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.

AN205336

F²MC-8FX Family, MB95200H/210H Series Watchdog Timer

This application note describes how to use the watch-dog timer, the functions of the watchdog timer and gives some examples.

Contents

1 Introduction....1
2 Watchdog Timer....1
2.1 Key Features....1
2.2 Block Diagram....2
2.3 Registers....3
2.4 NVR (Non-Volatile Register) Function....4
3 Interval Time....5
4 Usage and Examples....6
4.1 Functions and Operations of Watchdog Timer 6
4.2 Software Watchdog Timer....7
4.3 Hardware Watchdog Timer....8
5 Note on Using Watchdog Timer....9
6 Additional Information....9
6.1 Sample Code....10
7 Document History....14

1 Introduction

This application note describes how to use the watch-dog timer.

The application note describes the functions of the watchdog timer and gives some examples.

2 Watchdog Timer

This chapter introduces the basic function of the watchdog timer.

2.1 Key Features

The watchdog timer functions as a counter used to prevent program from running out of control. Once the watchdog timer is activated, its counter needs to be cleared at specified intervals regularly. A watchdog reset is generated if the timer is not cleared within a preset interval time based on Chapter 3. The watchdog timer that is not cleared may be due to problems in the program which has entered into an infinite loop or Stack over-run or other possible problems that caused the CPU to hang.

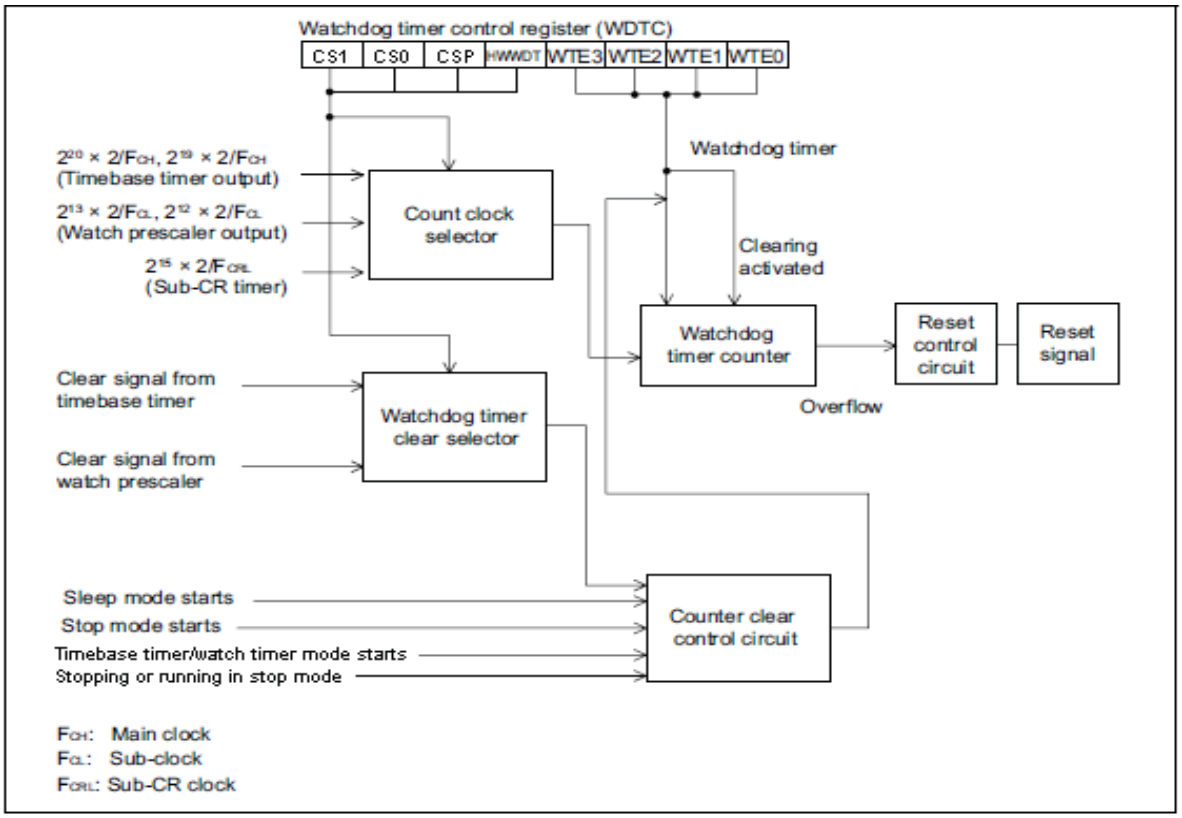
The watchdog timer has the following features:

- Count clock selector
- Watchdog timer counter
- Reset control circuit is used to generate the reset signal when the WDT counter overflows
- Watchdog timer clear selector is used to selects the watchdog timer clear signal
- Counter timer control circuit
- Hardware and software watchdog timer

2.2 Block Diagram

Figure 1 shows the internal block diagram of watchdog timer.

Figure 1. Block Diagram of Watchdog Timer



2.3 Registers

Please refer to Chapter 11 and Chapter 22 of MB95200H/210H Series Hardware Manual for detailed register setting.

2.3.1 Watchdog Timer Control Register (WDTC)

This register is used to activate or clear the watchdog timer.

Figure 2. WDTC

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
$000C_H$	CS1	CS0	CSP	HWWDT	WTE3	WTE2	WTE1	WTE0	
software	R/W	R/W	R/W	R0,WX	R0/W	R0/W	R0/W	R0/W	00000000_B
hardware	R0/WX	R0/WX	R1/WX	R1,WX	R0/W	R0/W	R0/W	R0/W	00110000_B

2.3.2 Watchdog Timer Selection ID Register (WDTH, WDTL)

The two registers are used to select hardware or software watchdog timer.

Figure 3. Watchdog Timer Selection ID Register (WDTH, WDTL)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
$0FEB_H$ WDTH	WDTH7	WDTH6	WDTH5	WDTH4	WDTH 3	WDTH 2	WDTH 1	WDTH 0	xxxxxxxx
$0FEC_H$ WDTL	WDTL7	WDTL6	WDTL5	WDTL4	WDTL 3	WDTL 2	WDTL 1	WDTL 0	xxxxxxxx
	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	

R/W : Readable/writeable (Read value is the same as write value)

R/WX : Read only (Readable, writing has no effect on operation)

R0/WX : Undefined bit (Read value is “0”, writing has no effect on operation)

R1/WX : Undefined bit (Read value is “1”, writing has no effect on operation)

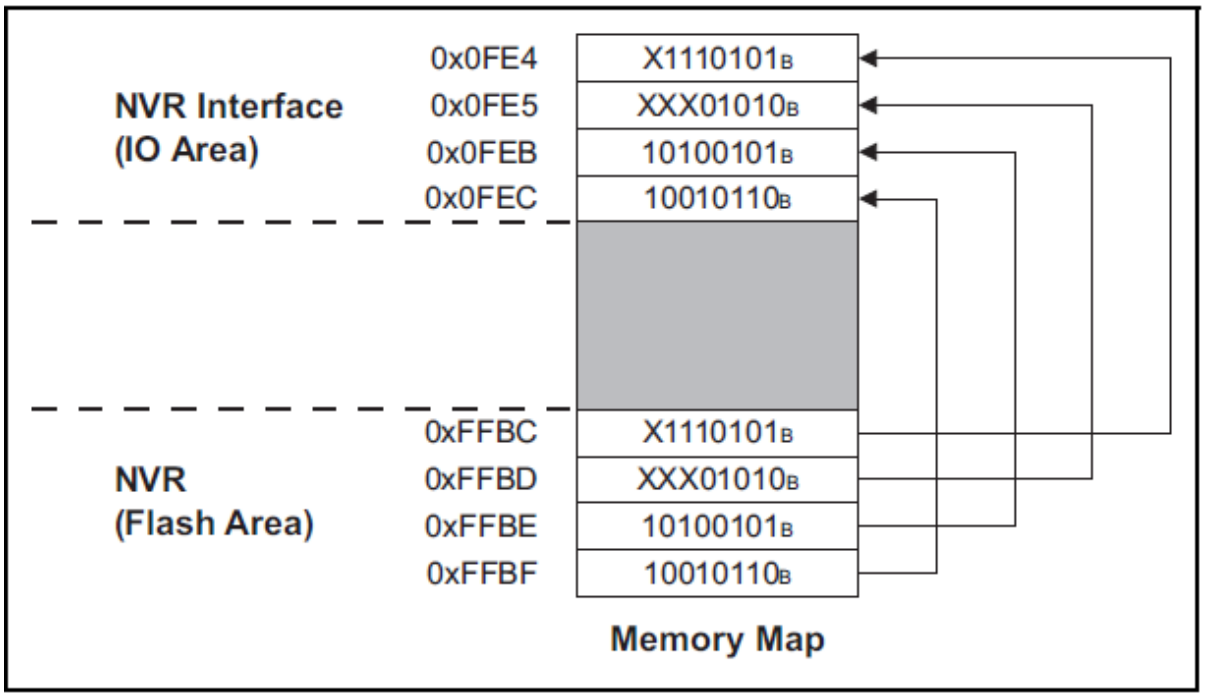
R0/W : Write only (Writable, “0” is read)

2.4 NVR (Non-Volatile Register) Function

The NVR interface enables users to select hardware or software watchdog timer by modifying the 16-bit watchdog timer selection ID. Please note that the watchdog timer selection ID cannot be modified while the CPU is running, therefore please modify NVR in Flash Area first, then the MCU will copy these values to NVR interface (IO Area) automatically after a reset.

Figure 4 shows the basic configuration of serial programming connection for flash memory products.

Figure 4. Retrieval of NVR during Reset



Please note that either the SWWDT or HWWDT can work at the same time. NVR is used to decide which one is working.

These 16 bits of WDTH and WDTL are loaded from the flash address $FFBE_H$, $FFBF_H$ after a reset. The initial values are determined by the pre-loaded values in the NVR flash area.

Write certain values to the address $FFBE_H$ and $FFBF_H$ to select watchdog timer's mode.

Table 1 shows watchdog timer selection ID.

Table 1. Watchdog Timer Selection ID

WDTH[7:0], WDTL[7:0]	Function
$A596_H$	The HWWDT is disabled and the SWWDT is enabled.
$A597_H$	The HWWDT is selected and the SWWDT is disabled. [It can be stopped in one of the standby modes (stop/sleep/timebase timer/watch mode).]
Other than the above values	The WHWDT is selected and the SWWDT is disabled. [It keeps running in one of the standby modes (stop/sleep/timebase timer/watch mode).]

3 Interval Time

This chapter describes the interval time of the watchdog timer

The interval times of the watchdog timer are shown in Table 2. If the counter of watchdog timer is not cleared, a watchdog reset is generated between the minimum time and the maximum time.

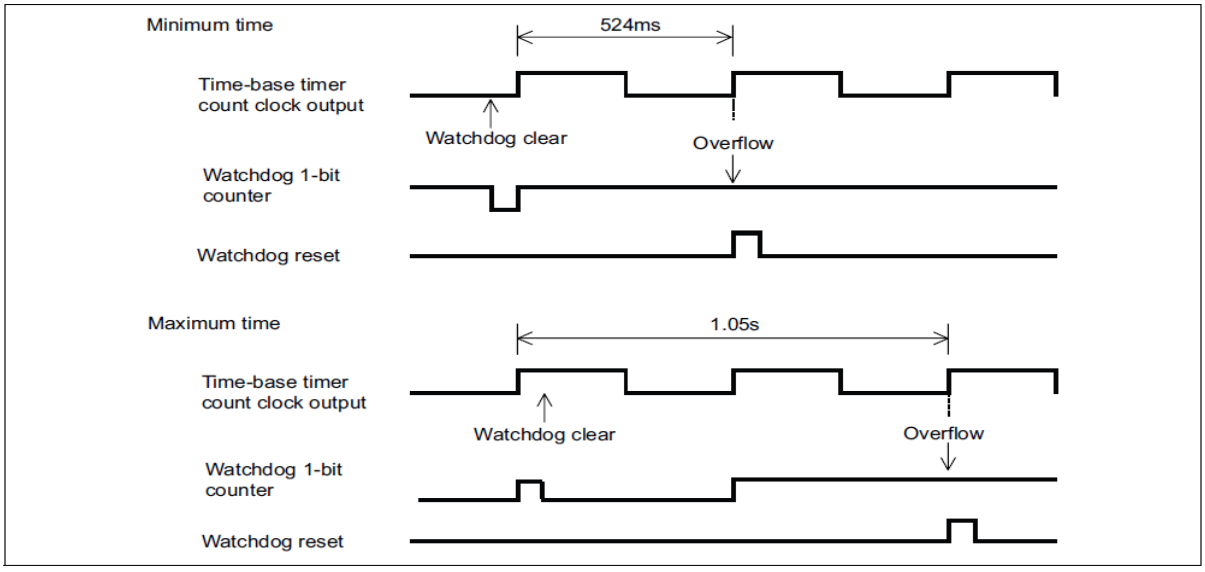
Table 2. Interval Times of Watchdog Timer

Count clock type	Count clock switch bit CS[1:0], CSP	Interval time		
		Minimum time		Maximum time
Timebase timer output (main clock = 4MHz)	000_B(SWWDT)	$2^{21}/F_{CH}$	524 ms	1.05 s
	010_B(SWWDT)	$2^{20}/F_{CH}$	262 ms	524 ms
Watch prescaler output (sub-clock = 32.768KHz)	100_B(SWWDT)	$2^{14}/F_{CL}$	500 ms	1.00 s
	110_B(SWWDT)	$2^{13}/F_{CL}$	250 ms	500 ms
Sub-CR timer (sub-CR clock = 50-200KHz)	$XX1_B$(SWWDT) or HWWDT*1	$2^{16}/F_{CRL}$	328 ms	2.62 s

*1: CS [1:0] = 00_B, CSP = 1_B (read only)

The interval time varies depending on the timing of clearing the watchdog timer. Figure 5 shows the correlation between the timing of clearing the watchdog timer and the interval time when the timebase timer output $2^{21}/F_{CH}$ (F_{CH}: main clock) is selected as the count clock (main clock = 4MHz).

Figure 5. Clearing Timing and Interval Time of Watchdog Timer



Please note that program must clear the counter of the watchdog timer within the minimum time.

4 Usage and Examples

Functions and examples for watchdog timer

This chapter describes the usage of watchdog timer and gives some examples.

4.1 Functions and Operations of Watchdog Timer

Please write certain values to NVR flash area address $FFBE_H$ and $FFBF_H$ to choose the watchdog timer operation mode. See Table 1 for detailed settings of mode choose.

The Watchdog Timer has two operate modes as below:

- Software watchdog

1. Write "$A596_H$" (enable software watchdog timer) to the address $FFBE_H$ and $FFBF_H$ on the flash memory, which are copied to the watchdog timer selection ID register WDTH/WDTL ($0FEB_H$/$0FEC_H$) after a reset.
2. The watchdog timer is activated when "0101B" is written to the watchdog control bits of the watchdog timer control register (WDTC: WTE3 to WTE0) for the first time after a reset. The count clock switch bits of the watchdog timer register (WDTC: CS1, CS0, CSP) should also be set at the same time.
3. Once the watchdog timer is activated, a reset is the only way to stop its operation. Please clear watchdog counter in a specified intervals regularly depend on the setting of WDTC: CS1, CS0, CSP. Refer to Table 3-1 for detailed interval time.

- Hardware watchdog

1. Write "$A597_H$" (WDT can be stopped in a standby mode) or any other value (WDT is enabled in every mode) except than "$A596_H$" or "$A597_H$" to the address $FFBE_H$ and $FFBF_H$ on the flash memory, which are copied to the watchdog timer selection ID register WDTH/WDTL ($0FEB_H$ /$0FEC_H$) after a reset.
2. The hardware watchdog timer starts automatically after a reset and cannot be stopped. Please clear its counter at specified intervals regularly to avoid generates a watchdog timer reset.
3. This timer is cleared by reset and resumes operation after the reset.

Please note that the watchdog timer is cleared at the same time as the timer selected as the count clock (timebase timer or watch prescaler) is cleared. For this reason, the watchdog timer cannot function as such, if the software is set to clear the selected timer repeatedly during the interval time of the watchdog timer

4.2 Software Watchdog Timer

Setting Procedure Example

The software watchdog timer is set up through the following procedure:

1. Ensure that the address $FFBE_H$, $FFBF_H$ values are "$A596_H$" (NVR flash area)
2. Select the count clock.(WDTC:CS1,CS0,CSP)
3. Activate the watchdog timer. (WDTC:WTE3 to WTE0 = 0101_B)
4. Clear the watchdog timer within the minimum interval time. (WDTC:WTE3 to WTE0 = 0101_B)

Refer Table 2.

The following example shows how to set up watchdog Timer for operation with software mode.

```
/* initial watchdog timer */
void InitWDT (void)
{
       WDTC = 0x05;         // set count clock is 2^21/F_CH
                            // start WDT counter
}


/* main routine */
void main (void)
{
       InitWDT();
       ...

       WDTC |= 0x05;        // clear WDT counter
}
```

main.c

```
#define HWD_DISABLE
;-------------------------------------------------------------------------
; Hard Watchdog
;-------------------------------------------------------------------------
#ifdef       HWD_DISABLE
             .SECTION  WDT, CONST, LOCATE=H'FFBE
                    .DATA.W   0xA596
#endif
```

startup.asm

Refer to Appendix Sample Code for project "SWWDT".

Please note that once the watchdog timer is activated, it cannot be stopped until a reset is generated.

4.3 Hardware Watchdog Timer

The hardware watchdog timer is set up through the following procedure:

1. Activate the watchdog timer by writing "$A597_H$" (WDT is enabled except in standby mode) or any other value (WDT is enabled in every mode) except other than "$A596_H$" and "$A597_H$" to the address $FFBE_H$ and $FFBF_H$ on the flash memory, which are copied to the watchdog timer selection ID register WDTH/WDTL ($0FEB_H$/$0FEC_H$)
2. The HWWDT starts automatically after a reset.
3. Clear the watchdog timer with minimum interval time. (WDTC:WTE3 to WTE0 = 0101_B)

Cause of while using the HWWDT, the count clock switch bits: CS1, CS0, CSP are fixed at "001", the interval time is fixed at $2^{16}/F_{CRL}$ too.

The following example shows how to set watchdog timer for operating in hardware mode.

```
/* Hardware watchdog timer starts automatically after a reset and */
/* cannot be stopped. The internal time is fixed to 2^16/FCRL       */

/* main routine */
void main (void)
{
       ...

       WDTC |= 0x05; // clear WDT counter

       ...
}
```

```
;-------------------------------------------------------------------------
; Hard Watchdog
;-------------------------------------------------------------------------
; #define HWD_DISABLE
#ifdef       HWD_DISABLE
             .SECTION  WDT, CONST, LOCATE=H'FFBE
                    .DATA.W   0xA596
#endif
```

main.c

startup.asm

Refer to Appendix Sample Code for project "HWWDT".

Please note that the hardware watchdog timer starts automatically after a reset and cannot stop its operation.

5 Note on Using Watchdog Timer

Take account of the following points when using the watchdog timer.

- Stopping the watchdog timer

Once activated, the watchdog timer cannot be stopped until a reset is generated.

If a HWWDT is select, it will resume operation after a reset.

- Selecting the count clock

Software watchdog timer

The count clock switch bits (WDTC: CS1, CS0, CSP) can be rewritten only when the watchdog control bits (WDTC: WTE3 to WTE0) are set to "0101B" upon the activation of the watchdog timer. The count clock switch bits cannot be written by a bit operation instruction. Moreover, the bit settings should not be changed once the timer is activated.

In sub-clock mode, the timebase timer does not operate because the main clock stops oscillating.

In order to operate the watchdog timer in sub-clock mode, it is necessary to select the watch prescaler as the count clock beforehand and set "WDTC: CS1, CS0, CSP" to "100_B" or "110_B" or "$XX1_B$".

Hardware watchdog timer

The count clock is fixed at $2^{16}/F_{CRL}$.

- Clearing the watchdog timer

Clearing the counter used for the count clock of the watchdog timer (timebase timer or watch prescaler or sub-CR timer) also clears the counter of the watchdog timer.

The counter of the watchdog timer is cleared when entering the sleep mode, stop mode or watch mode except in the case of selecting the hardware activation with the hardware watchdog timer running in a standby mode.

- Programming precaution

When creating a program in which the watchdog timer is cleared repeatedly in the main loop, set the processing time of the main loop including the interrupt processing time to the minimum watchdog timer interval time or shorter.

- Hardware watchdog (with timer running in a standby mode)

The watchdog timer does not stop in stop mode, sleep mode, timebase timer mode or watch mode. Therefore, the watchdog timer is not to be cleared by the CPU even if the internal clock stops. (in stop mode, sleep mode, timebase timer mode or watch mode).

Regularly release a standby mode and clear the watchdog timer. However, a watchdog reset may be generated depending on the setting of the oscillation stabilization wait time setting register after the CPU returns from stop mode in sub-clock mode or sub-CR mode.

Take account of the setting of the sub-clock stabilization wait time when selecting the sub-clock.

6 Additional Information

For more Information on MB95200 products, visit the following website:

http://www.cypress.com/8fx-mb95200

6.1 Sample Code

6.1.1 Project Name: SWWDT

Software watchdog timer

main.c

```
#include "mb95200.h"
/*-----------------------------------------------------------------------------
   name: Delay();
   function: delay function
-----------------------------------------------------------------------------*/
void Delay (unsigned int i)
{
       while(i--)
       {
              asm("\tNOP");
       }
}
/*-----------------------------------------------------------------------------
   name: InitWDT();
   function: initial watchdog timer
-----------------------------------------------------------------------------*/
void InitCompTimer (void)
{
       WDTC = 0x05;         // set count clock is 2^21/F_CH
                            // start WDT counter
}
/*-----------------------------------------------------------------------------
   name: main();
   function: main loop
-----------------------------------------------------------------------------*/
void main(void)
{
       PDR0_P05 = 0;        // initial value
       DDR0_P05 = 1;        // set P05 as output

       InitWDT ();
```

```
        while(1)
        {
                PDR0_P05 = ~PDR0_P05;       // show program is normal run
                Delay (500);

                WDTC |= 0x05;        // clear WDT timer within a certain amount of time

        }
}
startup.asm
;-------------------------------------------------------------------------------
; variable define declaration
;-------------------------------------------------------------------------------
#define HWD_DISABLE              ; if define this, Hard Watchdog will disable.
;-------------------------------------------------------------------------------
; Hard Watchdog
;-------------------------------------------------------------------------------
#ifdef        HWD_DISABLE
              .SECTION  WDT, CONST, LOCATE=H'FFBE
              .DATA.W   0xA596
#endif
```

6.1.2 Project Name: HWWDT

Hardware watchdog

main.c

```
#include "mb95200.h"
/* Hardware watchdog timer starts automatically after a reset and cannot be stopped */
/* The internal time is fixed to 2^16/F_CRL                                          */
/*-----------------------------------------------------------------------------
   name: Delay();
   function: delay function
-----------------------------------------------------------------------------*/
void Delay (unsigned int i)
{
       while(i--)
       {
              asm("\tNOP");
       }
}
/*-----------------------------------------------------------------------------
   name: main();
   function: main loop
-----------------------------------------------------------------------------*/
void main(void)
{
       PDR0_P05 = 0;              // initial value
       DDR0_P05 = 1;              // set P05 as output

       while(1)
       {
              PDR0_P05 = ~PDR0_P05;      // show program is normal run
              Delay (500);

              WDTC |= 0x05;       // clear WDT timer within a certain amount of time
       }
}
```

```
startup.asm
;-------------------------------------------------------------------------
; variable define declaration
;-------------------------------------------------------------------------
;#define HWD_DISABLE                    ; if define this, Hard Watchdog will disable.
;-------------------------------------------------------------------------
; Hard Watchdog
;-------------------------------------------------------------------------
#ifdef  HWD_DISABLE
             .SECTION  WDT, CONST, LOCATE=H'FFBE
             .DATA.W   0xA596
#endif
```

7 Document History

Document Title: AN205336 - F²MC-8FX Family, MB95200H/210H Series Watchdog Timer

Document Number: 002-05336

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	HUAL	03/20/2008	Initial release
			07/18/2008	Added URL in Chapter 6 Additional Information; modified figure number and some sample code.
*A	5267132	HUAL	05/11/2016	Migrated Spansion Application Note MCU-AN-500013-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

cypress.com/psoc

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

Cypress Developer Community

Community | Forums |Blogs | Video |Training

Technical Support

cypress.com/support

PSoC is a registered trademark and PSoC Creator 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
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2008-2016. 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.