I²C to SPI Bridge

AN49217

Author: Anu M D
Associated Project: Yes
Associated Part Family: CY8C29x66, CY8C27x43, CY8C21xxx
GET FREE SAMPLES HERE
Software Version: PSoC Designer™ 5.0
Associated Application Notes: None

Application Note Abstract

PSoC® is capable of interfacing with several communication protocols such as UART, SPI, USB, and I²C. It serves as a bridge between two buses. This application note demonstrates how PSoC's flexibility is used to configure it as an I²C to SPI bridge.

Introduction

I²C (Inter-Integrated-Circuit) is a multi-master serial communication bus protocol invented by Philips (now NXP). It is used to connect low speed peripherals on a motherboard, embedded system, or other systems with wired connections. Serial Peripheral Interface (SPI) is a synchronous high speed serial data link that operates in full duplex mode. PSoC's capability to manage both is used to make an I²C to SPI bridge.

Features

The following are some characteristics when using PSoC as an I²C to SPI bridge:

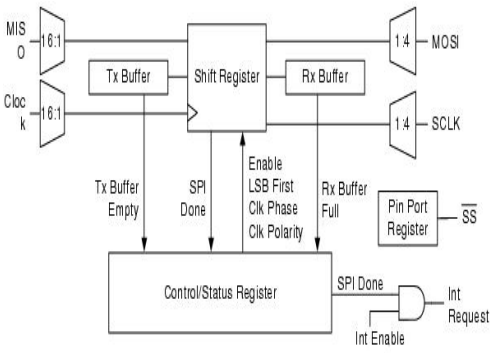
- I²C supports standard data rates of 100 kbps and 400 kbps; it also supports 50 kbps.
- SPI operates up to 4 Mbps.
- SPI clocking modes 0, 1, 2, and 3 are supported.
- Firmware selection of SPI clocking modes.
- Multiple slave select pins can be configured to connect multiple slaves.
- Configured for 3.3V or 5V operation.
- Programmable GPIOs.
- Additional digital and analog resources can be configured as PWMs, counters, timers, DMX512, IrDA, ADCs, DACs, and so on.
- Watch dog and sleep timers.
- Internal oscillator (option to also connect external crystal).
- Option to put in low power (sleep) mode when not in use.

General Description

The I²C bus is an industry standard, two-wire hardware interface developed by Philips for efficient inter IC communication. The master initiates all communication on the I²C bus and supplies the clock for all slave devices. The I2CHW User Module in PSoC implements an I²C device in hardware. This user module supports the standard mode with speeds up to 400 kbps. No digital or analog user blocks are consumed with this user module. The I2CHW User Module is compatible with other slave devices on the same bus.

The I²C block directly controls the data (SDA) and clock (SCL) signals to the external I²C interface through connections to two dedicated GPIO pins. The PSoC device firmware interacts with the I²C block through register reads and writes. Firmware synchronization is implemented through polling and interrupts.

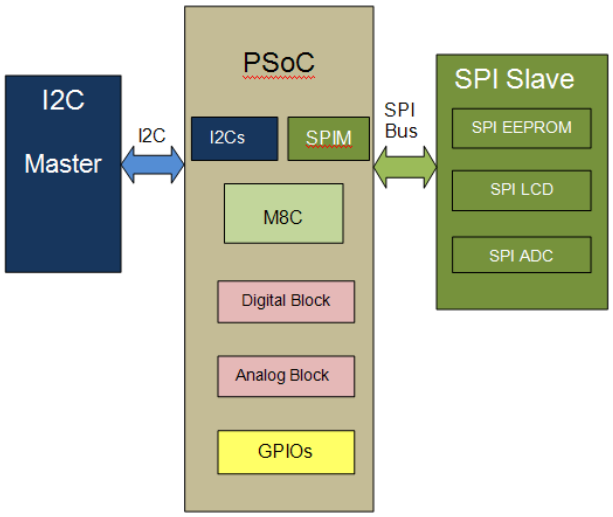
Figure 1. SPI Block Diagram



The SPIM User Module implements a Serial Peripheral Interconnect Master. It uses the Tx buffer, Rx buffer, Control and Shift registers of a digital communications type PSoC block and one or more Pin Port registers. The SPIM hardware transmits data from the master SPI device on the MOSI (Master Out, Slave In) signal and simultaneously receives data from the selected slave SPI device on the MISO (Master In, Slave Out) signal. The same SCLK signal is used for both transmit and receive of the master and slave data. The SPI User Module block diagram is shown in Figure 1.

The block level representation of an I^2C to SPI bridge application is shown in Figure 2.

Figure 2. I^2C Slave to SPI Master Block Diagram



Functional Description

PSoC is configured as an I^2C slave and SPI master. This is because it can receive data from an I^2C master and send this data to an SPI slave device. Also, the data received over the MISO line is stored so that it can be read by the I^2C master during the next I^2C read. The additional digital and analog blocks in the device are configured to function as PWMs, timers, counters, ADCs, DACs, comparators, or other communication peripherals such as IrDA and UART.

Multiple slave select lines can be configured and the appropriate slave selected, using firmware. This makes it possible for a single I^2C master to communicate with multiple SPI slave devices.

Logic can also be incorporated in the firmware to analyze the incoming stream of data and act intelligently. When the M8C receives data over I^2C, it analyzes the data and takes necessary actions depending on the command received.

RTC Example

The example project accompanying this application note demonstrates how to configure PSoC as an I^2C to SPI bridge and interface to a Dallas DS1305 Real Time Clock (RTC).

I^2C to SPI Bridge

The bridge works as an I^2C slave and an SPI master. It receives commands from the I^2C master and then writes or reads data to the SPI slave depending on the command received.

Placement

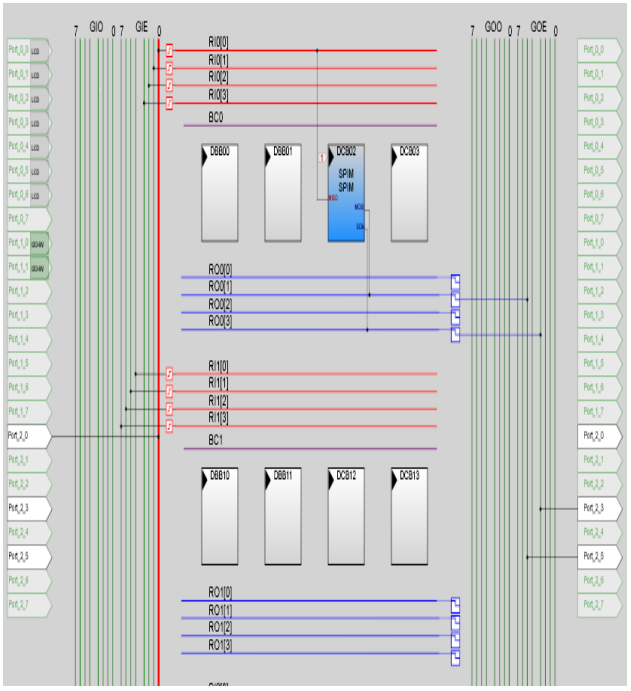
The I2CHW User Module does not consume any digital or analog blocks. The SPIM User Module consumes only one digital communication block. Therefore, any PSoC part starting from CY8C21xxx is used to make an I^2C to SPI bridge.

Routing

There are only two options for routing the signals to the I2CHW User Module. The SDA and SCL lines are routed to either (P1[0], P1[1]) or (P1[5], P1[7]). The SDA and SCL lines must be pulled up externally.

The SPI signals can be routed to any of the GPIOs using the various row and global interconnects.

Figure 3. Interconnect View



In the example project, the SDA and SCL signals are routed to P1[5] and P1[7]. P1[0] and P1[1] are not recommended for I^2C unless it is unavoidable. This is because these two pins are shared with the ISSP function; interactions between the two can cause problems. The MISO signal is routed to P2[0] through the global net GIE0 and row input net RI0[0]. The MOSI signal is routed to P2[5] through RO0[1] and GOE5. Similarly, SClk is routed

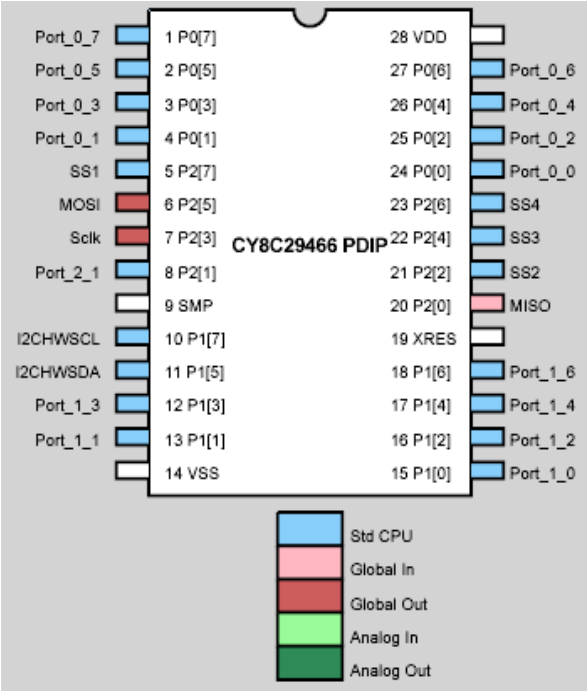
to P2[3]. GPIO P2[7] is configured as the default slave select (SS1) pin.

Any GPIO can be configured as Slave Select (SS). The SS pin must be set to StdCPU, Strong Drive mode.

Project Pinout

The pinout of the example project is shown in Figure 4.

Figure 4. I2C_to_SPI_RTC Pinout



Baud Rate

PSoC's I^{2}C has three possible clock rates available.

- 50 kbps Standard
- 100 kbps Standard
- 400 kbps Fast

To set the I^{2}C baud rate, set the I2C_Clock parameter (in the Device Editor).

The SPI baud rate is set by setting the clock source of the module. In the example project, the SPI master clock source is set to VC1. This is set to SysClk/16 (1.5 MHz) so that the bit rate is 750 kHz.

$$VC1 \equiv \frac{SysClk}{16} \equiv 1500000$$

$$BitRate \equiv \frac{VC1}{2} \equiv 750000$$

SPI can support up to 4 Mbps. However, this data rate cannot be maintained because the M8C processor must transfer the data from RAM to the SPI hardware. Also, the I^{2}C cannot send data that fast to the PSoC.

Interrupts

The SPI block has a selection of two interrupt sources: interrupt on TX Reg Empty (default) or interrupt on SPI Complete. Mode bit 1 in the Function register controls this selection. The interrupt type is set by choosing the desired value in the user module parameter window. If SPI Complete is selected as the block interrupt, the control register must still be read in the interrupt routine to clear this status bit; otherwise, no subsequent interrupts are generated.

I^{2}C slave interrupt is enabled or disabled by setting the Bit4 of I2C_CFG Register. I2CHW_EnableInt() is the API used to set this bit.

SPI Mode Selection

SPI mode can be set to any of the following modes using the API.

Table 1. SPI Modes

Mode	SCLK Edge Performing Data Latch	Clock Polarity	Notes
0	Leading	Non Inverting	Leading edge latches data. Data changes on trailing edge of clock.
1	Leading	Inverted	
2	Trailing	Non Inverting	Trailing edge latches data. Data changes on leading edge.
3	Trailing	Inverted	

In the example, the SPI is set to Mode2, MSB first by using the API:

```
SPIM_Start(SPIM_SPIM_MODE_2 |
SPIM_SPIM_MSB_FIRST);
```

Example I2C_Master

I^{2}C Master

A PSoC project is provided, which can be connected to the I2C_to_SPI_Bridge_RTC project. To demonstrate the functionality of the bridge, Switch(SW1) is connected to P0.1. Another Switch(SW2) is connected to P0.3. If SW1 is pressed, the current time (seconds, minutes, and hours) is sent to the RAM registers of SPI slave (DS1305 Registers 0xA0, 0xA1, and 0xA2). If SW2 is pressed, the data is retrieved from RAM registers of DS1305 and displayed on the LCD (first row).

A Timer8 User Module generates an interrupt every half second, which sets a flag to read data. When this flag is set, the I^{2}C master reads data (time and date) from SPI slave (DS1305) and displays the time on LCD (second row).

Write Data

To write data to SPI slave, the I²C master sends data in the following format to the bridge.

Process ID (0x01)	No. of bytes (N)	Start address	Byte1	Byte2	Byte N

Read Data

The I²C master first sends data in the following format to the bridge.

Process ID (0x02)	No. of bytes (N)	Start address

The I²C master then polls the bReadFlag, which is the first byte of the I²C slave (bridge) read buffer. When this flag is 1, it indicates that the bridge has finished reading the data from the SPI slave and the master can read the data from the bridge. The first byte of the data read is again the flag byte.

I2Cs-SPIm Bridge

The I²C to SPI bridge receives and decodes the data coming over the I²C bus. Depending on the process ID (first byte received in the receiver buffer), it does the following.

Process ID(0x01)-Write Data

If the process ID is (0x01), it writes data to the SPI slave .It sends the address byte followed by the N data bytes. It discards the data received over the MISO line.

Process ID(0x01)-Read Data

If Process ID is (0x02), it reads data from the SPI slave. It sends the address byte from which data must be read (start address), followed by dummy data (0x00). The data received over MISO line is stored in a buffer which is later read by I²C master. This is continued 'N' number of times.

Process ID(0x03)-Slave Select

If Process ID is (0x03), it indicates the slave it wants to communicate with. The slave select pins are configured according to the applications requirement.

When the bridge processes the commands, 'bReadFlag' is set to 0. The bridge sets the flag to 1 after it completes processing the command. This flag is monitored to know the status of the bridge.

SPIs

The slave is a Dallas Real Time clock: DS1305. The DS1305 serial alarm real time clock provides a full binary coded decimal (BCD) clock calendar that is accessed by a simple serial interface (SPI or three-wire). The clock and calendar provides seconds, minutes, hours, day, date, month, and year information.

Interfacing to the RTC

The register map of DS1305 is shown in Figure 5.

Figure 5. DS1305 Register Map

HEX ADDRESS		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	RANGE
READ	WRITE									
00H	80H	0	10 Seconds			Seconds				00–59
01H	81H	0	10 Minutes			Minutes				00–59
02H	82H	0	12	P	10 Hour	Hours				01–12 + P/A
				A						00–23
			24	10						
03H	83H	0	0	0	0	Day				1–7
04H	84H	0	0	10 Date		Date				1–31
05H	85H	0	0	10 Month		Month				01–12
06H	86H	10 Year				Year				00–99
–	–	Alarm 0								–
07H	87H	M	10 Seconds Alarm			Seconds Alarm				00–59
08H	88H	M	10 Minutes Alarm			Minutes Alarm				00–59
09H	89H	M	12	P	10 Hour	Hour Alarm				01–12 + P/A
				A						00–23
			24	10						
0AH	8AH	M	0	0	0	Day Alarm				01–07
–	–	Alarm 1								–
0BH	8BH	M	10 Seconds Alarm			Seconds Alarm				00–59
0CH	8CH	M	10 Minutes Alarm			Minutes Alarm				00–59
0DH	8DH	M	12	P	10 Hour	Hour Alarm				01–12 + P/A
				A						00–23
			24	10						
0EH	8EH	M	0	0	0	Day Alarm				01–07
0FH	8FH	Control Register								–
10H	90H	Status Register								–
11H	91H	Trickle Charger Register								–
12–1FH	92–9FH	Reserved								–
20–7FH	A0–FFH	96 Bytes User RAM								00–FF

The control register (address 0x8F) must be written with a 0 before writing to any other registers of DS1305. This is to set WP=0 (disable write protect) and start oscillator.

Process ID (0x03)	Slave to be selected

The next step is to initialize the address locations 80H to 86H (seconds, minutes, hours, day, date, month, and year registers).

Read the locations 00H to 06H continuously to update the time and date.

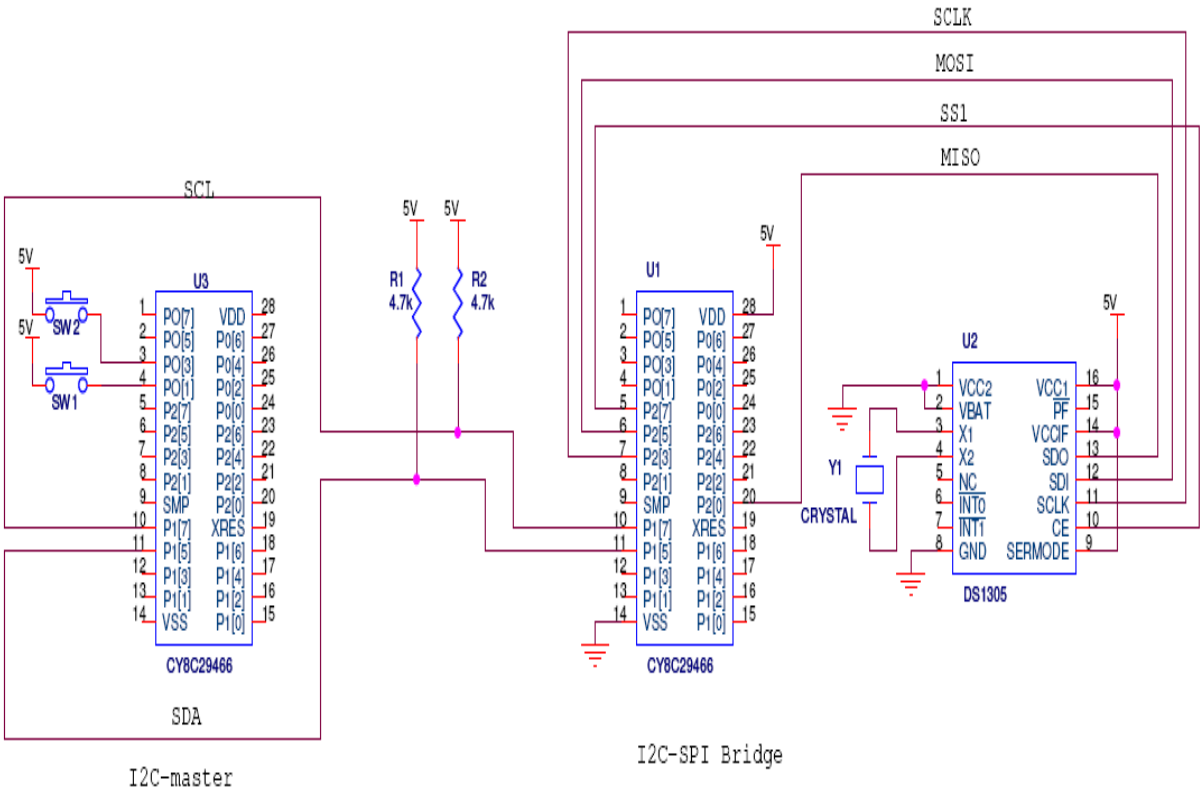
Locations 20H to 7FH are general purpose RAM locations. To write to these locations, follow the same write command format.

Summary

This application note demonstrates how to configure PSoC as an I^2C to SPI bridge. The project accompanying this application note can be implemented with other interface such as DMX512, IrDA, and USB(24x94). The protocol in this project can also be varied according to user requirements. Other functionality is allowed along with the bridge function. Unused digital and analog blocks can be configured as PWMs, counters, timers, ADCs, DACs, filters, and programmable gain amplifiers.

Appendix A

Figure 6. I^2C to SPI Bridge Test Setup



About the Author

Name: Anu M D

Title: Applications Engineer

Background: Anu works as an Applications Engineer in Cypress Semiconductor's Consumer and Computation Division, which focuses on PSoC Core applications.

Contact: anmd@cypress.com

Document History

Document Title: I^2C to SPI Bridge

Document Number: 001-49217

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2577353	ANMD	10/03/08	New application note

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com/