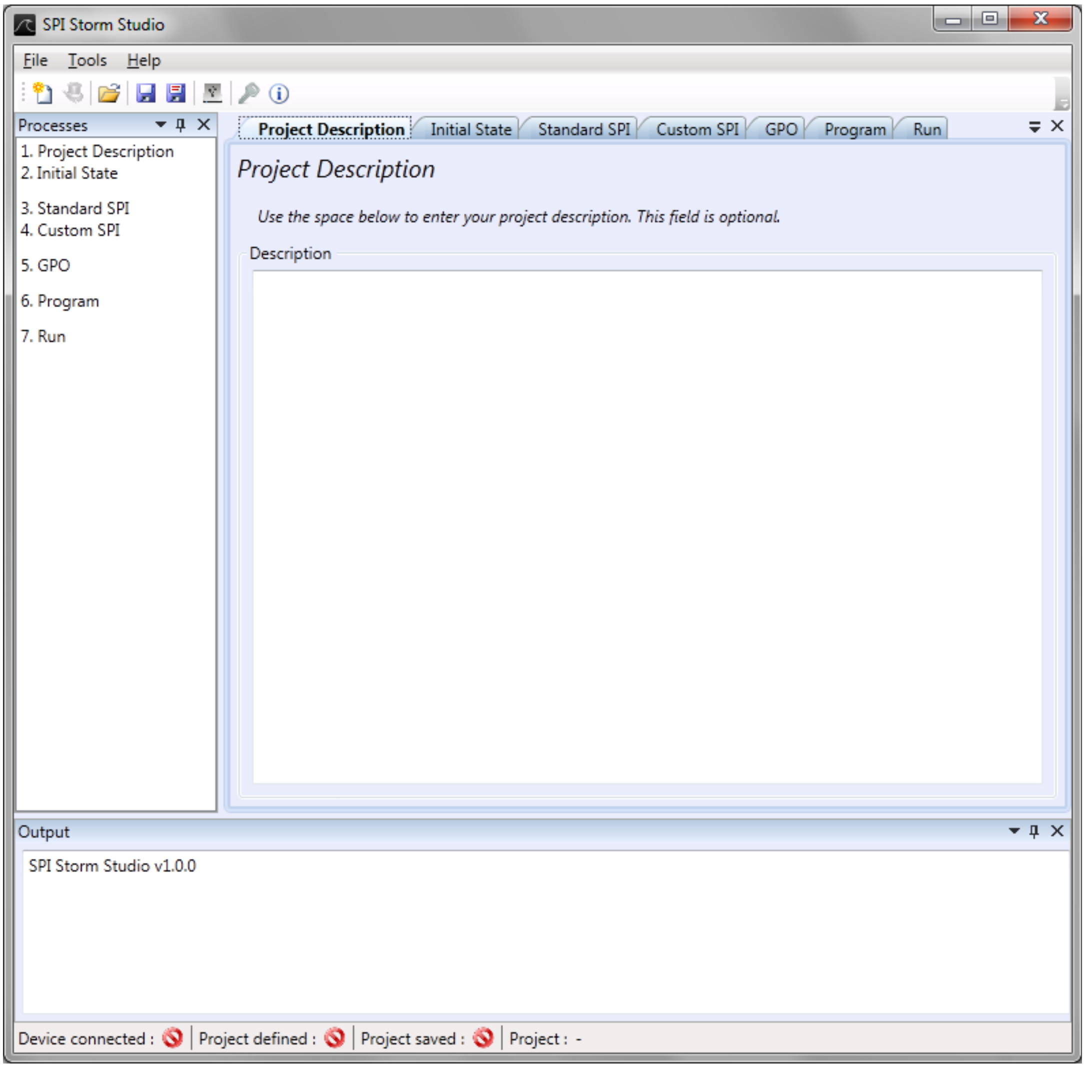
3.1 Starting the application

- Locate and click on the 'SPI Storm Studio' icon from your start menu program.



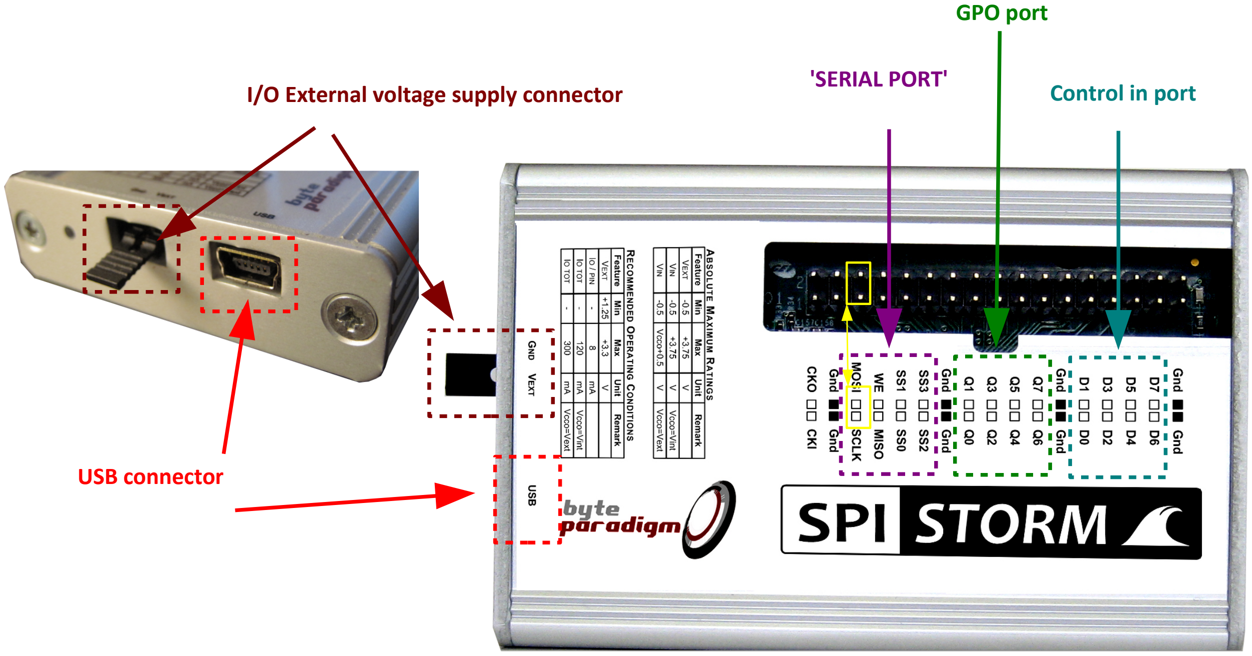


- At startup, SPI Storm Studio main window opens:



3.2 Connecting and configuring your SPI Storm device

3.2.1 SPI Storm Device at a glance

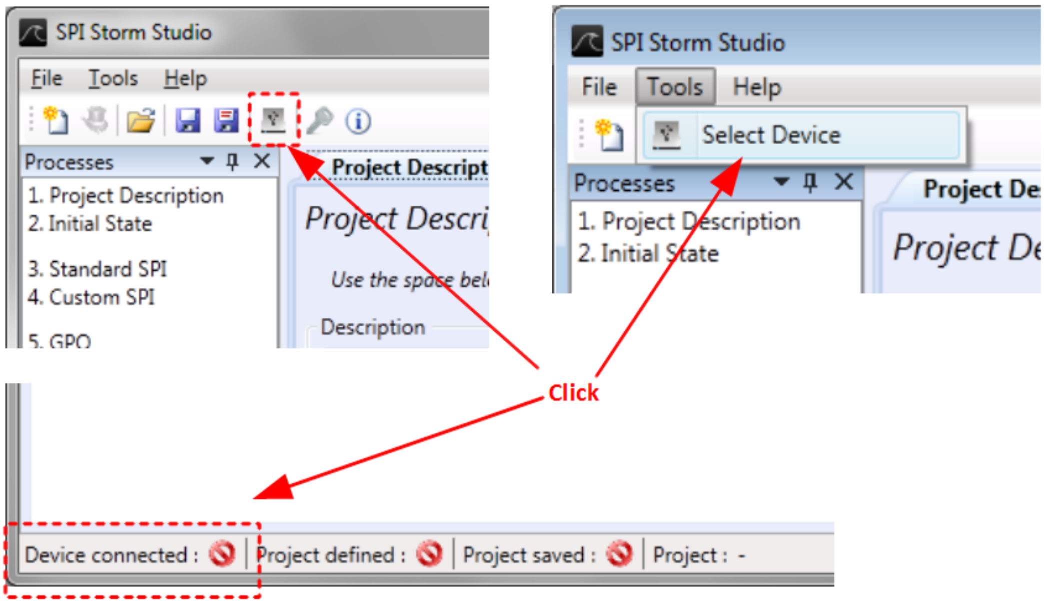


There are 3 main parts to the SPI Storm device connector:

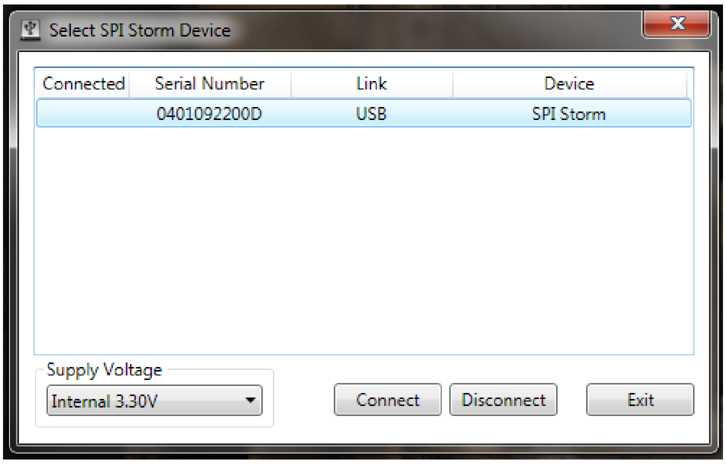
Port name	Number of I/O pins	I/O names	Purpose
SERIAL PORT	8, with 4 multi-purpose I/Os.	SCLK MOSI (DQ0) MISO (DQ1) WE SS0 SS1 SS2 (DQ2) SS3 (DQ3)	Flexible port for the generation of digital protocols using a clock signal (SCLK), up to 4 slave select signals (SS0 … SS3), and up to 4 data lines (MOSI, MISO / DQ0... DQ3). The direction of the data lines can be defined as output, input or hi-Z. Open drain I/O is also supported. Each I/O line has its specifics and limitations. Please refer to sections below.
GPO PORT	8	Q0 … Q7	General-purpose outputs: set of outputs where arbitrary digital pattern can be generated.
Control in PORT	8	D0 … D7	Set of input pins used for triggering.
GND pins	8	GND	Ground pins – SPI Storm device and slave devices must share the same ground signals.
Special clock I/Os	2	CKO CKI	CKO mirrors an internally generated clock signal; CKI can be used as an input pin for providing an external reference clock signal.

3.2.2 Establish a connection between SPI Storm Studio and SPI Storm USB device

- Connect your SPI Storm device to a USB port of your PC with the provided USB cable.
- SPI Storm device is powered through the USB port; once connected, a blue and a red LED are on.
- In SPI Storm Studio main window, click on **'Select Device'** button or select **Tools > Select Device** from the window menu.
- Alternatively, you can click on the **'Device connected area'** in the status bar at the bottom of the main window:



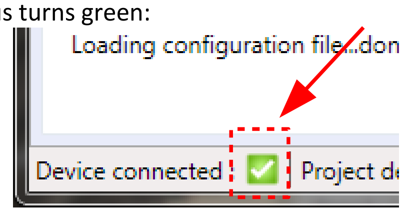
- The **'Device Selection window'** opens. The SPI Storm devices physically connected to the PC are listed. They are designated with their serial number.



- To establish a connection between one device and this session of SPI Storm Studio, select the desired device in the list and click on **'Connect'.**

  If your device is not physically connected to your PC or if the USB driver is not properly installed, the device won't be recognized and the list in the 'Select Device' Window will be empty.

– After a few seconds, the device is properly configured and ready to be used. The 'device selection window disappears and the device status turns green:



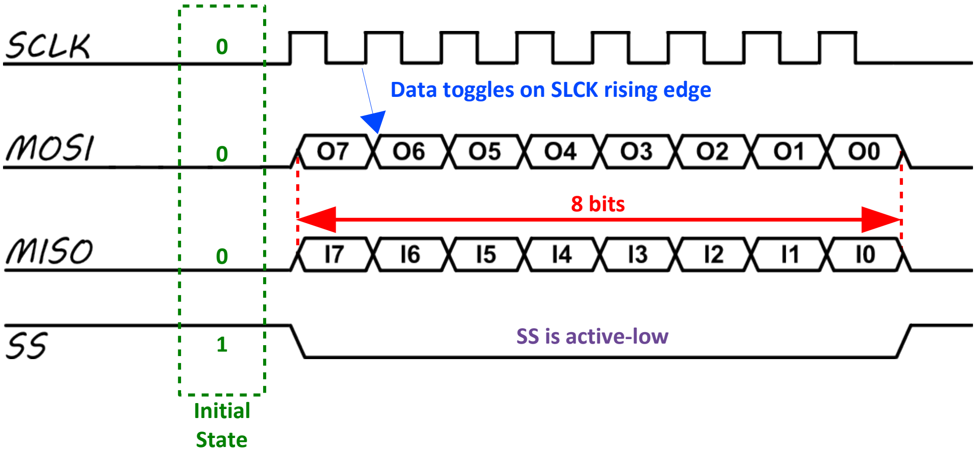
3.3 Step-by-step: how to define a protocol with SPI Storm Studio?

In this section, we'll show how to define a simple protocol with SPI Storm Studio. We'll start from a standard SPI protocol, and then show how to customize it to form a more specific serial protocol for communication.

3.3.1 The basics: defining a simple access with the standard SPI protocol:

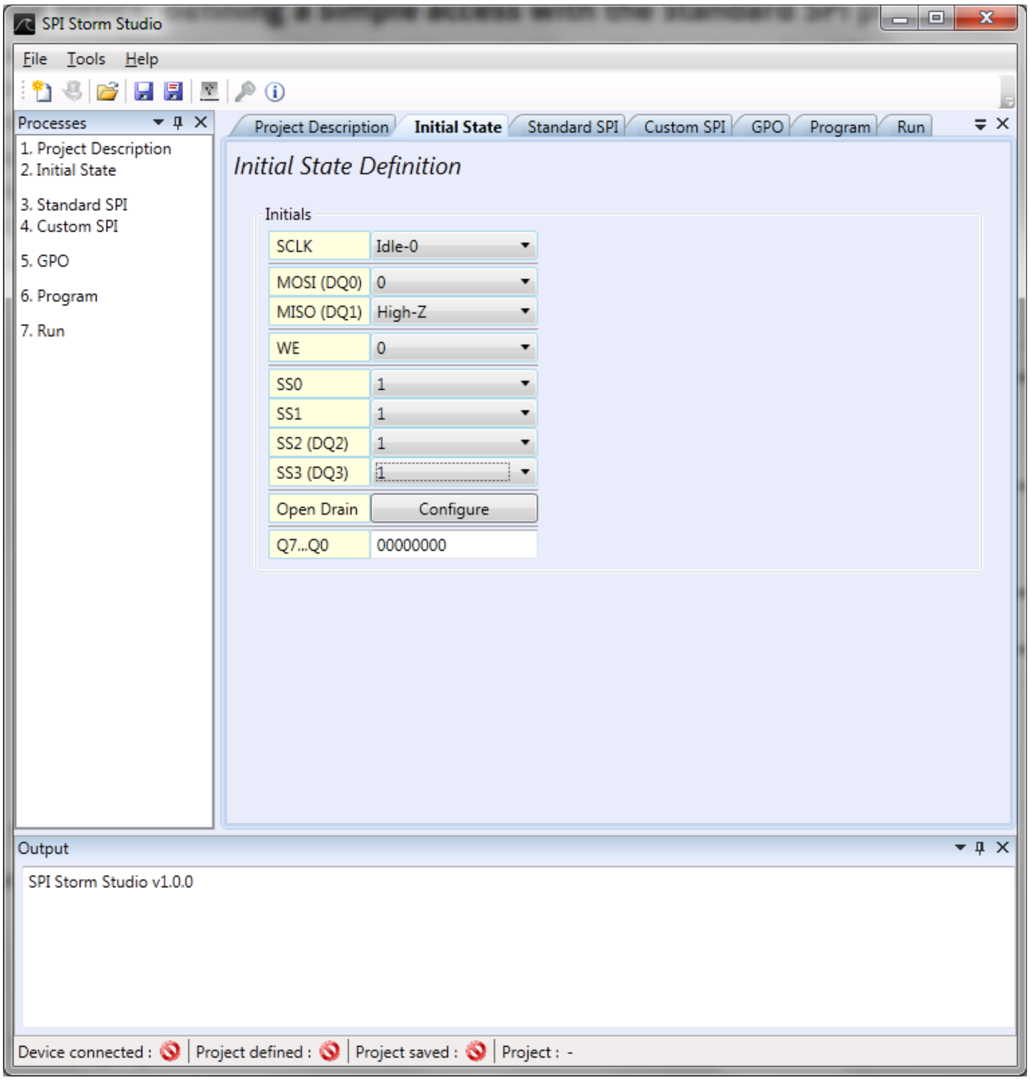
Let's get started. Here is the SPI protocol that should be used with one SPI slave that we'll call 'Device0'.
This protocol is a standard SPI protocol having the following characteristics:

– Initially, all signals lines are at low level, except SS lines, at high level.
– Access length : 8 bits (1 byte).
– Slave select signal's polarity : active low.
– SCLK IDLE level is 'low'.
– Data on MOSI / MISO are generated on SCLK **rising edge**.
– SCLK frequency is 25 MHz.



3.3.1.1 Setting up the initial state

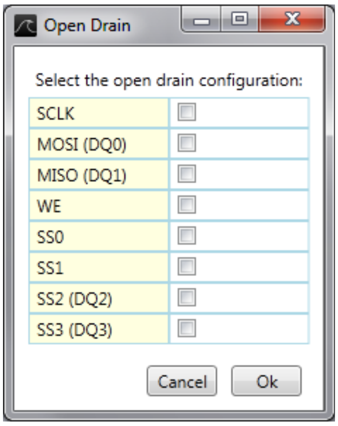
– Select the **'Initial State'** tab from the main window:



A drop-down box allows setting the initial value of each signal or group of signals.
For this example, click on the drop-down boxes and select the following values:

Signal / Group of signals / Control	**Possible values**	**Selected value**	**Comments**
SCLK	Idle-0, Idle-1, Running	Idle-0	The clock is non-running and at Idle-0 level at initial state
MOSI (DQ0)	High-Z, 0, 1	0	MOSI is the data line from the master to the slave(s). It is set at '0' initially.
MISO (DQ1)	High-Z, 0, 1	High-Z	MISO line is the data line from the slave to the master. It should not be driven in the initial state.
WE	0,1	Don't care	WE line is not used in this example. Please refer to section 5.1 below for more information about this signal.
SS0	0,1	1	We'll use SS0 as slave select line. As active-low signal, it is set to logic '1' during the initial state.
SS1	0,1	Don't care	

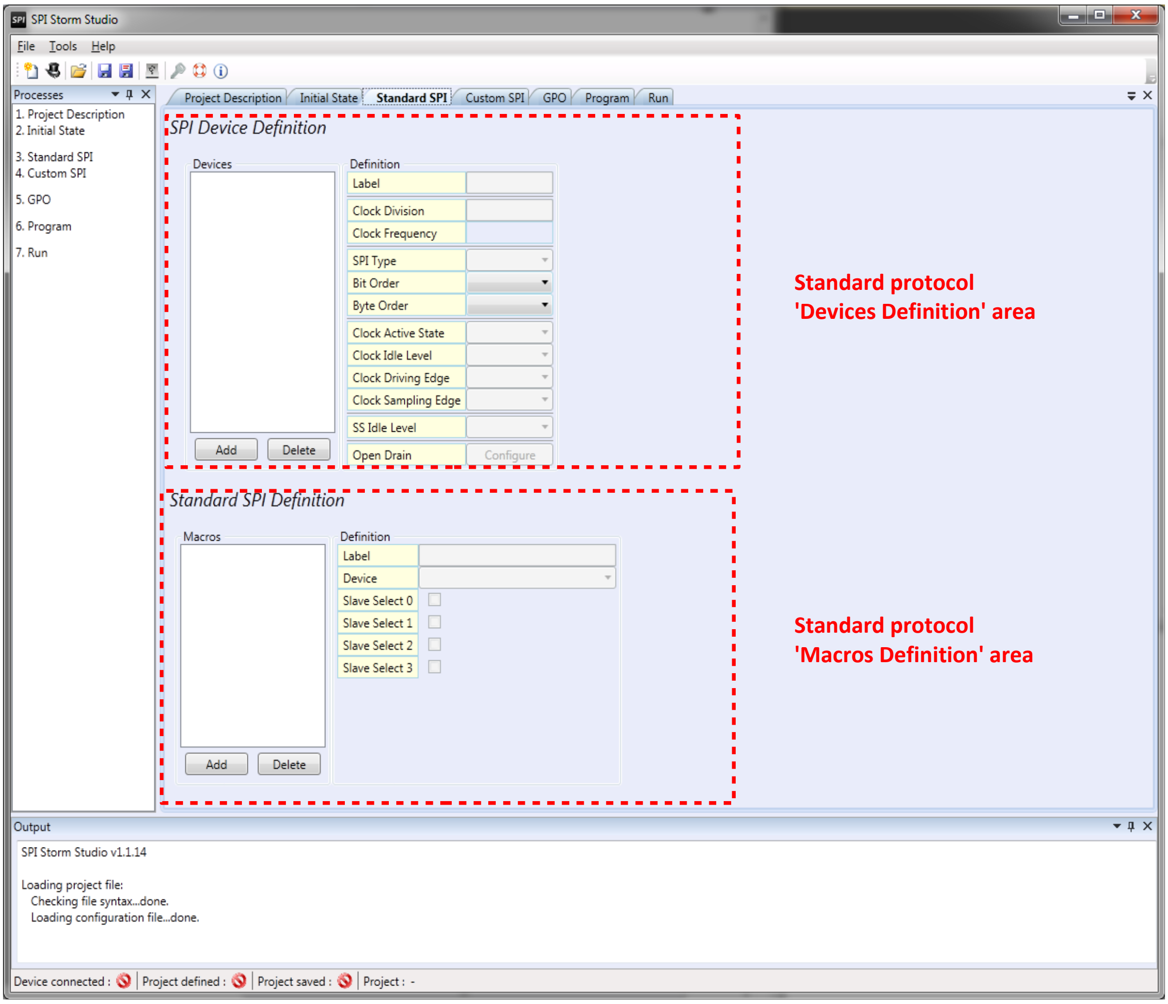
<table>
<tr><th>Signal / Group of signals / Control</th><th>Possible values</th><th>Selected value</th><th>Comments</th></tr>
<tr><td>SS2 (DQ2)</td><td>High-Z, 0, 1</td><td>Don't care</td><td rowspan="2">The other SS lines are not used in this example. Please refer to ### for more information about SS lines</td></tr>
<tr><td>SS3 (DQ3)</td><td>High-Z, 0, 1</td><td>Don't care</td></tr>
<tr><td>Open Drain</td><td colspan="3">Clicking on this button opens the 'Open Drain' controls for the I/Os during the initial phase.



A tick box is available for each signal.
☑ Checking the box sets the corresponding I/O in 'open-drain' mode.
☐ Leaving the box unchecked sets the corresponding I/O in 'normal mode' (firm LVCMOS voltage levels).</td></tr>
<tr><td>Q7... Q0</td><td>0 or 1 for each bit of the vector</td><td>Don't care</td><td>This controls the initial values of the GPO port. Please refer to ### for more details about the GPO port.</td></tr>
</table>

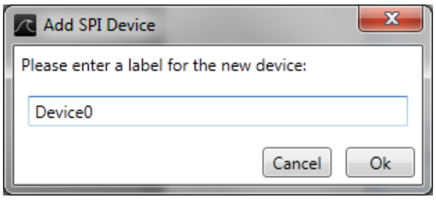
3.3.1.2 Setting up the standard SPI access

- Select the **'Standard SPI'** tab from the main window.
- There 2 main areas in this tab: **'Devices Definition'** and **'Macros Definition'.**
  - The 'Devices Definition' area defines the characteristics of the SPI protocol used for each device;
  - The 'Macros Definition' area associates each device to one of the four physical **slave select** lines of the device.

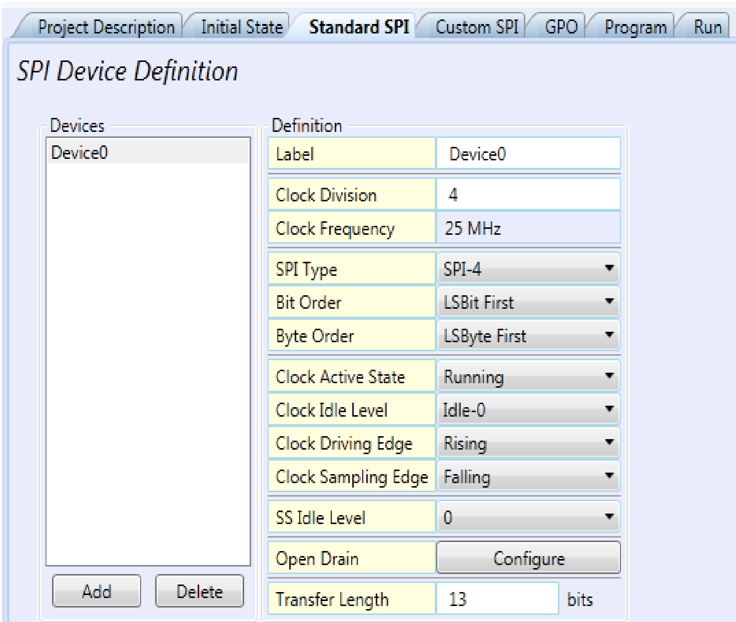


To define a device:

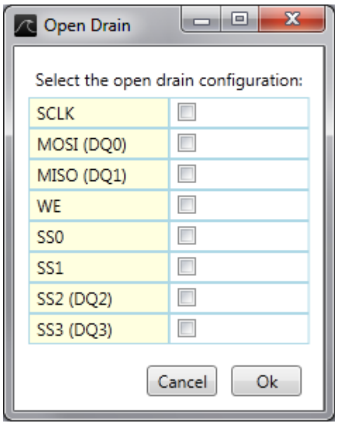
1) In the SPI Device Definition, click on **'Add' button.**
2) In the window that opens, specify a name for the device – let's use **'Device0'.** Click on **OK.**



3) Now your device is created and listed. Select Device0 in the list. You are now able to define the parameters of the access related to 'Device0'. To do so, modify the parameters contained in the text boxes and drop-down lists located to the right of the Devices list.

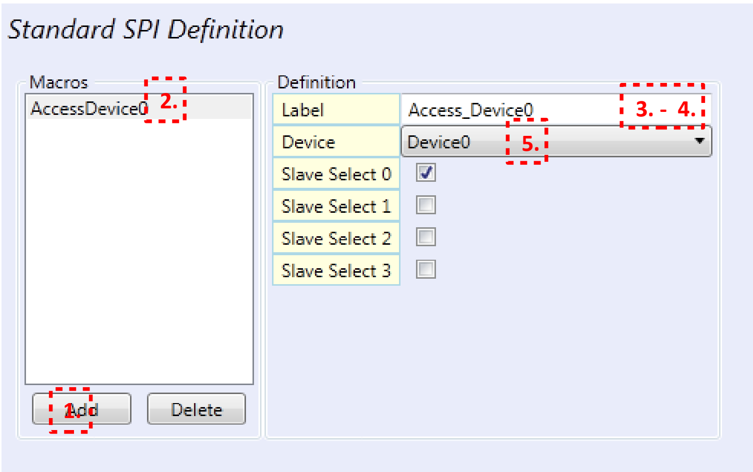


	Possible values	**Selected value**	**Comments**
Label	Any – this is a text label used to designate the access being defined	Device0	The text box can be used to change the name of the Device.
Clock Division	Any positive integer value from 1 to 1024	4	This defines the SCLK frequency as a dividing factor from a reference 100 MHz clock, according to the formula: FSCLK = 100 / (Clock Division). Defining a 'clock division' of 4 will result in a frequency equal to 100 / 4 = 25 MHz for SCLK
SPI Type	SPI-4, SPI-3, SPI-Dual, SPI-Quad	SPI-4	This drop down box lets you specify the type of standard protocol you would like to use. In this case, we are using a standard 'SPI' protocol – SPI-4
Bit Order	LSBit First, MSBit First	LSBit First	Defines the bit ordering within each byte of data.

	Possible values	Selected value	Comments
Byte Order	LSByte First, MSByte First	LSByte First	Defines the byte ordering
Clock Active State	Running, Idle-0, Idle-1	Running	This defines the behavior of SCLK while data is sent or received. In our example, SCLK toggles while data is sent on MOSI and data is received on MISO.
Clock Idle level	Running, Idle-0, Idle-1	Idle-0	Defines the level of the SCLK signal when the clock is not toggling.
Clock Driving edge	Rising, Falling	Rising	Defines on which edge of SCLK the data is generated.
Clock Sampling Edge	Rising, Falling	Falling	Defines on which edge of SCLK data is sampled.
SS Idle level	0,1	1	Defines the level of SS when it is not active.
Open Drain	Clicking on this button opens the 'Open drain' controls of the I/Os when executing accesses defined from the 'Standard SPI tab'. 	A tick box is available for each signal. ☑ Checking the box sets the corresponding I/O in 'open-drain' mode. ☐ Leaving the box unchecked sets the corresponding I/O in 'normal mode' (firm LVCMOS voltage levels).	
Transfer length	Any positive integer value > 1	8	Length of the transfer counted in bits.

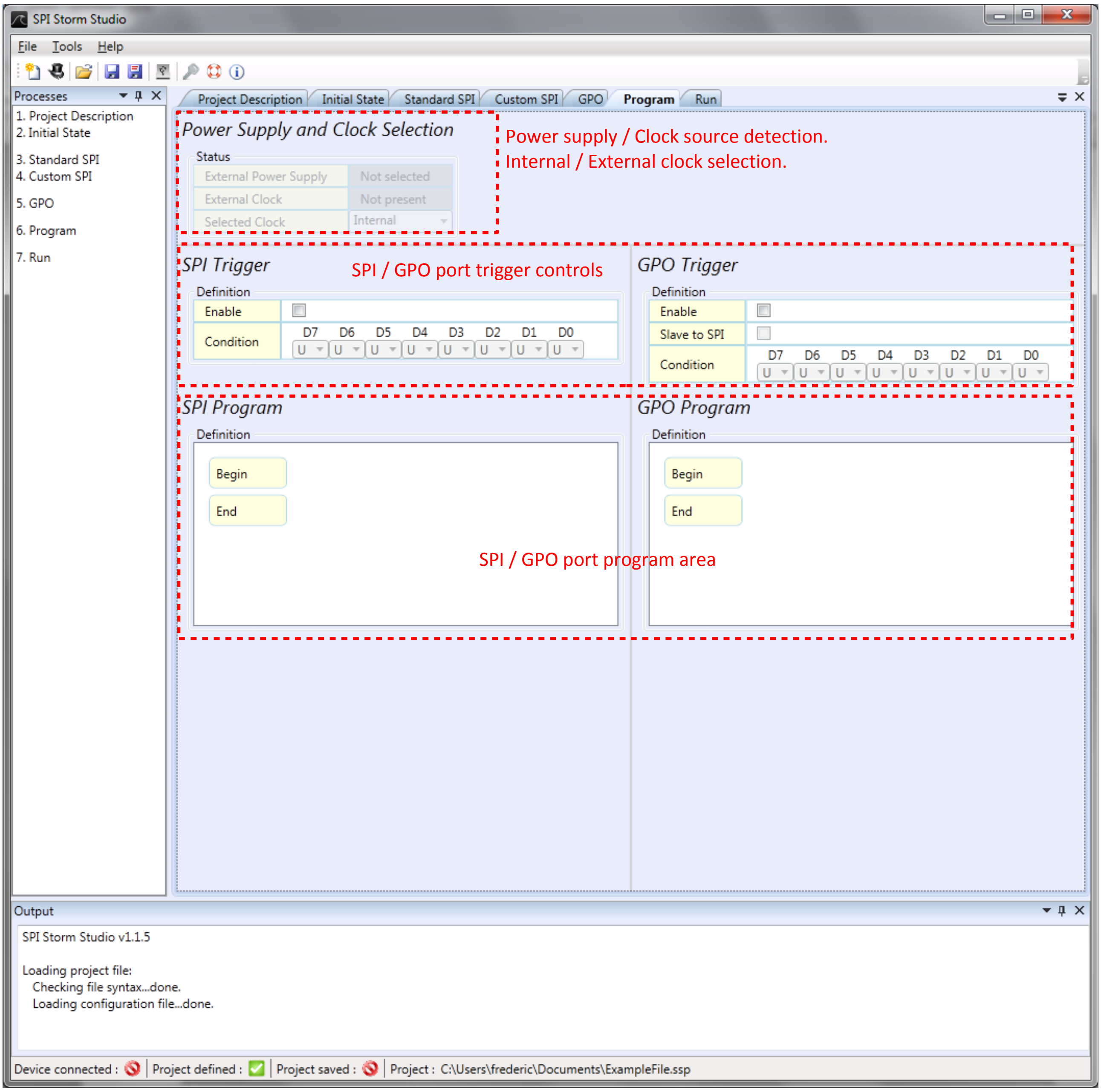
To define a macro in the 'Standard SPI' tab:

1. Click on 'Add' in the 'macros definition area'.
2. Specify a name for the macro being defined: example: 'AccessDevice0'
3. Select which 'device' you wish to associate with this macro – click on the drop-down list next to 'Device'. This list contains all the defined devices.
4. Select 'Device0'.
5. Associate a slave select line to this access by selecting the corresponding tick box.



3.3.1.3 Setting up a simple program that uses the configured macros

Switch to 'Program Tab'.

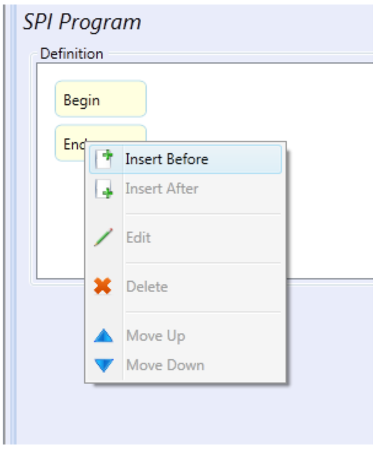


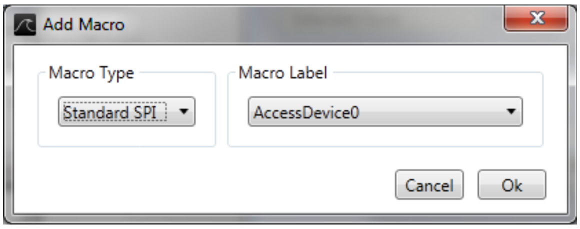
We'll now edit the 'SPI Program' to run a simple access with the SPI port.

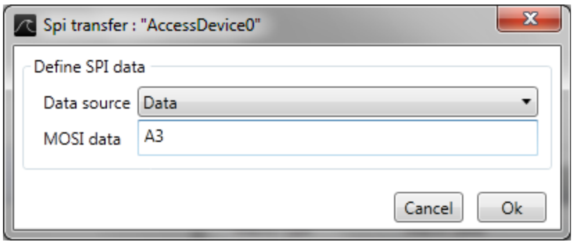
> Right-click on the 'End clause' in the 'SPI Program' area

Select **'Insert Before'** from the contextual menu that pops up. A window opens.

In the **'Add macro'** window, use the drop-down lists to select: 'Standard SPI' in the 'Macro type' and 'AccessDevice0' in the Macro Label.
Then click on **'OK'**.

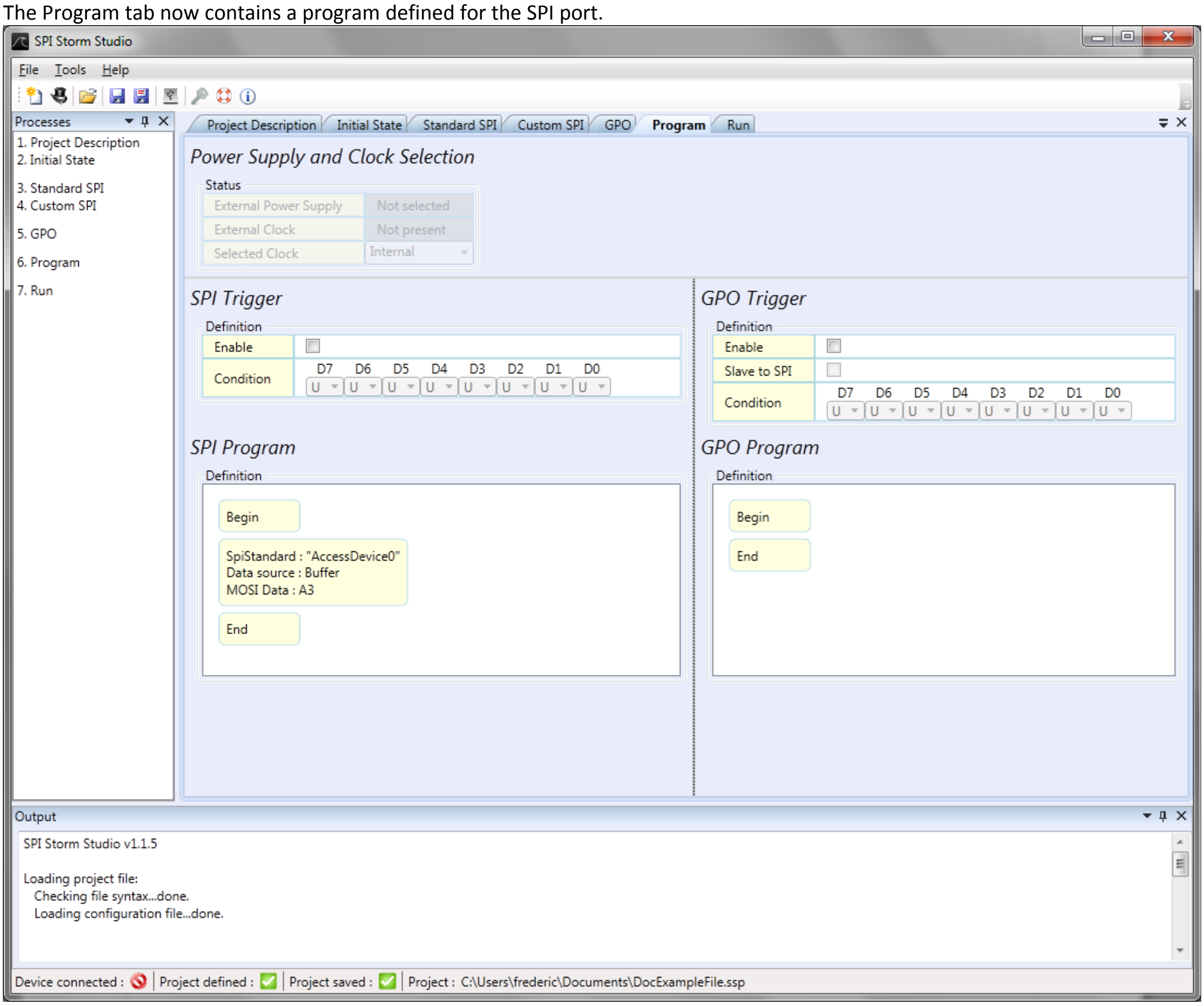






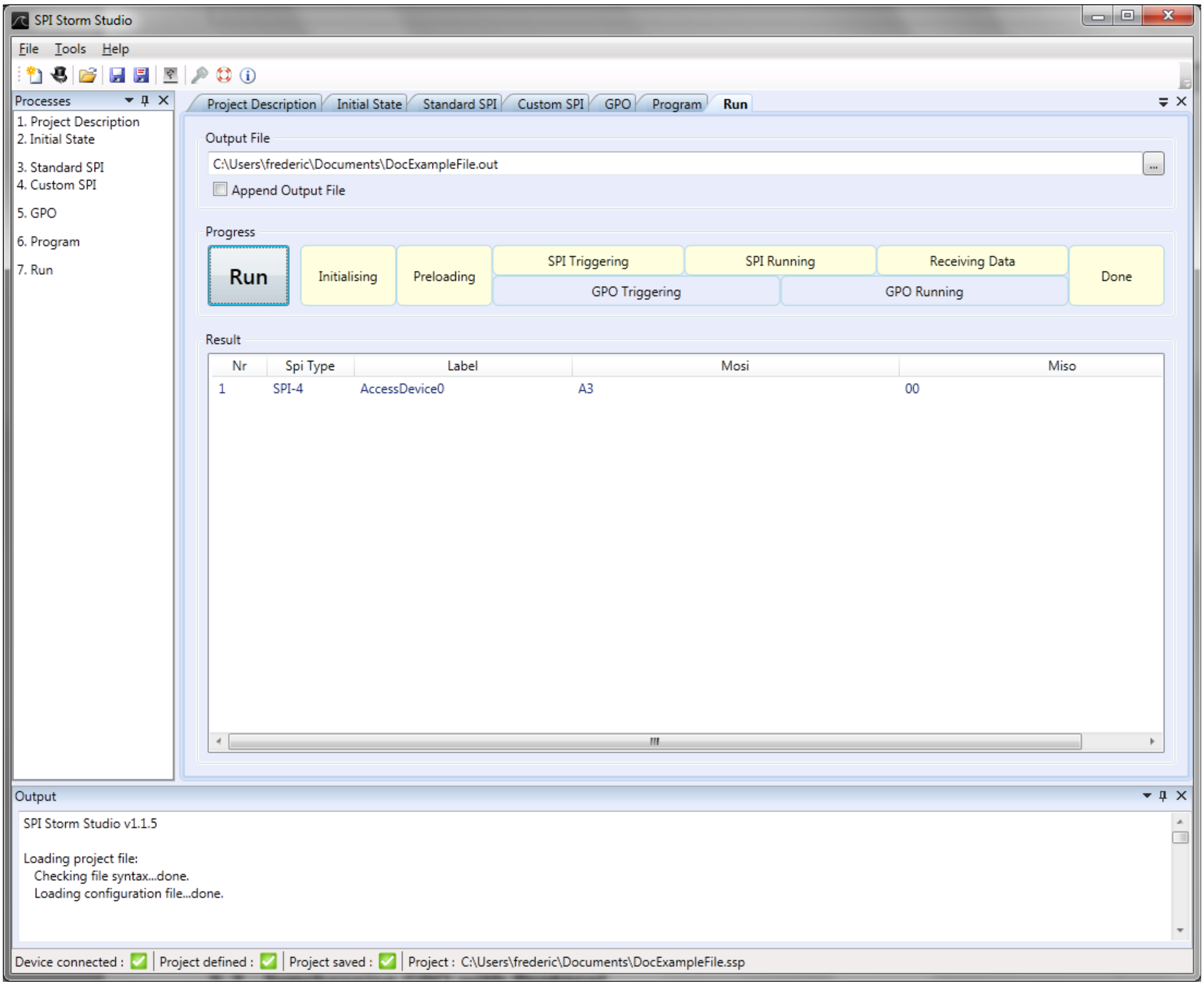
The window that opens allows defining the data sent on MOSI for this access. Type a value in hexadecimal (without '0x' prefix) – here: A3. Click on 'OK'.

The Program tab now contains a program defined for the SPI port.



3.3.1.4 Running the program

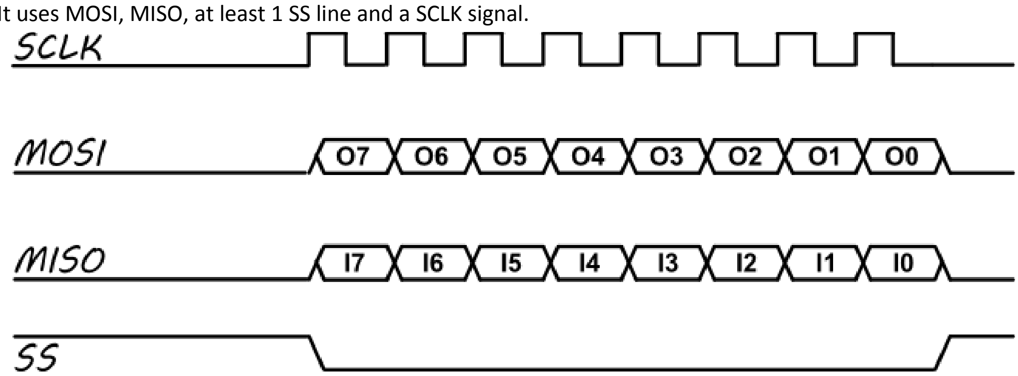
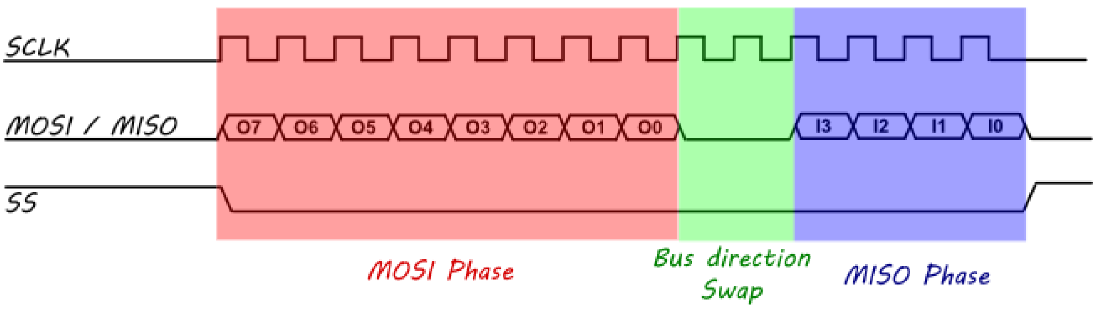
Switch to the 'Run' tab.
Connect your device (see section 3,2,2) and click on 'RUN'.

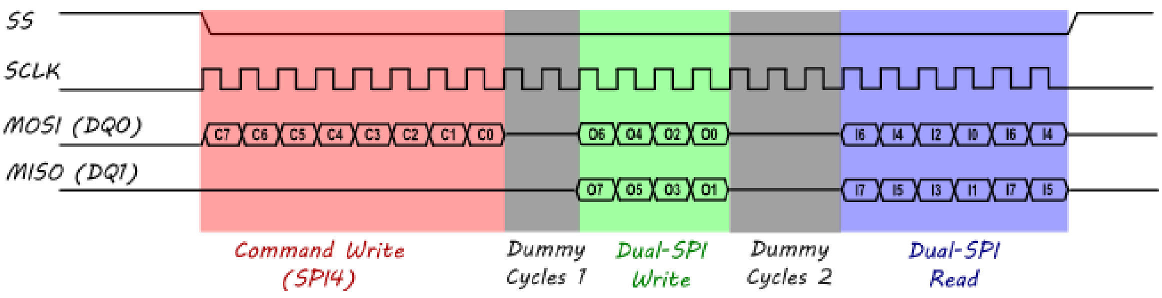
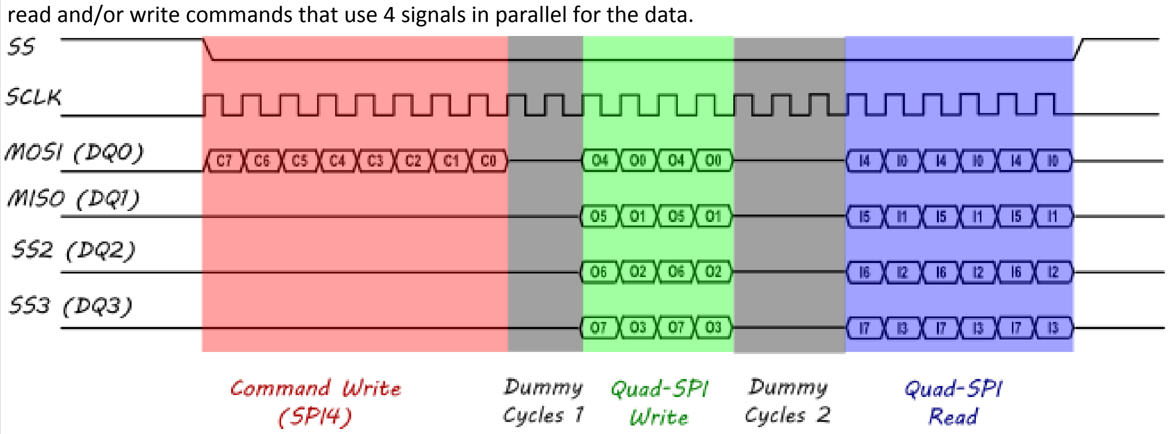


4 Defining Standard SPI protocol

4.1 Standard protocols in SPI Storm Studio

The 'standard protocols' in SPI Storm Studio are summarized in the table below.

Protocol	Description
SPI-4	This is the standard SPI protocol on 4 wires. It uses MOSI, MISO, at least 1 SS line and a SCLK signal.  SPI is a master-slave protocol simultaneously sends data onto the MOSI data line and samples data from the MISO line. 4 modes exist, according to the clock IDLE state and the clock phase relative to the data. SPI defines one SS (slave select) line per slave. The master also sends its own clock signal SCLK. Usually, SS is active-low but another convention can be used too. Usually, SCLK is held IDLE between transfers but a continuously toggling SCLK signal can be generated by SPI Storm too.
SPI-3	This is a variant of the SPI protocol that uses 3 wires only. It uses MOSI, at least 1 SS line and a SCLK signal.  SPI-3 uses one single data line for writing (data out) or reading data (data in). This allows saving on slave I/Os count. SPI-3 protocol is composed of 3 phases: - 'MOSI phase' (Master-Out-Slave-In), during which the DQ0 data line of SPI Storm is used as an output of the device. - 'Bus direction Swap' phase – an arbitrary number of clock cycles during which the data line is held at HI-Z (high impedance state). This phase is necessary for switching bus direction and avoid shortcuts. - 'MISO phase' (Master-In-Slave-Out), during which the DQ0 data line of SPI Storm is used as an input. During this phase, the slave answers to the master and the data that it sends is sampled by SPI Storm.

Protocol	Description
Dual-SPI	This is the dual-SPI protocol. It is a mixed protocol optionally initiated with a standard SPI (SPI-4) access followed by read and/or write commands that use 2 signals in parallel for the data.  The Dual-SPI protocol command defined in SPI Storm Studio is generic. According to the context and the slave, some of the phases described in the picture above must not be sent. To skip one of the phases above, its length has to be set to 0.
Quad-SPI	This is the quad-SPI protocol. It is a mixed protocol optionally initiated with a standard SPI (SPI-3) access followed by read and/or write commands that use 4 signals in parallel for the data. 

4.2 Data formats

Hexadecimal string without prefix representing the bits sent on / sampled from the data lines

Default format: most significant bit first / MS byte first.

Complete bytes must always be entered. (from SPI Storm Studio version 1.1.14)

Example: if 12 bits 0x123 must be used, it must be padded by one character.

Examples:
Convention : MS Byte first / MS bit first
Length: 12 bit – data : 5A30

Chronology	**1**	**2**
Byte level	0x5A	0x30
Bit level	**0-1-0-1-1-0-1-0**	**0-0-1-1**

Length: 17 bit – data : 305AF0

Chronology	**1**	**2**	**3**
Byte level	0x30	0x5A	0xF0
Bit level	0-0-1-1-0-0-0-0	0-1-0-1-1-0-1-0	1 (1-1-1-0-0-0-0 is not sent)

Examples:
Convention: LS Byte first / ls bit first
Length: 12 bit – data : 05A3

Chronology	**1**	**2**
Byte level	0xA3	0x05
Bit level	**1-1-0-0-0-1-0-1**	**1-0-1-0**

Length: 17 bit – data : 0305A3

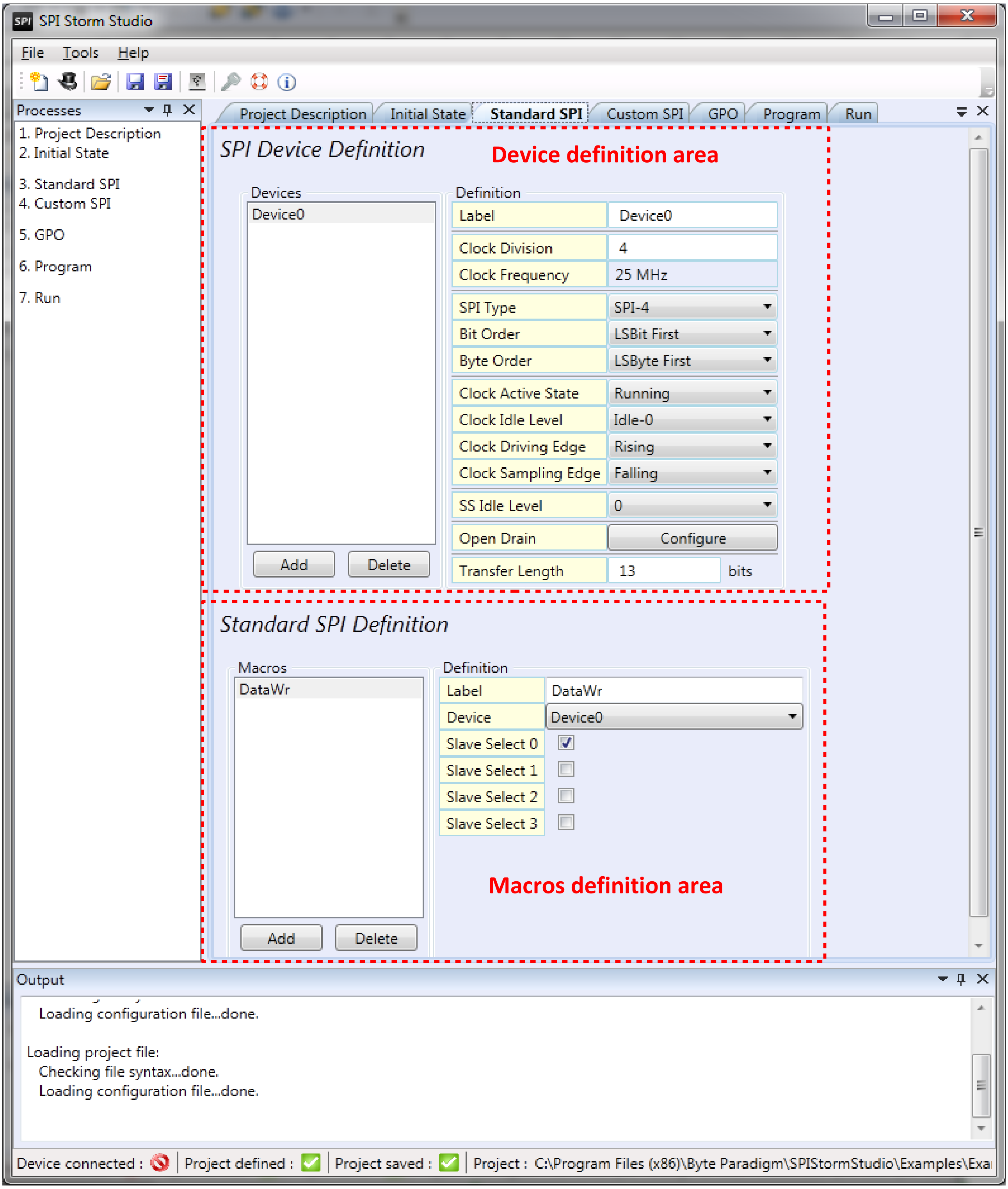
Chronology	**1**	**2**	**3**
Byte level	0xA3	0x05	0x03
Bit level	1-1-0-0-0-1-0-1	1-0-1-0-0-0-0-0	1

Other more advanced combinations can be used, however less intuitive.
Protocols using more than one data line (example: dual-SPI, quad-SPI) use an interlaced convention. Please refer to figures at section 4.1.

4.3 Using standard protocols in SPI Storm Studio

4.3.1 Overview

The 'Standard SPI tab' is used for setting up SPI Storm device to run with standard protocols.



To set up a standard protocol:

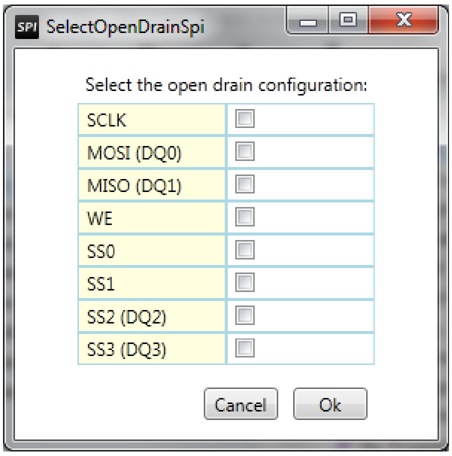
- One or multiple 'devices' must be defined. A 'device' specifies the characteristics of the protocol.
- One or multiple 'macros' must be defined. A 'standard protocol macro' associates a device to a specific slave select I/O.

4.3.2 Defining a standard protocol 'device'

To create a device, click on 'Add' in the 'Device definition area'. A window pops up, prompting for the device name.

The table below summarizes the parameters to be specified for each SPI Storm Studio standard protocol.

Parameter	Applies to	Valid values	Description
Label	ALL	String composed of letters and numerical characters	Device name
Clock Division	ALL	Integer value from 1 to 1024	Clock dividing factor. Defines the clock rate used by the device. The resulting clock rate is the RefClock frequency / Clock Division. If the internal clock is used, the RefClock frequency is 100 MHz.
Clock Frequency	ALL	Non editable	Displays the clock rate resulting from the specified division factor.
SPI Type	ALL	SPI-4, SPI-3, SPI-Dual, SPI-Quad	Allows choosing a standard protocol for the device
Bit Order	ALL	LSBit First, MSBit First	Defines the bit ordering within each byte of data.
Byte Order	ALL	LSByte First, MSByte First	Defines the byte ordering
Clock Active State	ALL	Running, Idle-0, Idle-1	Defines the state of the SCLK signal while sending / receiving data with the given protocol. Running = the clock toggles. Idle-0 = the clock is held at constant low logic level; Idle-1 = the clock is held at constant high logic level.
Clock Idle Level	ALL	Running, Idle-0, Idle-1	Defines the state of the SCLK signal while not sending / receiving data with the given protocol. Running = the clock toggles. Idle-0 = the clock is held at constant low logic level; Idle-1 = the clock is held at constant high logic level.
Clock Driving Edge	ALL	Rising, Falling	Defines the phase of the SCLK clock signal relative to the generated data. Rising = data is sent out with the SCLK signal rising edge Falling = data is sent out with the SCLK signal falling edge
Clock Sampling Edge	ALL	Rising; Falling	Defines the edge of the clock used for sampling

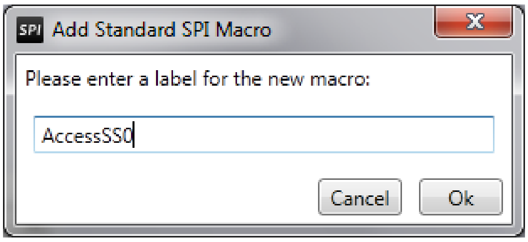
Parameter	Applies to	Valid values	Description
			data in. Rising = data is sampled with the clock rising edge Falling = data I sampled with the clock falling edge
SS Idle level	ALL	0, 1	Defines the IDLE level of the SS line used for this device. 0 = SS is at low logic level when IDLE 1 = SS is at high logic level when IDLE
Open Drain	ALL	Click on the 'Configure button' to set I/Os open-drain controls	 Checking a tick box in this window sets the corresponding I/O in open-drain.
Transfer Length	SPI-4	Integer value from 0 to 1.000.000	Defines the length in bits (clock cycles) of the SPI transfer.
Write Length	SPI-3	Integer value from 0 to 1.000.000	Defines the length in bits (clock cycles) of the write phase (MOSI phase) of the SPI-3 transfer.
Read Length	SPI-3	Integer value from 0 to 1.000.000	Defines the length in bits (clock cycles) of the read phase (MISO phase) of the SPI-3 transfer.
Swap latency	SPI-3	Integer value from 0 to 1.000.000	Defines the length in bits (clock cycles) of the swap latency phase of the SPI-3 transfer.
Swap Clock	SPI-3	Disabled, Enabled	Specifies whether the clock signal on SCLK should be active or IDLE during the swap latency phase of the SPI-3 transfer
Swap Level	SPI-3	High-Z, 0, 1	Defines the level of the MOSI (DQ0) data line during the swap latency phase of the SPI-3 transfer.
Command Write Length	SPI-Dual, SPI-Quad	Integer value from 0 to 1.000.000	Defines the length in bits of the 'Command write phase' of Dual- and Quad-SPI protocols transfers.
Dual Dummy 1 Length	SPI-Dual	Integer value from 0 to 1.000.000	Defines the length clock cycles of the 'Dual Dummy 1 phase' of Dual-SPI protocol transfers.
Dual Write Length	SPI-Dual	Integer value from 0 to 1.000.000 and multiple of 2	Defines the length in bits of the 'Dual Write' of Dual-SPI protocol transfers. This is the total number of bit interlaced onto the 2 data lines. This number must be a multiple of 2.
Dual Dummy 2 Length	SPI-Dual	Integer value from 0 to 1.000.000	Defines the length in clock cycles of the 'Dual

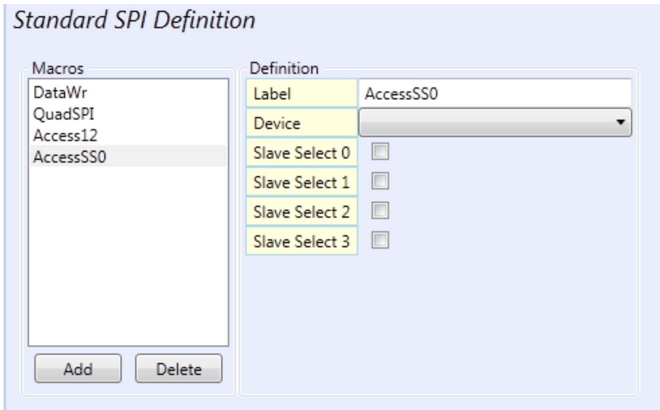
Parameter	Applies to	Valid values	Description
			Dummy 2 phase' of Dual-SPI protocol transfers.
Dual Read Length	SPI-Dual	Integer value from 0 to 1.000.000 and multiple of 2	Defines the length in bits of the 'Dual Read' of Dual-SPI protocol transfers. This is the total number of bit interlaced onto the 2 data lines. This number must be a multiple of 2.
Quad Dummy 1 Length	SPI-Quad	Integer value from 0 to 1.000.000	Defines the length clock cycles of the 'Quad Dummy 1 phase' of Quad-SPI protocol transfers.
Quad Write Length	SPI-Dual	Integer value from 0 to 1.000.000 and multiple of 4	Defines the length in bits of the 'Quad Write' of Quad-SPI protocol transfers. This is the total number of bit interlaced onto the 4data lines. This number must be a multiple of 4.
Quad Dummy 2 Length	SPI-Dual	Integer value from 0 to 1.000.000	Defines the length clock cycles of the 'Quad Dummy 2 phase' of Quad-SPI protocol transfers.
Quad Read Length	SPI-Dual	Integer value from 0 to 1.000.000 and multiple of 4	Defines the length in bits of the 'Quad Read' of Quad-SPI protocol transfers. This is the total number of bit interlaced onto the 4data lines. This number must be a multiple of 4.

4.3.3 Defining a standard protocol 'macro'

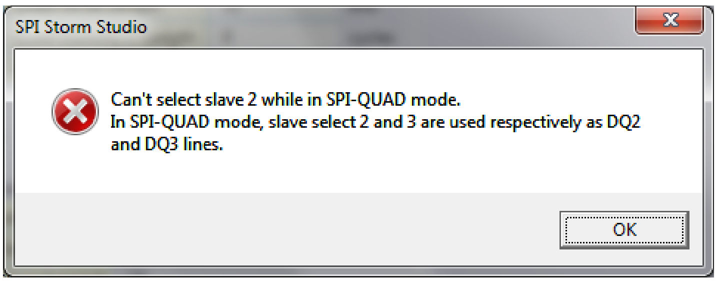
'Macros' are what is executed by SPI Storm Studio's program.

To create a new macro, click on 'Add button' and enter its name in the prompt that pops up.



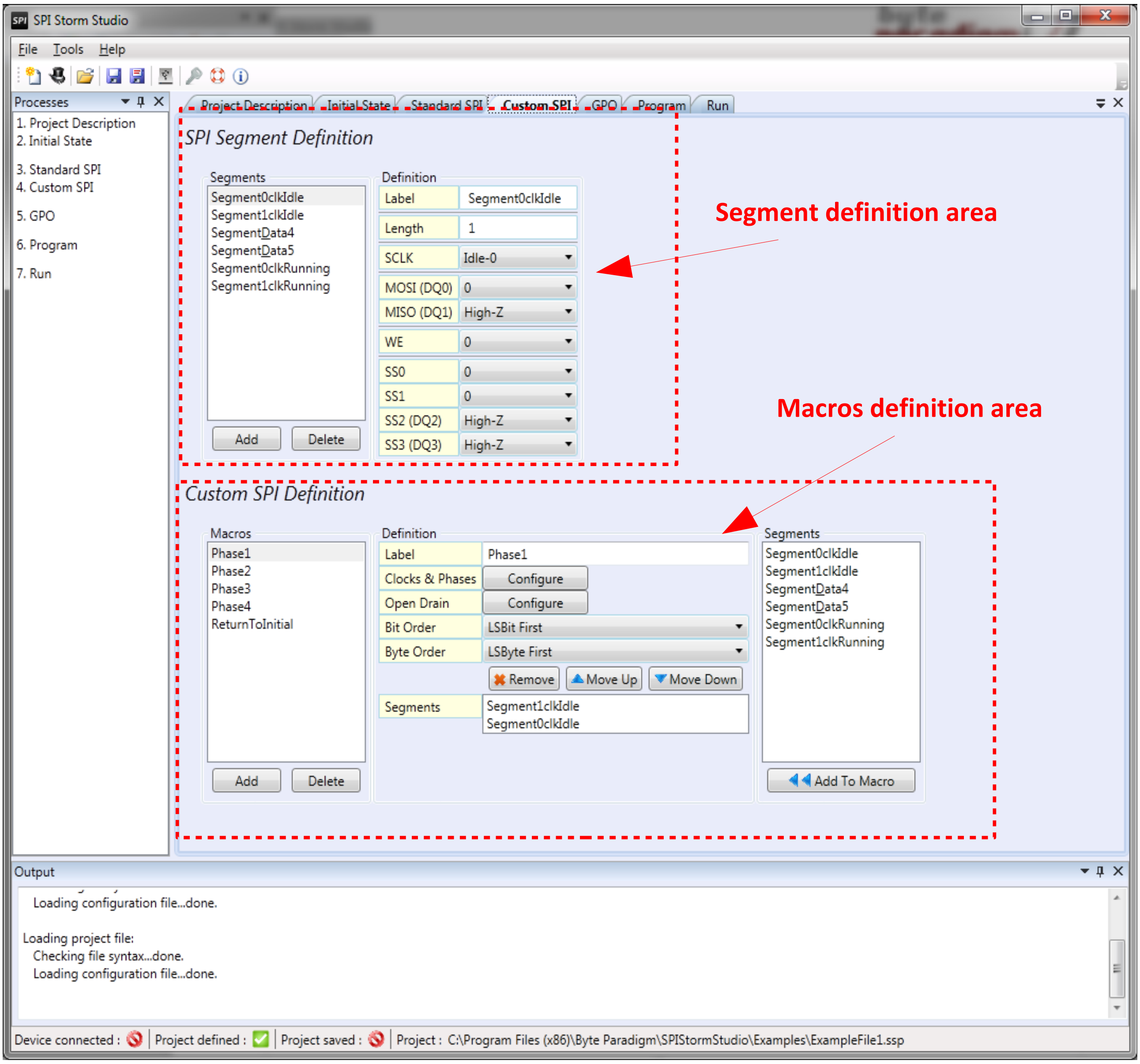


- Then, select a device from the 'Device drop-down list'. This associates a device to the macro being defined.
- Finally, select one or multiple slave select signals that have to be activated when executing this macro. Given slave select lines are multi-purpose I/O on SPI Storm, not all choices are available for all protocols. For instance, SS2 cannot be selected if the associated device uses SPI-Quad protocol: this line is automatically used as data line. If you need more control lines with such protocols, please use GPO lines. An example or error message is shown below.



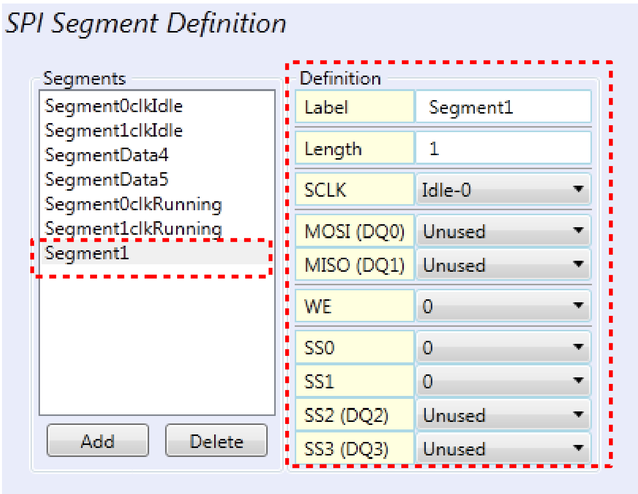
5 Defining custom serial protocol

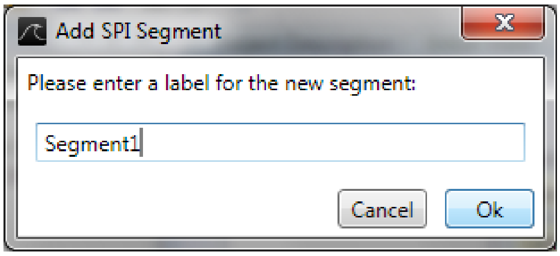
'Custom protocols' are defined from the 'Custom SPI tab'.
They are based on the definition of 'segments' and 'macros'. **'Segments'** are the **building blocks of macros.**
One or multiple segments are assembled to form a macro.



5.1 Defining a custom segment

To define a new segment, click on 'Add' button in the 'SPI Segment Definition' area. A window pops up, prompting for the segment name.





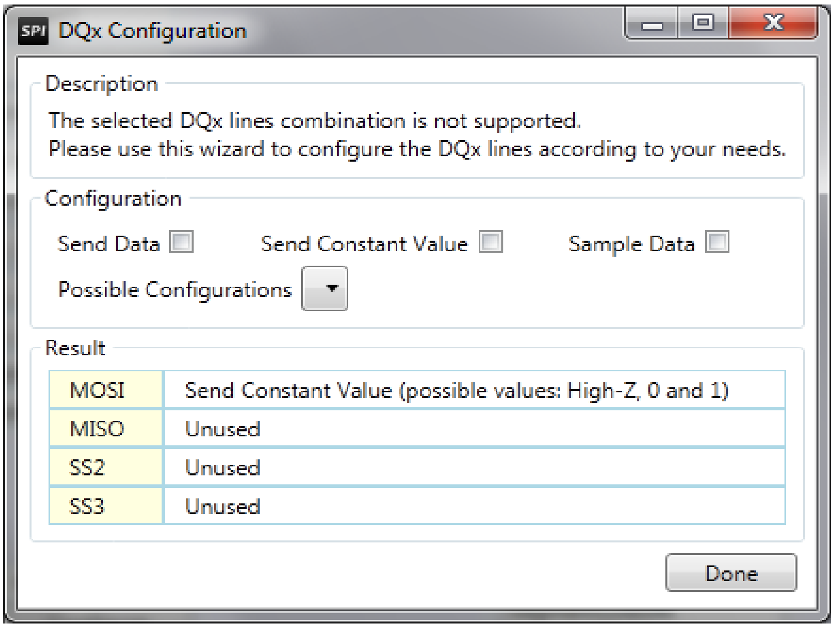
When the segment's name is highlighted in the 'SPI Segment Definition' list, its properties can be read and edited in the 'Definition area'. The following properties can be defined:

	Possible values	**Description**
Label	Any – this is a text label used to designate the segment's name	The text box can be used to change the name of the segment.
Length	1 to 500	This is the length of the segment in clock cycles. The clock period / frequency is defined in the 'macros area'. So, the same segment can be used at different clock rates.
SCLK	Running, Idle-0, Idle-1	Defines the behavior of the SCLK signal while the segment is executed. 'Running' means that SCLK will be toggling during segment execution; Idle-0 and Idle-1 will hold SCLK at low and high logic levels respectively.
MOSI (DQ0)	Unused, High-Z, 0, 1, Data, Sample	Defines the behavior of the corresponding signal, according to the list of possible values: Unused = unused High-Z = held high-impedance during segment execution. 0 = held at low logic level during segment execution 1 = held at high logic level during segment execution. Data = used generate data out. Sample = used to sample data in. **IMPORTANT:** **Not all arbitrary combinations are possible. See section '4.2 Rules for segment definition'**
ISO (DQ1)	Unused, High-Z, 0, 1, Data, Sample	
SS0	0, 1	
SS1	0, 1	
SS2 (DQ2)	Unused, High-Z, 0, 1, Data, Sample	
SS3 (DQ3)	Unused, High-Z, 0, 1, Data, Sample	
WE	0, 1	0 = active-low; 1 = active-high. WE will be made active at the above level if a data 'write' operation is executed on bidir data bus. Otherwise, remains at the defined default value.

5.2 Rules for custom segment definition

Not all combinations of values on the 'data lines' of the device are possible.
The 'data lines of the device' are MOSI (DQ0), MISO (DQ1), SS2(DQ2) and SS3 (DQ3).

When an illegal combination of the data lines is attempted, the following window pops up, enabling the selection of a valid combination:



The table below shows the list of possible combinations.

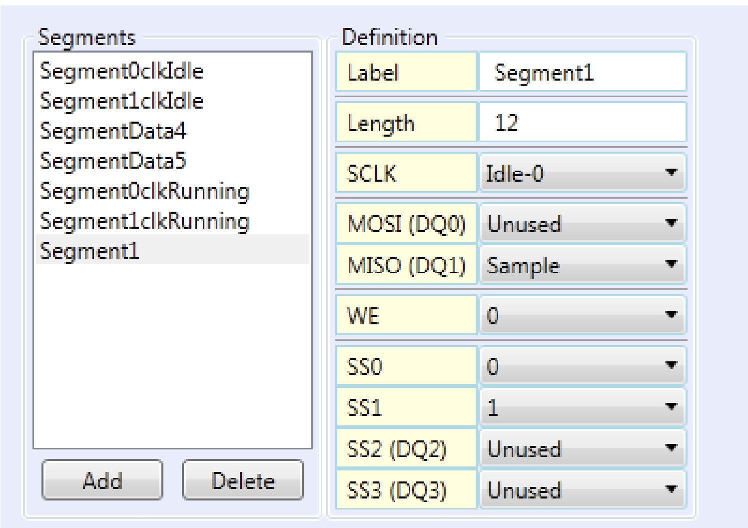
Combination name	Description
Unused	MOSI: Unused MISO: Unused SS2: Unused SS3: Unused
SPI-Wr	MOSI: Data MISO: Unused SS2: Unused SS3: Unused
SPI-Wr-SS23	MOSI: Data MISO: Unused SS2: Send Constant Value (possible values: High-Z, 0 and 1) SS3: Send Constant Value (possible values: High-Z, 0 and 1)
SPI-WrRd	MOSI: Data MISO: Sample SS2: Unused SS3: Unused

<table>
<tr><th>Combination name</th><th>Description</th></tr>
<tr><td>SPI-WrRd-SS23</td><td><table><tr><td>MOSI</td><td>Data</td></tr><tr><td>MISO</td><td>Sample</td></tr><tr><td>SS2</td><td>Send Constant Value (possible values: High-Z, 0 and 1)</td></tr><tr><td>SS3</td><td>Send Constant Value (possible values: High-Z, 0 and 1)</td></tr></table></td></tr>
<tr><td>SPI-Rd3</td><td><table><tr><td>MOSI</td><td>Sample</td></tr><tr><td>MISO</td><td>Unused</td></tr><tr><td>SS2</td><td>Unused</td></tr><tr><td>SS3</td><td>Unused</td></tr></table></td></tr>
<tr><td>SPI-Rd3-SS23</td><td><table><tr><td>MOSI</td><td>Sample</td></tr><tr><td>MISO</td><td>Unused</td></tr><tr><td>SS2</td><td>Send Constant Value (possible values: High-Z, 0 and 1)</td></tr><tr><td>SS3</td><td>Send Constant Value (possible values: High-Z, 0 and 1)</td></tr></table></td></tr>
<tr><td>SPI-Rd4</td><td><table><tr><td>MOSI</td><td>Unused</td></tr><tr><td>MISO</td><td>Sample</td></tr><tr><td>SS2</td><td>Unused</td></tr><tr><td>SS3</td><td>Unused</td></tr></table></td></tr>
<tr><td>SPI-Rd4-SS23</td><td><table><tr><td>MOSI</td><td>Unused</td></tr><tr><td>MISO</td><td>Sample</td></tr><tr><td>SS2</td><td>Send Constant Value (possible values: High-Z, 0 and 1)</td></tr><tr><td>SS3</td><td>Send Constant Value (possible values: High-Z, 0 and 1)</td></tr></table></td></tr>
<tr><td>SPI-WrD</td><td><table><tr><td>MOSI</td><td>Data</td></tr><tr><td>MISO</td><td>Data</td></tr><tr><td>SS2</td><td>Unused</td></tr><tr><td>SS3</td><td>Unused</td></tr></table></td></tr>
<tr><td>SPI-WrD-SS23</td><td><table><tr><td>MOSI</td><td>Data</td></tr><tr><td>MISO</td><td>Data</td></tr><tr><td>SS2</td><td>Send Constant Value (possible values: High-Z, 0 and 1)</td></tr><tr><td>SS3</td><td>Send Constant Value (possible values: High-Z, 0 and 1)</td></tr></table></td></tr>
<tr><td>SPI-RdD</td><td><table><tr><td>MOSI</td><td>Sample</td></tr><tr><td>MISO</td><td>Sample</td></tr><tr><td>SS2</td><td>Unused</td></tr><tr><td>SS3</td><td>Unused</td></tr></table></td></tr>
<tr><td></td><td></td></tr>
</table>

Combination name	Description
SPI-RdD-SS23	MOSI: Sample MISO: Sample SS2: Send Constant Value (possible values: High-Z, 0 and 1) SS3: Send Constant Value (possible values: High-Z, 0 and 1)
SPI-Wr-RdD	MOSI: Data MISO: Data SS2: Sample SS3: Sample
SPI-WrQ	MOSI: Data MISO: Data SS2: Data SS3: Data
SPI-RdQ	MOSI: Sample MISO: Sample SS2: Sample SS3: Sample
Apply-4	MOSI: Send Constant Value (possible values: High-Z, 0 and 1) MISO: Send Constant Value (possible values: High-Z, 0 and 1) SS2: Send Constant Value (possible values: High-Z, 0 and 1) SS3: Send Constant Value (possible values: High-Z, 0 and 1)

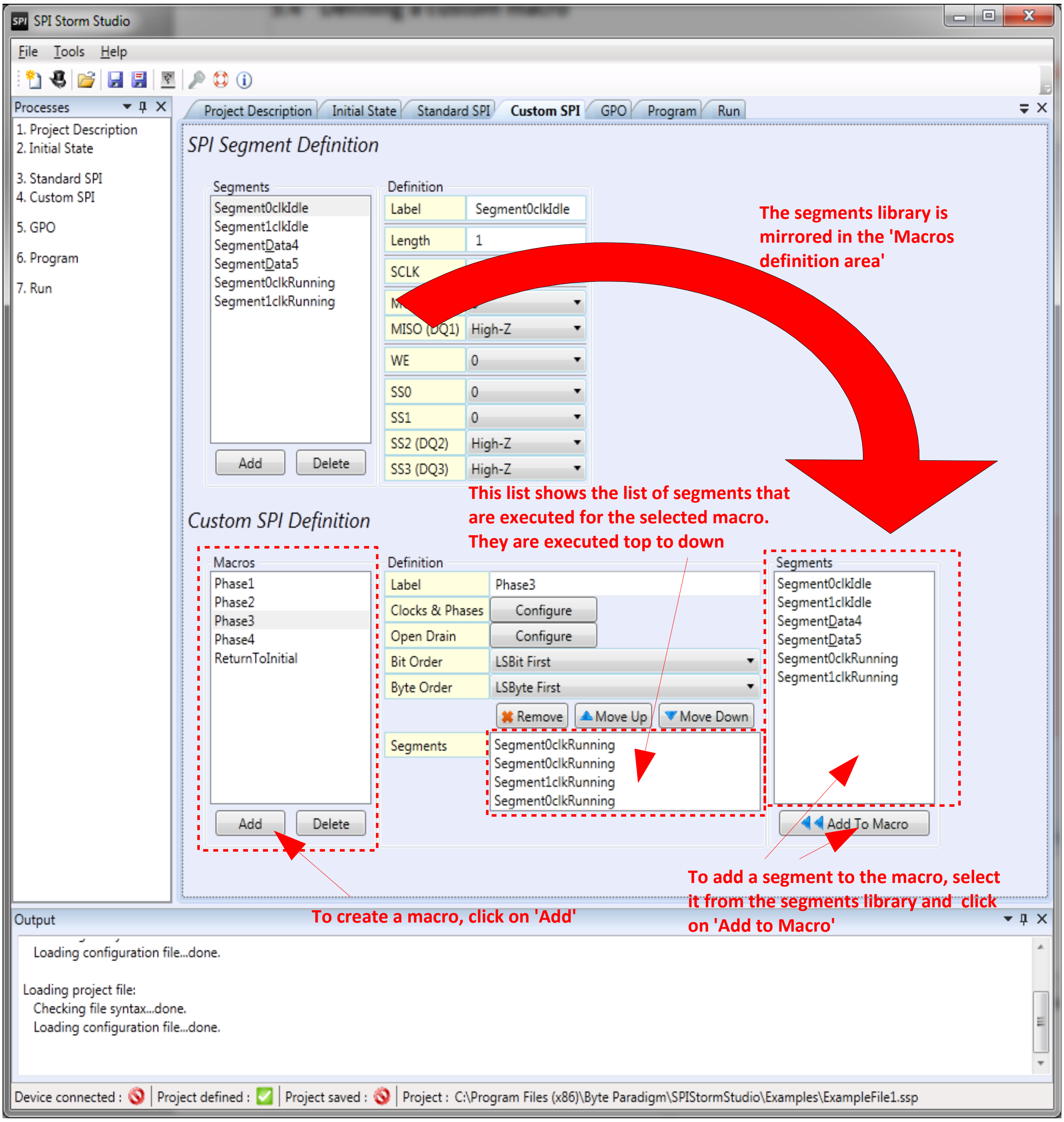
5.3 Custom segment example – detailed

Here is an example of segment definition:

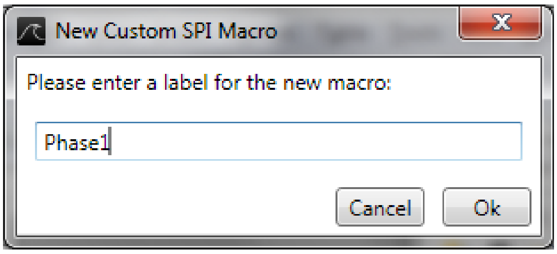


Label name : Segment1
Length : 12 clock cycles
SCLK is held at low level during the segment
MISO is sampled (12 bit)
WE : held at low level
SS0 : held at low level
SS1 : held at high level
Other bits are left unused.

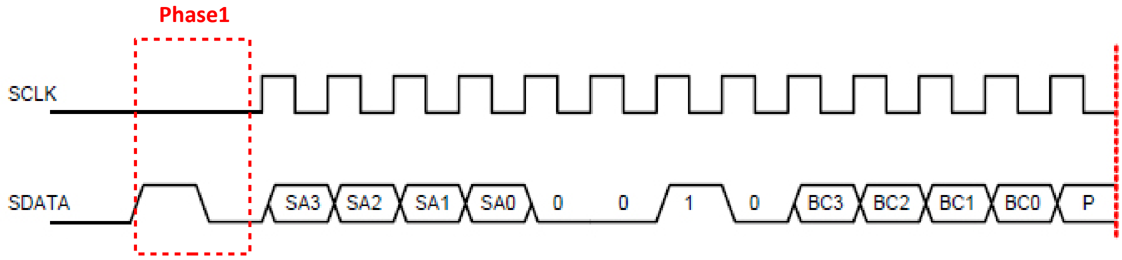
5.4 Defining a custom macro



To create a macro, click on 'Add' in the macro definition area. Specify a name in the pop-up window:



For instance, we'll detail the definition of 'Phase1':

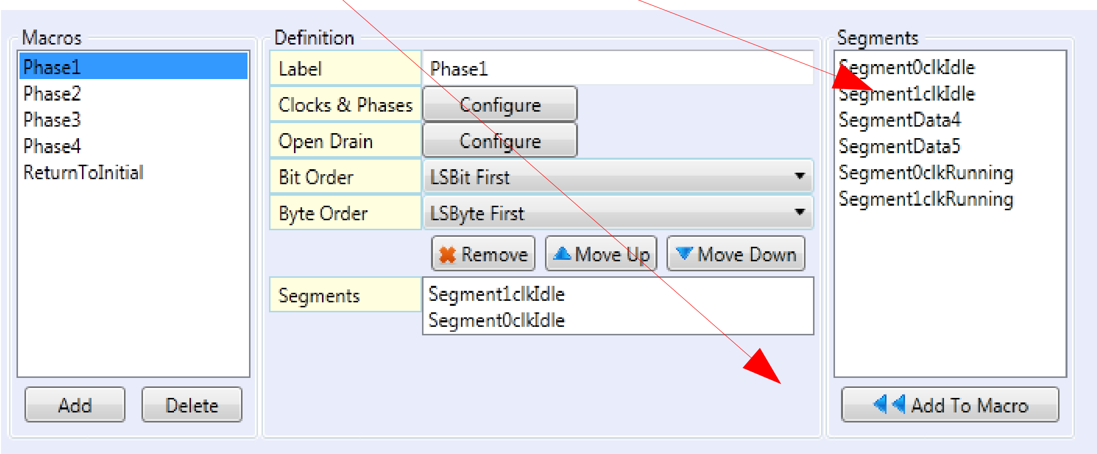


This macro is composed of the following segments:

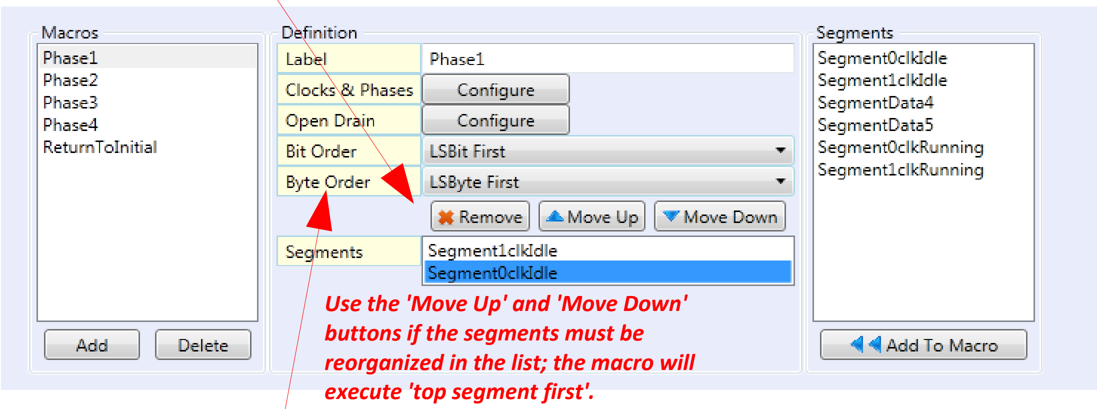
- Segment 'Segment1clkIdle', which sets MOSI to '1' while the SCLK is held low;
- Segment 'Segment0clkIdle', which sets MODI to '0' while the SCLK is held low.

Proceed as follow:

1) Select one segment in the library
2) Click on 'Add to macro' button.
3) Do so for all segments you would like to add to the macro.

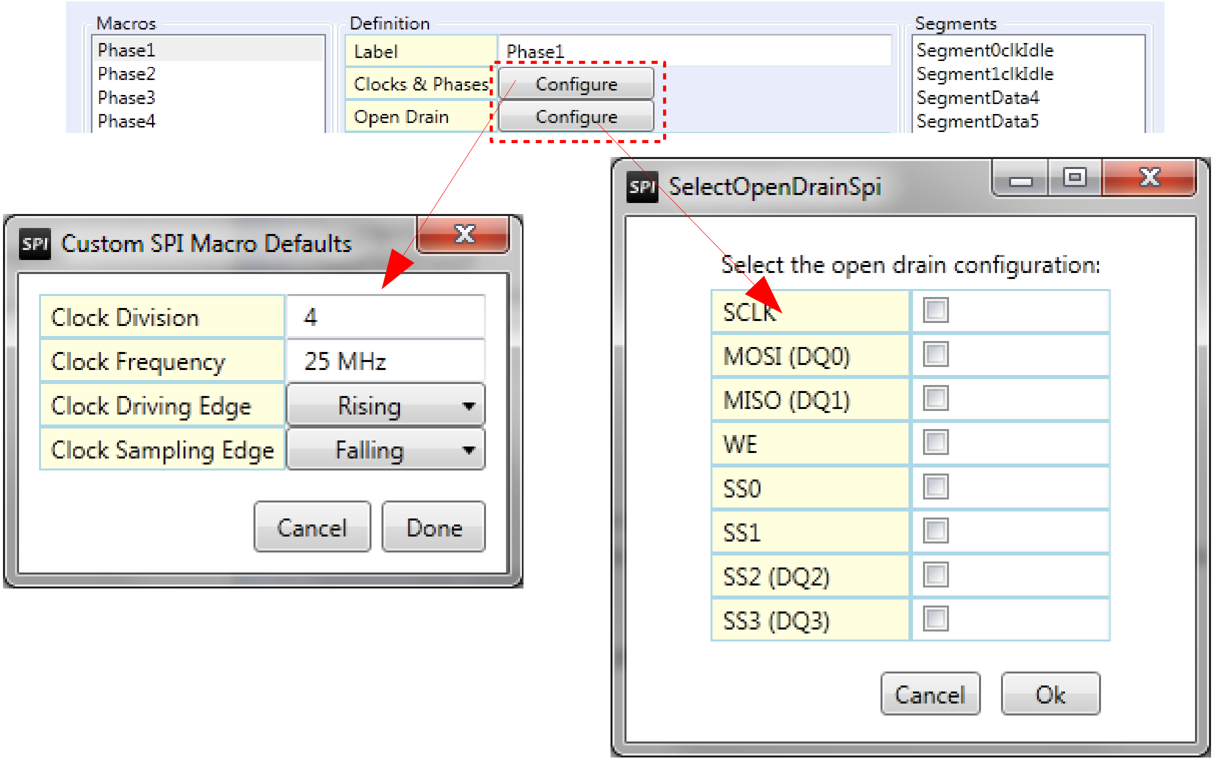


The segments are added to the macro:



Use the 'Bit Order' and 'Byte Order' drop down boxes to configure bit and byte ordering (see section 4.2 for more information).

Use the 'Configure' buttons to set up the clock and the I/O open-drain features;



In the 'Custom SPI Macro Defaults' window, you can specify the clock frequency and other settings related to clock. Clock frequency is actually defined by 'Clock division', which is a dividing factor for the clock relative to a 100 MHz reference clock.

For instance, typing '4' for clock division will result in 100 /4 = 25 MHz clock for SCLK.

In the 'SelectOpenDrainSpi' window, checking a tick box enables the 'open drain' configuration of the I/O.

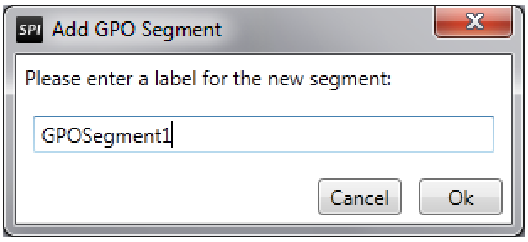
6 GPO patterns sequence

6.1 How to define GPO patterns

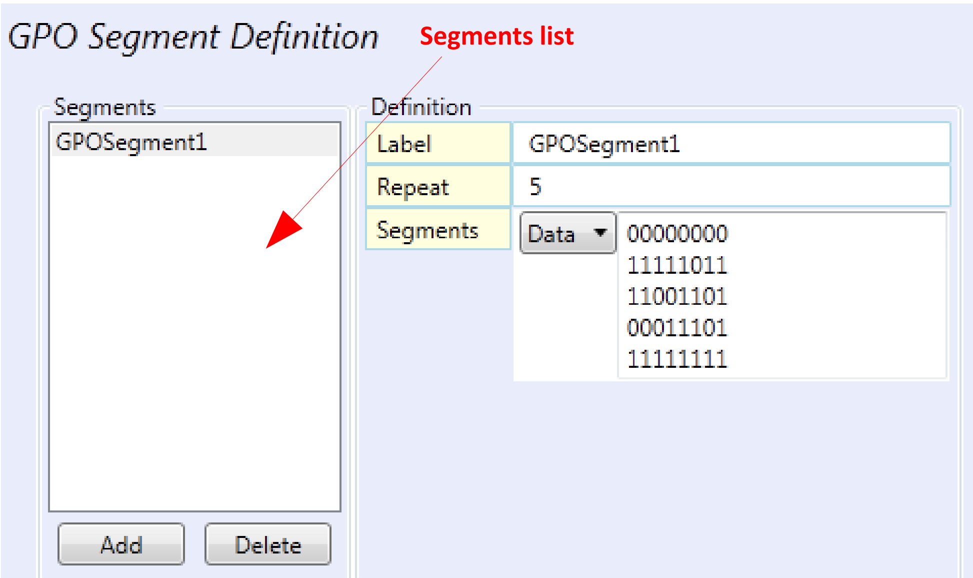
GPO (General Purpose Output) sequences are sequences of arbitrary digital patterns generated onto the 'GPO port' of SPI Storm. Unlike the 'Serial Port', SPI Storm's GPO port is output-only.

GPO sequences are defined from the 'GPO' tab in the main window. Clicking on 'Add' in the 'GPO Segment Definition' prompts for the name of a new GPO segment. As for custom serial protocols, GPO patterns are defined by 'macros'. Each macro is the assembly of one or multiple GPO segments.

6.1.1 Defining GPO segments



Once it is created, the segment name appears in the 'segments list'.

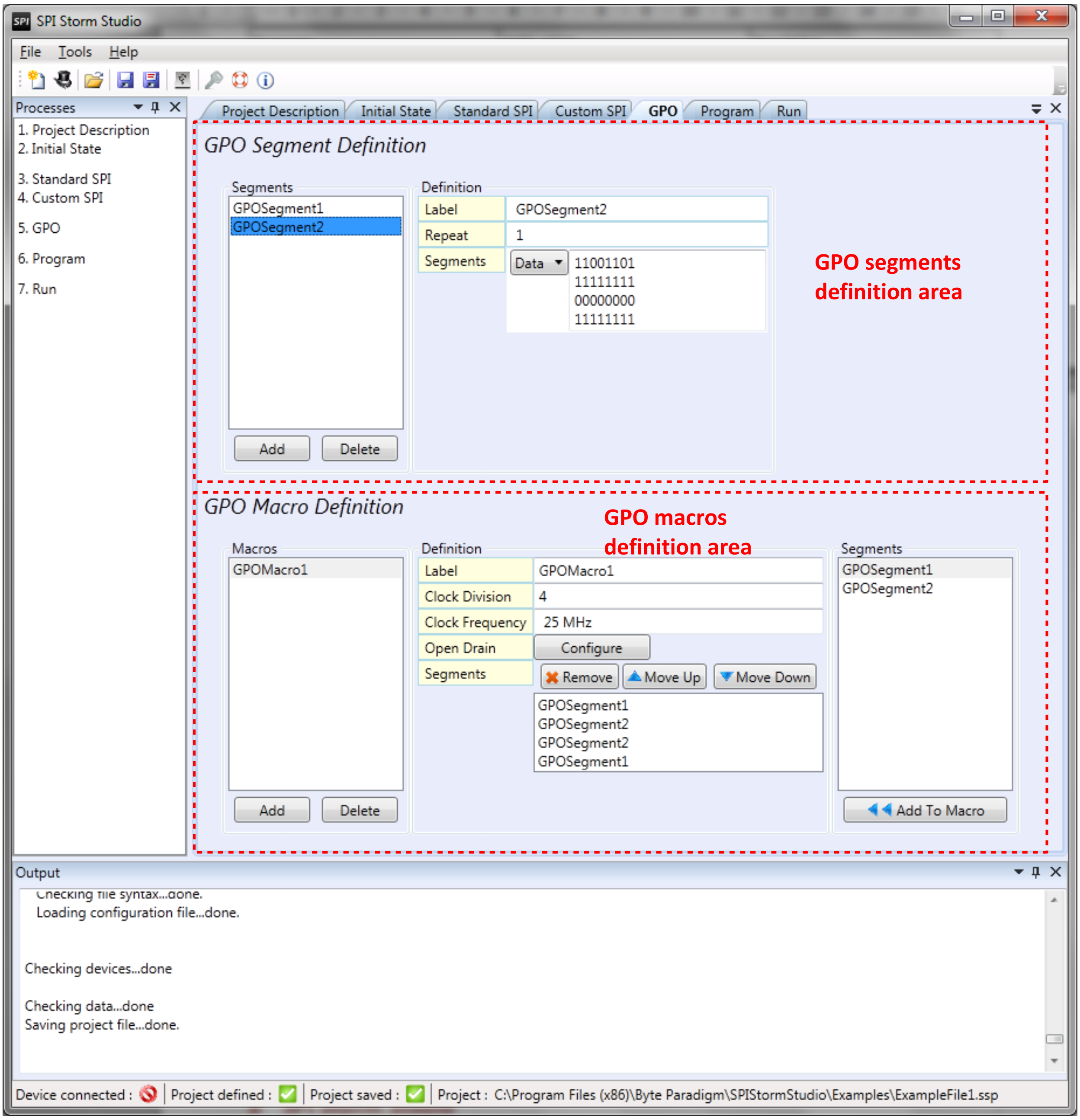


The table below shows the parameters used to define a GPO segment.

Parameter	Valid values	Description
Label	Any string composed of letters and figures	GPO segment name
Repeat	1 to 2147483647	Sets the number of times the data defined in 'Segments' has to be repeated
Segments	Series of 8-bit values representing the logic levels to be generated onto the GPO port	The following options are available for entering data: - 'Data' (selected from the drop down list): in this case, the logic patterns must be entered in the dialog window, one pattern for each line, according to the following format: b7b6b5b4b3b2b1b0 … with b# = 0 or 1 value.

Parameter	Valid values	Description
		Example: 00110111 - 'File' is selected from the drop down list: in this case, a text file must be specified. This text file contains the list of patterns to be sent according to the same format – that is, a list of 8-bit patterns; 1 pattern per line in the file. **Example:** 00000000 11111011 11001101 00011101 11111111

6.1.2 Defining GPO macros

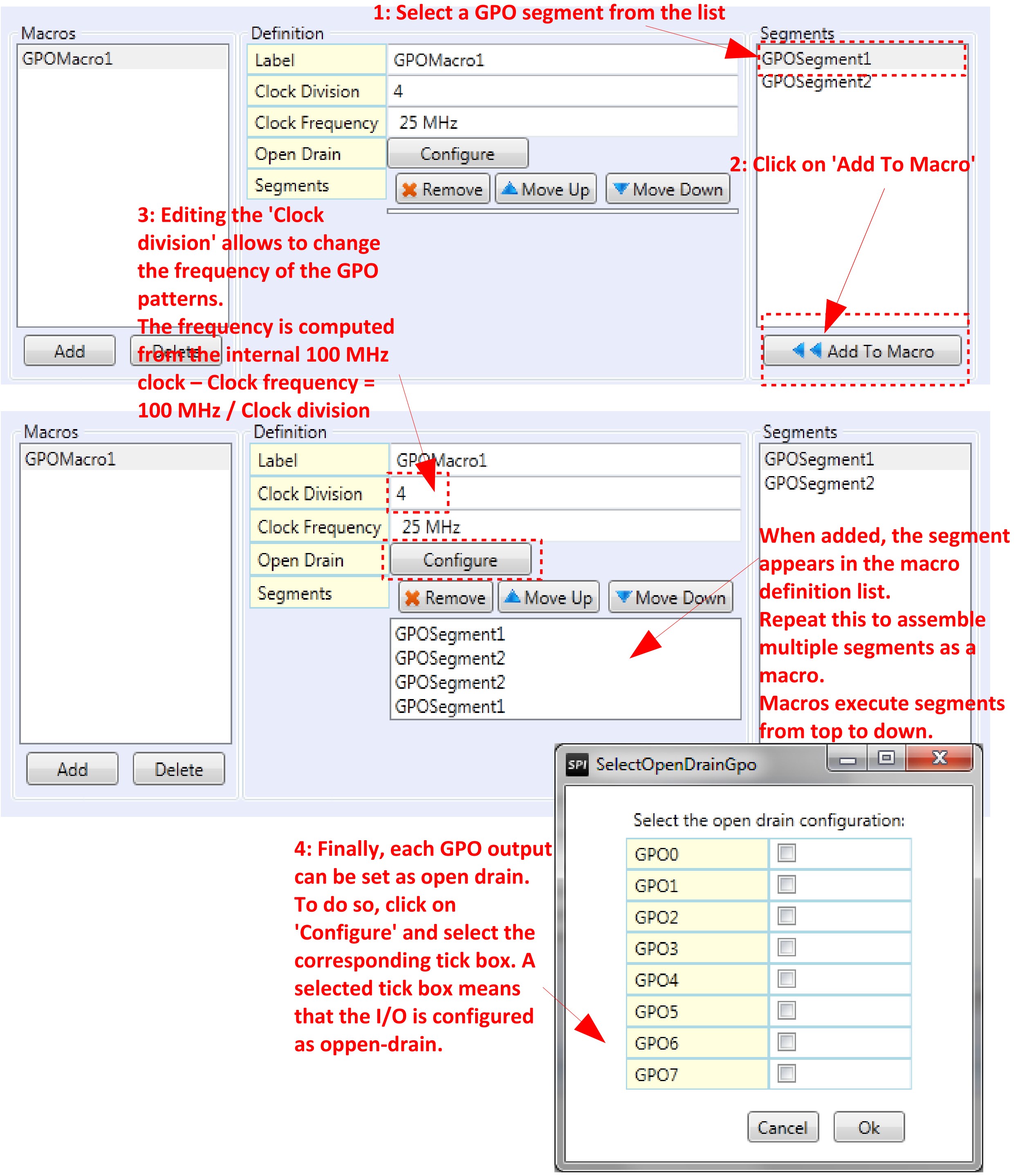


GPO macros are composed of one or multiple GPO segments. GPO Macros are defined from the 'GPO macros definition area' in the GPO tab.

To define a new GPO macro:

1) Click on 'Add' button below the Macros list.
2) A window pops up, prompting for macro name.
3) Once the macro is defined, set up its properties as follows:

– To set up a GPO macro:



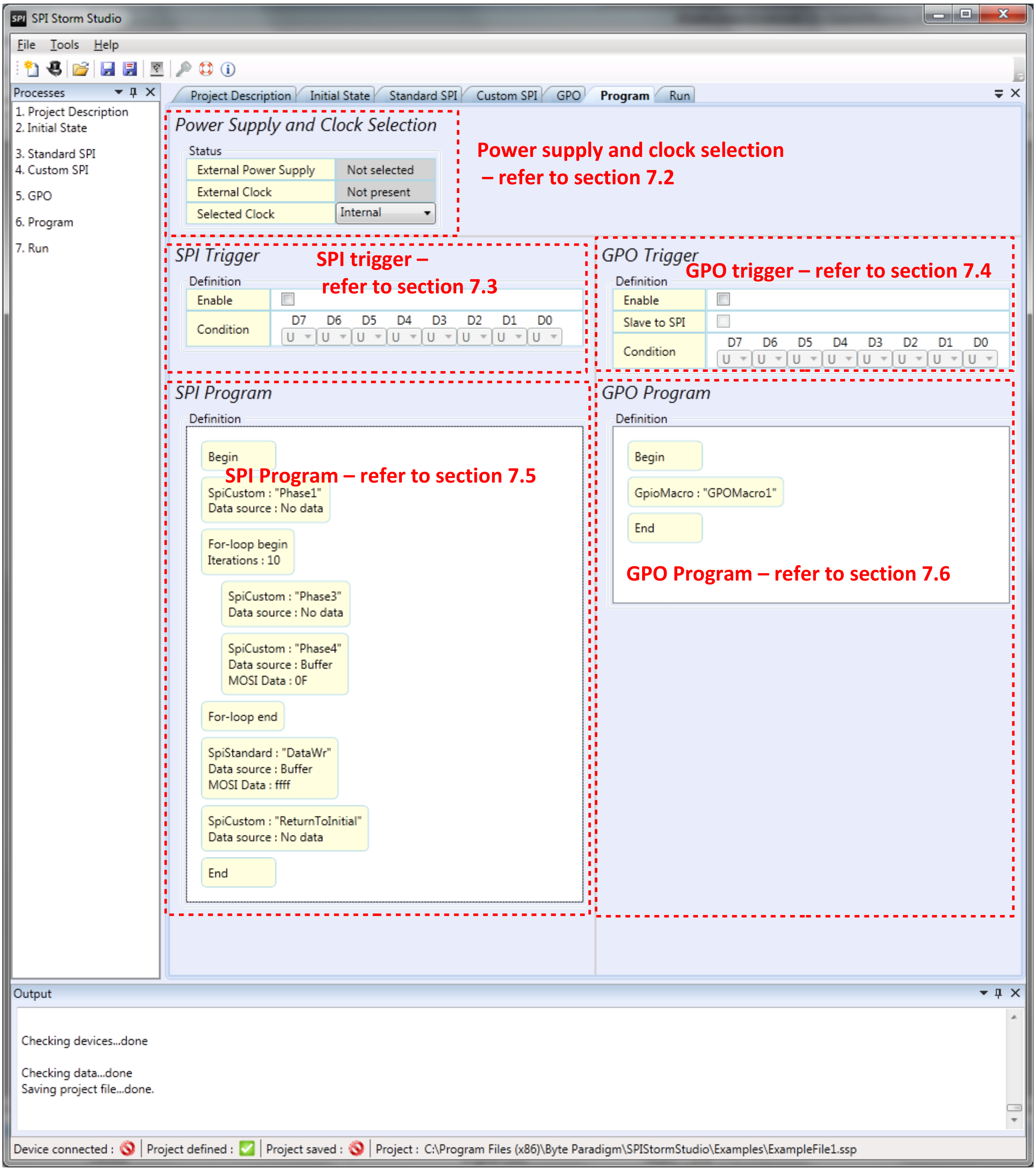
The following table summarizes the parameters of a GPO macro.

Parameter	Possible value	Description
Label	Any string composed of letters and numerical characters	GPO macro name
Clock division	Integer value from 1 to 1024.	Clock division factor. Defines the clock rate at which the GPO macro will be executed. The output rate of the GPO macro is: 100 MHz / (Clock division). The generated clock frequency for the GPO macro is shown in the 'Clock Frequency' field.
Open drain	(Click on button 'Configure' to change open drain configuration of GPO)	The 'Configure button' opens up a windows that allows configuring the GPO outputs as open-drain.
Segments	(list)	List of segments executed when running the selected GPO macro.

7 Defining a program

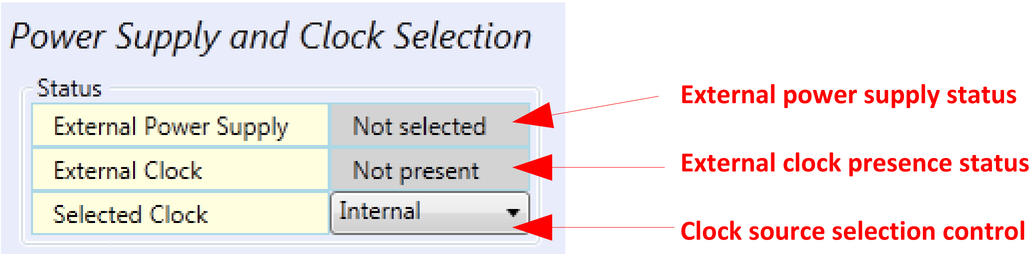
In SPI Storm Studio, 'programs' are defined through the 'Program' tab.

7.1 Program tab overview



7.2 Power supply and clock selection

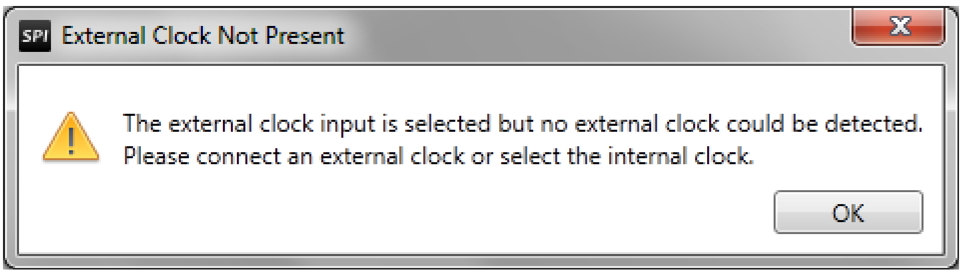
This area shows the status of external power supply and external clock presence. It also allows selecting between clock sources.



Field	Possible values	Description
External Power Supply	Not Selected / Selected	Returns the presence status of an external power supply source on the I/O External voltage supply connector.
External Clock	Not present / present	Returns the status of a clock signal presence detection on the Cki input pin of SPI Storm connector. A valid continuous clock signal must be present on this pin if an external clock reference signal is to be used.
Selected Clock	Internal / External	Controls the clock source to be used for executing a program in SPI Storm studio. If 'Internal' is selected, SPI Storm will use its own internal 100 MHz reference clock signal. If 'external is selected', an external reference clock signal must be supplied onto the CKI pin.

Remark:

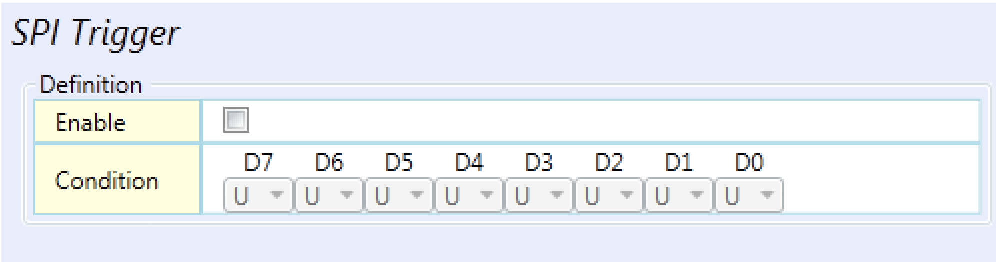
If an **'external clock source'** is selected and the external clock signal is not present at the CKI input pin, SPI Storm Studio cannot run its program. At program run (see section 7), the following message will appear if no valid reference is applied on the external clock input pin:



7.3 SPI trigger

This area is used to configure the trigger conditions used to start execution on the SPI port.

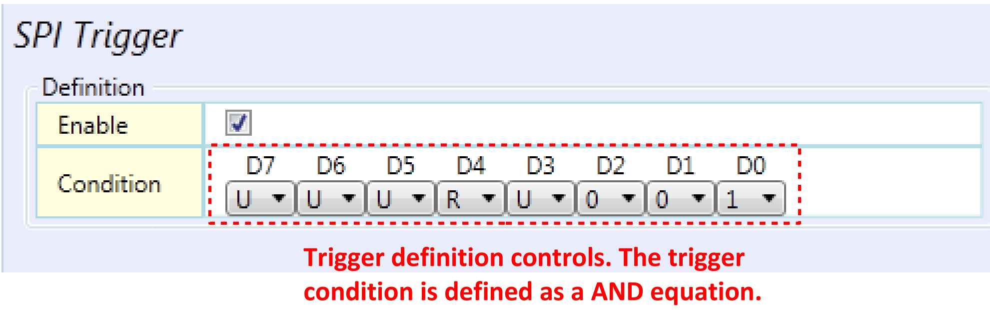
A 'trigger condition' is a logic expression built on the inputs signals of the 'control port'. Upon occurrence of this condition, SPI Storm will start executing the 'SPI Program' defined from the Program tab (see section 7.5).



To enable SPI trigger, check the 'Enable tick box' of the SPI trigger. Leaving this box unchecked disables the trigger: SPI Storm program will be executed as soon as the user clicks on the 'RUN' button in the 'Run' tab.

When a trigger is defined, clicking on the 'RUN' button in the 'Run' tab will just 'arm' the trigger: SPI Storm will wait until it detects the defined trigger condition and then start executing the program.

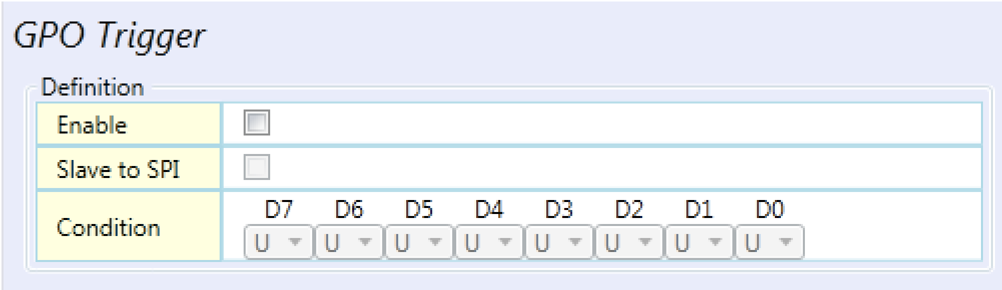
Once SPI trigger is **enabled**, the 'Condition' in SPI Trigger area become active and the trigger condition can be defined on the D7...D0 input pins.



D7 ... D0 possible values	Description
U	**Undefined** – don't care: this input pin is not used for the definition of the trigger condition.
0	**Low logic level :** in this case, a low logic level must be detected.
1	**High logic level :** in this case, a high logic level must be detected.
R	**Rising edge :** in this case, a transition from low logic level to high logic level must be detected.
F	**Falling edge :** in this case, a transition from high logic level to low logic level must be detected.

Example:
If the value 'UUUU1001' is defined in the D7...D0 bits of the SPI trigger, the SPI Program will be executed upon the detection of the binary value "1001" on the control port input pins D3 down to D0.

7.4 GPO trigger



The GPO trigger works on the same fashion as SPI trigger; it is also defined on the control port input pins.

When it is enabled, the option **'Slave to SPI'** allows using the SPI trigger as trigger condition for the GPO program execution. This allows synchronizing GPO program execution with the execution of SPI Program.

If the 'Slave to SPI condition' tick box is **left unchecked**, GPO trigger is defined independently .

7.5 SPI Program

7.5.1 Overview

This area allows defining 'programs' for the SPI port – that is, the sequence of SPI port macros to be executed.

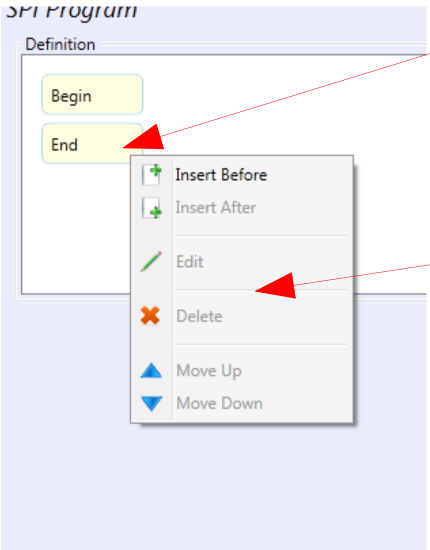
The table below gives the list of items that can be inserted in a 'program'.

Item type	Parameters	Description
Standard SPI macro	Optional: data out field ('MOSI data'), depending on macro.	Macros defined as standard accesses from the SPI tab
Custom SPI macro	Optional: data out field ('MOSI data'), depending on macro.	Macros defined as custom accesses from the Custom SPI tab
Loop	Number of iterations	For loop statement: allows executing a subsequence a predefined number of times.

(Explain the format of simple, dual and quad data).
(Explain where the data in is read back).

7.5.2 Building up a program from SPI Storm Studio GUI

7.5.2.1 Overview

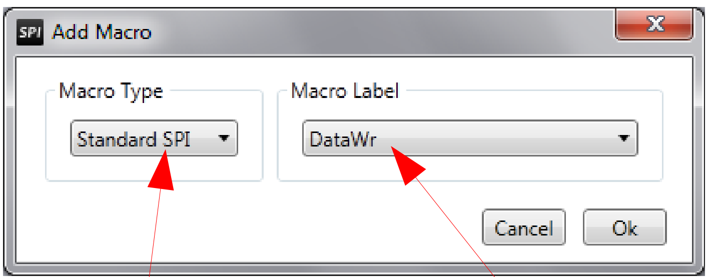


In the program area, right-click on one of the yellow frames, at the position where you would like to insert a program statement.

Select an action from the contextual menu. Valid choices are:

- Insert Before / After: insert a statement before / after the one that you clicked on.
- Edit : edit the selected statement
- Delete: delete / remove the selected statement
- Move Up / Move Down: reorder the program by moving statements up / down

When 'Insert Before' or 'Insert After' is selected, the following window pops up:

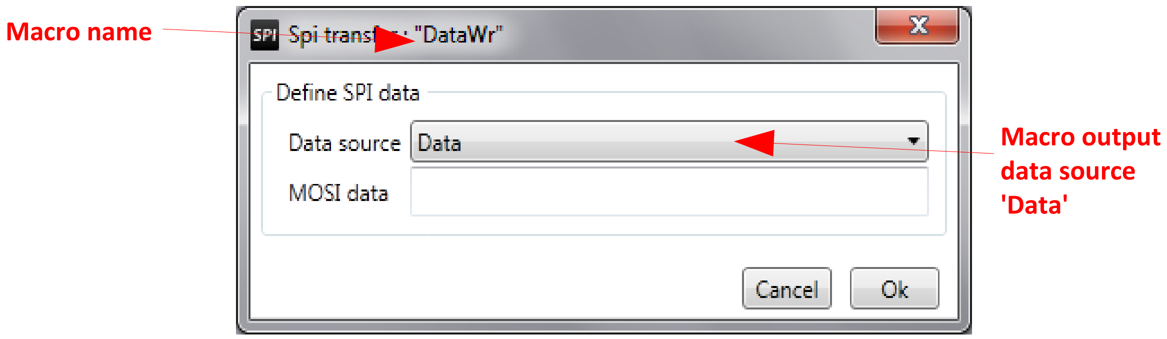


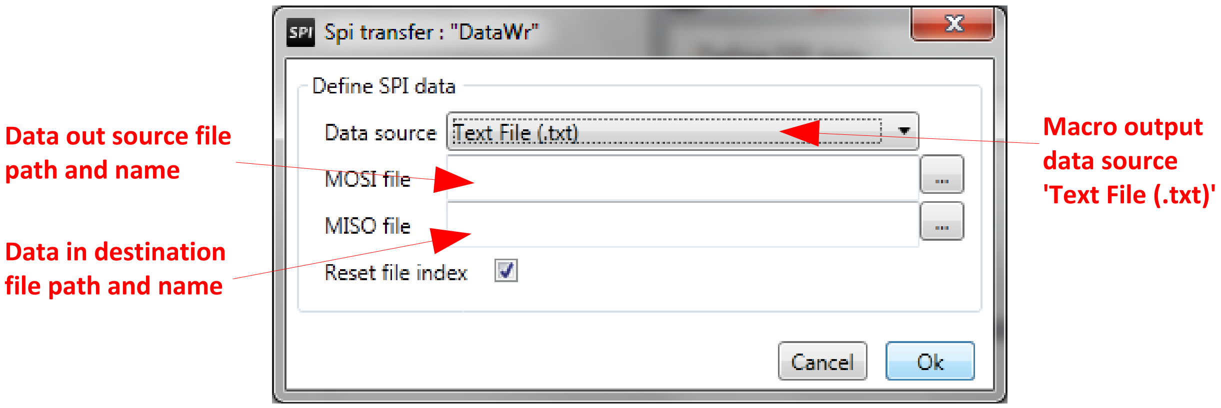
Select Macro Type from drop-down menu.
Valid choices are:
- Standard SPI
- Custom SPI
- Loop

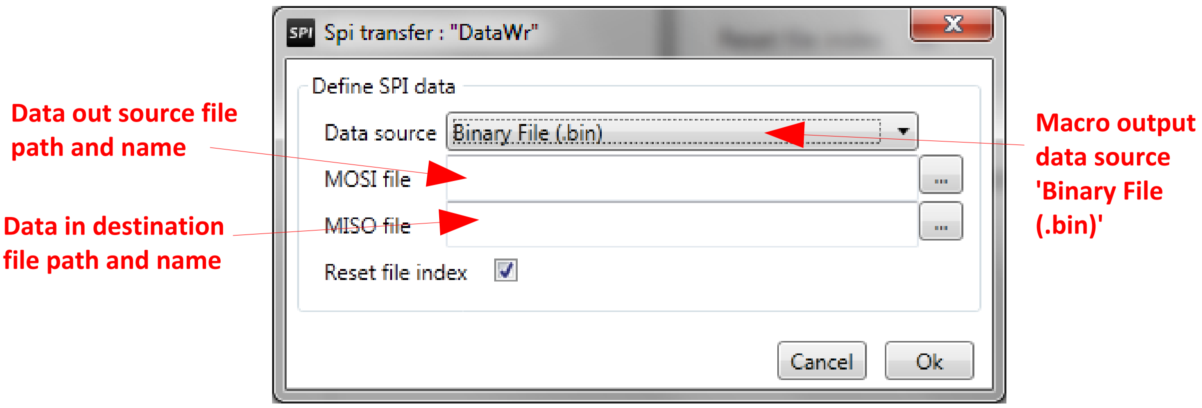
Select macro to insert from drop down.
The list is filled in with the macros of each types that were defined with 'Standard SPI' and 'Custom SPI' tabs of SPI Storm Studio

7.5.2.2 Inserting a Standard SPI or Custom SPI macro

Once the macro that you wish to insert is selected, click on 'OK'.
If the selected macro requires a data parameter, the following window appears. According to the selected output data source, the window format changes.



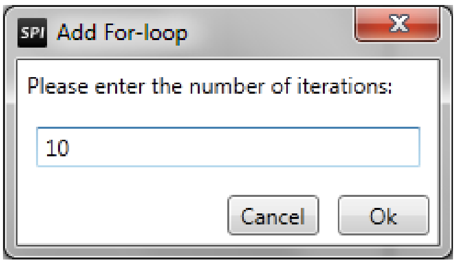




Macro parameter	Valid values	Description
Data source	Data Text File (.txt) Binary File (.bin)	Specifies the source for the data sent out by the macros. 'Data' specifies a constant value inserted in the program. 'Text file' specifies an external text-formatted file containing the data to be sent. 'Binary file' specifies an external binary-formatted file containing the data to be sent.
MOSI data	String representing a serial data as hexadecimal string.	Only valid if 'Data Source' is set to 'Data'. If the corresponding macro expects a constant data to be sent out onto the data lines. The number of used data lines and the total length depend of the defined macro. Example: if a 8 bit data string is defined, 'A9' (without the quotes) is an acceptable data format. Current default bit ordering is 'LS bit', LS byte first. Input data sampled by the called macros can be found from the log file (see 'Running a program'). ***Optional bit ordering will be available in future SPI Storm Studio versions. Please contact support@byteparadigm.com for availability.***
MOSI file	Path and file name containing the data to be sent out.	Only valid if 'Data Source' is set to 'Text File (.txt)' or 'Binary File (.bin)'. In this case, the source data is supplied by means of an external source file. - If text file is selected, each line bears 1 data vector formatted as an hexadecimal string without prefix. - 'Reset file index' allows re-starting at the beginning of the file at each loop iteration (if loops are used in the program). Otherwise, each successive call to the file will increment the index in the file and fetch the 'next data'.
MISO file	Path and file name used as destination for the data being read back by the called macros.	Only valid if 'Data Source' is set to 'Text File (.txt)' or 'Binary File (.bin)'. If the called macros sample data, this file is used as destination to store the sampled data.

7.5.2.3 Inserting a For Loop

In this case, the type 'Loop' is selected from the 'Macro type' drop-down menu.

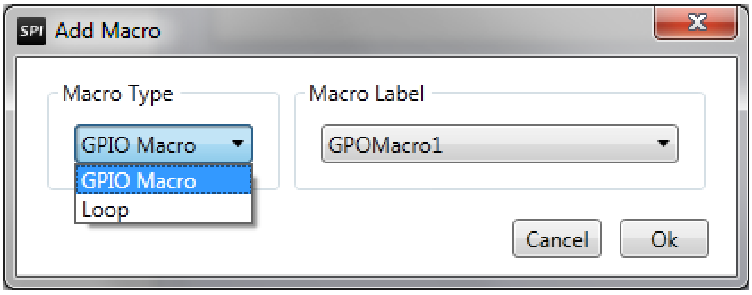


A dialog box pops up, requesting the for loop count.
Nested loops (loops in loops) are not allowed.

7.6 GPO Program

The method for inserting GPO macros and loops in the GPO Program area is identical to this of SPI Program. Please refer to section 6.5 for more details.

As opposed to SPI Program, a GPO Program will only make use of GPIO Macros defined from the GPO tab. For loops are also allowed, as shown in the 'Add Macro window' below.



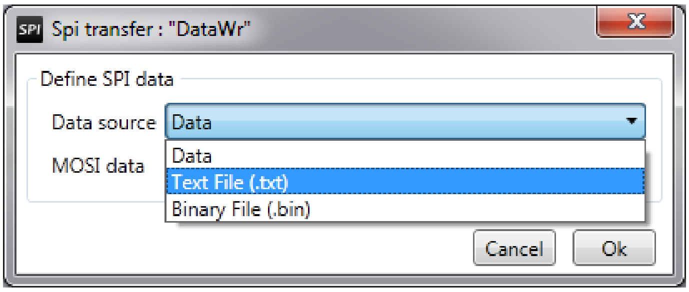
7.7 File formats

This section provides a detailed description of the file formats used in SPI Storm Studio.

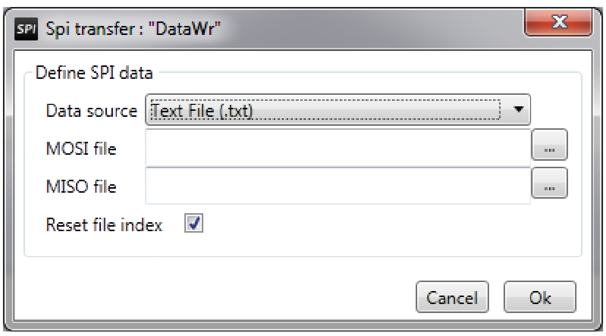
7.7.1 Standard and custom macro file format

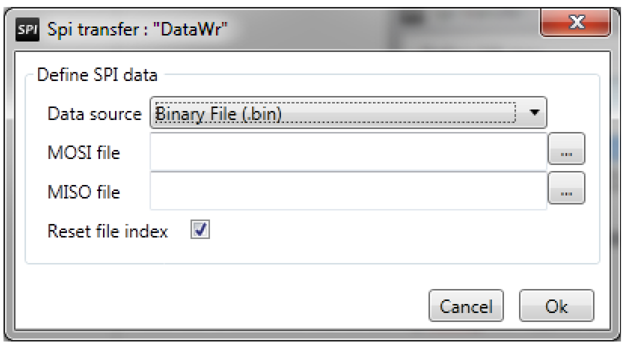
Context: Program tab, when inserting a macro in the SPI program.
When applicable, the data can be provided for macros as follows:

– Entering the data in the GUI (**Data** option – see below);
– Providing a text file (.txt) (**Text File (.txt)** option – see below);
– Providing a binary file (.bin) (**Binary File (.bin)** option – see below).



If one of the **file formats** is selected, one of the following dialog boxes appears:





MOSI file: path to the file for the data that are sent as output by the corresponding macro. This is the location from where the macro reads the data that it has got to send out.
MISO file: path to the file for the data that is read back by the corresponding macro. This is where the macro will write the read back data. No content for this file must be specified, as this file will be written after the execution of the macro.

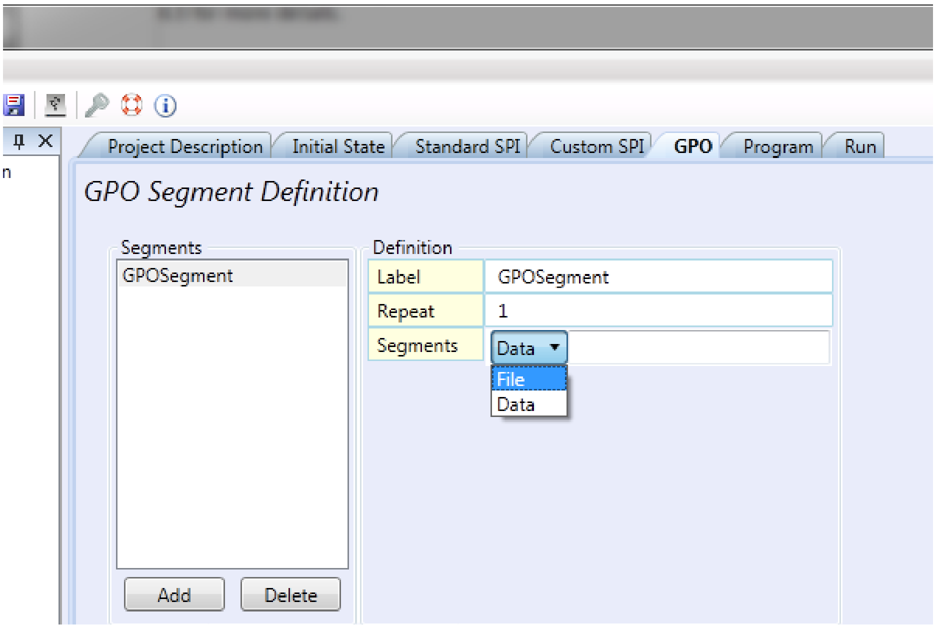
File format:	Text file (.txt)
File structure:	Each line in the file contains a single output data as required by the corresponding macro. Successive calls to the same macro increment the index in the file, allowing placing all the data related to a macro in the same file. Please refer to section 7.5.2.2 for additional options, such as 'reset file index'. Each line must contain the information equivalent to 1 data for the corresponding macro. The size of this data depends on the defined macro. For example, if the macro requires a total of 13 bit out, each line will have to provide 13 bit of data.
Data format in the file:	Hexadecimal number without prefix. The default hexadecimal format is MSBit first / MSByte first – this convention was adopted as a default from SPI Storm Studio version 1.1.14. Each macro bit / byte ordering is defined in the .ssp file through SPI Storm Studio interface. Hereafter is an example with convention **"most significant byte first, most significant bit first" = MSBit, MSByte first.** For instance: data specified on 16 bits as: **1A45** will be sent: **1A - 45** msbit first: **0-0-0-1-1-0-1-0-0-1-0-0-0-1-0-1** Below is an example with convention **"least significant byte first, least significant bit first = LSBit, LSByte first.** For instance: data specified on 16 bits as: **1A45** will be sent: **45 - 1A**, lsbit first: **1-0-1-0-0-0-1-0 0-1-0-1-1-0-0-0**
Example	Let's assume that we specify a file for macro 'DataWr', a standard SPI access with 13 bit length. Each data will have to be 13 bit long, which is formatted onto 4 hexadecimal characters. 1A45 0444 19A4 1BCD 0928 0017 1002

File format:	Binary file (.bin)
File structure:	This file contains all the output data required by the corresponding macro. Successive calls to the same macro increment the position in the file. The macro will fetch the data required for its execution and then increment its position in the file to fetch the next data. Please refer to section 7.5.2.2 for additional options, such as 'reset file index'.
Data format in the file:	Byte-padded data. E.g.: 9 bits of data is mapped onto 2 bytes of data. There is no carriage return in binary files. Data is sent least significant byte first, least significant bit first. Example:

	If the binary file contains 2 bytes of data: **A1 56** If serialized (depends on the format of the macro), data will be sent: Convention MSBit first:, MSByte first **0-1-0-1-0-1-1-0-1-0-1-0-0-0-0-1**- Convention: LSBit first:, LSByte first **1-0-0-0-0-1-0-1 0-1-1-0-1-0-1-0**

7.7.2 GPO segment file format

Data for GPO are specified at the 'segment' level. When a GPO segment is created, the data source can be entered in the GUI or specified as a file. See picture below.



File format:	Text file
File structure:	Each line in the file contains the data of a new 8-bit vector to be applied successively onto the GPO lines. Each line must contain the information for 8 bit (1 byte).
GPO vector format in the file:	Hexadecimal number without prefix, 2 characters, representing a byte of information. Convention: most significant bit first.
Example of file:	A5 33 55 12 66 70 01 10

7.7.3 Output file format

The output file lists the accesses executed by the SPI Storm Studio program. It is notably used to collect the data that is read back, when no input file is specified.

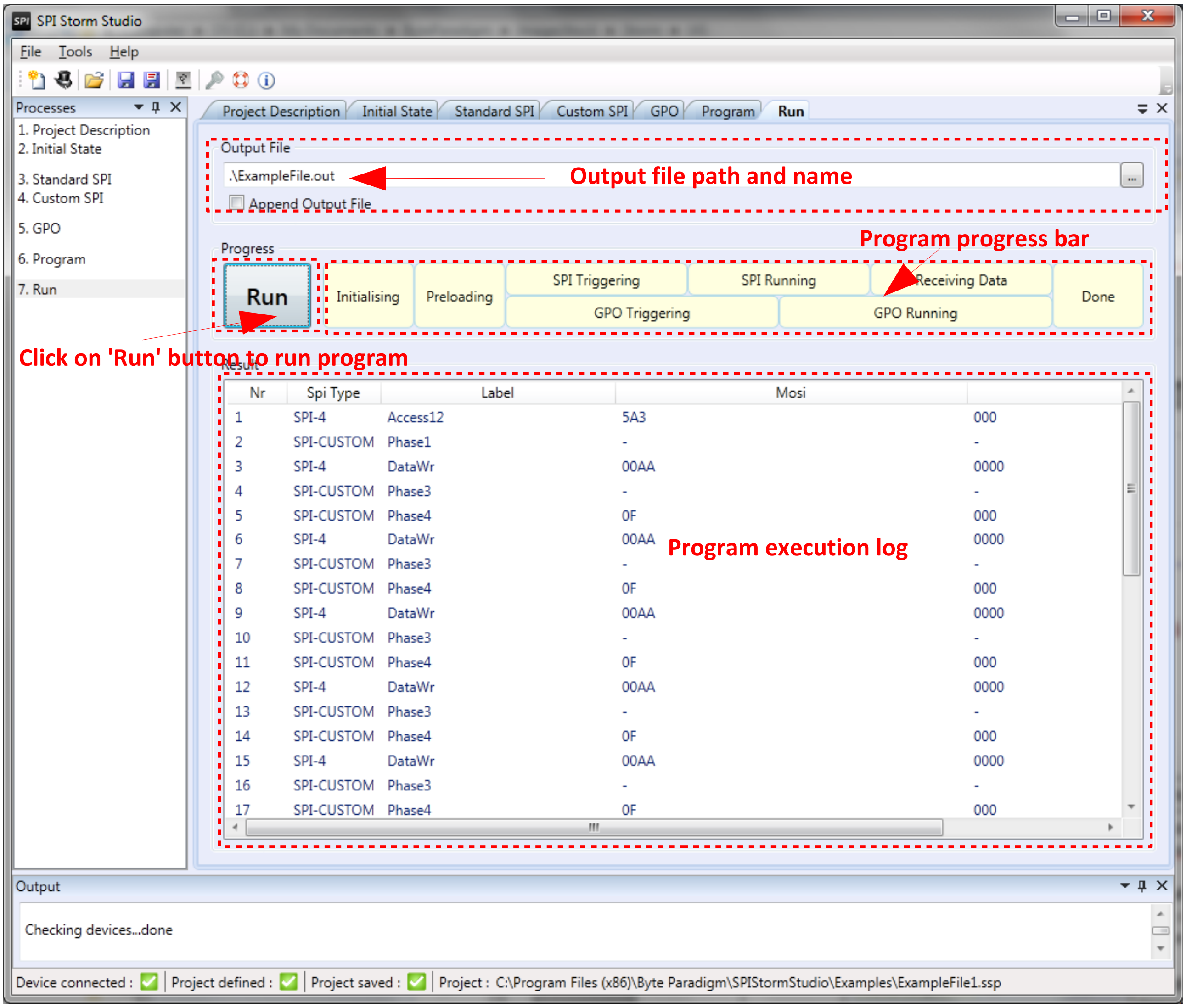
`[Xfer]`
`Nr   = 1`
`Spi4 = 131072`
`Mosi = 1A45`
`Miso = 0000`
	[Xfer] : marks the beginning of a tranfer.
	Nr = index of the transfer
	<macro type identifier> : keyword showing whether a standard SPI or a custom SPI macro was executed. The value that follows is an internal reference number for the macro.
	Mosi : if applicable and variable, shows the data sent by the macro. Nothing is written here if not used.
	Miso : if applicable, shows the data read back by the macro. Nothing is written here if not used.

Example:

```
[Xfer]
Nr   = 1
Spi4 = 131072
Mosi = 1A45
Miso = 0000

[Xfer]
Nr   = 2
Spi4 = 131072
Mosi = 0444
Miso = 0000

[Xfer]
Nr   = 3
SpiC = 262145
Mosi = 8
Miso =

[Xfer]
Nr   = 4
SpiC = 262144
Mosi =
Miso =
```

8 Running a program

Programs are run from the 'Run tab' of SPI Storm Studio. To run a program, a SPI Storm device must be connected to the PC. Please refer to section 3.2 to know how to connect your SPI Storm device.



SPI Storm Studio GUI allows running 'programs'.

For individual calls to macros, please refer to section 9: SPI Storm API.

9 SPI Storm API

9.1 Overview

SPI Storm Studio GUI must not be used to control SPI Storm device. SPI Storm Studio C API can be used to build up your own application / environment.

There is no difference between what can be achieved with the DLL or SPI Storm Studio GUI.

Basically, controlling SPI Storm device always consists in defining a SPI Storm Studio project file which contains the settings of the SPI Storm device and the library of macros that you need. Using the API from within your own environment can be more flexible, according to your application.

For instance, SPI Storm Studio API allows executing macros or programs directly. Calling macros directly can provide more flexibility to organize Standard SPI, Custom SPI and/or GPO accesses.

9.2 Detailed Functions Description

Please refer to document: SPI Storm Studio – C library user's guide: ug_SPIStormStudio_CLib.pdf, available from Byte Paradigm web site: http://www.byteparadigm.com/files/documents/ug_SPIStormStudio_CLib.pdf

9.3 Files Needed to Use the API

Please copy the following file into your application directory:

- SpiStorm.dll
- xspi*.bin
- Your SPI Storm Studio project file (.ssp)

9.4 Programming example

Programming examples can be found from Byte Paradigm's website http://www.byteparadigm.com/support/software-downloads/
.