

### Introduction

This design provides protocol convergence between widely used shared bus architectures: the serial peripheral interface (SPI) and the I<sup>2</sup>C bus. The Altera® Max® II CPLD serves as a bridge that allows hosts having an SPI interface to communicate with devices connected through an I<sup>2</sup>C bus.

### I<sup>2</sup>C and SPI

The I<sup>2</sup>C is a serial, two-wire, low-bandwidth, industry standard protocol used in embedded systems to communicate with various low-speed peripheral devices. The SPI, on the other hand, is a widely used, fast, four-wire, full duplex, serial communication interface. Many embedded systems today have SPI interfaces, making it difficult to connect them with peripheral devices in an I<sup>2</sup>C fashion. One way to achieve this connection is to modify the system, but this is economically inefficient. The best way to do this is to use a CPLD that serves as a bridge between the two interfaces.

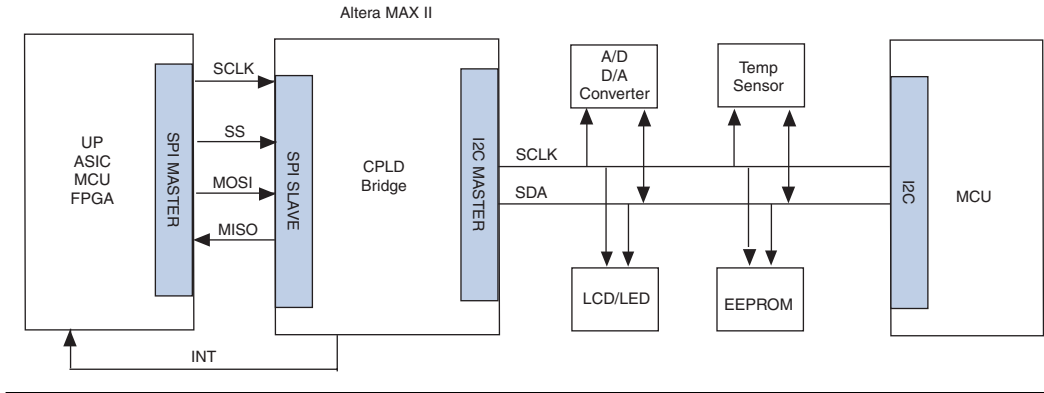
This design implements such a bridge using a MAX II CPLD as it provides greater flexibility, consumes less power, and can be economically integrated into the embedded system. The CPLD appears as an SPI slave to the host (SPI master) and acts as a master to the I<sup>2</sup>C bus.

### SPI to I<sup>2</sup>C Using MAX II CPLDs

This design enables an SPI-interface-equipped host to control data flow to other devices such as an A/D converter, LED controller, audio processor to read temperature sensors, hardware monitors, and diagnostic sensors, that are on an I<sup>2</sup>C interface.

Figure 1 shows implementing an SPI to an I<sup>2</sup>C interface using a MAX II CPLD. The bridge interfaces with the SPI host as an SPI slave using four wires, SS and SCLK signals for control, and MISO and MOSI signals for data. The side interfacing with the I<sup>2</sup>C bus has two wires and SCLK and SDA signals.

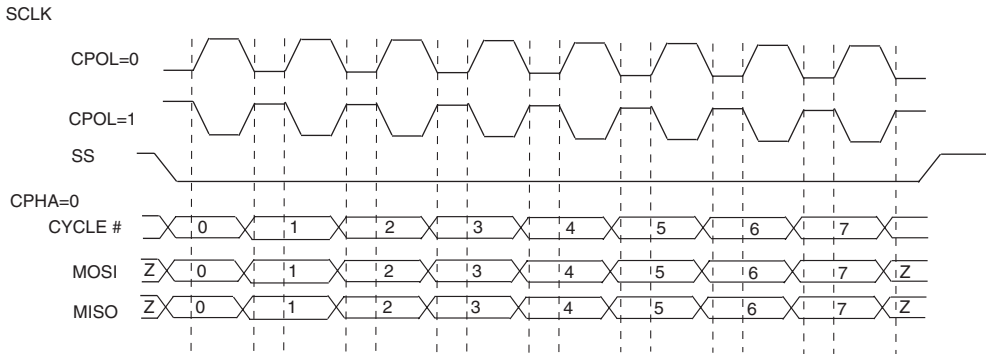
Figure 1. Implementing an SPI to I<sup>2</sup>C Interface Using a MAX II CPLD



### SPI Interface

The SPI bus normally has only one master and many slaves connected to it. The CPLD bridge acts as one of the slaves to the SPI master device. [Figure 2](#) shows the SPI timing diagram. [Table 1](#) shows the SPI interface pin descriptions.

Figure 2. SPI Timing Diagram



<b>Signal</b>	<b>Purpose</b>	<b>Direction</b>
SS	Slave select	Input (active low)
SCLK	SPI clock	Input
MISO	Master-in Slave-out	Output
MOSI	Master-out Slave-in	Output

The SPI host sends:

- command register (8 bit)
- data in (8 bit)

The SPI host receives:

- status register (8 bit)
- data out (8 bit)

The SPI word length is fixed at 16 bits. CPOL = 0 and CPHA = 1, as indicated in [Figure 2](#). In every SPI word, the command register dictates the functions on the I<sup>2</sup>C bus and the data in holds the data to be sent by the I<sup>2</sup>C bus. Similarly, the last bit of the status register is the acknowledge bit and the data out is the data received over the I<sup>2</sup>C line in the previous I<sup>2</sup>C cycle.

At the end of every SPI bus, the slave select line goes high, indicating a word complete and, per the value of command register at that time, an I<sup>2</sup>C bus cycle is executed. After a fixed delay, depending on the frequency of the I<sup>2</sup>C SCL, another SPI word can be sent. The minimum delay between two SPI words is the I<sup>2</sup>C SCL clock frequency.

## I<sup>2</sup>C Interface

The CPLD bridge acts as a master to the I<sup>2</sup>C bus. Since the purpose of this design is to provide an interface between an SPI master and an I<sup>2</sup>C slave device, multi-master support is not provided on the I<sup>2</sup>C bus. [Table 2](#) shows the I<sup>2</sup>C interface pin descriptions.

Signal	Purpose	Direction
SCLK	I <sup>2</sup> C serial clock	Output
SDA	I <sup>2</sup> C data bus	Bidirectional

The I<sup>2</sup>C functions are carried out based on the command register value received from the SPI side. The significance of the value stored in the command register is shown in [Table 3](#).

Command Register	Purpose on the I <sup>2</sup> C Line	Data In Register
10000000	Start/repeat start	Slave address + R/W
01000000	Write a byte	Data to be written
00100000	Read a byte	Don't care
00010000	Stop	Don't care
00000000	Null, wait state	Don't care

The data read in a particular I<sup>2</sup>C transaction is stored in the data out register and is read by the SPI master in its next SPI transaction. The last command word, 00000000 (b), is required for the SPI master to read the value of status and data out registers without doing anything on the I<sup>2</sup>C bus. The I<sup>2</sup>C command format is shown in [Figure 3](#).

**Figure 3. I<sup>2</sup>C Command Format**




## Implementation

This design can be implemented with an EPM240 or any other MAX II CPLD. Implementation involves using this design source code and allocating the appropriate signal and control lines to the general purpose I/O (GPIO) lines of the MAX II CPLD. An SPI master and an I<sup>2</sup>C slave are additional resources required to demonstrate this implementation.

The following steps detail the demonstration using the MDN-B2 board, a PC parallel-port-based SPI environment created using Freescale's SPIGen software (along with suitable parallel port interfacing hardware), and an I<sup>2</sup>C slave battery gauge module (supplied with the MDN-B2).

Table 4 shows the EPM240G pin assignments.

<i>Table 4. EPM240G Pin Assignments</i>	
Pin Assignments	
Signal	Pin
I2C_scl:	Pin 39
I2C_sda	Pin 40
SPI_cs	Pin 95
SPI_miso	Pin 91
SPI_mosi	Pin 92
SPI_clk	Pin 96

 Unused pins are assigned as **input tri-stated** in the Quartus<sup>®</sup> II software's device and pin option settings. SPI\_clk is assigned a 2.5-V Schmitt Trigger input in the I/O standards column in the Quartus II software during pin assignment. The Assignment Editor in the Quartus II software is used to enable **Auto Open Drain** on the I2C\_scl and I2C\_sda pins. Compile the design after turning on the appropriate settings.

## Design Notes

To demonstrate this design on the MDN-B2 demo board, complete the following:

1. To set up the SPI environment using a PC and its parallel port, download SPI emulation software such as the SPIGen from Freescale. This software is available for free download after accepting Freescale's terms and conditions of use and free registration.



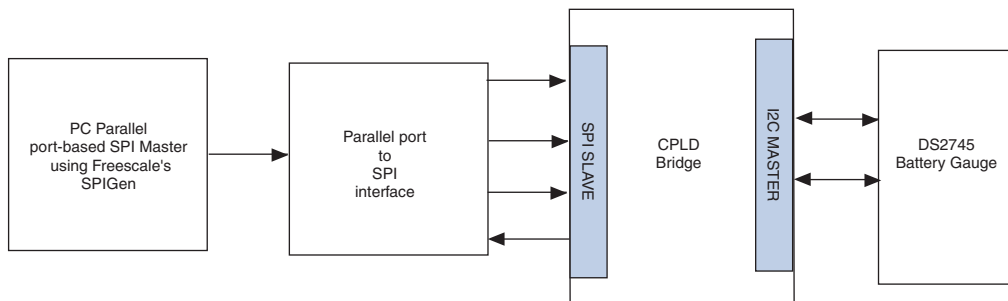
Visit the following website to download the SPI emulation software:

[http://www.freescale.com/files/soft\\_dev\\_tools/software/device\\_drivers/SPIGen.html](http://www.freescale.com/files/soft_dev_tools/software/device_drivers/SPIGen.html)


2. Once you register, download and install SPIGen. You will need to configure it to suit your application and work with the Parallelport-to-SPI dongle that is provided with the MDN-B2 demo board.
3. Open the SPIGen software and click the **Configure** menu. Select **Edit Configuration**. In the **General** tab, verify the default **Port Address** is the correct port address of the parallel port in your PC. If it is not, select the correct option. You can find your PC's parallel port address by looking at the settings in Control Panel > System > Hardware > Device Manager > Ports > ECP Printer Port (LPT1) > Resources.

Figure 4 shows the demo arrangement.

**Figure 4. Demo Arrangement**



4. Select a 16-bit format and choose to display the SPI values in hexadecimal (Intel-Format) file (.hex) format.
5. Configure the **SPI Pins** tab by completing the following:
  - a. Leave the pin assignments in their default settings (Chip Select: 2, Data In: 3, Data Out: 12, and Clock: 4).
  - b. Leave **Bit Order** in its default **MSB is sent first** setting.
  - c. Change **Chip Select** to **High when asserted**.

- d. Change **Data In** and **Data out** to **Low = 1**.
  - e. Change the **SPI Type** to **Type 4**.
  - f. Click **Apply** and **OK** to return to the main SPIGen screen.
6. Connect the Parallelport-to-SPI dongle to the parallel port of your PC. Use a parallel port extension cable to make it more convenient to connect the dongle to the demo board. The parallel-port-based SPI environment is now ready to use.
  7. Turn on the power to the demo board (using slide switch SW1). Download the design to the MAX II CPLD through the JTAG header JP5 on the demo board and a conventional programming cable (ByteBlaster™ II or USB-Blaster™).
  8. Keep SW4 on the demo board pressed before and during the start of the programming process. Once complete, turn off the power and remove the JTAG connector.
  9. Mount the DS2745 I<sup>2</sup>C battery gauge module (provided with the MDN-B2 demo board) on JP3 of the demo board. Ensure the red mark on the module matches pin#1 on JP3.
  10. Connect the Parallelport-to-SPI dongle to the demo board.
  11. Connect the 6-pin connector on the dongle's pigtail wire to the JP8 on the demo board. Ensure the red wire on this 6-pin connector matches pin#2 on JP8.
-  Note that JP8 is a 10 x 2 pin header; this 6-pin connector will only occupy a portion of JP8.
12. Turn on the demo board using slide switch SW1.
  13. [Table 5](#) indicates the sequence of the **.hex** data to send using the **Word to Send (DI)** field in the SPIGen software.

**Table 5. SPI "Data to Send" in Hex Using SPIGen (Part 1 of 2)**

Signal Direction	Command/Data
Send (DI)	80 90 (start + slave address)
Send (DI)	40 0C (write + memory addr)
Receive (DO)	01 00 (ack)
Send (DI)	80 91 (start + slave address)

**Table 5. SPI “Data to Send” in Hex Using SPIGen (Part 2 of 2)**

Signal Direction	Command/Data
Receive (DO)	01 00 (ack)
Send (DI)	20 00 (read)
Receive (DO)	01 ?? (ack + MSB data)
Send (DI)	00 00 (do nothing)
Receive (DO)	01 ?? (ack + LSB data)

14. Use the **.hex** data from [Table 5](#) and click **Send Once** each time you want to send the SPI data. Verify the acknowledgment data is received.
15. Observe the battery gauge data received. It is received in two parts, giving the MSB 8-bits data first and the LSB 3-bit data next (last row).
16. Vary the yellow preset on the battery gauge module and observe the varying data received. The yellow preset varies the input voltage to the battery gauge chip (Maxim DS2745) which provides the voltage readings through its I<sup>2</sup>C interface and in 11-bit 2’s complement form.



For more information, refer to:

[http://www.maxim-ic.com/quick\\_view2.cfm/qv\\_pk/4994](http://www.maxim-ic.com/quick_view2.cfm/qv_pk/4994)

## Source Code

This design has been implemented in Verilog and successful operation has been demonstrated using the MDN-B2 demo board, as referenced in the documentation. The source code, testbench, and complete Quartus II project are available at:

[http://www.altera.com/literature/an/an486\\_design\\_example.zip](http://www.altera.com/literature/an/an486_design_example.zip)

## Conclusion

As illustrated by this design, MAX II CPLDs are a great choice to implement industry standard interfaces such as the SPI and I<sup>2</sup>C. Their low power, easy power sequencing, and internal oscillator make them ideal programmable logic devices to implement interface converter applications such as the SPI to I<sup>2</sup>C.

## Referenced Documents

This application note references the following documents:

- [http://www.altera.com/literature/an/an486\\_design\\_example.zip](http://www.altera.com/literature/an/an486_design_example.zip)
- [http://www.freescale.com/files/soft\\_dev\\_tools/software/device\\_drivers/SPIGen.html](http://www.freescale.com/files/soft_dev_tools/software/device_drivers/SPIGen.html)
- [http://www.maxim-ic.com/quick\\_view2.cfm/qv\\_pk/4994](http://www.maxim-ic.com/quick_view2.cfm/qv_pk/4994)

## Additional Resources

The following are additional resources for this application note:

- MAX II CPLD home page:  
<http://www.altera.com/products/devices/cpld/max2/mx2-index.jsp>
- MAX II Device Literature page:  
<http://www.altera.com/literature/lit-max2.jsp>
- MAX II Power-Down Designs:  
<http://www.altera.com/support/examples/max/exm-power-down.html>
- MAX II Application Notes:  
*AN 428: MAX II CPLD Design Guidelines*  
*AN 422: Power Management in Portable Systems Using MAX II CPLDs*

## Document Revision History

Table 6 shows the revision history for this application note.

<i>Table 6. Document Revision History</i>		
<b>Date and Document Version</b>	<b>Changes Made</b>	<b>Summary of Changes</b>
December 2007, v1.0	Initial release.	—



101 Innovation Drive  
San Jose, CA 95134  
[www.altera.com](http://www.altera.com)  
Technical Support:  
[www.altera.com/support](http://www.altera.com/support)  
Literature Services:  
[literature@altera.com](mailto:literature@altera.com)

Copyright © 2007 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

