AN205453

F²MC - 8FX Family, MB95200 Series, Clock Development using Sub-Clock

This document describes how to use the sub-clock of MB95200H/210H series MCU to design a clock, and illustrates the development with an example.

Contents

1 Introduction....1
2 Function Realization....1
2.1 Clock Controller Initialization....1
2.2 Watch Prescaler Initialization....2
3 Register Introduction....4
3.1 Registers related to Clock Control....4
3.2 Watch Pre-scalar Control Register....8
3.3 Clock System Configuration Controller....9
4 Clock Precision....10
5 Software Design....11
5.1 Software Development....11
6 Sample Code....12
Document History....14

1 Introduction

This document describes how to use the sub-clock of MB95200H/210H series MCU to design a clock, and illustrates the development with an example.

2 Function Realization

This chapter introduces how to realize the function of a clock using sub-clock.

2.1 Clock Controller Initialization

To develop a clock, it is one of the best methods to use the watch prescaler whose source is the sub-clock, because the sub-clock keeps operating as long as it is enabled, no matter if the system is in run mode or in sleep mode. First initialize the clock controller and enable both main clock and sub-clock. (SYCC2: MOSCE =1; SYCC2: SOSCE = 1 ;).

2.2 Watch Prescaler Initialization

The interval timer function continuously generates interrupt requests at regular intervals, using the sub-clock divided by two as its count clock.

- The counter of the watch prescaler counts down and an interrupt request is generated whenever the selected interval time has elapsed.
- The interval time can be selected from the following eight types:

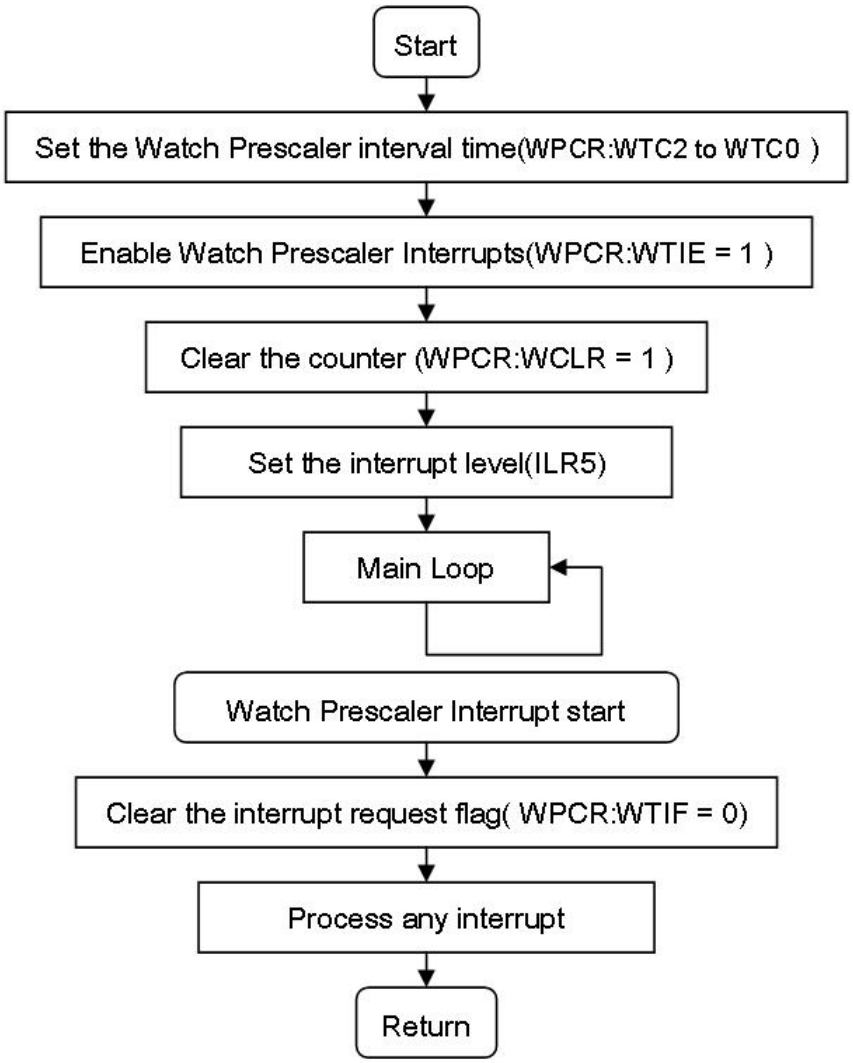
The table below lists the eight types of intervals:

Table 1. Interval Times of Watch Prescaler

	Interval time (Sub-CR clock) $(2^n \times 2/F_{CRL}{}^{*1})$	Interval time (Subclock) $(2^n \times 2/F_{CL}{}^{*2})$
n=10	20.48 [ms]	62.5 [ms]
n=11	40.96 [ms]	125 [ms]
n=12	81.92 [ms]	250 [ms]
n=13	163.84 [ms]	500 [ms]
n=14	327.68 [ms]	1 [s]
n=15	655.36 [ms]	2 [s]
n=16	1.311 [s]	4 [s]
n=17	2.621 [s]	8 [s]

The flow chart below indicates how to enter watch prescaler mode:

Figure 1. How to Enter Watch prescaler Mode



Setting procedures are as below:

- Initial setup
  1. Set the interrupt level. (ILR5)
  2. Set the interval time. (WPCR:WTC2 to WTC0)
  3. Enable interrupts. (WPCR:WTIE = 1)
  4. Clear the counter. (WPCR:WCLR = 1)
- Processing interrupts
  1. Clear the interrupt request flag. (WPCR:WTIF = 0)
  2. Process any interrupt.

3 Register Introduction

This chapter introduces the registers related to the clock development.

For more on register setting, please refer to Chapter 6 and Chapter 12 of the MB95200 Series Hardware Manual.

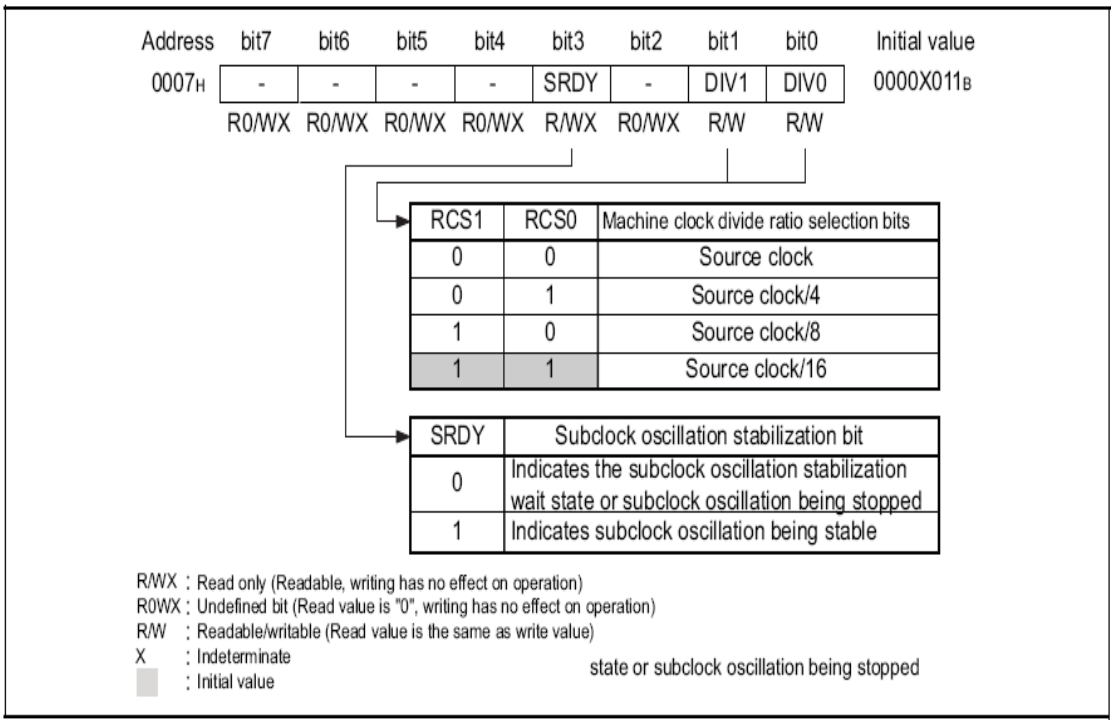
3.1 Registers related to Clock Control

This part introduces registers related to clock control one by one.

3.1.1 System Clock Control Register (SYCC)

The system clock control register (SYCC) is used to select the machine clock division ratio, and indicate the sub-clock oscillation stability.

Figure 2. System Clock Control Register (SYCC)



3.1.2 Oscillation Stabilization Wait Time Setting Register (WATR)

This register is used to set the oscillation stabilization wait time.

Figure 3. Oscillation Stabilization Wait Time Setting Register (WATR)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0005H	SWT3	SW72	SWT1	SWT0	MWT3	MWT2	MWT1	MWT0	11111111B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

MWT3	MWT2	MWT1	MWT0	Number of cycles	Main Oscillation Clock FCH = 4 MHz	
1	1	1	1	2^{14} - 2	(2^{14} - 2)/FCH	About 4.10 ms
1	1	1	0	2^{13} - 2	(2^{13} - 2)/FCH	About 2.05 ms
1	1	0	1	2^{12} - 2	(2^{12} - 2)/FCH	About 1.02 ms
1	1	0	0	2^{11} - 2	(2^{11} - 2)/FCH	511.5 µs
1	0	1	1	2^{10} - 2	(2^{10} - 2)/FCH	255.5 µs
1	0	1	0	2^{9} - 2	(2^{9} - 2)/FCH	127.5 µs
1	0	0	1	2^{8} - 2	(2^{8} - 2)/FCH	63.5 µs
1	0	0	0	2^{7} - 2	(2^{7} - 2)/FCH	31.5 µs
0	1	1	1	2^{6} - 2	(2^{6} - 2)/FCH	15.5 µs
0	1	1	0	2^{5} - 2	(2^{5} - 2)/FCH	7.5 µs
0	1	0	1	2^{4} - 2	(2^{4} - 2)/FCH	3.5 µs
0	1	0	0	2^{3} - 2	(2^{3} - 2)/FCH	1.5 µs
0	0	1	1	2^{2} - 2	(2^{2} - 2)/FCH	0.5 µs
0	0	1	0	2^{1} - 2	(2^{1} - 2)/FCH	0.0 µs
0	0	0	1	2^{1} - 2	(2^{1} - 2)/FCH	0.0 µs
0	0	0	0	2^{1} - 2	(2^{1} - 2)/FCH	0.0 µs

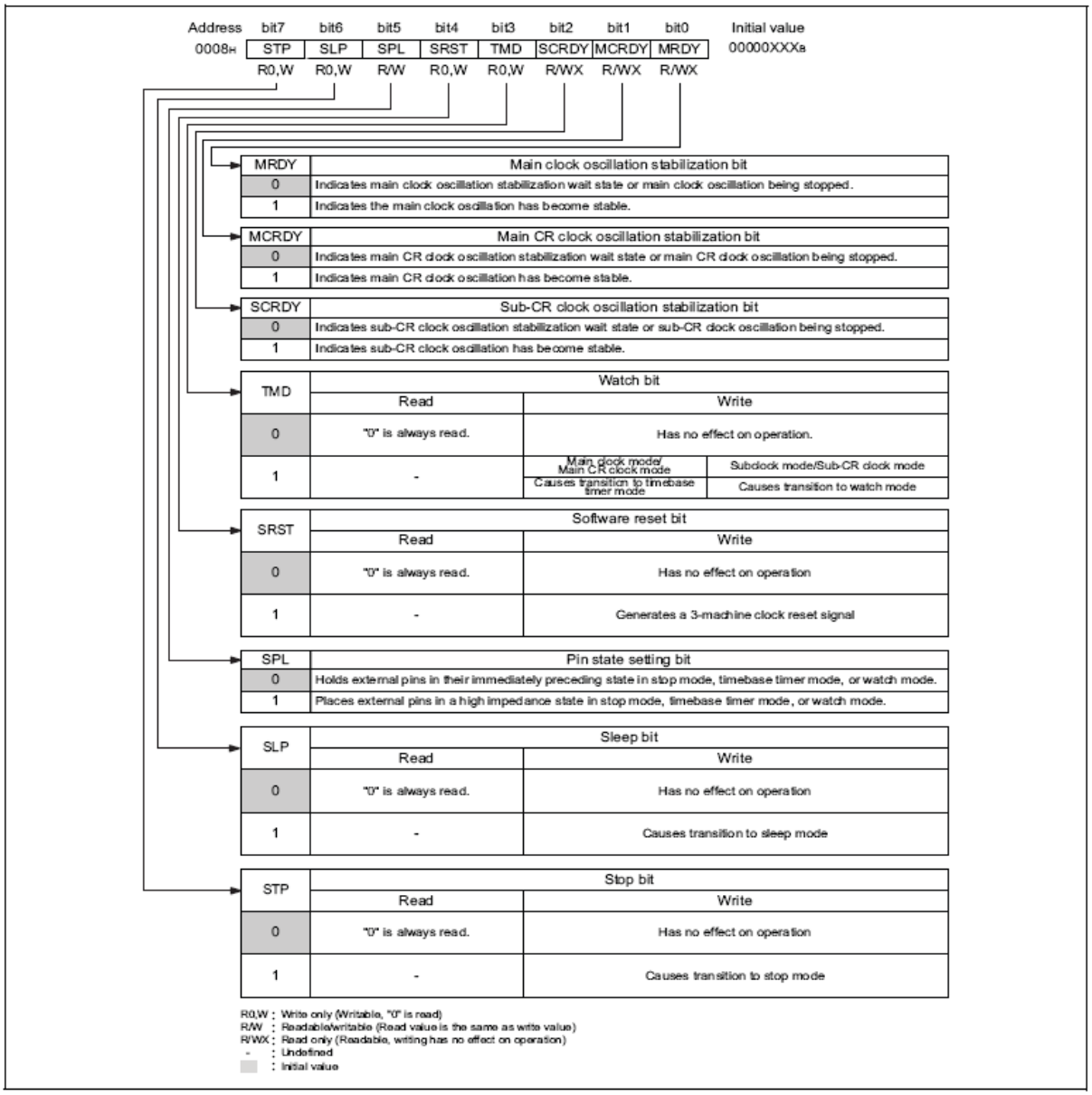
SWT3	SWT2	SWT1	SWT0	Number of cycles	Sub-oscillation Clock FCL = 32.768 kHz	
1	1	1	1	2^{15} - 2	(2^{15} - 2)/FCL	About 1.00 s
1	1	1	0	2^{14} - 2	(2^{14} - 2)/FCL	About 0.5 s
1	1	0	1	2^{13} - 2	(2^{13} - 2)/FCL	About 0.25 s
1	1	0	0	2^{12} - 2	(2^{12} - 2)/FCL	About 0.125 s
1	0	1	1	2^{11} - 2	(2^{11} - 2)/FCL	About 62.44 ms
1	0	1	0	2^{10} - 2	(2^{10} - 2)/FCL	About 31.19 ms
1	0	0	1	2^{9} - 2	(2^{9} - 2)/FCL	About 15.56 ms
1	0	0	0	2^{8} - 2	(2^{8} - 2)/FCL	About 7.75 ms
0	1	1	1	2^{7} - 2	(2^{7} - 2)/FCL	About 3.85 ms
0	1	1	0	2^{6} - 2	(2^{6} - 2)/FCL	About 1.89 ms
0	1	0	1	2^{5} - 2	(2^{5} - 2)/FCL	About 915.5 µs
0	1	0	0	2^{4} - 2	(2^{4} - 2)/FCL	About 427.2 µs
0	0	1	1	2^{3} - 2	(2^{3} - 2)/FCL	About 183.1 µs
0	0	1	0	2^{2} - 2	(2^{2} - 2)/FCL	About 61.0 µs
0	0	0	1	2^{1} - 2	(2^{1} - 2)/FCL	0.0 µs
0	0	0	0	2^{1} - 2	(2^{1} - 2)/FCL	0.0 µs

R/W : Readable/writable (Read value is the same as write value)
: Initial value (For mask ROM products, you can specify the initial value when ordering ROM.)

3.1.3 Standby Control Register (STBC)

The standby control register (STBC) is used to control transition from the RUN state to sleep mode, stop mode, time-base timer mode, or watch mode, to set the pin state in stop mode, time-base timer mode, and watch mode, and to control the generation of software resets.

Figure 4. Standby Control Register (STBC)



Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0008H	STP	SLP	SPL	SRST	TMD	SCRDY	MCRDY	MRDY	00000XXXB
	R0,W	R0,W	R/W	R0,W	R0,W	R/WX	R/WX	R/WX	

MRDY	Main clock oscillation stabilization bit
0	Indicates main clock oscillation stabilization wait state or main clock oscillation being stopped.
1	Indicates the main clock oscillation has become stable.

MCRDY	Main CR clock oscillation stabilization bit
0	Indicates main CR clock oscillation stabilization wait state or main CR clock oscillation being stopped.
1	Indicates main CR clock oscillation has become stable.

SCRDY	Sub-CR clock oscillation stabilization bit
0	Indicates sub-CR clock oscillation stabilization wait state or sub-CR clock oscillation being stopped.
1	Indicates sub-CR clock oscillation has become stable.

TMD	Watch bit		
	Read	Write	
0	"0" is always read.	Has no effect on operation.	
1	-	Main clock mode/ Main CR clock mode	Subclock mode/Sub-CR clock mode
		Causes transition to timebase timer mode	Causes transition to watch mode

SRST	Software reset bit	
	Read	Write
0	"0" is always read.	Has no effect on operation
1	-	Generates a 3-machine clock reset signal

SPL	Pin state setting bit
0	Holds external pins in their immediately preceding state in stop mode, timebase timer mode, or watch mode.
1	Places external pins in a high impedance state in stop mode, timebase timer mode, or watch mode.

SLP	Sleep bit	
	Read	Write
0	"0" is always read.	Has no effect on operation
1	-	Causes transition to sleep mode

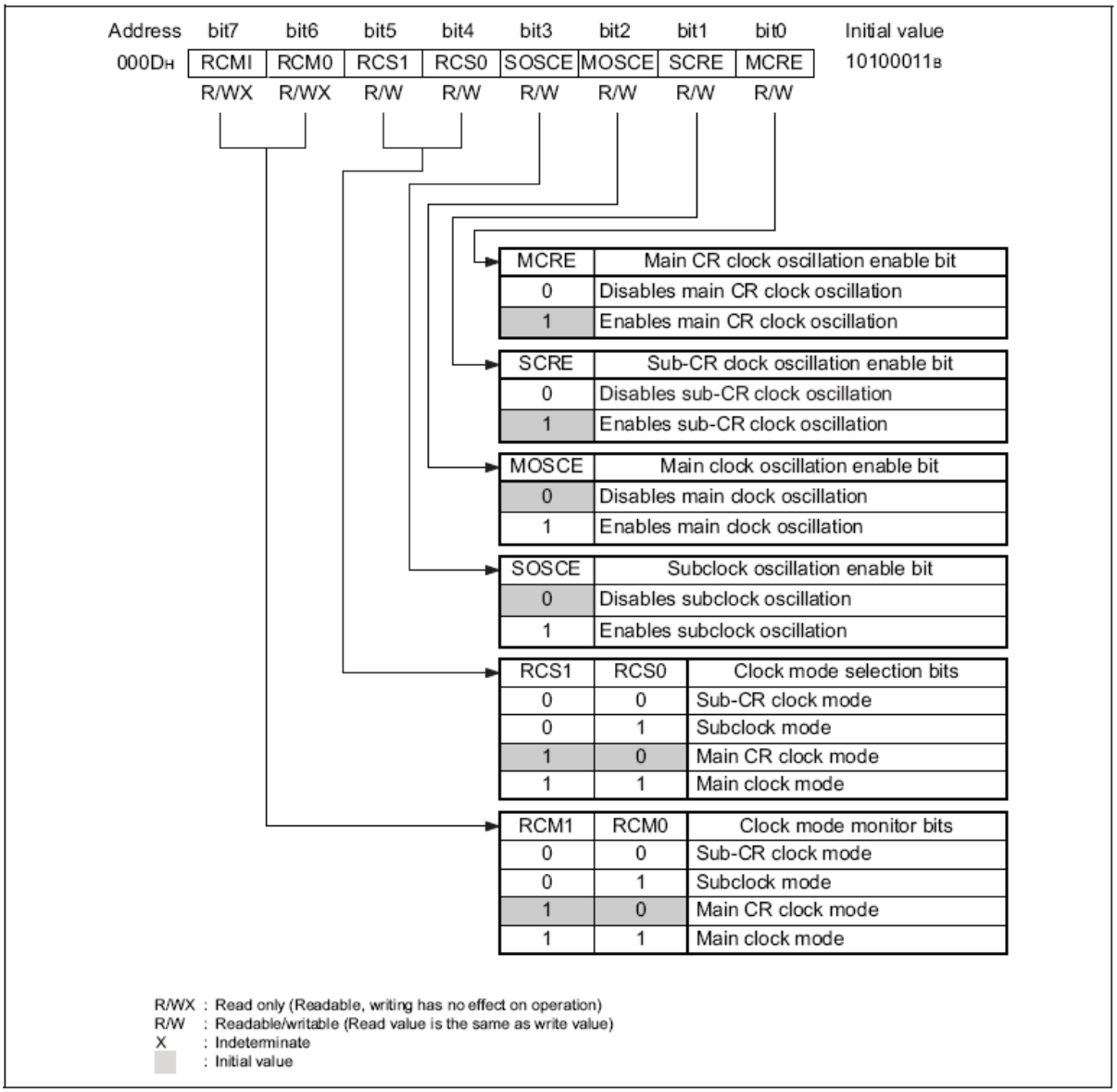
STP	Stop bit	
	Read	Write
0	"0" is always read.	Has no effect on operation
1	-	Causes transition to stop mode

R0,W : Write only (Writable, "0" is read)
R/W : Readable/writable (Read value is the same as write value)
R/WX : Read only (Readable, writing has no effect on operation)
- : Undefined
: Initial value

3.1.4 System Clock Control Register 2 (SYCC2)

The system clock control register 2 (SYCC2) is used to indicate and switch the current clock mode, and control sub-clock, sub-CR clock, main clock, main CR clock oscillations.

Figure 5. System Clock Control Register 2 (SYCC2)



3.2 Watch Pre-scalar Control Register

This part introduces the watch pre-scalar control register. The watch pre-scalar r control register (WPCR) is used to select the interval time, clear the counter, control interrupts and check the status.

Figure 6. Watch prescaler Control Register (WPCR)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000B$_H$	WTIF	WTIE	-	-	WTC2	WTC1	WTC0	WCLR	00000000$_B$
	R(RM1),W	R/W	R0/WX	R0/WX	R/X	R/W	R/W	R0,W	

WCLR	Watch timer initialization bit	
	Read	Write
0	"0" is always read	No change, No effect on operation
1	-	Clears watch prescaler counter

WTC2	WTC1	WTC0	Interval time (Sub-clock F_{CL}=32.768kHz)	Interval time (Sub-CR clock F_{CRL}=100kHz)
1	0	0	$2^{10} \times 2/F_{CL}$ (62.5ms)	$2^{10} \times 2/F_{CRL}$ (20.48ms)
0	0	0	$2^{11} \times 2/F_{CL}$ (125.ms)	$2^{11} \times 2/F_{CRL}$ (40.96ms)
0	0	1	$2^{12} \times 2/F_{CL}$ (250.ms)	$2^{12} \times 2/F_{CRL}$ (81.92ms)
0	1	0	$2^{13} \times 2/F_{CL}$ (500.ms)	$2^{13} \times 2/F_{CRL}$ (163.84ms)
0	1	1	$2^{14} \times 2/F_{CL}$ (1.s)	$2^{14} \times 2/F_{CRL}$ (327.68ms)
1	0	1	$2^{15} \times 2/F_{CL}$ (2.s)	$2^{15} \times 2/F_{CRL}$ (655.36ms)
1	1	0	$2^{16} \times 2/F_{CL}$ (4.s)	$2^{16} \times 2/F_{CRL}$ (1.311s)
1	1	1	$2^{17} \times 2/F_{CL}$ (8.s)	$2^{17} \times 2/F_{CRL}$ (2.621s)

WTIE	Interrupt request enable bit
0	Disables interrupt request output
1	Enables interrupt request output

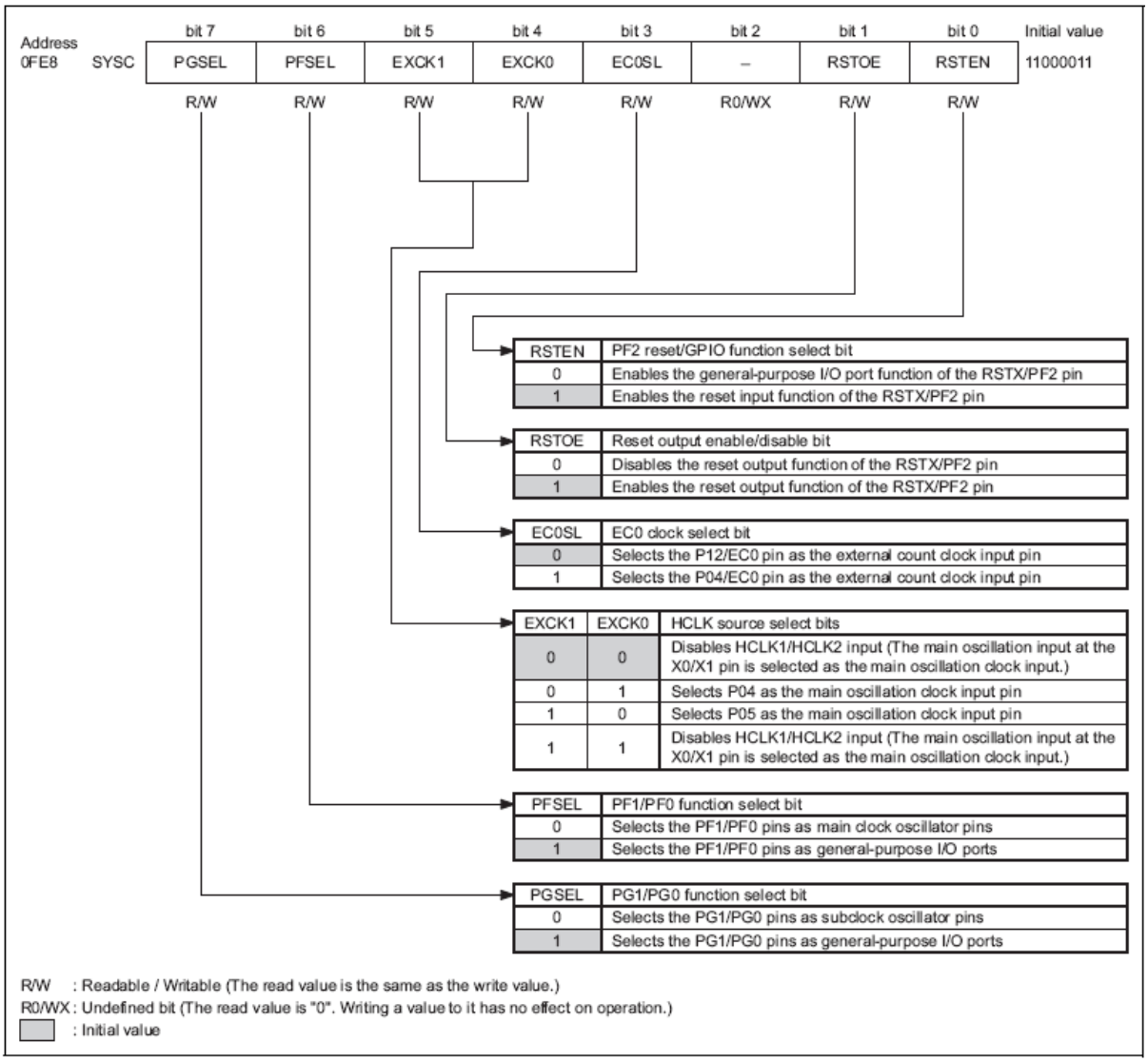
WTIF	Watch interrupt request flag bit	
	Read	Write
0	Interval time has not elapsed	Clears the bit
1	Interval time has elapsed	No change, No effect on operation

R(RM1),W : Readable/writable (Read value is different from write value, "1" is read by read-modify-write (RMW) instruction)
R/W: Readable/writable (Read value is the same as write value)
R0/WX: Undefined bit (Read value is "0", writing has no effect on operation)
R0,W: Write only (Writable, "0" is read)
-: Undefined
Initial value

3.3 Clock System Configuration Controller

The controller consists of the SYSC register, which is an 8-bit (bit 2 not used) register used to configure the clock and reset system. Never forget to enable the main clock and sub-clock input by setting the corresponding bits in the register.

Figure 7. Clock System Configuration Controller (SYSC)



4 Clock Precision

This chapter introduces the factors which may affect the precision of the clock developed.

To develop a clock using the sub-clock of MB95200 series MCU, two factors affecting clock precision need special attention; one is sub-clock's frequency. The interval timer function continuously generates interrupt requests at regular intervals, using the sub-clock divided by two as its count clock. Interval time = 2nx2/FCRL, the interval time can be selected from the following eight types:

Table 2. Interval Times of Watch Prescaler

	Interval time Sub-CR clock case $(2^n \times 2/F_{CRL}*1)$	Interval time Sub-clock case $(2^n \times 2/F_{CL}*2)$
n=10	20.48 [ms]	62.5 [ms]
n=11	40.96 [ms]	125 [ms]
n=12	81.92 [ms]	250 [ms]
n=13	163.84 [ms]	500 [ms]
n=14	327.68 [ms]	1 [s]
n=15	655.36 [ms]	2 [s]
n=16	1.311 [s]	4 [s]
n=17	2.621 [s]	8 [s]

It is recommended to use the sub-clock of 32.768 kHz frequency. The other factor affecting clock precision is the watch prescaler interval time. The selection of interval time can be based on the formula interrupt time=1second/interval time, the best selection is the interval time and interrupt time with the arithmetical compliment close to zero. For example: if the sub-clock is 32.768kHz and the watch prescaler is interrupted four times in one second, the interval time is equal to 1.s/4=250ms. Besides, the order period of clock code also can make difference, so the simplification of clock code is very important.

5 Software Design

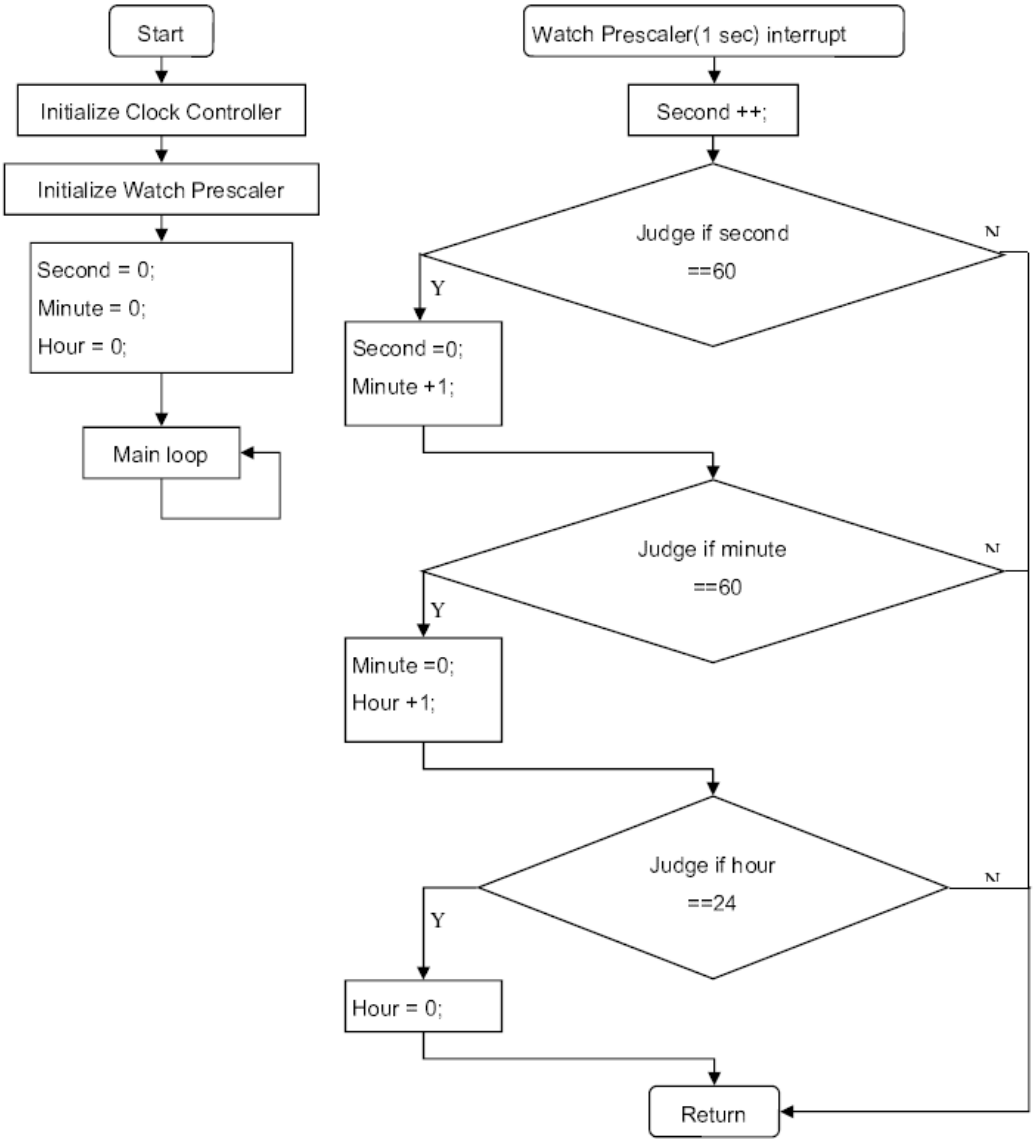
This chapter introduces software used in the clock development.

5.1 Software Development

Before performing this function, first initialize the system clock control registers, enable the main clock and sub-clock input, initialize the watch prescaler register, and set up interrupt levels. If the oscillation becomes stable, the prescaler starts operating, and an interrupt occurs. In interrupt processing, one second counts up automatically each time an interrupt occurs. When the second counting reaches 60, one minute counts up, and the second counting returns to 0. In the same way, when the minute counting has reached 60, one hour counts up, and the minute counting returns to 0. When the hour counting has reached 24, it clears to 0.

The flow chart is illustrated as below:

Figure 8. Flow Chart of Clock Design by Using Sub-clock



6 Sample Code

This chapter gives an example of clock development using the sub-clock.

The following example shows how to use the prescaler of sub-clock to develop a clock.

```
/*                                SAMPLE CODE                                */
/*-------------------------------------------------------------------------*/
/* Give a example for Clock */
/* Clock Controller initial */
void System_SubC_init(void)       // Clock Controller initial
{
       SYSC_PGSEL = 0;                    // Enable sub-clock input
       SYSC_PFSEL = 0;                    // Enable main-clock input
       WATR = 0xC0;
       STBC = 0x00;
       SYCC2 = 0xFC;                      // Enable sub-clock and main-clock
       SYCC = 0x00;
       while(SYCC_SRDY);
}
/* Watch Prescaler initial */
void Watch_prescalar_initial(void)
{
       WPCR = 0x46; // Enable interrupt, interval time = 1.s
       WPCR_WCLR = 1; // Clears watch prescaler counter
}
/* Watch Prescaler interrupt process */
__interrupt void Watch_Prescalar_Interrupt(void)
{
       WPCR_WTIF = 0;                     // Clear interrupt flag bit
       C_Second++;                 // Second ++
       Second_output();
       if(C_Second==0x3C)          // 60 seconds
       {
              C_Second = 0;               // Second return to 0
              Clock_minute_process(); // Minute ++
       }
}
/* Minute process*/
void Clock_minute_process(void)
{
       C_Minute++;                 // Minute ++
       if(C_Minute == 0x3C)              // 60 Minutes
       {
              C_Minute = 0;               // Minute return to 0
              Clock_hour_process();       // Hour ++
       }
}
```

```
/* Hour process*/
void Clock_hour_process(void)
{
       C_Hour++;                    // Hour ++
       Hour_output();
       if(C_Hour == 0x18)   // 24 Hours
       {
              C_Hour = 0;           // Hour return to 0
       }
}
/* Main process */
void main(void)
{
       InitIrqLevels();// initialize Interrupt level and IRQ vector table
       __EI();                              // enable interrupt
       C_Second = 0;
       C_Minute = 0;
       C_Hour = 0;
       System_SubC_init();                  // Clock Controller initial
       Watch_prescalar_initial();           // Watch prescaler initial
       while(1)
       {
       asm("\tNOP");
       asm("\tNOP");
       asm("\tNOP");
       }
```

Below is the initial vector:

```
void InitIrqLevels(void)
{
       ILR5 = 0xFC; // IRQ20: Watch timer / counter
}
. . .
__interrupt void Watch_Prescalar_Interrupt (void);
. . .
#pragma intvect Watch_Prescalar_Interrupt 20 // IRQ20: Watch timer /
counter
```

Document History

Document Title: AN205453 – F²MC - 8FX Family, MB95200 Series, Clock Development using Sub-Clock

Document Number: 002-05453

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	Benjamin. Yang	02/04/2009	First Draft
*A	5267156	HUAL	05/11/2016	Migrated Spansion Application Note “MCU-AN-500029-E-10” to Cypress format.
*B	5844573	AESATMP9	08/04/2017	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, 2009-2017. 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.