GPIO Expansion Using UART
Design Example

Table of Contents

Overview

In order to keep cost to a minimum, microcontroller chips typically have a limited number of general purpose I/Os (GPIOs) available, so external I/O expanders are needed to utilize the microcontroller's built-in serial interfaces. Microsemi's ProASIC®3 FPGAs and IGLOO® FPGAs offer low-cost and low-power solutions for microcontroller GPIO expansion using a number of different techniques. The design example provided with this document enables users to implement a low-gate-count UART-to-GPIO expander.

Design files for this design example can be downloaded from the Microsemi website:

http://soc.microsemi.com/download/rsc/?f=GPIO_Expansion_DF.

Design Description

This design can be used as a standalone block in an FPGA design, which can be a companion IC for a microcontroller to add features or improve performance through co-processing, or it can be considered as a module for an FPGA-based embedded processor application. The design can be used in applications such as keyboards (inputs), LEDs/LCDs (outputs), or for meeting any bidirectional interface requirement.

The design example contains a UART interface and four bidirectional, 8-bit ports. The design is implemented on an IGLOO device on the *M1AGL-DEV-KIT-SCS* development board, but can be easily adapted to work in a production system or many of Microsemi's other demonstration boards. The UART can be used as the interface to a processor; however, in this example, it interfaces to a UART-to-USB converter chip, which is connected to a PC via USB. Commands are sent through UART to tell the design if it should read or write, which port to connect to, and the data to be read or written.

The top-level block diagram of the design is shown in Figure 1 on page 3. The design makes use of Microsemi's CoreUART for sending and receiving the serial data pattern. CoreUART works at a frequency of 20 MHz. This clock is derived from a 48 MHz external clock oscillator, with the help of a

PLL. However, a PLL is not required to implement this design if the correct external frequency is provided.

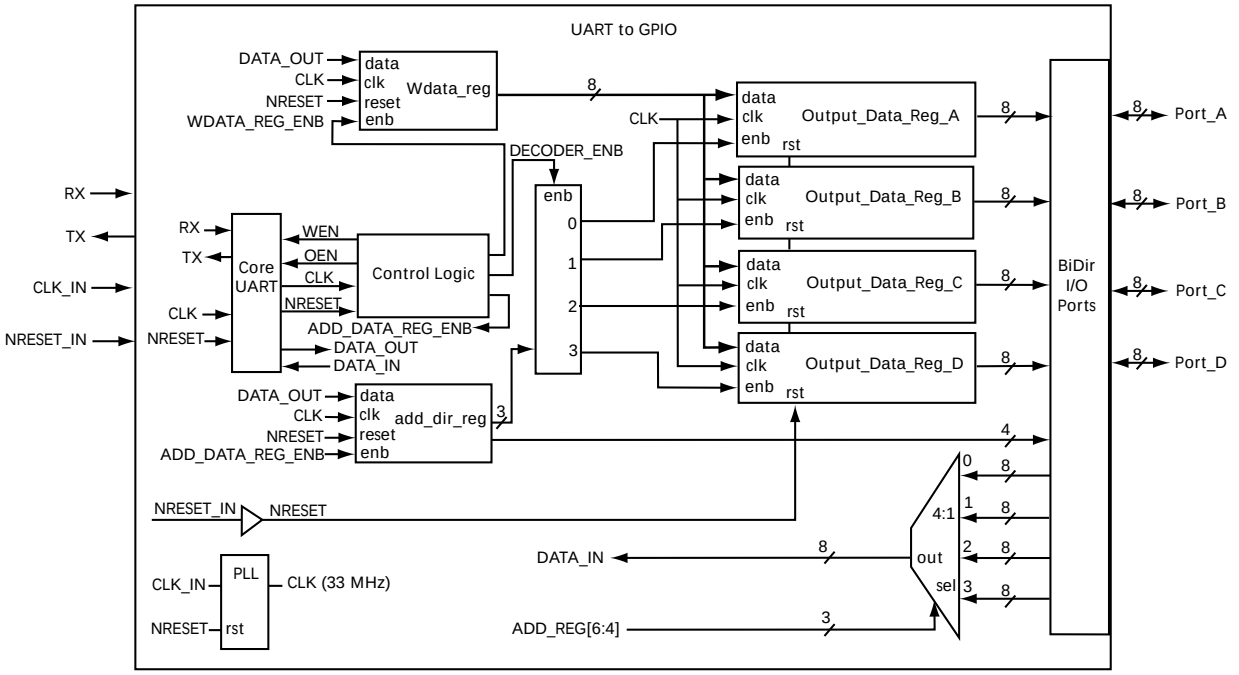

***Figure 1* • Address/Direction Register Details**

CoreUART must be configured with the following four parameters for this specific design.

baud_val

This parameter is required for setting the baud rate of the UART. It is a function of the system clock and the desired baud rate. The value should be set according to EQ 1.

$$\text{Baud rate} = \text{clk} / (\text{baud\_val} + 1) \times 16$$

EQ 1

In the IP core, input clock frequency for the UART Core is set to 20 MHz and baud rate is set to 9600. Changing the baud value changes the baud rate of the UART.

bit8

Control bit for setting the data bit width for both receive and transmit functions. When bit8 is logic 1, the data width is 8 bits; otherwise, the data width is 7 bits. In this design, bit8 is tied to 1.

parity_en

Control bit to enable or disable parity for both receive and transmit functions.

odd_n_even

Control bit to define odd or even parity for receive and transmit functions.

For a detailed description of CoreUART, refer to the *CoreUART handbook*.

The Control logic block interfaces the CoreUART with the rest of the logic. The necessary control signals for reading and writing from the UART as well as the GPIO interfaces are generated by the state machine of the Control logic block.

The ADD_DIR_Reg block shown in Figure 2 configures the port address and the direction (input or output) of the port.

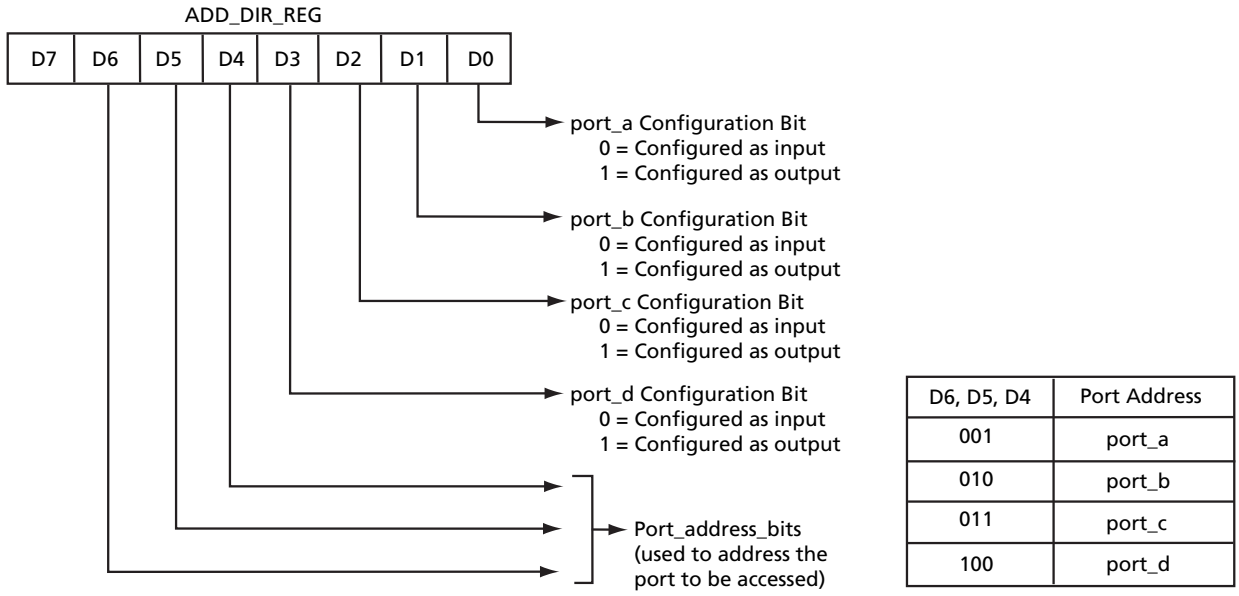

***Figure 2* • Address/Direction Register Details**

The three MSB bits of this register indicate the address of the port being addressed. The four LSB bits indicate the direction control bit of each of the four ports. For writing the data to any of the ports, that particular port must be addressed first, with the direction bit set as output. For reading data from any of the ports, the port must be addressed with the direction bit set as input.

The output data registers shown in Figure 1 on page 3 store the data being written to the GPIO ports. Depending upon the port being addressed, the corresponding Data Register will be updated. The contents of this register will be available, until that port content is overwritten.

The input data is selected, based on the port being addressed, and is fed to the data input of the CoreUART.

Interface Description

The Interface details of the IP are given in Table 1.

Table 1 • **Interface Description**

Port	Direction	Description
CLK_IN	Input	48 MHz input clock
NRESET_IN	Input	Active low Reset
RX	Input	Serial data in to the IP core
TX	Output	Serial data out for from the IP core
port_a	Bidirectional	8-bit data port a
port_b	Bidirectional	8-bit data port b
port_c	Bidirectional	8-bit data port c
port_d	Bidirectional	8-bit data port d

Utilization Details

This design was verified in the ARM® Cortex®-M1–enabled IGLOO M1AGL600V2-FG484 device, but can easily be instantiated in other IGLOO and ProASIC3 devices that contain the minimum required resources. The device utilization values for the M1AGL600V2-FG484 device are given in Table 2.

Table 2 • **Utilization Details**

Resource	Used/Total	Percentage
Core	307/ 13824	2.22%
Globals (chip + quadrant)	2/18	11.11%
PLL	1	100%
RAM/FIFO	0	0.0%

Software Interface and Details

The software for testing the IP is developed in the C language on a Windows® platform. The program configures the port direction as input or output and performs the read or write functions.

The software program is tested with Microsemi's IGLOO Development Kit (M1AGL-DEV-KIT-SCS). You will connect the pc running this software to the USB port of the IGLOO developments board.

The flow chart of a read/write cycle is shown in Figure 3.

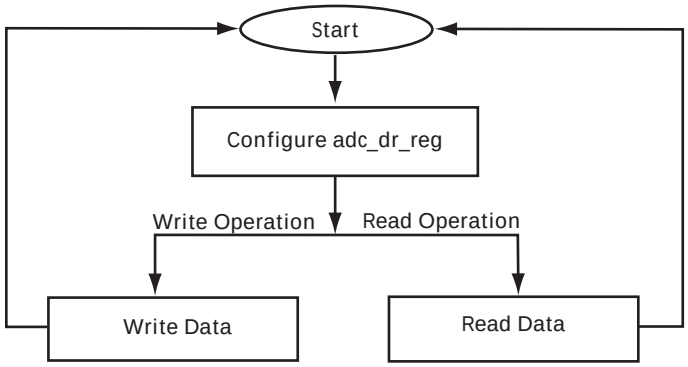

Figure 3 • **Read/Write Flow Chart**

First, the port to be accessed must be addressed with the direction (input or output). This step must be followed by either sending or receiving data, depending upon whether write or read was selected.

The usage of the software is covered in the "Program Execution" section.

Program the M1AGL600 device in the IGLOO development kit (M1AGL-DEV-KIT-SCS) using the STAPL file included with the design files.

Program Execution

When you execute the program GPIO_Expansion.exe, the program prompts you with the menu options required for data transfer. The program runs in a continuous loop, waiting for your input. To exit the program, press Ctrl-C.

The menu options are shown in Figure 4 through Figure 8 on page 7.

COM Port Selection

Select the communication port, which is the USB Port through which data will be transmitted or received. The IGLOO development board J2 is connected to this port on the PC/Laptop. Figure 4 shows the program prompt for the COM Port.

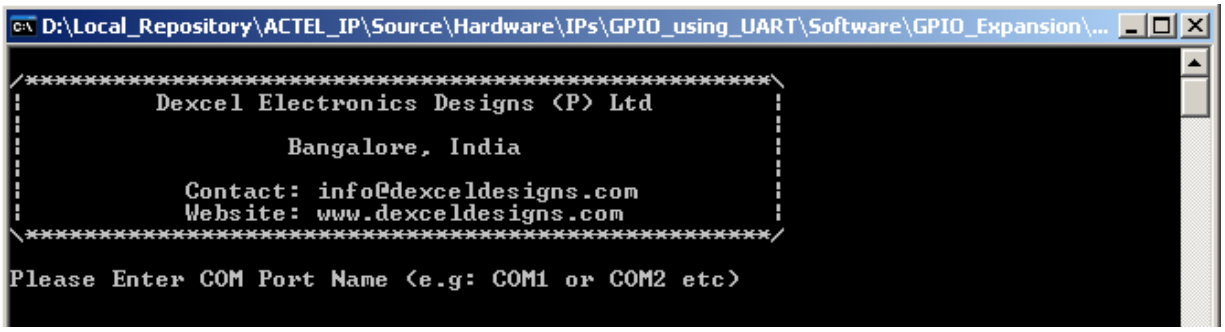

***Figure 4* • Entering the COM Port**

You can identify the COM port to which the IGLOO development board is connected as follows, beginning from the Windows Start menu: **Start** > **Settings** > **Control Panel** > **System** > **Hardware** > **Device Manager** > **Ports.**

For the IGLOO development board, the USB to UART bridge controller device CP2101 is listed, as shown in Figure 5.

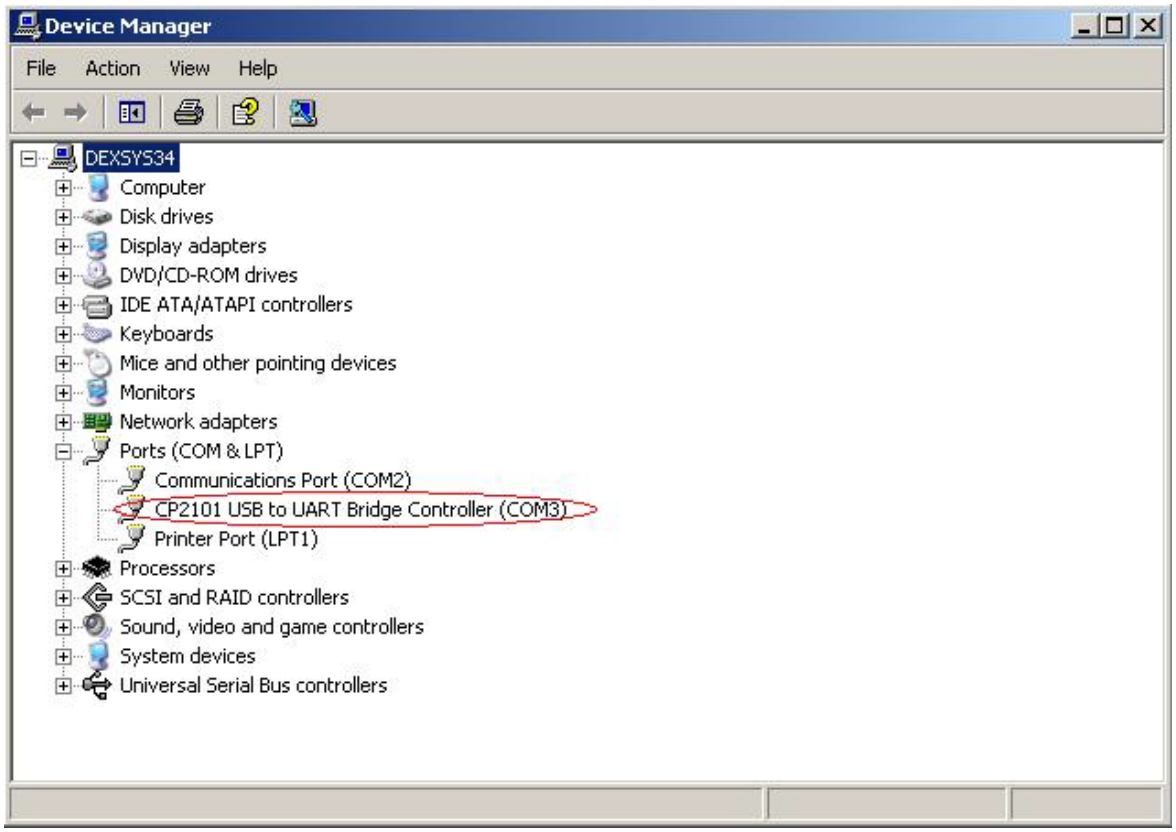

***Figure 5* • COM Port Identification**

In this example, the board is connected to COM3.

After entering the COM port, you are prompted to select the mode of operation.

Operation Mode Selection

Select whether to perform a read or write operation to the selected port.

Port Selection

Select the port intended for data transfer by typing the number corresponding to PortA, PortB, PortC, or PortD, as shown in the menu in Figure 6.

On the IGLOO development board, PortC is mapped to the bank of LEDs and PortD is mapped to the bank of switches, SW2. PortA and PortB are mapped to headers P5 and P1, as indicated in Table 3.

Table 3 • **Port Mapping**

Port	FPGA Pin	PCB Signal	Header
port_a(0)	A6	GPIOC_1	P5:5
port_a(1)	A7	GPIOC_3	P5:7
port_a(2)	A10	GPIOC_5	P5:9
port_a(3)	A11	GPIOC_7	P5:11
port_a(4)	B6	GPIOC_9	P5:13
port_a(5)	B7	GPIOC_11	P5:15
port_a(6)	B10	GPIOC_13	P5:17
port_a(7)	AB17	GPIOC_15	P5:19
port_b(0)	D10	GPIOA_2	P1:6
port_b(1)	G11	GPIOA_4	P1:8
port_b(2)	D11	GPIOA_6	P1:10
port_b(3)	H11	GPIOA_8	P1:12
port_b(4)	E12	GPIOA_10	P1:14
port_b(5)	G12	GPIOA_12	P1:16
port_b(6)	F13	GPIOA_14	P1:18
port_b(7)	E13	GPIOA_16	P1:20

Figure 6 • **Selecting the Ports**

Data to be Read/ Written

After the port selection, the read value is displayed in the console for a read operation (Figure 7).

```
2

Select Port
 1. for PortA
 2. for PortB
 3. for PortC
 4. for PortD

Press Ctrl-C to exit
4

Data read is 0x55
```

***Figure 7* • Data Read from Ports**

If it is a write operation, you must enter the value to the console. The data is reflected in the corresponding port that is selected (Figure 8). Note that PortD is not available in write mode in order to avoid conflict with the hardwired drivers on the M1AGL-DEV-KIT-SCS.

```
1

Select Port
 1. for PortA
 2. for PortB
 3. for PortC
Press Ctrl-C to exit

3

Select Data between 0x00 to 0xFF to Transmit
Press Ctrl-C to exit
55
```

***Figure 8* • Entering the Data to Be Written**

The program is written in C using the Visual C++ compiler 6.0. The files used for this application are GPIO_Main.c and UsbCom.c.

GPIO_Main.c

This file provides the main functionality of the program. User interaction and communication with the USB port is done inside this file. User input is validated and sent to the USB port sequentially.

UsbCom.c

This source file manages the USB communication.

Testing Scheme

Verification of the core is done by simulation in Model*Sim*® for best case and worst case timing delays. The simulation testbench generates the RX (serial input), clock, and reset input for the design. The simulation results for write and read cycles are illustrated in Figure 9 and Figure 10 on page 8.

Hardware validation was done on Microsemi's IGLOO development board (M1AGL-DEV-KIT-SCS). The application GPIO_Expansion.exe is used for communicating with the GPIOs. Depending upon the switch settings, the input values are displayed. Based on the output data entered, the LED pattern changes.

Timing Diagram

Figure 9 gives the sequence of a write cycle to port_c.

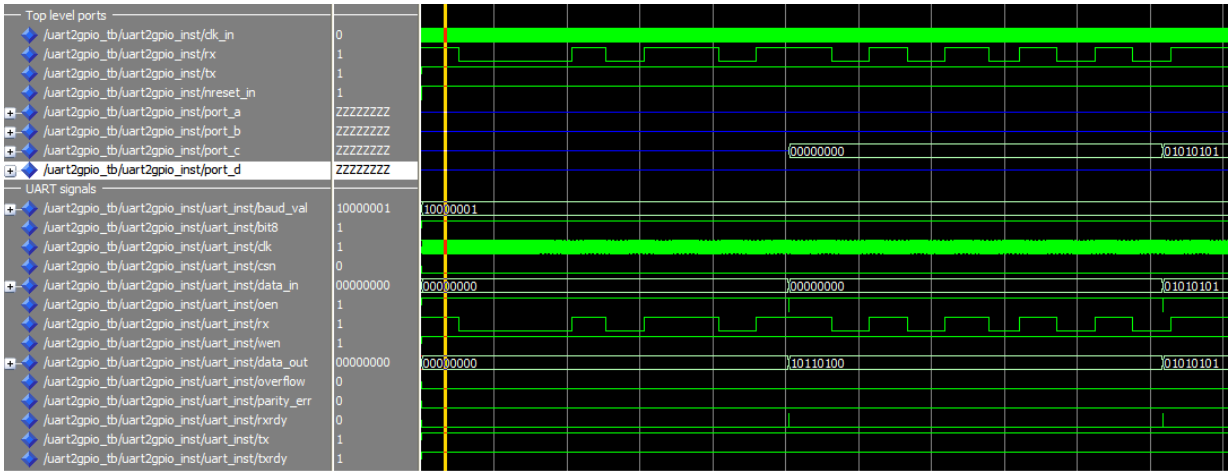

***Figure 9* • Write Cycle**

Figure 10 gives the sequence of a read cycle from port_b.

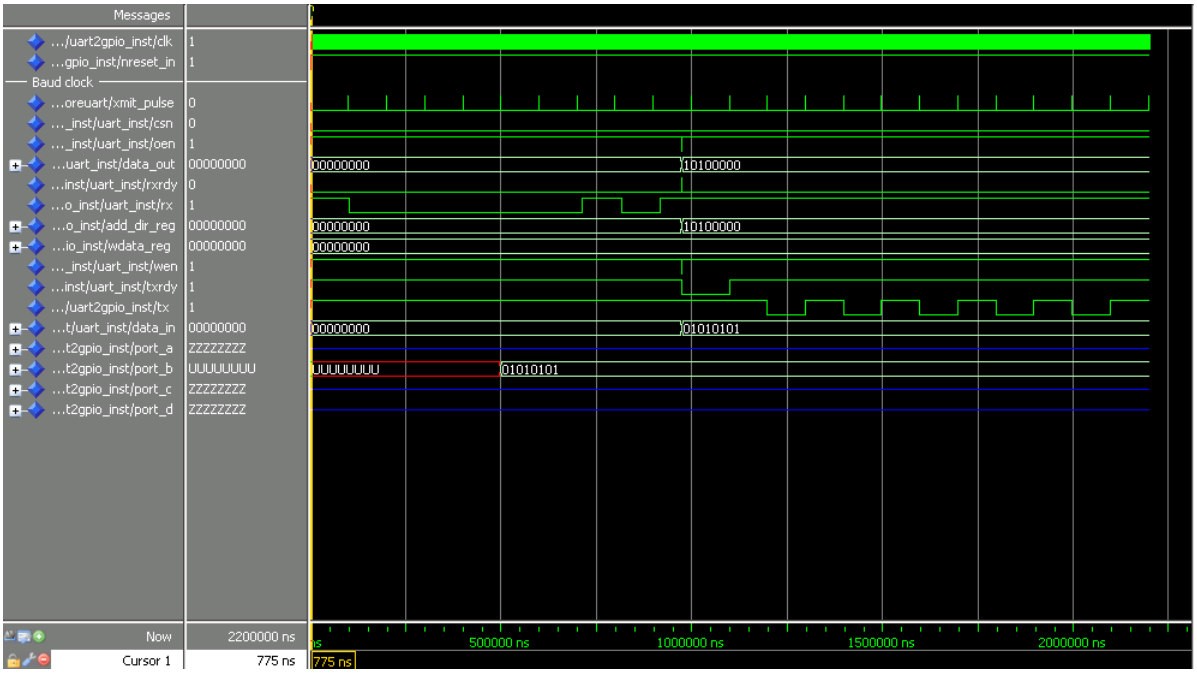

***Figure 10* • Read Cycle**

Conclusion

When targeted to Microsemi's low-power, low-cost IGLOO and ProASIC3 flash-based FPGAs, this design becomes a simple, low-power, economically viable solution as an I/O expander. System designers can continue to use low-cost processors by solving the requirement of more I/Os, features, or co-processing through the use of FPGAs, maximizing overall system and development costs.

List of Changes

The following table shows important changes made in this document for each revision.

Date	Changes	Page
Revision 1 (June 2015)	Non-technical Updates.	NA
Revision 0 (April 2009)	Initial Release.	NA

*Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.*

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

Microsemi Corporation (MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,600 employees globally. Learn more at **www.microsemi.com**.

51900194-1/06.15