

## Introduction

This application note illustrates the capability of Altera® MAX® II CPLDs to provide general purpose I/O pin expansion via an industry standard System Management Bus (SMBus).

## SMBus for GPIO Pin Expansion

To reduce package size and pin count, the number of General Purpose I/O (GPIO) pins in a microprocessor-based system are limited, but if the system has an SMBus interface, this design can provide it with additional GPIO pins via its SMBus. Furthermore, the additional GPIO pins provided when using MAX II CPLDs would consume less power than the I/O pins on the microprocessor.

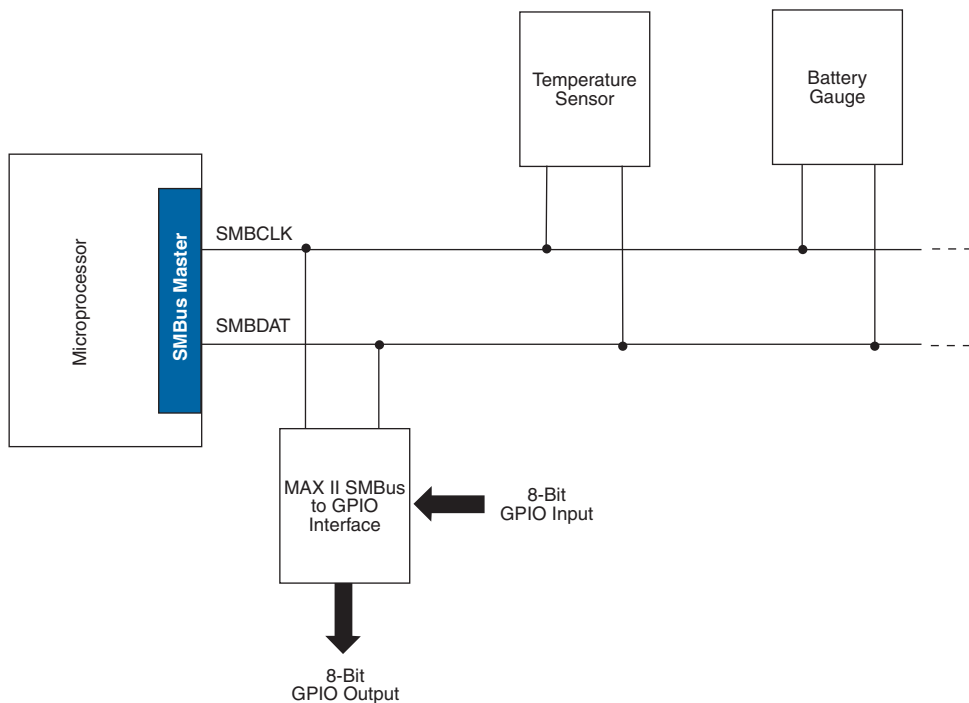
In some cases, you may be required to have access to the GPIO pins from a remote location within the embedded system. Since the SMBus interface is a two wire system, the design provides multiple input and output pins at the remote location without the need of additional wiring. This provides increased design flexibility with reduced complexity and software overheads.

You can easily connect and control peripheral controllers, LEDs, and other status indicators via these general purpose output pins. Similarly, you can couple user controls, push buttons, and switches on the front panel to the general purpose inputs provided on the CPLD.

## MAX II CPLD-Based SMBus to GPIO Pin Expander

The MAX II CPLD acts as a slave to the SMBus and has two pins on its SMBus interface: the SMBus clock *SMBCLK* and the data line *SMBDAT*. The host system which acts as an SMBus master, communicates with the MAX II Device (acting as an SMBus 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 SMBus, is received in parallel at the GPIO pins. Thus, all eight general purpose I/Os can be read/written at the same time (see [Figure 1](#)).

Figure 1. GPIO Pin Expansion Via an SMBus



## SMBus Interface

For the SMBus interface, the CPLD has an in-built 7-bit address and follows the general SMBus protocol. The start signal is sent by the master followed by the 7-bit address and a read/write bit. When the address broadcasted on the bus matches with 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 followed by another ACK signal. This way, the exchange of data continues until the Stop (P) signal is sent by the master. Table 1 shows the SMBus interface description.

<i>Table 1. SMBus Interface Description</i>		
Signal	Purpose	Direction
SMBCLK	Clock	Input (slave)
SMBDAT	Serial data	Bidirectional

Figure 2 shows the SMBus signal format.

**Figure 2. SMBus Signal Format**

S	ADDRESS	R/W	A C K	DATA	A C K	P
---	---------	-----	-------------	------	-------------	---

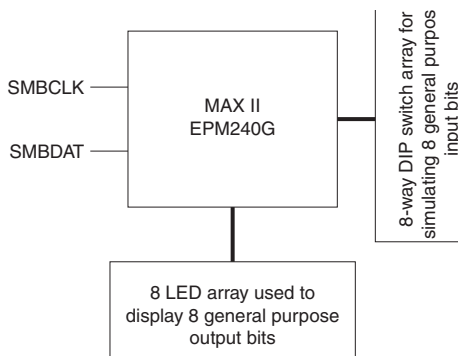
S = Start (SMBCLK high, SMBDAT high to low)  
 R/W = Read/Write (1 for Read, 0 for Write)  
 ACK = Acknowledge (SMBDAT held low by receiver)  
 P = Stop (SMBCLK high, SMBDAT low to high)  
 Default Slave Address = 0000000

## GPIO Interface

The data received on the SMBus is used to update the general purpose output pins until a stop or a repeat start condition is encountered whenever the master issues a write condition (R/W = 0). Similarly, when the bus 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 SMBus. This process continues until the Master issues a Stop or Repeat Start. Table 2 shows the GPIO pin descriptions. Figure 3 shows the GPIO pin expansion demonstration circuitry.

**Table 2. GPIO Pin Description**

Signal	Purpose	Direction
8-bit input	General	Input
8-bit output	General	Output

**Figure 3. GPIO Pin Expansion Demonstration Circuit**

## Implementation

You can implement this design example with an EPM240 or any other MAX II CPLD. Implementation involves using this design example source code and allocating SMBus bus lines and GPIO pin expansion inputs and outputs to the MAX II CPLD GPIOs. GPIO pin expansion is demonstrated on the MDN-B2 demo board and with a 2-wire simulator that is created using a PC parallel port and interfacing hardware to create an SMBus compliant 2-wire bus. Details about setting up the 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](http://www.maxim-ic.com/appnotes.cfm/an_pk/3230)

Similar software for a parallel port to 2-wire-interface setup can also be downloaded from:

[http://files.dalsemi.com/system\\_extension/AppNotes/AN3315/ParD S2W.exe](http://files.dalsemi.com/system_extension/AppNotes/AN3315/ParD S2W.exe)

This utility program uses the parallel port and its interfacing hardware to interact with the MAX II CPLD and provides the SMBDAT and SMBCLK connections as required on a SMBus 2-wire system. When implemented, this design allows inputs from the MDN-B2 demo board (set via DIP switches) to reach the SMBus Master. Likewise, data sent by the SMBus Master is available on the GPIO output ports (connected to LEDs on the demo board) of the MAX II CPLD. For this demonstration, the SMBus Master is the user interface on the PC running the parallel-port to 2-wire software.

Table 3 shows the implementation of this design example on the MDN-B2 demo board:

<b>Signal</b>	<b>Pin</b>
SMBCLK	pin 39
GPIO_output [0]	pin 69
GPIO_output [2]	pin 71
GPIO_output [4]	pin 73
GPIO_output [6]	pin 75
GPIO_input [0]	pin 55
GPIO_input [2]	pin 57
GPIO_input [4]	pin 61
GPIO_input [6]	pin 67
SMBDAT: pin 40	pin 40
GPIO_output [1]	pin 70
GPIO_output [3]	pin 72
GPIO_output [5]	pin 74
GPIO_output [7]	pin 76
GPIO_input [1]	pin 56
GPIO_input [3]	pin 58
GPIO_input [5]	pin 66
GPIO_input [7]	pin 68

Assign unused pins **As input tri-stated** in the Quartus II software. You must also enable the **Auto Open-Drain** setting on the SMBDAT pin. From the Assignments menu, select **Settings**. The **Settings** dialog box appears. In the **Category** list, select **Analysis & Synthesis Settings**. Turn on **Auto Open-Drain Pins**. Compile the design after assigning the pins and settings in the Quartus II software.

Demo notes (to demo this design on the MDN-B2 demo board):

- Turn on the power to the demo board (using slide switch SW1).
- Download the design on to the MAX II CPLD through the JTAG header JP5 on the demo board and a conventional programming cable (ByteBlaster II or USB-Blaster).
- 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.

- Setting up a parallel port driven SMBus environment on your PC:
  - Download a software utility such as the Maxim parallel port utility to communicate to the slave in the SMBus defined protocol. Install the parallel port software. Here we illustrate using the program **ParDS2W.exe** at:  
[http://files.dalsemi.com/system\\_extension/AppNotes/AN3315/ParDS2W.exe](http://files.dalsemi.com/system_extension/AppNotes/AN3315/ParDS2W.exe)
  - You must install a parallel port driver to enable access to the parallel port in WinXP or Win 2000 for this parallel port utility. Direct-IO ([www.direct-io.com](http://www.direct-io.com)) has a typical driver that you can use and which you can download at:  
[www.direct-io.com/Direct-IO/directio.exe](http://www.direct-io.com/Direct-IO/directio.exe)
  - After installation, you must configure the Direct-IO program. Open the Windows control panel and click on the Direct IO icon. Enter the Begin and End addresses of your parallel port (most often this is 378 through 37F, but confirm your PC's parallel port address by looking at settings in:  
Control- Panel/System/Hardware/Device-Manager/Ports/  
ECP Printer port (LPT)/Resources
  - If your parallel port is configured to any other type other than ECP, change it to ECP by changing the BIOS settings during start up of your PC.
  - Select the **Security Tab** of the Direct IO control panel and browse to the directory path of the **ParDS2W.exe** program. Click **Open**, and subsequently **Add**. You will see the path of this utility in the **Allowed Processes** field. Click **OK** to close the control panel window.
  - Attach the parallel port SMBus/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.
  - Attach the 4-pin socket on the pig tail of I2C/SMBus parallel port dongle to header (JP3) of the demo board so that the red mark on the socket meets pin#1 on the JP3 header.
  - Open the ParDS2W program, select the appropriate parallel port address of your PC (as seen while configuring Direct IO), and also set the **2-Wire Device Address** to 00H.
  - Finally, you can test the set up by using the **Test Circuit** tab to see if you have a Test PASS message in the Status window. If you do, the required 2-wire environment is now set up.
- Using the **One byte Write/Read** section in the **2-Wire Utility** section of the parallel port utility program, you can now do a read operation to read from the GPIO inputs (DIP switch SW5) and a write operation to write any 8-bit data to be displayed on the 8 LED array on the MDN-B2 demo board.

- To read from the DIP switch SW5, simply press one of the **Read** buttons in this section (since this utility is meant for reading from a memory device, you must enter any 2-digit Hex value in the **Address** field before doing a **Read**). The Data field adjacent to it will now display the 8-bit positions corresponding to the DIP switch settings.
- To perform a write operation, enter any valid Hex data in the **Address** field (**Data** field must be filled with any 2-digit Hex, but is a “don’t care” in this context) in this section of the parallel port utility program, followed by pressing the **Write** button. The LEDs corresponding to the data will light up.
- A read and write operation can be performed at the same time by entering the data to be written in the **Address** field and then performing a **Read** button operation. The program would then send (write) the **Address** field data to the SMBus slave device and also perform a read operation immediately thereafter. This is possible because this utility program can also be used to read from a Memory slave device where the memory address must be written first before reading from it.
- The **2-Wire Function** section also has the ability to do Read/Write operations. However, this must not be used. The 2-Wire Function features manually drives SMBus operations (Start/Stop/Write/Read etc.). Manual operations are slow, because the time taken to enter and perform each of the individual functions is at manual speeds. This slow speed causes the SMBCLK line to be low for > 25 ms, causing the SMBus slave to reset itself. To avoid such behavior, always use the **2-wire Utility** section for this demo.

## Source Code

This design example has been implemented in Verilog HDL 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:

[www.altera.com/literature/an/an484\\_design\\_example.zip](http://www.altera.com/literature/an/an484_design_example.zip)

## Conclusion

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

## Additional Resources

The following list contains additional resources:

- MAX II CPLD homepage:  
[www.altera.com/products/devices/cpld/max2/mx2-index.jsp](http://www.altera.com/products/devices/cpld/max2/mx2-index.jsp)

- MAX II Device Literature:  
[www.altera.com/literature/lit-max2.jsp](http://www.altera.com/literature/lit-max2.jsp)
- MAX II Power-Down Designs:  
[www.altera.com/support/examples/max/exm-power-down.html](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

## Revision History

Table 4 shows the revision history for this application note.

<i>Table 4. Revision History</i>		
Date and Version	Changes Made	Comments
December 2007, v1.0	Initial release.	—



101 Innovation Drive  
San Jose, CA 95134  
[www.altera.com](http://www.altera.com)  
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.

