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.

THIS SPEC IS OBSOLETE

Spec No: 001-67442

Spec Title: SPI IMPLEMENTATION USING SERIAL MODE-0 OF EZ-USB FX2LP(TM) - AN67442

Sunset Owner: Rama Sai Krishna (Sai Krishna) Vakkantula (rskv)

Replaced by: 001-14558

SPI Implementation Using Serial Mode-0 Of EZ-USB FX2LP™

Author: Shruti Maheshwari

Associated Project: Yes

Associated Part Family: CY7C68013A/CY7C68014A

Software Version: CY3684

Related Application Notes: None

To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/go/AN67442.

This application note describes the implementation of serial peripheral interface (SPI) protocol using the FX2LP UART port in serial mode 0. This demonstration uses FX2LP as the SPI master for transferring data to and from an AT25080A EEPROM device. The example code includes functions to the Write/Read byte to and from AT25080A EEPROM.

Introduction

Although EZ-USB FX2LP does not have a built-in SPI master controller, you can use the serial port 0/1 of EZ-USB FX2LP to achieve the SPI master functionality. In synchronous mode (Mode 0), EZ-USB FX2LP generates the serial clock. The serial port operates in half-duplex mode.

SPI over Serial Port 0

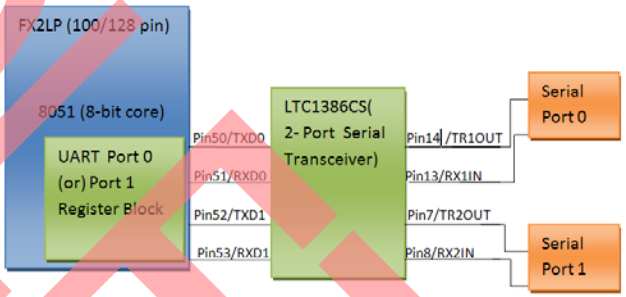
To implement SPI protocol over the serial port of EZ-USB, the following points need to be understood:

- Serial Port 0 in Mode 0 of EZ-USB
- SPI Bus Interface
- Implementing SPI protocol using FX2LP

Serial Port 0 in Mode 0 of EZ-USB

FX2LP chip has two serial ports: Serial port 0(SIO-0) and Serial port 1(SIO-1). Figure 1 shows the pin connections between the FX2LP chip and serial ports (0, 1) on the CY3684(FX2LP) development kit board provided by Cypress.

Figure 1. FX2LP-to-Serial Ports Connection on CY3684 DVK Board



The LTC1386CS is a two-port serial transceiver to transmit/receive data between FX2LP and the serial ports. For detailed pin connections, see *CY3684_A_SCH.PDF*, which is in the following location after you install the CY3684 EZ-USB FX2LP Development Kit software:

C:\Cypress\USB\Hardware\FX2LP\DEVBD REVA

Any of these serial ports configured in Serial mode 0 provides synchronous, half-duplex communication. In Mode-0, serial data output appears on the RXD0OUT pin; serial data is received on the RXD0 pin; and the TXD0 pin provides the shift clock for both transmit and receive.

The serial mode 0 baud rate is either CLKOUT/12 or CLKOUT/4, depending on the state of the SM2_0 bit. When SM2_0 = 0, the baud rate is CLKOUT /12; when SM2_0 = 1, the baud rate is CLKOUT /4.

Data transmission begins when an instruction writes to the SBUF0 SFR (Special Function Registers in 8051). The serial port shifts the data, LSB first, at the selected baud rate until the 8-bit value has been shifted out. TI_0 is set after the last bit is shifted out.

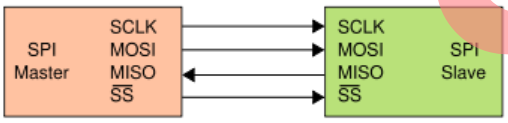
Data reception begins when the REN_0 bit is set and the RI_0 bit is cleared in the SCON_0 SFR. The shift clock is activated and the UART shifts data in on each rising edge of the shift clock until 8 bits have been received. After the last bit is shifted in, the RI_0 bit is set. Reception stops until the software clears the RI_0 bit.

SPI Bus Interface

SPI is a synchronous protocol that allows a master device to initiate communication with a slave device. Data is exchanged between these devices. The SPI bus specifies four logic signals:

- SCLK: Serial Clock (output from master)
- MOSI: Master Output, Slave Input (output from master)
- MISO: Master Input, Slave Output (output from slave)
- SS: Slave Select (active low, output from master)

Figure 2. SPI Protocol



Implementing SPI Protocol using FX2LP

The implementation of SPI protocol in half duplex communication mode involves three major stages:

- Pin I/O mapping and Hardware setup
- Firmware Details
- Testing the Hardware

Pin I/O Map and Hardware Setup

To implement SPI protocol using Serial port 0: Two pins, TXD0 and RXD0, come out of the FX2LP chip. Two more pins are required to implement the SPI protocol. The following table summarizes the pin mapping.

Table 1. SPI Signal -to-Serial Port-0 Pin Map

No.	SPI Signal	Serial Port0 Signal
1	SCLK	TXD0
2	MOSI	RXD0OUT-PORTE (Pin 3)
3	MISO	RXD0
4	SS	PORTC (Pin 2)

To test the SPI protocol, the following test hardware was selected:

- CY3684 FX2LP Development Kit Board.
- SPI-based AT25080A EEPROM.

Figure 3. FX2LP-to-SPI EEPROM Connection

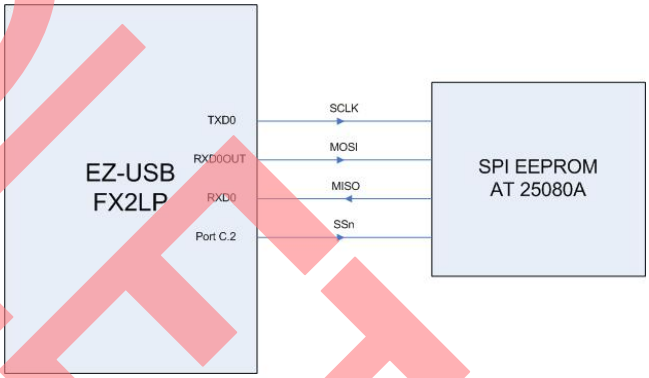


Figure 3 shows that SPI EEPROM can be connected to FX2LP. Because FX2LP works at 3.3 V, the supply voltage applied to EEPROM is 3.3 V. To control Slave Select (SS) from firmware, a GPIO pin 2 of Port C is used.

Firmware Details

The *vend_ax_spi* firmware example is based on the EZ-USB firmware framework files (*fw.c*, *EZUSB.LIB*, *fx2regs.h*, and so on). After installing the CY3684 EZ-USB FX2LP Development Kit, these can be found in `C:\Cypress\USB\` (the location varies based on installation). Refer to *DvkUsersGuide.pdf,* found in `C:\Cypress\USB\doc\General` (the location varies based on installation), for details on firmware frameworks.

The following sections describe modifications to the firmware frameworks for creating the firmware suitable for SPI communication.

Initialization

The SCON_0 register of serial port-0 must address the following items:

- **Serial Mode 0** — Set SM0_0 = SM1_0 = 0.
- **Baud Rate** — By default, it is set to 2 MHz. This frequency decides the SCK frequency.

The following table shows possible frequencies.

			BAUD:SCK Frequency	
CLK SPD1	**CLKS PD0**	**CLKOUT**	**SM2_0=1 (CLKOUT/4)**	**SM2_0=1 (CLKOUT/12)**
0	0	12 MHz	3 MHz	1 MHz
0	1	24 MHz	6 MHz	2 MHz
1	0	48 MHz	12 MHz	4 MHz

AT25080A allows a maximum of 5 MHz of SCK at Vcc = 3.3 V.

Therefore, the system can be tested for 1, 2, 3, 4 MHz.

- **Receive Enable** — Set REN_0 = 1 to receive data.
- **Interrupt Flags** — Set TI_0 = RI_0 = 1 so that these flags are zero only when transmitting or receiving. A quick check of their state can determine if the UART is busy. Also, if using interrupts, you can software-trigger the first interrupt and transmit or receive your first byte.

SM0_0	SM1_0	SM2_0	REN_0	N/A	N/A	TI_0	RI_0
b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	1	0	0	1	1

The final initialization is to turn on the alternate pin functions associated with the serial port. This example uses Serial Port 0. Set the following registers:

- **PORTECFG** = 0x08 – Enables RXD0OUT function on Port E.3 which is used as MOSI.
- **OEC** = 0x04 – Enables Port C.2 which is used as Slave Select and toggled by firmware.

TD_Init() function in frameworks takes care of EZ-USB FX2LP initialization. This can be found in VEND_AX.C.

```
void TD_Init(void)
   // Called once at startup
{
//Set CPUCS 0x00:-12Mhz;0x08:-24MHz;0x18:-
48Mhz;
      CPUCS = (CPUCS & ~(bmCLKSPD));
//Clear the set clk frequency
     CPUCS = (CPUCS | 0x08);
//Set CLKOUT = 24Mhz default

     SCON0 = 0x13;
//Mode 0, baud 24/12, enable receive
      PORTECFG = 0x08;
//Turn on uart pin rxd0out
      OEC = 0x04;
//Make CS# output

      CKCON &= 0xF8;
//Set stretch 0

}
```

SPI Low-Level Driver

Data is transmitted one byte at a time by calling the **spiwritebyte** function. Transmitting a byte consists of:

- Clearing the TI_0 flag.
- Writing the byte to the SBUF_0 register.
- Waiting until the TI_0 flag is set indicating that the last bit has been shifted out.

This function also uses a lookup table, **swap**, which reverses the bit order. This is done because the UART transmits LSB first, but the AT25080A is expecting MSB first.

Data is received one byte at a time by calling the **spireadbyte** function. Receiving a byte consists of:

- Clearing the RI_0 flag.
- Waiting until the RI_0 flag is set indicating that the last bit has been shifted in.
- Reading the SBUF_0 register.

This function also uses the swap lookup table to reverse the bit order. The driver functions are available in the spi_eeprom.c file (see the attached example project). The functions are as follows:

```
void spiwritebyte (BYTE d)
 {
 TI = FALSE; //Clear flag
 SBUF0 = swap1[d]; //Write byte
 while (!TI); //Wait until done
transmitting
 }

 BYTE spireadbyte (void)
 {
 RI = FALSE; //Clear flag
 while (!RI); //Wait until done receiving
 return (swap1[SBUF0]); //Return byte
 }
```

The vend_ax_spi firmware project was compiled using Keil uvision2 IDE environment.

Testing the Hardware

There are two methods for downloading the firmware before testing the SPI communication.

- Firmware download to FX2LP Internal RAM: This is a direct download to FX2LP 16 KB internal RAM memory.
- Firmware download to FX2LP DVK board EEPROM:Firmware after download will boot from the I2C EEPROM.

Both procedures on how to download are explained in detail in the application note titled *Vendor Command Design Guide for the FX2LP* - *AN45471*. In this application note, under section “HEX File Download,” the RAM download procedure is explained. For the EEPROM download, refer to the section “IIC File Programming” in the same application note.

After the download is complete, the hardware can be tested using vendor commands. These commands are used to perform various functions such as reading data, writing data, reading status register, and setting frequency of operation on the SPI EEPROM. Refer to the section “Vendor Command Testing” in the same application note. The test procedure is similar, but you must change the following parameters before triggering the vendor command from the *Cyconsole* application.

Table 2 shows the list of vendor commands used for SPI EEPROM.

Table 2. Vendor Commands for SPI EEPROM

Command	Purpose	Attributes
0xB2	EEPROM READ/WRITE	Setup Data 0 = 0xC0: Read 0x40:Write Setup Data 2: Address (LSB) Setup Data 3:Address (MSB) Setup Data 6: Length (LSB) Setup Data 7: Length(MSB)
0xB3	Read Status Register	Setup Data 0 = 0xC0: Read
0xB4	Set Frequency	Setup Data 2: 0x1: 1Mhz 0x2: 2Mhz 0x3: 3Mhz 0x4: 4Mhz 0x6: 6Mhz 0xC: 12Mhz

Note AT25080A EEPROM can be tested only for 1, 2, 3, or 4 MHz because the maximum frequency of operation is 5 MHz only. The system has been successfully tested up to 4 MHz.

The code for implementation can be found in function DR_VendorCmnd() in VEND_AX_spi.c (see the attached example project).

Summary

This application note shows an easy way to implement SPI communications using the EZ-USB serial ports in mode 0. A minimum of code is required. Very respectable performance can be achieved as demonstrated with AT25080A EEPROM. Application-specific modifications and optimizations such as Page Program and Sequential Read can be made to communicate with other SPI peripherals and to increase performance.

About the Author

Name: Shruti Maheshwari.

Document History

Document Title: SPI Implementation Using Serial Mode-0 of EZ-USB FX2LP™ - AN67442

Document Number: 001-67442

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3172056	NMMA	02/17/2011	New application note.
*A	4292377	GAYA	02/26/2014	Updated in new template. Completing Sunset Review.
*B	4372774	RSKV	05/07/2014	Obsolete document.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com