

8-bit Universal Asynchronous Receiver Transmitter

Example Name: Example_UART

Programming Language: C

Associated Part Families: CY8C29/27/24/22/21xxx, CY8C23x33, CY7C64215, CYWUSB6953, CY8CNP102, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120 CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300 CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28x43, CY8C28x52

Software Version: PSoC[®] Designer[™] 5.1 SP2

Related Hardware: [CY3210 PSoCEval1](#)

Objective

The operation of the 8-bit universal asynchronous receiver transmitter (UART) user module in PSoC[®] is demonstrated in this example.

Overview

A command and some parameters (separated by spaces) are transmitted to the PSoC by a personal computer using Windows Hyper Terminal, through the serial port. The data is received and decoded by PSoC and the command and parameters are echoed back to the PC (line by line).

User Module List and Placement

The following table lists the hardware resources occupied by the user module.

User Module	Placement
UART	DCB02 (TX Configuration) DCB03 (RX Configuration)

User Module Parameter Settings

The following table lists the UART user module parameter settings.

UART		
Parameter	Value	Comments
Clock	VC3	Input is 153.8 KHz clock (eight times baud rate - 19200). VC3 = Sys.Clk / 156
RX Input	Row_0_Input_1	Routed from pin P2[5] through GlobalInEven_5
TX Output	Row_0_Output_3	Routed to pin P2[7] through GlobalOutEven_7
TX InterruptMode	TXRegEmpty	Not used
Clock Sync	Sync to Sys.Clock	Clock is synchronized with the SysClk, as VC3 is a derivative of SysClk
RxCmdBuffer	Enable	Enables Command buffer, and stores the received command in a RAM buffer
RxBufferSize	32	Bytes length of buffer is 32 characters (including parameters)
Command Terminator	13	Carriage return (13) is the command terminator
Param_Delimiter	32	Space (32) is the parameter delimiter
IgnoreCharsBelow	32	Ignore control characters that have ASCII value below 32
Enable Backspace	Disable	Not used
Rx Output	None	Not used
Rx Clock Out	None	Not used
Tx Clock Out	None	Not used

UART		
Parameter	Value	Comments
InvertRX Input	Normal	Do not invert Rx Input

Note The clock to the TX8 user module should be eight times the desired baud rate. The parameters Command Terminator, ParamDelimiter, and IgoneCharsBelow are used by the high level UART API to process the commands received. When the UART receives the character set as Command Terminator, the API reports through a flag that a valid command has been received. When the character set as ParamDelimiter is encountered, the API understands that the characters that follow the ParamDelimiter are the next parameter in the command. The API ignores any characters below the value set in IgonreCharsBelow value and does not store them in the command buffer.

Global Resources

Important Global Resources		
Parameter	Value	Comments
CPU_Clock	SysClk/2	Sets the CPU frequency to 12 MHz
VC3 Source	SysClk/1	Set System Clock as the source for VC3
VC3 Divider	156	Divide 24 MHz system clock by 156 (effective baud rate is 19.2 kbps)

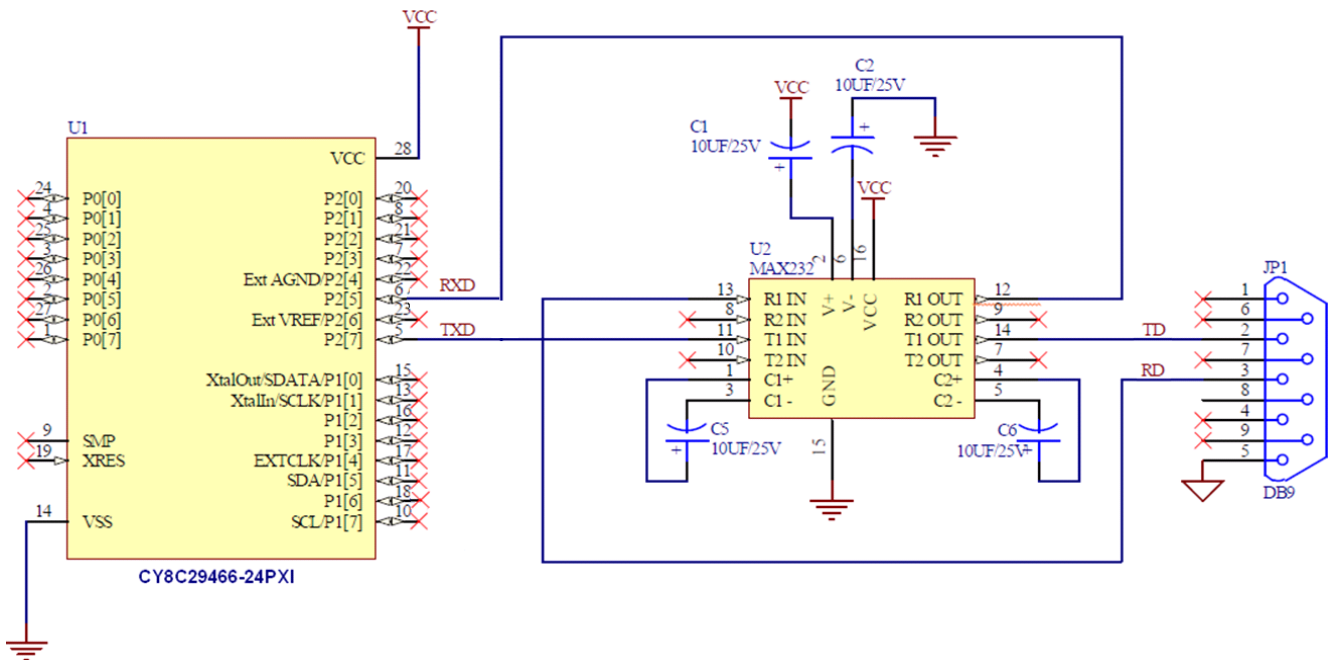
Note Leave all other global resources at their default.

Pin Configuration

Pin	Select	Drive	Interrupt
P2 [7]	GlobalOutEven_7	Strong	DisableInt
P2 [5]	GloballnEven_5	High Z	DisableInt

Hardware Connections

Figure 1.Schematic Diagram



MAX232, a RS232 transceiver, is used to translate the ± 10 V RS232 signals to transistor-transistor logic (TTL) level signals of the PSOC.

JP1 is a 9 pin female serial port connector is used to connect the project with a PC.

The code can be tested using [CY3210 – PSOC Eval1](#) board. This board has a RS232 transceiver and a serial port connector. To test the code using the CY3210 board, make the following connections:

- Connect P25 of J7 to RX of J13
- Connect P27 of J7 to TX of J13

Operation

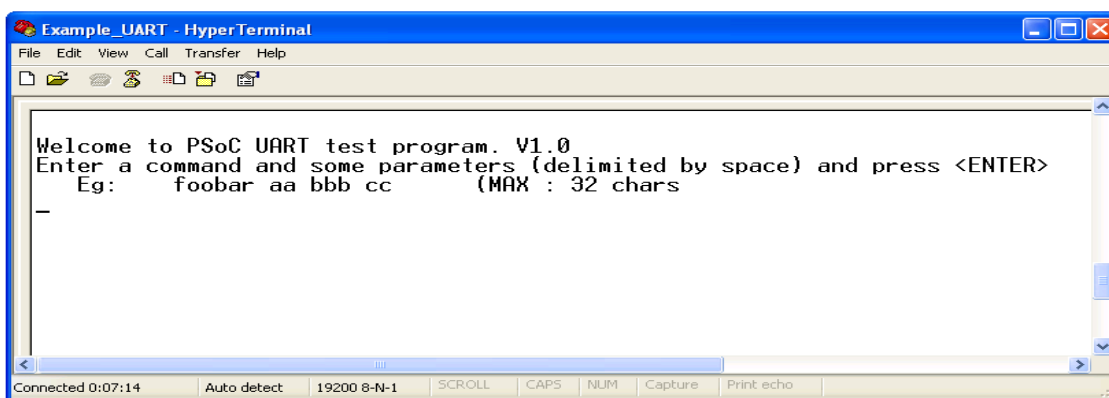
All hardware settings from the device configuration are loaded into the device and *main.c* is executed on program execution.

The 24-MHz system clock is divided by 156 (VC3) to produce a 153.8-KHz clock, which is provided to the UART user module. The transfer rate is 1/8 times the clock, that is, 19.2 kbps. Parity is set to none in the initial configuration for the block and may also be set using the UART APIs.

After setting up the **HyperTerminal** as explained in the section “Testing the Project” on page 5, click **Call** menu in the **HyperTerminal** and reset the PSoC.

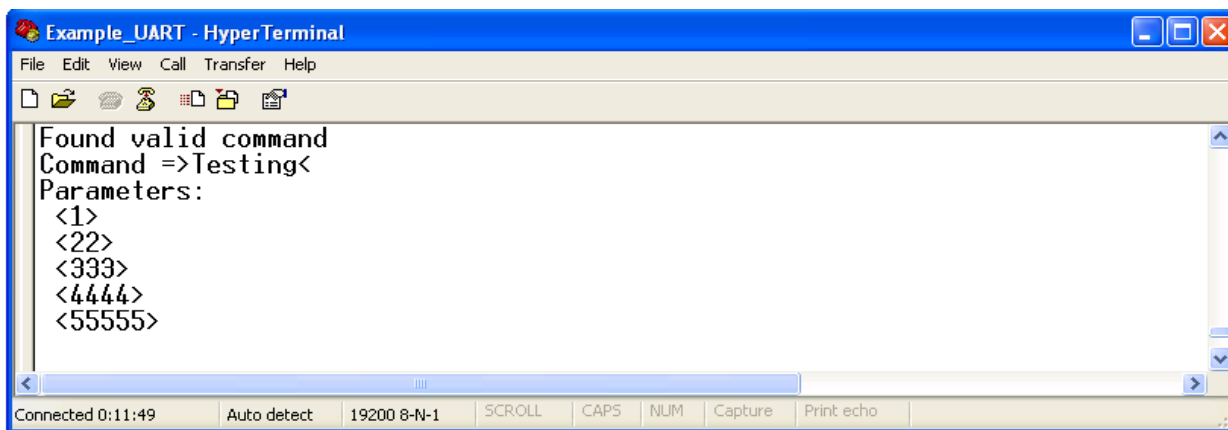
The **HyperTerminal** window displays the following message.

Figure 2. Welcome Screen on HyperTerminal



PSoC decodes the command and parameters according to the commands typed on the HyperTerminal. For example, when a string “Testing 1 22 333 4444 55555” is entered in hyper terminal, PSoC responds as shown in the following figure.

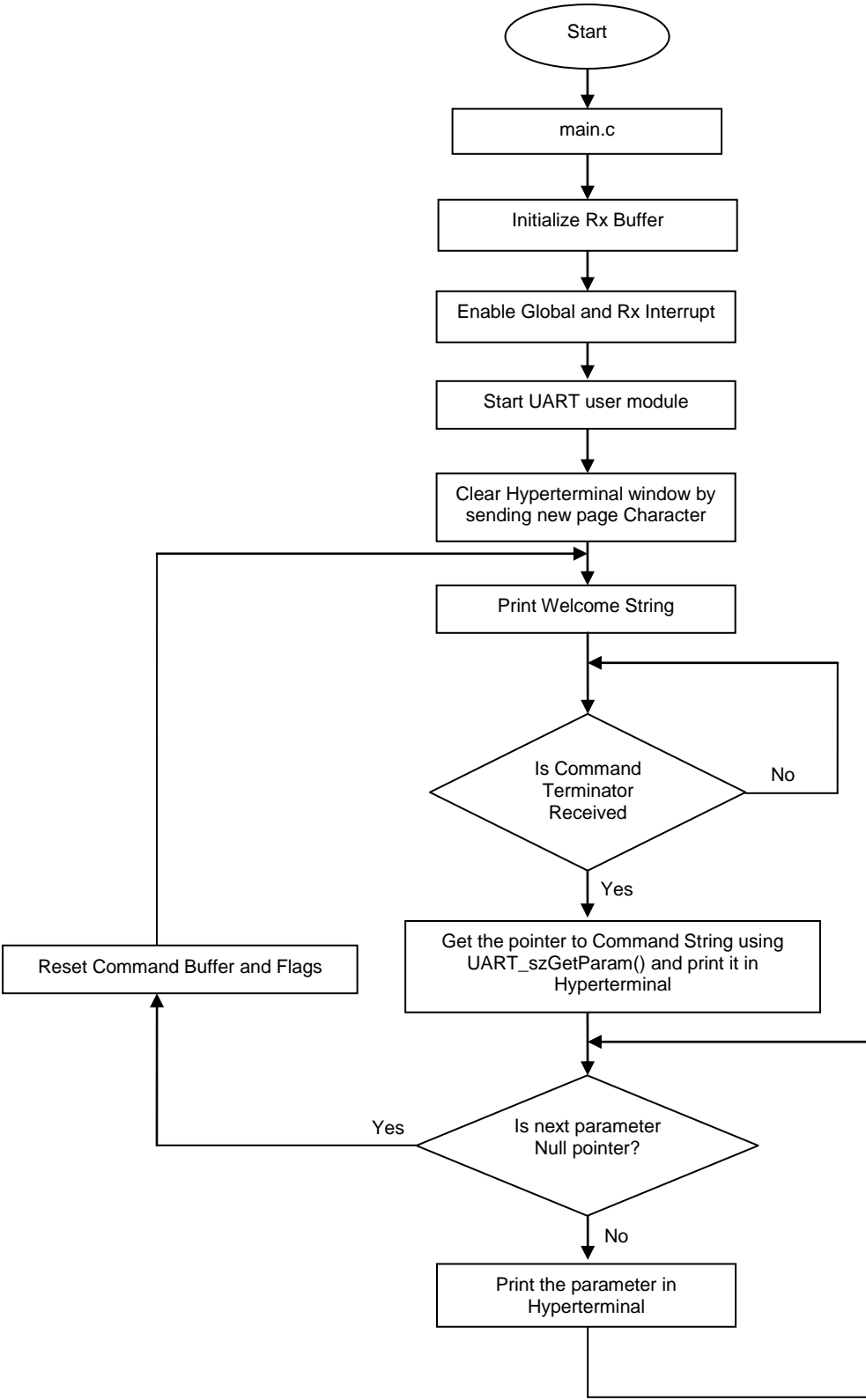
Figure 3. PSoC Response in HyperTerminal



Note For more information on user modules, see the specific user module datasheet included in the PSoC Designer™.

The flowchart of the program execution is shown as follows.

Figure 4. Program Execution Flowchart



Testing the Project

Use **HyperTerminal** (or any other terminal program) to test the project. Configure **HyperTerminal** as follows in Windows.

1. Connect the CY3210 board to the PC serial port using a serial port cable.
2. Start **HyperTerminal** using **Start → All Programs → Accessories → Communication → HyperTerminal**
3. Enter a Name for the connection, for example **Example_UART** and click **OK**.

Figure 5. Give a Name to the Connection



4. In the **Connect To** option, select the desired serial port (for example, COM2) from **Connect using** list and click **OK**.

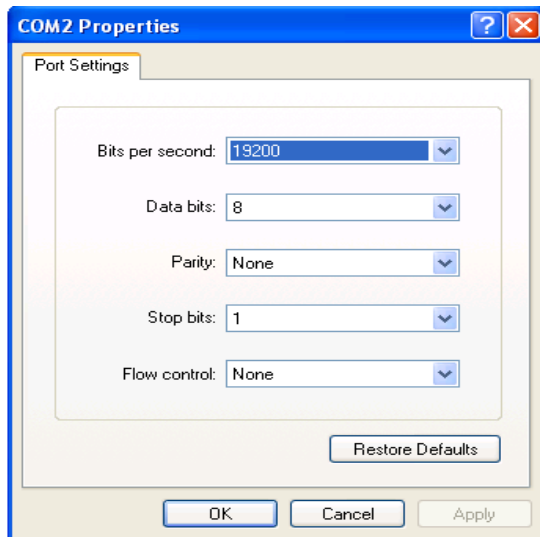
Figure 6. Select the Serial Port



5. In the **COM2** properties, configure the following parameters:

- Bits per second = 19200
- Data bits = 8
- Parity = None
- Stop Bits = 1
- Flow Control = None
- Click **Apply** and then click **OK**.

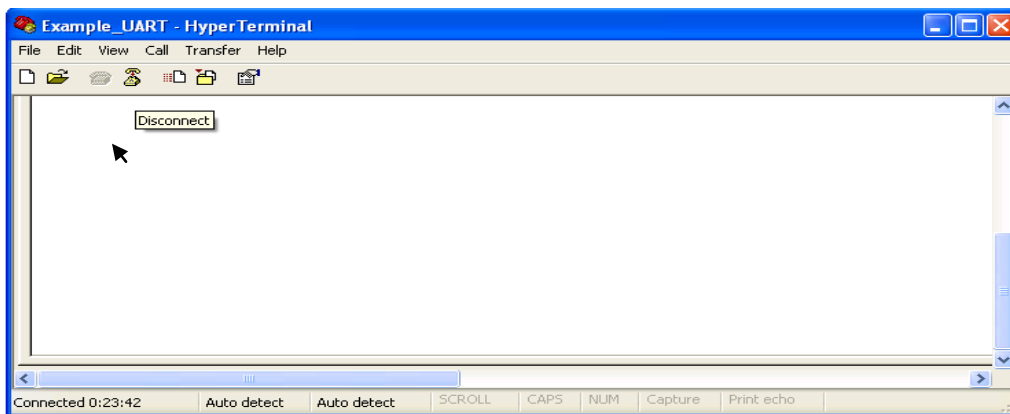
Figure 7.COM2 Properties Window



6. At this point, **HyperTerminal** connects to **COM2**.

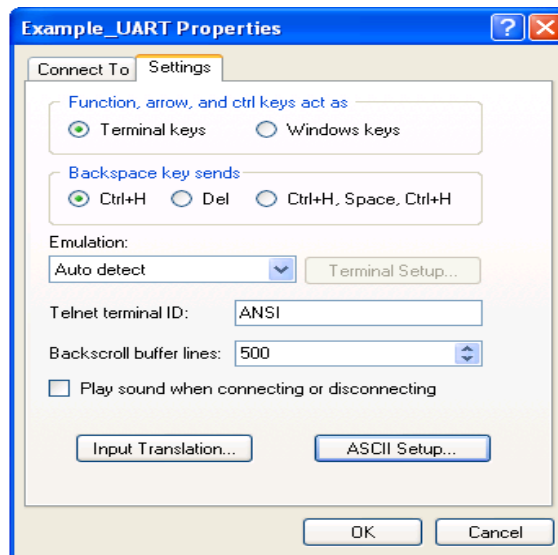
7. Click **Disconnect**.

Figure 8.Disconnect the Connection



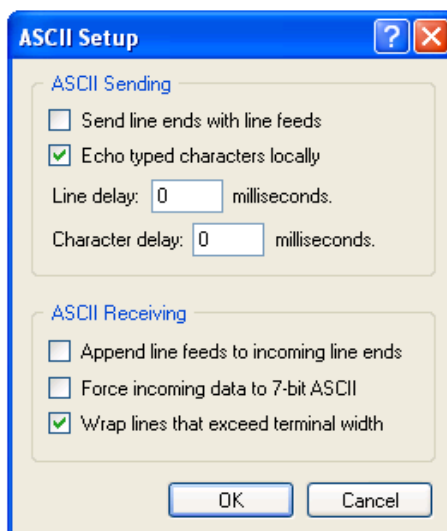
8. Select **File** → **Properties** → **Settings** menu and click **ASCII Setup**.

Figure 9. UART Properties Window



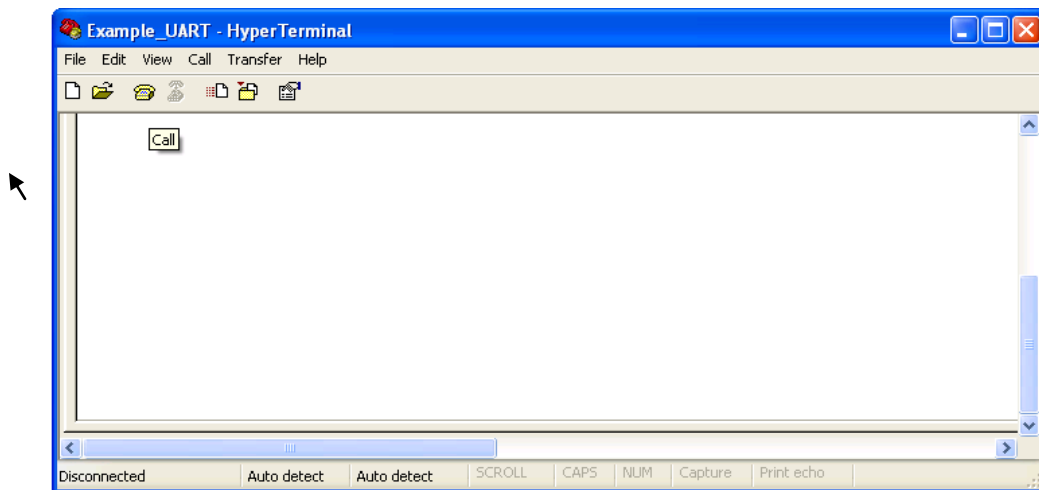
9. Enable **Echo typed characters locally** and click **OK**.

Figure 10. Setting Up ASCII



10. Click **Call**.

Figure 11. Connect to the Serial Port



11. Power-up the PSoC. The **HyperTerminal** displays the welcome screen. Type the command followed by parameters in **HyperTerminal**. PSoC decodes the command and echoes back to **HyperTerminal**.

PSoC is a registered trademark of Cypress Semiconductor Corp. PSoC Designer is a trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
<http://www.cypress.com/>

© Cypress Semiconductor Corporation, 2009-2011. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.