GPIO Pin Expansion Using I²C Bus Interface in Altera MAX Series

2014.09.22

AN-494 Subscribe Send Feedback

This design example shows the capability of Altera® MAX® II, MAX V and MAX 10 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.

The supported Altera devices 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.

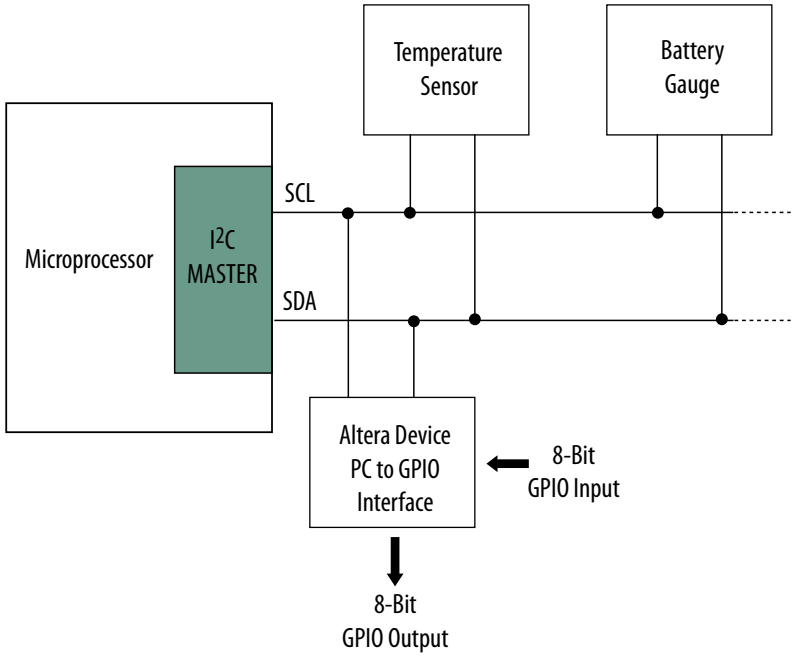
Related Information

- **Design Example for MAX II**
  Provides the MAX II design files for this application note (AN 494).
- **Design Example for MAX 10**
  Provides the MAX 10 design file for this application note (AN 494).
- **Power Management in Portable Systems Using MAX II CPLDs**
- **MAX II CPLD Design Guidelines**

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 device to serve various applications.

Figure 1: GPIO Pin Expansion via an I²C Bus



I²C Interface for GPIO Pin Expansion

The supported Altera device 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 device (which acts as an I²C slave). The device 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.

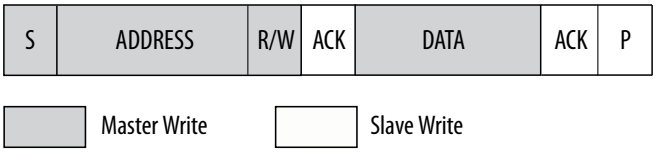
I²C Interface

For the I²C interface, the device (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.

Table 1: I2C Interface Pin Description

Signal	Purpose	Direction
SCL	I²C Clock	Output
SDA	I²C 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.

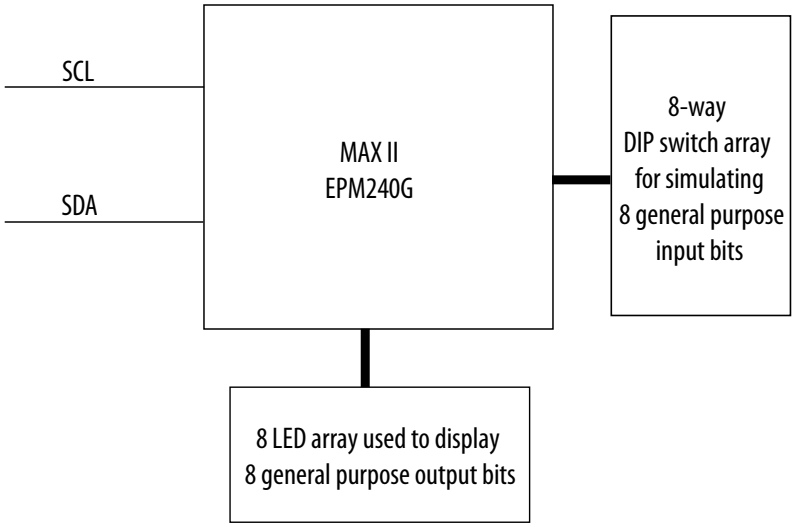
Table 2: GPIO Pin Description

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

GPIO Pin Expansion Using I²C Bus Interface Using MAX II Devices

The detailed description of the implementation is based on the MAX II devices. This application can also be implemented in MAX V and MAX 10 devices.

Figure 3: GPIO Pin Expansion Demonstration Circuit



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. This utility program uses the parallel port and its interfacing hardware to interact with the MAX II device 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 device. 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: 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

Note: 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.

Related Information

Dallas Semiconductor's Maxim application note AN3230
Provides details about setting up an I2C environment

Design Demonstration on the MDN-B2 Demo Board

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 device 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 I^2C 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 I^2C defined protocol. Install the parallel port software. (The ParDS2W.exe program downloaded from parallel port software from Direct-IO 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.
   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 I^2C 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 I^2C parallel port dongle to the I^2C 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 I^2C setup on the **Test Circuit** tab to see if you have a Test PASS message in the Status window. If you do, the I^2C environment is set.
5. With the parallel port utility you can now perform write and read operations in I^2C using the 2-Wire Functions.
6. To perform a write I^2C 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.

Related Information

Parallel port driver from Direct-IO

Acknowledgments

Design example adapted for Altera MAX 10 FPGAs by:

Orchid Technologies Engineering and Consulting, Inc.

Maynard, Massachusetts 01754

TEL: 978-461-2000

WEB: **www.orchid-tech.com**

EMAIL: **info@orchid-tech.com**

Document Revision History

Date	Version	Changes
September 2014	2014.09.22	• Added MAX V and MAX 10 devices. • Updated template. • Restructured document.
December 2007	1.0	Initial release.