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.

AN205271

F²MC-8FX 家族 8 位微型控制器 MB95200H/210H 系列 基本固件设置

本文档介绍了如何对 Cypress F²MC-8FX 家族 MB95200H/210H 系列进行固件设置。

本文档同时介绍了初始化程序，堆栈指针（SP），直接存储区指针（DP），时钟控制以及待机控制。

目录

1 概要 1
2 初始化程序.....2
2.1 初始化堆栈.....2
2.2 SP 和 DP 设置.....2
3 中断向量.....3
3.1 中断级别.....3
3.2 中断处理程序函数原型.....3
3.3 向量定义.....3
3.4 默认中断服务程序.....4
3.5 实例.....4
4 时钟控制器.....5
4.1 时钟控制器概要.....5
4.2 时钟控制器的结构图.....5
4.3 寄存器.....6
5 时钟模式选择.....7
5.1 介绍.....7
5.2 主要特性.....7
5.3 主时钟.....7
5.4 副时钟.....8
5.5 主 CR 时钟.....8
5.6 副 CR 时钟.....9
5.7 时钟分频率.....9
6 待机模式选择.....9
6.1 主要特性.....9
6.2 待机模式和时钟输入状态.....10
6.3 时钟模式和待机模式的组合.....10
6.4 休眠模式.....11
6.5 停止模式.....11
6.6 时基定时器模式.....11
6.7 计时模式.....12
6.8 待机模式切换代码.....13
7 更多信息.....13
A 附录.....14
A.1 代码范例.....14
文档修改记录.....28

1 概要

本文档介绍了如何对 Cypress F²MC-8FX 家族 MB95200H/210H 系列进行固件设置。

本文档同时介绍了初始化程序，堆栈指针（SP），直接存储区指针（DP），时钟控制以及待机控制。

2 初始化程序

开始文件 startup.asm 是重置操作后的入口，包括针对编译器和设备操作的多种设置。它初始化堆栈和变量的初始值。

2.1 初始化堆栈

启动 MCU 时，必须初始化堆栈，设置堆栈大小，并定义堆栈区。

初始化的代码示例：

```
//Sample code for initialization
//definition to stack area
      .SECTION    STACK,     STACK,     ALIGN=1
      .RES.B   128-2
STACK_TOP:
      .RES.B   2
```

注意：上述代码截取自 start.asm，参见附录。

2.2 SP 和 DP 设置

SP 和 DP 是专用寄存器。

堆栈指针（SP）是一个 16 位的寄存器，它保存堆栈推动和弹出指令导致的中断或子程序调用发生时引用的地址。重置后的的初始值为“0000H”。

DP 指直接存储区指针。DP 位于程序状态（PS）寄存器的第 10-8 位，用于指明直接地址要访问的区域。

初始化的代码示例：

```
    .SECTION CODE,CODE,ALIGN=1
;---------------------------------------------
;set stack pointer
;---------------------------------------------
      MOVW     A , #STACK_TOP
      MOVW     SP, A
// Set Register bank Pointer 0 / set Direct bank Pointer 0 (0x80...0xFF)
//The PS register consists of the register bankpointer (RP), direct
//pointer (DP), and condition code register (CCR).
       MOVW       A, PS
       MOVW       A, #0x07FF// RP=0, DP=0, I=0
       ANDW       A
       MOVW       PS, A
```

注意：上述代码截取自 start.asm，参见附录。

3 中断向量

开始基本固件设置后，必须初始化中断向量。

用户可使用 C 文件 vector.c。该 C 文件公布了中断级别和数字，包括默认中断处理程序。相应的中断控制寄存器也可被初始化。

3.1 中断级别

文件 vector.c 首先定义中断级别函数 void InitIrqLevels (void)。

```
void InitIrqLevels (void)
{
      ILR0 = 0Xff    // IRQ0:  external interrupt ch.4       --> 01
                     // IRQ1: external interrupt ch.5        --> 01
                     // IRQ2: external interrupt ch.2/ch.6   --> 01
                     // IRQ3: external interrupt ch.3/ch.7   --> 01
    ......
}
```

注意：上述代码截取自 vector.c，参见附录。

该函数初始化为每一个共享中断通道定义中断级别的中断控制寄存器。FF 指最低优先级。

3.2 中断处理程序函数原型

中断处理程序的函数原型将在级别初始化后公布。

```
__interrupt void DefaultIRQHandler (void)
// Add your own prototypes here like above.
```

注意：上述代码截取自 vector.c，参见附录。

3.3 向量定义

向量数字与中断处理函数相关联。对于未使用的中断，请使用默认的中断处理程序。

```
#pragma   intvect  DefaultIRQHandler 0//IRQ0: external interrupt ch.4
#pragma   intvect  DefaultIRQHandler 1//IRQ1: external interrupt ch.5
#pragma   intvect  DefaultIRQHandler 2//IRQ2: external interrupt ch.2/ch.6
#pragma   intvect  DefaultIRQHandler 3//IRQ3: external interrupt ch.3/ch.7
......
```

注意：上述代码截取自 vector.c，参见附录。

3.4 默认中断服务程序

默认中断服务程序在 vector.c 文件旁列出。该函数通过一个无限循环停止系统。对于调试来说，有必要在此设置一个断点检测未初始化的中断。

默认中断处理程序的代码示例：

```
//default interrupt handler
__interrupt void DefaultIRQHandler (void)
{
      __DI(); // disable interrupts
      While (1)
__wait_nop ();   // halt system
}
```

注意：上述代码截取自 vector.c，参见附录。

3.5 实例

如果需要在使用中断时修改中断向量，例如设置外部中断 ch4 as key_in，请修改 vector.c 中的代码。

```
#pragma intvect Key_in      1     // IRQ0:  external interrupt ch4
```

注意：上述代码截取自 vector.c，参见附录。

中断级别的定义：

```
// defines the interrupt levels
void InitIrqLevels (void)
{
      ILR0=0xFC   // IRQ0:  external interrupt   ch.4         --> Level 00
                  // IRQ1:  external interrupt   ch.5         --> 01
                  // IRQ2:  external interrupt ch.2 | ch.6    --> 01
                   // IRQ3:  external interrupt ch.3 | ch.7   --> 01
 ......       //elide the following code
}
```

注意：上述代码截取自 vector.c，参见附录。

在文件 main.c 中，输入中断处理程序函数_interrupt void Key_Int (void)。

```
//interrupt response function
_interrupt voidKey_int (void)// key_int external interrupt ch.4
{
           // enter your interrupt handler function(s) here
}
```

注意：上述代码截取自 main.c，参见附录。

4 时钟控制器

本章介绍了时钟控制器的函数和操作。

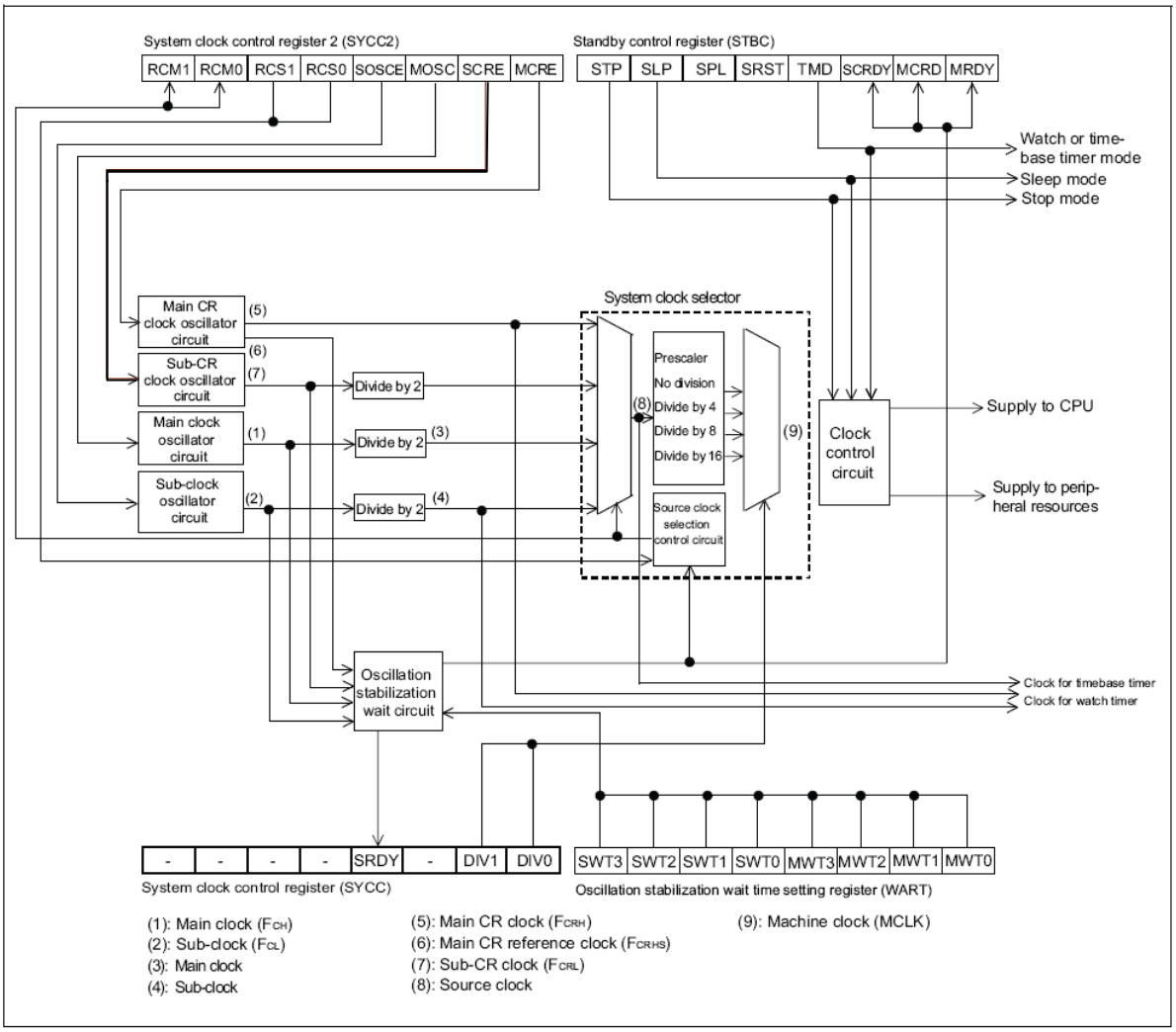
4.1 时钟控制器概要

F²MC-8FX 家族有一个内置时钟控制器，用于优化功耗。它包括一个双外部时钟产品支持外部主时钟和外部副时钟，以及一个单外部时钟产品仅支持外部主时钟。

时钟控制器启用/禁用时钟振荡，启用/禁用对内部电路的时钟信号输入，选择时钟源，以及控制内部 CR 振荡器和分频电路。

4.2 时钟控制器的结构图

图 1. 时钟控制器结构图



4.3 寄存器

时钟控制器包括四个寄存器：系统时钟控制寄存器（SYCC），待机控制寄存器（STBC），系统时钟控制寄存器 2（SYCC2）以及振荡稳定等待时间设置寄存器（WATR）。

参见 MCU MB95200H/210H 系列硬件手册的第六章了解更多信息。

图 2 显示了系统时钟控制寄存器（SYCC）的配置。该寄存器用于控制机械时钟分频率选择。

图 2. 系统时钟控制寄存器的配置（SYCC）

地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	初始值
0007_H	-	-	-	-	SRDY	-	DIV1	DIV0	$0000X011_B$
	R0/WX	R0/WX	R0/WX	R0/WX	R/WX	R0/WX	R/W	R/W	

图 3 显示了振荡稳定等待时间设置寄存器（WATR）的配置。该寄存器用于设置振荡稳定等待时间。

图 3. 振荡稳定等待时间设置寄存器（WATR）的配置

地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	初始值
0005_H	SWT3	SWT2	SWT1	SWT0	MWT3	MWT2	MWT1	MWT0	11111111_B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

图 4 显示了待机控制寄存器（STBC）的配置。该寄存器用于控制从运行模式到休眠模式、停止模式、时基定时器模式、计时模式的转换，在停止模式，时基定时器模式和计时模式中设置引脚状态，以及控制软件重置的产生。

图 4. 待机控制寄存器（STBC）

地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	初始值
0008_H	STP	SLP	SPL	SRST	TMD	SCRDY	MCRDY	MRDY	$00000XXX_B$
	R0/W	R0/W	R/W	R0/W	R0/W	R/WX	R/WX	R/WX	

图 5 显示了系统时钟控制寄存器 2（SYCC2）的配置。该寄存器用于指示和切换当前时钟模式，以及控制副时钟、主时钟、副 CR 时钟，主 CR 时钟和主时钟振荡。

图 5. 系统时钟控制寄存器 2 的配置（SYCC2）

地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	初始值
$000D_H$	RCM1	RCM0	RCS1	RCS0	SOSCE	MOSCE	SCRE	MCRE	10100011_B
	R/WX	R/WX	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: 读/写（读取值与烧写值一致）

R/WX: 只读（可读，烧写无效）

R0/WX: 未定义（读取“0”，烧写无效）

R0/W: 只写（可写，读取“0”）

5 时钟模式选择

5.1 介绍

一共有四种时钟（主时钟，副时钟，主 CR 时钟，副 CR 时钟）可供选择（单外部时钟产品只有三种时钟）。

5.2 主要特性

- 可选择的主时钟源
  - 外部主时钟（频率高达 32.5MHz，最大机械时钟频率为 16.25 MHz）
  - 主内部 CR 时钟（1/8/10 MHz）
- 可选择的副时钟源
  - 外部副时钟（32.768 kHz）
  - 副 OSC 时钟（32.768 kHz）
  - 副内部 CR 时钟（典型值： 100 kHz，最小值： 50 kHz，最大值：200 kHz）
- 初始时钟为主 CR 时钟（8 MHz）

5.3 主时钟

主时钟为外部主时钟。外部时钟的最大频率为 32.5MHz，最大机械时钟频率为 16.25 MHz。使用主时钟前，确认 SYSC_PFSEL 位为 0。该位用于选择 PF1/PF2 引脚的功能。如果该位为 0，则 PF1/PF2 引脚用作主时钟振荡器引脚。

注意：参考硬件手册的第 23 章了关于 PFSEL 设置的更多信息。

更新 SYCC2_MOSCE 启用主时钟，更新 SYCC2_RCS 选择主时钟模式，更新 WATR 改变振荡稳定等待时间，并检查 STBC_MRDY 指示主时钟振荡是否稳定。

如果设置 WATR 为 0x0F，振荡稳定等待时间为$(2^{14}-2)/F_{CH}$：大约 4.10ms。

以下代码显示了如何操作主时钟，包括如何启用主时钟，切换至主时钟模式，更新振荡稳定等待时间以及等待振荡稳定。

```
// enable the Main clock oscillation
SYCC2_MOSCE = 1;
//Clock Mode Selection,select the main clock mode
SYCC2_RCS0 = 0x01;
SYCC2_RCS1 = 0x01;
//update Oscillation Stabilization Wait Time
//when Main Oscillation Clock F_CH=4MHz
WATR = 0X0F;         //Oscillation Stabilization Wait Timemean
                     // (2^14-2)/F_CH About 4.10 ms
While(!STBC_MRDY);   //Indicates main clock oscillation being stable
```

注意：上述代码截取自工程 Clock_Change，参见附录。

5.4 副时钟

副时钟是外部副时钟。外部副时钟的最大频率是 32.768 kHz，相应的副时钟频率为 32.768 kHz。使用副时钟前，确认 SYSC_PGSEL 位为 0。该位用于选择 PG1/PG2 引脚的功能。如果该位为 0，PG1/PG2 引脚选作副时钟振荡器引脚。

注意：参考硬件手册的第 23 章了关于 PGSEL 设置的更多信息。

更新 SYCC2_SOSCE 启用副时钟，更新 SYCC2_RCS 选择副时钟模式，更新 WATR 改变振荡稳定等待时间，并检查 SYCC_SRDY 指示主时钟振荡是否稳定。

如果设置 WATR 为 0xF0，振荡稳定等待时间为$(2^{15}-2)/F_{CL}$：大约 1.00s。

以下代码显示了如何操作副时钟，包括如何启用副时钟，切换至副时钟模式，更新振荡稳定等待时间以及等待振荡稳定。

```
// enable the sub-clock oscillation
SYCC2_SOSCE = 1;
//Clock Mode Selection,select the sub-clock mode
SYCC2_RCS0 = 0x01;
SYCC2_RCS1 = 0x00;
//Update the Oscillation Stabilization Wait Time
// when sub Oscillation ClockF_CL=32.768kHz
WATR = 0xF0;          //(2^15-2)/F_CL About 1.00s
while (!SYCC_SRDY); //Indicates sub-clock oscillation being stable
```

注意：上述代码截取自工程 Clock_Change，参见附录。

5.5 主 CR 时钟

主 CR 时钟是主内部 CR 时钟，有三个时钟频率：1 MHz，8 MHz，10 MHz。更新 NVR 可改变主 CR 时钟。

更新 SYCC2_MCRE 可启用主 CR 时钟，更新 SYCC2_RCS 选择主 CR 时钟模式，并检查 STBC_MCRDY 指示主 CR 时钟振荡是否稳定。

以下代码显示了如何操作主 CR 时钟，包括如何改变主 CR 时钟频率，启用主 CR 时钟，切换至主 CR 时钟模式，更新振荡稳定等待时间以及等待振荡稳定。

```
; change the main CR clock frequency
MOV A ,    0x0FE4        ;READ CRTH for NVR trimming value protect
AND A ,    #0x9F      ;Just CLEAR CRTH [6:5]
OR  A ,   #0x60       ; Main CR clock update to 8MHz
; Can update the Main CR clock as below setting
; 0x00: 1MHz;  0x40: 10MHz;   0x60: 8MHz
MOV 0x0FE4 , A        ; WRITE CRTH to enable the update
```

注意：上述代码截取自工程 1 start.asm，参见附录。

```
// enable the main CR clock oscillation
SYCC2_MCRE = 1;
// Clock Mode Selection, select the main CR clock mode
SYCC2_RCS0 = 0x00;
SYCC2_RCS1 = 0x01;
//update the Oscillation Stabilization Wait Time
//F_CRHS represents the main clock frequency
//2^8/F_CRHS = Main CR oscillation Stabilization Wait Time
while(!STBC_MCRDY);//Indicates main CR clock oscillation being stable
```

注意：上述代码截取自工程 Clock_Change，参见附录。振荡稳定等待时间根据 CR 启动时序而改变。

主 CR 振荡稳定等待时间= $2^8/F_{CRHS}$。F_{CRHS} 指 CR 时钟频率。

5.6 副 CR 时钟

副 CR 时钟是单内部时钟，典型频率为 100 kHz，最小频率为 50 kHz，最大频率为 200 kHz。

更新 SYCC2_SCRE 可启用副 CR 时钟，更新 SYCC2_RCS 选择副 CR 时钟模式，并检查 STBC_SCRDY 指示副 CR 时钟振荡是否稳定。

以下代码显示了如何操作副 CR 时钟，包括如何启用副 CR 时钟，切换至副 CR 时钟模式，更新振荡稳定等待时间以及等待振荡稳定。

```
// enable the Sub-CR Clock oscillation
SYCC2_SCRE = 1;
// Clock Mode Selection, select the Sub-CR Clock mode
SYCC2_RCS = 0x00;
//update the Oscillation Stabilization Wait Time
//FCRHS represents the main clock frequency
//2^4/FCRL = Sub-CR oscillation Stabilization Wait Time
while (!STBC_SCRDY); //Indicates sub-CR clock oscillation being stable
```

注意：上述代码截取自工程 Clock_Change，参见附录。振荡稳定等待时间根据 CR 启动的时序改变。主 CR 振荡稳定等待时间= $2^4/F_{CRL}$。F_{CRL} 指副 CR 时钟频率。

5.7 时钟分频率

机械时钟由源时钟根据分频率产生。设置 SYCC_DIV 划分源时钟。如果 SYCC_DIV 设置为 0x11，源时钟将被除以 16。

以下代码显示了如何设置时钟分频率。

```
//source clock can be main clock divided by 2, sub-clock divided by 2
//sub-CR clock divided by 2 or main CR clock no div
SYCC_DIV = 0x00;      // Source clock (No division)
                      // SYCC_DIV = 0x01 mean Source clock/4
                      // SYCC_DIV = 0x02 mean Source clock/8
                      // SYCC_DIV = 0x03 mean Source clock/16
```

注意：上述代码截取自工程 Clock_Change，参见附录。

6 待机模式选择

待机模式一共有四种：休眠模式，时基定时器模式，计时模式和停止模式。时钟控制器根据每种待机模式选择是否启用或禁用时钟振荡以及输入时钟至内部电路。

除时基定时器模式和计时模式外，待机模式可独立于时钟模式设置。

6.1 主要特性

- 低功耗模式（待机模式）
  - 停止模式
  - 休眠模式
  - 计时模式
  - 时基定时器模式
- 初始模式
  - 正常操作情况

6.2 待机模式和时钟输入状态

表 1. 待机模式和时钟输入状态

待机模式	时钟提供状态
休眠模式	停止向 CPU 提供时钟，CPU 停止操作，但其他外围资源继续操作。
时基定时器模式	仅向时基定时器和计时预分频器提供时钟信号，停止向其他电路提供时钟。除时基定时器，计时预分频器，外部中断，和低电压检测重置（可选）外的所有功能将停止。 时基定时器模式是主时钟模式或主 CR 时钟模式唯一的待机模式。
计时模式	停止主时钟振荡，仅向计时预分频器提供时钟信号，停止向其他电路提供时钟。除计时预分频器，外部中断，和低电压检测重置（可选）外的所有功能将停止。 计时模式是副时钟模式或副 CR 时钟模式唯一的待机模式。
停止模式	停止主时钟振荡，副时钟振荡，并停止所有时钟信号的输入。除外部中断和低电压检测重置（可选）外的所有功能将停止。

6.3 时钟模式和待机模式的组合

图 6 列出了时钟模式和待机模式的组合，以及他们各自内部电路的操作状态。

参见 MCU MB95200H/210H 系列硬件手册的第六章了解更多信息。

图 6. 待机模式和时钟模式的组合以及内部操作状态

Function	RUN				Sleep				Timebase Timer		Watch prescaler		Stop	
	Main clock mode	Main CR clock mode	Sub-clock mode (Dual-system clock product)	Sub-CR clock mode	Main clock mode	Main CR clock mode	Sub-clock mode (Dual-system clock product)	Sub-CR clock mode	Main clock mode	Main CR clock mode	Sub-clock mode (Dual-system clock product)	Sub-CR clock mode	Main CR clock mode	Sub-CR clock mode
Main clock	Operating	Stopped*1	Stopped		Operating	Stopped*1	Stopped		Operating	Stopped*1	Stopped		Stopped	
Main CR clock	Stopped*2	Operating	Stopped		Stopped*2	Operating	Stopped		Stopped	Operating	Stopped		Stopped	
Sub-clock	Operating*3		Operating		Operating*3		Operating		Operating*3		Operating		Operating*3	Stopped
Sub-CR clock	Stopped*4		Stopped*4	Operating	Stopped*4		Stopped*4	Operating	Stopped*4		Stopped*4	Operating	Stopped*4	Stopped
CPU	Operating		Operating		Stopped		Stopped		Stopped		Stopped		Stopped	
ROM RAM	Operating		Operating		Value held		Value held		Value held		Value held		Value held	
I/O ports	Operating		Operating		Output held		Output held		Output held		Output held		Output held	
Timebase timer	Operating		Stopped		Operating		Stopped		Operating		Stopped		Stopped	
Watch prescaler	Operating*3		Operating		Operating*3		Operating		Operating*3		Operating		Operating*3	Stopped
External interrupt	Operating		Operating		Operating		Operating		Operating		Operating		Operating	
Hardware watchdog timer	Operating		Operating		Operating		Operating		Operating		Operating		Operating	
Software watchdog timer	Operating		Operating		Stopped		Stopped		Stopped		Stopped		Stopped	
Low-voltage detection reset	Operating		Operating		Operating		Operating		Operating		Operating		Operating	
Other peripheral resources	Operating		Operating		Operating		Operating		Stopped		Stopped		Stopped	

注意：在切换至时钟模式后设置待机模式。

6.4 休眠模式

6.4.1 介绍

休眠模式停止 CPU 以及软件看门狗定时器的操作。

6.4.2 休眠模式下的操作

休眠模式停止 CPU 以及软件看门狗定时器的操作时钟。在该模式下，CPU 停止工作，同时保留寄存器以及在切换至休眠模式前 RAM 中的内容，但是除看门狗定时器外，其他外围资源继续操作。

6.4.3 切换与退出

烧写“1”至待机控制寄存器（STBC: SLP）的休眠位使设备进入休眠模式。

以下代码是从运行模式到休眠模式的 STBC 设置。

```
STBC_SLP = 1;        // Causes transition to sleep mode
```

注意：上述代码截取自工程 Mode_Change，参见附录。

重置或来自外围资源的中断将从休眠模式中释放设备。

6.5 停止模式

6.5.1 介绍

停止模式停止主时钟。

6.5.2 停止模式下的操作

停止模式停止主时钟，主 CR 时钟和副时钟。在该模式下，设备停止除外部中断和低电压间检测重置外的所有的功能，同时保留寄存器以及在切换至停止模式前 RAM 中的内容。

6.5.3 切换和退出

烧写“1”至待机控制寄存器（STBC: STP）的停止位使设备进入停止模式。此时，如果待机控制寄存器（STBC: SPL）的引脚状态设置位为“0”，保持外部引脚状态。如果该位为“1”，外部引脚的状态变为高阻抗。

以下代码显示了如何切换至停止模式：

```
STBC_STP = 1;         //Causes transition to stop mode
```

注意：上述代码截取自工程 Mode_Change，参见附录。

重置或外部中断将从停止模式中释放设备。从停止模式中恢复后，如果有必要，初始化每个外围资源。

6.6 时基定时器模式

6.6.1 介绍

时基定时器模式只允许主时钟振荡，副时钟振荡，时基定时器以及计时预分频器工作。CPU 和外围资源的操作时钟在该模式下停止。

6.6.2 时基定时器模式下的操作

在时基定时器模式下，除时基定时器外，主时钟供给停止。设备停止除时基定时器，外部中断和低电压检测重置外的所有功能，同时保留寄存器以及在切换至时基定时器模式前 RAM 中的内容。

6.6.3 切换和退出

切换至时基定时器模式：SYCC2_RCS 设置为 0x02 或 0x03，烧写“1”至待机控制寄存器（STBC: TMD）的计时位。

以下代码显示了如何切换至时基定时器模式：

```
SYCC2_RCS0 = 0x00;
SYCC2_RCS1 = 0x01;
STBC_TMD = 1;        // Causes the device to enter timebase timer mode
```

注意：上述代码截取自工程 Mode_Change，参见附录。

退出时基定时器模式：重置，时基定时器中断或外部中断将从时基定时器模式中释放设备。从时基定时器模式中恢复后，如果有必要，初始化每个外围资源。

6.7 计时模式

6.7.1 介绍

计时模式只允许副时钟，副 CR 时钟和计时预分频器工作。CPU 和外围资源的操作时钟在该模式下停止。

6.7.2 计时模式下的操作

在计时模式下，CPU 和外围资源的的操作时钟停止工作。设备停止除计时预分频器外的所有功能。低电压检测重置，同时保留寄存器以及在切换至计时模式前 RAM 中的内容。

6.7.3 切换和退出

切换至计时模式：设置 SYCC2_RCS 为 0x00 或 0x01，并烧写“1”至待机控制寄存器（STBC: TMD）的计时位。

只有在时钟模式为副时钟模式或副 CR 时钟模式时，设备才能进入计时模式。

```
SYCC2_RCS0 = Ox00;
SYCC2_RCS1 = 0x00;
STBC_TMD = 1;        // Causes the device to enter watch mode
```

注意：上述代码截取自工程 Mode_Change，参见附录。

退出计时模式：重置，计时中断或外部中断将从计时模式中释放设备。从计时模式中恢复后，如果有必要，初始化每个外围资源。

6.8 待机模式切换代码

以下代码可实现待机模式间的切换，如停止模式，计时模式等，可更新至任何想要的模式。

```
//The following code can realize the transition of the standby mode from
//one to another mode, such as a stop mode, a watch mode
//The following software is for demonstration purpose only
#define switchmode  normal //set transition to some standby modes
// Can define the switchmode to Stop, Sleep, Watch or Timebase
voidmain (void)
{
      MCU_initialization();
      while(1)
    {  // choice standby mode what you want transition to
      switch (switchmode)
        {
            case normal: break;
            case stop:    STBC_STP = 1;
                          break;           //normal to stop mode
            case sleep:   STBC_SLP = 1;
                          break;           //normal to sleep mode
            case watch:   SYCC2_SOSCE = 1; //Enable Sub Clock
                      SYCC2_RCS0 = 0x01;  //Select Sub Clock
                      SYCC2_RCS1 = 0x00;  //Select Sub Clock
                   WATR = 0xF0;           //About 1.00s Wait Time
                      while (!SYCC_SRDY); //Wait Clock stable
                      STBC_TMD = 1;
                         break;           //normal to watch mode
            case Timebase:   SYCC2_MOSCE = 1;    //Enable Main Clock
                      SYCC2_RCS0 = 0x01;  //Select Main Clock
                      SYCC2_RCS1 = 0x01;
                   WATR = 0X0F;           // About 4.10ms Wait Time
                      while (!STBC_MRDY); //Wait Clock stable
                      STBC_TMD = 1;
                      break;              //normal to timebase time mode
      default:        break;
        }
      vDelay(2000);                   //delay the time
      led_display();                  //lighten three LED
      }
}
```

注意：上述代码截取自工程 Mode_Change，参见附录。

7 更多信息

如欲了解有关 Cypress 微控制器产品的更多详情，敬请访问以下网址：

http://www.cypress.com/cypress-microcontrollers

A 附录

A.1 代码范例

A.1.1 工程 1: Basic_Initialization

NAME: Start.asm

FUNCTION: Initialization MCU

```
;===============================================================================
; F2MC-8FX Family SOFTUNE C Compiler sample startup routine,
;===============================================================================
; Sample code for initialization
;-------------------------------------------------------------------
.PROGRAM  start
        .TITLE      start
;-------------------------------------------------------------------
; variable define declaration
; #define HWD_DISABLE
; if define this, Hard Watchdog will disable.
; external declaration of symbols
;-------------------------------------------------------------------
        .EXPORT __start
          .IMPORT      _main
          .IMPORT      LMEMTOMEM
          .IMPORT      LMEMCLEAR
          .IMPORT      _RAM_INIT
          .IMPORT      _ROM_INIT
          .IMPORT      _RAM_DIRINIT
          .IMPORT      _ROM_DIRINIT
;-------------------------------------------------------------------
; definition to stack area
;-------------------------------------------------------------------
           .SECTION    STACK, STACK, ALIGN=1
           .RES.B   128-2
STACK_TOP:
        .RES.B     2
;-------------------------------------------------------------------
; definition to start address of data, const and code section
;-------------------------------------------------------------------
        .SECTION DIRDATA,    DIR,          ALIGN=1
        .SECTION DIRINIT,    DIR,          ALIGN=1
        .SECTION DATA,   DATA,    ALIGN=1
        .SECTION INIT,     DATA,    ALIGN=1
;-------------------------------------------------------------------
; The Mode Byte is defined at the beginning of the start.asm
;-------------------------------------------------------------------
           .SECTION    RESVECT,    CONST, LOCATE=H'FFFD
     .DATA.B0
     .DATA.W      __start
;-------------------------------------------------------------------
//code area
;-------------------------------------------------------------------
           .SECTION    CODE,   CODE,   ALIGN=1
__start:
;-------------------------------------------------------------------
; set stack pointer
;-------------------------------------------------------------------
        MOVW          A,    #STACK_TOP
```

```
        MOVW    SP, A
;---------------------------------------------------------------------
; Set Register bank Pointer 0
;---------------------------------------------------------------------
        MOVW    A,  PS
        MOVW    A,  #0x07FF// RP=0, DP=0, I=0
        ANDW    A
        MOVW    PS, A
;---------------------------------------------------------------------
; Set ILM to the lowest level (3)
;---------------------------------------------------------------------
        MOVW    A,  PS
        MOVW    A,  #0x0030
        ORW         A
        MOVW    PS, A
；change the main CR clock frequency
        MOV     A ,     0x0FE4      ;READ CRTH for NVR trimming value protect
        AND     A ,  #0x9F          ;Just CLEAR CRTH [6:5]
        OR      A ,  #0x60          ; Main CR clock update to 8MHz
；Can update the Main CR clock as below setting
；0x00: 1MHz;  0x40: 10MHz;   0x60: 8MHz
        MOV     0x0FE4 , A          ； WRITE CRTH to enable the update
;---------------------------------------------------------------------
; copy initial value *CONST (ROM) section to *INIT(RAM) section
;---------------------------------------------------------------------
#macro  ICOPY   src_addr,   dest_addr,  src_section
        MOVW    EP, #\src_addr
        MOVW    A,  #\dest_addr
        MOVW    A,  #SIZEOF(\src_section)
        CALL    LMEMTOMEM
#endm
        ICOPY   _ROM_INIT, _RAM_INIT, INIT
        ICOPY   _ROM_DIRINIT, _RAM_DIRINIT, DIRINIT
;---------------------------------------------------------------------
; zero clear of *VAR section
;---------------------------------------------------------------------
#macro  FILL0   src_section
        MOVW    A,  #\src_section
        MOVW    A,  #SIZEOF (\src_section)
        CALL    LMEMCLEAR
#endm
        FILL0   DIRDATA
        FILL0   DATA
;---------------------------------------------------------------------
; call main routine
;---------------------------------------------------------------------
        CALL    _main
End:    JMP         end
;---------------------------------------------------------------------
; Hard Watchdog
;---------------------------------------------------------------------
#ifdef  HWD_DISABLE
        .SECTION WDT,   CONST, LOCATE=H'FFBE
        .DATA.W  0xA596
#endif
;---------------------------------------------------------------------
; reset vector
;---------------------------------------------------------------------
        .SECTION RESET, CONST, LOCATE=0xFFFC
```

```
        .DATA.B 0xFF
        .DATA.B 0
        .DATA.H __start
        .END    __start
```

NAME: vector.c

FUNCTION: Interrupt level (priority) setting and Interrupt vector definition

```
/*****************************************************************************
NAME:   vector.c
FUNCTION:  Interrupt level (priority) setting
        Interrupt vector definition
*****************************************************************************/
#include"mb95200.h"
;---------------------------------------------------------------------
//This function pre-sets all interrupt control registers. It can be used
//to set all interrupt priorities in static applications. If this file
// contains assignments to dedicated resources, verify that the appropriate
//controller is used.
//---------------------------------------------------------------------
voidInitIrqLevels (void)
{
        ILR0 = 0xFC;        // IRQ0:  external interrupt ch.4       --> Level 00
                            // IRQ1:  external interrupt ch.5       --> 01
                            // IRQ2:  external interrupt ch.2/ch.6 --> 01
                            // IRQ3:  external interrupt ch.3/ ch.7      --> 01
        ILR1 = 0xFF;        // IRQ4:  UART/SIO ch.0
                            // IRQ5:  8/16-bit timer ch.0 (lower)
                            // IRQ6:  8/16-bit timer ch.0 (upper)
                            // IRQ7:  LIN-UART (reception)
        ILR2 = 0xFF;        // IRQ8:  LIN-UART (transmission)
                            // IRQ9:  8/16-bit PPG ch.1 (lower) / UART/SIO ch.1
                            // IRQ10: 8/16-bit PPG ch.1 (upper) / I2C ch.1
                            // IRQ11: 16-bit reload timer ch.0
        ILR3 = 0xFF;        // IRQ12: 8/16-bit PPG ch.0 (upper)
                            // IRQ13: 8/16-bit PPG ch.0 (lower)
                            // IRQ14: 8/16-bit timer ch1 (upper)
                            // IRQ15: 16-bit PPG ch.0 + ch.2
        ILR4 = 0xFF;        // IRQ16: 16-bit reload timer ch.1 / I2C ch.0
                            // IRQ17: 16-bit PPG ch.1
                            // IRQ18: 10-biat A/D-converter
                            // IRQ19: Timebase timer
        ILR5 = 0xFF;        // IRQ20: Watch timer / counter
                            // IRQ21: external interrupt ch 8-11
                            // IRQ22: 8/16-bit timer ch1 (lower) / external
                            //interrupt ch.12-15
                            // IRQ23: Flash | Custom ch.1
}
//---------------------------------------------------------------------
//   Prototypes
// Add your own prototypes here. Each vector definition needed is a
//prototype. Either do it here or include a header file containing them.
//---------------------------------------------------------------------
//extern unsigned int delay_timer;
__interrupt void DefaultIRQHandler (void);
__interrupt void Key_int (void);
```

```
//-----------------------------------------------------------------------
//Vector definition
//Use the following statements to define vectors.
//All resources related to the vectors are predefined.
//Remaining software interrupts can be added here as well.
//-----------------------------------------------------------------------
#pragma intvect Key_int 0                  //IRQ0: external interrupt ch.4
#pragma intvect DefaultIRQHandler 1        //IRQ1: external interrupt ch.5
#pragma intvect DefaultIRQHandler 2        //IRQ2: external interrupt ch.2|ch.6
#pragma intvect DefaultIRQHandler 3        //IRQ3: external interrupt ch.3|ch.7
#pragma intvect DefaultIRQHandler 5        //IRQ5: 8/16-bit timer ch.0 (lower)
#pragma intvect DefaultIRQHandler 6        //IRQ6: 8/16-bit timer ch.0 (upper)
#pragma intvect DefaultIRQHandler 7        //IRQ7: LIN-UART (reception)
#pragma intvect DefaultIRQHandler 8        //IRQ8: LIN-UART (transmission)
#pragma intvect DefaultIRQHandler 14       //IRQ14: 8/16-bit timer ch1 (upper)
#pragma intvect DefaultIRQHandler 18       //IRQ18: 10-biat A/D-converter
#pragma intvect DefaultIRQHandler 19       //IRQ19: Timebase timer
#pragma intvect DefaultIRQHandler 20       //IRQ20:Watchtimer/ counter
#pragma intvect DefaultIRQHandler 22               //IRQ22: 8/16-bit timer ch.1
                                           //(lower)/interrupt ch.12-15
#pragma intvect DefaultIRQHandler 23       //IRQ23: Flash / Custom ch.1
__interrupt void DefaultIRQHandler (void)
{
__DI ();                                   // disable interrupts
        While (1)
        __wait_nop ();                     // halt the system
}
```

NAME: Main.c

Function: external transition interrupt as key input

```
/*****************************************************************************
NAME:         MAIN.C
FUNCTION:  The following code can realize initialization the MCU
           The following software is for demonstration purpose only
*****************************************************************************/
#include"mb95200.h"
#define switchmode stop    //set transition to some standby modes

void vSysInit (void)
{
        //elide code
        InitIrqLevels ();
        __EI ();
}
__interrupt void Key_int (void)
{
        //elide key functions
}
        //include vSysInit(),__interrupt and other functions
voidmain (void)
{
        vSysInit ();
        while (1)
        {
          //enter other test codes
        }
}
```

A.1.2 工程 2: Clock_Change

```
NAME:       Main.c
Function:   choose the clock, clock mode, clock divide ratio
```

```
/*****************************************************************************
//The following code can realize choose the clock and clock mode
//The following software is for demonstration purposes only
/*****************************************************************************
NAME:       MAIN.C
FUNCTION:  Change the system clock and lighten three LED
           LED flicker frequency different in different clock
*****************************************************************************/
#include"mb95200.h"
#define MAIN            0x00
#define SUB         0x01
#define MAIN_CR         0x02
#define SUB_CR      0x03
unsigned char switchclock = MAIN;       //select the start clock
unsigned char toggle_status = 0;        //LED change bit
unsigned char i;
/*****************************************************************************
NAME:       MCU initialization
FUNCTION: Initialization the IO port, system clock, interrupt level
*****************************************************************************/
void MCU_initialization()
{
    __DI();
    SYSC = 0x03;
  /*IO port*/
    PDR0_P05=1;
    DDR0_P05=1;             //Enable output
  PDR6_P63=1;
  PDR6_P64=1;
  DDR6_P63=1;               //Enable output
  DDR6_P64=1;               //Enable output

  /*external interrupt*/
    EIC30=0x55;             //INT06 INT07 enable falling edge
    //source clock can be main clock divided by 2, sub-clock divided by 2
      //sub-CR clock divided by 2 or main CR clock no div
      SYCC_DIV = 0x00;          // Source clock (No division)
                                // SYCC_DIV = 0x01 mean Source clock/4
                                // SYCC_DIV = 0x02 mean Source clock/8
                                // SYCC_DIV = 0x03 mean Source clock/16
    /* initialise Interrupt level register and IRQ vector table*/
    InitIrqLevels();
  __EI();
}
/*****************************************************************************
NAME:       led_display()
FUNCTION:  Set three LED light cycle one by one
*****************************************************************************/
void led_display()
{
    switch(toggle_status)
  {
    case 0:                 //lighten the PDR0_P05 (LED2)
      {
```

```
            PDR0_P05=0;
                PDR6_P64=1;
                PDR6_P63=1;
                toggle_status=1;
                break;}
            case 1:                 //lighten the PDR6_P64 (LED3)
            {
                PDR0_P05=1;
                PDR6_P64=0;
                PDR6_P63=1;
                toggle_status=2;
                break;}
            case 2:                 //lighten the PDR6_P63 (LED4)
            {
                PDR0_P05=1;
                PDR6_P64=1;
                PDR6_P63=0;
                toggle_status=0;
                break;}
        }
    }
```

```
/*****************************************************************************
NAME:         vDelay
FUNCTION: Delay the fix time
*****************************************************************************/
void vDelay (unsigned int uiCount)
{
  while(uiCount--)
  {
    asm ("\tNOP");
  }
}
/*****************************************************************************
NAME:         __interrupt void external_int06(void)
FUNCTION:  Change the clock
*****************************************************************************/
__interrupt void external_int06 (void)
{
        EIC30_EIR0=0;
      switch (++switchclock)
        {
          case MAIN:
                    // enable the Main clock oscillation
                    SYCC2_MOSCE = 1;
                    // Clock Mode Selection, select the main clock mode
                    SYCC2_RCS0 = 0x01;
                    SYCC2_RCS1 = 0x01;
                    //update Oscillation Stabilization Wait Time
                    //when Main Oscillation Clock FCH=4MHz
                    WATR = 0X0F;
                    // Oscillation Stabilization Wait Time mean
                    // (2^14-2)/F_CH About 4.10 ms
                    while (!STBC_MRDY);
                    break;
          case SUB:
                    // enable the sub-clock oscillation
                    SYCC2_SOSCE = 1;
                    // Clock Mode Selection, select the sub-clock mode
                    SYCC2_RCS0 = 0x01;
                    SYCC2_RCS1 = 0x00;
                    //Update the Oscillation Stabilization Wait Time
                    // when sub Oscillation Clock FCL=32.768 kHz
                    WATR = 0xF0;     //(215-2)/FCL About 1.00s
                    //Indicates sub-clock oscillation being stable
                    while (!SYCC_SRDY);
                    break;
          case MAIN_CR:
                    // enable the main CR clock oscillation
                    SYCC2_MCRE = 1;
                    // Clock Mode Selection, select the main CR clock mode
                    SYCC2_RCS0 = 0x00;
                    SYCC2_RCS1 = 0x01;
                    //update the Oscillation Stabilization Wait Time
                    //F_CRHS represents the main clock frequency
                    //2^8/F_CRHS = Main CR oscillation Stabilization Wait Time
                    //Indicates main CR clock oscillation being stable
                    while (!STBC_MCRDY);
                    break;
          case SUB_CR:
                    // enable the Sub-CR Clock oscillation
```

```
                SYCC2_SCRE = 1;
                // Clock Mode Selection, select the Sub-CR Clock mode
                SYCC2_RCS = 0x00;
                //update the Oscillation Stabilization Wait Time
                //FCRHS represents the main clock frequency
                //2^4/FCRL = Sub-CR oscillation Stabilization Wait Time
                //Indicates sub-CR clock oscillation being stable
                while (!STBC_SCRDY);
                break;
        default:
            switchclock = MAIN;
              SYCC2=0x34;
                //update Oscillation Stabilization Wait Time
                WATR = 0X03;
             //Indicates main-clock oscillation being stable
                while (!STBC_MRDY);
                break;
    }
}
/*****************************************************************************
NAME:         main ()
FUNCTION: lighten three LED
*****************************************************************************/
void main()
{
        MCU_initialization();
        while(1)
    {
        vDelay(10);           //delay the time
          led_display();      //lighten three LED
    }
}
```

```
NAME:         vector.c
FUNCTION:  Interrupt level setting and interrupt vector definition
```

```
/*****************************************************************************
NAME:         vector.c
FUNCTION:  Interrupt level (priority) setting
                  Interrupt vector definition
*****************************************************************************/
#include"mb95200.h"
//---------------------------------------------------------------------------
//This function pre-sets all interrupt control registers. It can be used
//to set all interrupt priorities in static applications. If this file
//contains assignments to dedicated resources, verify that the appropriate //controller is used.
//---------------------------------------------------------------------------
voidInitIrqLevels (void)
{
        ILR0 = 0x1F;          // IRQ0:  external interrupt ch.4
                                    // IRQ1:  external interrupt ch.5
                                    // IRQ2:  external interrupt ch.2/ch.6
          // IRQ3:  external interrupt ch.3/ch.7
   //... ...
}
//---------------------------------------------------------------------------
//     Prototypes
//Add your own prototypes here. Each vector definition needed is a
//prototype. Either do it here or include a header file containing them.
//---------------------------------------------------------------------------
__interrupt void DefaultIRQHandler (void);
__interrupt void external_int06 (void);
//---------------------------------------------------------------------------
//Vector definition
//Use the following statements to define vectors.
//All resources related to the vectors are predefined.
//Remaining software interrupts can be added here as well.
//---------------------------------------------------------------------------
#pragma intvect DefaultIRQHandler 0     //IRQ0: external interrupt ch.4
#pragma intvect DefaultIRQHandler 1     //IRQ1: external interrupt ch.5
#pragma intvect external_int06 2          //IRQ2: external interrupt ch.2|ch.6
//... ...
__interrupt void DefaultIRQHandler (void)
{
        __DI ();                                // disable interrupts
        while (1)
        __wait_nop ();                  // halt the system
}
```

A.1.3 工程 3: Mode_Change

```
 NAME:   Main.c
 Function:      standby mode change
/*****************************************************************************
//The following code can realization transition from normal mode to other
//mode, such as stop mode, watch mode
//The following software is for demonstration purposes only
//You can update the switchmode to the mode what you want
/*****************************************************************************
NAME:         MAIN.C
FUNCTION:       Change the mode and lighten three LED to one fix light
                Change the mode and LED light fixed
*****************************************************************************/
#include"mb95200.h"
#define normal          0x00
#define stop            0x01
#define sleep           0x02
#define Timebase0x03
#define watch                   0x04
unsigned char switchmode = normal;          //Select the start mode
unsigned char toggle_status = 0;            //LED change bit
unsigned char i;
/*****************************************************************************
NAME:         MCU initialization
FUNCTION: Initialization the IO port, system clock, interrupt level
*****************************************************************************/
void MCU_initialization()
{
        __DI();
    /*system clock*/
    SYSC = 0x03;
    SYCC2 = 0x34;
    /*IO port*/
    PDR0_P05=1;
    DDR0_P05=1;                         //Enable output
    PDR6_P63=1;
    PDR6_P64=1;
    DDR6_P63=1;                         //Enable output
    DDR6_P64=1;                         //Enable output
    /*external interrupt*/
    EIC30=0x55;                     //INT06 enable falling edge
    /*initialise Interrupt level register and IRQ vector table*/
    InitIrqLevels();
    __EI();
}
/*****************************************************************************
NAME:         led_display()
FUNCTION:       Set three LED light cycle one by one
*****************************************************************************/
void led_display()
{
        switch(toggle_status)
    {
          case 0:                       //Lighten the PDR0_P05 (LED2)
          {
              PDR0_P05=0;
              PDR6_P64=1;
              PDR6_P63=1;
```

```
                toggle_status=1;
                break;
            }
            case 1:                     //Lighten the PDR6_P64 (LED3)
            {
                PDR0_P05=1;
                PDR6_P64=0;
                PDR6_P63=1;
                toggle_status=2;
                break;
            }
            case 2:                     //Lighten the PDR6_P63 (LED4)
            {
                PDR0_P05=1;
                PDR6_P64=1;
                PDR6_P63=0;
                toggle_status=0;
                break;
            }
        }
}

/*****************************************************************************
NAME:       vDelay
FUNCTION:       Delay the fix time
*****************************************************************************/
void vDelay (unsigned int uiCount)
{
        while(uiCount--)
        {
        asm ("\tNOP");
        }
}
/*****************************************************************************
NAME:       __interrupt void external_int06(void)
FUNCTION:       Change the mode
*****************************************************************************/
__interrupt void external_int06(void)
{
        EIC30_EIR0=0;
        if(switchmode <= 3)
                switchmode++;
          if(switchmode >= 4)
                switchmode = 1;
}
/*****************************************************************************
NAME:       __interrupt void external_int07(void)
FUNCTION:       Reset the mode to normal
*****************************************************************************/
__interrupt void external_int07(void)
{
        EIC30_EIR1 = 0;
        switchmode = normal;
}
```

```
/*****************************************************************************
NAME:         main ()
FUNCTION:  Lighten three LED, when change the mode, only one LED light
*****************************************************************************/
void main()
{
        MCU_initialization();
     while(1)
     {
        switch (switchmode)
          {
        case normal:    break;
                case stop:     STBC_STP = 1;//Causes transition to stop mode
                               break;                          //Normal to stop mode
                case sleep:    STBC_SLP = 1;//Causestransition to sleepmode
                               break;                          //Normal to sleep mode
                case watch:    SYCC2_SOSCE = 1;  //Enable Sub Clock
                                    SYCC2_RCS0 = 0x01;       //Select Sub Clock
                                          SYCC2_RCS1 = 0x00;
                                    WATR = 0xF0;             //About 1.00s Wait Time
                               while (!SYCC_SRDY);//Wait Clock stable
                               STBC_TMD = 1;
                                    //Causes the device to enter watch mode
                               break;                  //normal to watch mode
                case Timebase:SYCC2_MOSCE = 1;     //Enable Main Clock
                               SYCC2_RCS0 = 0x01;       //Select Main Clock
                               SYCC2_RCS1 = 0x01;
                                    WATR = 0X0F;          // About 4.10 ms Wait Time
                               while (!STBC_MRDY);//Wait Clock stable
                               STBC_TMD = 1;
                          //Causes the device to enter timebase timer mode
                               break;          //normal to timebase time mode
                default:
                               break;
          }
          vDelay(2000);                          //delay the time
          led_display();                         //lighten three LED
     }
}
```

```
 NAME:         vector.c
 FUNCTION:  Interrupt level setting and interrupt vector definition
/*****************************************************************************
NAME:   vector.c
FUNCTION:  Interrupt level (priority) setting
                Interrupt vector definition
*****************************************************************************/
#include"mb95200.h"
//---------------------------------------------------------------------
//This function pre-sets all interrupt control registers. It can be used
//to set all interrupt priorities in static applications. If this file
//contains assignments to dedicated resources, verify that the appropriate //controller is used.
//---------------------------------------------------------------------
voidInitIrqLevels (void)
{
        ILR0 = 0x1F;          // IRQ0:  external interrupt ch.4
                                    // IRQ1:  external interrupt ch.5
                                    // IRQ2:  external interrupt ch.2/ch.6
          // IRQ3:  external interrupt ch.3/ch.7
//... ...
}
//---------------------------------------------------------------------
//Prototypes
//Add your own prototypes here. Each vector definition needed is a
//prototype. Either do it here or include a header file containing them.
//---------------------------------------------------------------------
__interrupt void DefaultIRQHandler (void);
__interrupt void external_int06 (void);
__interrupt void external_int07 (void);
//---------------------------------------------------------------------
//Vector definition
//Use the following statements to define vectors.
//All resources related to the vectors are predefined.
//Remaining software interrupts can be added here as well.
//---------------------------------------------------------------------
#pragma intvect DefaultIRQHandler 0    //IRQ0: external interrupt ch.4
#pragma intvect DefaultIRQHandler 1    //IRQ1: external interrupt ch.5
#pragma intvect external_int06 2       //IRQ2: external interrupt ch.2/ch.6
#pragma intvect external_int07 3       //IRQ3: external interrupt ch.3/ch.7
//... ...
__interrupt void DefaultIRQHandler (void)
{
        __DI ();                       // disable interrupts
        while (1)
        __wait_nop ();                 // halt the system
}
-- END --
```

文档修改记录

文档标题：AN205271 - F²MC-8FX 家族 8 位微型控制器 MB95200H/210H 系列 基本固件设置

文档编号：002-05737

修订版	ECN	变更者	提交日期	变更说明
**	—	HUAL	03/26/2008	初稿
			07/21/2008	修改第五章时钟模式选择代码及注释
			02/26/2010	删除第 11，13，24 页中关于 12.5M CR 的描述
*A	5327450	HUAL	06/28/2016	已将 Spansion 应用手册《MCU-AN-500002-Z-12》转换成 Cypress 格式。

全球销售和设计支持

赛普拉斯公司拥有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。如果想要查找离您最近的办事处，请访问赛普拉斯所在地。

产品

ARM® Cortex® 微控制器	cypress.com/arm
汽车级	cypress.com/automotive
时钟与缓冲器	cypress.com/clocks
接口	cypress.com/interface
照明和电源控制	cypress.com/powerpsoc
存储器	cypress.com/memory
PSoC	cypress.com/psoc
触摸感应	cypress.com/touch
USB 控制器	cypress.com/usb
无线/射频	cypress.com/wireless

PSoC® 解决方案

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

赛普拉斯开发者社区

论坛 | 项目 | 视频 | 博客 | 培训 | 组件

技术支持

cypress.com/support

PSoC 是赛普拉斯半导体公司的注册商标。PSoC Creator 是赛普拉斯半导体公司的商标。 此处引用的所有其他商标或注册商标都归其各自所有者所有。

赛普拉斯半导体
198 Champion Court
San Jose, CA 95134-1709

电话：408-943-2600
传真：408-943-4730
网站地址 ：www.cypress.com

©赛普拉斯半导体公司，2008-2016 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属个人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码的形式向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。
在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。在适用法律允许的限度内，赛普拉斯保留更改本文件的权利，届时将不另行通知。赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失的其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何索赔、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的索赔，并使之免受损失。
赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。敬请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。