THIS SPEC IS OBSOLETE

Spec No: 002-05326

Spec Title: AN205326 - F2MC-8FX FAMILY MB95200H/210H SERIES WILD REGISTER FUNCTION

Replaced by: None

AN205326

F²MC-8FX Family MB95200H/210H Series Wild Register Function

This application note describes how to use the wild register function and illustrates how to use it with the sample code.

OBSOLETE

Contents

1 Introduction....1
2 Wild Register Function....1
2.1 Wild Register Function....1
2.2 Block Diagram of Wild Register Function....1
2.3 Registers of Wild Register Function....3
3 Software Configuration of Wild Register....6
3.1 Setting Procedure for Wild Register Function....6
3.2 Software Setting Flow....6
3.3 Example Code....7
3.4 Software Debug Environment....8
4 Typical Hardware Connection Example....9
4.1 Typical Hardware Connection....9
4.2 Operation Flow....9
4.3 Firmware Consist....9
5 Notes on using Wild Register....9
5.1 Wild Register Applicable Addresses....9
5.2 Notes on Software....9
6 Additional Information....10
A Appendix....11
A.1 Sample Code....11
Document History....13

1 Introduction

This application note describes how to use the wild register function.

This note describes the wild register function and illustrates how to use it with the sample code.

2 Wild Register Function

This chapter describes the wild register function

2.1 Wild Register Function

The wild register consists of 3 wild register data setting registers, 3 wild register address setting registers, a 1-byte address compare enable register and a 1-byte data test setting register. If addresses and data that are to be modified are set to these registers, the ROM data can be replaced with modification data set in the registers. Data of up to three different addresses can be modified.

The wild register function can be used to debug a program after creating the mask and to patch bugs in the program.

2.2 Block Diagram of Wild Register Function

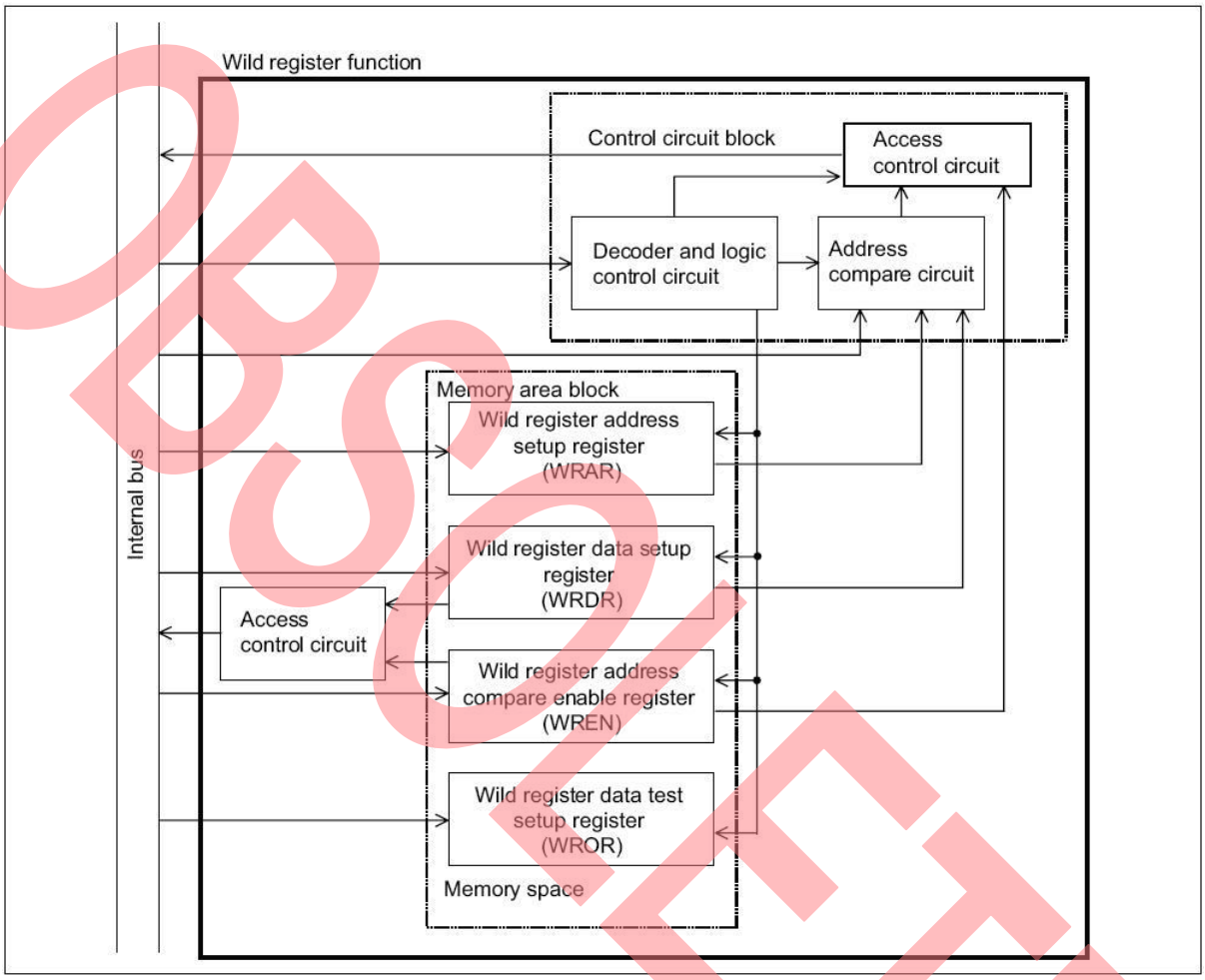
The block diagram of the wild register is shown below. The wild register consists of the following blocks:

- Memory area block
  Wild register data setting register (WRDR0 to WRDR2)
  Wild register address setting register (WRAR0 to WRAR2)
  Wild register address compare enable register (WREN)
  Wild register data test setting register (WROR)

- Control circuit block

Figure 1 shows the block diagram of the wild register function.

Figure 1. Block Diagram of Wild Register Function



2.2.1 Memory Area Block

The memory area block consists of the wild register data setting registers (WRDR), wild register address setting registers (WRAR), wild register address compare enable register (WREN) and wild register data test setting register (WROR). The wild register function is used to specify the addresses and data that need to be replaced. The wild register address compare enable register (WREN) enables the wild register function for each wild register data setting register (WRDR). In addition, the wild register data test setting register (WROR) enables the normal read function for each wild register data setting register (WRDR).

2.2.2 Control Circuit Block

This circuit compares the actual address data with addresses set in the wild register address setting registers (WRAR). If they match, the circuit outputs data from the wild register data setting register (WRDR) to the data bus. The operation of the control circuit block is controlled by the wild register address compare enable register (WREN).

OBSOLETE

2.3 Registers of Wild Register Function

The registers of the wild register function include the wild register data setting registers (WRDR), wild register address setting registers (WRAR), wild register address compare enable register (WREN) and wild register data test setting register (WROR).

Register List of Wild Register Function

Figure 2. Register List of Wild Register Function

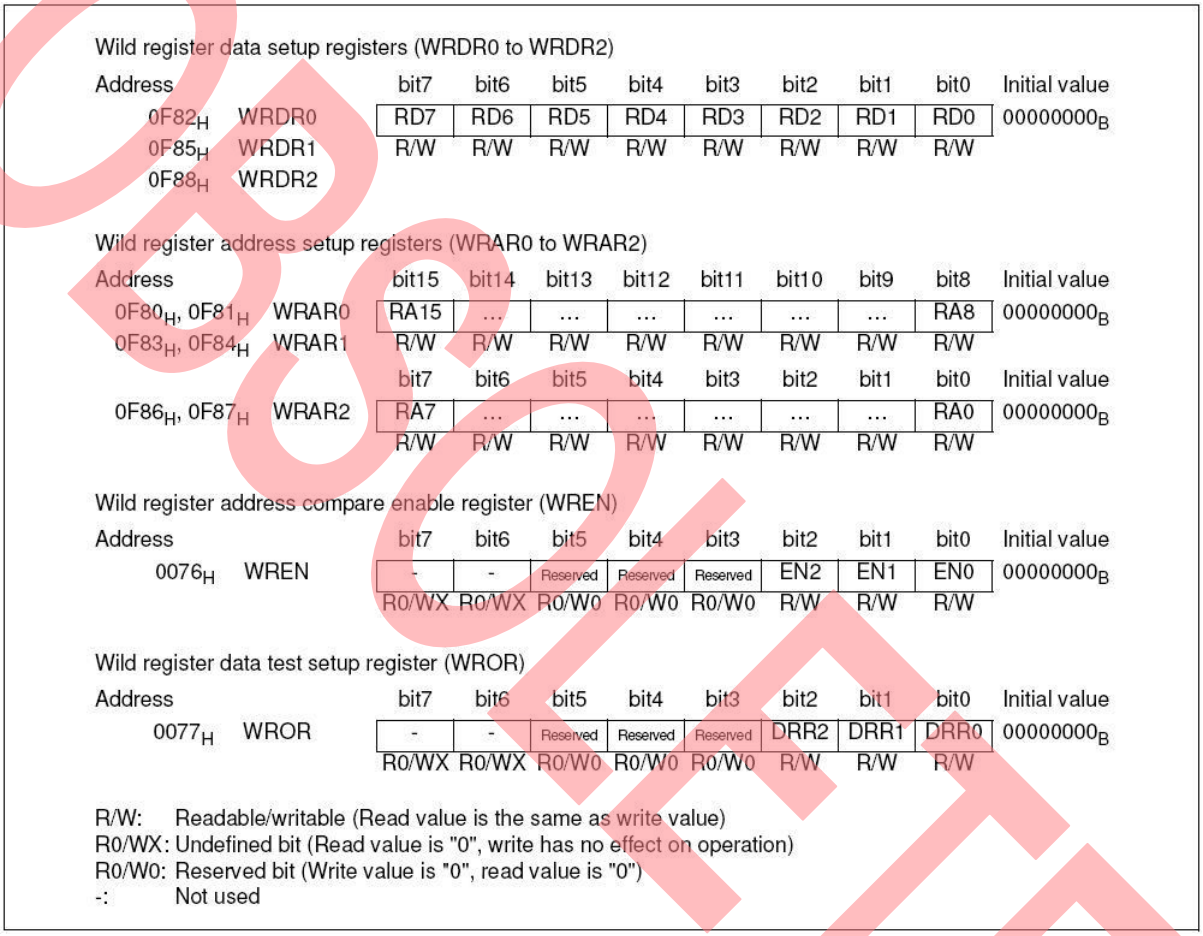


Figure 2 shows all registers of the wild register function. For detailed information, please check Chapter 13 of the MB95200 series Hardware Manual.

A wild register number is assigned to each wild register address setting register (WRAR) and wild register data setting register (WRDR).

Table 1. Numbers Corresponding to Wild Register

Wild register number	Wild register address setup register (WRAR)	Wild register data setup register (WRDR)
0	WRAR0	WRDR0
1	WRAR1	WRDR1
2	WRAR2	WRDR2

OBSOLETE

2.3.1 Wild Register Data Setting Registers (WRDR0 to WRDR2)

The wild register data setting registers (WRDR0 to WRDR2) use the wild register function to specify the data to be amended.

Table 2. Functions of Bits in WRDR

Name		Function
WRDR0	bit7 to bit0	These bits specify the data to be amended by the wild register function. These bits are used to set the amendment at the address assigned by the wild register address setting register (WRAR). Data is valid at an address corresponding to one of the wild register numbers. The read access to one of these bits is enabled only when the data test setting bit in the wild register data test setting register (WROR) corresponding to the bit to be read is set to “1”.
WRDR1		
WRDR2		

2.3.2 Wild Register Address Setting Registers (WRAR0 to WRAR2)

The wild register address setting registers (WRAR0 to WRAR2) set the address to be amended by the wild register function.

Table 3. Functions of Bits in WRAR

Name		Function
WRAR0	bit15 to bit0	These bits set the address to be amended by the wild register function. These bits are used to set the address to be assigned to amendment data. The address is specified according to the wild register number corresponding to a wild register address setting register.
WRAR1		
WRAR2		

OBSOLETE

2.3.3 Wild Register Address Compare Enable Register (WREN)

The wild register address compare enable register (WREN) enables/disables the operations of the wild register function using their respective wild register numbers.

Table 4. Function Description of WREN

Bit name		Function
bit7, bit6	Undefined bits	These bits are undefined. When these bits are read, they always return "0". Writing values to these bits has no effect on the operation.
bit5 to bit3	Reserved bits	These bits are reserved. When these bits are read, they always return "0". Always set these bits to "0".
bit2 to bit0	EN2, EN1, EN0: Wild register address compare enable bits	These bits enable/disable the operation of the wild register. EN0 corresponds to wild register number 0. EN1 corresponds to wild register number 1. EN2 corresponds to wild register number 2. Writing "0": disables the operation of the wild register function. Writing "1": enables the operation of the wild register function.

2.3.4 Wild Register Data Test Setting Register (WROR)

The wild register data test setting register (WROR) enables/disables reading from the corresponding wild register data setting register (WRDR0 to WRDR2).

Table 5. Function Description of WROR

Bit name		Function
bit7, bit6	Undefined bits	These bits are undefined. When these bits are read, they always return "0". Writing values to these bits has no effect on the operation.
bit5 to bit3	Reserved bits	These bits are reserved. When these bits are read, they always return "0". Always set these bits to "0".
bit2 to bit0	DRR2, DRR1,DRR0: Wild register data test setting bits	These bits enable/disable the normal reading from the corresponding data setting register of the wild register. DRR0 enables/disables reading from the wild register data setting register (WRDR0). DRR1 enables/disables reading from the wild register data setting register (WRDR1). DRR2 enables/disables reading from the wild register data setting register (WRDR2). Writing "0": disables reading. Writing "1": enables reading.

OBSOLETE

3 Software Configuration of Wild Register

This chapter describes the software setting procedure for the wild register function

3.1 Setting Procedure for Wild Register Function

Prepare a special program that can read the value to be set in the wild register from external memory (e.g. EEPROM or FRAM) in the user program before using the wild register function. The setting method for the wild register is shown below.

This section does not include information on the method of communications between the external memory and the device.

Write the address of the built-in ROM code that will be modified to the wild register address setting register (WRAR0 to WRAR2).

Write a new code to the wild register data setting register (WRDR0 to WRDR2) corresponding to the wild register address setting register to which the address has been written.

Write “1” to the EN bit in the wild register address compare enable register (WREN) corresponding to the wild register number to enable the wild register function represented by that wild register number.

Table 6 shows the register setting procedure for the wild register function.

Table 6. Register Setting Procedure for Wild Register

Step	Operation	Operation example
1	Read replacement data from a peripheral function outside through a certain communication method.	Suppose the built-in ROM code to be modified is at the address F011H and the data to be modified at "B5H" and There are three built-in ROM codes to be modified
2	Write the replacement address to a wild register address setting register (WRAR0 to WRAR2).	Set wild register address setup registers (WRAR0 = F011H, WRAR1 = ..., WRAR2 = ...).
3	Write a new ROM code (replacement for the built-in ROM code) to the wild register data setup register (WRDR0 to WRDR2).	Set the wild register data setup registers (WRDR0 = B5H, WRDR1 =..., WRDR2 =...).
4	Enable the EN bit in the wild register address compare enable register (WREN) corresponding to the wild register number of the wild register function used.	Setting bit 0 of the address compare enable register (WREN) to "1" enables the wild register function of the wild register number 0. If the address matches the value set in the wild register address setting register (WRAR), the value of the wild register data setting register (WRDR) will replace the built-in ROM code. When replacing more than one built-in ROM code, enable the related EN bits in the wild register address compare enable register (WREN) corresponding to respective built-in ROM codes.

3.2 Software Setting Flow

To operate the wild register function, the following settings are required.

Choosing a wild register number(0, 1, 2)

Writing an address to WRARx

Writing a new code into WRDRx

Setting WREN to enable the wild register function

The wild register function will modify code specified if the address matched the setting address

PROFIT

3.3 Example Code

The example codes describe how to use the wild register to replace the ROM data at a specified address.

```
/*****************************************************************************
/*the example code set modified address and modified data
/*WRAR2 = c081H, WRDR2 = 0x66, choose wild register number 2
*****************************************************************************/

#include "mb95200.h"

/*****************************************************************************
NAME  :  vSysInit ()
FUNCTION: Initialize wild register. Choose WRAR2 work
*****************************************************************************/
void vSysInit (void)
{
    WRAR2 = 0xC081;             //ROM ADDL address = 0xC081
      WRDR2 = 0x66;                 //be modified data = 0x66
      WREN_EN2 = 1;                 //enable wild 2
      WROR_DRR2 = 1;                //enable reading
}

/*****************************************************************************
NAME  :  main ()
FUNCTION: Main loop, modified PDR0 = 0x66
*****************************************************************************/
void main (void)
{
      vSysInit ();                  //initialize function

      #pragma asm
            jmp   0xC080                  //jump to 0xC080
      #pragma endasm

      while (1)                     //main loop
      {
            #pragma asm
                  .SECTION    Test, CODE, Locate = 0xC080
                  MOV   A, #44H
                  MOV 0x00, A
//setting PDR0=0x44, but it will be modified 0x66
            #pragma endasm

            asm ("\tNOP");
      }
}
```

OBSOLETE

Note: The full example code is referred to the project. The project name is Wild_Register.

3.4 Software Debug Environment

Figure 3 shows how to set a specified address and amendment data by the wild register.

The software set and debugs flow.

Initialize wild register: WRAR2 = 0xC081 (modified address), WRDR2 = 0x66 (modified data).

In main loop, address 0xC081 is set to Prot0 = 0x44.

Enter running, the wild register set Prot0 = 0x66 when the address is 0xC081.

Figure 3. Environment of Debug

```
       23:   WRARH2 = 0xC0;          //ROM ADDH
C0FA: 04C0            MOV     A,#C0
C0FC: 610F86          MOV     >\__wrarh2,A
       24:   WRARL2 = 0x81;          //ROM ADDL address = C081H
C0FF: 0481            MOV     A,#81
C101: 610F87          MOV     >\__wrarl2,A
       25:   WRDR2 =  0x66;          //modified data = 66
C104: 0466            MOV     A,#66
C106: 610F88          MOV     >\__wrdr2,A
       26:   WREN_EN2 = 1;           //enable wild 2
C109: AA76            SETB    \__wren:02
       27:   WROR_DRR2 = 1;          //enable reading
C10B: AA77            SETB    \__wror:02
       28:
       29: }
C10D: 20              RET
       30:
       31: /*---------------------------------------------------------------------------
       32:    Name: main ();
       33:    Function: main loop, modified PDR0 = 0x66;
       34: ---------------------------------------------------------------------------*/
       35:
       36: void main(void)
       37: {
       38:
       39:         vSysInit();                   //initial funciton
C10E: 31C0F4          CALL    \vSysInit
       40:
       41:         #pragma asm
       42:                 jmp     0xC080        //jump to 0xC080
C111: 21C080          JMP     C080
       43:         #pragma endasm
       44:
       45:
       46:         while(1)                      //main loop
       47:         {
       48:                 #pragma asm
       49:                 .SECTION        Test, CODE, Locate = 0xC080
       50:                 MOV     A,#44H
C080: 0444            MOV     A,#44
       51:                 MOV     __pdr0,A //setting port0 = 0x44,but modified 0x66
C082: 4500            MOV     \__pdr0,A
       52:                 #pragma endasm
       53:
       54:                 asm("\tNOP");         //delay timer
C084: 00              NOP
       55:
       56:         }
```

OBSOLETE

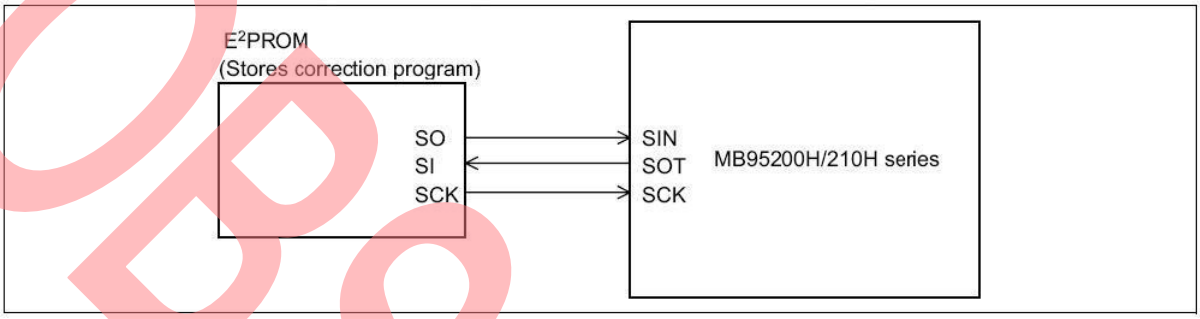
4 Typical Hardware Connection Example

This chapter describes the hardware interface of the wild register

4.1 Typical Hardware Connection

Shown below is a typical hardware connection example applied when using the wild register function.

Figure 4. Typical Hardware Connection Example



Note: It is recommended using this circuit when the MB95200 is connected to EEPROM.

4.2 Operation Flow

The circuit operate flow when MCU connect to EEPROM.

MCU read EEPROM data
Copy EEPROM data to wild register
Enable wild register
Modified data

4.3 Firmware Consist

Need to consist below firmware.

EEPROM : Modified address for the wild register
Modified data for the wild register
MB95200 : Communication interface with EEPROM
Wild register module initialization function

5 Notes on using Wild Register

The chapter describes the precaution for the wild register

5.1 Wild Register Applicable Addresses

The wild register is applicable to all addresses in the address space except "0078H". As address "0078H" is used as a mirror address for the register bank pointer and direct bank pointer, this address cannot be patched.

5.2 Notes on Software

Before the wild register function is used to debug a program that has been created the mask and to patch bugs in the program, the external memory communication interface in MB95200 original Firmware should be secured, so that the MB95200 can read the specified address and data to be modified from external memory (such as EEPROM or FRAM) and then set the wild register.

OBSOLETE

6 Additional Information

For more information on Cypress microcontrollers, visit the following website:

www.cypress.com/documentation/application-notes/mb95200-wild-register

OBSOLETE

A Appendix

A.1 Sample Code

A.1.1 Project Name: Wild_Register

NAME : Wild_Register

FUNCTION: the example code set the built-in ROM code to be modified in the address 0xC081 and the data to be modified to "0x66".

main.c

```
#include "mb95200.h"

/*****************************************************************************
NAME    :       vSysInit ()
FUNCTION:       Initialize wild register. Choose WRAR2 work
*****************************************************************************/
void vSysInit (void)
{
        DDR0 = 0xff;                    //P0 output
        PDR0 = 0xff;                    //port0 setting 0

        WRAR2 = 0xC081;                 //ROM ADDL address = 0xC081
        WRDR2 = 0x66;                   //modified data = 0x66
        WREN_EN2 = 1;                   //enable wild register 2
        WROR_DRR2 = 1;                  //enable reading
}

/*****************************************************************************
NAME    :       main ()
FUNCTION:       Main loop, modified PDR0 = 0x66
*****************************************************************************/
void main (void)
{
        vSysInit ();                            //initialize function

        #pragma asm
                jmp     0xC080                  //jump to 0xC080
        #pragma endasm
```

OBSOLETE

```
        while (1)                                   //main loop
        {
                #pragma asm
                        .SECTION        Test, CODE, Locate = 0xC080
                        MOV     A, #44H
                        MOV     0x00, A
                                //setting port0 = 0x44, but will be modified to 0x66 by wild register
                #pragma endasm

                asm ("\tNOP");                      //delay timer
        }
}
```

OBSOLETE

Document History

Document Title: AN205326 - F²MC-8FX Family MB95200H/210H Series Wild Register Function

Document Number: 002-05326

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	—	CBZH	03/20/2008	Initial release.
			07/22/2008	Modified the figure name.
*A	5260126	CBZH	05/09/2016	Migrated Spansion Application note from MCU-AN-500008-E-11 to Cypress format.
*B	5731032	CBZH	05/12/2017	Obsolete the document.

OBSOLETE

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, 2008-2017. 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, 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.

OBSOLETE