Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

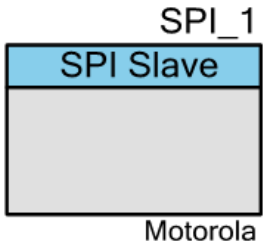
Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

SPI (SCB_SPI_PDL)

2.0

Features



- Original SPI protocol as defined by Motorola
  - All four clock polarity and phase options
- TI: Uses a short pulse on “spi_select” to indicate start of transaction
- National Semiconductor (Microwire): Transmission and Reception occur separately
- Supports a configurable 4- to 16-bit data width for communicating at nonstandard SPI data widths
- Provides DMA Support

General Description

The SCB_SPI_PDL Component provides an industry-standard, 4-wire SPI interface, with additional support for TI mode and National Semiconductor (Microwire) mode. In addition to the standard 8-bit word length, the Component supports a configurable 4 to 16-bit word length for communicating with nonstandard SPI word lengths.

The SCB_SPI_PDL Component is a graphical configuration entity built on top of the SCB driver available in the Peripheral Driver Library (PDL). It allows schematic-based connections and hardware configuration as defined by the Component Configure dialog.

When to Use a SCB_SPI_PDL Component

You can use the SCB_SPI_PDL Component any time the PSoC device must interface with one or more SPI devices. The SCB_SPI_PDL Component can also be used with many devices implementing a shift-register-type serial interface.

Definitions

- SPI – Serial Peripheral Interface. The original SPI protocol was defined by Motorola. The SCB_SPI_PDL Component supports the following SPI modes:
  - Motorola (MT) – The original SPI protocol as defined by Motorola. It is a full duplex protocol: transmission and reception occur at the same time.
  - Texas Instruments (TI) – The Texas Instruments’ SPI protocol redefines the use of the “spi_select” signal. It uses the signal to indicate the start of a data transfer,

rather than a low active slave select signal. The start of a transfer is indicated by a high active pulse of a single bit transfer period. This pulse may occur one cycle before the transmission of the first data bit, or may coincide with the transmission of the first data bit. The transmitted clock SCLK is a free running clock.

- National Semiconductors (NS) Microwire – the National Semiconductors' SPI protocol is a half-duplex protocol. Rather than transmission and reception occurring at the same time, transmission and reception take turns (transmission happens before reception). A single "idle" bit transfer period separates transmission from reception.

- MOSI – Master Out Slave In. Master controlled data line.
- MISO – Master In Slave Out. Slave controlled data line.
- SS – Salve Select. Master controlled slave select.
- SCLK – Serial Clock. Master controlled clock.
- CPHA – Clock Phase. Defines which phase (rising or falling edge) of SCLK data is sampled on.
- CPOL – Clock Polarity. Defines SCLK idle polarity.

Quick Start

1. Drag a SPI (SCB) Component from the Component Catalog *Cypress/Communications/SPI* folder onto your schematic (the placed instance takes the name SPI_1).
2. Double-click to open the Configure dialog.
3. On the **Basic** tab, select the **Data rate** at which the SPI interface is expected to communicate.
4. Open Design-Wide Resources Pin Editor, and select pins for the SPI interface. The choice of pins that can be used for the SPI interface is limited.
5. Build the project in order to verify the correctness of your design. This will add the required PDL modules to the Workspace Explorer and generate the configuration data for the SPI_1 instance.
6. In the *main.c* file, initialize the peripheral and start the application:

   Interrupt parameter is "Internal" (uses SPI High-Level communication cy_scb driver functions; refer to Interrupt Service Routine section):

```
/* Implement ISR for SPI_1 */
void SPI_1_Isr(void)
{
    Cy_SCB_SPI_Interrupt(SPI_1_HW, &SPI_1_context);
}
```

```
/* Allocate TX and RX buffers */
#define BUFFER_SIZE    (64UL)
uint8_t bufferTx[BUFFER_SIZE];
uint8_t bufferRx[BUFFER_SIZE];

/* Initialize SCB for SPI operation with GUI selected settings */
(void) Cy_SCB_SPI_Init(SPI_1_HW, &SPI_1_config, &SPI_1_context);

/* Hook interrupt service routine and enable interrupt */
Cy_SysInt_Init(&SPI_1_SCB_IRQ_cfg, &SPI_1_Isr);
NVIC_EnableIRQ(SPI_1_SCB_IRQ_cfg.intrSrc);

/* Enable SPI */
Cy_SCB_SPI_Enable(SPI_1_HW);

/* Execute transfer
* When configured as master it starts transfer to the slave.
* When configured as slave it prepares for master transfer.
*/
(void) Cy_SCB_SPI_Transfer(SPI_1_HW, bufferTx, bufferRx, BUFFER_SIZE,
                           &SPI_1_context);
```

Interrupt parameter is “External” (uses SPI Low-Level communication cy_scb driver functions; refer to Interrupt Service Routine section):

```
/* Allocate buffer */
#define BUFFER_SIZE (64UL)
uint8_t buffer[BUFFER_SIZE];

/* Initialize SCB for SPI operation with GUI selected settings */
(void) Cy_SCB_SPI_Init(SPI_1_HW, &SPI_1_config, NULL);

/* Enable SPI */
Cy_SCB_SPI_Enable(SPI_1_HW);

/* Put data into TX FIFO.
* When configured as master it starts transfer to the slave.
* When configured as slave it prepares for master transfer.
*/
Cy_SCB_SPI_WriteArrayBlocking(SPI_1_HW, buffer, BUFFER_SIZE);
```

7. Build and program the device.

Input/Output Connections

This section describes the various input and output connections for the SCB_SPI_PDL Component. An asterisk (*) in the following list indicates that it may not be shown on the Component symbol for the conditions listed in the description of that I/O.

Name	Type	Description
clock*	Digital Input	Clock that operates this block. The presence of this terminal varies depending on the **Enable Clock from terminal** parameter.

Name	Type	Description
interrupt*	Digital Output	This signal can only be connected to an interrupt Component or left unconnected. The presence of this terminal varies depending on the **Interrupt** parameter.
rx_dma*	Digital Output	This signal is used to trigger a DMA transfer to get data from the RX FIFO and can only be connected to DMA channel trigger input. The output of this terminal is controlled by the RX FIFO Level parameter. The presence of this terminal varies depending **RX Output** parameter.
tx_dma*	Digital Output	This signal is used to trigger a DMA transfer to put data into the TX FIFO and can only be connected to DMA channel trigger input. The output of this terminal is controlled by the TX FIFO Level parameter. The presence of this terminal varies depending **TX Output** parameter.
s_mosi*	Digital Input	The slave mosi input receives the Master Output Slave Input (MOSI) signal from a master device. The presence of this terminal varies depending on the **Show SPI terminals** and **Remove MOSI** parameters.
s_miso*	Digital Output	The slave miso output carries the Master In Slave Out (MISO) signal to a master device. The presence of this terminal varies depending on the **Show SPI terminals** and **Remove MISO** parameters.
s_sclk*	Digital Input	The slave sclk input receives the Serial Clock (SCLK) signal from a master device. The presence of this terminal varies depending on the **Show SPI terminals** and **Remove SCLK** parameters.
s_ss*	Digital Input	The slave ss input receives the Slave Select (SS) signal from a master device. The presence of this terminal varies depending on the **Show SPI** terminals and **Number of SS** parameters.
m_mosi*	Digital Output	The master mosi output carries the Master Output Slave In (MOSI) signal to a slave device. The presence of this terminal varies depending on the **Show SPI terminals** and **Remove MOSI** parameters.
m_miso*	Digital Input	The master miso input receives the Master Input Slave Output (MOSI) signal from a slave device. The presence of this terminal varies depending on the **Show SPI terminals** and **Remove MISO** parameters.
m_sclk*	Digital Output	The master sclk output carries the Serial Clock (SCLK) signal to a slave device. The presence of this terminal varies depending on the **Show SPI terminals** and **Remove SCLK** parameters.
m_ss0* – m_ss3*	Digital Output	The master ss0 – ss3 output carries the Slave Select (SS) signal to a slave device. The maximum number of hardware controlled slave select lines is 4. The presence of this terminal varies depending on the **Show SPI terminals** and **Number of SS** parameters.

Internal Pins Configuration

By default, the SPI pins are buried inside Component:

- Slave pins: SPI_1_miso_s, SPI_1_mosi_s, SPI_1_sclk_s, SPI_1_ss_s

- Master pins: SPI_1_miso_m, SPI_1_mosi_m, SPI_1_sclk_m, SPI_1_ss0_m, SPI_1_ss1_m, SPI_1_ss2_m, SPI_1_ss3_m.

These pins are buried because they use dedicated connections and are not routable as general purpose signals. For more information, refer to the I/O System section in the device Technical Reference Manual (TRM).

The preferred method to change pins configuration is to enable **Show Terminals** option on the Pins Tab and configure pins connected to the Component. Alternatively, the cy_gpio driver API can be used.

Note The instance name is not included in the Pin Names provided in the following tables:

Pin Name	Direction	Drive Mode	Initial Drive State	Threshold	Slew Rate	Descriptiion
mosi_s	Input	High Impedance Digital	High	CMOS	–	The mosi_s input pin receives the Master Output Slave Input (MOSI) signal from a master device.
miso_s	Output	Strong Drive	High	–	Fast	The s_miso output pin drives the Master In Slave Out (MISO) signal to the master device on the bus. The pin output enable is assigend to 0 when slave is not selected.
sclk_s	Input	High Impedance Digital	High	CMOS	–	The sclk_s input pin receives the Serial Clock (SCLK) signal. It provides the slave synchronization clock input to the device.
ss_s	Input	High Impedance Digital	High	CMOS	–	The ss_s input pin receives the Slave Select (SS) signal to the device.

Pin Name	Direction	Drive Mode	Initial Drive State	Threshold	Slew Rate	Descriptiion
mosi_m	Output	Strong Drive	High	–	Fast	The mosi_m output pin drives the Master Output Slave Input (MOSI) signal from the master device on the bus.
miso_m	Input	High Impedance Digital	High	CMOS	–	The miso_m input pin receives the Master In Slave Out (MISO) signal from a slave device.
sclk_m	Output	Strong Drive	High	_	Fast	The sclk_m output pin drives the Serial Clock (SCLK) signal. It provides the master synchronization clock output to the slave devices on the bus.
ss0_m – ss3_m	Output	Strong Drive	High	–	Fast	The ss0_m – ss3_m output pin drives the Slave Select (SS) signal to the slave devices on the bus.

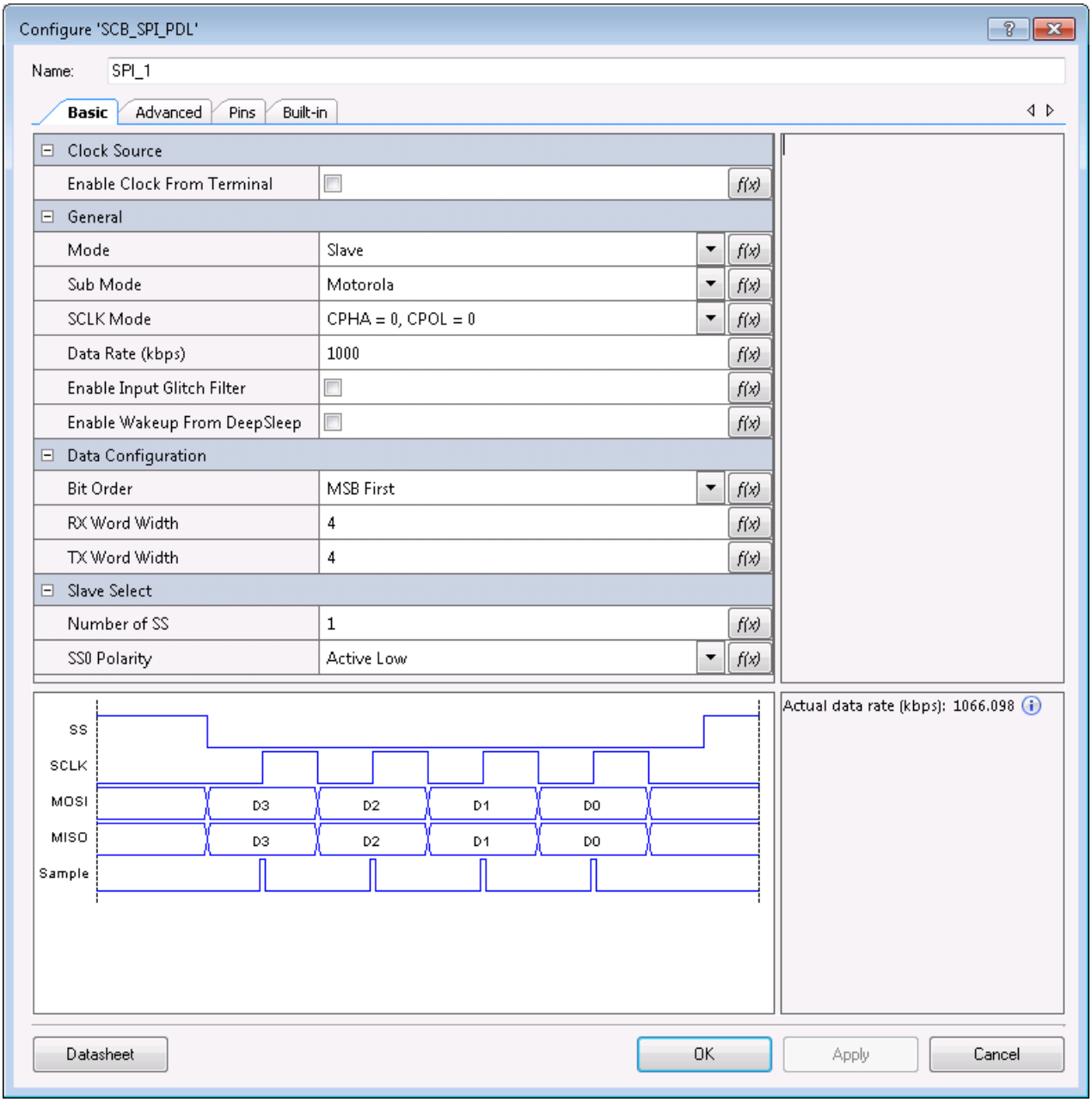
The Input threshold level for **input pins** is CMOS, which should be used for the vast majority of application connections. The Input Buffer for **output pins** is disabled so as not to cause current linkage in low power mode. Reading the status of these pins always returns zero. To get the current status, the input buffer must be enabled before a status read. The other **input pins** and **output pins** parameters are set to default. Refer to Pins Component datasheet for more information about default parameters values.

Component Parameters

The SCB_SPI_PDL Component Configure dialog allows you to edit the configuration parameters for the Component instance.

Basic Tab

This tab contains the Component parameters used in the general peripheral initialization settings.



Parameter Name	Description
Enable Clock from Terminal	This parameter allows choosing between an internally configured clock (by the Component) or an externally configured clock (by the user) for Component operation.
Mode	This parameter specifies the mode of the SPI operation as: the slave or master.
Sub Mode	This parameter specifies the sub-mode of the SPI as: Motorola, TI (Start Coincides), TI (Start Precedes), or National Semiconductor (Microwire).
SCLK Mode	This parameter specifies the serial clock phase (CPHA) and polarity (CPOL) combination.
Data Rate (kbps)	This parameter specifies the data rate in kbps. The actual data rate may differ based on the available clock frequency and Component settings. The range: 1-25000 kbps (master), 1-15000 kbps (slave).
Oversample	This parameter defines how many Component clocks are used to generate the SCLK period (only applicable for the master mode). When the oversample is even the first and second phase of the clock period are the same. Otherwise the first phase of the clock signal period is one component clock cycle longer than the second phase. The range: 4-16 (MISO presents) and 2-16 (MISO is removed).
Actual Data Rate (kbps) [1]	The actual data rate displays the data rate at which the Component will operate with current settings. The factors that affect the actual data rate calculation are: the nominal frequency of the Component clock (internal or external) and oversampling factor (only for the master mode).
Enable Input Glitch Filter	This parameter applies a digital 3-tap median filter to the SPI input lines.
Enable Wakeup from DeepSleep	This parameter enables the Component to wake the system from Deep Sleep when an SPI slave-select event occurs (only applicable for slave mode, sub mode Motorola and internal interrupt).
Enable MISO Late Sampling	This option allows the master to sample the MISO signal half a SCLK period later (on the alternate serial clock edge). Late sampling addresses the round-trip delay associated with transmitting SCLK from the master to the slave and transmitting MISO from the slave to the master.
SCLK Free Running	This parameter allows the master to generate SCLK continually. It is useful when the master SCLK is connected to the slave device that uses it for functional operation rather than only the SPI functionality.
Bit Order	This parameter defines the direction in which the serial data is transmitted. When set to the MSB first, the most-significant bit is transmitted first. When set to the LSB first, the least-significant bit is transmitted first.
RX Data Width	This option defines the width of a single data element for the RX direction in bits. This number must match with TX Word Width for all SPI sub-modes except National Semiconductor (Microwire). The range: 4-16.
TX Data Width	This option defines the width of a single data element for the TX direction in bits. This number must match with RX Word Width for all SPI sub-modes except National Semiconductor (Microwire). The range: 4-16.

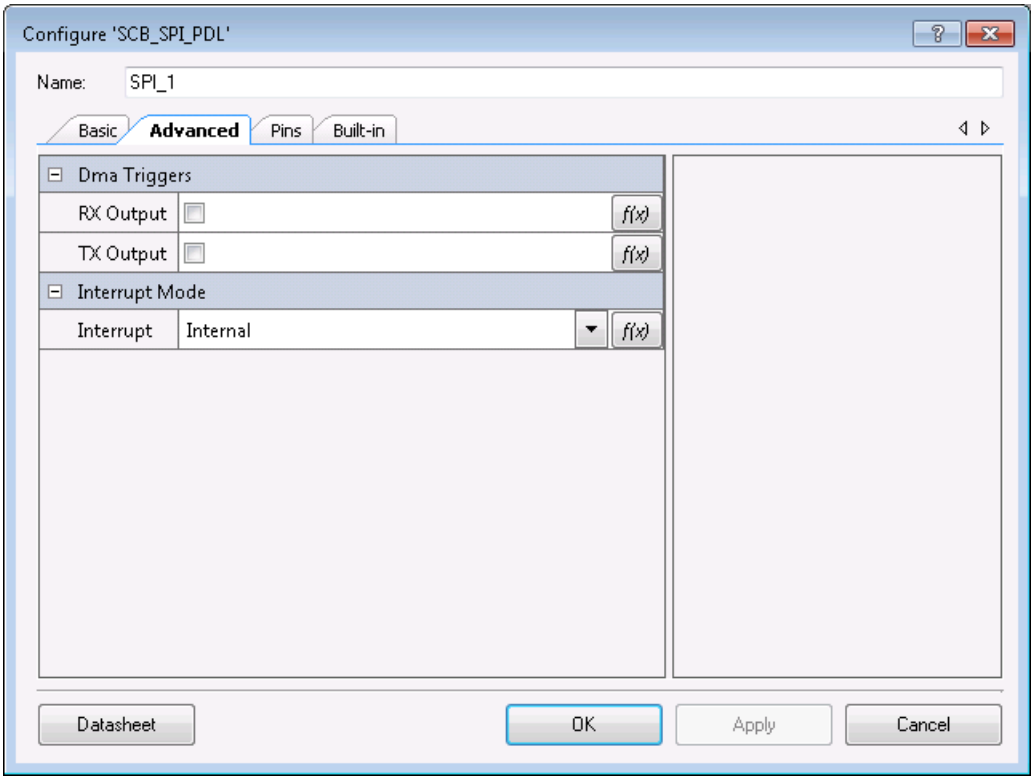
[1] For actual data rate calculation, the following assumptions are made:

- For master mode, the external slave device characteristics are not taken into count and PCB delays are 0. To determine the data rate of your system, refer to the Data Rate Calculations section.
- For the slave mode, the external master captures data on the half SCLK period after the driving edge. If the master captures data half SCLK period later, then the actual data rate is twice what is displayed. To determine the data rate of your system, refer to the Data Rate Calculations section.

Parameter Name	Description
Deassert SS Between Data Elements	This parameter determines if individual data transfers are separated by the slave select de-selection.
Number of SS	This parameter determines the number of the slave-select lines. The slave has a single slave-select line. The master has up to 4 lines. The range: 1-4.
SS0 Polarity	This parameter defines the active polarity of the slave-select 0 signal as Active Low or Active High.
SS1 Polarity	This parameter defines the active polarity of the slave-select 1 signal as Active Low or Active High.
SS2 Polarity	This parameter defines the active polarity of the slave-select 2 signal as Active Low or Active High.
SS3 Polarity	This parameter defines the active polarity of the slave-select 3 signal as Active Low or Active High.

Advanced Tab

This tab contains the Interrupt configuration settings.

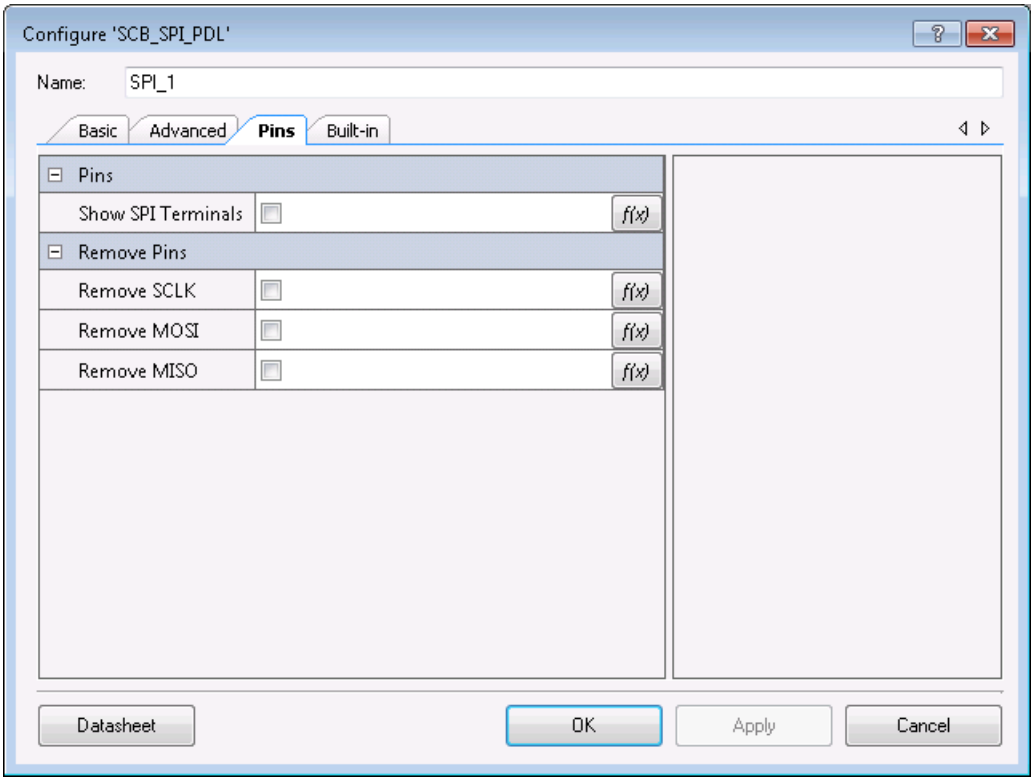


Parameter Name	Description
RX Output	This parameter enables the RX trigger output terminal (rx_dma) of the Component. This terminal must be connected to the DMA trigger input or left unconnected.
TX Output	This parameter enables the TX trigger output terminal (tx_dma) of the Component. This terminal must be connected to the DMA trigger input or left unconnected.

Parameter Name	Description
RX FIFO Level	This parameter determines behavior of signal that drives the RX FIFO Above Level interrupt source and RX trigger output as follows: the signal is active while the number of data elements in the RX FIFO is greater than the value of RX FIFO Level. For example, the RX FIFO has 8 data elements and the RX FIFO level is 0. The signal remains active until all data elements are read from the RX FIFO. The range: 0 – 127 (when TX/RX Word Width <= 8), and 0 - 63 otherwise.
TX FIFO Level	This parameter determines the behavior of the signal that drives the TX FIFO Below Level interrupt source and TX trigger output as follows: the signal is active while the number of data elements in the TX FIFO is less than the value of the TX FIFO Level. For example, the TX FIFO has 0 data elements (empty) and the TX FIFO level is 7. The signal remains active until TX FIFO is loaded to contain 7 data elements. The range: 0 – 127 (when TX/RX Word Width <= 8), and 0 - 63 otherwise.
Interrupt	Internal or External; refer to Interrupt Service Routine section.
SPI Done	This parameter enables the SPI done interrupt source to trigger the interrupt output.
SPI Bus Error	This parameter enables the SPI bus error interrupt source to trigger the interrupt output.
RX FIFO not Empty	This parameter enables the RX FIFO not-empty interrupt source to trigger the interrupt output.
RX FIFO Above Level	This parameter enables the RX FIFO above-level interrupt source to trigger the interrupt output.
RX FIFO Full	This parameter enables the RX FIFO full interrupt source to trigger the interrupt output.
RX FIFO Overflow	This parameter enables the RX FIFO overflow interrupt source to trigger the interrupt output.
RX FIFO Underflow	This parameter enables the RX FIFO underflow interrupt source to trigger the interrupt output.
TX FIFO Empty	This parameter enables the TX FIFO empty interrupt source to trigger the interrupt output.
TX FIFO Below Level	This parameter enables the TX FIFO below-level interrupt source to trigger the interrupt output.
TX FIFO not Full	This parameter enables the TX FIFO not-full interrupt source to trigger the interrupt output.
TX FIFO Overflow	This parameter enables the TX FIFO overflow interrupt source to trigger the interrupt output.
TX FIFO Underflow	This parameter enables the TX FIFO underflow interrupt source to trigger the interrupt output.

Pins Tab

This tab contains the Interrupt configuration settings.



Parameter Name	Description
Show SPI Terminals	This parameter removes internal pins and expose signals to terminals. These terminals must be connected to the pins or SmartIO Component.
Remove SCLK	This option allows a removal of the SCLK pin from the SPI interface.
Remove MOSI	This option allows a removal of the MOSI pin from the SPI interface.
Remove MISO	This option allows a removal of the MISO pin from the SPI interface.

Application Programming Interface

The Application Programming Interface (API) is provided by the SCB driver module from the PDL. The driver is copied into the “pdl\drivers\peripheral\scb\” directory of the application project after a successful build.

Refer to the PDL documentation for a detailed description of the complete API. To access this document, right-click on the Component symbol on the schematic and choose the “**Open PDL Documentation…**” option in the drop-down menu.

The Component generates the configuration structures and base address described in the Global Variables and Preprocessor Macros sections. Pass the generated data structure and the base address to the associated SCB driver function in the application initialization code to configure the peripheral. Once the peripheral is initialized, the application code can perform run-time changes by referencing the provided base address in the driver API functions.

By default, PSoC Creator assigns the instance name SPI_1 to the first instance of a Component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol.

Global Variables

The SCB_SPI_PDL Component populates the following peripheral initialization data structure(s). The generated code is placed in C source and header files that are named after the instance of the Component (e.g., *SPI_1.c*). Each variable is also prefixed with the instance name of the Component.

cy_stc_scb_spi_config_t const SPI_1_config

The instance-specific configuration structure. The pointer to this structure should be passed to Cy_SCB_SPI_Init function to initialize Component with GUI selected settings.

cy_stc_scb_spi_context_t SPI_1_context

The instance-specific context structure. It is used for internal configuration and data keeping for the SPI. The user should not modify anything in this structure. If only Low Level functions are used this is not needed.

Preprocessor Macros

The SCB_SPI_PDL Component generates the following preprocessor macro(s). Note that each macro is prefixed with the instance name of the Component (e.g. “SPI_1”).

#define SPI_1_HW ((CySCB_Type *) SPI_1_SCB__HW)

The pointer to the base address of the hardware

#define SPI_1_SPI_SLAVE_SELECT0 (SPI_1_SCB__SS0_POSITION)

The slave select line 0 constant that takes into account pin placement

#define SPI_1_SPI_SLAVE_SELECT1 (SPI_1_SCB__SS1_POSITION)

The slave select line 1 constant that takes into account pin placement

#define SPI_1_SPI_SLAVE_SELECT2 (SPI_1_SCB__SS2_POSITION)

The slave select line 2 constant that takes into account pin placement

#define SPI_1_SPI_SLAVE_SELECT3 (SPI_1_SCB__SS3_POSITION)

The slave select line 3 constant that takes into account pin placement

Data in RAM

The generated data may be placed in flash memory (const) or RAM. The former is the more memory-efficient choice if you do not wish to modify the configuration data at run-time. Under the **Built-In** tab of the Configure dialog set the parameter CONST_CONFIG to make your selection. The default option is to place the data in flash.

Interrupt Service Routine

Interrupt processing is optional for the SCB_SPI_PDL Component; therefore, the Component provides a parameter to choose between Internal and External placement of the Interrupt Component:

- Internal: This means that the Interrupt Component is placed inside the SCB_SPI_PDL Component. This allows use of SPI High-Level communication cy_scb driver functions. However, you must configure the interrupt controller to call the cy_scb driver SPI ISR, and you must enable the corresponding interrupt. The ISR must call Cy_SCB_SPI_Interrupt function that implements cy_scb driver SPI ISR functionality. Refer to the code example in the Quick Start section.
- External: This means that you are responsible for interrupt processing. There is no internal Interrupt Component. Instead, an output terminal is provided to connect to an external Interrupt Component. Use the **Interrupt Source** parameters to configure one or more sources to trigger the interrupt output. This external option allows you to implement your own interrupt handler. When this option is chosen only SPI Low-Level communication functions from the cy_scb driver, can be used. You can also leave the output terminal not connected and the Component will not process an interrupt at all.

Refer to cy_scb driver documentation for the list of SPI High-Level and Low-Level communication API.

Code Examples and Application Notes

Code Examples

PSoC Creator provides access to code examples in the Code Example dialog. For Component-specific examples, open the dialog from the Component Catalog or an instance of the Component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the "Code Example" topic in the PSoC Creator Help for more information.

There are also numerous code examples that include schematics and example code available online at the Cypress Code Examples web page.

Application Notes

Cypress provides a number of application notes describing how PSoC can be integrated into your design. You can access the Cypress Application Notes search web page at www.cypress.com/appnotes.

Functional Description

Data Rate Configuration

This Component must meet the data rate requirement of the connected SPI bus. This Component provides the following modes:

Slave Mode

The frequency of the connected clock source is the only configurable parameter used in determining the maximum data rate at which the slave can operate. The connected clock is the clock that runs the SCB hardware. The frequency of the connected clock source must be fast enough to provide enough oversampling to capture the MOSI signal and drive the MISO signal.

The Component provides the following methods to configure **Data Rate**:

- Set the desired **Data Rate**. This option uses a clock that is internal to the Component (this clock still uses clock divider resources). Based on the data rate, the Component asks PSoC Creator to create a clock with a frequency to satisfy data rate requirements. Refer to the Clock Selection section for information about how the clock frequency is calculated.
- Connect a user-configurable clock to the Component. This option is controlled by the **Enable Clock from Terminal** parameter, which must be enabled. In this mode, you must ensure the connected clock frequency provides enough oversampling.

Regardless of the chosen method, the Component will display the **Actual data rate**. This value assumes the specific external master configuration and parameters; therefore, the actual data rate at which your system can operate might be less than displayed. Refer to the Clock Selection section for information about the external master configuration and parameters. Refer also to the Actual Data Rate Calculations section.

Master Mode

The data rate is determined by the connected clock source and the oversample factor. These two factors are used to set the frequency of SCLK. One SCLK period is equal to the period of the connected clock multiplied by the oversample factor.

When the oversample is even, the first and second phase of the clock period are the same. Otherwise, the first phase of the clock signal period is one Component clock cycle longer than the second phase. The level of the first phase of the clock period depends on CPOL settings: 0 = low level and 1 = high level.

The Component provides the following methods to configure **Data Rate**:

- Set the desired **Data Rate and Oversample**. This option uses a clock that is internal to the Component (this clock still uses clock divider resources). The Component asks PSoC Creator to create a clock with a frequency equal to (Data Rate * Oversample) in kHz. Iterate over the **Oversample** parameter to get actual date rate closest to the desired.
- Connect a user-configurable clock to the Component. This option is controlled by the **Enable Clock from Terminal** parameter, which must be enabled. You must also configure the **Oversample** parameter. This method provides full control of the data rate configuration.

Regardless of the chosen method, the Component will display the **Actual data rate**. This value does not take to account such factors as PCB delays and slave parameters. Therefore, the actual data rate at which your system can operate might be less than displayed. Refer to the Actual Data Rate Calculations section for more information.

Actual Data Rate Calculations

The Component calculates the actual data rate for master or slave devices. This value is based on Component parameters. The master ignores parameters of the external slave and assumes PCB delays are 0. The slave assumes a specific external master configuration.

The main factor limiting the maximum data rate between master and slave is the round-trip path delay. This delay is the length of time it takes for a master SCLK signal to be sent to the slave plus the length of time it takes for a slave to assign the MISO signal, which must be sampled by the master. The master MISO setup time must be met to sample the MISO signal properly. The following equation specifies the round-trip path delay:

$$t_{ROUND_TRIP_DELAY} = t_{SCLK_PD_PCB} + t_{DSO} + t_{MISO_PD_PCB} + t_{DSI}$$

- $t_{SCLK_PD_PCB}$ is the PCB path delay of SCLK from the pin of the master device to the pin of the slave device.
- t_{DSO} is the time from the slave receiving the SPI clock edge at the pin to the MISO changing at the pin
- $t_{MISO_PD_PCB}$ is the PCB path delay of MISO from the pin of the slave device to the pin of the master device.
- t_{DSI} is the setup time of MISO signal to be sampled correctly by the master.

t_{DSI} is commonly listed in the master device datasheet. For the PSoC 6 device t_{DSI} is spec'd with late sample enabled. It must be noted that the spec is listing the setup time before the edge that would have been used if late sample was not enabled.

t_{DSO} is commonly listed in the slave device datasheet.

When $t_{ROUND_TRIP_DELAY}$ is calculated, the maximum communication data rate between master and slave can be defined as follows:

$$f_{SCLK(max)} = 1 / (2 * t_{ROUND_TRIP_DELAY})$$

The assumption is that the master samples the MISO signal one half SCLK period after the driving edge.

When the master can sample the MISO signal, one full SCLK period after the driving edge (late MISO sampling), the communication data rate is doubled and calculated as follows:

$$f_{SCLK(max)} = 1 / t_{ROUND_TRIP_DELAY}$$

Refer the Oversampling and Bit Rate sub-section within the Serial Peripheral Interface (SPI) section in the device TRM for more information about MISO sampling by the master device.

Clock Selection

The SCB_SPI_PDL Component provides an **Enable Clock from Terminal** parameter to choose between an internally configured clock (by the Component) or an externally configured clock for Component operation:

- Internally configured means that the Component is responsible for clock configuration. It requests the system to provide the clock frequency necessary to operate with selected the data rate.
  - Slave mode: the **Data Rate** parameter defines the clock frequency as follows:

    $f_{clk_scb} = N / ((0.5 * t_{clk_scb}) – 20 \text{ nsec} - t_{DSI}$
    - N is 3 when "Enable Input Glitch Filter" is false and 4 when true.
    - t_{DSI} Is external master delay which is assumed to be 16.66 nsec
    - The master captures data on the half SCLK period after the driving edge.

If the external master is capable of capturing the data one full SCLK cycle later than the displayed number can be doubled.

- Master mode: the **Oversample** and **Data Rate** parameters define clock frequency in kHz as (Data Rate * Oversample).

- Externally configured means that Component provides clock terminal for Clock Component connection. You must then configure the Clock Component appropriately.

For more information about SPI clock configuration, refer to the Serial Peripheral Interface (SPI) Oversampling and Bit Rate sub-section in the device TRM.

DMA Support

The SCB_SPI_PDL Component supports Direct Memory Access (DMA) transfers. The Component may transfer to/from the following sources.

Name of DMA Source / Destination	Length	Direction	DMA Req Signal	DMA Req Type	Description
RX_FIFO_RD	Word	Source	rx_dma	Level sensitive [2]	The data available in the RX FIFO is copied into place defined by destination. The DMA request signal is active while the number of data elements in the RX FIFO is greater than the value of RX FIFO Level.
TX_FIFO_WR	Word	Destination	tx_dma	Level sensitive [2]	The available data from source is copied into the TX FIFO. The DMA request signal is active while the number of data elements in the TX FIFO is less than the value of the TX FIFO Level.

Low-Power Modes

The SCB_SPI_PDL Component requires specific processing to support Deep Sleep and Hibernate modes. The cy_scb driver module provides callback functions to handle power mode transition. These functions must be registered using the cy_syspm driver before entering Deep Sleep or Hibernate mode appropriately. Refer to the Low Power Support section of the cy_scb driver, or the System Power Management section of the PDL Documentation for more details about callback registration.

The Slave is capable of waking up the device from Deep Sleep mode on slave select event. To enable this functionality, select the Enable Wakeup from Deep Sleep Mode parameter and register the Deep Sleep callback.

The Master is **not** able to wake up the device from Deep Sleep mode.

---

[2] To properly handle DMA level request signal activation and de-activation from the SCB peripheral block, the DMA Descriptor typically must be configured to re-trigger after 16 Clk_Slow cycles.

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the Component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator Components
- specific deviations – deviations that are applicable only for this Component

This section provides information on Component-specific deviations. Refer to PSoC Creator Help > Building a PSoC Creator Project > Generated Files (PSoC 6) for information on MISRA compliance and deviations for files generated by PSoC Creator.

The SCB_SPI_PDL Component does not have any specific deviations.

This Component uses firmware drivers from the cy_scb, cy_sysint, and cy_gpio PDL modules. Refer to the PDL documentation for information on their MISRA compliance and specific deviations.

This Component has the following embedded Components: clock, interrupt and pin. Refer to the corresponding Component datasheets for information on their MISRA compliance and specific deviations.

Registers

Refer to the Serial Communication Block Registers section in the device TRM.

Resources

The SCB_SPI_PDL Component uses single SCB peripheral block configured for the SPI operation.

DC and AC Electrical Characteristics

Note Final characterization data for PSoC 6 devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site.

Component Changes

This section lists the major changes in the Component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
2.0.b	Updated the datasheet.	Added Low-Power Modes section.
2.0.a	Minor datasheet edits.	
2.0	Updated the underlying version of PDL driver.	
1.0.a	Datasheet updates.	
1.0	Initial Version	

© Cypress Semiconductor Corporation, 2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical Components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical Component is any Component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.