



GPIO Pin Expansion Using I²C Bus Interface in MAX II CPLDs

December 2007, ver. 1.0

Application Note 494

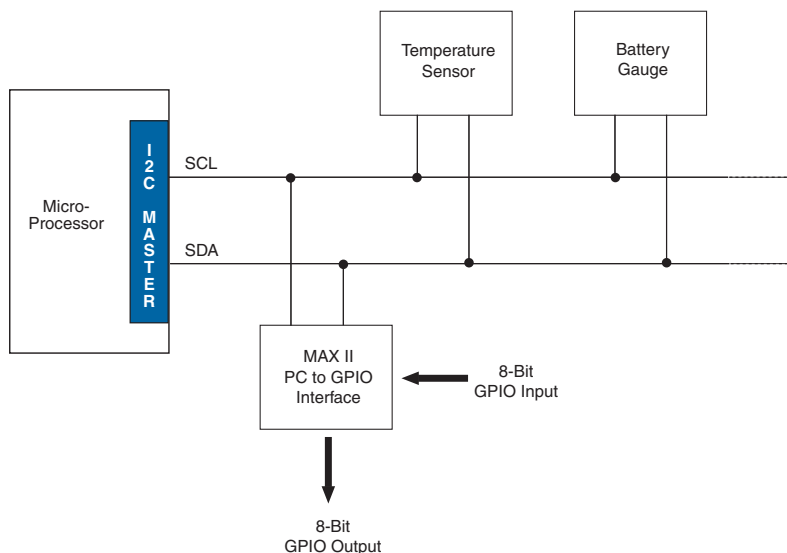
Introduction

This design example illustrates the capability of Altera® MAX® II CPLDs to provide general purpose I/O (GPIO) pin expansion via an industry standard I²C bus. To reduce package size and pin count, the number of general purpose I/Os are limited in many microprocessor-based systems. However, if the system has an I²C interface, this design example shows how to add additional GPIO pins via the I²C bus. With the help of MAX II CPLDs, the additional GPIO pins consume less power compared to I/O pins on a microprocessor.

GPIO Pin Expansion and I²C

In some cases, it may be required to have access to the GPIO pins from a relatively long PCB trace path within the system (such as in the two different parts of a clamshell cell phone). Because the I²C interface is a two-wire system, the design provides multiple input and output pins at the remote end with just a common two-wire trace. This provides increased design flexibility and also adds to the physical compactness of the entire system. It also enables smaller packaging and a reduced pin count.

Devices such as fan controllers, LED status displays, and status indicators can be easily connected and controlled via the general purpose output pins. Similarly, devices such as reset pins and push button switches can be easily coupled to the general purpose inputs provided on the CPLD to serve various applications.

Figure 1. GPIO Pin Expansion via an I²C Bus

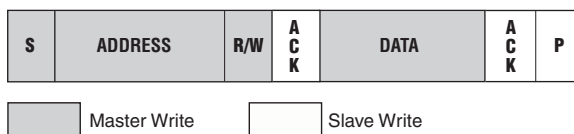
I²C Interface for GPIO Pin Expansion

The MAX II CPLD acts as a slave on the I²C bus and has two pins on its I²C interface: the I²C clock SCL and the I²C data line SDA. The host system, which acts as an I²C master, communicates with the MAX II device (which acts as an I²C slave). The CPLD presents eight general purpose input ports and eight general purpose output ports to the host. Data, which is transmitted serially over the I²C bus, is received in parallel at the GPIO pins. This way, all eight general purpose I/Os can be read or written at the same time.

I²C Interface

For the I²C interface, the CPLD (I²C slave) has a built-in 7-bit address and follows the general I²C protocol. The start signal is sent by the master, followed by the 7-bit address and an R/W bit. When the address broadcast on the I²C bus matches a slave device's address, an ACK (acknowledge) signal is sent by the device followed by data according to the read or write signal sent by the master. This is then followed by another ACK signal. The exchange of data continues in this manner until the Stop (P) signal is sent by the master.

Signal	Purpose	Direction
SCL	I ² C Clock	Output
SDA	I ² C Serial Data	Bidirectional

Figure 2. I²C Signal Format

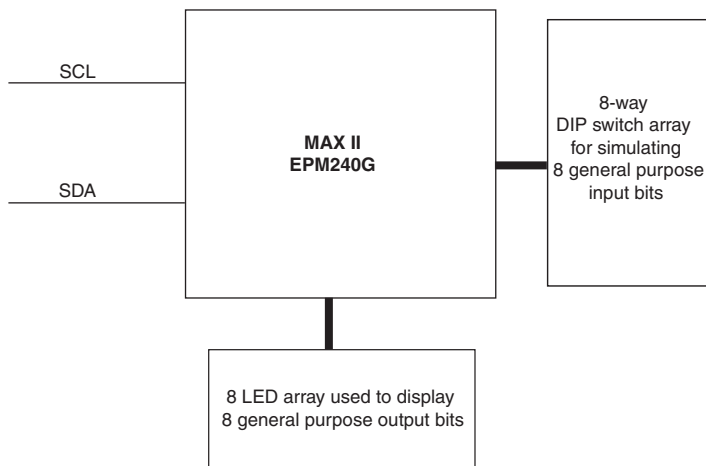
S = Start (SCLK high, SDA high to low)
 R/W = Read/Write (1 for Read, 0 for Write)
 ACK = Acknowledgement (SDA held low by receiver)
 P = Stop (SCLK high, SDA low to high)
 Default Slave Address = 0000000 (00h)

GPIO Interface

Whenever the master issues a write condition (R/W=0), the data received on the I²C bus is used to update the general purpose output pins until a stop or a repeat start condition is encountered. Similarly, when the I²C master issues a read condition (R/W = 1), the values at the general purpose input pins are sampled at the ACK bit and transmitted serially over the I²C bus. This process continues until the master issues a stop or repeat start.

Signal	Purpose	Direction
8-bit Input	General	In
8-bit Output	General	Out

Figure 3. GPIO Pin Expansion Demonstration Circuit



Implementation

This design example can be implemented with an EPM240G device or any other MAX II CPLD. Implementation involves using this design example source code and allocating I²C bus lines and GPIO pin expansion inputs and outputs to MAX II GPIOs. GPIO pin expansion is demonstrated on the MDN-B2 demo board with the help of an I²C simulator that is created using a PC parallel port and interfacing hardware to create an I²C compliant two-wire bus. Details about setting up an I²C environment is described in the Dallas Semiconductor's Maxim application note AN3230, which can be found at www.maxim-ic.com/appnotes.cfm/an_pk/3230. In addition to the free software that can be downloaded from this URL, another similar software program for a parallel port I²C setup can be downloaded from:

http://files.dalsemi.com/system_extension/AppNotes/AN3315/ParDS2W.exe

This utility program uses the parallel port and its interfacing hardware to interact with the MAX II CPLD and provides the SDA and SCL connections, as required on an I²C two-wire system. When implemented, this design allows inputs from the MDN-B2 demo board (set via DIP switches) to reach the I²C master. Similarly, data sent by the I²C master is available on the GPIO output ports (connected to LEDs on the demo board) of the MAX II CPLD. The I²C master in this demonstration is the user interface on the PC running the parallel port I²C software.

The following details the implementation of this design example on the MDN-B2 demo board. Table 3 lists the EPM240G pin assignments for this design example.

EPM240G Pin Assignments			
Signal	Pin	Signal	Pin
SCLK	pin 39	SDA	pin 40
GPIO_output[0]	pin 69	GPIO_output[1]	pin 70
GPIO_output[2]	pin 71	GPIO_output[3]	pin 72
GPIO_output[4]	pin 73	GPIO_output[5]	pin 74
GPIO_output[6]	pin 75	GPIO_output[7]	pin 76
GPIO_input[0]	pin 55	GPIO_input[1]	pin 56
GPIO_input[2]	pin 57	GPIO_input[3]	pin 58
GPIO_input[4]	pin 61	GPIO_input[5]	pin 66
GPIO_input[6]	pin 67	GPIO_input[7]	pin 68

Assign unused pins **As input-tristated** in the Quartus® II software. You must also enable the **Auto Open-Drain** setting on the SCLK and SDA pins. To do this, on the Assignments menu, click **Settings** and then select **Analysis and Synthesis Settings** to enable the **Auto Open-Drain** setting. These settings are followed by a compilation cycle.

Design Notes

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

1. Turn on the power to the demo board using the slide switch SW1.
2. 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™).
3. Keep SW4 on the demo board pressed before and during the start of the programming process. After programming, turn off the power and remove the JTAG connector.
4. To set up a parallel port driven I²C environment on your PC, perform the following:

- a. Download a software utility, such as the Maxim parallel port utility, to communicate with the slave in the I²C defined protocol. Install the parallel port software. (The ParDS2W.exe program is used in this example.)
 - b. You must install a parallel port driver to enable access to the parallel port in Windows XP or Windows 2000 for this parallel port utility. You can download a driver from Direct-IO at: www.direct-io.com/Direct-IO/directio.exe
 - c. After installation, you must configure the Direct-IO program. Open the Windows control panel and click the Direct IO icon. Enter the **Begin** and **End** addresses of your parallel port (normally, this is 378 through 37F; however, confirm your PC's parallel port address by looking at the settings in Control Panel/System/Hardware/Device Manager/Ports/ECP Printer port (LPT)/Resources
 - d. Configure the parallel port to ECP by changing the BIOS settings when you start up your PC.
 - e. Next, select the **Security** tab of the Direct IO control panel and browse to the directory path of the ParDS2W.exe program. Click **Open** and then click **Add** to add the program. The path of this utility is shown in the **Allowed Processes** field. Click **OK**.
 - f. Attach the parallel port I²C dongle that is supplied along with the MDN-B2 demo board. Use an extension chord, if necessary, to extend the parallel port connection closer to your demo board.
 - g. Attach the 4-pin socket on the pig tail of the I²C parallel port dongle to the I²C header (JP3) of the demo board so that the red mark on the socket meets pin 1 on the JP3 header.
 - h. Open the ParDS2W program, select the appropriate parallel port address of your PC (as seen when configuring Direct IO) and set the **2-Wire Device Address** to 00h.
 - i. Finally, you can test the I²C setup on the **Test Circuit** tab to see if you have a Test PASS message in the Status window. If you do, the I²C environment is set.
5. With the parallel port utility you can now perform write and read operations in I²C using the 2-Wire Functions.

6. To perform a write I²C operation, click **Start** and then click **Write Byte**. Enter a hex byte in the field adjacent to **Write Data** and click **Write Data**. Observe the corresponding value on the eight red LEDs. Click **Stop** after each write operation.
7. Similarly, a read operation is performed by clicking **Start** and then **Read Byte**. The Read window displays the settings on the SW5 dip switch on the demo board. Click **Stop** after each read operation.

Source Code

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

www.altera.com/literature/an/an494_design_example.zip

Conclusion

As illustrated by this design example, MAX II CPLDs are an excellent choice to implement industry standard interfaces, such as the I²C. Their low power, easy power-on feature, and internal oscillator make them ideal programmable logic devices to implement applications such as I²C interfaces to provide GPIO pin expansion.

Additional Resources

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

Document Revision History

Table 4 shows the revision history for this application note.

Date and Document Version	Changes Made	Summary of Changes
December 2007 v1.0	Initial release.	—



101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support:
www.altera.com/support/
Literature Services:
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.

