AN205396

F²MC-8FX Family MB95200 Series 8-Bit Microcontroller Basic API

Associated Part Family: MB95200 Series

This document introduces API for Clock, CSV, Standby mode, Wilder register, Interrupt and External Interrupt.

Contents

1 Introduction....1
2 Clock Controller....1
2.1 Clock Controller Library Function List....1
2.2 Clock Library Function Detail....2
2.3 Usage Demo....5
3 Clock Supervisor Counter....6
3.1 Clock Supervisor Counter Function List....6
3.2 Clock Supervisor Counter Function Detail....6
3.3 Usage Demo....7
4 Standby Mode....8
4.1 Standby Mode Library Function List....8
4.2 Standby Mode Function Detail....8
4.3 Usage Demo....10
5 Wild Register Function....11
5.1 Wild Register Library Function List....11
5.2 Wild Register Function Detail....11
5.3 Usage Demo....13
6 Interrupts....15
6.1 Interrupts Library Function List....15
6.2 Interrupts Function Detail....16
6.3 Usage Demo....21
7 External Interrupt....23
7.1 External Interrupt Library Function List....23
7.2 External Interrupt Function Detail....23
7.3 Usage Demo....24
8 Additional Information....25
Document History....26

1 Introduction

This document introduces API for Clock, CSV, Standby mode, Wilder register, Interrupt and External Interrupt.

2 Clock Controller

The clock library has five functions to realize control clock, including setting clock, getting clock, dividing status, setting clock divide and setting oscillator stabilization time.

2.1 Clock Controller Library Function List

Table 1 lists the Clock library functions.

Table 1. Clock Library Functions

Function name	Description
void Clock_Set_CPU_Clock(uint CLOCK)	Set the machine clock
uchar Clock_Get_CPU_Clock_status(void)	Get the current clock status
void Clock_Set_Divide(uchar Clock_Div)	Set the machine clock divide
uchar Clock_Get_Divide_Status(void)	Get the current clock divide status
void Clock_Stabilization_Time_Set(uint CLOCK_TIME)	Set clock stabilization time

2.2 Clock Library Function Detail

2.2.1 Clock_Set_CPU_Clock Function

Table 2 describes Clock_Set_CPU_Clock function.

Table 2. Clock_Set_CPU_Clock Function

Function name	Clock_Set_CPU_Clock
Function prototype	void Clock_Set_CPU_Clock(uint CLOCK)
Behavior description	Set the machine clock select parameter ‘clock’ to correct value and will set the clock as MCU run clock.
Input parameter	CLOCK
Return value	None

For example:

If you want select main clock as MCU run clock, you can set the function: Clock_Set_CPU_Clock(Main_Clock) and succeed.

Table 3 describes the Clock parameter values.

Table 3. Clock Definition

Clock	Description
Main_Clock	Set main clock as machine clock
Sub_Clock	Set sub clock as machine clock
Main_CR_Clock	Set main CR clock as machine clock
Sub_CR_Clock	Set sub clock as machine clock

2.2.2 Clock_Get_CPU_Clock_Status Function

Table 4 describes Clock_Get_CPU_Clock_Status function.

Table 4. Clock_Get_CPU_Clock_Status Function

Function name	Clock_Get_CPU_Clock_Status
Function prototype	uchar Clock_Get_CPU_Clock_status(void)
Behavior description	Get the run clock
Input parameter	None
Return value	Return value has four virtual values: 0x00(Sub_CR_clock), 0x01(Sub_clock), 0x02(main_CR_clock), 0x03(main_clock).

2.2.3 Clock_Set_Divide Function

Table 5 describes Clock_Set_Divide function.

Table 5. Clock_Set_Divide Function

Function name	Clock_Set_Divide
Function prototype	void Clock_Set_Divide(uchar Clock_Div)
Behavior description	Set the machine clock divide
Input parameter	Clock_Div
Return value	None

This function can set the clock divide. Select parameter 'Clock_Div' to correct value and will set the clock divide.

Table 6 describes the Clock_Div parameter values.

Table 6. Clock_Div Function

Clock_Div	Description
Clock_Div_No	Set clock divide no as machine clock
Clock_Div_4	Set clock divide 4 as machine clock
Clock_Div_8	Set clock 8 as machine clock
Clock_Div_16	Set clock 16 as machine clock

2.2.4 Clock_Get_Divide_Status Function

Table 7 describes Clock_Get_Divide_Status function.

Table 7. Clock_Get_Divide_Status Function

Function name	Clock_Get_Divide_Status
Function prototype	uchar Clock_Get_Divide_Status(void)
Behavior description	get the clock divide status
Input parameter	None
Return value	Return clock divide status, refer to table 2-6

The function can get the clock divide status. Select parameter 'clock' to correct value and will get the clock divide status. The return value has four virtual values: Clock_Div_No, Clock_Div_4, Clock_Div_8 and Clock_Div_16.

2.2.5 Oscl_Stabilization_Time_Set Function

Table 8 describes Oscl_Stabilization_Time_Set function.

Table 8. Oscl_Clock_Stabilization_Time Function

Function name	Oscl_Stabilization_Time_Set
Function prototype	void Clock_Stabilization_Time_Set(uint CLOCK_TIME)
Behavior description	Set clock stabilization wait time
Input parameter	CLOCK_TIME
Return value	None

The function can set the clock stabilization wait time. Select parameter CLOCK_TIME to correct value and will set the clock stabilization wait time.

Table 9 describes CLOCK_TIME parameter definition.

Table 9. CLOCK_TIME Definition

CLOCK_TIME	Description
FCH_14_DIV	(214-2)/FCH
FCH_13_DIV	(213-2)/FCH
FCH_12_DIV	(212-2)/FCH
FCH_11_DIV	(211-2)/FCH
FCH_10_DIV	(210-2)/FCH
FCH_9_DIV	(29-2)/FCH
FCH_8_DIV	(28-2)/FCH
FCH_7_DIV	(27-2)/FCH
FCH_6_DIV	(26-2)/FCH
FCH_5_DIV	(25-2)/FCH
FCH_4_DIV	(24-2)/FCH
FCH_3_DIV	(23-2)/FCH
FCH_2_DIV	(22-2)/FCH
FCH_1_DIV	(21-2)/FCH
FCL _15_DIV	(215-2)/FCL
FCL_14_DIV	(214-2)/FCL
FCL_13_DIV	(213-2)/FCL
FCL_12_DIV	(212-2)/FCL
FCL_11_DIV	(211-2)/FCL
FCL_10_DIV	(210-2)/FCL
FCL_9_DIV	(29-2)/FCL
FCL_8_DIV	(28-2)/FCL
FCL_7_DIV	(27-2)/FCL
FCL_6_DIV	(26-2)/FCL
FCL_5_DIV	(25-2)/FCL
FCL_4_DIV	(24-2)/FCL
FCL_3_DIV	(23-2)/FCL
FCL_2_DIV	(22-2)/FCL
FCL_1_DIV	(21-2)/FCL

2.3 Usage Demo

2.3.1 Clock Control

Control clock includes setting stabilization time, setting clock divide and setting clock mode. Select clock has four values: main clock, sub clock, main CR clock, sub CR clock. The clock stabilization time is changed by the CLOCK_TIME. First set the clock stabilization time, second select what clock use as source clock and last set the clock divide.

For example, if you want set the main clock, select main clock mode and set the clock to source/4, add the following function to you program.

```
        //the sample code to control main clock
void clock_control(void)
{
        Clock_Stabilization_Time_Set(FCH_14_DIV);       //set main clock stabilization time
        Clock_Set_CPU_Clock(Main_Clock);                // set main clock
        Clock_Set_Divide(Clock_Div_4);                  // set the clock to source/4
}
```

2.3.2 Get Clock Status

When you want to know the clock mode and clock divide status, you can use uchar Clock_Get_CPU_Clock_status(void) and uchar Clock_Get_Divide_Status(void) function.

The sample as:

```
Void get_clock_status(void)
{
        uchar state,state1;
        state = Clock_Get_CPU_Clock_status ();      //return run clock mode
        state1 = Clock_Get_Divide_Status ();        //return the clock divide status
}
```

3 Clock Supervisor Counter

This section introduces all functions of Clock library.

3.1 Clock Supervisor Counter Function List

Table 10 lists the Clock Supervisor Counter library functions.

Table 10. Clock Supervisor Counter Function

Function name	Description
uchar Clock_Supervisor_Counter (uchar Mod,uint EX_size)	Initialize the CSV and check the external clock frequency.

3.2 Clock Supervisor Counter Function Detail

3.2.1 Clock_Supervisor_Counter Function

Table 11 describes the Clock_Supervisor_Counter function.

Table 11. Clock_Supervisor_Counter Function

Function name	Clock_Supervisor_Counter
Function prototype	uchar Clock_Supervisor_Counter (uchar Mod,uint EX_size)
Behavior description	Initialize the CSV and check the external clock frequency
Input parameter1	Mod, Set the mode of check
Input parameter2	EX_size, get the size of external clock frequency For example: external clock frequency is 1 MHz, EX_size = 1000
Return value	0, initialization is failing 1, external clock frequency is correct 2. external clock frequency is not correct

This function use to check external clock frequency. You must startup main CR clock (SYCC2: MCRE=1, RCS=2) and enable main clock (SYSC: PFSEL=0; SYCC2: MOSCE=1) or sub clock (SYSC: PGSEL=0; SYCC2: SOSCE=1) before use this function. You can set the size of main CR clock (CRTH: CRSEL=0~3 <1 MHz, 10 MHz, 8 MHz>) to check external clock frequency.

Table 12 describes the Mod parameter values.

Table 12. Mod Definition

Mod	Description
Main_clock	Main clock
Sub_clock	Sub clock

3.3 Usage Demo

3.3.1 Start

Initialize the Clock Supervisor Counter, set the mode to check main clock or sub clock for your need, and input the external clock frequency to calculate the optimal interval time.

For example:

We should check the main clock, and the main clock frequency is 4 MHz, the operation as following:

Clock_Supervisor_Counter (main_clock, 4000) or Clock_Supervisor_Counter (0x00, 4000).

3.3.2 Get the Result of Check

Clock_Supervisor_Counter () function can return the result of check, the value of return is “0” or “1” or “2”, ”0” explain the parameter error, “1” explain the external clock frequency isn’t correct, “2” explain the external clock frequency is correct.

4 Standby Mode

This section introduces all functions of standby library.

In routine, the standby mode library has three functions: Standby_Mode_Enter (STANDBY_MODE), Mode_Pin_State(MODE_PIN_STATUS) and Software_Reset().This function can realize transition from normal mode to other mode, such as stop mode, watch mode.

4.1 Standby Mode Library Function List

Table 13 lists the Standby mode library functions.

Table 13. Standby Mode Functions

Function name	Description
void Standby_Mode_Enter(uchar STANDBY_MODE)	Select standby mode
void Mode_Pin_State(uchar MODE_PIN_STATUS)	High_Impedance / Hold_Preceding
void Software_Reset(void)	Generates a 3-machine clock reset signal

4.2 Standby Mode Function Detail

4.2.1 Standby_Mode_Enter Function

Table 14 describes Standby_Mode_Enter function.

Table 14. Standby_Mode_Enter Function

Function name	Standby_Mode_Enter
Function prototype	void Standby_Mode_Enter(uchar STANDBY_MODE)
Behavior description	realize transition from normal mode to other mode
Input parameter	STANDBY_MODE
Return value	None

Select parameter ‘STANDBY_MODE’ to correct value and it can realize transition from normal mode to other mode, such as stop mode, watch mode. For example, if you want MCU run in stop mode, you can set the function: Standby_Mode_Enter(Stop_Mode) and succeed.

Table 15 describes the STANDBY_MODE parameter values.

Table 15. STANDBY_MODE Definition

STANDBY_MODE	Description
Stop_Mode	Set standby mode to stop mode
Sleep_Mode	Set standby mode to sleep mode
Watch_mode	Set standby mode to watch mode
Timebase_timer_Mode	Set standby mode to Timebase timer mode

4.2.2 Mode_Pin_State Function

Table 16 describes Mode_Pin_State function.

Table 16. Mode_Pin_State function

Function name	Mode_Pin_State
Function prototype	void Mode_Pin_State(uchar MODE_PIN_STATUS)
Behavior description	realize setting the pin state after enter one standby mode
Input parameter	MODE_PIN_STATUS, Pin state can select high impedance and hold preceding status
Return value	None

Select parameter 'MODE_PIN_STATUS' to correct value and it can realize setting the pin state after enter one standby mode. Pin state can select high impedance and hold preceding status.

Table 17 describes the MODE_PIN_STATUS parameter values.

Table 17. MODE_PIN_STATUS Definition

MODE_PIN_STATUS	Description
High_Impedance	places external pins in a high impedance state in stop mode, timebase timer mode, or watch mode
Hold_Preceding	Holds external pins in their immediately preceding state in stop mode, timebase timer mode, or watch mode

4.2.3 Software_Reset Function

Table 18 describes Software_Reset function.

Table 18. Software_Reset Function

Function name	Software_Reset
Function prototype	void Software_Reset(void)
Behavior description	generates a 3-machine clock reset signal
Input parameter	None
Return value	None

The function can generates a 3-machine clock reset signal.

4.3 Usage Demo

4.3.1 Set Standby Mode and Software Reset

If you want to set the standby mode, just use Standby_Mode_Enter(STANDBY_MODE) function. Select parameter 'STANDBY_MODE' to correct value and it can realize transition from normal mode to other mode. Run software_reset(void) function can realize reset MCU.

The sample as:

```
        If (change_to_stop_mode)
{
        Mode_Pin_State(High_Impedance);   // places external pins in a high impedance
        Standby_Mode_Enter(Stop_Mode);    //set stop mode
}
```

5 Wild Register Function

This section introduces all functions of wild register library.

5.1 Wild Register Library Function List

Table 19 lists the wild register library functions.

Table 19. Wild Register Functions

Function name	Description
void Wild_Register_Initial(uint Channel, uint Address, uchar Data)	Select wild register operation channel, modified address and data
void Wild_Register_Read (uint Channel,uchar Status)	Enable or disable read wild register
void Wild_Register_Status(uint Channel,uchar Status)	Enable or disable wild register work

5.2 Wild Register Function Detail

5.2.1 Wild_Register_Init Function

Table 20 describes Wild_Register_Initial function.

Table 20. Wild_Register_Initial Function

Function name	Wild_Register_Initial
Function prototype	void Wild_Register_Initial(uint Channel,uint Address,uchar Data)
Behavior description	Select wild register operation channel, modified address and data
Input parameter1	Channel, Select operation wild register
Input parameter2	Address, Modified address
Input parameter3	Data, Modified data
Return value	None

The function select channel of the wild register, set modified address and modified data, and then enables corresponding wild register address compare enable register (WREN).

Table 21 describes the Channel parameter values.

Table 21. Channel Definition

Channel	Description
Wild_Channel_0	wild register 0
Wild_Channel_1	wild register 1
Wild_Channel_2	wild register 2

5.2.2 Wild_Register_Read

Table 22 describes Wild_Register_Read function.

Table 22. Wild_Register_Read Function

Function name	Wild_Register_Read
Function prototype	void Wild_Register_Read (uint Channel,uchar Status)
Behavior description	Enable or disable read wild register data setting register
Input parameter1	Channel, Select operation channel Refer to table5-3 for more details on the allowed values for this parameter.
Input parameter2	Status, Enable or disable read the wild register data setting register
Return value	None

Note: Input parameter Status values be relate to Channel parameter.

For example: when Channel is Wild_Channel_0, Status only has two input values: En_Register0 and Dis_Register0.

Table 23 describes the Status parameter values.

Table 23. Status Definition

Status	Description
En_Register0	WRDR0 enable read
Dis_Register0	WRDR0 disable read
En_Register1	WRDR1 enable read
Dis_Register1	WRDR1 disable read
En_Register2	WRDR2 enable read
Dis_Register2	WRDR2 disable read

5.2.3 Wild_Register_Status

Table 24 describes Wild_Register_Status function.

Table 24. Wild_Register_Status Function

Function name	Wild_Register_Status
Function prototype	void Wild_Register_Status(uint Channel,uchar Status)
Behavior description	Enable or disable wild register work
Input parameter1	Channel, Select operation channel Refer to table5-3 for more details on the allowed values for this parameter.
Input parameter2	Status, Enable or disable wild register operation. Refer to table5-5 for more details on the allowed values for this parameter.
Return value	None

Note: Input parameter Status values be relate to Channel parameter.

For example: when Channel is Wild_Channel_0, Status only has two input values: En_Register0 and Dis_Register0.

5.3 Usage Demo

5.3.1 Modified Data

Sample code:

```
void initialize (void)
{
    Wild_Register_Init (Wild_Channel_0, 0xC080, 0x66);
                    // 0xC080 is modified address, 0x66 is modified data
    Wild_Register_Init (Wild_Channel_1, 0xC081, 0x77);
                    //0x C081 is modified address, 0x77 is modified data
    Wild_Register_Init (Wild_Channel_1, 0xC082, 0x88);
                    // 0xC082 is modified address, 0x88 is modified data
    Wild_Register_Status (Wild_Channel_0, En_Register0);
    Wild_Register_Status (Wild_Channel_1, En_Register1);
    Wild_Register_Status (Wild_Channel_2, En_Register2);
}
```

Note: The modified address and modified data is user setting.

5.3.2 Wild Register Disable Work

Sample code

```
void wild_register_disable (void)
{
    Wild_Register_Status (Wild_Channel_0, Dis_Register0); // disable wild register 0
    Wild_Register_Status (Wild_Channel_1, Dis_Register1); // disable wild register 1
    Wild_Register_Status (Wild_Channel_2, Dis_Register2); // disable wild register 2
}
```

5.3.3 Read Wild Register

Sample code

```
void wild_register_read_enable (void)
{
    Wild_Register_Read (Wild_Channel_0, En_Register0); //enable read wild register 0
    Wild_Register_Read (Wild_Channel_1, En_Register1); //enable read wild register 1
    Wild_Register_Read (Wild_Channel_2, En_Register2); //enable read wild register 2
}

void wild_register_read_disable (void)
{
    Wild_Register_Read (Wild_Channel_0, Dis_Register0);// disable read wild register 0
    Wild_Register_Read (Wild_Channel_1, Dis_Register1); // disable read wild register 1
    Wild_Register_Read (Wild_Channel_2, Dis_Register2); // disable read wild register 2
}
```

6 Interrupts

This section introduces all functions of Interrupt library.

In routine, the interrupt library has functions to realize control interrupt, it including set interrupt, get interrupt status, control interrupt, clear interrupt request and get interrupt request

6.1 Interrupts Library Function List

Table 25 lists the interrupts library functions.

Table 25. Interrupts Functions

Function name	Description
void External_Interrupt(uint Int_name,uchar State)	Ext_Interrupt enable/ disable
void External_Interrupt_Level(uint Int_name_Level)	Set external interrupt level
uchar External_Interrupt_Request(uint Int_name_Request)	Clear the external interrupt
void Flash_memory_Interrupt(uchar Flash_State)	interrupt enable / disable
void Flash_memory_Level(uchar Flash_Level)	Set interrupt level
uchar Flash_memory_Request(void)	Clear interrupt request
void Watchtimer_Counter_Interrupt(uchar Watchtimer_State)	interrupt enable / disable
void Watchtimer_Counter_Level(uchar Watchtimer_Level)	Set interrupt level
uchar Watchtimer_Counter_Request(void)	Clear interrupt request
void Timebase_Time_Interrupt(uchar Timebase_State)	interrupt enable / disable
void Timebase_Time_Level(uchar Timebase_Level)	Set interrupt level
uchar Timebase_Time_Request(void)	Clear interrupt request
void AD_Converter_Interrupt(uchar AD_State)	interrupt enable / disable
void AD_Converter_Level(uchar AD_Level)	Set interrupt level
uchar AD_Converter_Request(void)	Clear interrupt request
void LIN_UART_Receive_Interrupt(uchar UART_Receive_State)	interrupt enable / disable
void LIN_UART_Receive_Level(uchar UART_Receive_Level)	Set interrupt level
uchar LIN_UART_Receive_Request(void)	Clear interrupt request
void LIN_UART_Transmit_Interrupt(uchar UART_Transmit_State)	interrupt enable / disable
void LIN_UART_Transmit_Level(uchar UART_Transmit_Level)	Set interrupt level
uchar LIN_UART_Transmit_Request(void)	Clear interrupt request
void Composite_Time_0_Lower_Interrupt(uchar Lower_0_State)	interrupt enable / disable
void Composite_Time_0_Lower_Level(uchar Lower_0_Level)	Set interrupt level
uchar Composite_Time_0_Lower_Request(void)	Clear interrupt request
void Composite_Time_0_Upper_Interrupt(uchar Upper_0_State)	interrupt enable / disable
void Composite_Time_0_Upper_Level(uchar Upper_0_Level)	Set interrupt level
uchar Composite_Time_0_Upper_Request(void)	Clear interrupt request
void Composite_Time_1_Upper_Interrupt(uchar Upper_1_State)	interrupt enable / disable
void Composite_Time_1_Upper_Level(uchar Upper_1_Level)	Set interrupt level
Function name	Description

Function name	Description
uchar Composite_Time_1_Upper_Request(void)	Clear interrupt request
void Composite_Time_1_Lower_Interrupt(uchar Lower_1_State)	interrupt enable / disable
void Composite_Time_1_Lower_Level(uchar Lower_1_Level)	Set interrupt level
uchar Composite_Time_1_Lower_Request(void)	Clear interrupt request

6.2 Interrupts Function Detail

6.2.1 External_Interrupt Function

Table 26 describes External_Interrupt function.

Table 26. External_Interrupt Function

Function name	External_Interrupt
Function prototype	void External_Interrupt(uint Int_name,uchar State)
Behavior description	Set external interrupt channel and state
Input parameter1	Int_name, extern interrupt channel
Input parameter2	State: Interrupt_Enable / Interrupt_Disable
Return value	None

The function can set the external interrupt enable and disable. Select parameter ‘Int_Name’ and ‘State’ to correct value and will set the interrupt enable or disable.

Table 27 describes the Int_name parameter values.

Table 27. Int_name Definition

Int_name	Description
Ext_Int2	Set external Interrupt channel 2
Ext_Int3	Set external Interrupt channel 3
Ext_Int4	Set external Interrupt channel 4
Ext_Int5	Set external Interrupt channel 5
Ext_Int6	Set external Interrupt channel 6
Ext_Int7	Set external Interrupt channel 7

Table 28 describes the State parameter values.

Table 28. State Definition

State	Description
Interrupt_Disable	Control external interrupt disable
Interrupt_Enable	Control external interrupt enable

6.2.2 External_Interrupt_Level Function

Table 29 describes External_Interrupt_Level function.

Table 29. External_Interrupt_Level Function

Function name	External_Interrupt_Level
Function prototype	void External_Interrupt_Level(uint Int_name_Level);
Behavior description	set the interrupt level of an interrupt request
Input parameter	Int_name_Level, external interrupt channel and level values
Return value	None

Table 30 describes the Int_name_Level parameter values.

Table 30. Int_name_Level Definition

Int_name_Level	Description
Ext_Int2_Level_0	Set external interrupt 2 channel level to 0
Ext_Int2_Level_1	Set external interrupt 2 channel level to 1
Ext_Int2_Level_2	Set external interrupt 2 channel level to 2
Ext_Int2_Level_3	Set external interrupt 2 channel level to 3
Ext_Int3_Level_0	Set external interrupt 3 channel level to 0
Ext_Int3_Level_1	Set external interrupt 3 channel level to 1
Ext_Int3_Level_2	Set external interrupt 3 channel level to 2
Ext_Int3_Level_3	Set external interrupt 3 channel level to 3
Ext_Int4_Level_0	Set external interrupt 4 channel level to 0
Ext_Int4_Level_1	Set external interrupt 4 channel level to 1
Ext_Int4_Level_2	Set external interrupt 4 channel level to 2
Ext_Int4_Level_3	Set external interrupt 4 channel level to 3
Ext_Int5_Level_0	Set external interrupt 5 channel level to 0
Ext_Int5_Level_1	Set external interrupt 5 channel level to 1
Ext_Int5_Level_2	Set external interrupt 5 channel level to 2
Ext_Int5_Level_3	Set external interrupt 5 channel level to 3
Ext_Int6_Level_0	Set external interrupt 6 channel level to 0
Ext_Int6_Level_1	Set external interrupt 6 channel level to 1
Ext_Int6_Level_2	Set external interrupt 6 channel level to 2
Ext_Int6_Level_3	Set external interrupt 6 channel level to 3
Ext_Int7_Level_0	Set external interrupt 7 channel level to 0
Ext_Int7_Level_1	Set external interrupt 7 channel level to1
Ext_Int7_Level_2	Set external interrupt 7 channel level to 2
Ext_Int7_Level_3	Set external interrupt 7 channel level to 3

6.2.3 External_Interrupt_Request Function

Table 31 describes External_Interrupt_Request function.

Table 31. External_Interrupt_Request Function

Function name	External_Interrupt_Request
Function prototype	uchar External_Interrupt_Request(uint Int_name_Request)
Behavior description	return the interrupt request and clear external interrupt request
Input parameter	Int_name_Request
Return value	Interrupt request values: 0(no request) or 1(have request)

The function can control the interrupt. Select parameter ‘Int_name_Request‘ to correct value and clear request ,return the values of request.

Table 32 describes the External_Interrupt_Request parameter values.

Table 32. External_Interrupt_Request Definition

Int_name_Request	Description
Ext_Int2_Request	select external interrupt channel 2
Ext_Int3_Request	select external interrupt channel 3
Ext_Int4_Request	select external interrupt channel 4
Ext_Int5_Request	select external interrupt channel 5
Ext_Int6_Request	select external interrupt channel 6
Ext_Int7_Request	select external interrupt channel 7

6.2.4 Other Interrupt Control Function

Table 33 describes other interrupt control function.

Table 33. Other Interrupt Control Function

Function name	Other interrupt control function
Function prototype1	void Flash_memory_Interrupt(uchar Flash_State)
Function prototype2	void Watchtimer_Counter_Interrupt(uchar Watchtimer_State)
Function prototype3	void Timebase_Time_Interrupt(uchar Timebase_State)
Function prototype4	void AD_Converter_Interrupt(uchar AD_State)
Function prototype5	void LIN_UART_Receive_Interrupt(uchar UART_Receive_State)
Function prototype6	void LIN_UART_Transmit_Interrupt(uchar UART_Transmit_State)
Function prototype7	void Composite_Time_0_Lower_Interrupt(uchar Lower_0_State)
Function prototype8	void Composite_Time_0_Upper_Interrupt(uchar Upper_0_State)
Function prototype9	void Composite_Time_1_Upper_Interrupt(uchar Upper_1_State)
Function prototype10	void Composite_Time_1_Lower_Interrupt(uchar Lower_1_State)
Behavior description	Control interrupt status, enable or disable
Input parameter	xx
Return value	None

Note: xx mean all above interrupt function input parameter name.

Above all the function can set the relevant parameter to control relevant interrupt enable or disable. Select the relevant parameter to correct value can control the interrupt.

Table 34 describes the xx_State parameter values.

Table 34. State Definition

XX_State	Description
Interrupt_Disable	Control interrupt disable
Interrupt_Enable	Control interrupt enable

Note: xx_State mean all above interrupt function input parameter name.

6.2.5 Other Interrupt Level Function

Table 35 describes other interrupt level function.

Table 35. Other Interrupt Level Function

Function name	Other interrupt level function
Function prototype1	void Flash_memory_Level(uchar Flash_Level)
Function prototype2	void Watchtimer_Counter_Level(uchar Watchtimer_Level)
Function prototype3	void Timebase_Time_Level(uchar Timebase_Level)
Function prototype4	void AD_Converter_Level(uchar AD_Level)
Function prototype5	void LIN_UART_Receive_Level(uchar UART_Receive_Level)
Function prototype6	void LIN_UART_Transmit_Level(uchar UART_Transmit_Level)
Function prototype7	void Composite_Time_0_Lower_Level(uchar Lower_0_Level)
Function prototype8	void Composite_Time_0_Upper_Level(uchar Upper_0_Level)
Function prototype9	void Composite_Time_1_Upper_Level(uchar Upper_1_Level)
Function prototype10	void Composite_Time_1_Lower_Level(uchar Lower_1_Level)
Behavior description	Set interrupt level
Input parameter	xx
Return value	None

Note: xx mean all above function input parameter name.

Above all the function can set the relevant interrupt level. Select the relevant parameter to correct value can set the level of relevant interrupt.

Table 36 describes the Relevant_Level parameter values.

Table 36. Relevant_Level Definition

Relevant_Level	Description
Level_0	Relevant interrupt level to 0
Level_1	Relevant interrupt level to 1
Level_2	Relevant interrupt level to 2
Level_3	Relevant interrupt level to 3

6.2.6 Other Clear and Get Interrupt Request Function

Table 37 describes other clear and get interrupt request function.

Table 37. Other Clear and Get Interrupt Request Function

Function name	Other clear and get interrupt request function
Function prototype1	uchar Flash_memory_Request(void);
Function prototype2	uchar Watchtimer_Counter_Request(void);
Function prototype3	uchar Timebase_Time_Request(void);
Function prototype4	uchar AD_Converter_Request(void);
Function prototype5	uchar LIN_UART_Receive_Request(void);
Function prototype6	uchar LIN_UART_Transmit_Request(void);
Function prototype7	uchar Composite_Time_0_Lower_Request(void);
Function prototype8	uchar Composite_Time_0_Upper_Request(void);
Function prototype9	uchar Composite_Time_1_Upper_Request(void);
Function prototype10	uchar Composite_Time_1_Lower_Request(void);
Behavior description	return the interrupt request and clear interrupt request
Input parameter	None
Return value	Interrupt request values: 0(no request) or 1(have request)

Above all the function can clear the relevant interrupt Request. After run this function will return the status of relevant interrupt request.

6.3 Usage Demo

6.3.1 External Interrupt

If you want to use the interrupt, it will relate with set interrupt, level and interrupt control. When you want to set the external interrupt, you can use the External_Interrupt(int Int_name,uchar State) function. Set the interrupt level use the External_Interrupt_Level(int Int_name_Level) function. Use the External_Interrupt_Request(int Int_name_request) function can get the interrupt request and clear it.

The sample as:

```
Void Interrupt_function(void)
{
        External_Interrupt_Level(Ext_Int2_Level_0);          //set the extern interrupt and level 0
        External_Interrupt(Ext_Int2,Interrupt_Enable);//enable the extern 2 interrupt
}
__interrupt void external_int2( void )
{
      … …
        External_Interrupt_Request(Ext_Int2_Request);// get the interrupt request and clear it
}
```

6.3.2 Other Interrupt

When you want to set other interrupt, you can use the interrupt set function, interrupt level set function and interrupt control function.

The sample as:

```
void Interrupt_function(void)
{
    Timebase_Time_Interrupt(Interrupt_Enable);//set Timebase time interrupt enable
    AD_Converter_Interrupt(Interrupt_Disable);//set AD interrupt disable
    //set composite time interrupt enable
    Composite_Time_1_Lower_Interrupt(Interrupt_Enable);
    Timebase_Time_Level(Level_0);
    AD_Converter_Level(Level_1);
    Composite_Time_1_Lower_Level(Level_2);
}
__interrupt void Interrupt_Timebase_Time(void)
{
    … …
    Interrtup_status = Timebase_Time_Request(); //get the interrupt request and clear it
}
```

7 External Interrupt

This section introduces all functions of external interrupt library.

7.1 External Interrupt Library Function List

Table 38 lists the External Interrupt library functions.

Table 38. External Interrupt Functions

Function name	Description
void External_Interrupt_Init(uint channel, uchar Edge)	Setting the external interrupts input channel and the edge polarity

7.2 External Interrupt Function Detail

7.2.1 External_Interrupt_Init Function

Table 39 describes Standby_Mode_Enter function.

Table 39. External_Interrupt_Init Function

Function name	External_Interrupt_Init
Function prototype	void External_Interrupt_Init(uint Channel, uchar Edge)
Behavior description	Setting the external interrupts input channel and the edge polarity
Input parameter1	Channel, Choose the external interrupts input pin
Input parameter2	Edge, Select the edge polarity
Return value	None

Note: Input parameter Edge values be relate to Channel parameter.

For example: when Channel is Ch_2, Edge only has four input values: No_2, Rising_2, Falling_2 and Both_2.

Table 40 describes the Channel parameter values.

Table 40. Channel Definition

Channel	Description
Ch_2	External interrupt ch.2
Ch_3	External interrupt ch.3
Ch_4	External interrupt ch.4
Ch_5	External interrupt ch.5
Ch_6	External interrupt ch.6
Ch_7	External interrupt ch.7

Table 41 describes the Edge parameter values.

Table 41. Edge Definition

Edge	Description
No_2	Extern interrupt ch.2 No edge detection
Rising_2	Extern interrupt ch.2 Rising edge
Falling_2	Extern interrupt ch.2 Falling edge
Both_2	Extern interrupt ch.2 Both edge
No_3	Extern interrupt ch.3 No edge detection
Rising_3	Extern interrupt ch.3 Rising edge
Falling_3	Extern interrupt ch.3 Falling edge
Both_3	Extern interrupt ch.3 Both edge
No_4	Extern interrupt ch.4 No edge detection
Rising_4	Extern interrupt ch.4 Rising edge
Falling_4	Extern interrupt ch.4 Falling edge
Both_4	Extern interrupt ch.4 Both edge
No_5	Extern interrupt ch.5 No edge detection
Rising_5	Extern interrupt ch.5 Rising edge
Falling_5	Extern interrupt ch.5 Falling edge
Both_5	Extern interrupt ch.5 Both edge
No_6	Extern interrupt ch.6 No edge detection
Rising_6	Extern interrupt ch.6 Rising edge
Falling_6	Extern interrupt ch.6 Falling edge
Both_6	Extern interrupt ch.6 Both edge
No_7	Extern interrupt ch.7 No edge detection
Rising_7	Extern interrupt ch.7 Rising edge
Falling_7	Extern interrupt ch.7 Falling edge
Both_7	Extern interrupt ch.7 Both edge

7.3 Usage Demo

External_Interrupt_Init()

Select the external interrupt input channel and select the edge polarity. This function can refer to API for Interrupt.

External_Interrupt_Init(channel_7,Rising_Edge); //channel 7,rising edge

8 Additional Information

For more information about how to use MB95200H EV-board, BGM Adaptor and SOFTUNE, please refer to SKT MB2146-410-01-E User Manual, or visit website:

http://www.cypress.com/documentation/software-and-drivers/f2mc-8fx-mb95200h210h-series-starter-kit-mb2146-410a-01-e-setup

Document History

Document Title: AN205396 – F²MC-8FX Family MB95200 Series 8-Bit Microcontroller Basic API

Document Number: 002-05396

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	HUAL	03/13/2009	Initial release
*A	5264567	HUAL	05/18/2016	Migrated Spansion Application Note MCU-AN- 500022-E-10 to Cypress format.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

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

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corporation. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 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.