The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN204902

F²MC-8FX Family MB95310/370 Series 8-Bit Microcontroller Infrared Remote Function API

Associated Part Family: MB95310/370 Series

This document introduces API for infrared remote function.

Contents

1 Introduction....1
2 Background....1
3 Description of Infrared Remote Theory....2
4 MB95F310 Infrared Remote Register....3
5 Infrared Remote Library Function List....4
6 Infrared Remote Function Detail....4
6.1 Initial_Remot Function....4
6.2 Interrupt Function....5
6.3 Remote_detect Function....5
7 Usage Demo....6
7.1 Hardware Design....6
7.2 Steps for Software Design....6
7.3 Steps for Adding Remote Library....7
8 Debug....10
9 Additional Information....12
Document History....13

1 Introduction

This document introduces API for infrared remote function.

Infrared remote function is generally used in TV, audio system and air conditioner. In following chapters we describe theory and library of infrared remote. Chapter three is the theory of infrared remote and chapter four is function library of infrared remote. In function library we should set up three functions to control infrared remote, Initials remote interrupt, remote interrupt function to receive infrared code and key judge function to distinguish which key pressed.

2 Background

This section introduces background of infrared remote.

Infrared remote is widely used in TV, VCD, DVD, air condition and so on, because it is easy to be used, easy to be bought and it is very cheap.

Infrared remote has occurred about twenty five years. It encodes the pressed key and transfers this key value to “0” and “1” and then sends out; because each key has the only code so user can judge the key value according to the code. So user does not need to use the key board to decode the key value, in which the complicated circuit and the I/O port are left out.

3 Description of Infrared Remote Theory

This section describes theory of infrared remote.

Infrared remote has many types which are decided by remote chip. In following description uPD6121G will be described.

Figure 1 describes the configuration of frame which is the remote code.

Figure 1. Remote Code

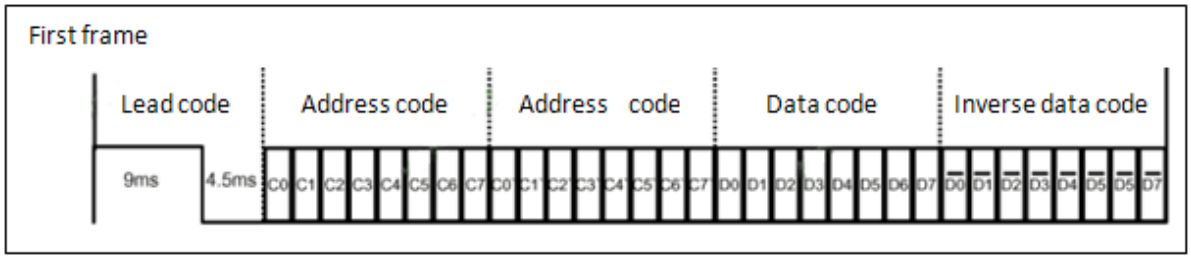
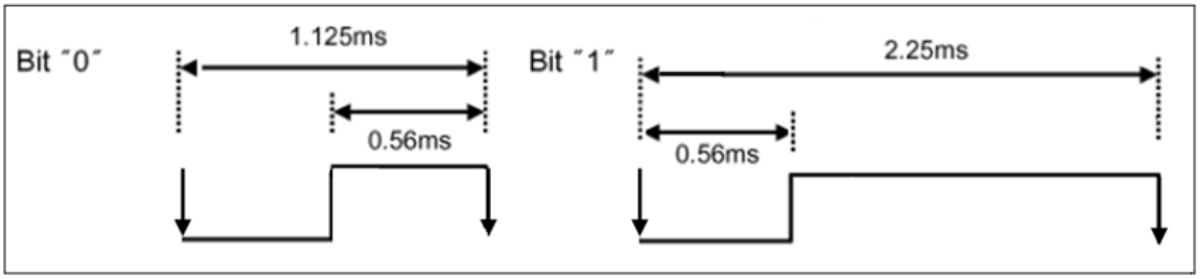


Figure 2 describes the encode “0” and “1”

Figure 2. Encode Wave



For different remote the code and the configuration is different.

Normally the remote is generating high level except code status it will generate low and high level.

4 MB95F310 Infrared Remote Register

This section describes MB95F310 infrared remote register.

In MB95F310 series MCU remote function is realized by 8/16-BIT composite timer input capture module. Following Table 1 describes registers in composite timer input capture.

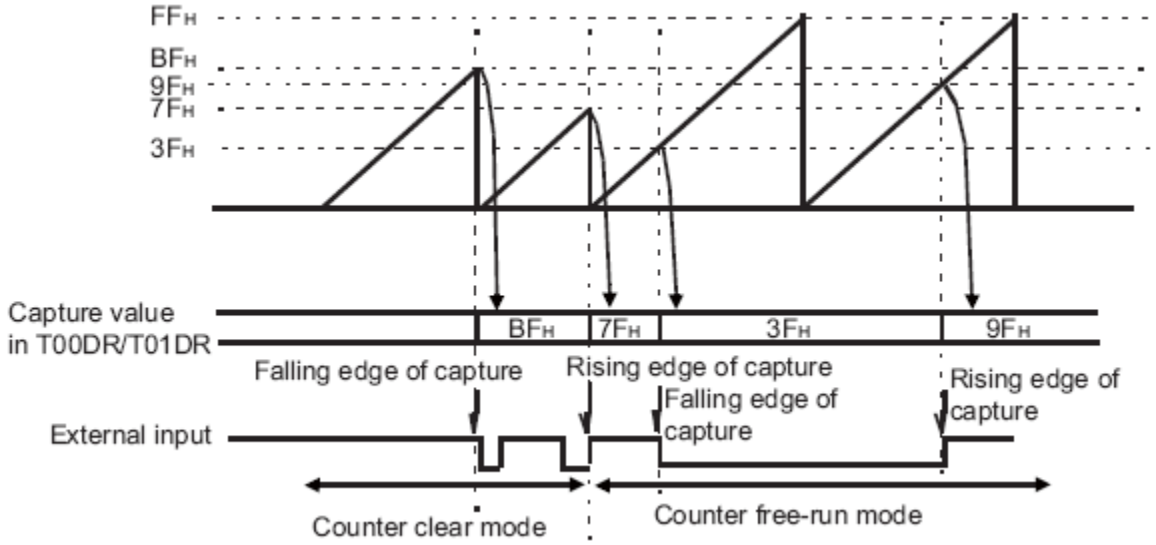
Table 1. All Register List

Register		Description
Timer0	T00CR0	Low 8bits of stats control register0
	T01CR0	High 8bits of stats control register0
Timer0	T00CR1	Low 8bits of stats control register1
	T01CR1	High 8bits of stats control register1
Timer0	T00DR	Low 8bits of data register
	T01DR	High 8bits of data register
Timer0	TMCR0	Timer mode control register
Timer1	T10CR0	Low 8bits of stats control register0
	T11CR0	High 8bits of stats control register0
Timer1	T10CR1	Low 8bits of stats control register1
	T11CR1	High 8bits of stats control register1
Timer1	T10DR	Low 8bits of data register
	T11DR	High 8bits of data register
Timer1	TMCR1	Timer mode control register

For detailed register please refer to MB95F310 Hardware Chapter 18.

Figure 3 describes these registers work condition in detail.

Figure 3. Register Work Condition



5 Infrared Remote Library Function List

This section introduces all functions in infrared remote library in project Simulate remote.prj which MCU are MB95F310.

Table 2 lists the infrared remote functions.

Table 2. Remote Capture Functions

Function name	Description
void initial_Remot(void)	Initializes remote capture register
__interrupt void inputcapture0 (void)	Interrupt to receive remote code
unsigned char Remote_detect(void)	According remote code to judge which key pressed

6 Infrared Remote Function Detail

This section introduces the detail of infrared remote functions.

6.1 Initial_Remot Function

Table 3 describes initial_Remot function.

Table 3. Initial_Remot Function

Function name	Initial_I2C
Function prototype	Void initial_Remot(void)
Behavior description	Initializes remote capture condition
Input parameter	None
Return value	None
Example	The library function sets clock use internal clock, counter interval is 0.64 µs and falling edge trigger counter: initial_Remot();

If user wants to change capture condition, please refer to register T00CR0, T00CR1 and TMCR0.

6.2 Interrupt Function

Table 4 describes __interrupt function.

Table 4. Interrupt Function Description

Function prototype	__interrupt void inputcapture0 (void)
Behavior description	When captured a falling edge, the interrupt is occurred , and the pulse is received and the width of the pulse will be countered
Input parameter	None
Return value	None
Example	__interrupt void inputcapture0 ();

Different remote the configuration is different so the code is different; maybe it is 16-bit code maybe it is 32-bit code, user can review the code by oscilloscope. In this remote.prj the remote code is set to match 16-bit remote code.

6.3 Remote_detect Function

Table 5 describes Remote_detect function.

Table 5. Remote_detect Function

Function name	AD_Read
Function prototype	unsigned char Remote_detect(void)
Behavior description	Judges the code and then decides which key has been pressed according to the code
Input parameter	None
Return value	Key value of the remote
Example	[variable]= Remote_detect();

The key value can be decided by user's remote encode.

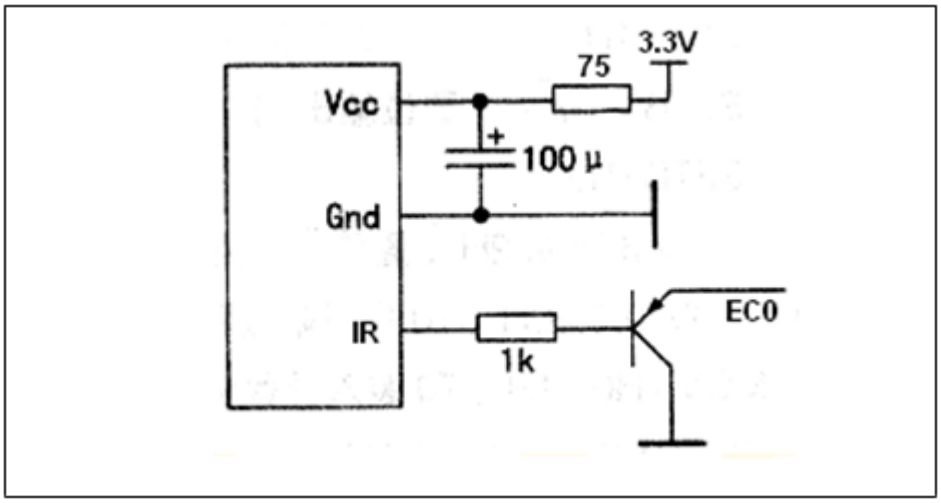
7 Usage Demo

This section describes something we must pay attention to when we use.

7.1 Hardware Design

For hardware design the following figure may be referred.

Figure 4. Remote Hardware Design



7.2 Steps for Software Design

- For infrared remote software design, initialization is the first step, which opens the interrupt, sets the interval time and trigger condition.

  Following setting is referred.

Figure 5. Initialization Design

```
Void initial_Remot(void)
{
    TMCR0 = 0x00;
    T00CR0 = 0x4b;
    T00CR1 = 0xa0;
}
```

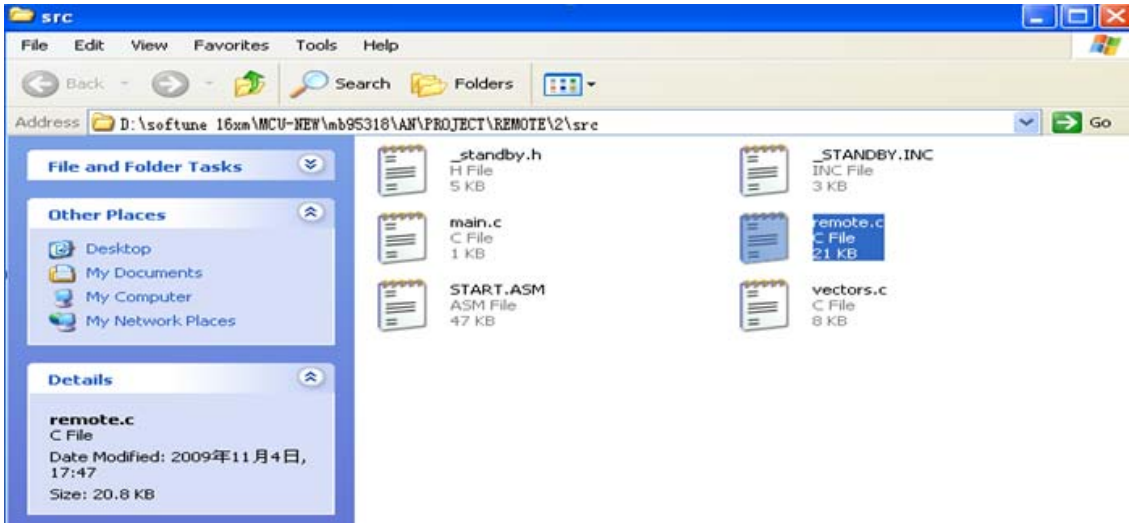
- Detecting remote code is the important step. Each pulse is one bit of code. Each time enter into the interrupt one bit is detected. If the remote code is 16bits, after sixteen enter into interrupt, the key code is generated.
- The last step is key value judge. Different keys have different codes, so user can judge the key value by code which is generated in step two.

7.3 Steps for Adding Remote Library

When use this project please refer to following steps.

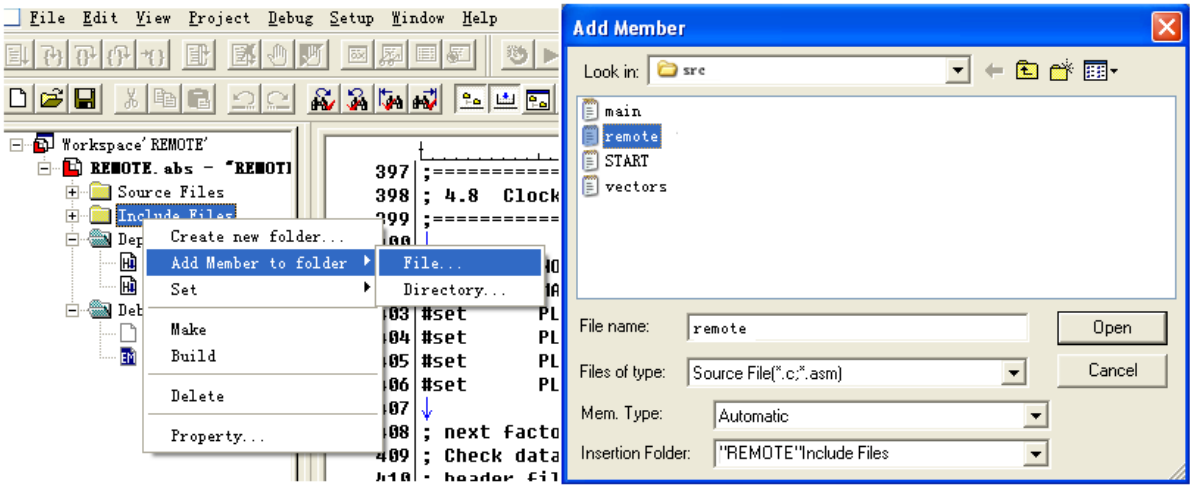
- First step is to add library to document, Figure 6 describes this step

Figure 6. Library Use First Step



- Second step is to add function to project, Figure 7 describes this step

Figure 7. Library Use Second Step



- Third step is to add initial function to main.c, Figure 8 describes this step.

Figure 8. Library Use Third Step

```
void main(void)
{
    unsigned char KeyVA;
    InitIrqLevels();
    __EI();

    Initial_Inter();

    while(1)
    {
        KeyVA = Remote_detect();
    }
}
```

- Fourth step is to add interrupt function to vector.c, Figure 9 describes this step.

Figure 9. Library Use Fourth Step

```
/*-----------------------------------------------------------------------
   Prototypes
   Add your own prototypes here. Each vector definition needs is proto-
   type. Either do it here or include a header file containing them.
-----------------------------------------------------------------------*/

__interrupt void DefaultIRQHandler (void);
__interrupt void inputcapture0 (void);
/*-----------------------------------------------------------------------
```

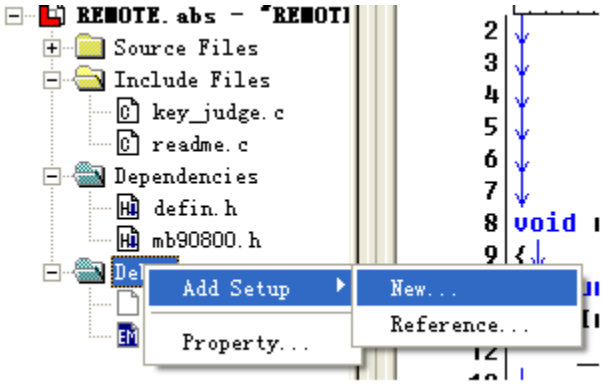
- Fifth step is to add remote_detect function to main.c, Figure 10 describes this step.

Figure 10. Library Use Fifth Step

```
 8 void main(void)
 9 {
10     unsigned char KeyVA;
11     InitIrqLevels();
12     __EI();
13
14     Initial_Inter();
15
16     while(1)
17     {
18         KeyVA = Remote_detect();
19     }
20 }
```

- The sixth step is debugging, set a environment before debugging, Figure 11 describes this step

Figure 11. Library Use Sixth Step



For more condition please refer to 8. Debug.

8 Debug

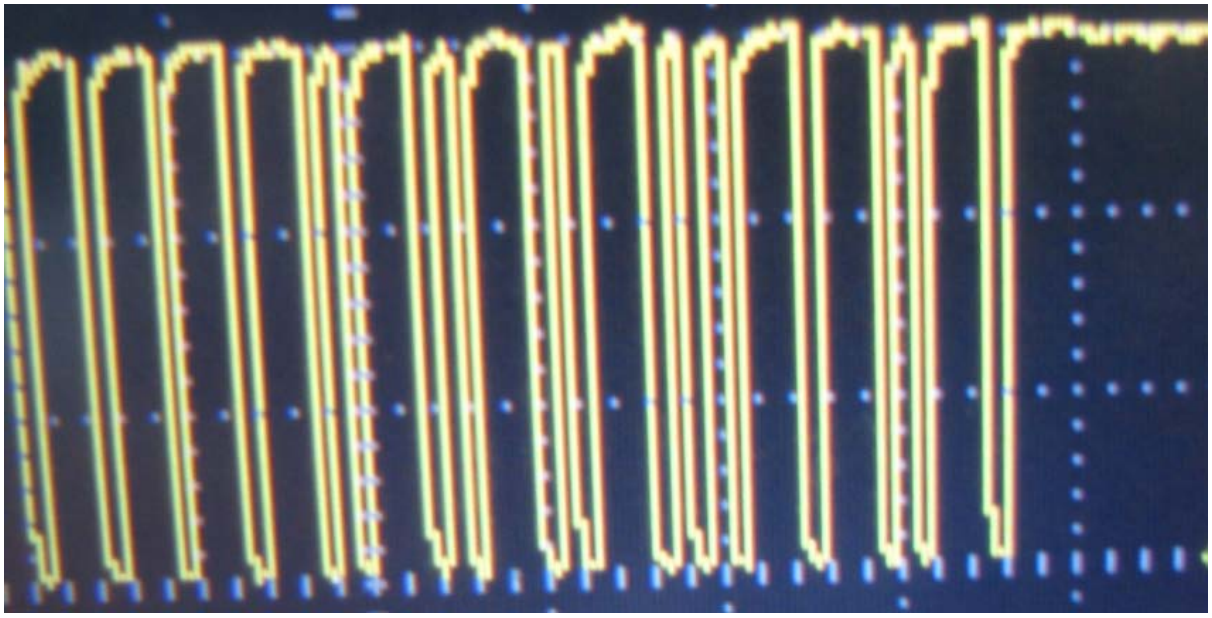
This section describes how to debug the sample code on EV-board and what will happen when the code is running.

There is a simple project remote.prj for debugging. This project is based on our EV-board MB2146-450-E and the target MCU is MB95F310.

When debugging, the hardware linking please refer to Figure 4.

Press one key and the wave will be detected by oscilloscope as Figure 12.

Figure 12. Key Code

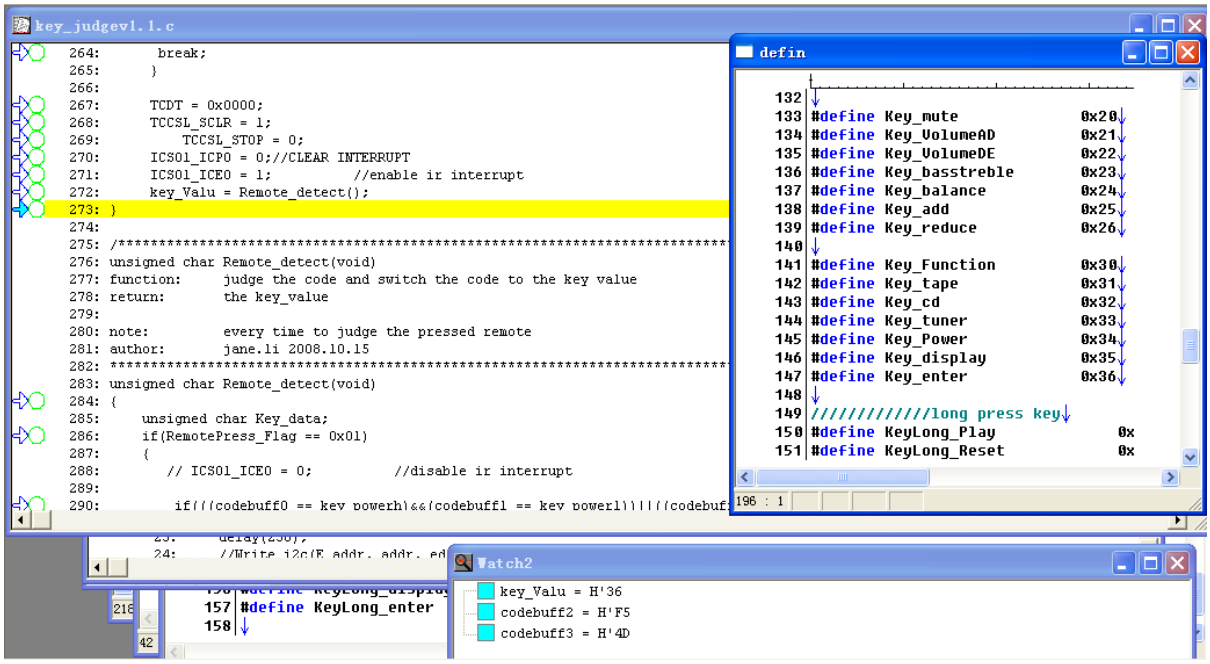


The wide wave is binary “1”, and the short wave is binary “0”.

From this picture we can judge the remote code is “1111 0101 0100 1101” in binary

Run project, the key value will be read by Remote_detect function. And key value will be read out by global variable “key_Valu”. In this sample the **ENTER** key is pressed. Please refer to Figure 13 for detailed result.

Figure 13. Debugging Description



9 Additional Information

For more information about how to use MB95310 EV-board, BGM Adaptor and SOFTUNE, please refer to EV-Board MB2146-450-E User Manual, or visit website:

http://www.cypress.com/documentation/application-notes/mb95310370-mb2146-450-e-lcd-evb-user-manual

Document History

Document Title: AN204902 – F²MC-8FX Family MB95310/370 Series 8-Bit Microcontroller Infrared Remote Function API

Document Number: 002-04902

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	HUAL	11/06/2009	Original version
			03/17/2011	Modified Chinese remark
*A	5245363	HUAL	05/18/2016	Migrated Spansion Application Note MCU-AN- 500066-E-11 to Cypress format.

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
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

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

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corporation. 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
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2009-2016. 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, 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.