



Copyright © 2004-2007. Cypress Semiconductor. All Rights Reserved.

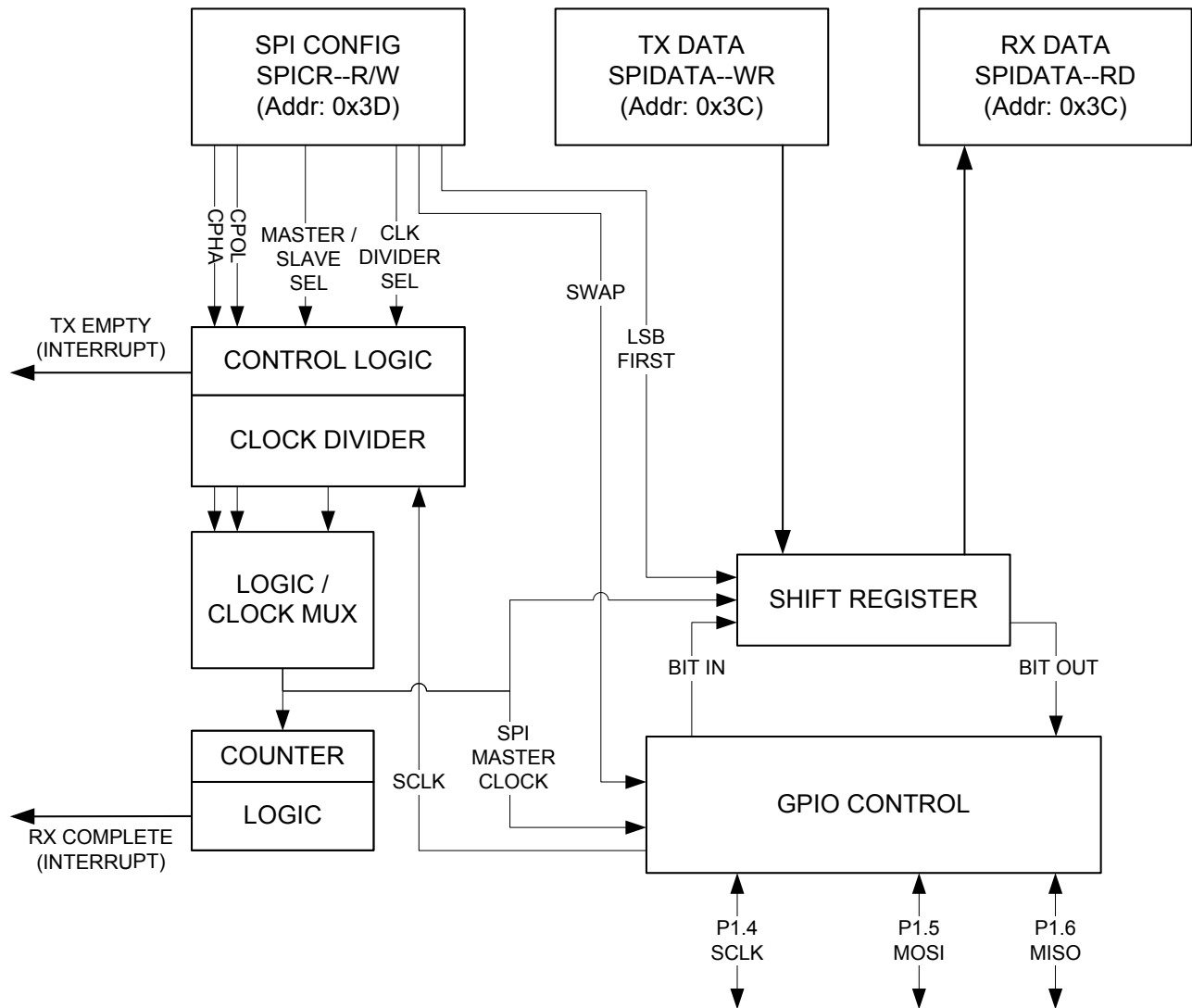
Resources	API Memory (Bytes)		Pins (per External I/O)
	Flash	RAM	
CY7C639/638/633/602/601xx			
MOSI/MISO/SCLK	38	0	3 - 4
SDIO/SCK	38	0	2 - 3

## Features and Overview

- Supports Serial Peripheral Interconnect (SPI) Master protocol
- Supports SPI clocking modes 0, 1, 2, and 3
- Programmable interrupt on SPI-done condition
- SPI Slave devices can be independently selected

The SPIM User Module is a Serial Peripheral Interconnect Master. It performs full duplex synchronous 8-bit data transfers. SCLK phase, SCLK polarity, and LSB First can be specified to accommodate most SPI clocking modes. Controlled by user supplied software, you are able to configure the slave select signal to control one or more SPI Slave devices.

The SPI User Module also supports two wire serial devices such as optical mouse sensors that employ an SDIO/SCK interface.



**SPIM Block Diagram**

## Parameters and Resources

### SPIPins

The SPIPins User Module Parameter selects the pins to be used by the SPI block. The parameter allows two wire (SDIO/SCK) or three wire (MOSI/MISO/SCLK) operation. If the selected device supports both 5 or 3.3 Volt operation, the parameter provides voltage selection as well.

The SPI Pins parameter programs the P14CR, P15CR, and P16CR configuration registers.

SPI Pins Choices	Voltage Options	
	CY7C639xx CY7C638xx CY7C633xx	CY7C601xx CY7C602xx
MOSI(P1.5)/MISO(P1.6)/SCLK(P1.4)	5V or 3.3V	N/A
SDIO(P1.5)/SCK(P1.4)	5V or 3.3V	N/A
The CY7C601xx and CY7C602xx devices do not offer dual voltage support for the SPI pins.		

### BitOrder

The BitOrder user module parameter selects the bit order for the SPI data stream. Selections are LSBFirst or MSBFirst.

The BitOrder user module parameter programs the LSB First field of the SPI Configuration Register (SPICR).

### CPOL

The CPOL user module parameter selects the idle state for the SPI Clock.

The CPOL user module parameter programs the CPOL field of the SPI Configuration Register (SPICR).

### CPHA

The CPHA user module parameter selects SPI Clock Phase on which the SPI data is sampled.

The CPHA user module parameter programs the CPHA field of the SPI Configuration Register (SPICR).

### ClockDivider

The ClockDivider user module parameter selects SPI Clock ClockDivider. The ClockDivider is applied to the CPUCLK to generate the SPI Clock.

ClockDivider	SCLK Select	SCLK Frequency when CPUCLK =	
		12 Mhz	24 Mhz
6	00	2 Mhz	4 Mhz
12	01	1 Mhz	2 Mhz
48	10	250 Khz	500 Khz
96	11	125 Khz	250 Khz

The ClockDivider user module parameter programs the SCLK Select field of the SPI Configuration Register (SPICR).

## Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow the designer to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the include files.

## SPIM\_Start

**Description:**

Starts the SPIM User Module. This function is provided for API consistency.

**C Prototype:**

```
void SPIM_Start(void)
```

**Assembly:**

```
call SPIM_Start
```

**Parameters:**

None

**Return Value:**

None

**Side Effects:**

The A and X registers may be altered by this function.

## SPIM\_Stop

**Description:**

Disables the SPIM module. This function is provided for API consistency.

**C Prototype:**

```
void SPIM_Stop(void)
```

**Assembly:**

```
call SPIM_Stop
```

**Parameters:**

None

**Return Value:**

None

**Side Effects:**

The A and X registers may be altered by this function.

## SPIM\_EnableInt

**Description:**

Enables the SPIM TX Buffer Full and RX Complete interrupts.

**Note** You can add custom ISR code to the ISRs found in the SPIMINT.asm file in the project lib directory.

**C Prototype:**

```
void SPIM_EnableInt(void)
```

**Assembly:**

```
call SPIM_EnableInt
```

**Parameters:**

None

**Return Value:**

None

**Side Effects:**

The A and X registers may be altered by this function.

## SPIM\_DisableInt

**Description:**

Disables the SPIM TX Buffer Full and RX Complete interrupts.

**C Prototype:**

```
void SPIM_DisableInt(void)
```

**Assembly:**

```
call SPIM_DisableInt
```

**Parameters:**

None

**Return Value:**

None

**Side Effects:**

The A and X registers may be altered by this function.

## SPIM\_SetMOSI

**Description:**

Sets the output pin for Master Out, Slave In.

**C Prototype:**

```
void SPIM_SetMOSI(BYTE bPin)
```

**Assembly:**

```
mov  A, bPin  
call SPIM_SetMOSI
```

**Parameters:**

BYTE bPin: 0x00--SPIM\_MOSI\_P15, 0x01--SPIM\_MOSI\_P16. It is passed in Accumulator.

**Return Value:**

None

**Side Effects:**

The A and X registers may be altered by this function.

## SPIM\_SetMISO

**Description:**

Sets the output pin for Master In, Slave Out

**C Prototype:**

```
void SPIM_SetMISO(BYTE bPin)
```

**Assembly:**

```
mov  A, bPin  
call SPIM_SetMISO
```

**Parameter:**

BYTE bPin: 0x00--SPIM\_MISO\_P16, 0x01--SPIM\_MISO\_P15. It is passed in Accumulator.

**Return Value:**

None

**Side Effects:**

The A and X registers may be altered by this function.

## SPIM\_bIO

### Description:

Transmits a single data byte and returns a received data byte from a slave device.

### C Prototype:

```
BYTE SPIM_bIO(void)
```

### Assembly:

```
call SPIM_bIO
mov  bRxDatA, A
```

### Parameters:

None

### Return Value:

Data byte received from the slave SPI is returned in the Accumulator.

### Side Effects:

The A and X registers may be altered by this function.

### Side Effects:

The status bits are cleared after this function is called. The A and X registers may be altered by this function.

## Sample Firmware Source Code

In C, use the Start API to begin operation and call the Stop API when done.

```
#include "SPIM.h"

SPIM_Start();
... // (application processing)
SPIM_Stop();
```

The following C code fragment shows reading from a SPI EEPROM for the MOSI/MISO/SCLK selection:

```
SPIM_Start();
SPIM_SetMOSI(SPIM_MOSI_P16); // Set MOSI to P1.6/MISO P1.5
EEPROM_Reset(); // Reset the EEPROM (user provided function)
SPI_SELECT(); // Assert slave select (user provided function)
SPIM_bIO(EEP_READ); // Send a read command
SPIM_bIO(0x00); // Address 0:0
SPIM_bIO(0x00);

for (i = 0; i < 32; ++i)
{
    aRD[i] = SPIM_bIO(0); // Read some data
}
SPI_UNSELECT(); // Deassert the slave select (user provided function)
```

The following C code fragment shows reading register 0x01 from a SDIO optical sensor using the SDIO/SCK selection:

```
SPIM_Start();
SPIM_SetMOSI(SPIM_MOSI_P15); // Set MOSI to P1.5
SPIM_bIO((0x01)); // Write the register number
timer_delay_100_usec(); // delay (user provided function)
```

```
SPIM_SetMISO(SPIM_MISO_P15); // Reverse the SDIO line
data = SPIM_bIO(0); // Read the register data
```

© Cypress Semiconductor Corporation, 2004-2007. The Source Code in this document is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and/or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.