AN1130

Interfacing the CY7B923 and CY7B933 (HOTLink®) to Clocked FIFOs

Associated Project: No
Associated Part Family: CY7B923 / CY7B933
Software Version: NA
Related Application Notes: AN1162

To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/go/AN1130.

AN1130 discusses the interfacing issues between the Cypress HOTLink® Transmitter/Receiver and Cypress Clocked FIFOs.

Introduction

This application note describes the interfacing issues between the Cypress CY7B923/CY7B933 (HOTLink®) transmitter/receiver and Cypress clocked FIFOs. The HOTLink-FIFO interface is capable of performing parallel bus transactions at rates of up to 33 Mbytes/s and serial transfers at rates of up to 330 Mbits/s. The FIFO serves as an asynchronous storage buffer between the data bus and the serial link.

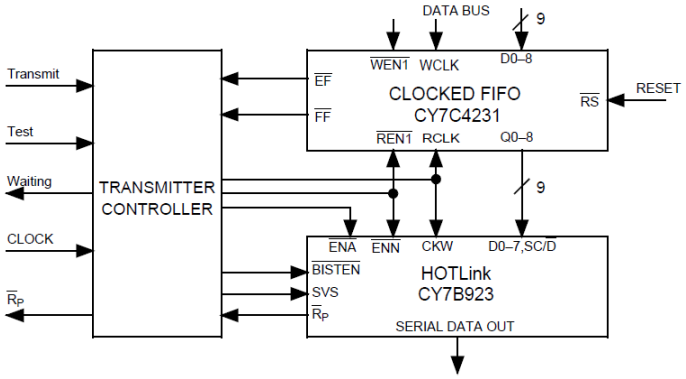
Transmitter Interface

This section describes the design considerations of a high-speed serial transmitter with First-In First-Out (FIFO) data buffers. The interface design supports basic data transmission control and serial link testing. The transmitter design is intended to interface to a higher-level system controller responsible for handling bus transactions and the serial link protocol. The interface is a primitive building block that is easily modified to meet system requirements.

Data Path and Controller

The transmitter interface consists of a single CY7C4231 clocked FIFO interfacing directly to the HOTLink Transmitter. A transmitter controller supplies the control signals to both the FIFO and the HOTLink Transmitter. The architecture of the controller is left unspecified, but it can be implemented in a CPLD. State diagrams and generic timing diagrams are provided. A block diagram of the transmitter interface is shown in Figure 1.

Figure 1. Transmitter Interface Block Diagram



Built-In Self-Test

The transmitter is capable of checking the functionality of the transmitter serial connection by exercising the Built-In Self-Test (BIST) mode of HOTLink. To initiate BIST, the $\overline{BISTEN}$ pin is held LOW, resulting in the transmission of the repeating character 1010101010. The HOTLink $\overline{ENA}$ (Enable Parallel Data) pin is then pulled LOW to enable transmission of the BIST test pattern. The HOTLink Transmitter will assert the $\overline{R}_P$ (Read Pulse) pin HIGH at the beginning of BIST and will pulse it LOW once per BIST loop. During BIST, HOTLink ignores data at its parallel port and the FIFO must not perform any reads.

Resetting the FIFO

The higher-level controller should reset or clear the FIFO at power-up, before a new block of data is transmitted, or if an error occurs. Resetting the FIFO is accomplished by asserting the $\overline{RS}$ (Master Reset) pin on the FIFO LOW. Neither a read nor a write can occur on the cycles immediately preceding, during, or following the assertion of $\overline{RS}$. To insure that this condition is met, the interface controller must be in the IDLE state (Figure 2) during the entire Master Reset cycle. Proper FIFO reset also requires that $\overline{RS}$ be glitch free. The higher-level controller is responsible for coordinating the read and write ports and insuring that the reset conditions are met.

Controller State Description

For applications requiring high-speed asynchronous data buffering, the FIFO read and write ports should be controlled by separate control circuitry synchronized to the FIFO ports. The FIFO write port interfaces directly to a 9-bit data bus. Data is written into the FIFO by asserting $\overline{WEN1}$ to enable the write clock (WCLK). Data may be written at any time as long as the FIFO is not full (as indicated by the FIFO full flag) and a FIFO reset cycle is not in progress.

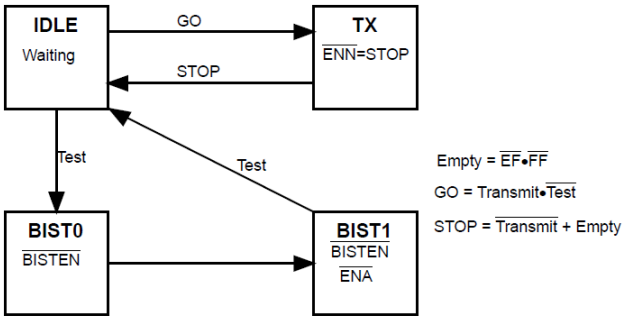
The FIFO read port interfaces to the HOTLink transmitter parallel port. Control of this interface is the focus of this section. The transmitter interface state machine controls FIFO-HOTLink data transactions and initiates the HOTLink BIST. The interface state machine is under the control of a higher-level controller responsible for both the serial protocol and the data bus/FIFO transactions.

The interface controller is a simple state machine as shown in

. While the state machine waits in the IDLE state, HOTLink will transmit Sync fill characters (K28.5). When the Transmit signal is asserted by the higher-level controller, the transmitter state machine transitions to the TX state. The TX state reads 9-bit words out of the FIFO into the HOTLink Transmitter until a Stop condition is detected (the FIFO is empty or the Transmit signal is deasserted). Reading data from the FIFO is accomplished by asserting $\overline{REN1}$ LOW. The same signal is connected to $\overline{ENN}$ (Enable Next Parallel Data) pin of HOTLink. Assertion of ENN causes data on the next rising edge of the clock to be latched into the HOTLink Transmitter. The functionality of the $\overline{ENN}$ pin is specifically designed to operate with the pipelined architecture of clocked FIFOs. After a Stop condition is detected, the state machine returns to the IDLE state and asserts the Waiting signal.

The state diagram includes test states for exercising the BIST capabilities of HOTLink. The BIST loop is entered when the higher-level controller asserts the Test signal while the transmitter state machine is in the IDLE state. The BIST0 state asserts $\overline{BISTEN}$ to initiate to the transmission of the repeating character 1010101010. The BIST1 state then asserts $\overline{ENA}$ to start the BIST pattern generation. The higher-level controller could monitor $\overline{R}_P$ to count the number of BIST patters sent. BIST will conclude when the higher-level controller deasserts Test after the desired number of BIST patterns has been sent. Control then returns back to the IDLE state.

Figure 2. Transmitter Controller State Diagram



Critical Timing Analysis

Timing diagrams are provided for the transmitter interface. The analysis assumes that the state machine state bits are accessible sooner than any data or input control signal.

FIFO-HOTLink Transmitter Data timing is governed by the FIFO access time (t_A = 10 ns) and the data setup time for HOTLink (t_{DS} = 5 ns).

$$t_A + t_{DS} \le t_{WCLK} \qquad \text{Equation 1}$$

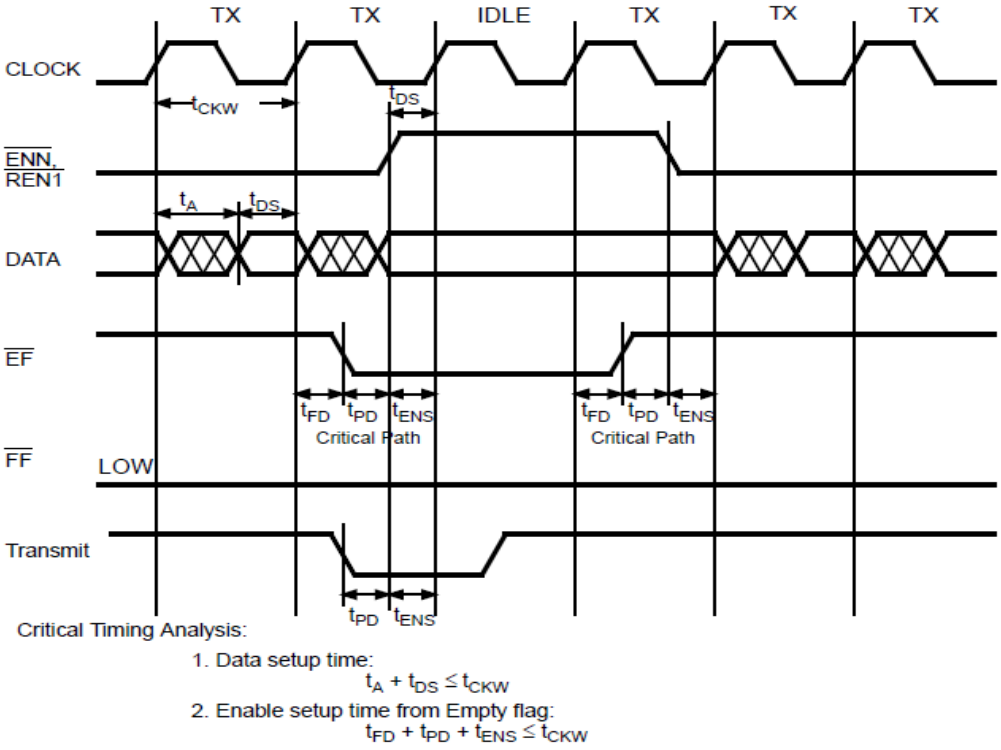
With clock periods greater than 30 ns, the data has no trouble meeting these timing constraints.

The critical timing path of the FIFO-HOTLink Transmitter interface is due to the delay associated with decoding the flags and generating the enable for the clocked FIFO ($\overline{REN1}$) and HOTLink ($\overline{ENN}$). Note that these are the same signals, but $\overline{REN1}$ requires a longer setup time than $\overline{ENN}$. The delay due to the state machine decoding the flags and generating the enable is represented as t_{PD}. The FIFO flag delay, t_{FD}, is 10 ns. The read enable setup time for the FIFO, t_{ENS}, is 7 ns.

$$t_{PD} \le t_{WCLK} - t_{ENS} - t_{FD} \qquad \text{Equation 2}$$

A 30-ns clock period leaves the controller 13 ns to generate the $\overline{ENN}$ signal. A timing diagram is provided in Figure 3.

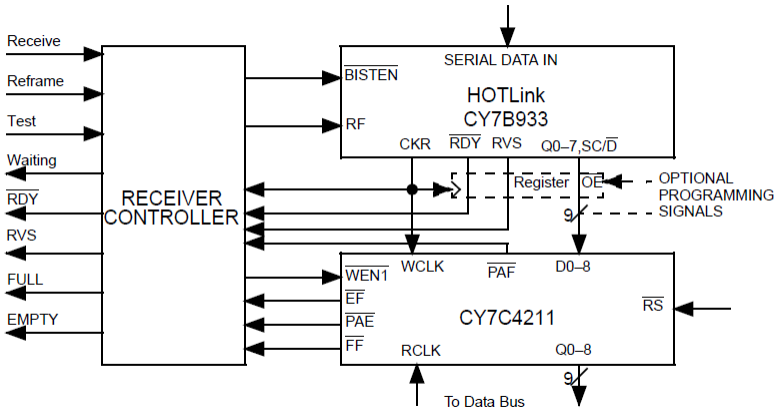
Figure 3. Transmitter Timing Diagram



Receiver Interface

The receiver interface uses a single CY7C4231 clocked FIFO to buffer the parallel data presented by the HOTLink Receiver. The CY7C4211 FIFO features programmable flags and three-state output drivers for bus applications. The HOTLink receiver interface is capable of receiving serial data at rates of up to 330 Mbits/second and then writing 9-bit words in the FIFO. Words in the FIFO can be read to the data bus at rates of up to 70 MBytes/s. A higher-level controller is responsible for coordinating the receiver interface and bus transactions according to the serial link protocol. Figure 4 shows a block diagram of the receiver HOTLink-FIFO interface.

Figure 4. Receiver Interface Block Diagram



Reframe

The HOTLink serial receiver must synchronize itself with the proper word alignment of the incoming data. Assertion of the HOTLink RF (Reframe) input forces HOTLink to synchronize its internal bit counter with the boundary of a received K28.5 character. HOTLink will respond by asserting $\overline{RDY}$ LOW when the first K28.5 is received. The receiver state machine controller should be designed to synchronize HOTLink at the beginning of data reception or after excessive errors have been received.

Data Path and Controller

The receiver state machine responds to control signals from a higher-level controller. The higher-level controller initiates data reception by asserting the Receive signal to the receiver state machine. Nine-bit words from the HOTLink parallel port are stored into the CY7C4211 FIFO each time $\overline{RDY}$ is asserted LOW. $\overline{RDY}$ will pulse LOW when new data is available at the HOTLink parallel port and will be HIGH when a pad sequence is received (multiple K28.5 SYNC codes). $\overline{RDY}$ is used to prevent the FIFO from filling with SYNC characters. Data storage will stop immediately when Receive is deasserted.

If the FIFO becomes full, it will ignore attempted writes. Full and Empty flags are decoded so that the higher-level controller can detect when the FIFO contains data or is completely full.

The CY7C4211 features programmable Almost Full and Almost Empty flags. The distance that these flags become active from the Empty and Full FIFO boundary is programmed during the FIFO Master Reset cycle. The distance can be set such that a flag is asserted when a fixed length packet of data has been received. The higher-level controller responds to the flag by reading the data packet out of the FIFO. The Almost Full flag is useful for preventing data from being lost. This flag can be programmed to compensate for the response latency of the higher-level controller so that data can be read from the FIFO before it becomes full. The decoding of the programmable flag signals is left out of the controller design for clarity.

Optional Pipeline Register

The optional pipeline register increases interface speed by capturing the $\overline{RDY}$ pulse and easing the control signal timing margins. $\overline{RDY}$ is a delayed 60% LOW duty cycle signal shaped for asynchronous FIFOs. Without the pipeline register, the LOW phase of $\overline{RDY}$ leaves less than ½t_{CKR} – 10 ns to generate the FIFO write enable and meet the set-up time. A clock period of 40 ns (250 Mbit/second) leaves a manageable 10 ns for the receiver state machine to generate the FIFO write enable, but as the clock period decreases to 30 ns (330 Mbit/second), the enable generation time shrinks to only 5 ns. This timing difficulty is overcome by pipelining the interface. The data and status signals must be pipelined to insure the proper word is written into the FIFO. The timing implications are considered in the section on critical timing analysis.

A data pipeline register with three-state output drivers can also be used to isolate the HOTLink Receiver parallel port from the FIFO write port while programming the CY7C4211 FIFO flags. A 9-bit program word from an external source can be written into the FIFO during a Master Reset cycle. The program word sets the Almost Empty and Almost Full flags and sets the FIFO parity option.

Resetting and Programming the FIFO

The higher-level controller should perform a FIFO Master Reset cycle after power-up, before new data is received, if an error occurs, or in order to program the FIFO flags. A Master Reset cycle is accomplished by asserting the RS pin on the FIFO LOW. Proper resetting or programming requires that $\overline{RS}$ be glitch free. In addition, neither a read nor a write can occur on the cycles immediately preceding, during, or following the assertion of $\overline{RS}$ unless the FIFO is being programmed. If the FIFO is not being programmed, the receiver state machine should remain in the WAIT state during the Master Reset cycle.

In order to program the FIFO, the higher-level controller should put the data pipeline register in the high impedance state. The program word is then supplied to the FIFO by an external source (data bus, controller, and so on). This word is written into the FIFO internal program register during the Master Reset cycle on the rising edge of the clock that is enabled by $\overline{WEN1}$ asserted LOW.

Built-In Self-Test

The Built-In Self-Test mode is exercised by asserting the $\overline{BISTEN}$ pin on the HOTLink Receiver. Upon entering BIST, the HOTLink Receiver will wait for the BIST initialization code and then assert $\overline{RDY}$ LOW when the code has been received. $\overline{RDY}$ will pulse HIGH once per received BIST loop. RVS will pulse HIGH if a byte pattern mismatch occurs. $\overline{RDY}$ and RVS can be monitored by the higher-level controller to characterize the integrity of the link.

Controller State Description

A state diagram for a receiver state machine is shown in Figure 5. Five simple signals control the interface. The Receive signal instructs the state machine to store words into the FIFO when $\overline{RDY}$ pulses LOW. Deassertion of Receive ends data reception abruptly. The Reframe signal tells the state machine to synchronize the HOTLink Receiver to the serial data. The Test signal forces the HOTLink Receiver to enter BIST mode and the Program signal causes the state machine to write a word into the FIFO internal program register. The Waiting output signal is asserted when the state machine is in the WAIT state.

Full and Empty signals are decoded for the convenience of the higher-level controller to assist in reading data out of the FIFO. The programmable flags may also be decoded if they have been programmed. It is important that the flags be monitored because a full FIFO will ignore attempted writes. The higher-level controller is responsible for insuring that the FIFO does not become full.

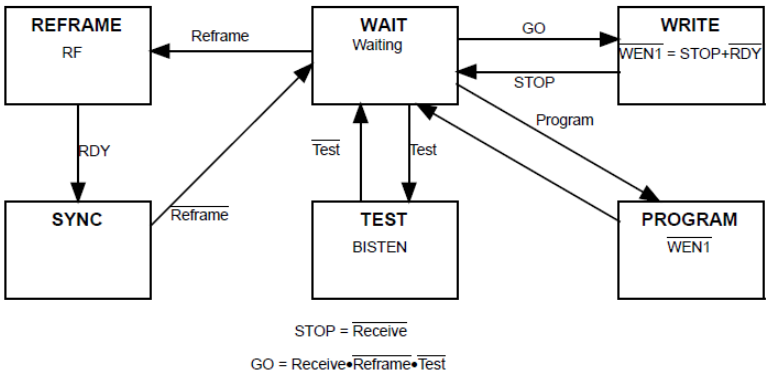
The REFRAME state is entered by the assertion of Reframe from the WAIT state. The REFRAME state is used to synchronize the receiver to the incoming serial data stream. When the state machine asserts RF, the HOTLink Receiver synchronizes its internal bit counter with received K28.5 characters. RDY will pulse LOW when the first synchronized K28.5 character is available. The state machine will return to the WAIT state when the serial data has been resynchronized and $\overline{Reframe}$ is deasserted.

Data reception is initiated by asserting the Receive signal while the state machine is in the WAIT state. The controller will immediately transition to the WRITE state and store data when $\overline{RDY}$ is asserted LOW. The WRITE state continually writes valid characters into the FIFO until Receive is deasserted. Control then returns to the WAIT state and Waiting is asserted.

The BIST state is included for handling the Built-In Self-Test. During BIST, writing to the FIFO is disabled. Assertion of RVS will signal a character reception error. $\overline{RDY}$ will pulse once per BIST loop and should be used to count the number of BIST loops received. The higher-level controller could monitor these signals in order to characterize the link.

The PROGRAM state writes the program word into the FIFO internal program register. This state is entered from the WAIT state at the command of the higher-level controller. Programming should only be performed during a Master Reset cycle ($\overline{RS}$ LOW). In order to meet the FIFO programming timing requirements, it is recommended that at least one clock cycle occur on each side of the program cycle while $\overline{RS}$ is LOW. The higher-level controller is responsible for meeting the specific programming timing requirements discussed in the Resetting and Programming the FIFO section of the CY7C4211 datasheet.

Figure 5. Receiver Controller State Diagram

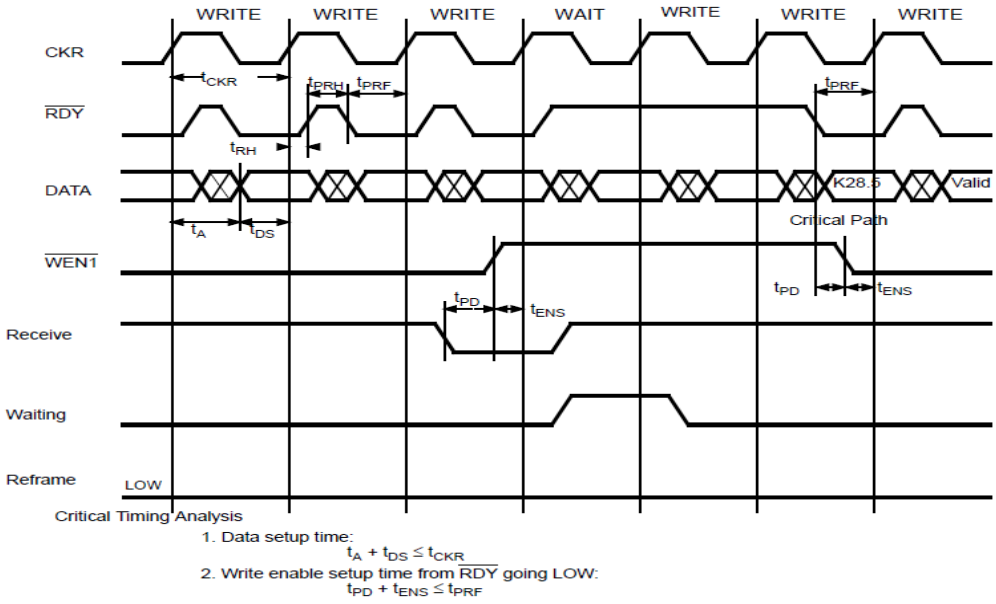


Critical Timing Analysis

Timing analysis for both the pipelined and unpipelined interface is presented in this section. A Timing diagram is provided for the receiver interface that does not include the optional register. Critical timing relationships are provided at the bottom of Figure 6. This diagram highlights the critical timing of the $\overline{RDY}$ pulse. The interface timing with pipeline registers is straight forward and the results are presented below.

The timing analysis assumes that the state machine state bits are stable and valid before any critical signal is available to the state machine and that state bit setup time is not an issue. This assumption allows the state machine timing to be modeled by its combinatorial t_{PD}.

Figure 6. Receiver Timing Diagram



Unregistered Timing

The delayed $\overline{RDY}$ pulse tightens the timing margins on the receiver controller. The state machine combinatorial delay for generating output control signals from valid inputs is modeled as t_{PD}. The FIFO enable setup time is t_{ENS} = 4 ns. Assuming t_{CKR} is 30 ns, the constraint on t_{PD} is Write enable generation time from $\overline{RDY}$ LOW:

$$t_{PD} \leq 1/2\ t_{CKR} - t_{ENS} - 3\ ns = 7\ ns \qquad \text{Equation 3}$$

A 40-ns clock period eases the timing constraint to a more reasonable 10 ns.

The parallel data have no problem meeting the timing constraints imposed by a 30-ns clock period. The HOTLink Receiver access time, t_A, is 9 ns and the FIFO data setup time, t_{DS}, is 4 ns:

Critical data timing:

$$\mathbf{t_A + t_{DS} \leq t_{CKR}} \qquad \text{Equation 4}$$

This assumes no trace delays or clock skew.

Registered Timing

With the optional pipeline register inserted, the timing constraint on the controller is eased. A register access time, t_{AR}, of 10 ns and set-up time, t_{SU}, of 5 ns are assumed. Using a 30-ns clock, the HOTLink Receiver access time is $t_A = t_{CKR}/5 + 3$ ns = 9 ns.

The constraint on the combinatorial delay through the controller is Write enable generation time from $\overline{RDY}$ LOW:

$$\mathbf{t_P \leq t_{CKR} - t_{AR} - t_{ENS} = 16\ ns} \qquad \text{Equation 5}$$

The HOTLink data and $\overline{RDY}$ pulse timing constraints to the pipeline register are:

Data setup time:

$$\mathbf{t_A + t_{SU} \leq t_{CKR}} \qquad \text{Equation 6}$$

RDY setup time:

$$\mathbf{t_{SU} \leq 1/2 t_{CKR} - 3\ ns} \qquad \text{Equation 7}$$

These constraints are easily met.

Summary

The HOTLink transmitter/receiver interfaces to clocked FIFOs can operate at speeds up to 330 Mbits/s with no external logic. Simple state machine controllers can be used to enable the transmission and reception of serial data and enable the HOTLink Built-In Self-Test capability.

Document History

Document Title: AN1130 - Interfacing the CY7B923 and CY7B933 (HOTLink®) to Clocked FIFOs

Document Number: 001-30921

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1494324	SAAC	09/20/2007	New spec
*A	3393273	SAAC	10/05/2011	Updated Transmitter Interface: Updated Data Path and Controller: Updated description. Updated Figure 1. Updated Receiver Interface: Updated description. Updated Figure 4. Updated Data Path and Controller: Updated description. Updated to new template.
*B	4574030	YLIU	11/19/2014	Updated to new template. Completing Sunset Review.
*C	5877314	AESATMP8	09/08/2017	Updated logo and Copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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

© Cypress Semiconductor Corporation, 2007-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.