eSi-RISC

PROCESSOR IP

# eSi-SPI

# 1 Contents

# 2 Overview

The eSi-SPI core can be used to implement full-duplex, synchronous, serial communications. It supports the following features:

- Master or slave operation.
- Programmable word size (1 to 32 bits).
- Programmable bit ordering (MSB first / LSB first).
- Programmable clock polarity (CPOL) and phase (CHPA).
- Programmable bit rate.
- Automatic and manual chip-select generation.
- Multiple chip-select outputs.
- Configurable TX and RX FIFO.
- Configurable support for parallel as well as serial transfers.
- Supports multi-master and multi-slave operation.
- Auto TX and RX to reduce bus bandwidth requirements.
- AMBA 3 APB slave interface.
- DMA flow-control interface.

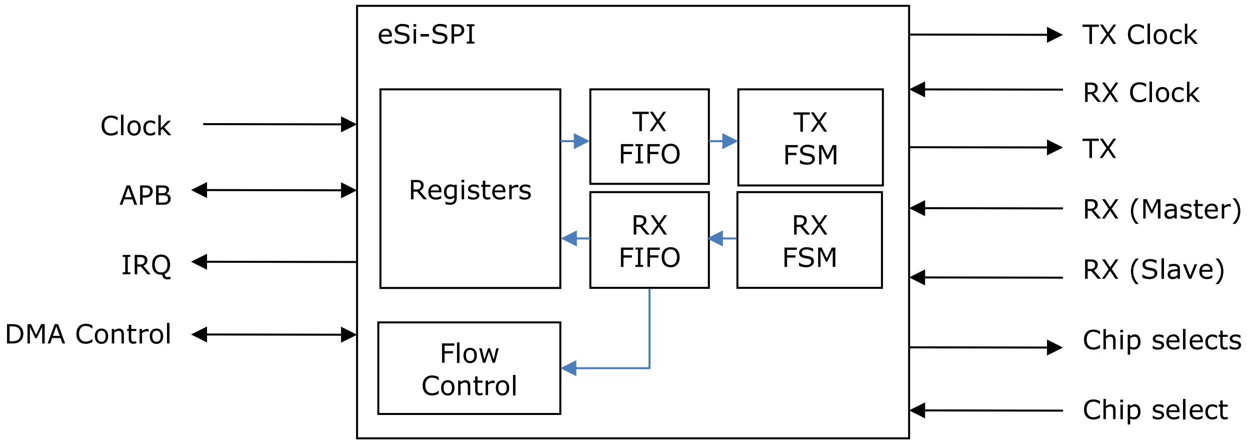

**Figure 1: eSi-SPI**

# 3 Hardware Interface

| | |
|---|---|
| **Module Name** | `esi_apb_spi` |
| **HDL** | Verilog |
| **Technology** | Generic |
| **Source Files** | esi_apb_spi.v, esi_fifo.v, esi_async_small_fifo.v, esi_peripheral_flow_control.v, esi_sync.v, esi_sync_n.v, esi_global_include.v, esi_global_cfg_include.v, esi_i2c_cfg_include.v |

| Configuration Option | Values | Description |
|---|---|---|
| `ASYNC_SLAVE` | `TRUE`, `FALSE` | Determines whether the slave interface is implemented asynchronously (`sck_in` is used as a clock) or synchronously (`sck_in` is re-sampled to `clk`) |

**Table 1: Configuration Options – Defined in esi_spi_cfg_include.v**

| Port | Type | Description |
|---|---|---|
| `tx_fifo_depth` | Integer | Specifies the depth of the TX FIFO. Minimum of 2 and value must be a power of 2 |
| `rx_fifo_depth` | Integer | Specifies the depth of the RX FIFO. Minimum of 2 and value must be a power of 2 |
| `fifo_width` | Integer | Specifies the width of the FIFOs. Valid range is [1,`BITS`] |
| `data_pins` | Integer | Specifies the width of the `tx_data` and `rx_data` busses. Valid range is [1, `fifo_width`] |
| `chip_select_pins` | Integer | Specifies the width of the `tx_chip_select_n` bus. Valid range is [1, `BITS`] |
| `sckz_reset` | Bit | Reset value for the `io_control.sckz` register |
| `txdz_reset` | Bit | Reset value for the `io_control.txdz` register |
| `csz_reset` | Bit | Reset value for the `io_control.csz` register |
| `apb_data_width` | Integer | Width of APB data bus |
| `apb_address_width` | Integer | Width of APB address bus |

**Table 2: Parameters**

| Port | Direction | Width | Description |
|---|---|---|---|
| `clk` | Input | 1 | Clock used for serial data transmission and reception. This clock must be enabled when `cactive` is asserted. This clock will be divided by the value in the `cycles_per_bit` register. |
| `pclk` | Input | 1 | APB clock |
| `presetn` | Input | 1 | APB reset, active-low |
| `paddr` | Input | `apb_address_width` | APB address. Only 8 LSBs are used |
| `psel` | Input | 1 | APB slave select |
| `penable` | Input | 1 | APB enable |
| `pwrite` | Input | 1 | APB write |
| `pdebug` | Input | 1 | APB noninvasive debug read |
| `pwdata` | Input | `apb_data_width` | APB write data |
| `rx_data` | Input | `data_pins` | Receive data in master mode |
| `rx_data_slave` | Input | `data_pins` | Receive data in slave mode |
| `sck_in` | Input | 1 | Receive clock. Only used in slave |

| | | | |
|---|---|---|---|
| | | | mode |
| `rx_chip_select_n` | Input | 1 | Receive chip select, active-low. Only used in slave mode |
| `tx_ack` | Input | `1` | Acknowledges `tx_ready` after transfer complete |
| `rx_ack` | Input | `1` | Acknowledges `rx_ready` after transfer complete |
| `cactive` | Output | 1 | Indicates `clk` should be active |
| `pready` | Output | 1 | APB ready |
| `prdata` | Output | `apb_data_width` | APB read data |
| `pslverr` | Output | 1 | APB slave error |
| `slave` | Output | 1 | Indicates SPI is in slave mode, rather than master mode |
| `sck_out` | Output | 1 | Transmit clock. Only valid in master mode |
| `sck_out_enable` | Output | 1 | Indicates `sck` should be an output when high (master mode) or an input or high-Z when low (slave mode or when disabled) |
| `tx_data` | Output | `data_pins` | Transmit data |
| `tx_data_out_enable` | Output | `1` | Indicates `tx_data` should be driven outputs when high (master mode or slave mode and `rx_chip_select_n` asserted) or high-Z when low |
| `tx_chip_select_n` | Output | `chip_select_pins` | Transmit chip select, active-low. Only used in master mode |
| `tx_chip_select_n_out_enable` | Output | `1` | Indicates `tx_chip_select_n` should be driven outputs when high or high-Z when low |
| `interrupt_n` | Output | 1 | Interrupt request, active-low |
| `tx_ready` | Output | 1 | Indicates device can accept new data |
| `rx_ready` | Output | 1 | Indicates device has data to be read |

**Table 3: I/O Ports**

For complete details of the APB signals, please refer to the AMBA 3 APB Protocol v1.0 Specification available at:

http://www.arm.com/products/system-ip/amba/amba-open-specifications.php

The SPI does not include internal synchronizing flip-flops. These should be implemented externally for the `rx_data_slave`, `rx_chip_select_n` and `sck_in` ports if the transmitting clock domain is asynchronous to `clk` and `ASYNC_SLAVE` is `FALSE`. For maximum slave performance, synchronizers should not be implemented when `ASYNC_SLAVE` is `TRUE`.

# 4 Software Interface

## 4.1 Register Map

| Register | Address offset | Access | Description |
|---|---|---|---|
| tx_data | 0x00 | W | Transmit register |
| rx_data | 0x04 | R | Receive register |
| status | 0x08 | R/W | Status register |
| control | 0x0c | R/W | Control register |
| cycles_per_bit | 0x10 | R/W | Cycles per bit register |
| txae_thresh | 0x14 | R/W | Transmit FIFO almost empty threshold |
| rxaf_thresh | 0x18 | R/W | Receive FIFO almost full threshold |
| tx_count | 0x1c | R | Count of entries used in TX FIFO |
| rx_count | 0x20 | R | Count of entries used in RX FIFO |
| auto_trx_control | 0x24 | R/W | Auto TRX control register |
| auto_trx_count | 0x28 | R/W | Auto TRX count register |
| cs_control | 0x2c | R/W | Chip select control register |
| cs_enable | 0x30 | R/W | Chip select enable register |
| cs_assert_bits | 0x34 | R/W | Chip select assert bits register |
| cs_deassert_bits | 0x38 | R/W | Chip select deassert bits register |
| cs_idle_bits | 0x3c | R/W | Chip select idle bits register |
| io_control | 0x40 | R/W | I/O control register |

**Table 4: Register Map**

### 4.1.1 Transmit Data Register

Data to be transmitted over the serial interface should be written to the transmit register, in order to be written to the TX FIFO. The transmit data register should not be written to while the `TXF` bit in the `status` register is set, otherwise data loss may occur. The number of bits transmitted is determined by the `control.BPW` register.

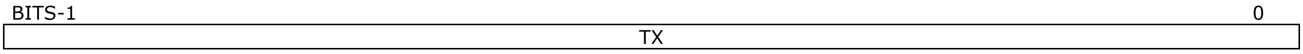

**Figure 2: Format of the `tx_data` register**

### 4.1.2 Receive Data Register

Data that is received over the serial interface and stored in the RX FIFO can be read in the receive register. A read from the received data register will remove the data from the RX FIFO, unless `pdebug` is high. Setting `pdebug` high allows a debugger to display the first element in the RX FIFO, without affecting the program being debugged.

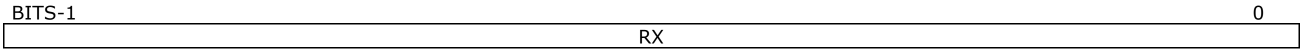

**Figure 3: Format of the `rx_data` register**

### 4.1.3 Status Register

The status register contains a selection of flags that indicate the current status of the SPI. To clear a bit in the status register, write a 1 to it. Writing 0 will leave it unchanged.

| 15 | 14 | 13 | 12 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RXAF | TXAE | B | - | | RXO | RXF | RXE | TXU | TXO | TXF | TXE |

**Figure 4: Format of the `status` register**

| Register | Values | Description |
|---|---|---|
| TXE | 0 - Not empty<br>1 - Empty | Transmit FIFO empty |
| TXF | 0 - Not full<br>1 - Full | Transmit FIFO full |
| TXO | 0 - No overflow<br>1 - Overflow | Transmit FIFO overflow. Sticky |
| TXU | 0 - No underrun<br>1 - Underrun | Transmit FIFO underrun. Sticky. Not valid for asynchronous slave |
| RXE | 0 - Not empty<br>1 - Empty | Receive FIFO empty |
| RXF | 0 - Not full<br>1 - Full | Receive FIFO full |
| RXO | 0 - No overflow<br>1 - Overflow | Receive FIFO overflow. Sticky |
| B | 0 – Idle<br>1 – Busy | Indicates whether the SPI bus is busy. A chip-select should not be deasserted until this indicates idle |
| TXAE | 0 – Not almost empty<br>1 – Almost empty | Transmit FIFO almost empty |
| RXAF | 0 – Not almost full<br>1 – Almost full | Receive FIFO almost full |

**Table 5: Fields of the `status` register**

### 4.1.4 Control Register

The control register contains a selection of flags that control the operation of the SPI.

| 15 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BPW | | P | RF | RXIE | TXIE | CPOL | CPHA | BO | WS | M | E |

**Figure 5: Format of the `control` register**

| Register | Values | Description |
|---|---|---|
| E | 0 - Disabled<br>1 - Enabled | Enables the SPI. When disabled, data will not be received or transmitted |
| M | 0 - Master<br>1 - Slave | Operating mode |
| WS | 0 - 8-bits<br>1 - 16-bits | Word size. Only used if `control.BPW` equals 0 |
| BO | 0 - MSB first<br>1 - LSB first | Bit ordering |
| CPHA | 0 - Sample on leading edge<br>1 - Sample of trailing edge | Clock phase |
| CPOL | 0 - Idle low<br>1 - Idle high | Clock polarity |
| TXIE | 0 - Disabled | Transmit interrupt enable |

| | | |
|---|---|---|
| | 1 - Enabled | |
| `RXIE` | 0 - Disabled<br>1 - Enabled | Receive interrupt enable |
| `RF` | 0 – Do not reset<br>1 – Reset FIFOs | Reset FIFOs. When written with a 1, transmit or receive FIFOs will be cleared. This bit clears automatically |
| `P` | 0 – Serial<br>1 – Parallel | When `data_pins` is greater than 1, this controls whether data is transmitted serially (only on the least significant bit of the `tx_data` or `rx_data` bus) or in parallel |
| `BPW` | 0 – `BITS` | Bits per word. Specifies the number of bits that are transmitted and received. For parallel transfers, this value needs to be divisible by `data_pins` |

**Table 6: Fields of the `control` register**

### 4.1.5 Cycles Per Bit Register

The cycles per bit register is a 16-bit integer that determines how many cycles of the clock, `clk`, the transmit clock, `sck_out`, is held high for and low for, when in master mode. Use of a 16-bit register provides support for a wide range of clock frequencies and bit rates. The minimum value for `cycles_per_bit` is 2. When set to 2, each high and low phase of `sck_out` is 1 `clk` cycle, resulting in a total period of `cycles_per_bit`. When set to an odd value, the low phase will last for an extra cycle.

The `cycles_per_bit` register is not used in slave mode. When `ASYNC_SLAVE` is `FALSE`, the maximum `sck_in` frequency is 1/8th of `clk`. When `ASYNC_SLAVE` is `TRUE` `sck_in` may run at higher frequencies.

| 15 | 0 |
|---|---|
| | |

**Figure 6: Format of the `cycles_per_bit` register**

### 4.1.6 Transmit FIFO Almost Empty Threshold Register

The transmit FIFO almost empty threshold register sets the count of used entries in the transmit FIFO, below which, the `status.TXAE` flag and `tx_ready` will be set.

| 15 | | log2(tx_fifo_depth) | 0 |
|---|---|---|---|
| AEIE | - | TH | |

**Figure 7: Format of the `txae_thresh` register**

| Register | Values | Description |
|---|---|---|
| `TH` | 0 – `tx_fifo_depth` | Almost empty threshold |
| `AEIE` | 0 – Interrupt disabled<br>1 – Interrupt enabled | Almost empty interrupt enable |

**Table 7: Fields of the `txae_thresh` register**

### 4.1.7 Receive FIFO Almost Full Threshold Register

The receive FIFO almost full threshold register sets the count of used entries in the receive FIFO, above which, the `status.RXAF` flag and `rx_ready` will be set.

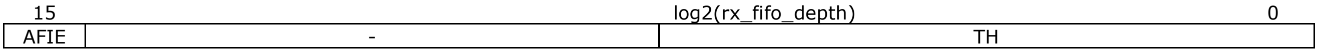

**Figure 8: Format of the `rxaf_thresh` register**

| Register | Values | Description |
|---|---|---|
| TH | 0 – `rx_fifo_depth` | Almost full threshold |
| AFIE | 0 – Interrupt disabled<br>1 – Interrupt enabled | Almost full interrupt enable |

**Table 8: Fields of the `rxaf_thresh` register**

### 4.1.8 Transmit FIFO Count Register

The transmit FIFO count register indicates the count of used entries in the transmit FIFO.

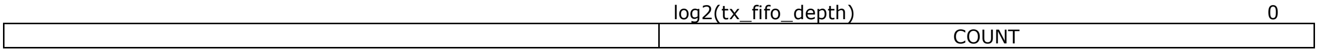

**Figure 9: Format of the `tx_count` register**

### 4.1.9 Receive FIFO Count Register

The receive FIFO count register indicates the count of used entries in the receive FIFO.

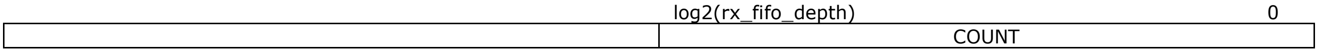

**Figure 10: Format of the `rx_count` register**

### 4.1.10 Auto TRX Control Register

The auto transmit / receive control register contains a selection of flags that control automatic transmit or receive by the SPI. When `auto_trx_control.AT`=1, the TX FIFO will have the value 0 written to it, the number of times specified by the `auto_trx_count` register. When `auto_trx_control.AT`=2, the RX FIFO will be read the number of times specified by the `auto_trx_count` register and the values will be discarded. This feature can be used to eliminate APB accesses to the corresponding FIFOs, when unidirectional transfers are being performed. When in auto-transmit master mode, the SPI will not transmit while the RX FIFO is full, to avoid data loss.

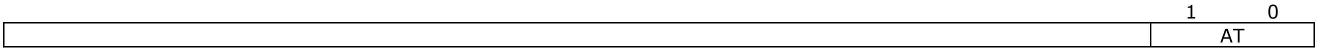

**Figure 11: Format of the `auto_trx_control` register**

| Register | Values | Description |
|---|---|---|
| AT | 0 – Disabled<br>1 – Auto-transmit | Enables auto-transmit of auto-receive |

| | | |
|---|---|---|
| | 2 – Auto-receive | |

**Table 9: Fields of the `auto_trx_control` register**

### 4.1.11 Auto TRX Count Register

The auto transmit / receive count register indicates how many times the TX FIFO will be written (when `auto_trx_control.AT`=1) or the RX FIFO will be read (when `auto_trx_control.AT`=2).

| BITS-1 … 0 |
|---|
| COUNT |

**Figure 12: Format of the `auto_trx_count` register**

### 4.1.12 Chip Select Control Register

The chip select control register provides registers to control the operation of the chip select outputs, `tx_chip_select_n`.

| | 1 | 0 |
|---|---|---|
| - | M | S |

**Figure 13: Format of the `cs_control` register**

| Register | Values | Description |
|---|---|---|
| `S` | 0 – Active low<br>1 – Active high | Sense of chip-select outputs |
| `M` | 0 – Automatic<br>1 – Manual | Whether the chip-selects are asserted automatically or manually |

**Table 10: Fields of the `cs_control` register**

### 4.1.13 Chip Select Enable Register

The chip select enable register is a bitmask of `chip_select_pins` bits that determines which chip-select outputs will be asserted when a transfer occurs. Typically, as SPI transfers are bi-directional, this should just be a single bit, to ensure that contention does not occur on the receive data signals. However, if contention is prevented at a system level, it is possible to set multiple bits in this register to transmit data simultaneously to multiple slaves.

| `chip_select_pins`-1 | | 2 | 1 | 0 |
|---|---|---|---|---|
| En | … | E2 | E1 | E0 |

**Figure 14: Format of the `cs_enable` register**

| Register | Values | Description |
|---|---|---|
| `En` | 0 – Disabled<br>1 – Enabled | Determines whether the corresponding chip select output with be asserted when a transfer occurs |

**Table 11: Fields of the `cs_enable` register**

### 4.1.14 Chip Select Assert Bits Register

The chip select assert bits register determines how many bit periods the chip select outputs, `tx_chip_select_n`, will be asserted for before outputting the clock or data, when in master mode (`control.M` equals 0) and automatic chip select control (`cs_control.M` equals 0).

|  | 7 | 0 |
|---|---|---|
|  | AB |  |

**Figure 15: Format of the `cs_assert_bits` register**

### 4.1.15 Chip Select Deassert Bits Register

The chip select deassert bits register determines how many bit periods after the data is transmitted, before the chip select outputs, `tx_chip_select_n`, will be deasserted, when in master mode (`control.M` equals 0) and automatic chip select control (`cs_control.M` equals 0).

|  | 7 | 0 |
|---|---|---|
|  | DB |  |

**Figure 16: Format of the `cs_deassert_bits` register**

### 4.1.16 Chip Select Idle Bits Register

The chip select idle bits register determines the minimum number of bit periods after the chip select outputs, `tx_chip_select_n`, are deasserted, before they will be asserted again, when in master mode (`control.M` equals 0) and automatic chip select control (`cs_control.M` equals 0). This value will be exceeded if data does not become available in the TX FIFO.

|  | 7 | 0 |
|---|---|---|
|  | IB |  |

**Figure 17: Format of the `cs_idle_bits` register**

### 4.1.17 IO Control Register

The I/O control register allows the outputs from the SPI to be put in a high-Z state when the SPI is disabled, to allow another SPI master control of the bus.

|  | 2 | 1 | 0 |
|---|---|---|---|
|  | CSZ | TXDZ | SCKZ |

**Figure 18: Format of the `io_control` register**

| Register | Values | Description |
|---|---|---|
| `SCKZ` | 0 – Driven<br>1 – High-Z | Determines whether `sck_out_enable` is asserted when `control.E` is 0, as thus whether the serial clock output is driven or high-Z when the SPI is disabled |
| `TXDZ` | 0 – Driven<br>1 – High-Z | Determines whether `tx_data_out_enable` is asserted when `control.E` is 0, as thus whether the transmit data lines are driven or high-Z when the SPI is disabled |
| `CSZ` | 0 – Driven | Determines whether |

| | | |
|---|---|---|
| | 1 – High-Z | tx_chip_select_n_out_enable is asserted when control.E is 0, as thus whether the chip-select outputs are driven or high-Z when the SPI is disabled |

**Table 12: Fields of the io_control register**

## 4.2 Interrupts

The SPI supports the following interrupts.

- Transmit interrupt
- Transmit FIFO almost empty interrupt
- Receive interrupt
- Receive FIFO almost full interrupt

The transmit interrupt will be raised when the transmit buffer is empty and the TXIE flag in the control register is set to 1. This indicates that the transmitter has no data to transmit.

The transmit FIFO almost empty interrupt will be raised when the transmit FIFO is almost empty as determined by tx_thresh.TH and the AEIE flag in the tx_thresh register is set to 1.

The receive interrupt will be raised when the receiver buffer is not empty and the RXIE flag in the control register is set to 1. This indicates that the receiver has received some data.

The receive FIFO almost full interrupt will be raised when the receive FIFO is almost full as determined by rx_thresh.TH and the AFIE flag in the rx_thresh register is set to 1.

## 4.3 Bit Ordering

The control.BO register determines the bit ordering of the data written from the FIFO as it is transmitted on the serial / parallel data bus.

| Value | data_pins | control.BO | Transmitted data |
|---|---|---|---|
| h17 | 1 | 0 – MSB | tx_data[0] = 0,0,0,1,0,1,1,1 |
| h17 | 1 | 1 – LSB | tx_data[0] = 1,1,1,0,1,0,0,0 |
| h17 | 4 | 0 – MSB | tx_data[0] = 0,0<br>tx_data[1] = 0,1<br>tx_data[2] = 0,1<br>tx_data[3] = 1,1 |
| h17 | 4 | 1 – LSB | tx_data[0] = 1,1<br>tx_data[1] = 1,0<br>tx_data[2] = 1,0<br>tx_data[3] = 0,0 |

**Table 13: Bit Ordering Examples**

## 4.4 Chip Select Generation

When automatic chip select generation is enabled (cs_control.M equals 0), the chip select outputs, tx_chip_select_n, are automatically asserted before transfers occur and are automatically deasserted when a transfer completes and the transmit FIFO is empty.

The time between the assertion of the chip select outputs and the transmission of data can be controlled by the cs_assert_bits register. The time between the transmission of the last bit of data and the deassertion of the chip select is controlled by the cs_deassert_bits register. A minimum delay between deassertion and reassertion of the chip selects is controlled by the cs_idle_bits register. Each of these registers are specified in bit periods, and thus are multiplied by the value in the cycles_per_bit register.

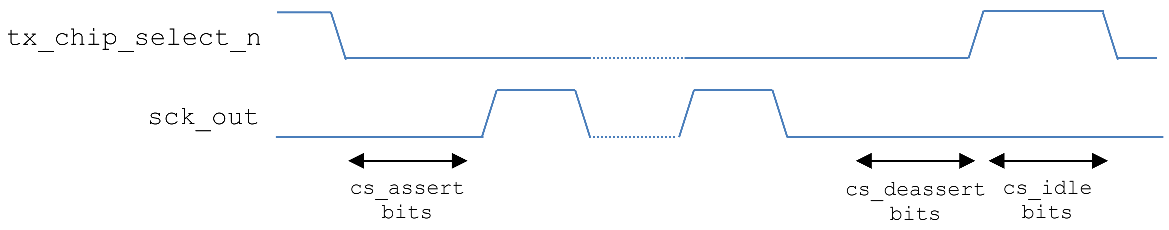

**Figure 19: Automatic Chip Select Generation Timing**

To keep throughput high, the chip select is not automatically deasserted between words, if there is more data in the transmit FIFO. If this is desired, the transmit FIFO should only be written with one word at a time, and the status.BUSY flag should be checked to be clear, before writing further data in to the transmit FIFO.

The chip select will be deasserted when a transfer is complete and there is no data in the transmit FIFO. If you wish for the chip select to be asserted continuously, chip select control should be set to manual (cs_control.M equals 1). In this mode, the cs_assert_bits, cs_deasert_bits and cs_idle_bits registers are not used.

# 5 Revision History

| Hardware Revision | Software Release | Description |
|---|---|---|
| 1 | 1.0.0 | Initial release. |
| 2 | 2.3.2 | Added `tx_ack` and `rx_ack` ports. |
| 3 | 3.2.12 | Added TX and RX FIFOs.<br>Added `status.TXAE` field.<br>Added `status.RXAF` field.<br>Added `txae_thresh` register.<br>Added `rxaf_thresh` register.<br>Added `control.RF` field.<br>Added `control.BITS` field. |
| 4 | 3.3.9 | Added `tx_count` register.<br>Added `rx_count` register. |
| 5 | 4.0.2 | Added `auto_trx_control` register.<br>Added `auto_trx_count` register. |
| 6 | 4.1.1 | Added `status.B` field. |
| 7 | 5.0.4 | Added `data_pins` parameter.<br>Added `control.P` field.<br>Renamed `rx_data_master` as `rx_data`.<br>Renamed `rx_clk` as `sck_in`.<br>Rename `tx_clk` as `sck_out`.<br>Added `sck_out_enable` pin. |
| 8 | 5.0.7 | Added `ASYNC_SLAVE` configuration option.<br>Made `status.TXO` and `status.RXO` flag sticky.<br>`control.CPOL` and control.`CPHA` valid in slave mode as well as master mode. |
| 9 | 5.0.8 | Added `chip_select_pins` parameter.<br>Added `cs_control` register.<br>Added `cs_enable` register.<br>Added `cs_assert_bits` register.<br>Added `cs_deassert_bits` register.<br>Added `cs_idle_bits` register. |
| 10 | 6.0.2 | Added `pdebug` input. |
| 11 | 6.0.3 | `tx_ready` is driven by TX FIFO almost empty, rather than not full.<br>`rx_ready` is driven by RX FIFO almost full, rather than not empty. |
| 12 | 6.0.4 | Added `tx_data_out_enable` and `tx_chip_select_n_out_enable` ports.<br>Added `io_control` register. |

**Table 14: Revision History**