SN8F22E80B Series

USER'S MANUAL

SN8F22E83B/SN8F22E831B
SN8F22E84B
SN8F22E87B
SN8F22E88B

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 1.0	2014/11/11	First version released.
VER 1.1	2014/11/13	1. Modify PWCH20 & PWCH21 register define.
VER 1.2	2015/01/27	1. Modify ISP FLASH ROM ERASE OPERATION Example.
VER 1.3	2015/03/20	1. Modify WDT overflow time. 2. Modify ADT register define.
VER 1.4	2015/05/04	1. Modify USB Pin Descriptions. 2. Update VDD &USBVDD operating voltage 6.0V.
VER 1.5	2015/07/06	1. Update QFN46 Package Information. 2. Modify ADR register. 3. Add the Notice of Interrupt Register Application 4. Add the Notice of SIOM register. (Setting SIOM by mov instruction except FSTART)
VER 1.6	2015/09/09	1. Add the description of IHRC value reload method. 2. Modify the flash ISP sample code.
VER 1.7	2015/11/26	1. Delete SYSTEM CLOCK descriptions in section 4.4.1 and section 4.4.2 . 2. Modify TC0 EVENT COUNTER descriptions. 3. Modify T1 16-BIT BASIC TIMER title. 4. Modify T0 and T1 example. 5. Update QFN24 Package Information.
VER 1.8	2016/09/19	1. Add SN8F22E831B Package Information.
VER 1.9	2017/03/22	1. Modify Operating ambient temperature range

Table of Content

1 PRODUCT OVERVIEW

1.1 FEATURES

- **Memory configuration**
  Flash ROM size:16K*16 bits.
  User RAM size: 1024*8 bits.
  USB RAM size: 256*8 bits.

- **8 levels stack buffer.**

- **I/O pin configuration**
  Bi-directional: P0, P1, P2.1~P2.7, P4, and P5.
  Wakeup: P0, P1 level change.
  Pull-up resisters: P0, P1, P2.1~P2.7, P4, and P5.
  External interrupt: P0, P1.
  ADC input pin: AIN0~AIN15.

- **Full Speed USB 2.0**
  Conforms to USB specification version 2.0.
  3.3v regulator output for USB D+ pin internal 1.5k pull-up resistor.
  Supports one Full speed USB device address.
  One control endpoint and 6 configurable Isochronous/interrupt/bulk endpoints.
  EP0 supports 64-byte FIFO depth.
  Programmable EP1~EP6 FIFO depth.
  Total 7 endpoints share 256-byte USB RAM.
  Supports USB power plug-in detection 3.6v.
  Supports PS/2 mode shared with D+/D- pins.

- **Powerful instructions**
  Instruction's length is one word.
  Most of instructions are one cycle only.
  All ROM area JMP instruction.
  All ROM area lookup table function (MOVC).

- **15 interrupt sources**
  11 internal interrupts: T0, T1, TC0, TC1, TC2, SIO, ADC, WAKE, USB, UPDET, and NDT
  4 external interrupts: INT0, INT1, P0, P1

- **Build in Embedded ICE function.**

- **Four 8-bit timers. (T0, TC0, TC1, TC2).**
  T0: Basic timer.
  TC0: Timer/PWM/Pulse/Counter.
  TC1: Timer/PWM/Pulse/Counter.
  TC2: Timer/PWM/Pulse/Counter.

- **One 16-bit timer (T1) with Timer and Counter.**

- **3-channel duty/cycle programmable PWMs with extension, from a generator to output PWM, Buzzer, IR carrier and single pulse.**

- **16 channel 12-bit SAR ADC**
  Sixteen external ADC input
  AIN0 supports battery voltage direct detection.

- **Serial Interface: SIO.**

- **On chip watchdog timer and clock source.**

- **Three system clocks**
  External high clock: 12MHz and 16MHz.
  Internal high clock: RC type 16MHz.
  Internal low clock: RC type 32KHz.

- **Four operating modes**
  Normal mode: Both high and low clock active.
  Slow mode: Low clock only.
  Sleep mode: Both high and low clock stop.
  Green mode: Periodical wakeup by timer.

- **In-System Programming (ISP) supported.**

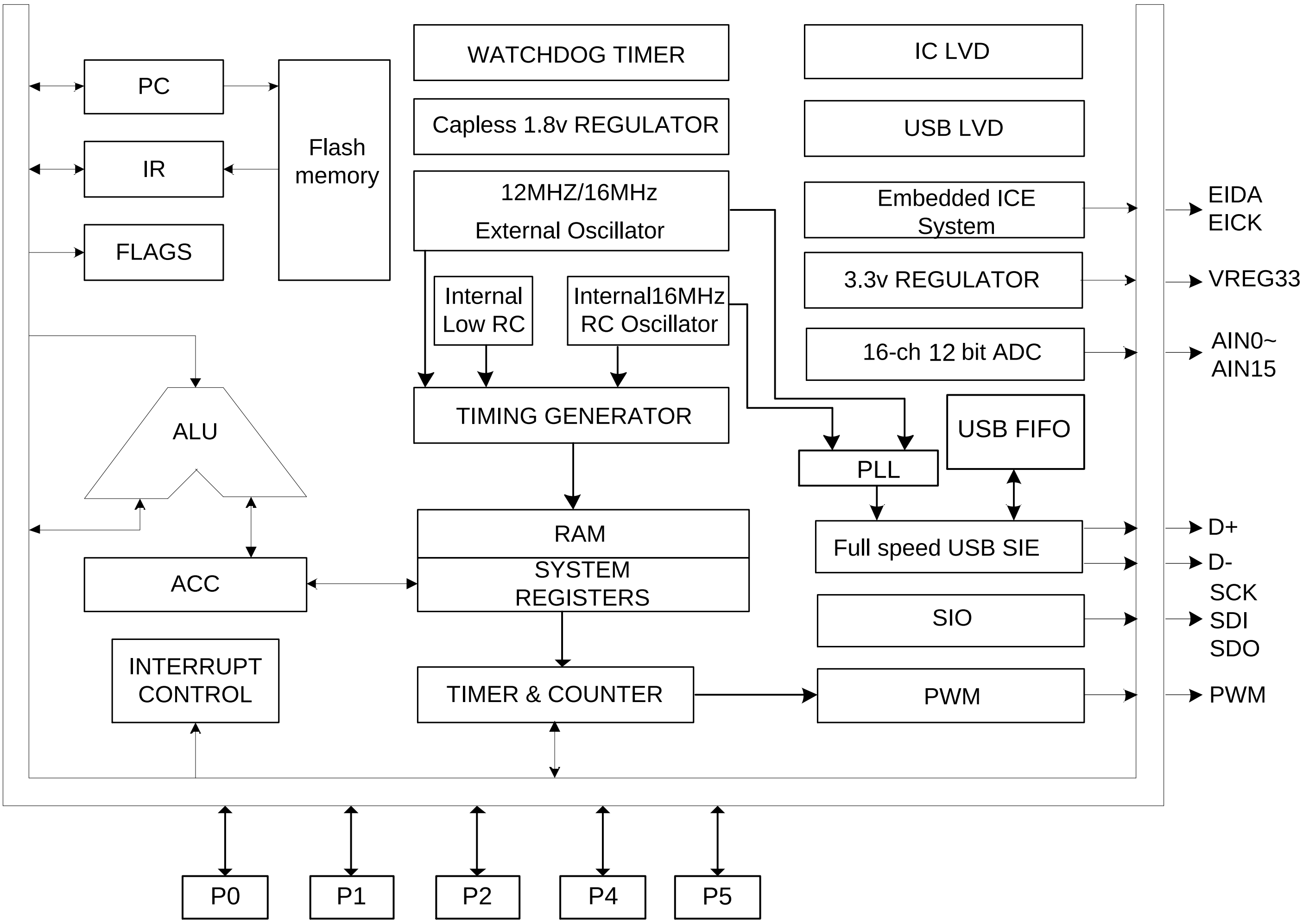
- **Build in PLL 48MHz for USB SIE.**

- **Package (Chip form support)**
  LQFP 48 pin
  QFN 46 pin
  SOP 28 pin
  QFN 24 pin
  SSOP 24 pin

- **Features Selection Table**

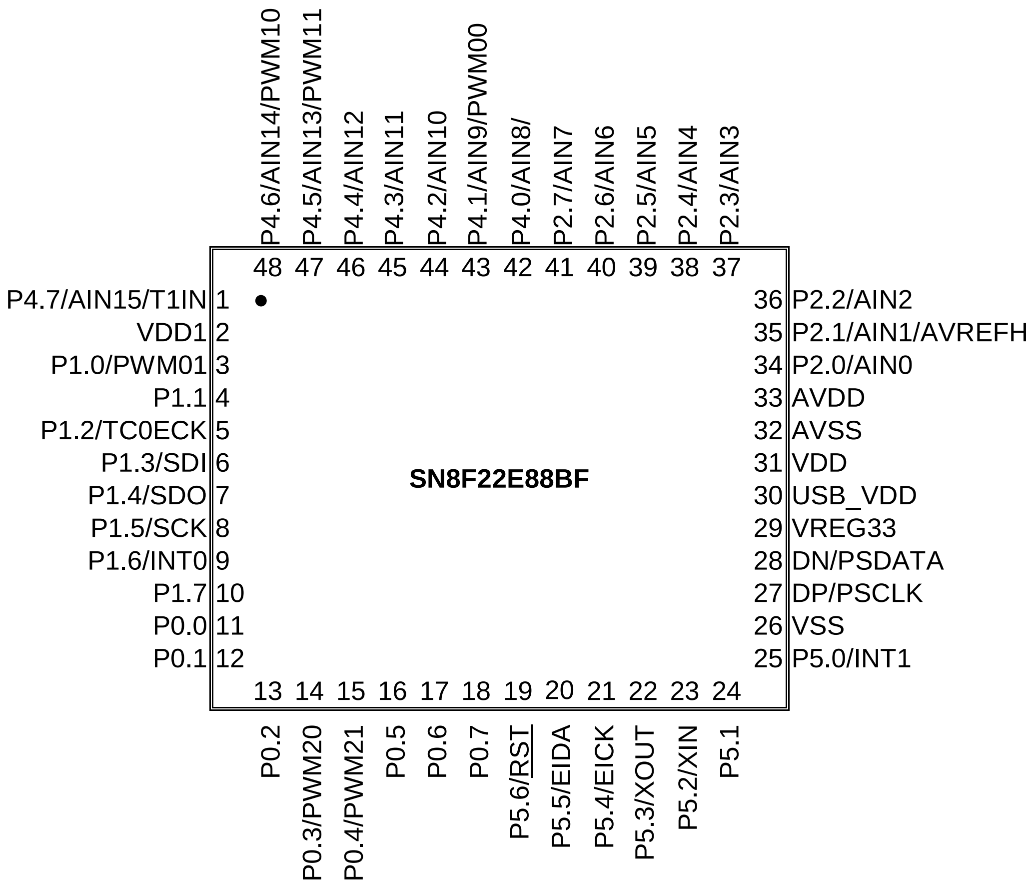
CHIP	ROM	RAM	Stack	USB FS	Timer	I/O	PWM	SIO (SPI)	ADC CH	Ext.INT	ISP/ Embedded ICE	Operating Voltage	Package
SN8F22E88B	16K*16	1024*8	8	V	8-bit*4 16-bit*1	39	6-ch	V	16	2	V	1.8V~6.0V	LQFP48
SN8F22E87B	16K*16	1024*8	8	V	8-bit*4 16-bit*1	37	6-ch	V	16	2	V	1.8V~6.0V	QFN46
SN8F22E84B	16K*16	1024*8	8	V	8-bit*4 16-bit*1	22	4-ch	V	7	1	V	1.8V~6.0V	SOP28
SN8F22E83B	16K*16	1024*8	8	V	8-bit*4 16-bit*1	18	1-ch	V	6	1	V	1.8V~6.0V	QFN24 (4*4)
SN8F22E831B	16K*16	1024*8	8	V	8-bit*4 16-bit*1	18	3-ch	V	6	1	V	1.8V~6.0V	SSOP24

1.2 SYSTEM BLOCK DIAGRAM

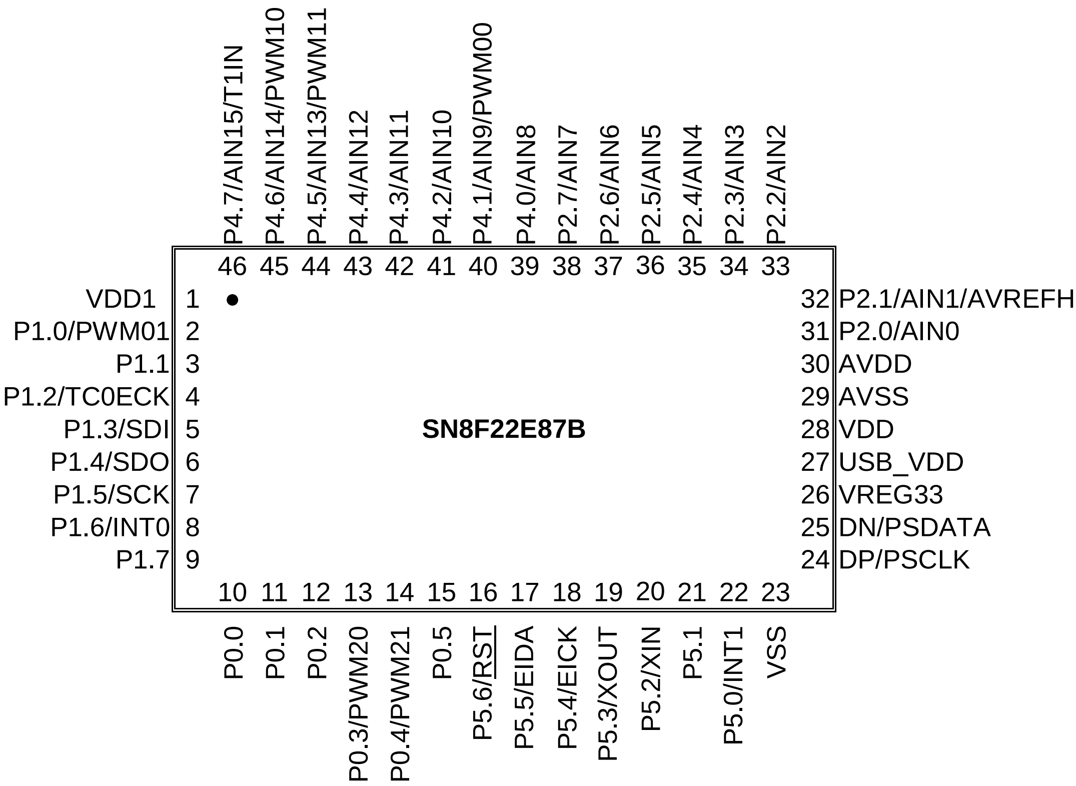


1.3 PIN ASSIGNMENT

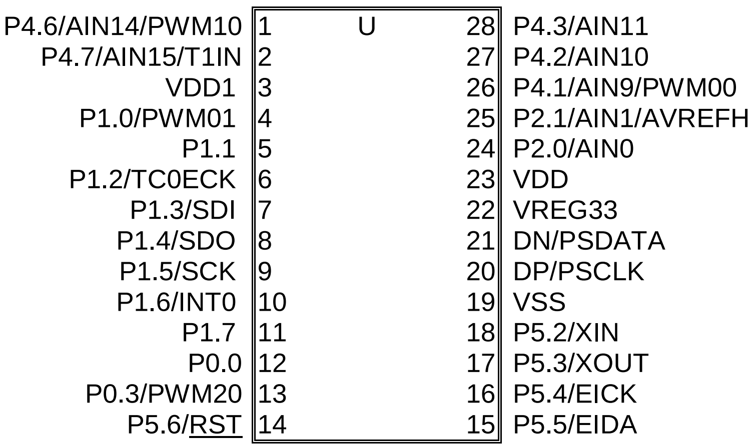
SN8F22E88BF (LQFP 48 pins)



SN8F22E87BJ (QFN 46 pins)

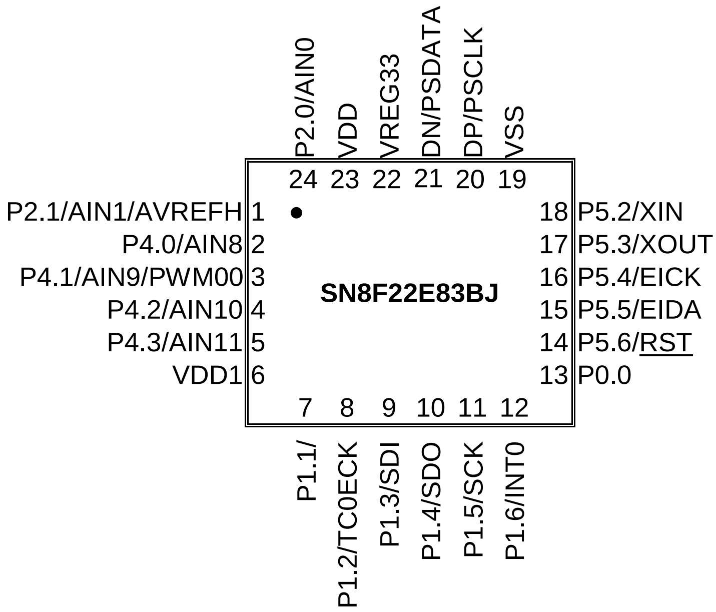


SN8F22E84BS (SOP 28 pins)

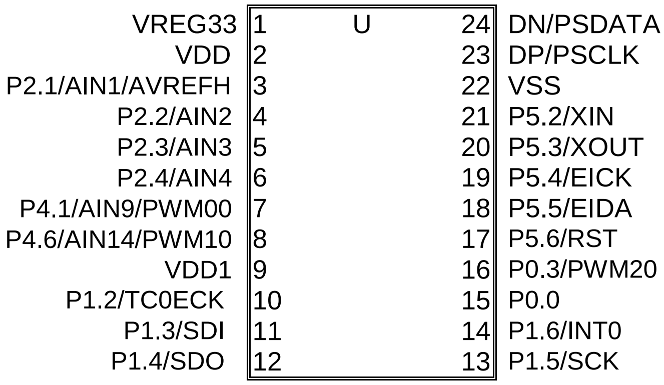


SN8F22E84BS

SN8F22E83BJ (QFN 24 pins)



SN8F22E831BX (SSOP 24 pins)



SN8F22E831BX

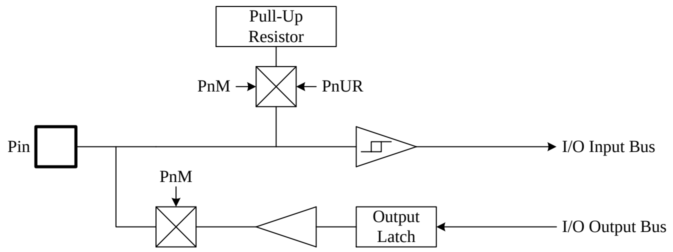
1.4 PIN DESCRIPTIONS

PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P P	SN8F22E88BF/J : Power supply input pins for digital circuit.
		SN8F22E84BS : Power supply input pins for digital and analog circuit.
		SN8F22E83BJ : Power supply input pins for digital , analog and USB circuit.
AVDD, AVSS	P	Power supply input pins for analog circuit.
VREG33	O	3.3v voltage output from USB 3.3v regulator.
USBVDD	P	Power supply input pins for USB circuit.
VDD1	P	Power supply input pins for the I/O power of P1.0~P1.7.
DP/PSCLK	I/O	USB Differential D+ signal line.
		PSCLK: PS/2 clock pin with internal 5K pull-up resistor.
DN/PSDATA	I/O	USB Differential D- signal line.
		PSDATA: PS/2 data pin wit internal 5K pull-up resistor.
P0[2:0]	I/O	Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
P0.3/PWM20	I/O	P0.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		PWM20: TC2 programmable PWM output pin 0.
P0.4/PWM21	I/O	P0.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		PWM21: TC2 programmable PWM output pin 1.
P0[7:5]	I/O	Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
P1.0/PWM01	I/O	P1.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		PWM01: TC0 programmable PWM output pin 1.
P1.1	I/O	P1.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
P1.2/TC0ECK	I/O	P1.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		TC0ECK: TC0 event counter input pin.
P1.3/SDI	I/O	P1.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		SDI: SIO data input pin.
P1.4/SDO	I/O	P1.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		SDO: SIO data output pin.
P1.5/SCK	I/O	P1.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		SCK: SIO clock pin.
P1.6/INT0	I/O	P1.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		INT0: External interrupt 0 input pin.
P1.7	I/O	P1.7: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
P2.0/AIN0	I	P2.0: Input only pin. Schmitt trigger structure as input mode.
		AIN0: ADC channel 0 input pin.
P2.1/AIN1/AVREFH	I/O	P2.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		AIN1: ADC channel 1 input pin.
		AVREFH: ADC external high reference voltage input.
P2[7:2]/AIN[7:2]	I/O	P2.2~P2.7: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		AIN2~AIN7: ADC channel 2~7 input pin.
P4[4:0]/AIN[12:8]	I/O	P4.0~P4.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		AIN8~AIN12: ADC channel 8 input pin.
P4.1/PWM00	I/O	PWM00: TC0 programmable PWM output pin 0.
P4.5/PWM11/ AIN13	I/O	P4.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		PWM11: TC1 programmable PWM output pin 1.
		AIN13: ADC channel 13 input pint.
P4.6/ PWM10/AIN14	I/O	P4.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		PWM10: TC1 programmable PWM output pin 0.
		AIN14: ADC channel 13 input pint.
P4.7/T1IN/AIN15	I/O	P4.7: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		T1IN: T1 event counter input pin.
		AIN15: ADC channel 15 input pin.
P5.0/INT1	I/O	P5.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		INT1: External interrupt 0 input pin.
P5.1	I/O	P5.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
P5.2/XIN	I/O	P5.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		XIN: Oscillator input pin while external oscillator enable.
P5.3/XOUT	I/O	P5.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		XOUT: Oscillator output pin while external oscillator enable.
P5.4/EICK	I/O	P5.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		EICK: Embedded ICE clock pin.
P5.5/EIDA	I/O	P5.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.

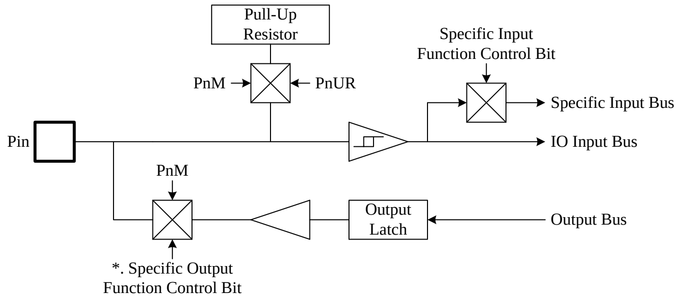
		EIDA: Embedded ICE data pin.
P5.6/RST	I/O	P5.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor.
		RST: System external reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”

1.5 PIN CIRCUIT DIAGRAMS

- **Normal Bi-direction I/O Pin.**

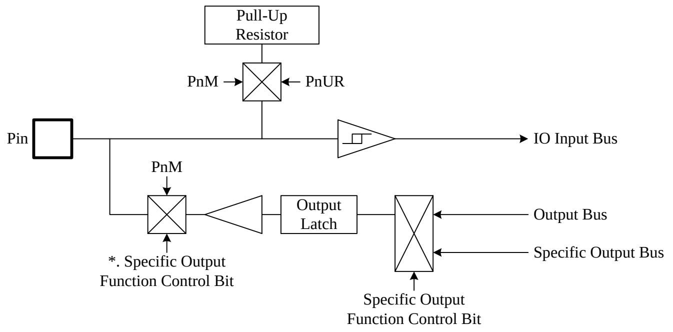


- **Bi-direction I/O Pin Shared with Specific Digital Input Function, e.g. INT0, Event counter, SIO.**



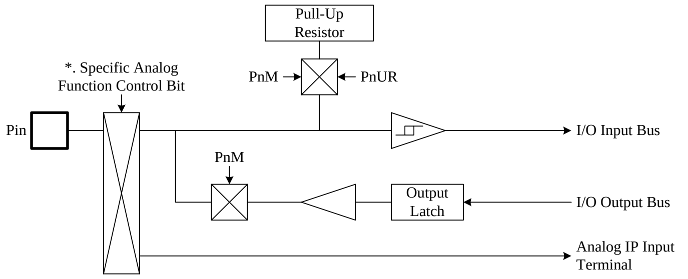
***. Some specific functions switch I/O direction directly, not through PnM register.**

- **Bi-direction I/O Pin Shared with Specific Digital Output Function, e.g. PWM, SIO.**



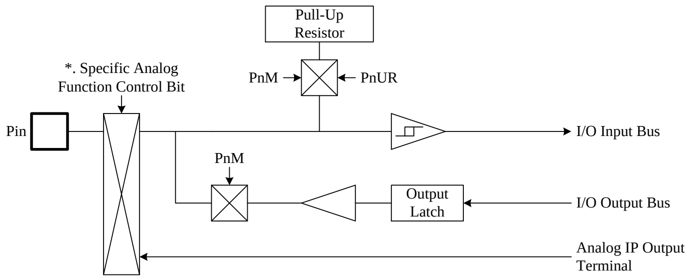
***. Some specific functions switch I/O direction directly, not through PnM register.**

- **Bi-direction I/O Pin Shared with Specific Analog Input Function, e.g. XIN, ADC.**



***. Some specific functions switch I/O direction directly, not through PnM register.**

- **Bi-direction I/O Pin Shared with Specific Analog Output Function, e.g. XOUT…**



***. Some specific functions switch I/O direction directly, not through PnM register.**

2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (ROM)

☞ ***16K words ROM***

Address	***ROM***	Comment
0000H	*Reset vector*	Reset vector
0001H . 0007H	*General purpose area*	User program
0008H	***WAKE Interrupt vector***	**Interrupt vector**
0009H	***INT0 Interrupt vector***	
000AH	***INT1 Interrupt vector***	
000BH	***UPDET Interrupt vector***	
000CH	***USB Interrupt vector***	
000DH	***P0 Interrupt vector***	
000EH	***P1 Interrupt vector***	
000FH	***T0 Interrupt vector***	
0010H	***T1 Interrupt vector***	
0011H	***TC0 Interrupt vector***	
0012H	***TC1 Interrupt vector***	
0013H	***TC2 Interrupt vector***	
0014H	***NDT Interrupt vector***	
0015H	***ADC Interrupt vector***	
0016H	***SIO Interrupt vector***	
0017H 3FF7H	*General purpose area*	User program End of user program
3FF8H … 3FFFH	Reserved	

2.1.1 RESET VECTOR (0000H)

A one-word vector address area is used to execute system reset.

- **Power On Reset (POR=1).**
- **Watchdog Reset (WDT=1).**
- **External Reset (RST=1).**

After power on reset, external reset or watchdog timer overflow reset, then the chip will restart the program from address 0000h and all system registers will be set as default values. It is easy to know reset status from POR, WDT, and RST flags of PFLAG register. The following example shows the way to define the reset vector in the program memory.

> **Example: Defining Reset Vector**

```
		ORG	0		; 0000H
		JMP	START		; Jump to user program address.
		…

		ORG	17H
START:					; 0017H, The head of user program.
		…			; User program
		…


		ENDP			; End of program
```

> ✸ ***Note: The head of user program should skip interrupt vector area to avoid program execution error.***

2.1.2 INTERRUPT VECTOR (0008H~0016H)

A 15-word vector address area is used to execute interrupt request. If any interrupt service executes, the program counter (PC) value is stored in stack buffer and jump to 0008h~0016h of program memory to execute the vectored interrupt. This interrupt is multi-vector and each of interrupts points to unique vector. Users have to define the interrupt vector. The following example shows the way to define the interrupt vector in the program memory.

	ROM	Priority
0008H	WAKE Interrupt vector	1
0009H	INT0 Interrupt vector	2
000AH	INT1 Interrupt vector	3
000BH	UPDET Interrupt vector	4
000CH	USB Interrupt vector	5
000DH	P0 Interrupt vector	6
000EH	P1 Interrupt vector	7
000FH	T0 Interrupt vector	8
0010H	T1 Interrupt vector	9
0011H	TC0 Interrupt vector	10
0012H	TC1 Interrupt vector	11
0013H	TC2 Interrupt vector	12
0014H	NDT Interrupt vector	13
0015H	ADC Interrupt vector	14
0016H	SIO Interrupt vector	15

When one interrupt request occurs, and the program counter points to the correlative vector to execute interrupt service routine. If WAKE interrupt occurs, the program counter points to ORG 8. If INT0 interrupt occurs, the program counter points to ORG 9. In normal condition, several interrupt requests happen at the same time. So the priority of interrupt sources is very important, or the system doesn't know which interrupt is processed first. The interrupt priority is follow vector sequence. ORG 8 is priority 1. ORG 9 is priority 2. In the case, the interrupt processing priority is as following.

If WAKE, ADC, T0, SIO and TC2 interrupt requests happen at the same time, the system processing interrupt sequence is WAKE, T0, TC2, ADC, and then SIO. The system processes WAKE interrupt service routine first, and then processes T0 interrupt routine…Until finishing processing all interrupt requests.

➢ **Example:**

Interrupt Request Occurrence Sequence: (2~8 interrupt requests occur during WAKE interrupt service routine execution.)

1	2	3	4	5	6	7	8
WAKE	ADC	TC1	T0	SIO	INT0	TC2	NDT

Interrupt Processing Sequence:

1	2	3	4	5	6	7	8
WAKE	INT0	T0	TC1	TC2	NDT	ADC	SIO

➢ **Example: Defining Interrupt Vector. The interrupt service routine is following user program.**

```
.CODE
            ORG     0               ; 0000H
            JMP     START           ; Jump to user program address.
            …
            JMP     ISR_WAKE        ; Jump to interrupt service routine address.
            JMP     ISR_INT0
            JMP     ISR_INT1
            JMP     ISR_UPDET
            JMP     ISR_USB
            JMP     ISR_P0
            JMP     ISR_P1
            JMP     ISR_T0
            JMP     ISR_T1
            JMP     ISR_TC0
            JMP     ISR_TC1
            JMP     ISR_TC2
            JMP     ISR_NDT
            JMP     ISR_ADC
            JMP     ISR_SIO
            ORG     17H
START:                              ; 0017H, The head of user program.
            …                       ; User program.
            …
            …
            JMP     START           ; End of user program.
            …
ISR_WAKE:                           ; The head of interrupt service routine.
                                    ; Save ACC and 0x80~0x8F register to buffers.
            …
                                    ; Load ACC and 0x80~0x8F register from buffers.
            RETI                    ; End of interrupt service routine.
ISR_INT0:                           ;
                                    ; Save ACC and 0x80~0x8F register to buffers.
            …
                                    ; Load ACC and 0x80~0x8F register from buffers.
            RETI                    ; End of interrupt service routine.
            …
            …
            …
            …
            …
ISR_SIO:                            ;
                                    ; Save ACC and 0x80~0x8F register to buffers.
            …
                                    ; Load ACC and 0x80~0x8F register from buffers.
            RETI                    ; End of interrupt service routine.


            ENDP                    ; End of program.
```

✸ ***Note: It is easy to understand the rules of SONIX program from demo programs given above. These points are as following:***

1. ***The address 0000H is a "JMP" instruction to make the program starts from the beginning.***
2. ***The address 0008H~0016H is interrupt vector.***
3. ***User's program is a loop routine for main purpose application.***

2.1.3 LOOK-UP TABLE DESCRIPTION

In the ROM's data lookup function, Y register is pointed to middle byte address (bit 8~bit 15) and Z register is pointed to low byte address (bit 0~bit 7) of ROM. After MOVC instruction executed, the low-byte data will be stored in ACC and high-byte data stored in R register.

- **Example: To look up the ROM data located "TABLE1".**

```
                B0MOV      Y, #TABLE1$M    ; To set lookup table1's middle address
                B0MOV      Z, #TABLE1$L    ; To set lookup table1's low address.
                MOVC                       ; To lookup data, R = 00H, ACC = 35H

                                           ; Increment the index address for next address.
                INCMS      Z               ; Z+1
                JMP        @F              ; Z is not overflow.
                INCMS      Y               ; Z overflow (FFH → 00), → Y=Y+1
                NOP                        ;
                                           ;
 @@:            MOVC                       ; To lookup data, R = 51H, ACC = 05H.
                …                          ;
TABLE1:         DW         0035H           ; To define a word (16 bits) data.
                DW         5105H
                DW         2012H
                …
```

> ✸ ***Note: The Y register will not increase automatically when Z register crosses boundary from 0xFF to 0x00. Therefore, user must be take care such situation to avoid look-up table errors. If Z register is overflow, Y register must be added one. The following INC_YZ macro shows a simple method to process Y and Z registers automatically.***

- **Example: INC_YZ macro.**

```
INC_YZ          MACRO
                INCMS      Z               ; Z+1
                JMP        @F              ; Not overflow

                INCMS      Y               ; Y+1
                NOP                        ; Not overflow
@@:
                ENDM
```

➢ **Example: Modify above example by "INC_YZ" macro.**

```
            B0MOV   Y, #TABLE1$M    ; To set lookup table1's middle address
            B0MOV   Z, #TABLE1$L    ; To set lookup table1's low address.
            MOVC                    ; To lookup data, R = 00H, ACC = 35H

            INC_YZ                  ; Increment the index address for next address.
                                    ;
@@:         MOVC                    ; To lookup data, R = 51H, ACC = 05H.
            …                       ;
TABLE1:     DW      0035H           ; To define a word (16 bits) data.
            DW      5105H
            DW      2012H
            …
```

The other example of look-up table is to add Y or Z index register by accumulator. Please be careful if "carry" happen.

➢ **Example: Increase Y and Z register by B0ADD/ADD instruction.**

```
            B0MOV   Y, #TABLE1$M    ; To set lookup table's middle address.
            B0MOV   Z, #TABLE1$L    ; To set lookup table's low address.

            B0MOV   A, BUF          ; Z = Z + BUF.
            B0ADD   Z, A

            B0BTS1  FC              ; Check the carry flag.
            JMP     GETDATA         ; FC = 0
            INCMS   Y               ; FC = 1. Y+1.
            NOP
GETDATA:                            ;
            MOVC                    ; To lookup data. If BUF = 0, data is 0x0035
                                    ; If BUF = 1, data is 0x5105
                                    ; If BUF = 2, data is 0x2012
            …

TABLE1:     DW      0035H           ; To define a word (16 bits) data.
            DW      5105H
            DW      2012H
            …
```

2.1.4 JUMP TABLE DESCRIPTION

The jump table operation is one of multi-address jumping function. Add low-byte program counter (PCL) and ACC value to get one new PCL. If PCL is overflow after PCL+ACC, PCH adds one automatically. The new program counter (PC) points to a series jump instructions as a listing table. It is easy to make a multi-jump program depends on the value of the accumulator (A).

> ✸ ***Note: PCH only support PC up counting result and doesn't support PC down counting. When PCL is carry after PCL+ACC, PCH adds one automatically. If PCL borrow after PCL–ACC, PCH keeps value and not change.***

➢ **Example: Jump table.**

```
        ORG     0X0100          ; The jump table is from the head of the ROM boundary

        B0ADD   PCL, A          ; PCL = PCL + ACC, PCH + 1 when PCL overflow occurs.
        JMP     A0POINT         ; ACC = 0, jump to A0POINT
        JMP     A1POINT         ; ACC = 1, jump to A1POINT
        JMP     A2POINT         ; ACC = 2, jump to A2POINT
        JMP     A3POINT         ; ACC = 3, jump to A3POINT
```

SONIX provides a macro for safe jump table function. This macro will check the ROM boundary and move the jump table to the right position automatically. The side effect of this macro maybe wastes some ROM size.

➢ **Example: If "jump table" crosses over ROM boundary will cause errors.**

```
@JMP_A  MACRO   VAL
        IF      (($+1) !& 0XFF00) !!= (($+(VAL)) !& 0XFF00)
        JMP     ($ | 0XFF)
        ORG     ($ | 0XFF)
        ENDIF
        B0ADD   PCL, A
        ENDM
```

> ✸ ***Note: "VAL" is the number of the jump table listing number.***

➢ **Example: "@JMP_A" application in SONIX macro file called "MACRO3.H".**

```
        B0MOV   A, BUF0         ; "BUF0" is from 0 to 4.
        @JMP_A  5               ; The number of the jump table listing is five.
        JMP     A0POINT         ; ACC = 0, jump to A0POINT
        JMP     A1POINT         ; ACC = 1, jump to A1POINT
        JMP     A2POINT         ; ACC = 2, jump to A2POINT
        JMP     A3POINT         ; ACC = 3, jump to A3POINT
        JMP     A4POINT         ; ACC = 4, jump to A4POINT
```

If the jump table position is across a ROM boundary (0x00FF~0x0100), the "@JMP_A" macro will adjust the jump table routine begin from next RAM boundary (0x0100).

- **Example: "@JMP_A" operation.**

; Before compiling program.

```
ROM address
                B0MOV     A, BUF0       ; "BUF0" is from 0 to 4.
                @JMP_A    5             ; The number of the jump table listing is five.
0X00FD          JMP       A0POINT       ; ACC = 0, jump to A0POINT
0X00FE          JMP       A1POINT       ; ACC = 1, jump to A1POINT
0X00FF          JMP       A2POINT       ; ACC = 2, jump to A2POINT
0X0100          JMP       A3POINT       ; ACC = 3, jump to A3POINT
0X0101          JMP       A4POINT       ; ACC = 4, jump to A4POINT
```

; After compiling program.

```
ROM address
                B0MOV     A, BUF0       ; "BUF0" is from 0 to 4.
                @JMP_A    5             ; The number of the jump table listing is five.
0X0100          JMP       A0POINT       ; ACC = 0, jump to A0POINT
0X0101          JMP       A1POINT       ; ACC = 1, jump to A1POINT
0X0102          JMP       A2POINT       ; ACC = 2, jump to A2POINT
0X0103          JMP       A3POINT       ; ACC = 3, jump to A3POINT
0X0104          JMP       A4POINT       ; ACC = 4, jump to A4POINT
```

2.1.5 CHECKSUM CALCULATION

The last ROM address is reserved area. User should avoid these addresses (last address) when calculate the Checksum value.

➢ **Example: The demo program shows how to calculated Checksum from 00H to the end of user's code.**

```
                MOV         A,#END_USER_CODE$L
                B0MOV       END_ADDR1, A        ; Save low end address to end_addr1
                MOV         A,#END_USER_CODE$M
                B0MOV       END_ADDR2, A        ; Save middle end address to end_addr2
                CLR         Y                   ; Set Y to 00H
                CLR         Z                   ; Set Z to 00H
@@:
                MOVC
                B0BCLR      FC                  ; Clear C flag
                ADD         DATA1, A            ; Add A to Data1
                MOV         A, R
                ADC         DATA2, A            ; Add R to Data2
                JMP         END_CHECK           ; Check if the YZ address =   the end of code
AAA:
                INCMS       Z                   ; Z=Z+1
                JMP         @B                  ; If Z != 00H calculate to next address
                JMP         Y_ADD_1             ; If Z = 00H increase Y
END_CHECK:
                MOV         A, END_ADDR1
                CMPRS       A, Z                ; Check if Z = low end address
                JMP         AAA                 ; If Not jump to checksum calculate
                MOV         A, END_ADDR2
                CMPRS       A, Y                ; If Yes, check if Y = middle end address
                JMP         AAA                 ; If Not jump to checksum calculate
                JMP         CHECKSUM_END        ; If Yes checksum calculated is done.

Y_ADD_1:
                INCMS       Y                   ; Increase Y
                NOP
                JMP         @B                  ; Jump to checksum calculate
CHECKSUM_END:
                …
                …
END_USER_CODE:                                  ; Label of program end
```

2.2 DATA MEMORY (RAM)

☞ ***1024 X 8-bit RAM***

	Address	*RAM Location*	
	000H		RAM Bank 0
	"	***General Purpose Area***	
	"		
	"		
BANK 0	07FH		
	080H		080h~0FFh of Bank 0 store system registers.
	"	***System Register***	
	"		
	"		
	0FFH		End of Bank 0
BANK 1	100H		RAM Bank 1
	"	***General Purpose Area***	
	"		
	"		
	1FFH		End of Bank 1
BANK 2	200H		RAM Bank 2
	"	***General Purpose Area***	
	"		
	"		
	2FFH		End of Bank 2
BANK 3	300H		RAM Bank 3
	"	***General Purpose Area***	
	"		
	"		
	3FFH		End of Bank 3
BANK 4	400H		RAM Bank 4
	"	***General Purpose Area***	
	"		
	"		
	47FH		End of Bank 4

The 128-byte general purpose RAM is separated into Bank 0. Sonix provides “Bank 0” type instructions (e.g. b0mov, b0add, b0bts1, b0bset…) to control Bank 0 RAM directly.

☞ ***256 X 8-bit RAM***

Endpoints	Address	RAM Location	
EP 0	500H “ “ “ 53FH	***General Purpose Area***	EP0 FIFO address always starts from 0x500, and has the depth of 64-byte.
EP1~EP6	540H “ “ “ 5FFH	***EP1 ~ EP6 FIFO Area***	EP1~EP6 share with 192-byte. Each EP's FIFO start address and its depth are controlled by USB registers.

2.2.1 SYSTEM REGISTER

2.2.1.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	X	PFLAG	RBANK	W0	W1	W2	W3	W4	W5	W6	W7
9	@HL	@YZ	UPLL	PCL	PCH	OSCM	WDTR	INTRQ0	INTRQ1	INTEN0	INTEN1	P0W	P1W	P0IE	P1IE	PEDGE
A	P0M	P1M	P2M	P4M	P5M	P0	P1	P2	P4	P5	P0UR	P1UR	P2UR	P4UR	P5UR	PECMD
B	SIOM	SIOR	SIOB	T0M	T0C	T1M	T1C_L	T1C_H	TC0M	TC0C	TC0R	TC0D	TC1M	TC1C	TC1R	TC1D
C	TC2M	TC2C	TC2R	TC2D	PWCH	PWES	P2CON	P4CON	ADM	ADB	ADR	ADT	UDA	USTATUS	USTATUS1	USB_INT_EN
D	EP_ACK	EP_NAK	UCFG	UEPDIR	UEPISO	UE0R	UE0R_C	UE1R	UE1R_C	UE2R	UE2R_C	UE3R	UE3R_C	UE4R	UE4R_C	UE5R
E	UE5R_C	UE6R	UE6R_C	UPID	UFRMNO_L	UFRMNO_H	UFRECTL	-	UTOGGLE	EP2FIFO_ADDR	EP3FIFO_ADDR	EP4FIFO_ADDR	EP5FIFO_ADDR	EP6FIFO_ADDR	PS2M	STKP
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 SYSTEM REGISTER DESCRIPTION

H, L = Working, @HL addressing register.
R = Working register and ROM look-up data buffer
Y, Z = Working, @YZ and ROM addressing register
X = Working and ROM address register
PFLAG = Special flag register.
RBANK = RAM bank selection register
W0~W7= Working register.
UPLL = Clock switch.
PCH, PCL = Program counter
OSCM = Oscillator mode register
WDTR = Watchdog timer clear register.
INTRQ0,1 = Interrupt request register
INTEN0,1 = Interrupt enable register
PnW= Port n wakeup control register
PnIE = Port n interrupt enable register
PEDGE = P1.6, P5.0 edge direction register.
PnM = Port n input/output mode register.
Pn = Port n data buffer.
PnUR = Port n pull-up resister control register
PECMD= ISP command register.
SIOM = SIO mode control register.
SIOR = SIO clock rate control register.
SIOB = SIO data buffer.
T0M = T0 mode register.
T0C = T0 counting register.
T1M = T1 mode register.
T1C_L,H = T1 counting register.
TC0M = TC0 mode register.
TC0C = TC0 counting register.
TC0R = TC0 auto-reload data buffer.
TC0D = TC0 duty control register.
TC1M = TC1 mode register.
TC1C = TC1 counting register.
TC1R = TC1 auto-reload data buffer.
TC1D = TC1 duty control register.
TC2M = TC2 mode register.
TC2C = TC2 counting register.
TC2R = TC2 auto-reload data buffer.
TC2D = TC2 duty control register.
PWCH = PWM output channel control register.
PWES = PWM extension control register.
P2CON,P4CON = P2, P4 configuration register.
ADM = ADC mode register.
ADB = ADC data buffer.
ADR = ADC resolution select register.
ADT = ADC reference voltage registers.
UDA= USB control register.
USTATUS, 1= USB status register
USB_INT_EN = USB interrupt enable/disable control register.
EP_ACK= Endpoint ACK flag register.
EP_NAK= Endpoint NAK flag register.
UCFG= USB hardware control register.
UEPDIR= Endpoint IN/OUT direction setting.
UEPISO= Endpoint ISO mode setting.
UEnR= EPn control registers.
UEnR_C= EPn byte counter register
UPID= USB bus control register.
UFRMNO_L, H= USB Frame Number register.
UFRECTL= IHRC frequency control function.
UTOGGLE= USB endpoint toggle bit control register.
EPnFIFO_ADDR= EPn FIFO start address of USB FIFO.
PS2M= PS2 control register.
STKP = Stack pointer buffer.
STK0~STK7 = Stack 0 ~ stack 7 buffer.

2.2.1.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
085H	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0	R/W	X
086H	POR	WDT	RST	STKOV		C	DC	Z	R/W	PFLAG
087H						RBNKS2	RBNKS1	RBNKS0	R/W	RBANK
088H	W0BIT7	W0BIT6	W0BIT5	W0BIT4	W0BIT3	W0BIT2	W0BIT1	W0BIT0	R/W	W0
089H	W1BIT7	W1BIT6	W1BIT5	W1BIT4	W1BIT3	W1BIT2	W1BIT1	W1BIT0	R/W	W1
08AH	W2BIT7	W2BIT6	W2BIT5	W2BIT4	W2BIT3	W2BIT2	W2BIT1	W2BIT0	R/W	W2
08BH	W3BIT7	W3BIT6	W3BIT5	W3BIT4	W3BIT3	W3BIT2	W3BIT1	W3BIT0	R/W	W3
08CH	W4BIT7	W4BIT6	W4BIT5	W4BIT4	W4BIT3	W4BIT2	W4BIT1	W4BIT0	R/W	W4
08DH	W5BIT7	W5BIT6	W5BIT5	W5BIT4	W5BIT3	W5BIT2	W5BIT1	W5BIT0	R/W	W5
08EH	W6BIT7	W6BIT6	W6BIT5	W6BIT4	W6BIT3	W6BIT2	W6BIT1	W6BIT0	R/W	W6
08FH	W7BIT7	W7BIT6	W7BIT5	W7BIT4	W7BIT3	W7BIT2	W7BIT1	W7BIT0	R/W	W7
090H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
091H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
092H		EHS_EN	IHRC_EN	EHSRDY	UPLLRDY	UPLLSRC1	UPLLSRC0	UPLLEN		UPLL
093H	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
094H			PC13	PC12	PC11	PC10	PC9	PC8	R/W	PCH
095H				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
096H	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
097H	T0IRQ	P1IRQ	P0IRQ	USBIRQ	UPDETIRQ	P50IRQ	P16IRQ	WAKEIRQ	R/W	INTRQ0
098H		SIOIRQ	ADCIRQ	NDTIRQ	TC2IRQ	TC1IRQ	TC0IRQ	T1IRQ	R/W	INTRQ1
099H	T0IEN	P1IEN	P0IEN	USBIEN	UPDETIEN	P50IEN	P16IEN	WAKEIEN	R/W	INTEN0
09AH		SIOIEN	ADCIEN	NDTIEN	TC2IEN	TC1IEN	TC0IEN	T1IEN	R/W	INTEN1
09BH	P07W	P06W	P05W	P04W	P03W	P02W	P01W	P00W	R/W	P0W
09CH	P17W		P15W	P14W	P13W	P12W	P11W	P10W	R/W	P1W
09DH	P07IE	P06IE	P05IE	P04IE	P03IE	P02IE	P01IE	P00IE	R/W	P0IE
09EH	P17IE		P15IE	P14IE	P13IE	P12IE	P11IE	P10IE	R/W	P1IE
09FH	UPDETS	UPDET_EN	P20VSS_EN		P50G1	P50G0	P16G1	P16G0	R/W	PEDGE
0A0H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0A1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0A2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M		R/W	P2M
0A3H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0A4H		P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P4M
0A5H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0A6H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0A7H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2
0A8H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4
0A9H		P56	P55	P54	P53	P52	P51	P50	R/W	P5
0AAH	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R	R/W	P0UR
0ABH	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	R/W	P1UR
0ACH	P27R	P26R	P25R	P24R	P23R	P22R	P21R		R/W	P2UR
0ADH	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	R/W	P4UR
0AEH		P56R	P55R	P54R	P53R	P52R	P51R	P50R	R/W	P5UR
0AFH	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0	W	PECMD
0B0H	SENB	START	SRATE1	SRATE0	MLSB	SCLKMD	CPOL	CPHA	R/W	SIOM
0B1H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	W	SIOR
0B2H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB
0B3H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0B4H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0B5H	T1ENB	T1rate2	T1rate1	T1rate0	T1CKS	T1ARC	T1G1	T1G0	R/W	T1M
0B6H	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0	R/W	T1C_L
0B7H	T1C15	T1C14	T1C13	T1C12	T1C11	T1C10	T1C9	T1C8	R/W	T1C_H
0B8H	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS1	TC0CKS0	TC0PO	PWM0OUT	R/W	TC0M
0B9H	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0BAH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0BBH	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0	R/W	TC0D
0BCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0		TC1CKS0	TC1PO	PWM1OUT	R/W	TC1M
0BDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0BEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0BFH	TC1D7	TC1D6	TC1D5	TC1D4	TC1D3	TC1D2	TC1D1	TC1D0	R/W	TC1D
0C0H	TC2ENB	TC2rate2	TC2rate1	TC2rate0		TC2CKS0	TC2PO	PWM2OUT	R/W	TC2M
0C1H	TC2C7	TC2C6	TC2C5	TC2C4	TC2C3	TC2C2	TC2C1	TC2C0	R/W	TC2C

0C2H	TC2R7	TC2R6	TC2R5	TC2R4	TC2R3	TC2R2	TC2R1	TC2R0	W	TC2R
0C3H	TC2D7	TC2D6	TC2D5	TC2D4	TC2D3	TC2D2	TC2D1	TC2D0	R/W	TC2D
0C4H			PWCH20	PWCH21	PWCH11	PWCH10	PWCH01	PWCH00	R/W	PWCH
0C5H					PW2ES1	PW2ES0	PW1ES1	PW1ES0	R/W	PWES
0C6H	P2CON7	P2CON6	P2CON5	P2CON4	P2CON3	P2CON2	P2CON1	P2CON0	R/W	P2CON
0C7H	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	R/W	P4CON
0C8H	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0	R/W	ADM
0C9H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0CAH	AIN0IG_EN	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0CBH			EVHENB						R/W	ADT
0CCH	UDE	UDA6	UDA5	UDA4	UDA3	UDA2	UDA1	UDA0	R/W	UDA
0CDH	ERR_TIMEOUT	ERR_SETUP	EP0_PRESETUP	EP0_OUT_STALL	EP0_IN_STALL	EP0SETUP	EP0IN	EP0OUT	R/W	USTATUS
0CEH	NDT2	NDT1		SOF	BUS_RESUME	BUS_WAKEUP	BUS_RST	SUSPEND	R/W	USTATUS1
0CFH			SOF_IE	EPN_ACK_IE	EPN_NAK_IE	BUS_IE	BUSWK_IE	USB_IE	R/W	USB_INT_EN
0D0H		EP6_ACK	EP5_ACK	EP4_ACK	EP3_ACK	EP2_ACK	EP1_ACK	EP0_ACK	R/W	EP_ACK
0D1H		EP6_NAK	EP5_NAK	EP4_NAK	EP3_NAK	EP2_NAK	EP1_NAK	EP0_NAK	R/W	EP_NAK
0D2H	ULVD36_EN	VREG33_EN	PHY_EN	DPPU_EN	ESD_EN	USGIG_EN	VREG33DIS_EN	USBRAM_EN	R/W	UCFG
0D3H			UE6D	UE5D	UE4D	UE3D	UE2D	UE1D	R/W	UEPDIR
0D4H			UE6ISO	UE5ISO	UE4ISO	UE3ISO	UE2ISO	UE1ISO	R/W	UEPISO
0D5H	UE0E	UE0M1	UE0M0	OUT_STALL_EN	IN_STALL_EN				R/W	UE0R
0D6H		UE0C6	UE0C5	UE0C4	UE0C3	UE0C2	UE0C1	UE0C0	R/W	UE0R_C
0D7H	UE1E	UE1M1	UE1M0						R/W	UE1R
0D8H	UE1C7	UE1C6	UE1C5	UE1C4	UE1C3	UE1C2	UE1C1	UE1C0	R/W	UE1R_C
0D9H	UE2E	UE2M1	UE2M0						R/W	UE2R
0DAH	UE2C7	UE2C6	UE2C5	UE2C4	UE2C3	UE2C2	UE2C1	UE2C0	R/W	UE2R_C
0DBH	UE3E	UE3M1	UE3M0						R/W	UE3R
0DCH	UE3C7	UE3C6	UE3C5	UE3C4	UE3C3	UE3C2	UE3C1	UE3C0	R/W	UE3R_C
0DDH	UE4E	UE4M1	UE4M0						R/W	UE4R
0DEH	UE4C7	UE4C6	UE4C5	UE4C4	UE4C3	UE4C2	UE4C1	UE4C0	R/W	UE4R_C
0DFH	UE5E	UE5M1	UE5M0						R/W	UE5R
0E0H	UE5C7	UE5C6	UE5C5	UE5C4	UE5C3	UE5C2	UE5C1	UE5C0	R/W	UE5R_C
0E1H	UE6E	UE6M1	UE6M0						R/W	UE6R
0E2H	UE6C7	UE6C6	UE6C5	UE6C4	UE6C3	UE6C2	UE6C1	UE6C0	R/W	UE6R_C
0E3H						UBDE	DDP	DDN	R/W	UPID
0E4H	FRMNO7	FRMNO6	FRMNO5	FRMNO4	FRMNO3	FRMNO2	FRMNO1	FRMNO0	R	UFRMNO_L
0E5H						FRMNO10	FRMNO9	FRMNO8	R	UFRMNO_H
0E6H	PHYPRM_EN									PHYPRM
0E8H			EP6_DATA01	EP5_DATA01	EP4_DATA01	EP3_DATA01	EP2_DATA01	EP1_DATA01	R/W	UTOGGLE
0E9H	EP2FIFO7	EP2FIFO6	EP2FIFO5	EP2FIFO4	EP2FIFO3	EP2FIFO2	EP2FIFO1	EP2FIFO0	R/W	EP2FIFO_ADDR
0EAH	EP3FIFO7	EP3FIFO6	EP3FIFO5	EP3FIFO4	EP3FIFO3	EP3FIFO2	EP3FIFO1	EP3FIFO0	R/W	EP3FIFO_ADDR
0EBH	EP4FIFO7	EP4FIFO6	EP4FIFO5	EP4FIFO4	EP4FIFO3	EP4FIFO2	EP4FIFO1	EP4FIFO0	R/W	EP4FIFO_ADDR
0ECH	EP5FIFO7	EP5FIFO6	EP5FIFO5	EP5FIFO4	EP5FIFO3	EP5FIFO2	EP5FIFO1	EP5FIFO0	R/W	EP5FIFO_ADDR
0EDH	EP6FIFO7	EP6FIFO6	EP6FIFO5	EP6FIFO4	EP6FIFO3	EP6FIFO2	EP6FIFO1	EP6FIFO0	R/W	EP6FIFO_ADDR
0EEH	PS2ENB					SDA	SDAM	SCKM	R/W	PS2M
0EFH	GIE	LVD24	LVD33	ULVD36		STKPB2	STKPB1	STKPB0	R/W	STKP
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H					S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H					S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H					S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H					S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H					S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH					S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH					S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH					S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H
0FFH					S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ ***Note:***

1. *To avoid system error, make sure to put all the "0" and "1" as it indicates in the above table.*
2. *All of register names had been declared in SN8ASM assembler.*
3. *One-bit name had been declared in SN8ASM assembler with "F" prefix code.*
4. *"b0bset", "b0bclr", "bset", "bclr" instructions are only available to the "R/W" registers.*

2.2.2 ACCUMULATOR

The ACC is an 8-bit data register responsible for transferring or manipulating data between ALU and data memory. If the result of operating is zero (Z) or there is carry (C or DC) occurrence, then these flags will be set to PFLAG register. ACC is not in data memory (RAM), so ACC can't be access by "B0MOV" instruction during the instant addressing mode.

➢ **Example: Read and write ACC value.**

```
; Read ACC data and store in BUF data memory

                MOV         BUF, A

; Write a immediate data into ACC

                MOV         A, #0FH

; Write ACC data from BUF data memory

                MOV         A, BUF
```

The system will store ACC and working registers (0x80-0x8F) by hardware automatically when interrupt executed.

➢ **Example: Protect ACC and working registers.**

```
.CODE
INT_SERVICE:
                                        ; Save ACC to buffer.
                                        ; Save working registers to buffer.
                …           .
                …
                                        ; Load working registers form buffers.
                                        ; Load ACC form buffer.

                RETI                    ; Exit interrupt service vector
```

2.2.3 PROGRAM FLAG

The PFLAG register contains the arithmetic status of ALU operation, system reset status and LVD detecting status. POR, WDT, and RST bits indicate system reset status including power on reset, LVD reset, reset by external pin active and watchdog reset. C, DC, Z bits indicate the result status of ALU operation. LVD24, LVD33 bits indicate LVD detecting power voltage status.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	RST	STKOV	-	C	DC	Z
Read/Write	R	R	R	R	-	R/W	R/W	R/W
After Reset	-	-	-	-	-	0	0	0

Bit 7 **POR:** Power on reset and LVD brown-out reset indicator.
0 = Non-active.
1 = Reset active. LVD announces reset flag.

Bit 6 **WDT:** Watchdog reset indicator.
0 = Non-active.
1 = Reset active. Watchdog announces reset flag.

Bit 5 **RST:** External reset indicator.
0 = Non-active.
1 = Reset active. External reset announces reset flag.

Bit 4 **STKOV:** Stack overflow indicator.
0 = Non-overflow.
1 = Stack overflow.

Bit 2 **C:** Carry flag
1 = Addition with carry, subtraction without borrowing, rotation with shifting out logic “1”, comparison result ≥ 0.
0 = Addition without carry, subtraction with borrowing signal, rotation with shifting out logic “0”, comparison result < 0.

Bit 1 **DC:** Decimal carry flag
1 = Addition with carry from low nibble, subtraction without borrow from high nibble.
0 = Addition without carry from low nibble, subtraction with borrow from high nibble.

Bit 0 **Z:** Zero flag
1 = The result of an arithmetic/logic/branch operation is zero.
0 = The result of an arithmetic/logic/branch operation is not zero.

2.2.4 PROGRAM COUNTER

The program counter (PC) is a 14-bit binary counter separated into the high-byte 6 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

Besides, it can be replaced with specific address by executing CALL or JMP instruction. When JMP or CALL instruction is executed, the destination address will be inserted to bit 0 ~ bit 13.

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

☞ **ONE ADDRESS SKIPPING**

There are nine instructions (CMPRS, INCS, INCMS, DECS, DECMS, BTS0, BTS1, B0BTS0, B0BTS1) with one address skipping function. If the result of these instructions is true, the PC will add 2 steps to skip next instruction.

If the condition of bit test instruction is true, the PC will add 2 steps to skip next instruction.

```
                B0BTS1   FC          ; To skip, if Carry_flag = 1
                JMP      C0STEP      ; Else jump to C0STEP.
                …
                …
C0STEP:         NOP

                B0MOV    A, BUF0     ; Move BUF0 value to ACC.
                B0BTS0   FZ          ; To skip, if Zero flag = 0.
                JMP      C1STEP      ; Else jump to C1STEP.
                …
                …
C1STEP:         NOP
```

If the ACC is equal to the immediate data or memory, the PC will add 2 steps to skip next instruction.

```
                CMPRS    A, #12H     ; To skip, if ACC = 12H.
                JMP      C0STEP      ; Else jump to C0STEP.
                …
                …
C0STEP:         NOP
```

If the destination increased by 1, which results overflow of 0xFF to 0x00, the PC will add 2 steps to skip next instruction.

INCS instruction:

```
                INCS     BUF0
                JMP      C0STEP        ; Jump to C0STEP if ACC is not zero.
                …
                …
C0STEP:         NOP
```

INCMS instruction:

```
                INCMS    BUF0
                JMP      C0STEP        ; Jump to C0STEP if BUF0 is not zero.
                …
                …
C0STEP:         NOP
```

If the destination decreased by 1, which results underflow of 0x01 to 0x00, the PC will add 2 steps to skip next instruction.

DECS instruction:

```
                DECS     BUF0
                JMP      C0STEP        ; Jump to C0STEP if ACC is not zero.
                …
                …
C0STEP:         NOP
```

DECMS instruction:

```
                DECMS    BUF0
                JMP      C0STEP        ; Jump to C0STEP if BUF0 is not zero.
                …
                …
C0STEP:         NOP
```

MULTI-ADDRESS JUMPING

Users can jump around the multi-address by either JMP instruction or ADD M, A instruction (M = PCL) to activate multi-address jumping function. Program Counter supports **"ADD M,A"**, **"ADC M,A"** and **"B0ADD M,A"** instructions for carry to PCH when PCL overflow automatically. For jump table or others applications, users can calculate PC value by the three instructions and don't care PCL overflow problem.

> ✸ ***Note: PCH only support PC up counting result and doesn't support PC down counting. When PCL is carry after PCL+ACC, PCH adds one automatically. If PCL borrow after PCL–ACC, PCH keeps value and not change.***

- **Example: If PC = 0323H (PCH = 03H, PCL = 23H)**

```
; PC = 0323H
                MOV       A, #28H
                B0MOV     PCL, A          ; Jump to address 0328H
                …

; PC = 0328H
                MOV       A, #00H
                B0MOV     PCL, A          ; Jump to address 0300H
                …
```

- **Example: If PC = 0323H (PCH = 03H, PCL = 23H)**

```
; PC = 0323H
                B0ADD     PCL, A          ; PCL = PCL + ACC, the PCH cannot be changed.
                JMP       A0POINT         ; If ACC = 0, jump to A0POINT
                JMP       A1POINT         ; ACC = 1, jump to A1POINT
                JMP       A2POINT         ; ACC = 2, jump to A2POINT
                JMP       A3POINT         ; ACC = 3, jump to A3POINT
                …
                …
```

2.2.5 H, L REGISTERS

The H and L registers are the 8-bit buffers. There are two major functions of these registers.

- Can be used as general working registers
- Can be used as RAM data pointers with @HL register

081H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

080H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
L	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

➢ **Example: If want to read a data from RAM address 20H of bank_0, it can use indirectly addressing mode to access data as following.**

```
		B0MOV	H, #00H		; To set RAM bank 0 for H register
		B0MOV	L, #20H		; To set location 20H for L register
		B0MOV	A, @HL		; To read a data into ACC
```

➢ **Example: Clear general-purpose data memory area of bank 0 using @HL register.**

```
		CLR	H		; H = 0, bank 0
		B0MOV	L, #07FH	; L = 7FH, the last address of the data memory area
CLR_HL_BUF:
		CLR	@HL		; Clear @HL to be zero
		DECMS	L		; L – 1, if L = 0, finish the routine
		JMP	CLR_HL_BUF	; Not zero

		CLR	@HL
END_CLR:				; End of clear general purpose data memory area of bank 0
		…
		…
```

2.2.6 Y, Z REGISTERS

The Y and Z registers are the 8-bit buffers. There are three major functions of these registers.

- Can be used as general working registers
- Can be used as RAM data pointers with @YZ register
- Can be used as ROM data pointer with the MOVC instruction for look-up table

084H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Y	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

083H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Z	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

➢ **Example: Uses Y, Z register as the data pointer to access data in the RAM address 025H of bank0.**

```
            B0MOV      Y, #00H          ; To set RAM bank 0 for Y register
            B0MOV      Z, #25H          ; To set location 25H for Z register
            B0MOV      A, @YZ           ; To read a data into ACC
```

➢ **Example: Uses the Y, Z register as data pointer to clear the RAM data.**

```
            B0MOV      Y, #0            ; Y = 0, bank 0
            B0MOV      Z, #07FH         ; Z = 7FH, the last address of the data memory area

CLR_YZ_BUF:
            CLR        @YZ              ; Clear @YZ to be zero

            DECMS      Z                ; Z – 1, if Z= 0, finish the routine
            JMP        CLR_YZ_BUF       ; Not zero

            CLR        @YZ
END_CLR:                                ; End of clear general purpose data memory area of bank 0
            …
```

2.2.7 R REGISTER

R register is an 8-bit buffer. There are two major functions of the register.

- Can be used as working register
- For store high-byte data of look-up table
  (MOVC instruction executed, the high-byte data of specified ROM address will be stored in R register and the low-byte data will be stored in ACC).

082H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

> ✸ ***Note: Please refer to the "LOOK-UP TABLE DESCRIPTION" about R register look-up table application.***

2.2.8 X REGISTERS

X register is an 8-bit buffer and only general working register purpose.

- Can be used as general working registers

085H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

2.2.9 System Control Registers

UPLL register is an 8-bit buffer and only general working register purpose.

- Can be used as general working registers

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UPLL	-	EHS_EN	IHRC_EN	EHSRDY	UPLLRDY	UPLLSRC1	UPLLSRC0	UPLLEN
Read/Write	-	R/W	R/W	R	R	R/W	R/W	R/W
After reset	-	0	0	0	0	0	0	0

Bit 6 **EHS_EN:** External high-speed clock enable.
0 = Disable EHS X'TAL.
1 = Enable EHS X'TAL. HW will enable Noise filter automatically.

Bit 5 **IHRC_EN:** Internal high-speed clock enable.
0 = Disable internal 16 MHz RC oscillator.
1 = Enable internal 16 MHz RC oscillator.

Bit 4 **EHSRDY:** EHX X'tal clock ready flag.
0 = EHS X'tal clock is not ready
1 = EHS X'tal clock is ready.

Bit 3 **UPLLRDY:** PLL clock ready flag.
0 = PLL locked is not ready
1 = PLL locked ready.

Bit [2:1] **UPLLSRC[1:0]:** System PLL clock source.

00 = IHRC 16 MHz oscillator
01 = IHRC 16 MHz oscillator
10 = EHS X'TAL 12MHz
11 = EHS X'TAL 16MHz

Bit 0 **UPLLEN:** PLL enable
0 = Disable
1 = Enable

2.3 ADDRESSING MODE

2.3.1 IMMEDIATE ADDRESSING MODE

The immediate addressing mode uses an immediate data to set up the location in ACC or specific RAM.

➢ **Example: Move the immediate data 12H to ACC.**

```
MOV	A, #12H		; To set an immediate data 12H into ACC.
```

➢ **Example: Move the immediate data 12H to R register.**

```
B0MOV	R, #12H		; To set an immediate data 12H into R register.
```

✸ ***Note: In immediate addressing mode application, the specific RAM must be 0x80~0x8F working register.***

2.3.2 DIRECTLY ADDRESSING MODE

The directly addressing mode moves the content of RAM location in or out of ACC.

➢ **Example: Move 0x12 RAM location data into ACC.**

```
B0MOV	A, 12H		; To get a content of RAM location 0x12 of bank 0 and save in
			ACC.
```

➢ **Example: Move ACC data into 0x12 RAM location.**

```
B0MOV	12H, A		; To get a content of ACC and save in RAM location 12H of
			bank 0.
```

2.3.3 INDIRECTLY ADDRESSING MODE

The indirectly addressing mode is to access the memory by the data pointer registers (H/L, Y/Z).

Example: Indirectly addressing mode with @HL register

```
B0MOV	H, #0		; To clear H register to access RAM bank 0.
B0MOV	L, #12H		; To set an immediate data 12H into L register.
B0MOV	A, @HL		; Use data pointer @HL reads a data from RAM location
			; 012H into ACC.
```

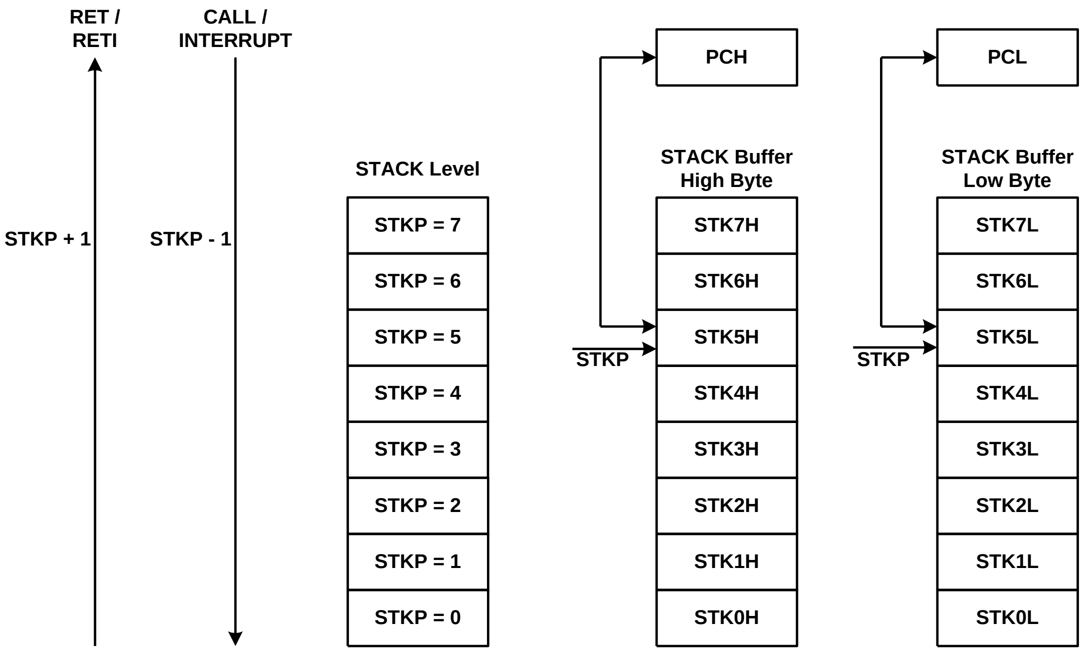
Example: Indirectly addressing mode with @YZ register

```
B0MOV	Y, #0		; To clear Y register to access RAM bank 0.
B0MOV	Z, #12H		; To set an immediate data 12H into Z register.
B0MOV	A, @YZ		; Use data pointer @YZ reads a data from RAM location
			; 012H into ACC.
```

2.4 STACK OPERATION

2.4.1 OVERVIEW

The stack buffer has 8-level. These buffers are designed to push and pop up program counter's (PC) data when interrupt service routine and "CALL" instruction are executed. The STKP register is a pointer designed to point active level in order to push or pop up data from stack buffer. The STKnH and STKnL are the stack buffers to store program counter (PC) data.



2.4.2 STACK POINTER

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 13-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses. The two stack operations are writing to the top of the stack (push) and reading from the top of stack (pop). Push operation decrements the STKP and the pop operation increments each time. That makes the STKP always point to the top address of stack buffer and write the last program counter value (PC) into the stack buffer.

0EFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	LVD24	LVD33	ULVD36	-	STKPB2	STKPB1	STKPB0
Read/Write	R/W	R	R	R	-	R/W	R/W	R/W
After reset	0	-	-	-	-	1	1	1

Bit[2:0] **STKPBn:** Stack pointer (n = 0 ~ 2)

➢ **Example: Stack pointer (STKP) reset, we strongly recommended to clear the stack pointers in the beginning of the program.**

```
MOV     A, #00000111B
B0MOV   STKP, A
```

2.4.3 STACK BUFFER

The program counter (PC) value is stored in the stack buffer before a CALL instruction executed or during interrupt service routine. Stack operation is a LIFO type (Last in and first out). The stack pointer (STKP) and stack buffer (STKnH and STKnL) are located in the system register area bank 0.

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	-	-	SnPC11	SnPC10	SnPC9	SnPC8
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnL	SnPC7	SnPC6	SnPC5	SnPC4	SnPC3	SnPC2	SnPC1	SnPC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

STKn = STKnH , STKnL (n = 7 ~ 0)

2.4.4 STACK OVERFLOW INDICATOR

If stack pointer is normal and not overflow, the program execution is correct. If stack overflows, the program counter would be incorrect making program execution error. STKOV bit is stack pointer overflow indicator to monitor stack pointer status. When STKOV=0, stack pointer status is normal. If STKOV=1, stack overflow occurs, and the program execution would be error. The program can take measures to recover program execution from stack overflow situation through STKOV bit.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	RST	STKOV	-	C	DC	Z
Read/Write	R	R	R	R	-	R/W	R/W	R/W
After Reset	-	-	-	-	-	0	0	0

Bit 4 **STKOV:** Stack overflow indicator.
0 = Non-overflow.
1 = Stack overflow.

> ✸ ***Note: If STKOV bit is set as stack overflowing, only system reset event can clear STKOV bit, e.g. watchdog timer overflow, external reset pin low status or LVD reset.***

➢ **Example: Stack overflow protection through watchdog reset. Watchdog timer must be enabled.**

```
MAIN:
                …
StackChk:
                B0BTS1    STKOV
                JMP       MAIN          ; STKOV=0, program keeps executing.
                JMP       $             ; STKOV=1, stack overflows, and use “jump here” operation
                                        ; making watchdog timer overflow to trigger system reset.
```

➢ **Example: Stack overflow protection through external reset. External reset function must be enabled, and one GPIO pin (output mode) connects to external reset pin.**

```
MAIN:
                …
StackChk:
                B0BTS1    STKOV
                JMP       MAIN          ; STKOV=0, program keeps executing.
                B0BCLR    P1.0          ; STKOV=1, stack overflows, and set P1.0 output low status to
                                        ; force reset pin to low status to trigger system reset.
```

2.4.5 STACK OPERATION EXAMPLE

The two kinds of Stack-Save operations refer to the stack pointer (STKP) and write the content of program counter (PC) to the stack buffer are CALL instruction and interrupt service. Under each condition, the STKP decreases and points to the next available stack location. The stack buffer stores the program counter about the op-code address. The Stack-Save operation is as the following table.

Stack Level	STKP Register			Stack Buffer		STKOV	Description
	STKPB2	STKPB1	STKPB0	High Byte	Low Byte		
0	1	1	1	Free	Free	0	-
1	1	1	0	STK0H	STK0L	0	-
2	1	0	1	STK1H	STK1L	0	-
3	1	0	0	STK2H	STK2L	0	-
4	0	1	1	STK3H	STK3L	0	-
5	0	1	0	STK4H	STK4L	0	-
6	0	0	1	STK5H	STK5L	0	-
7	0	0	0	STK6H	STK6L	0	-
8	1	1	1	STK7H	STK7L	0	-
> 8	1	1	0	-	-	**1**	**Stack Over, error**

There are Stack-Restore operations correspond to each push operation to restore the program counter (PC). The RETI instruction uses for interrupt service routine. The RET instruction is for CALL instruction. When a pop operation occurs, the STKP is incremented and points to the next free stack location. The stack buffer restores the last program counter (PC) to the program counter registers. The Stack-Restore operation is as the following table.

Stack Level	STKP Register			Stack Buffer		STKOV	Description
	STKPB2	STKPB1	STKPB0	High Byte	Low Byte		
8	1	1	1	STK7H	STK7L	0	-
7	0	0	0	STK6H	STK6L	0	-
6	0	0	1	STK5H	STK5L	0	-
5	0	1	0	STK4H	STK4L	0	-
4	0	1	1	STK3H	STK3L	0	-
3	1	0	0	STK2H	STK2L	0	-
2	1	0	1	STK1H	STK1L	0	-
1	1	1	0	STK0H	STK0L	0	-
0	1	1	1	Free	Free	0	-

✸ ***Note: When stack overflow occurs, the system detects the condition and set STKOV flag (“Logic 1”). STKOV flag can’t be cleared by program.***

2.5 CODE OPTION TABLE

The code option is the system hardware configurations including system clock rate, noise filter option, watchdog timer operation, LVD option, reset pin option and Flash ROM security control. The code option items are as following table:

Code Option	Content	Function Description
High_Clk	IHRC_16M	High speed internal 16MHz RC. XIN/XOUT pins are bi-direction GPIO mode.
	16M X'tal	High speed crystal /resonator (e.g. 16MHz) for external high clock oscillator.
	12M X'tal	High speed crystal /resonator (e.g. 12MHz) for external high clock oscillator.
Fcpu	Fhosc/1	Normal mode instruction cycle is 1 high speed oscillator clocks.
	Fhosc/2	Normal mode instruction cycle is 2 high speed oscillator clocks.
	Fhosc/4	Normal mode instruction cycle is 4 high speed oscillator clocks.
	Fhosc/8	Normal mode instruction cycle is 8 high speed oscillator clocks.
Low_Fcpu	Flosc/1	Slow mode instruction cycle is 1 low speed oscillator clocks.
	Flosc/2	Slow mode instruction cycle is 2 low speed oscillator clocks.
	Flosc/4	Slow mode instruction cycle is 4 low speed oscillator clocks.
	Flosc/8	Slow mode instruction cycle is 8 low speed oscillator clocks.
Noise_Filter	Enable	Enable Noise Filter.
	Disable	Disable Noise Filter.
WDT_CLK	Flosc/4	Watchdog timer clock source Flosc/4.
	Flosc/8	Watchdog timer clock source Flosc/8.
	Flosc/16	Watchdog timer clock source Flosc/16.
	Flosc/32	Watchdog timer clock source Flosc/32.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
Reset_Pin	Reset	Enable External reset pin.
	P56	Enable P5.6
External Reset Length	No	Disable external reset de-bounce time.
	128*ILRC	Enable external reset de-bounce time.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
LVD	LVD_L	LVD18 Reset, LVD24 Off, LVD33 Off
	LVD_M	LVD18 Reset, LVD24 Flag, LVD33 Off
	LVD_H	LVD18 Reset, LVD24 Reset, LVD33 Flag
	LVD_MAX	LVD18 Reset, LVD24 Reset, LVD33 Reset

2.5.1 Fcpu code option

Fcpu means instruction cycle whose clock source includes high/low speed oscillator in different operating modes. High_Fcpu and Low_Fcpu code options select instruction cycle pre-scaler to decide instruction cycle rate. In normal mode (high speed clock), the system clock source is high speed oscillator, and Fcpu clock rate has four options including Fhosc/1, Fhosc/2, Fhosc/4, Fhosc/8. In slow mode (low speed clock), the system clock source is internal low speed RC oscillator, and the Fcpu including Flosc/1, Flosc/2, Flosc/4, Flosc/8.

2.5.2 Reset_Pin code option

The reset pin is shared with general input only pin controlled by code option.

- **Reset:** The reset pin is external reset function. When falling edge trigger occurring, the system will be reset.
- **P56:** Set reset pin to general bi-direction pin (P5.6). The external reset function is disabled and the pin is bi-direction pin.

2.5.3 Security code option

Security code option is Flash ROM protection. When enable security code option, the ROM code is secured and not dumped complete ROM contents.

2.5.4 Noise Filter code option

Noise Filter code option is a power noise filter manner to reduce noisy effect of system clock. If noise filter enable, In high noisy environment, enable noise filter, enable watchdog timer and select a good LVD level can make whole system work well and avoid error event occurrence.

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

- Power on reset
- Watchdog reset
- Brown out reset
- External reset (only supports external reset pin enable situation)

When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The POR, WDT and RST flags indicate system reset status. The system can depend on POR, WDT and RST status and go to different paths by program.

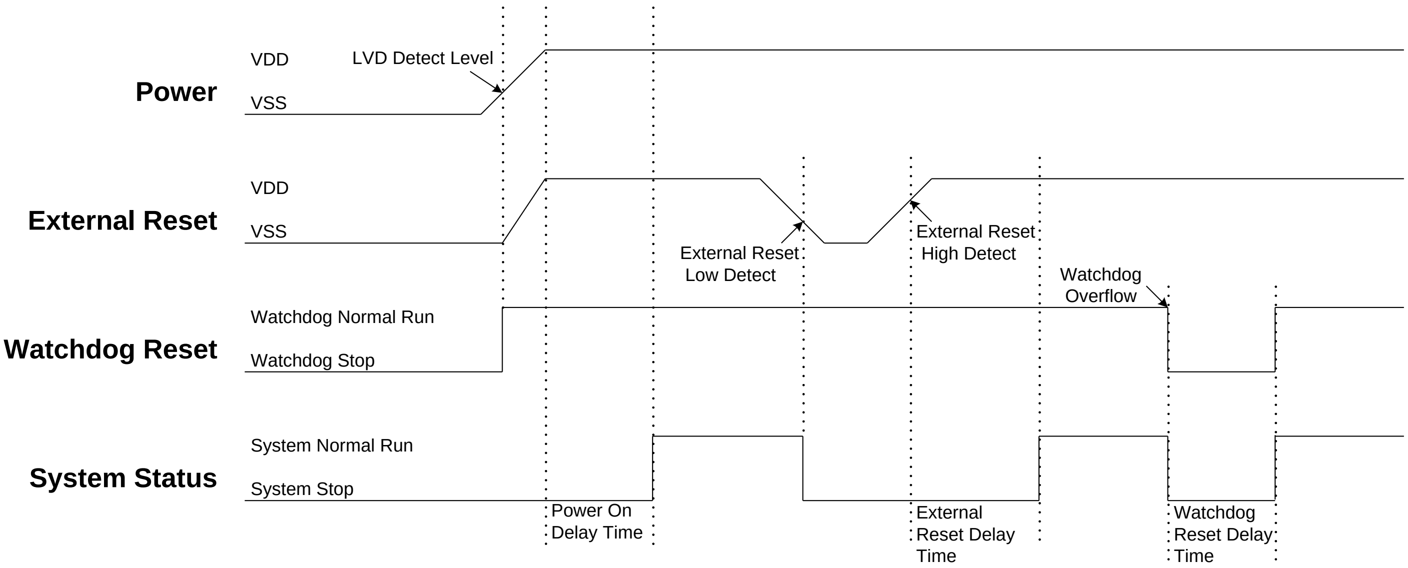
086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	RST	STKOV	-	C	DC	Z
Read/Write	R	R	R	R	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

Bit 7 **POR:** Power on reset and LVD brown-out reset indicator.
0 = Non-active.
1 = Reset active. LVD announces reset flag.

Bit 6 **WDT:** Watchdog reset indicator.
0 = Non-active.
1 = Reset active. Watchdog announces reset flag.

Bit 5 **RST:** External reset indicator.
0 = Non-active.
1 = Reset active. External reset announces reset flag.

Finishing any reset sequence needs some time. The system provides complete procedures to make the power on reset successful. For different oscillator types, the reset time is different. That causes the VDD rise rate and start-up time of different oscillator is not fixed. RC type oscillator's start-up time is very short, but the crystal type is longer. Under client terminal application, users have to take care the power on reset time for the master terminal requirement. The reset timing diagram is as following.



3.2 POWER ON RESET

The power on reset depend no LVD operation for most power-up situations. The power supplying to system is a rising curve and needs some time to achieve the normal voltage. Power on reset sequence is as following.

- **Power-up:** System detects the power voltage up and waits for power stable.
- **External reset (only external reset pin enable):** System checks external reset pin status. If external reset pin is not high level, the system keeps reset status and waits external reset pin released.
- **System initialization:** All system registers is set as initial conditions and system is ready.
- **Oscillator warm up:** Oscillator operation is successfully and supply to system clock.
- **Program executing:** Power on sequence is finished and program executes from ORG 0.

3.3 WATCHDOG RESET

Watchdog reset is a system protection. In normal condition, system works well and clears watchdog timer by program. Under error condition, system is in unknown situation and watchdog can't be clear by program before watchdog timer overflow. Watchdog timer overflow occurs and the system is reset. After watchdog reset, the system restarts and returns normal mode. Watchdog reset sequence is as following.

- **Watchdog timer status:** System checks watchdog timer overflow status. If watchdog timer overflow occurs, the system is reset.
- **System initialization:** All system registers is set as initial conditions and system is ready.
- **Oscillator warm up:** Oscillator operation is successfully and supply to system clock.
- **Program executing:** Power on sequence is finished and program executes from ORG 0.

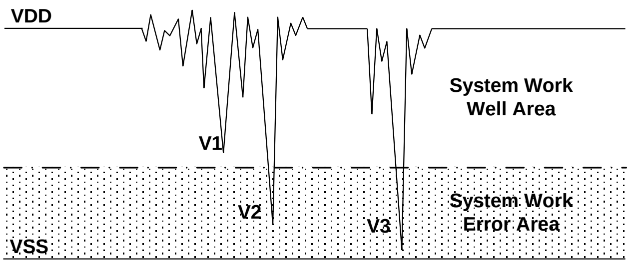
Watchdog timer application note is as following.

- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the best structure to enhance the watchdog timer function.

> ✷ ***Note: Please refer to the "WATCHDOG TIMER" about watchdog timer detail information.***

3.4 BROWN OUT RESET

The brown out reset is a power dropping condition. The power drops from normal voltage to low voltage by external factors (e.g. EFT interference or external loading changed). The brown out reset would make the system not work well or executing program error.



Brown Out Reset Diagram

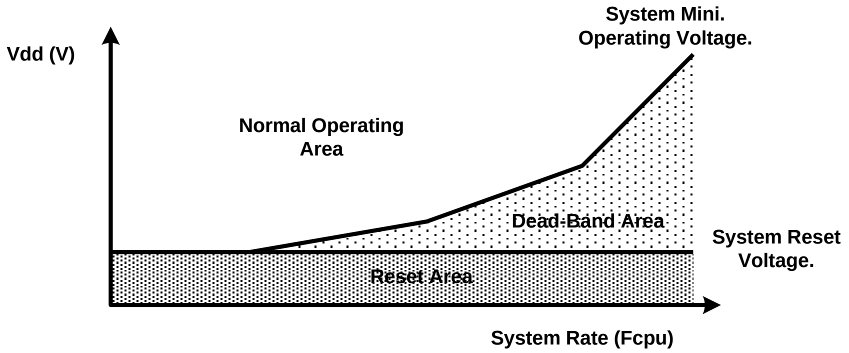
The power dropping might through the voltage range that's the system dead-band. The dead-band means the power range can't offer the system minimum operation power requirement. The above diagram is a typical brown out reset diagram. There is a serious noise under the VDD, and VDD voltage drops very deep. There is a dotted line to separate the system working area. The above area is the system work well area. The below area is the system work error area called dead-band. V1 doesn't touch the below area and not effect the system operation. But the V2 and V3 is under the below area and may induce the system error occurrence. Let system under dead-band includes some conditions.

DC application:
The power source of DC application is usually using battery. When low battery condition and MCU drive any loading, the power drops and keeps in dead-band. Under the situation, the power won't drop deeper and not touch the system reset voltage. That makes the system under dead-band.

AC application:
In AC power application, the DC power is regulated from AC power source. This kind of power usually couples with AC noise that makes the DC power dirty. Or the external loading is very heavy, e.g. driving motor. The loading operating induces noise and overlaps with the DC power. VDD drops by the noise, and the system works under unstable power situation.
The power on duration and power down duration are longer in AC application. The system power on sequence protects the power on successful, but the power down situation is like DC low battery condition. When turn off the AC power, the VDD drops slowly and through the dead-band for a while.

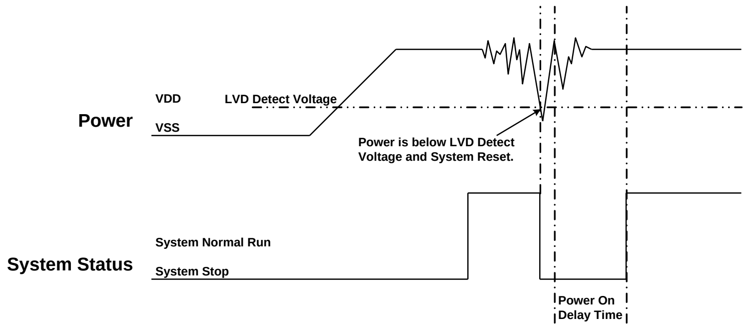
3.4.1 THE SYSTEM OPERATING VOLTAGE

To improve the brown out reset needs to know the system minimum operating voltage which is depend on the system executing rate and power level. Different system executing rates have different system minimum operating voltage. The electrical characteristic section shows the system voltage to executing rate relationship.



Normally the system operation voltage area is higher than the system reset voltage to VDD, and the reset voltage is decided by LVD detect level. The system minimum operating voltage rises when the system executing rate upper even higher than system reset voltage. The dead-band definition is the system minimum operating voltage above the system reset voltage.

3.4.2 LOW VOLTAGE DETECTOR (LVD)



The LVD (low voltage detector) is built-in Sonix 8-bit MCU to be brown out reset protection. When the VDD drops and is below LVD detect voltage, the LVD would be triggered, and the system is reset. The LVD detect level is different by each MCU. The LVD voltage level is a point of voltage and not easy to cover all dead-band range. Using LVD to improve brown out reset is depend on application requirement and environment. If the power variation is very deep, violent and trigger the LVD, the LVD can be the protection. If the power variation can touch the LVD detect level and make system work error, the LVD can't be the protection and need to other reset methods. More detail LVD information is in the electrical characteristic section.
The LVD is three levels design (1.8V/2.4V/3.3V) and controlled by LVD code option. The 1.8V LVD is always enable for power on reset and Brown Out reset. The 2.4V LVD includes LVD reset function and flag function to indicate VDD status function. The 3.3V includes flag function to indicate VDD status. LVD flag function can be an **easy low battery detector**. LVD24, LVD33 flags indicate VDD voltage level. For low battery detect application, only checking LVD24, LVD33 status to be battery status. This is a cheap and easy solution.

0EFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	LVD24	LVD33	ULVD36	-	STKPB2	STKPB1	STKPB0
Read/Write	R/W	R	R	R	-	R/W	R/W	R/W
After Reset	0	-	-	-	-	1	1	1

Bit 6 **LVD24:** LVD24 low voltage detect indicator.
0 = Vdd > LVD24 detect level.
1 = Vdd < LVD24 detect level.

Bit 5 **LVD33:** LVD33 low voltage detect indicator.
0 = Vdd > LVD33 detect level.
1 = Vdd < LVD33 detect level.

LVD	**LVD Code Option**		
	LVD_L	**LVD_M**	**LVD_H**
1.8V Reset	Available	Available	Available
2.4V Flag	-	Available	-
2.4V Reset	-	-	Available
3.3V Flag	-	-	Available

LVD_L
If VDD < 1.8V, system will be reset.
Disable LVD24 and LVD33 bit of PFLAG register.
LVD_M
If VDD < 1.8V, system will be reset.
Enable LVD24 bit of PFLAG register. If VDD > 2.4V, LVD24 is "0". If VDD <= 2.4V, LVD24 flag is "1".
Disable LVD33 bit of PFLAG register.
LVD_H
If VDD < 2.4V, system will be reset.
Enable LVD33 bit of PFLAG register. If VDD > 3.3V, LVD33 is "0". If VDD <= 3.3V, LVD33 flag is "1".
LVD_MAX
If VDD < 3.3V, system will be reset.

✸ ***Note:***
1. ***After any LVD reset, LVD24, LVD33 flags are cleared.***
2. ***The voltage level of LVD 2.4V or 3.3V is for design reference only. Don't use the LVD indicator as precision VDD measurement.***
3. ***In LVD18 reset and external reset period, the power consumption is about 60uA; In LVD24/LVD33 reset period, the power consumption is about 1~2mA.***

Bit 4 **ULVD36:** USB Power's LVD 36 indicator.
0 = USB Vdd ≦ 3.6V.
1 = USB Vdd > 3.6V.

3.4.3 BROWN OUT RESET IMPROVEMENT

How to improve the brown reset condition? There are some methods to improve brown out reset as following.

- **LVD reset**
- **Watchdog reset**
- **Reduce the system executing rate**
- **External reset circuit. (Zener diode reset circuit, Voltage bias reset circuit, External reset IC)**

> ✷ ***Note:***
> 1. ***The " Zener diode reset circuit", "Voltage bias reset circuit" and "External reset IC" can completely improve the brown out reset, DC low battery and AC slow power down conditions.***
> 2. ***For AC power application and enhance EFT performance, the system clock is 4MHz/4 (1 mips) and use external reset (" Zener diode reset circuit", "Voltage bias reset circuit", "External reset IC"). The structure can improve noise effective and get good EFT characteristic.***

Watchdog reset:
The watchdog timer is a protection to make sure the system executes well. Normally the watchdog timer would be clear at one point of program. Don't clear the watchdog timer in several addresses. The system executes normally and the watchdog won't reset system. When the system is under dead-band and the execution error, the watchdog timer can't be clear by program. The watchdog is continuously counting until overflow occurrence. The overflow signal of watchdog timer triggers the system to reset, and the system return to normal mode after reset sequence. This method also can improve brown out reset condition and make sure the system to return normal mode.
If the system reset by watchdog and the power is still in dead-band, the system reset sequence won't be successful and the system stays in reset status until the power return to normal range. Watchdog timer application note is as following.

Reduce the system executing rate:
If the system rate is fast and the dead-band exists, to reduce the system executing rate can improve the dead-band. The lower system rate is with lower minimum operating voltage. Select the power voltage that's no dead-band issue and find out the mapping system rate. Adjust the system rate to the value and the system exits the dead-band issue. This way needs to modify whole program timing to fit the application requirement.

External reset circuit:
The external reset methods also can improve brown out reset and is the complete solution. There are three external reset circuits to improve brown out reset including "Zener diode reset circuit", "Voltage bias reset circuit" and "External reset IC". These three reset structures use external reset signal and control to make sure the MCU be reset under power dropping and under dead-band. The external reset information is described in the next section.

3.5 EXTERNAL RESET

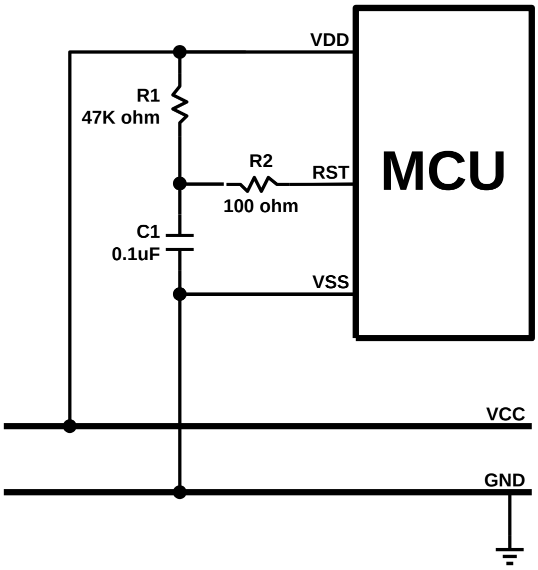
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the low voltage and the system is reset. The external reset operation actives in power on and normal running mode. During system power-up, the external reset pin must be high level input, or the system keeps in reset status. External reset sequence is as following.

- **External reset (only external reset pin enable):** System checks external reset pin status. If external reset pin is not high level, the system keeps reset status and waits external reset pin released.
- **System initialization:** All system registers is set as initial conditions and system is ready.
- **Oscillator warm up:** Oscillator operation is successfully and supply to system clock.
- **Program executing:** Power on sequence is finished and program executes from ORG 0.

The external reset can reset the system during power on duration, and good external reset circuit can protect the system to avoid working at unusual power condition, e.g. brown out reset in AC power application…

3.6 EXTERNAL RESET CIRCUIT

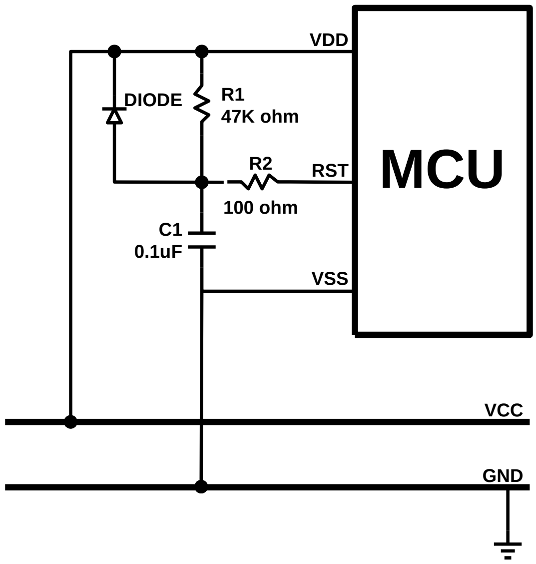
3.6.1 Simply RC Reset Circuit



This is the basic reset circuit, and only includes R1 and C1. The RC circuit operation makes a slow rising signal into reset pin as power up. The reset signal is slower than VDD power up timing, and system occurs a power on signal from the timing difference.

> ✸ ***Note: The reset circuit is no any protection against unusual power or brown out reset.***

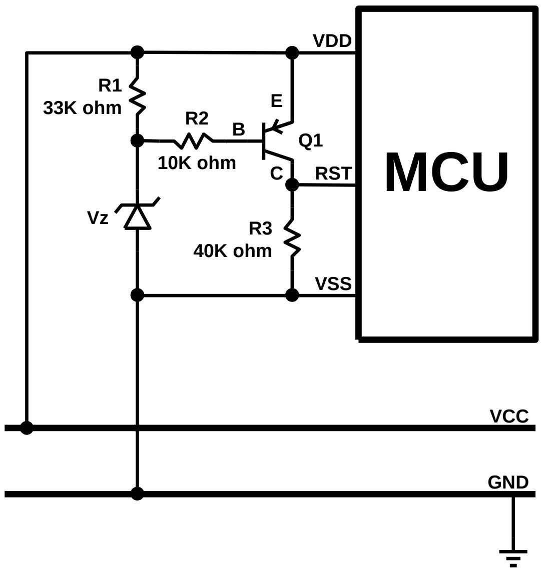
3.6.2 Diode & RC Reset Circuit



This is the better reset circuit. The R1 and C1 circuit operation is like the simply reset circuit to make a power on signal. The reset circuit has a simply protection against unusual power. The diode offers a power positive path to conduct higher power to VDD. It is can make reset pin voltage level to synchronize with VDD voltage. The structure can improve slight brown out reset condition.

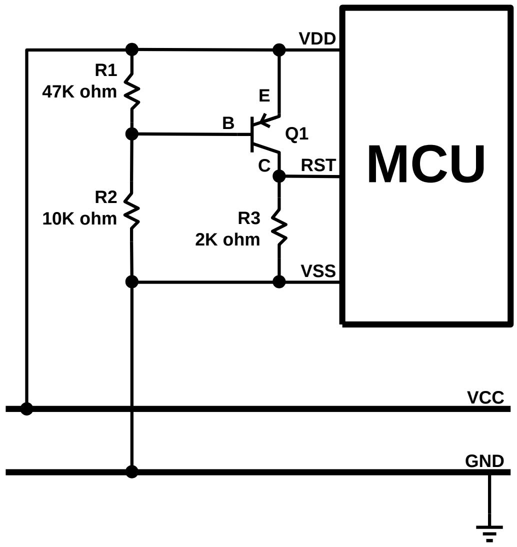
> ✸ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.6.3 Zener Diode Reset Circuit



The zener diode reset circuit is a simple low voltage detector and can **improve brown out reset condition completely**. Use zener voltage to be the active level. When VDD voltage level is above "Vz + 0.7V", the C terminal of the PNP transistor outputs high voltage and MCU operates normally. When VDD is below "Vz + 0.7V", the C terminal of the PNP transistor outputs low voltage and MCU is in reset mode. Decide the reset detect voltage by zener specification. Select the right zener voltage to conform the application.

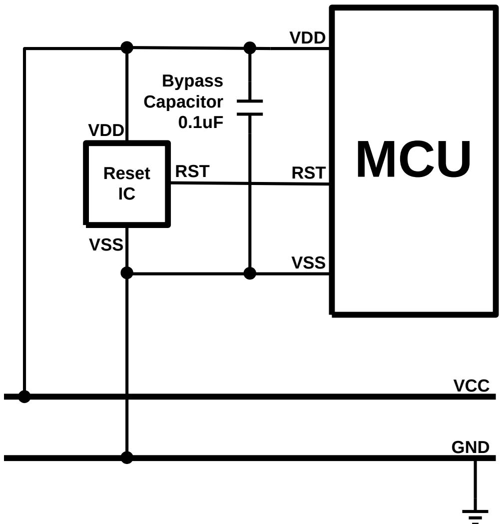
3.6.4 Voltage Bias Reset Circuit



The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to "0.7V x (R1 + R2) / R1", the C terminal of the PNP transistor outputs high voltage and MCU operates normally. When VDD is below "0.7V x (R1 + R2) / R1", the C terminal of the PNP transistor outputs low voltage and MCU is in reset mode.
Decide the reset detect voltage by R1, R2 resistances. Select the right R1, R2 value to conform the application. In the circuit diagram condition, the MCU's reset pin level varies with VDD voltage variation, and the differential voltage is 0.7V. If the VDD drops and the voltage lower than reset pin detect level, the system would be reset. If want to make the reset active earlier, set the R2 > R1 and the cap between VDD and C terminal voltage is larger than 0.7V. The external reset circuit is with a stable current through R1 and R2. For power consumption issue application, e.g. DC power system, the current must be considered to whole system power consumption.

✸ ***Note: Under unstable power condition as brown out reset, "Zener diode rest circuit" and "Voltage bias reset circuit" can protects system no any error occurrence as power dropping. When power drops below the reset detect voltage, the system reset would be triggered, and then system executes reset sequence. That makes sure the system work well under unstable power situation.***

3.6.5 External Reset IC



The external reset circuit also use external reset IC to enhance MCU reset performance. This is a high cost and good effect solution. By different application and system requirement to select suitable reset IC. The reset circuit can improve all power variation.

4 SYSTEM CLOCK

4.1 OVERVIEW

The micro-controller is a dual clock system including high-speed and low-speed clocks. The high-speed clock includes internal high-speed oscillator and external oscillators selected by “High_CLK” code option. The low-speed clock is from internal low-speed oscillator controlled by “CLKMD” bit of OSCM register. Both high-speed clock and low-speed clock can be system clock source through a divider to decide the system clock rate.

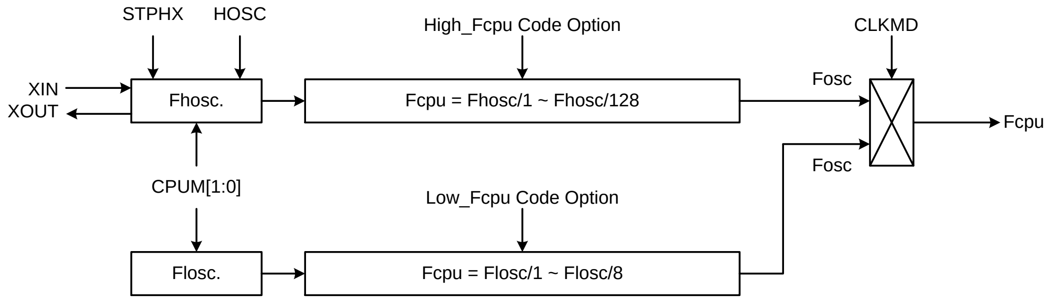
- **High-speed oscillator**

Internal high-speed oscillator is 16MHz RC type called “**IHRC16**”.
External high-speed oscillator includes crystal/ceramic (12MHz, 16MHz), RC type and External Clock.

- **Low-speed oscillator**

Internal low-speed oscillator is 32KHz RC type called “**ILRC**”.

- **System clock block diagram**



- HOSC: High_Clk code option.
- Fhosc: External high-speed clock / Internal high-speed RC clock.
- Flosc: Internal low-speed RC clock (about 32KHz@1.8V and @5V).
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.2 FCPU (INSTRUCTION CYCLE)

The system clock rate is instruction cycle called “**Fcpu**” which is divided from the system clock source and decides the system operating rate. Fcpu rate is selected by High_Fcpu code option and the range is **Fhosc/1~Fhosc/8** under system normal mode. If the system high clock source is external 4MHz crystal, and the High_Fcpu code option is Fhosc/4, the Fcpu frequency is 16MHz/4 = 4MHz. Under system slow mode, the Fcpu range is **Flosc/1~Flosc/8** controlled by Low_Fcpu code option, If Low_Fcpu code option is Flosc/4, the Fcpu frequency is 32KHz/4=8KHz.

4.3 NOISE FILTER

The Noise Filter controlled by “Noise_Filter” code option is a low pass filter and supports external oscillator including RC and crystal modes. The purpose is to filter high rate noise coupling on high clock signal from external oscillator.
In high noisy environment, enable “Noise_Filter” code option is the strongly recommendation to reduce noise effect.

4.4 SYSTEM HIGH-SPEED CLOCK

The system high-speed clock has internal and external two-type. The external high-speed clock includes 4MHz, 12MHz, crystal/ceramic, RC type and External Clock. These high-speed oscillators are selected by “**High_CLK**” code option.

4.4.1 HIGH_CLK CODE OPTION

For difference clock functions, Sonix provides multi-type system high clock options controlled by “High_CLK” code option. The High_CLK code option defines the system oscillator types including IHRC_16M, RC, 12M X’tal, 4M X’tal and Ext_CLK. These oscillator options support different bandwidth oscillator.

- **IHRC_16M:** The system high-speed clock source is internal high-speed 16MHz RC type oscillator. In the mode, XIN and XOUT pins are bi-direction GPIO mode, and not to connect any external oscillator device.
- **16M X’tal:** The system high-speed clock source is external high-speed crystal/ceramic.
- **12M X’tal:** The system high-speed clock source is external high-speed crystal/resonator.

4.4.2 INTERNAL HIGH-SPEED OSCILLATOR RC TYPE (IHRC)

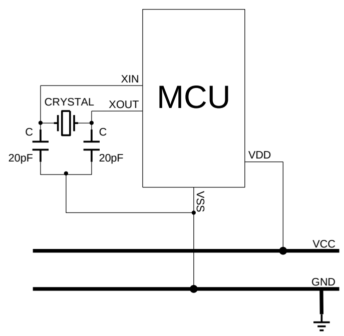
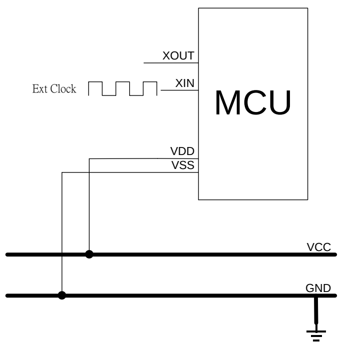
The internal high-speed oscillator is 16MHz RC type. The accuracy is ±2% under commercial condition. When the “High_CLK” code option is “IHRC_16M”, the internal high-speed oscillator is enabled.

- **IHRC_16M:** The system high-speed clock is internal 16MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.

4.4.3 EXTERNAL HIGH-SPEED OSCILLATOR

The external high-speed oscillator includes 4MHz, 12MHz, RC type and External Clock. The 4MHz and 12MHz oscillators support crystal and ceramic types connected to XIN/XOUT pins with 20pF capacitors to ground. The RC type is a low cost RC circuit only connected to XIN pin. The capacitance is not below 100pF, and use the resistance to decide the frequency. The external clock only connects to XIN pin.

4.4.4 EXTERNAL OSCILLATOR APPLICATION CIRCUIT

CRYSTAL/CERAMIC	External Clock
	

> ✷ ***Note: Connect the Crystal/Ceramic and C as near as possible to the XIN/XOUT/VSS pins of micro-controller. Connect the R and C as near as possible to the VDD pin of micro-controller.***

4.5 SYSTEM LOW-SPEED CLOCK

The system low clock source is the internal low-speed oscillator built in the micro-controller. The low-speed oscillator uses RC type oscillator circuit. The frequency is affected by the voltage and temperature of the system. In common condition, the frequency of the RC oscillator is about 32KHz.

The internal low RC supports watchdog clock source and system slow mode controlled by “CLKMD” bit of OSCM register.

- ***Flosc = Internal low RC oscillator (about 32KHz).***
- ***Slow mode Fcpu = Flosc/ 1 ~ Flosc/8 controlled by Low_Fcpu code option.***

When watchdog timer is disabled and system is in power down mode, the internal low RC stops.

➢ **Example: Stop internal low-speed oscillator by power down mode as watchdog timer disable**

```
        B0BSET      FCPUM0      ; To stop external high-speed oscillator and internal low-speed
                                ; oscillator called power down mode (sleep mode).
```

4.6 OSCM REGISTER

The OSCM register is an oscillator control register. It controls oscillator status, system mode.

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-
Read/Write	-	-	-	R/W	R/W	R/W	R/W	-
After reset	-	-	-	0	0	0	0	-

Bit 1 **STPHX:** External high-speed oscillator control bit.
0 = External high-speed oscillator free run.
1 = External high-speed oscillator free run stop. Internal low-speed RC oscillator is still running.

Bit 2 **CLKMD:** System high/Low clock mode control bit.
0 = Normal (dual) mode. System clock is high clock.
1 = Slow mode. System clock is internal low clock.

Bit[4:3] **CPUM[1:0]:** CPU operating mode control bits.
00 = normal.
01 = sleep (power down) mode.
10 = green mode.
11 = reserved.

“STPHX” bit controls internal high speed RC type oscillator and external oscillator operations. When “STPHX=0”, the external oscillator or internal high speed RC type oscillator active. When “STPHX=1”, the external oscillator or internal high speed RC type oscillator are disabled. The STPHX function is depend on different high clock options to do different controls.

4.7 SYSTEM CLOCK MEASUREMENT

Under design period, the users can measure system clock speed by software instruction cycle (Fcpu). This way is useful in RC mode.

➢ **Example: Fcpu instruction cycle of external oscillator.**

```
        B0BSET      P0M.0       ; Set P0.0 to be output mode for outputting Fcpu toggle signal.

@@:
        B0BSET      P0.0        ; Output Fcpu toggle signal in low-speed clock mode.
        B0BCLR      P0.0        ; Measure the Fcpu frequency by oscilloscope.
        JMP         @B
```

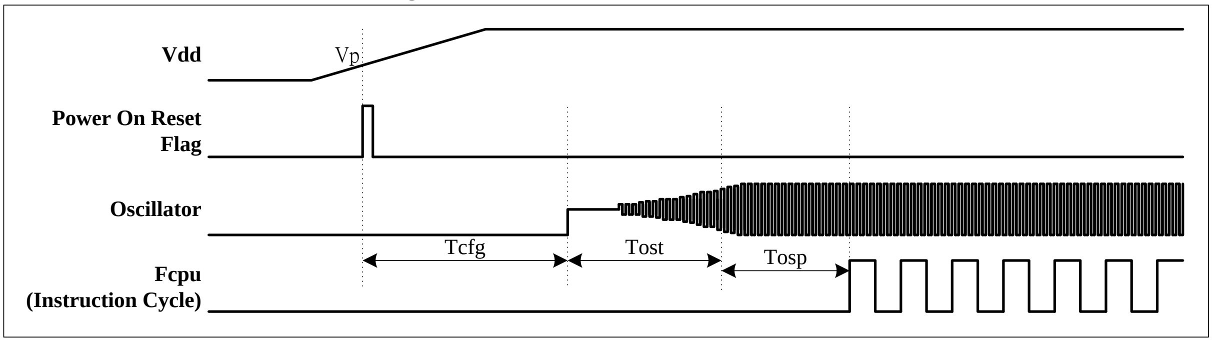
✸ ***Note: Do not measure the RC frequency directly from XIN; the probe impendence will affect the RC frequency.***

4.8 SYSTEM CLOCK TIMING

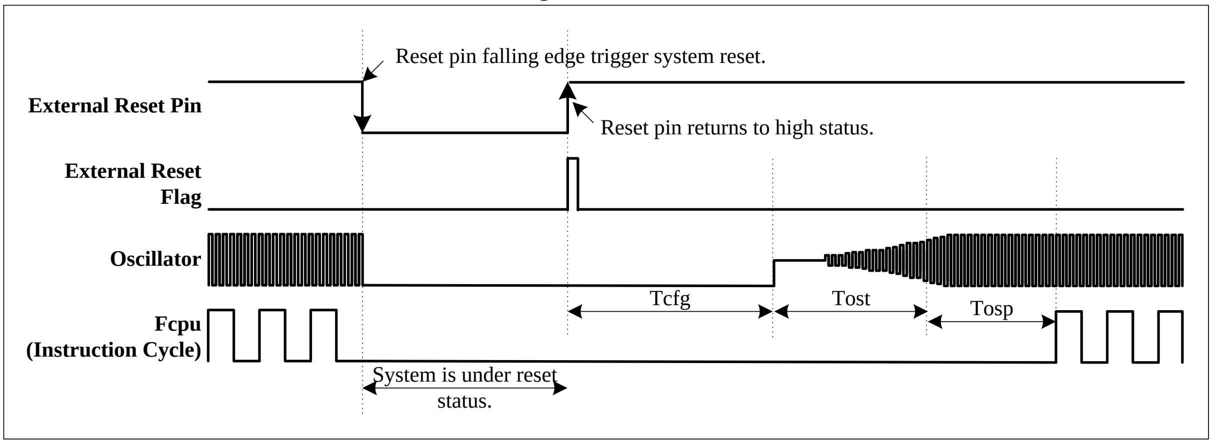
Parameter	Symbol	Description	Typical
Hardware configuration time	Tcfg	$2048*F_{ILRC}$	64ms @ F_{ILRC} = 32KHz

Oscillator start up time	Tost	The start-up time is depended on oscillator's material, factory and architecture. Normally, the low-speed oscillator's start-up time is lower than high-speed oscillator. The RC type oscillator's start-up time is faster than crystal type oscillator.	-
Oscillator warm-up time	Tosp	Oscillator warm-up time of reset condition. 2048*F_{hosc} (Power on reset, LVD reset, watchdog reset, external reset pin active.)	128us @ F_{hosc} = 16MHz
		Oscillator warm-up time of power down mode wake-up condition. 2048*F_{hosc} ……Crystal/resonator type oscillator, e.g. 12MHz crystal, 16MHz crystal…	X'tal: 192us @ F_{hosc} = 12MHz 128us @ F_{hosc} = 16MHz

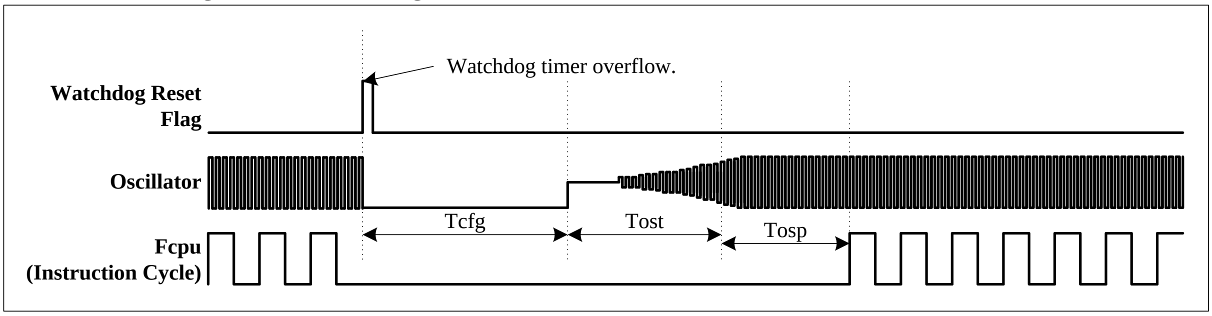
- **Power On Reset Timing**



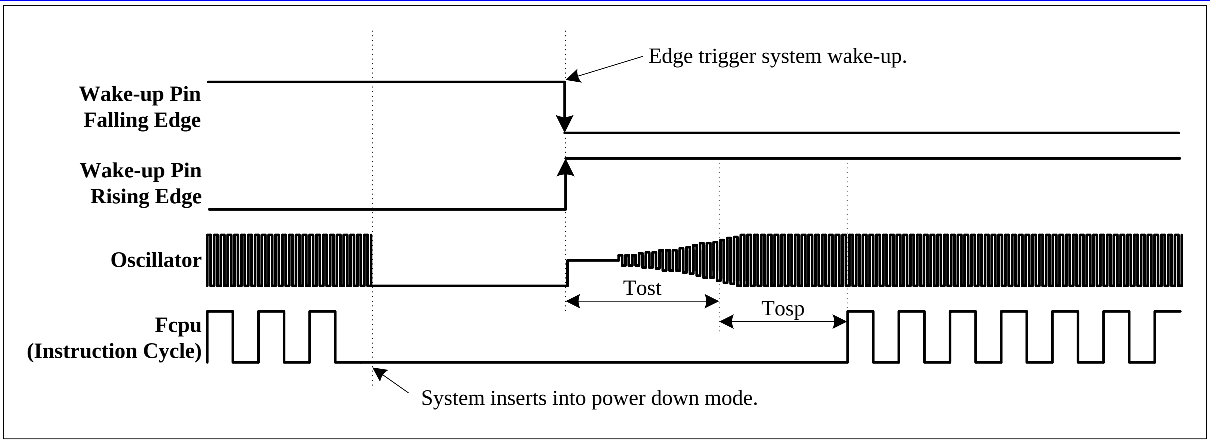
- **External Reset Pin Reset Timing**



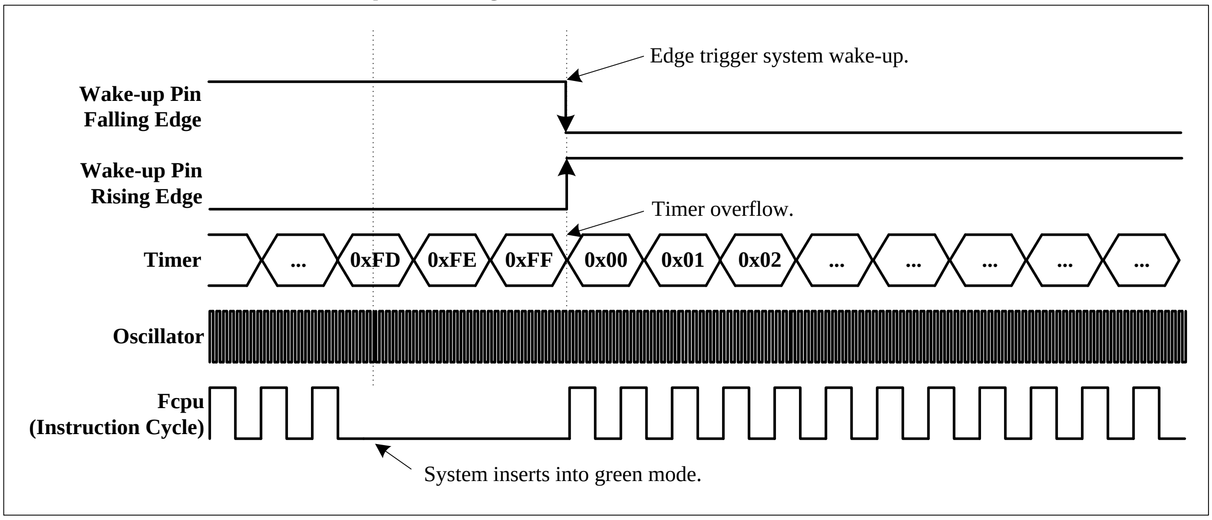
- **Watchdog Reset Timing**



- **Power Down Mode Wake-up Timing**

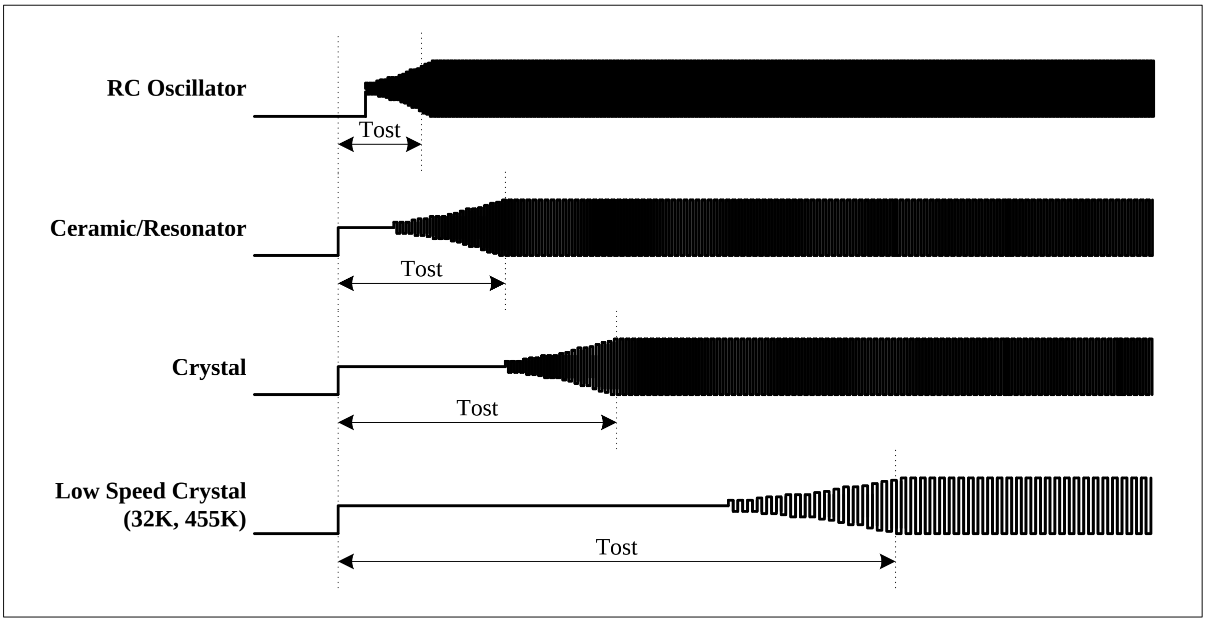


- **Green Mode Wake-up Timing**



- **Oscillator Start-up Time**

The start-up time is depended on oscillator's material, factory and architecture. Normally, the low-speed oscillator's start-up time is lower than high-speed oscillator. The RC type oscillator's start-up time is faster than crystal type oscillator.



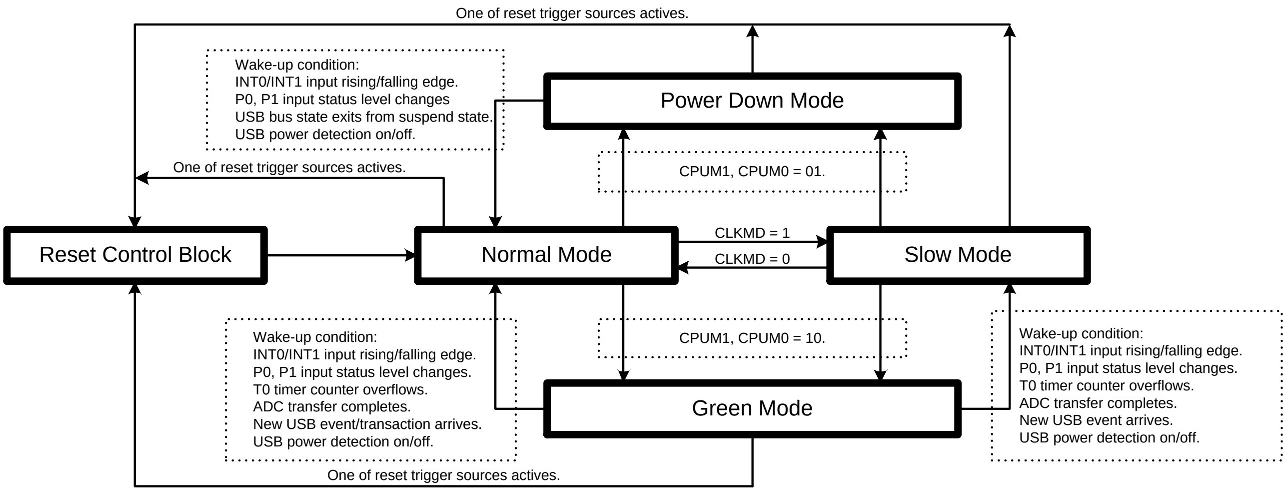
5 SYSTEM OPERATION MODE

5.1 OVERVIEW

The chip builds in four operating mode for difference clock rate and power saving reason. These modes control oscillators, op-code operation and analog peripheral devices' operation.

- Normal mode: System high-speed operating mode.
- Slow mode: System low-speed operating mode.
- Power down mode: System power saving mode (Sleep mode).
- Green mode: System ideal mode.

Operating Mode Control Block



Operating Mode Clock Control Table

Operating Mode	Normal Mode	Slow Mode	Green Mode	Power Down Mode
IHRC	**IHRC:** Running **Ext. OSC:** Disable	**IHRC:** By STPHX **Ext. OSC:** Disable	**IHRC:** By STPHX **Ext. OSC:** Disable	Stop
ILRC	Running	Running	Running	Stop
Ext. Osc.	**IHRC:** Disable **Ext. OSC:** Running	**IHRC:** Disable **Ext. OSC:** By STPHX	**IHRC:** By STPHX **Ext. OSC:** By STPHX	Stop
CPU instruction	Executing	Executing	Stop	Stop
T0 timer	Active By T0ENB	Active By T0ENB	Active By T0ENB	Inactive
TC0 timer (Timer, Event counter, PWM)	Active By TC0ENB	Active By TC0ENB	Active By TC0ENB	Inactive
TC1 timer (Timer, PWM)	Active By TC1ENB	Active By TC1ENB	Active By TC1ENB	Inactive
TC2 timer (Timer, PWM)	Active By TC2ENB	Active By TC2ENB	Active By TC2ENB	Inactive
TC3 timer (Timer, PWM)	Active By TC3ENB	Active By TC3ENB	Active By TC3ENB	Inactive
SIO	Active as enable	Inactive	Inactive	Inactive
UART	Active as enable	Inactive	Inactive	Inactive
ADC	Active as enable	Active as enable	Active as enable	Inactive
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option
Internal interrupt	All active	All active	All active	All inactive
External interrupt	All active	All active	All active	All inactive
Wakeup source	-	-	P0, P1, T0, Reset, ADC, INT0, INT1, USB event arrives (Bus state change or USB transaction completes) USB power detection	P0, P1, Reset, INT0, INT1 USB event (Bus state change) USB power detection

- **Ext.Osc:** External high-speed oscillator (XIN/XOUT).
- **IHRC:** Internal high-speed oscillator RC type.
- **ILRC:** Internal low-speed oscillator RC type.

> ✸ ***Note: SIO and UART inactive in slow mode and green mode, because the clock source doesn't exist. Use firmware to disable SIO/UART function before inserting slow mode and green mode.***

5.2 NORMAL MODE

The Normal Mode is system high clock operating mode. The system clock source is from high speed oscillator. The program is executed. After power on and any reset trigger released, the system inserts into normal mode to execute program. When the system is wake-up from power down mode, the system also inserts into normal mode. In normal mode, the high speed oscillator actives, and the power consumption is largest of all operating modes.

- The program is executed, and full functions are controllable.
- The system rate is high speed.
- The high speed oscillator and internal low speed RC type oscillator active.
- Normal mode can be switched to other operating modes through OSCM register.
- Power down mode is wake-up to normal mode.
- Slow mode is switched to normal mode.
- Green mode from normal mode is wake-up to normal mode.

5.3 SLOW MODE

The slow mode is system low clock operating mode. The system clock source is from internal low speed RC type oscillator. The slow mode is controlled by CLKMD bit of OSCM register. When CLKMD=0, the system is in normal mode. When CLKMD=1, the system inserts into slow mode. The high speed oscillator won't be disabled automatically after switching to slow mode, and must be disabled by SPTHX bit to reduce power consumption. In slow mode, the system rates are Flosc/1, Flosc/2, Flosc/4, Flosc/8 (Flosc is internal low speed RC type oscillator frequency) controlled by code option.

- The program is executed, and full functions are controllable.
- The system rate is low speed (Flosc/1, Flosc/2, Flosc/4, Flosc/8 controlled by code option).
- The internal low speed RC type oscillator actives, and the high speed oscillator is controlled by STPHX=1. In slow mode, to stop high speed oscillator is strongly recommendation.
- Slow mode can be switched to other operating modes through OSCM register.
- Power down mode from slow mode is wake-up to normal mode.
- Normal mode is switched to slow mode.
- Green mode from slow mode is wake-up to slow mode.

5.4 POWER DOWN MDOE

The power down mode is the system ideal status. No program execution and oscillator operation. Only internal regulator actives to keep all control gates status, register status and SRAM contents. The power down mode is waked up by P1 hardware level change trigger. P1 wake-up function is controlled by P1W register. Any operating modes into power down mode, the system is waked up to normal mode. Inserting power down mode is controlled by CPUM0 bit of OSCM register. When CPUM0=1, the system inserts into power down mode. After system wake-up from power down mode, the CPUM0 bit is disabled (zero status) automatically, and the WAKE bit set as “1”.

- The program stops executing, and full functions are disabled.
- All oscillators including external high speed oscillator, internal high speed oscillator and internal low speed oscillator stop.
- The system inserts into normal mode after wake-up from power down mode.
- The power down mode wake-up source is P1 level change trigger.
- After system wake-up from power down mode, the WAKE bit set as “1” and cleared by program.
- If wake-up source is external interrupt source, the WAKE bit won’t be set, and external interrupt IRQ bit is set. The system issues external interrupt request and executes interrupt service routine.

> ✷ ***Note: If the system is in normal mode, to set STPHX=1 to disable the high clock oscillator. The system is under no system clock condition. This condition makes the system stay as power down mode, and can be wake-up by P1 level change trigger.***

5.5 GREEN MODE

The green mode is another system ideal status not like power down mode. In power down mode, all functions and hardware devices are disabled. But in green mode, the system clock source keeps running, so the power consumption of green mode is larger than power down mode. In green mode, the program isn’t executed, but the timer with wake-up function actives as enabled, and the timer clock source is the non-stop system clock. The green mode has 2 wake-up sources. One is the P1 level change trigger wake-up. The other one is internal timer with wake-up function occurring overflow. That’s mean users can setup one fix period to timer, and the system is waked up until the time out. Inserting green mode is controlled by CPUM1 bit of OSCM register. When CPUM1=1, the system inserts into green mode. After system wake-up from green mode, the CPUM1 bit is disabled (zero status) automatically, and the WAKE bit set as “1”.

- The program stops executing, and full functions are disabled.
- Only the timer with wake-up function actives.
- The oscillator to be the system clock source keeps running, and the other oscillators operation is depend on system operation mode configuration.
- If inserting green mode from normal mode, the system insets to normal mode after wake-up.
- If inserting green mode from slow mode, the system insets to slow mode after wake-up.
- The green mode wake-up sources are P1 level change trigger and unique time overflow.
- After system wake-up from power down mode, the WAKE bit set as “1” and cleared by program.
- If wake-up source is external interrupt source, the WAKE bit won’t be set, and external interrupt IRQ bit is set. The system issues external interrupt request and executes interrupt service routine.
- If the function clock source is system clock, the functions are workable as enabled and under green mode, e.g. Timer, PWM, event counter…But the functions doesn’t has wake-up function.

> ✷ ***Note: Sonix provides “GreenMode” macro to control green mode operation. It is necessary to use “GreenMode” macro to control system inserting green mode.***
> ***The macro includes three instructions. Please take care the macro length as using BRANCH type instructions, e.g. bts0, bts1, b0bts0, b0bts1, ins, incms, decs, decms, cmprs, jmp, or the routine would be error.***

5.6 OPERATING MODE CONTROL MACRO

Sonix provides operating mode control macros to switch system operating mode easily.

Macro	Length	Description
SleepMode	1-word	The system insets into Sleep Mode (Power Down Mode).
GreenMode	3-word	The system inserts into Green Mode.
SlowMode	2-word	The system inserts into Slow Mode and stops high speed oscillator.
Slow2Normal	5-word	The system returns to Normal Mode from Slow Mode. The macro includes operating mode switch, enable high speed oscillator, high speed oscillator warm-up delay time.

➢ **Example: Switch normal/slow mode to power down (sleep) mode.**

```
        SleepMode                       ; Declare "SleepMode" macro directly.
```

➢ **Example: Switch normal mode to slow mode.**

```
        SlowMode                        ; Declare "SlowMode" macro directly.
```

➢ **Example: Switch slow mode to normal mode (The external high-speed oscillator stops).**

```
        Slow2Normal                     ; Declare "Slow2Normal" macro directly.
```

➢ **Example: Switch normal/slow mode to green mode.**

```
        GreenMode                       ; Declare "GreenMode" macro directly.
```

➢ **Example: Switch normal/slow mode to green mode and enable T0 wake-up function.**

```
; Set T0 timer wakeup function.
        B0BCLR      FT0IEN              ; To disable T0 interrupt service
        B0BCLR      FT0ENB              ; To disable T0 timer
        MOV         A,#20H              ;
        B0MOV       T0M,A               ; To set T0 clock = Fcpu / 64
        MOV         A,#74H
        B0MOV       T0C,A               ; To set T0C initial value = 74H (To set T0 interval = 10 ms)
        B0BCLR      FT0IEN              ; To disable T0 interrupt service
        B0BCLR      FT0IRQ              ; To clear T0 interrupt request
        B0BSET      FT0ENB              ; To enable T0 timer

; Go into green mode
        GreenMode                       ; Declare "GreenMode" macro directly.
```

5.7 WAKEUP

5.7.1 OVERVIEW

Under power down mode (sleep mode) or green mode, program doesn't execute. The wakeup trigger can wake the system up to normal mode or slow mode. The wakeup trigger sources are external trigger (P1 level change) and internal trigger (T0 timer overflow). The wakeup function builds in interrupt operation issued IRQ flag and trigger system executing interrupt service routine as system wakeup occurrence.

- Power down mode is waked up to normal mode. The wakeup trigger is only external trigger (P1 level change)
- Green mode is waked up to last mode (normal mode or slow mode). The wakeup triggers are external trigger (P1 level change) and internal trigger (T0 timer overflow).
- Wakeup interrupt function issues WAKEIRQ as system wakeup from power down mode or green mode. If WAKEIEN is "1" meaning enable, the wakeup event triggers program counter point to interrupt vector (ORG 8) executing interrupt service routine.

> ✷ ***Note: If wake-up source is external interrupt source, the WAKE bit won't be set, and external interrupt IRQ bit is set. The system issues external interrupt request and executes interrupt service routine.***

5.7.2 WAKEUP TIME

When the system is in power down mode (sleep mode), the high clock oscillator stops. When waked up from power down mode, MCU waits for 2048 external high-speed oscillator clocks and 64 internal high-speed oscillator clocks as the wakeup time to stable the oscillator circuit. After the wakeup time, the system goes into the normal mode.

> ✷ ***Note: Wakeup from green mode is no wakeup time because the clock doesn't stop in green mode.***

The value of the external high clock oscillator wakeup time is as the following.

The Wakeup time = 1/Fosc * 2048 (sec) + high clock start-up time

➢ **Example: In power down mode (sleep mode), the system is waked up. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.**

The wakeup time = 1/Fosc * 2048 = 0.128 ms (Fosc = 16MHz)
The total wakeup time = 0.128 ms + oscillator start-up time

The value of the internal high clock oscillator RC type wakeup time is as the following.

The Wakeup time = 1/Fosc * 64 (sec) + high clock start-up time

➢ **Example: In power down mode (sleep mode), the system is waked up. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.**

The wakeup time = 1/Fosc * 64 = 4 us (Fhosc = 16MHz)

> ✷ ***Note: The high clock start-up time is depended on the VDD and oscillator type of high clock.***

5.7.3 P0W WAKEUP CONTROL REGISTER

Under power down mode (sleep mode) and green mode, the I/O ports with wakeup function are able to wake the system up to normal mode. The wake-up trigger edge is level changing. When wake-up pin occurs rising edge or falling edge, the system is waked up by the trigger edge. Port 0 has wakeup function. Port 0 is controlled by the P0W register.

09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0W	P07W	P06W	P05W	P04W	P03W	P02W	P01W	P00W
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit[7:0] **P00W~P07W:** Port 1 wakeup function control bits.
0 = Disable P1n wakeup function.
1 = Enable P1n wakeup function.

5.7.4 P1W WAKEUP CONTROL REGISTER

Under power down mode (sleep mode) and green mode, the I/O ports with wakeup function are able to wake the system up to normal mode. The wake-up trigger edge is level changing. When wake-up pin occurs rising edge or falling edge, the system is waked up by the trigger edge. Port 1 has wakeup function. Port 1 is controlled by the P1W register.

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	-	P15W	P14W	P13W	P12W	P11W	P10W
Read/Write	R/W	--	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	-	0	0	0	0	0	0

Bit[7], [5:0]**P10W~P17W:** Port 1 wakeup function control bits.
0 = Disable P1n wakeup function.
1 = Enable P1n wakeup function.

5.7.5 P0IE INTERRUPT CONTROL REGISTER

09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0IE	P07IE	P06IE	P05IE	P04IE	P03IE	P02IE	P01IE	P00IE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit[7:0] **P00IE~P07IE:** Port 0 level change interrupt function control bit.
0 = Disable P0n level change function.
1 = Enable P0n level change function

5.7.6 P1IE INTERRUPT CONTROL REGISTER

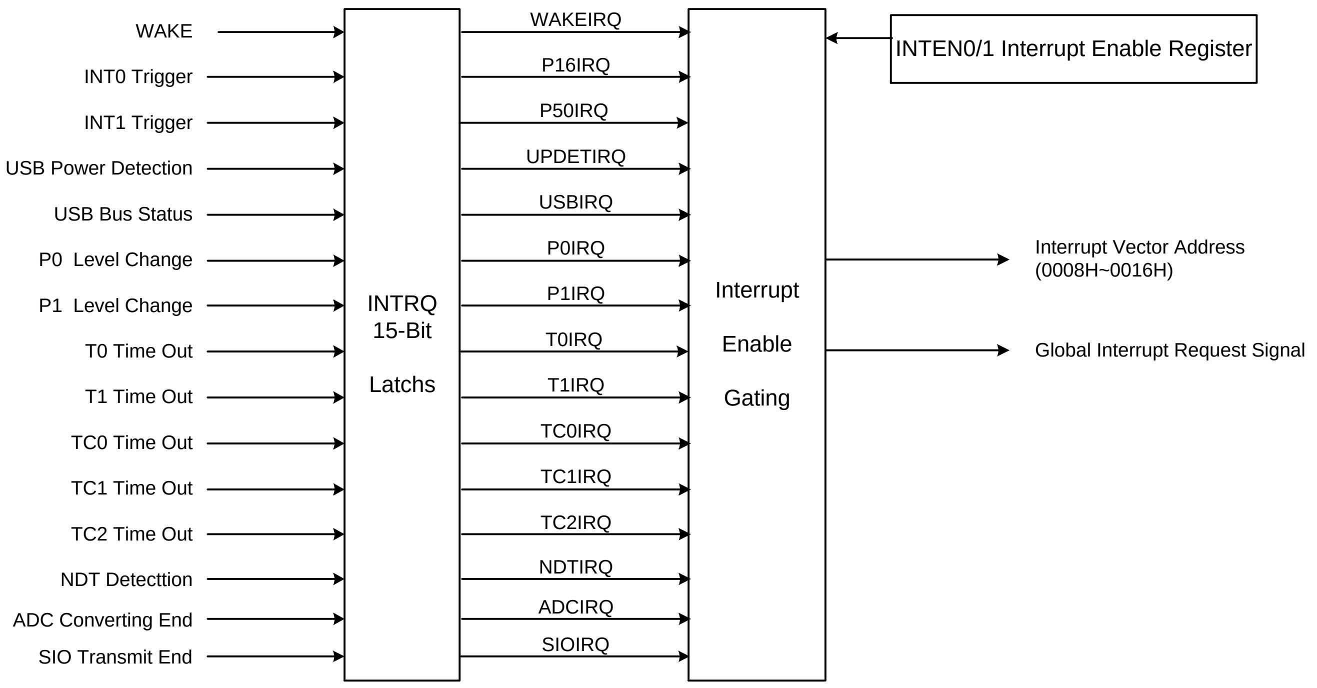
09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1IE	P17IE	-	P15IE	P14IE	P13IE	P12IE	P11IE	P10IE
Read/Write	R/W	--	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	-	0	0	0	0	0	0

Bit[7], [5:0]**P10IE~P17UE:** Port 1 level change interrupt function control bit.
0 = Disable P1n level change function.
1 = Enable P1n level change function.

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 15 interrupt sources, including 2 external interrupt (INT0/INT1) and 13 internal interrupt (WAKE/UPDET/USB/P0/P1/T0 /T1/TC0/TC1/TC2/NDT/ADC/SIO). The external interrupt can wakeup the chip while the system is switched from power down mode to high-speed normal mode, and interrupt request is latched until return to normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to "0" for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to "1" to accept the next interrupts' request. The interrupt request signals are stored in INTRQ register.



✸ ***Note: The GIE bit must enable during all interrupt operation.***

6.2 INTERRUPT OPERATION

Interrupt operation is controlled by IRQ and IEN bits. The IRQ is interrupt source event indicator, no matter what interrupt function status (enable or disable). The IEN control the system interrupt execution. If IEN = 0, the system won't jump to interrupt vector to execute interrupt routine. If IEN = 1, the system executes interrupt operation when each of interrupt IRQ flags actives.

- **IEN = 1 and IRQ = 1, the program counter points to interrupt vector and execute interrupt service routine.**
- **0x80~0x87 working registers include L, H, R, Z, Y, X, PFLAG, RBANK, W0~W7.**

6.3 INTEN INTERRUPT ENABLE REGISTER

INTEN is the interrupt request control register including eleven internal interrupts, two external interrupts enable control bits. One of the register to be set "1" is to enable the interrupt request function. Once of the interrupt occur, the stack is incremented and program jump to ORG 8~16 to execute interrupt service routines. The program exits the interrupt service routine when the returning interrupt service routine instruction (RETI) is executed.

099H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN0	T0IEN	P1IEN	P0IEN	USBIEN	UPDETIEN	P50IEN	P16IEN	WAKEIEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **WAKEIEN:** Wakeup interrupt control bit.
0 = Disable Wakeup interrupt function.
1 = Enable Wakeup interrupt function.
Bit 1 **P16IEN:** External P1.6 interrupt (INT0) control bit.
0 = Disable INT0 interrupt function.
1 = Enable INT0 interrupt function.
Bit 2 **P50IEN:** External P5.0 interrupt (INT1) control bit.
0 = Disable INT1 interrupt function.
1 = Enable INT1 interrupt function.
Bit 3 **UPDETIEN:** USB power detection interrupt control bit.
0 = Disable USB power detection interrupt function.
1 = Enable USB power detection interrupt function.
Bit 4 **USBIEN:** USB interrupt control bit.
0 = Disable USBinterrupt function.
1 = Enable USBinterrupt function.
Bit 5 **P0IEN:** P0 level change wakeup interrupt control bit.
0 = Disable P0 interrupt function.
1 = Enable P0 interrupt function.
Bit 6 **P1IEN:** P1 level change wakeup interrupt control bit.
0 = Disable P0 interrupt function.
1 = Enable P1 interrupt function.
Bit 7 **T0IEN:** T0 timer interrupt control bit.
0 = Disable T0 interrupt function.
1 = Enable T0 interrupt function.

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN1	-	SIOIEN	ADCIEN	NDTIEN	TC2IEN	TC1IEN	TC0IEN	T1IEN
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	0	0	0	0	0	0	0

Bit 0 **T1IEN:** T1 timer interrupt control bit.
0 = Disable T1 timer interrupt function.
1 = Enable T1 timer interrupt function.
Bit 1 **TC0IEN:** TC0 timer interrupt control bit.
0 = Disable TC0 timer interrupt function.
1 = Enable TC0 timer interrupt function.
Bit 2 **TC1IEN:** TC1 receive interrupt control bit.
0 = Disable TC1 timer interrupt function.
1 = Enable TC1 timer interrupt function.
Bit 3 **TC2IEN:** TC02transmit interrupt control bit.
0 = Disable TC02 timer interrupt function.
1 = Enable TC02 timer interrupt function.
Bit 4 **NDTIEN:** NDT interrupt control bit.
0 = Disable NDT interrupt function.
1 = Enable NDT interrupt function.
Bit 5 **ADCIEN:** ADC interrupt control bit.
0 = Disable ADC interrupt function.
1 = Enable ADC interrupt function.

Bit 6 **SIOIEN:** SIO interrupt control bit.
0 = Disable SIO interrupt function.
1 = Enable SIO interrupt function.

> ✸ ***Note: For some applications, the system has to loop IEN bits enable /disable while operating. User must follow the rule: The instruction of clear IEN bits (EX: B0BCLR FTC0IEN) must be followed by JMP $+1.***

➢ **Example: Loop IEN bits enable /disable**

```
Initial:
            B0BCLR      FTC0IRQ         ;Initial TC0 Interrupt
            B0BSET      FTC0IEN
            MOV         a, #00100100B   ;Initial TC0 function
            B0MOV       TC0M, a
            MOV         a, #0x55
            B0MOV       TC0C, a
            B0MOV       TC0R, a

            B0BSET      FTC0ENB         ;Enable TC0 timer
            B0BSET      FGIE            ;Global interrupt enable

_Main:
            ……
            B0BCLR      FTC0IEN         ;Disable IEN bits
            JMP         $+1             ;Disable IEN bits followed by JMP $+1
            ……
            B0BSET      FTC0IEN         ;Enable IEN bits
            ……
            JMP         _Main
```

6.4 INTRQ INTERRUPT REQUEST REGISTER

INTRQ is the interrupt request flag register. The register includes all interrupt request indication flags. Each one of the interrupt requests occurs, the bit of the INTRQ register would be set "1". The INTRQ value needs to be clear by programming after detecting the flag. In the interrupt vector of program, users know the any interrupt requests occurring by the register and do the routine corresponding of the interrupt request.

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ0	T0IRQ	P1IRQ	P0IRQ	USBIRQ	UPDETIRQ	P50IRQ	P16IRQ	WAKEIRQ
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **WAKEIRQ:** WAKEIRQ interrupt request flag.
0 = None WAKEIRQ interrupt request.
1 = WAKEIRQ interrupt request.

Bit 1 **P16IRQ:** External P1.6 interrupt (INT0) request flag.
0 = None P16IRQ interrupt request.
1 = P16IRQ interrupt request.

Bit 2 **P50IRQ:** External P5.0 interrupt (INT1) request flag.
0 = None P50IRQ interrupt request.
1 = P50IRQ interrupt request.

Bit 3 **UPDETIRQ:** UPDETIRQ interrupt request flag.
0 = None UPDETIRQ interrupt request.
1 = UPDETIRQ interrupt request.

Bit 4 **USBIRQ:** USBIRQ interrupt request flag.
0 = None USBIRQ interrupt request.
1 = USBIRQ interrupt request.

Bit 5 **P0IRQ:** P0IRQ interrupt request flag.
0 = None P0IRQ interrupt request.
1 = P0IRQ interrupt request.

Bit 6 **P1IRQ:** P1IRQ interrupt request flag.
0 = None P1IRQ interrupt request.
1 = P1IRQ interrupt request.

Bit 7 **T0IRQ:** T0IRQ timer interrupt request flag.
0 = None T0IRQ interrupt request.
1 = TC3 interrupt request.

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ1	-	SIOIRQ	ADCIRQ	NDTIRQ	TC2IRQ	TC1IRQ	TC0IRQ	T1IRQ
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	0	0	0	0	0	0	0

Bit 0 **T1IRQ:** T1IRQ timer interrupt request flag.
0 = None T1IRQ interrupt request.
1 = T1IRQ interrupt request.

Bit 1 **TC0IRQ:** TC0IRQinterrupt request flag.
0 = None TC0IRQ timer interrupt request.
1 = TC0IRQ interrupt request.

Bit 2 **TC1IRQ:** TC1IRQ timer receive interrupt request flag.
0 = None TC1IRQ interrupt request.
1 = TC1IRQ interrupt request.

Bit 3 **TC2IRQ:** TC2IRQ timer interrupt request flag.
0 = None TC2IRQ interrupt request.
1 = TC2IRQ interrupt request.

Bit 4 **NDTIRQ:** NDTIRQ interrupt request flag.
0 = None NDTIRQ interrupt request.
1 = NDTIRQ interrupt request.

Bit 5 **ADCIRQ:** ADC interrupt request flag.
0 = None ADC interrupt request.
1 = ADC interrupt request.

Bit 6 **SIOIRQ:** SIOIRQ interrupt request flag.
0 = None SIOIRQ interrupt request.
1 = SIOIRQ interrupt request.

6.5 GIE GLOBAL INTERRUPT OPERATION

GIE is the global interrupt control bit. All interrupts start work after the GIE = 1 It is necessary for interrupt service request. One of the interrupt requests occurs, and the program counter (PC) points to the interrupt vector (ORG 8~14) and the stack add 1 level.

0EFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	LVD24	LVD33	ULVD36	-	STKPB2	STKPB1	STKPB0
Read/Write	R/W	R	R	R	-	R/W	R/W	R/W
After reset	0	-	-	-	-	1	1	1

Bit 7 **GIE:** Global interrupt control bit.
0 = Disable global interrupt.
1 = Enable global interrupt.

- **Example: Set global interrupt control bit (GIE).**

```
        B0BSET      FGIE            ; Enable GIE
```

✸ ***Note: The GIE bit must enable during all interrupt operation.***

6.6 EXTERNAL INTERRUPT OPERATION (INT0~INT1)

Sonix provides 2 sets external interrupt sources in the micro-controller. INT0 and INT1 are external interrupt trigger sources and build in edge trigger configuration function. When the external edge trigger occurs, the external interrupt request flag will be set to "1" when the external interrupt control bit enabled. If the external interrupt control bit is disabled, the external interrupt request flag won't active when external edge trigger occurrence. When external interrupt control bit is enabled and external interrupt edge trigger is occurring, the program counter will jump to the interrupt vector (ORG 0x0009, 0x000A) and execute interrupt service routine.
The external interrupt builds in wake-up latch function. That means when the system is triggered wake-up from power down mode, the wake-up source is external interrupt source (P1.6 or P5.0), and the trigger edge direction matches interrupt edge configuration, the trigger edge will be latched, and the system executes interrupt service routine fist after wake-up.

09FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	UPDETS	UPDET_EN	P20VSS_EN	-	P50G1	P50G0	P16G1	P16G0
Read/Write	R	R/W	R/W	-	R/W	R/W	R/W	R/W
After reset	0	0	0	-	1	0	1	0

Bit[7] **UPDETS:** USB VDD existence status. This flag is only valid when UPDET_EN control bit is '1'.
0 = USB VDD does not exist.
1 = USB VDD exists.

Bit[6] **UPDET_EN:** USB VDD detect function control bit.
0 = Disable USB VDD detect function
1 = Enable USB VDD detect function.

✷ ***Note: After enabling UPDET_EN bit, firmware should wait for a delay time to let power detection be stable.***

Bit[5] **P20VSS_EN:** P2.0's 20K discharge path to GND control bit.
0 = Disable P2.0 discharge path to GND.
1 = Enable P2.0 discharge path to GND.

Bit[3:2] **P50G [1:0]:** INT1 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

Bit[1:0] **P16G[1:0]:** INT0 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

Example: Setup INT0 interrupt request and bi-direction edge trigger.

```
        MOV     A, #03H
        B0MOV   PEDGE, A        ; Set INT0 interrupt trigger as bi-direction edge.

        B0BSET  FP16IEN         ; Enable INT0 interrupt service
        B0BCLR  FP16IRQ         ; Clear INT0 interrupt request flag
        B0BSET  FGIE            ; Enable GIE
```

Example: INT0 interrupt service routine.

```
        ORG     9               ; Interrupt vector
        JMP     INT_SERVICE
INT_SERVICE:
        …
```

```
            B0BTS1      FP16IRQ         ; Check P16IRQ
            JMP         EXIT_INT        ; P16IRQ = 0, exit interrupt vector

            B0BCLR      FP16IRQ         ; Reset P16IRQ
            …                           ; INT0 interrupt service routine
EXIT_INT:
            …
            RETI                        ; Exit interrupt vector
```

6.7 T0 INTERRUPT OPERATION

When the T0C counter occurs overflow, the T0IRQ will be set to "1" however the T0IEN is enable or disable. If the T0IEN = 1, the trigger event will make the T0IRQ to be "1" and the system enter interrupt vector. If the T0IEN = 0, the trigger event will make the T0IRQ to be "1" but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

➢ **Example: T0 interrupt request setup.**

```
		B0BCLR		FT0IEN			; Disable T0 interrupt service
		B0BCLR		FT0ENB			; Disable T0 timer
		MOV		A, #20H			;
		B0MOV		T0M, A			; Set T0 clock = Fcpu / 64
		MOV		A, #74H			; Set T0C initial value = 74H
		B0MOV		T0C, A			; Set T0 interval = 10 ms

		B0BSET		FT0IEN			; Enable T0 interrupt service
		B0BCLR		FT0IRQ			; Clear T0 interrupt request flag
		B0BSET		FT0ENB			; Enable T0 timer

		B0BSET		FGIE			; Enable GIE
```

Example: T0 interrupt service routine.

```
		ORG		0FH			; Interrupt vector
		JMP		INT_SERVICE
INT_SERVICE:

		…

		B0BTS1		FT0IRQ			; Check T0IRQ
		JMP		EXIT_INT		; T0IRQ = 0, exit interrupt vector

		B0BCLR		FT0IRQ			; Reset T0IRQ
		MOV		A, #74H
		B0MOV		T0C, A			; Reset T0C.
		…					; T0 interrupt service routine
		…
EXIT_INT:
		…


		RETI					; Exit interrupt vector
```

6.8 T1 INTERRUPT OPERATION

When the T1C counter occurs overflow, the T1IRQ will be set to “1” however the T1IEN is enable or disable. If the T1IEN = 1, the trigger event will make the T1IRQ to be “1” and the system enter interrupt vector. If the T1IEN = 0, the trigger event will make the T1IRQ to be “1” but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

➢ **Example: T1 interrupt request setup.**

```
            B0BCLR      FT1IEN          ; Disable T1 interrupt service
            B0BCLR      FT1ENB          ; Disable T1 timer
            MOV         A, #20H         ;
            B0MOV       T1M, A          ; Set T1 clock = Fcpu / 64
            MOV         A, #74H         ; Set T1C initial value = 74H
            B0MOV       T1C_L, A        ; Set T1 interval = 10 ms
            CLR         T1C_H
            B0BSET      FT1IEN          ; Enable T1 interrupt service
            B0BCLR      FT1IRQ          ; Clear T1 interrupt request flag
            B0BSET      FT1ENB          ; Enable T1 timer

            B0BSET      FGIE            ; Enable GIE
```

Example: T1 interrupt service routine.

```
            ORG         10H             ; Interrupt vector
            JMP         INT_SERVICE
INT_SERVICE:

            …

            B0BTS1      FT1IRQ          ; Check T1IRQ
            JMP         EXIT_INT        ; T1IRQ = 0, exit interrupt vector

            B0BCLR      FT1IRQ          ; Reset T1IRQ
                                        ; T1 auto re-count (T1ARC=1)
            …                           ; T1 interrupt service routine
            …
EXIT_INT:
            …


            RETI                        ; Exit interrupt vector
```

6.9 TC0 INTERRUPT OPERATION

When the TC0C counter overflows, the TC0IRQ will be set to "1" no matter the TC0IEN is enable or disable. If the TC0IEN and the trigger event TC0IRQ is set to be "1". As the result, the system will execute the interrupt vector. If the TC0IEN = 0, the trigger event TC0IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the TC0IEN is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: TC0 interrupt request setup.**

```
		B0BCLR		FTC0IEN			; Disable TC0 interrupt service
		B0BCLR		FTC0ENB			; Disable TC0 timer
		MOV		A, #10H			;
		B0MOV		TC0M, A			; Set TC0 clock = Fcpu / 64
		MOV		A, #74H			; Set TC0C initial value = 74H
		B0MOV		TC0C, A			; Set TC0 interval = 10 ms

		B0BSET		FTC0IEN			; Enable TC0 interrupt service
		B0BCLR		FTC0IRQ			; Clear TC0 interrupt request flag
		B0BSET		FTC0ENB			; Enable TC0 timer

		B0BSET		FGIE			; Enable GIE
```

➢ **Example: TC0 interrupt service routine.**

```
		ORG		11H			; Interrupt vector
		JMP		INT_SERVICE
INT_SERVICE:

		…

		B0BTS1		FTC0IRQ			; Check TC0IRQ
		JMP		EXIT_INT		; TC0IRQ = 0, exit interrupt vector

		B0BCLR		FTC0IRQ			; Reset TC0IRQ
		MOV		A, #74H
		B0MOV		TC0C, A			; Reset TC0C.
		…					; TC0 interrupt service routine
		…
EXIT_INT:
		…

		RETI					; Exit interrupt vector
```

6.10 TC1 INTERRUPT OPERATION

When the TC1C counter overflows, the TC1IRQ will be set to "1" no matter the TC1IEN is enable or disable. If the TC1IEN and the trigger event TC1IRQ is set to be "1". As the result, the system will execute the interrupt vector. If the TC1IEN = 0, the trigger event TC1IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the TC1IEN is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

Example: TC1 interrupt request setup.

```
        B0BCLR      FTC1IEN         ; Disable TC1 interrupt service
        B0BCLR      FTC1ENB         ; Disable TC1 timer
        MOV         A, #10H         ;
        B0MOV       TC1M, A         ; Set TC1 clock = Fcpu / 64
        MOV         A, #74H         ; Set TC1C initial value = 74H
        B0MOV       TC1C, A         ; Set TC1 interval = 10 ms

        B0BSET      FTC1IEN         ; Enable TC1 interrupt service
        B0BCLR      FTC1IRQ         ; Clear TC1 interrupt request flag
        B0BSET      FTC1ENB         ; Enable TC1 timer

        B0BSET      FGIE            ; Enable GIE
```

Example: TC1 interrupt service routine.

```
        ORG         12H             ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

        …

        B0BTS1      FTC1IRQ         ; Check TC1IRQ
        JMP         EXIT_INT        ; TC1IRQ = 0, exit interrupt vector

        B0BCLR      FTC1IRQ         ; Reset TC1IRQ
        MOV         A, #74H
        B0MOV       TC1C, A         ; Reset TC1C.
        …                           ; TC1 interrupt service routine
        …
EXIT_INT:
        …

        RETI                        ; Exit interrupt vector
```

6.11 TC2 INTERRUPT OPERATION

When the TC2C counter overflows, the TC2IRQ will be set to “1” no matter the TC2IEN is enable or disable. If the TC2IEN and the trigger event TC2IRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the TC2IEN = 0, the trigger event TC2IRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the TC2IEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

Example: TC2 interrupt request setup.

```
        B0BCLR      FTC2IEN        ; Disable TC2 interrupt service
        B0BCLR      FTC2ENB        ; Disable TC2 timer
        MOV         A, #10H        ;
        B0MOV       TC2M, A        ; Set TC2 clock = Fcpu / 64
        MOV         A, #74H        ; Set TC2C initial value = 74H
        B0MOV       TC2C, A        ; Set TC2 interval = 10 ms

        B0BSET      FTC2IEN        ; Enable TC2 interrupt service
        B0BCLR      FTC2IRQ        ; Clear TC2 interrupt request flag
        B0BSET      FTC2ENB        ; Enable TC2 timer

        B0BSET      FGIE           ; Enable GIE
```

Example: TC2 interrupt service routine.

```
        ORG         13H            ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

        …

        B0BTS1      FTC2IRQ        ; Check TC2IRQ
        JMP         EXIT_INT       ; TC2IRQ = 0, exit interrupt vector

        B0BCLR      FTC2IRQ        ; Reset TC2IRQ
        MOV         A, #74H
        B0MOV       TC2C, A        ; Reset TC2C.
        …                          ; TC2 interrupt service routine
        …
EXIT_INT:
        …

        RETI                       ; Exit interrupt vector
```

6.12 NDT INTERRUPT OPERATION

When the NDT detected, the NDTIRQ will be set to "1" no matter the NDTIEN is enable or disable. If the NDTIEN and the trigger event NDTIRQ is set to be "1". As the result, the system will execute the interrupt vector. If the NDTIEN = 0, the trigger event NDTIRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the NDTIEN is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: NDT interrupt request setup.**

```
        B0BCLR      FNDTIEN         ; Disable NDT interrupt service

        B0BSET      FNDTIEN         ; Enable NDT interrupt service
        B0BCLR      FNDTIRQ         ; Clear NDT interrupt request flag
        B0BSET      FGIE            ; Enable GIE

        B0BSET      FESD_EN         ; Start NDT detection
```

➢ **Example: NDT interrupt service routine.**

```
        ORG         14H             ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

        …

        B0BTS1      FNDTIRQ         ; Check NDTIRQ
        JMP         EXIT_INT        ; NDTIRQ = 0, exit interrupt vector

        B0BCLR      FNDTIRQ         ; Reset NDTIRQ
        …                           ; NDT interrupt service routine
        …
EXIT_INT:
        …

        RETI                        ; Exit interrupt vector
```

6.13 ADC INTERRUPT OPERATION

When the ADC converting successfully, the ADCIRQ will be set to "1" no matter the ADCIEN is enable or disable. If the ADCIEN and the trigger event ADCIRQ is set to be "1". As the result, the system will execute the interrupt vector. If the ADCIEN = 0, the trigger event ADCIRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the ADCIEN is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: ADC interrupt request setup.**

```
            B0BCLR      FADCIEN         ; Disable ADC interrupt service

            MOV         A, #10110000B   ;
            B0MOV       ADM, A          ; Enable P4.0 ADC input and ADC function.
            MOV         A, #00000000B   ; Set ADC converting rate = Fhosc/16
            B0MOV       ADR, A

            B0BSET      FADCIEN         ; Enable ADC interrupt service
            B0BCLR      FADCIRQ         ; Clear ADC interrupt request flag
            B0BSET      FGIE            ; Enable GIE

            B0BSET      FADS            ; Start ADC transformation
```

➢ **Example: ADC interrupt service routine.**

```
            ORG         15H             ; Interrupt vector
            JMP         INT_SERVICE
INT_SERVICE:

            …

            B0BTS1      FADCIRQ         ; Check ADCIRQ
            JMP         EXIT_INT        ; ADCIRQ = 0, exit interrupt vector

            B0BCLR      FADCIRQ         ; Reset ADCIRQ
            …                           ; ADC interrupt service routine
            …
EXIT_INT:
            …

            RETI                        ; Exit interrupt vector
```

6.14 SIO INTERRUPT OPERATION

When the SIO converting successfully, the SIOIRQ will be set to “1” no matter the SIOEN is enable or disable. If the SIOIEN and the trigger event SIOIRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the SIOIEN = 0, the trigger event SIOIRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the SIOIEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: SIO interrupt request setup.**

```
                B0BSET          FSIOIEN         ; Enable SIO interrupt service
                B0BCLR          FSIOIRQ         ; Clear SIO interrupt request flag
                B0BSET          FGIE            ; Enable GIE
```

➢ **Example: SIO interrupt service routine.**

```
                ORG             16H             ; Interrupt vector
                JMP             INT_SERVICE
INT_SERVICE:
                …

                B0BTS1          FSIOIRQ         ; Check SIOIRQ
                JMP             EXIT_INT        ; SIOIRQ = 0, exit interrupt vector

                B0BCLR          FSIOIRQ         ; Reset SIOIRQ
                …                               ; SIO interrupt service routine
                …
EXIT_INT:
                …

                RETI                            ; Exit interrupt vector
```

6.15 MULTI-INTERRUPT OPERATION

Under certain condition, the software designer uses more than one interrupt requests. Processing multi-interrupt request requires setting the priority of the interrupt requests. The IRQ flags of interrupts are controlled by the interrupt event. Nevertheless, the IRQ flag “1” doesn’t mean the system will execute the interrupt vector. In addition, which means the IRQ flags can be set “1” by the events without enable the interrupt. Once the event occurs, the IRQ will be logic “1”. The IRQ and its trigger event relationship is as the below table.

Interrupt Name	***Trigger Event Description***
WAKEIRQ	Wake-up from power down or green mode.
INT0IRQ	P1.6 trigger controlled by PEDGE.
INT1IRQ	P5.0 trigger controlled by PEDGE.
UPDETIRQ	USB power detection.
USBIRQ	USB interrupt.
P0	P0 level change.
P1	P1 level change.
T0IRQ	T0C overflow.
T1IRQ	T1C overflow.
TC0IRQ	TC0C overflow.
TC1IRQ	TC1C overflow.
TC2IRQ	TC2C overflow.
NDTIRQ	NDT detection.
ADCIRQ	ADC converting end.
SIOIRQ	SIO transmitter successfully.

For multi-interrupt conditions, two things need to be taking care of. One is that it is multi-vector and each of interrupts points to unique vector. Two is users have to define the interrupt vector. The following example shows the way to define the interrupt vector in the program memory.

➢ **Example: Check the interrupt request under multi-interrupt operation**

```
            ORG         8               ; Interrupt vector
            JMP         ISR_WAKE
            JMP         ISR_INT0
            JMP         ISR_INT1
            JMP         ISR_UPDET
            JMP         ISR_USB
            JMP         ISR_P0
            JMP         ISR_P1
            JMP         ISR_T0
            JMP         ISR_T1
            JMP         ISR_TC0
            JMP         ISR_TC1
            JMP         ISR_TC2
            JMP         ISR_NDT
            JMP         ISR_ADC
            JMP         ISR_SIO

ISR_WAKE:                               ; WAKE-UP interrupt service routine

            RETI                        ; Exit interrupt vector
ISR_INT0:                               ; INT0 interrupt service routine
                                        ;
            RETI                        ; Exit interrupt vector
ISR_INT1:                               ; INT1 interrupt service routine

            RETI                        ; Exit interrupt vector
            …
            …
```

```
ISR_SIO:                                    ; SIO interrupt service routine

            RETI                            ; Exit interrupt vector
```

7 I/O PORT

7.1 OVERVIEW

The micro-controller builds in 18 pin I/O. Most of the I/O pins are mixed with analog pins and special function pins. The I/O shared pin list is as following.

I/O Pin		Shared Pin		Shared Pin Control Condition
Name	Type	Name	Type	
P0.0	I/O			
P0.1	I/O			
P0.2	I/O			
P0.3	I/O	PWM20	DC	TC2ENB=1, PWM2OUT=1, PWCH20=1
P0.4	I/O	PWM21	DC	TC2ENB=1, PWM2OUT=1, PWCH21=1
P0.5	I/O			
P0.6	I/O			
P0.7	I/O			
P1.0	I/O	PWM01	DC	TC0ENB=1, PWM0OUT=1, PWCH00=1
P1.1	I/O			
P1.2	I/O			
P1.3	I/O	SDI	DC	SENB=1, SIOMX=1
P1.4	I/O	SDO	DC	SENB=1, SIOMX=1
P1.5	I/O	SCK	DC	SENB=1
P1.6	I/O	INT0	DC	P16IEN=1
P1.7	I/O			
P2.0	I/O	AIN[0]	AC	ADENB=1,GCHS=1,CHS[4:0]=0000b
P2.1	I/O	AVREFH	AC	EVHENB =1
		AIN[1]	AC	ADENB=1,GCHS=1,CHS[4:0]=0001b
P2.2	I/O	AIN[2]	AC	ADENB=1,GCHS=1,CHS[4:0]=0010b
P2.3	I/O	AIN[3]	AC	ADENB=1,GCHS=1,CHS[4:0]=0011b
P2.4	I/O	AIN[4]	AC	ADENB=1,GCHS=1,CHS[4:0]=0100b
P2.5	I/O	AIN[5]	AC	ADENB=1,GCHS=1,CHS[4:0]=0101b
P2.6	I/O	AIN[6]	AC	ADENB=1,GCHS=1,CHS[4:0]=0110b
P2.7	I/O	AIN[7]	AC	ADENB=1,GCHS=1,CHS[4:0]=0111b
P4.0	I/O	AIN[8]	AC	ADENB=1,GCHS=1,CHS[4:0]=1000b
P4.1	I/O	AIN[9]	AC	ADENB=1,GCHS=1,CHS[4:0]=1001b
		PWM00	DC	TC0ENB=1, PWM0OUT=1, PWCH01=1
P4.2	I/O	AIN[10]	AC	ADENB=1,GCHS=1,CHS[4:0]=1010b
P4.3	I/O	AIN[11]	AC	ADENB=1,GCHS=1,CHS[4:0]=1011b
P4.4	I/O	AIN[12]	AC	ADENB=1,GCHS=1,CHS[4:0]=1100b
P4.5	I/O	AIN[13]	AC	ADENB=1,GCHS=1,CHS[4:0]=1101b
		PWM11	DC	TC1ENB=1, PWM1OUT=1, PWCH11=1
P4.6	I/O	PWM10	DC	TC1ENB=1, PWM1OUT=1, PWCH10=1
		AIN[14	AC	ADENB=1,GCHS=1,CHS[4:0]=1110b
P4.7	I/O	AIN[15]	AC	ADENB=1,GCHS=1,CHS[4:0]=1111b
		T1IN	DC	T1G1=1
P5.0	I/O	INT1	DC	P50IEN=1
P5.1	I/O			
P5.2	I/O	XIN	AC	High_CLK code option = 12M, 16M
P5.3	I/O	XOUT	AC	High_CLK code option = 12M, 16M

P5.4	I/O	EICK	DC	Embedded ICE mode.
P5.5	I/O	EIDA	DC	Embedded ICE mode.
P5.6	I/O	RST	DC	Reset_Pin code option = Reset

*** DC: Digital Characteristic. AC: Analog Characteristic. HV: High Voltage Characteristic.**

7.2 I/O PORT MODE

The port direction is programmed by PnM register. When the bit of PnM register is "0", the pin is input mode. When the bit of PnM register is "1", the pin is output mode.

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2M	P27M	P26M	P25M	P24M	P23M	P22M	P21M	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
After reset	0	0	0	0	0	0	0	-

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	P56M	P55M	P54M	P53M	P52M	P51M	P50M
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	0	0	0	0	0	0	0

Bit[7:0] **PnM[7:0]:** Pn mode control bits. (n = 0, 1, 2, 4 ,5).
0 = Pn is input mode.
1 = Pn is output mode.

✱ ***Note: Users can program them by bit control instructions (B0BSET, B0BCLR).***

➢ **Example: I/O mode selecting**

```
CLR     P0M         ; Set all ports to be input mode.
CLR     P1M
CLR     P2M
CLR     P4M
CLR     P5M

MOV     A, #0FFH    ; Set all ports to be output mode.
B0MOV   P0M, A
B0MOV   P1M, A
B0MOV   P2M, A
```

```
B0MOV      P4M, A
B0MOV      P5M, A

B0BCLR     P1M.0          ; Set P1.0 to be input mode.
B0BSET     P1M.0          ; Set P1.0 to be output mode.
```

7.3 I/O PULL UP REGISTER

The I/O pins build in internal pull-up resistors and only support I/O input mode. The port internal pull-up resistor is programmed by PnUR register. When the bit of PnUR register is “0”, the I/O pin’s pull-up is disabled. When the bit of PnUR register is “1”, the I/O pin’s pull-up is enabled.

0AAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0ABH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0ACH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2UR	P27R	P26R	P25R	P24R	P23R	P22R	P21R	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
After reset	0	0	0	0	0	0	0	-

0ADH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	-	P56R	P55R	P54R	P53R	P52R	P51R	P50R
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

- **Example: I/O Pull-up Register**

```
MOV        A, #0FFH       ; Enable Port0, 1, 2, 4, 5 Pull-up register,
B0MOV      P0UR, A        ;
B0MOV      P1UR, A
B0MOV      P2UR, A        ;
B0MOV      P4UR, A
B0MOV      P5UR, A
```

7.4 I/O PORT DATA REGISTER

0A5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	P07	P06	P05	P04	P03	P02	P01	P00
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	P17	P16	P15	P14	P13	P12	P11	P10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2	P27	P26	P25	P24	P23	P22	P21	P20
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
After reset	0	0	0	0	0	0	0	0

0A8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	P47	P46	P45	P44	P43	P42	P41	P40
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	P56	P55	P54	P53	P52	P51	P50
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	0	0	0	0	0	0	0

✸ ***Note: The P12 keeps "1" when external reset enable by code option.***

➢ **Example: Read data from input port.**

```
		B0MOV		A, P0		; Read data from Port 0
		B0MOV		A, P1		; Read data from Port 1
		B0MOV		A, P2		; Read data from Port 2
		B0MOV		A, P4		; Read data from Port 4
		B0MOV		A, P5		; Read data from Port 5
```

➢ **Example: Write data to output port.**

```
		MOV		A, #0FFH	; Write data FFH to all Port.
		B0MOV		P0, A
		B0MOV		P1, A
		B0MOV		P2, A
		B0MOV		P4, A
		B0MOV		P5, A
```

➢ **Example: Write one bit data to output port.**

```
		B0BSET		P1.0		; Set P1.0 and P1.3 to be "1".
		B0BSET		P1.3

		B0BCLR		P1.0		; Set P1.0 and P1.3 to be "0".
		B0BCLR		P1.3
```

8 TIMERS

8.1 WATCHDOG TIMER

The watchdog timer (WDT) is a binary up counter designed for monitoring program execution. If the program goes into the unknown status by noise interference, watchdog timer overflow signal raises and resets MCU. Watchdog timer clock source is internal low-speed oscillator 32KHz RC type and through programmable pre-scaler controlled by WDT_CLK code option.

Watchdog timer interval time = 256 * 1/ (Internal Low-Speed oscillator frequency/WDT Pre-scalar) …sec
= 256 / (32KHz/WDT Pre-scaler) …sec

Internal low-speed oscillator	WDT pre-scaler	Watchdog interval time
Flosc=32KHz	Flosc/4	256/(32000/4)=32ms
	Flosc/8	256/(32000/8)=64ms
	Flosc/16	256/(32000/16)=128ms
	Flosc/32	256/(32000/32)=256ms

The watchdog timer has three operating options controlled “WatchDog” code option.

- **Disable:** Disable watchdog timer function.
- **Enable:** Enable watchdog timer function. Watchdog timer actives in normal mode and slow mode. In power down mode and green mode, the watchdog timer stops.
- **Always_On:** Enable watchdog timer function. The watchdog timer actives and not stop in power down mode and green mode.

✸ ***Note:*** **In high noisy environment, the “Always_On” option of watchdog operations is the strongly recommendation to make the system reset under error situations and re-start again.**

Watchdog clear is controlled by WDTR register. Moving **0x5A** data into WDTR is to reset watchdog timer.

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Example: An operation of watchdog timer is as following. To clear the watchdog timer counter in the top of the main routine of the program.

```
Main:
        MOV         A, #5AH         ; Clear the watchdog timer.
        B0MOV       WDTR, A
        …
        CALL        SUB1
        CALL        SUB2
        …
        JMP         MAIN
```

➢ **Example: Clear watchdog timer by "@RST_WDT" macro of Sonix IDE.**

```
Main:
                @RST_WDT                        ; Clear the watchdog timer.
                …
                CALL            SUB1
                CALL            SUB2
                …
                JMP             MAIN
```

Watchdog timer application note is as following.

- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the best structure to enhance the watchdog timer function.

➢ **Example: An operation of watchdog timer is as following. To clear the watchdog timer counter in the top of the main routine of the program.**

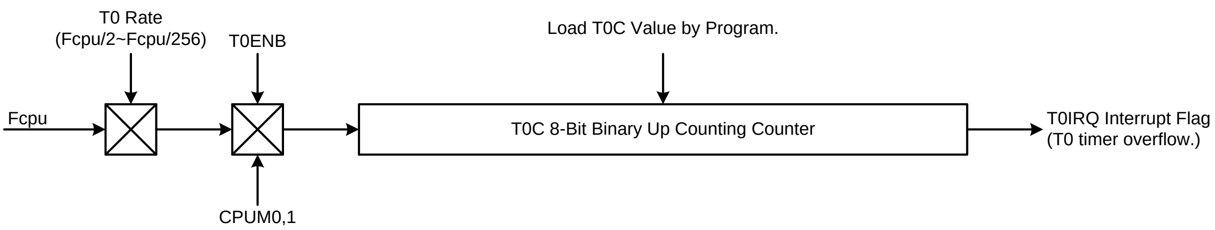
```
Main:
                …                               ; Check I/O.
                …                               ; Check RAM
Err:            JMP $                           ; I/O or RAM error. Program jump here and don't
                                                ; clear watchdog. Wait watchdog timer overflow to reset IC.

Correct:                                        ; I/O and RAM are correct. Clear watchdog timer and
                                                ; execute program.
                MOV             A, #5AH         ; Clear the watchdog timer.
                B0MOV           WDTR, A
                …
                CALL            SUB1
                CALL            SUB2
                …
                …
                …
                JMP             MAIN
```

8.2 T0 8-BIT BASIC TIMER

8.2.1 OVERVIEW

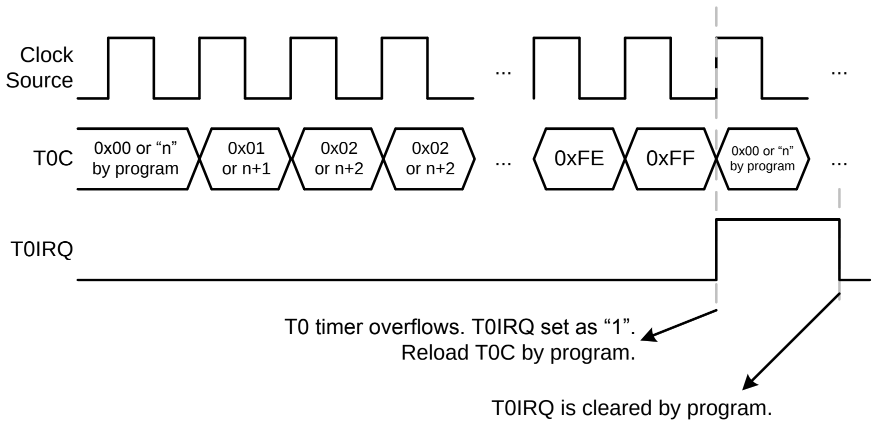
The T0 timer is an 8-bit binary up timer with basic timer function. The basic timer function supports flag indicator (T0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T0M, T0C registers. The T0 builds in green mode wake-up function. When T0 timer overflow occurs under green mode, the system will be waked-up to last operating mode.

- **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** T0 timer function supports interrupt function. When T0 timer occurs overflow, the T0IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Green mode function:** T0 timer keeps running in green mode and wakes up system when T0 timer overflows.



8.2.2 T0 Timer Operation

T0 timer is controlled by T0ENB bit. When T0ENB=0, T0 timer stops. When T0ENB=1, T0 timer starts to count. T0C increases "1" by timer clock source. When T0 overflow event occurs, T0IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is T0C count from full scale (0xFF) to zero scale (0x00). T0 doesn't build in double buffer, so load T0C by program when T0 timer overflows to fix the correct interval time. If T0 timer interrupt function is enabled (T0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 000FH) and executes interrupt service routine after T0 overflow occurrence. Clear T0IRQ by program is necessary in interrupt procedure. T0 timer can works in normal mode, slow mode and green mode. In green mode, T0 keeps counting, set T0IRQ and wakes up system when T0 timer overflows.



T0 clock source is Fcpu (instruction cycle) through T0rate[2:0] pre-scalar to decide Fcpu/2~Fcpu/256. T0 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

T0rate[2:0]	T0 Clock	T0 Interval Time			
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=16MHz, Fcpu=Fhosc/16	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)
000b	Fcpu/256	16.384	64	65.536	256
001b	Fcpu/128	8.192	32	32.768	128
010b	Fcpu/64	4.096	16	16.384	64
011b	Fcpu/32	2.048	8	8.192	32
100b	Fcpu/16	1.024	4	4.096	16
101b	Fcpu/8	0.512	2	2.048	8
110b	Fcpu/4	0.256	1	1.024	4
111b	Fcpu/2	0.128	0.5	0.512	2

8.2.3 T0M MODE REGISTER

T0M is T0 timer mode control register to configure T0 operating mode including T0 pre-scaler, clock source…These configurations must be setup completely before enabling T0 timer.

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	-
Read/Write	R/W	R/W	R/W	R/W	-	-	-	-
After reset	0	0	0	0	-	-	-	-

Bit [6:4] **T0RATE[2:0]:** T0 timer clock source select bits.
000 = Fcpu/256, 001 = Fcpu/128, 010 = Fcpu/64, 011 = Fcpu/32, 100 = Fcpu/16, 101 = Fcpu/8, 110 = Fcpu/4,111 = Fcpu/2.

Bit 7 **T0ENB:** T0 counter control bit.
0 = Disable T0 timer.
1 = Enable T0 timer.

8.2.4 T0C COUNTING REGISTER

T0C is T0 8-bit counter. When T0C overflow occurs, the T0IRQ flag is set as “1” and cleared by program. The T0C decides T0 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T0C register, and then enable T0 timer to make sure the first cycle correct. After one T0 overflow occurs, the T0C register is loaded a correct value by program.

0B4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0C	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of T0C initial value is as following.

T0C initial value = 256 - (T0 interrupt interval time * T0 clock rate)

- **Example: T0 calculation T0C to obtain 10ms T0 interval time. T0 clock source is Fcpu = 16MHz/8= 2MHz. Select T0RATE=001 (Fcpu/128).**
  T0 interval time = 10ms. T0 clock rate = 16MHz/4/128

$$\begin{aligned} \text{T0C initial value} &= 256 - (\text{T0 interval time} * \text{input clock}) \\ &= 256 - (10ms * 16MHz / 8 / 128) \\ &= 256 - (10^{-2} * 16 * 10^{6} / 8 / 128) \\ &= 64 \end{aligned}$$

8.2.5 T0 TIMER OPERATION EXAMPLE

- **T0 TIMER CONFIGURATION:**

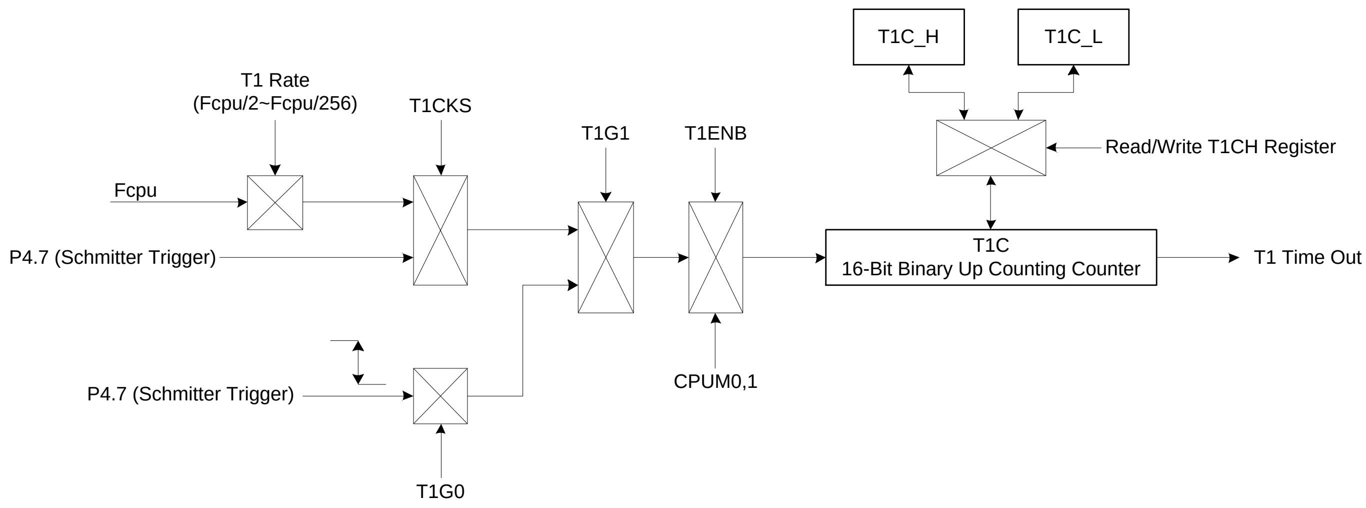
```
; Reset T0 timer.
        MOV         A, #0x00        ; Clear T0M register.
        B0MOV       T0M, A

; Set T0 clock source and T0 rate.
        MOV         A, #0nnn0000b
        B0MOV       T0M, A

; Set T0C register for T0 Interval time.
        MOV         A, #value
        B0MOV       T0C, A

; Clear T0IRQ
        B0BCLR      FT0IRQ

; Enable T0 timer and interrupt function.
        B0BSET      FT0IEN          ; Enable T0 interrupt function.
        B0BSET      FT0ENB          ; Enable T0 timer.
```

8.3 T1 16-BIT BASIC TIMER

8.3.1 OVERVIEW

The T1 is a 16-bit binary up timer and event counter. If T1 timer occurs an overflow (from FFFFH to 0000H), it will continue counting and issue a time-out signal to trigger T1 interrupt to request interrupt service.

- **16-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **External event counter:** Counts system "events" of external clock signals at the P4.7/T1IN input pin. The trigger edge is controlled by T1G1, T1G0 bits.
- **Input signal plus width measurement:** In T1G1 = 1 of event counter mode, the input signal high and low plus width from T1IN can be measure by T1C_H, T1C_L buffers.



- **T1 purpose control table.**

T1 purpose	T1 clock source	T1CKS	T1G1	T1G0	Edge Direction
16-bit timer	Fcpu	0	0	-	-
16-bit event counter (P4.7 input frequency measurement)	P4.7/T1IN	1	0	0	Falling edge.
				1	Rising edge.
P4.7 pulse width measurement	P4.7/T1IN	0	1	0	High pulse width measurement.
				1	Low pulse width measurement.

Note: In pulse width measurement mode, the T1CKS must be set as "0" by program.

8.3.2 T1M MODE REGISTER

T1M is T1 timer mode control register to configure T1 operating mode including T1 pre-scaler, clock source…These configurations must be setup completely before enabling T1 timer.

0B5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1rate2	T1rate1	T1rate0	T1CKS	T1ARC	T1G1	T1G0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **T1G0:** T1G0: T1 trigger direction select bit.
0 = Falling edge trigger (T1G1=0). High pulse width measurement (T1G1=1).
1 = Rising edge trigger (T1G1=0). Low pulse width measurement (T1G1=1).

Bit 1 **T1G1:** T1 pulse width measurement control bit.
0 = Disable T1 pulse width measurement.
1 = Enable T1 pulse width measurement.

Bit 2 **T1ARC:** T1 timer clock source select bits.
0 = Disable. T1 would stop counting after T1C_H:T1C_L overflow (0xFFFF to 0x0000) occurs.
1 = Enable. T1 would automatically re-count after T1C_H:T1C_L overflow (0xFFFF to 0x0000) occurs.

Bit 3 **T1CKS:** T1 timer clock source select bit.
0 = Fcpu.
1 = External clock source from P4.7.

Bit [6:4] **T1RATE[2:0]:** T1 timer clock source select bits.
000 = Fcpu/256, 001 = Fcpu/128, 010 = Fcpu/64, 011 = Fcpu/32, 100 = Fcpu/16, 101 = Fcpu/8, 110 = Fcpu/4,111 = Fcpu/2.

Bit 7 **T1ENB:** T1 counter control bit.
0 = Disable T1 timer.
1 = Enable T1 timer.

8.3.3 T1C COUNTING REGISTER

T1C_L and T1C_H are T1 16-bit counter. When T1C overflow occurs, the T1IRQ flag is set as "1" and cleared by program. The T1C_L and T1C_H decide T1 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T1C register, and then enable T1 timer to make sure the first cycle correct. After one T1 overflow occurs, the T1C_L and T1C_H register is loaded a correct value by program.

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1C_L	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0B7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1C_H	T1C15	T1C14	T1C135	T1C12	T1C11	T1C10	T1C9	T1C8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of T1C initial value is as following.

T1C initial value = 256 - (T1 interrupt interval time * T1 clock rate)

- **Example: To set 1ms interval time for T1 interrupt. High clock is 16MHz. Fcpu=Fosc/2. Select T1RATE=001 (Fcpu/128).**
  T1 interval time = 1s. T1 clock rate = 16MHz/2/128

*T1C initial value = 65536 - (T1 interval time * input clock)*
*= 65536 - (1s * 16MHz / 2 / 128)*
= 65536- (1 16 * 10^6 / 2/ 128)*
= 3036
= BDCH

The basic timer table interval time of T1.

T1RATE	T1CLOCK	High speed mode (Fcpu = 16MHz / 2)	
		Max overflow interval	One step = max/256
000	Fcpu/256	2.097 s	32.00 us
001	Fcpu/128	1.048 s	16.00 us
010	Fcpu/64	524.288 ms	8.00 us
011	Fcpu/32	262.144 ms	4.00 us
100	Fcpu/16	131.072 ms	2.00 us
101	Fcpu/8	65.536 ms	1.00 us
110	Fcpu/4	32.768 ms	0.50 us
111	Fcpu/2	16.384 ms	0.25 us

8.3.4 T1 TIMER OPERATION EXAMPLE

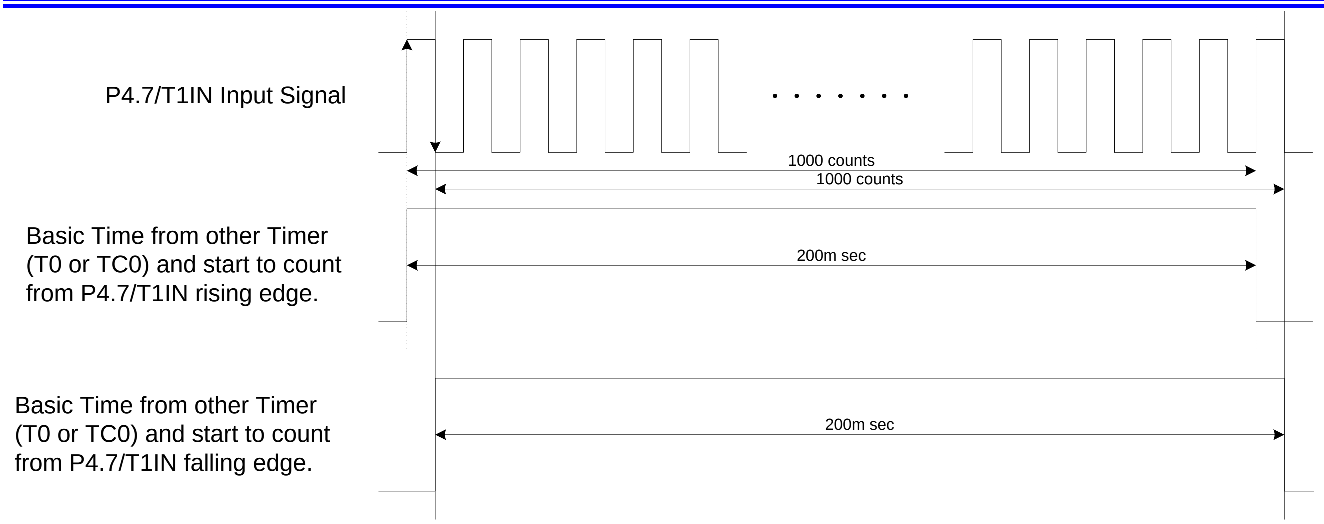
- **Stop T1 timer counting, disable T1 interrupt function and clear T1 interrupt request flag.**

```
; Reset T1 timer.
            B0BCLR      FT1ENB          ; T1 timer.
            B0BCLR      FT1IEN          ; T1 interrupt function is disabled.
            B0BCLR      FT1IRQ          ; T1 interrupt request flag is cleared.

; Set T1 timer rate.
            MOV         A, #0xxx0000b   ;The T1 rate control bits exist in bit4~bit6 of T1M. The
                                        ; value is from x000xxxxb~x111xxxxb.
            B0MOV       T1M, A          ; T1 timer is disabled.

; Set T1 interrupt interval time.
            MOV         A,#0E5H
            B0MOV       T1C_L, A        ; Set T1C_L value.
            MOV         A,#48H
            B0MOV       T1C_H, A        ; Set T1C_H value.

; Set T1 timer function mode.
            B0BSET      FT1IEN          ; Enable T1 interrupt function.

; Enable T1 timer
            B0BSET      FT1ENB          ; Enable T1 timer.
```

8.3.5 T1IN INPUT FREQUENCY MEASUREMENT

T1 is a 16-bit timer/counter. The 16-bit counter buffer (T1C_H, T1C_L) is long enough to measure high-speed signal's frequency from P4.7/T1IN input. For P4.7/T1IN input frequency measurement, T1 counts input signal clock number and using other timer (T0 to TC0) to generate one period time to be the basic time to calculate input frequency. When the period time is time out, disable T1 counter and the number of P4.7/T1IN input clock is stored in T1C_H and T1C_L registers. Using the basic time to divide by input clock number and get the one cycle period. The reciprocal of one cycle period is the input signal's frequency from P4.7/T1IN (F=1/T). If the T1C_H, T1C_L = 0x1388 =5000 and the basic time 50m sec, the input frequency is 100KHz. Input frequency = 1/ (50ms/5000) = 100KHz

- **T1 clock source is event counter input (P4.7/T1IN).**
- **T0 or TC0 clock source is internal clock (Fcpu).**



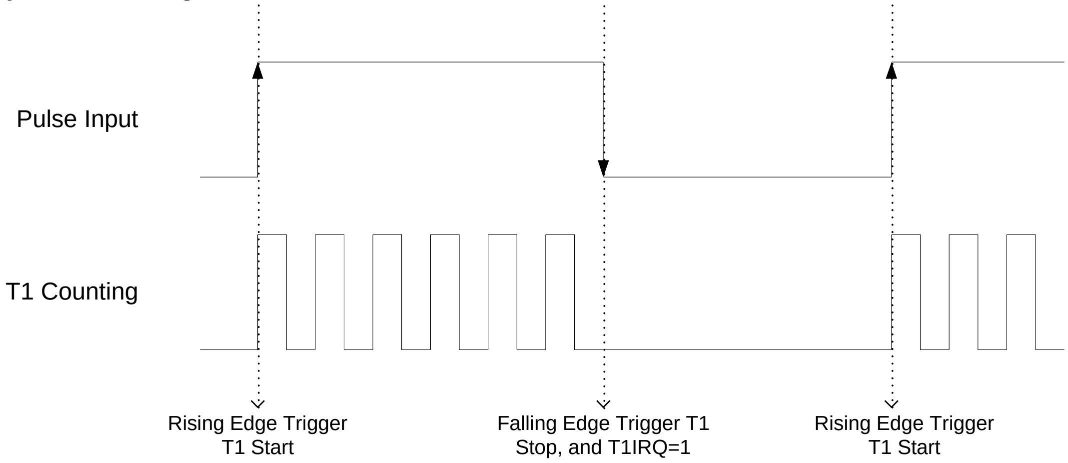
- **Example: Using T0 to measure P4.7/T1IN input 100KHz frequency. The basic time generated by T0 is 12.5ms. Fhosc=16MHz, Fcpu=Fhosc/4=4MHz. The P4.7/T1IN input trigger is rising edge.**

```
; Set T1 mode..
        CLR         T1M             ; Clear T1M register.
        MOV         A, #0
        B0MOV       T1C_L, A
        B0MOV       T1C_H, A        ; Clear T1 counter buffers.

        B0BSET      FT1CKS          ; Set T1 to event counter mode.
        B0BSET      FT1G0           ; Event counter input trigger is rising edge.
        B0BCLR      FT1IRQ          ; Clear T1 interrupt request flag.


; Set T0 mode..
        B0BCLR      T0ENB           ; Disable T0 timer.
        B0BCLR      T0IEN           ; Disable T0 interrupt function.
        B0BCLR      T0IRQ           ; Disable T0 interrupt request flag.

        MOV         A, #00h
        B0MOV       T0M, A          ; Set T0 rate is Fcpu/256.

        MOV         A, #61
        B0MOV       T0C, A          ; Set T0 interval time to 12.5ms.

; Start to measure P4.7/T1IN input frequency.
; Chk_1st_Eg:
        B0BTS1      P0.1            ; Check P0.1 rising edge.
        JMP         Chk_1st_Eg
        B0BSET      FT1ENB          ; Enable T1 counter.
        B0BSET      FT0ENB          ; Enable T0 timer.

Chk_50ms
        B0BTS1      FT0IRQ          ; Check T0 12.5ms. time out.
        JMP         Chk_50ms
        B0BCLR      FT1ENB          ; Stop T1 counter.

Cal_Freq:                           ; Calculate P4.7/T1IN input frequency from T1C_H, T1C_L.
        B0MOV       A, T1C_H        ; T1C_H = 13H
        …
        B0MOV       A, T1C_L        ; T1C_L = 88H
        …
```

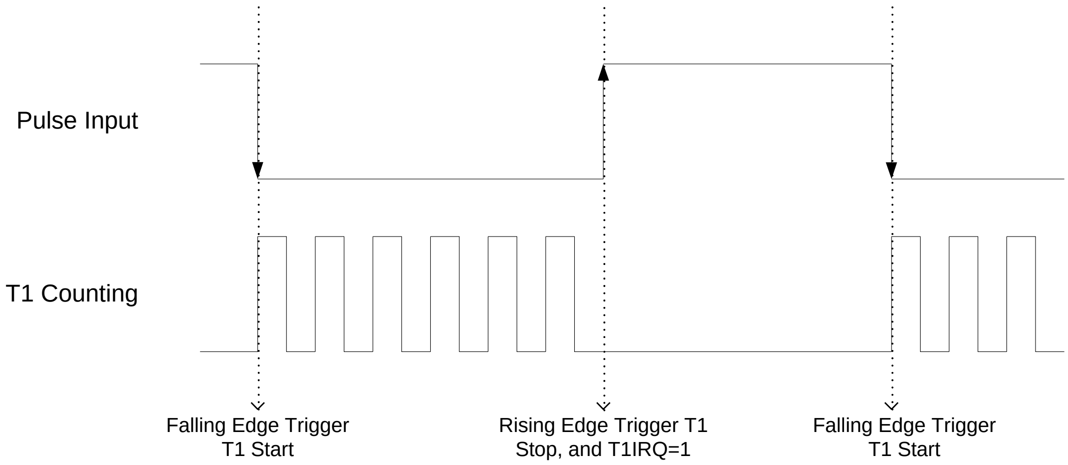
8.3.6 T1IN INPUT PULSE WIDTH MEASUREMENT

T1 builds in pulse width measurement function and controlled by T1G1 bit of T1M register. T1G1=0, disable pulse width measurement. T1G1=1, enable pulse width measurement. The input signal is from P4.7/T1IN or RFC. Please refer to "RFC DESCRIPTION" for RFC pulse width measurement. The section explains the pulse width measurement of P4.7/T1IN. The pulse includes high and low directions and controlled by T1G0 bit of T1M register. The time base of pulse width measurement is T1 and the clock source is from internal clock (Fcpu). T1IRQ=0, T1 starts to count when trigger occurrence. T1IRQ=1, T1 stops counting and T1C_H, T1C_L data are latched. T1IRQ is cleared by program and T1 starts to count automatically. The T1C_H, T1C_L dumped by program is during T1 stopping period, or T1 is counting and the T1C data is counted.

☞ **T1G1, T1G0=10: Enable high pulse width measurement. Rising edge trigger starts T1 to count and falling edge stops T1 counting.**



☞ **T1G1, T1G0=11: Enable low pulse width measurement. Falling edge trigger starts T1 to count and rising edge stops T1 counting.**



- **Example: Measure 50Hz clock input pulse width from P4.7/T1IN pin using T1 pulse width measurement. Fhosc=16MHz, Fcpu=Fhosc/4=4MHz. T1 base time is 64us. Pulse width is T1C buffer * T1 base time. T1C_H, T1C_L = 0x0027=39, Pulse width = 64us *39 = 2.496ms. (Pulse width of 50Hz signal is 10ms)**

High Pulse Width Measurement.

```
; Set T1 mode.
        CLR         T1M             ; Clear T1M register and set T1 rate is Fcpu/256.
        MOV         A, #0
        B0MOV       T1C_L, A
        B0MOV       T1C_H, A        ; Clear T1 counter buffers.

        B0BSET      FT1G1           ; Enable pulse width measurement.
        B0BCLR      FT1G0           ; Select high pulse direction.
        B0BCLR      FT1IRQ
        B0BSET      FT1ENB          ; Enable T1 timer.
```

```
;Check T1IRQ=1.
Chk_T1IRQ:
            B0BTS1      FT1IRQ          ; Check T1IRQ=1.
            JMP         Chk_T1IRQ
            B0MOV       A, T1C_H        ; Pulse width measure end.
            …                           ; T1C_H=00h
            B0MOV       A, T1C_L        ; T1C_L=27h
            …
```

;Measure next high pulse.

```
            MOV         A, #0           .
            B0MOV       T1C_L, A
            B0MOV       T1C_H, A        ; Clear T1 counter buffers.
            B0CLR       T1IRQ           ; Clear T1 interrupt request flag and start T1 again.
            JMP         Chk_T1IRQ
```

Low Pulse Width Measurement.

```
; Set T1 mode.
            CLR         T1M             ; Clear T1M register and set T1 rate is Fcpu/256.
            MOV         A, #0
            B0MOV       T1C_L, A
            B0MOV       T1C_H, A        ; Clear T1 counter buffers.

            B0BSET      FT1G1           ; Enable pulse width measurement.
            B0BSET      FT1G0           ; Select low pulse direction.
            B0BCLR      FT1IRQ
            B0BSET      FT1ENB          ; Enable T1 timer.

;Check T1IRQ=1.
Chk_T1IRQ:
            B0BTS1      FT1IRQ          ; Check T1IRQ=1.
            JMP         Chk_T1IRQ
            B0MOV       A, T1C_H        ; Pulse width measure end.
            …                           ; T1C_H=00h
            B0MOV       A, T1C_L        ; T1C_L=27h
            …
```

;Measure next high pulse.

```
            MOV         A, #0           .
            B0MOV       T1C_L, A
            B0MOV       T1C_H, A        ; Clear T1 counter buffers.
            B0CLR       T1IRQ           ; Clear T1 interrupt request flag and start T1 again.
            JMP         Chk_T1IRQ
```

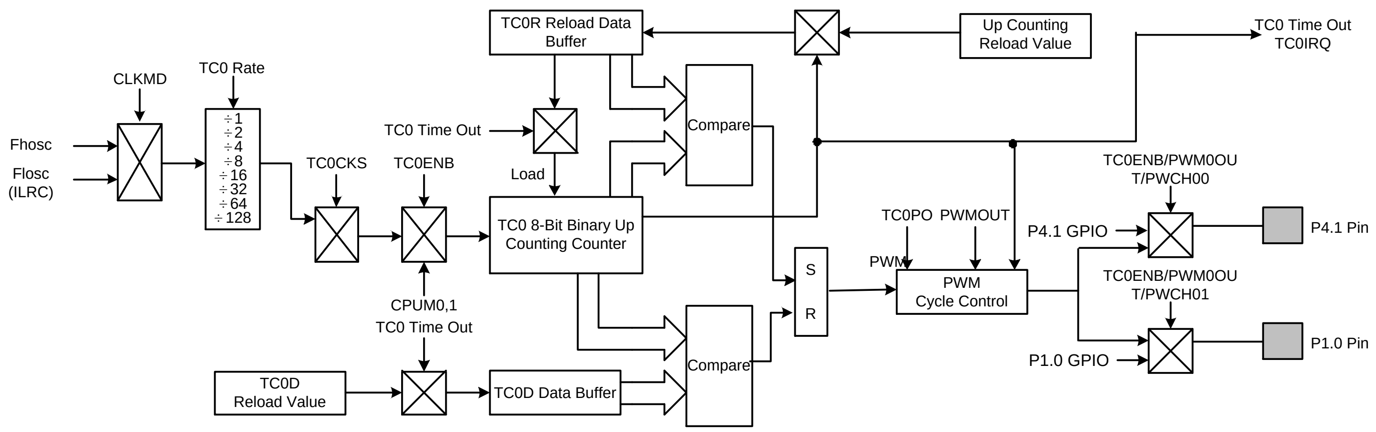
8.4 TC0 8-BIT TIMER/COUNTER

8.4.1 OVERVIEW

The TC0 timer is an 8-bit binary up timer and supports normal 8-bit timer, event counter, PWM, One Pulse PWM function. If TC0 timer occurs an overflow (from 0xFF to 0x00), it will continue counting and issue a time-out signal to indicate TC0 time out event. TC0 builds in event counter function. The clock source is from external input pin (P1.2) controlled by TC0CKS. When TC0CKS = 1, TC0 clock source is selected from P4.0 and enables event counter function. TC0 builds in PWM function. The PWM is duty/cycle programmable controlled by TC0R and TC0D registers. It is easy to implement buzzer, PWM and IR carry signal. TC0 PWM function also supports one pulse output signal that means only output one cycle PWM signal, not continuous. The enabled PWM channel exchanges from GPIO to PWM output. TC0 counter supports auto-reload function which always enabled. When TC0 timer overflow occurs, the TC0C will be reloaded from TC0R automatically. The auto-reload function is always enabled. The TC0 doesn't build in green mode wake-up function.

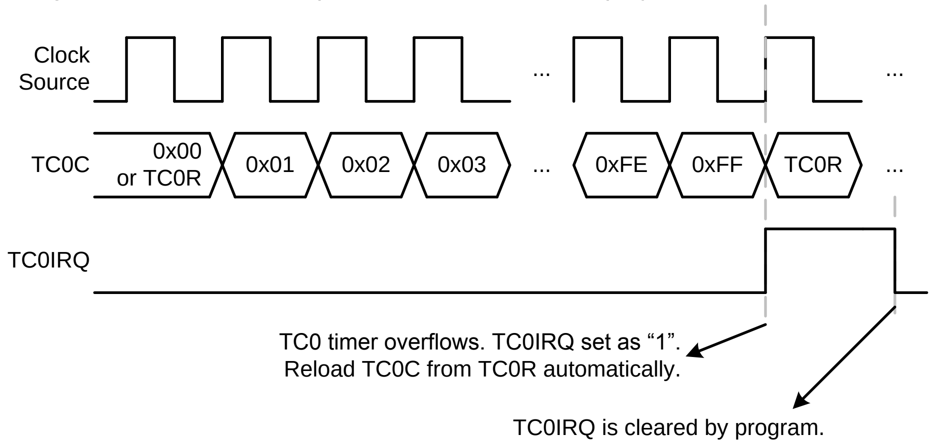
The main purposes of the TC0 timer are as following.

- **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** TC0 timer function supports interrupt function. When TC0 timer occurs overflow, the TC0IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Event Counter:** The event counter function counts the external clock counts.
- **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by TC0R and TC0D registers.
- **One Pulse PWM:** The one pulse PWM is controlled by TC0PO bit. When TC0PO=0, TC0 is normal timer mode or PWM function mode. When TC0PO=1, TC0 is one pulse PWM function. When PWMOUT=1, one pulse PWM outputs and the TC0IRQ is issued as TC0 counter overflow, PWMOUT bit is cleared automatically, the PWM channel returns to GPIO mode and last status.
- **2-Channel PWM output:** The 2-channel PWM output are controlled by PWCH[1:0] bits.
- **Green mode function:** All TC0 functions (timer, PWM, event counter, auto-reload, extension) keeps running in green mode, but no wake-up function. Timer IRQ actives as any IRQ trigger occurrence, e.g. timer overflow…



8.4.2 TC0 TIMER OPERATION

TC0 timer is controlled by TC0ENB bit. When TC0ENB=0, TC0 timer stops. When TC0ENB=1, TC0 timer starts to count. Before enabling TC0 timer, setup TC0 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…TC0C increases "1" by timer clock source. When TC0 overflow event occurs, TC0IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is TC0C count from full scale (0xFF) to zero scale (0x00). In difference function modes, TC0C value relates to operation. If TC0C value changing effects operation, the transition of operations would make timer function error. So TC0 builds in double buffer to avoid these situations happen. The double buffer concept is to flash TC0C during TC0 counting, to set the new value to TC0R (reload buffer), and the new value will be loaded from TC0R to TC0C after TC0 overflow occurrence automatically. In the next cycle, the TC0 timer runs under new conditions, and no any transitions occur. The auto-reload function is no any control interface and always actives as TC0 enables. If TC0 timer interrupt function is enabled (TC0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 0011H) and executes interrupt service routine after TC0 overflow occurrence. Clear TC0IRQ by program is necessary in interrupt procedure. TC0 timer can works in normal mode, slow mode and green mode. Under green mode, TC0 keep counting, set TC0IRQ and outputs PWM, can't wake-up system.



TC0 provides different clock sources to implement different applications and configurations. TC0 clock source includes Fcpu (instruction cycle), Fhosc (high speed oscillator) and external input pin (P1.2) controlled by TC0CKS[1:0] bits. TC0CKS0 bit selects the clock source is from Fcpu or Fhosc. If TC0CKS0=0, TC0 clock source is Fcpu through TC0rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. If TC0CKS0=1, TC0 clock source is Fhosc through TC0rate[2:0] pre-scalar to decide Fhosc/1~Fhosc/128. TC0CKS1 bit controls the clock source is external input pin or controlled by TC0CKS0 bit. If TC0CKS1=0, TC0 clock source is selected by TC0CKS0 bit. If TC0CKS1=1, TC0 clock source is external input pin that means to enable event counter function. TC0rate[2:0] pre-scalar is unless when TC0CKS1=1 conditions. TC0 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

TC0CKS0	TC0rate[2:0]	TC0 Clock	TC0 Interval Time			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	8.192	32	32.768	128
0	001b	Fcpu/64	4.096	16	16.384	64
0	010b	Fcpu/32	2.048	8	8.192	32
0	011b	Fcpu/16	1.024	4	4.096	16
0	100b	Fcpu/8	0.512	2	2.048	8
0	101b	Fcpu/4	0.256	1	1.024	4
0	110b	Fcpu/2	0.128	0.5	0.512	2
0	111b	Fcpu/1	0.064	0.25	0.256	1
1	000b	Fhosc/128	2.048	8	8.192	32
1	001b	Fhosc/64	1.024	4	4.096	16
1	010b	Fhosc/32	0.512	2	2.048	8
1	011b	Fhosc/16	0.256	1	1.024	4
1	100b	Fhosc/8	0.128	0.5	0.512	2
1	101b	Fhosc/4	0.064	0.25	0.256	1
1	110b	Fhosc/2	0.032	0.125	0.128	0.5
1	111b	Fhosc/1	0.016	0.0625	0.064	0.25

8.4.3 TC0M MODE REGISTER

TC0M is TC0 timer mode control register to configure TC0 operating mode including TC0 pre-scalar, clock source, PWM function…These configurations must be setup completely before enabling TC0 timer.

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS1	TC0CKS0	TC0PO	PWM0OUT
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **PWM0OUT:** PWM output control bit.
0 = Disable PWM output function, and P4.1 & P1.0 (controlled by PWCH[1:0] bits) are GPIO mode.
1 = Enable PWM output function, and P4.1 & P1.0 (controlled by PWCH[1:0] bits) output PWM signal.

Bit 1 **TC0PO:** TC0 pulse output function control bit.
0 = Disable TC0 pulse output function.
1 = Enable TC0 pulse output function.

Bit 2 **TC0CKS0:** TC0 clock source select bit.
0 = Fcpu.
1 = Fhosc.

Bit 3 **TC0CKS1:** TC0 clock source select bit.
0 = Internal clock (Fcpu and Fhosc controlled by TC0CKS0 bit).
1 = External input pin (P1.2) and enable event counter function. **TC0rate[2:0] bits are useless.**

Bit [6:4] **TC0RATE[2:0]:** TC0 timer clock source select bits.
TC0CKS0=0 -> 000 = Fcpu/128, 001 = Fcpu/64, 010 = Fcpu/32, 011 = Fcpu/16, 100 = Fcpu/8, 101 = Fcpu/4, 110 = Fcpu/2,111 = Fcpu/1.
TC0CKS0=1 -> 000 = Fhosc/128, 001 = Fhosc/64, 010 = Fhosc/32, 011 = Fhosc/16, 100 = Fhosc/8, 101 = Fhosc/4, 110 = Fhosc/2,111 = Fhosc/1.

Bit 7 **TC0ENB:** TC0 timer control bit.
0 = Disable TC0 timer.
1 = Enable TC0 timer.

8.4.4 TC0C COUNTING REGISTER

TC0C is TC0 8-bit counter. When TC0C overflow occurs, the TC0IRQ flag is set as “1” and cleared by program. The TC0C decides TC0 interval time through below equation to calculate a correct value. It is necessary to write the correct value to TC0C register and TC0R register first time, and then enable TC0 timer to make sure the fist cycle correct. After one TC0 overflow occurs, the TC0C register is loaded a correct value from TC0R register automatically, not program.

0B9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0C	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of TC0C initial value is as following.

TC0C initial value = 256 - (TC0 interrupt interval time * TC0 clock rate)

8.4.5 TC0R AUTO-RELOAD REGISTER

TC0 timer builds in auto-reload function, and TC0R register stores reload data. When TC0C overflow occurs, TC0C register is loaded data from TC0R register automatically. Under TC0 timer counting status, to modify TC0 interval time is to modify TC0R register, not TC0C register. New TC0C data of TC0 interval time will be updated after TC0 timer overflow occurrence, TC0R loads new value to TC0C register. But at the first time to setup TC0M, TC0C and TC0R must be set the same value before enabling TC0 timer. TC0 is double buffer design. If new TC0R value is set by program, the new value is stored in 1st buffer. Until TC0 overflow occurs, the new value moves to real TC0R buffer. This way can avoid any transitional condition to affect the correctness of TC0 interval time and PWM output signal.

0BAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of TC0R initial value is as following.

TC0R initial value = 256 - (TC0 interrupt interval time * TC0 clock rate)

☞ **Example: To calculation TC0C and TC0R value to obtain 100us TC0 interval time. TC0 clock source is Fhosc = 16MHz. Select TC0RATE = 011 (Fosc/16).**
TC0 interval time = 100us. TC0 clock rate = 16MHz/16

*TC0C/TC0R initial value = 256 - (TC0 interval time * input clock)*
*= 256 - (100us * 16MHz / 16)*
*= 256 - (100 * 10^{-6} * 16 * 10^{6} / 16)*
= 9CH

8.4.6 TC0D PWM DUTY REGISTER

TC0D register's purpose is to decide PWM duty. In PWM mode, TC0R controls PWM's cycle, and TC0D controls the duty of PWM. The operation is base on timer counter value. When TC0C = TC0D, the PWM high duty finished and exchange to low level. It is easy to configure TC0D to choose the right PWM's duty for application.

0BBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0D	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

The equation of TC0D initial value is as following.

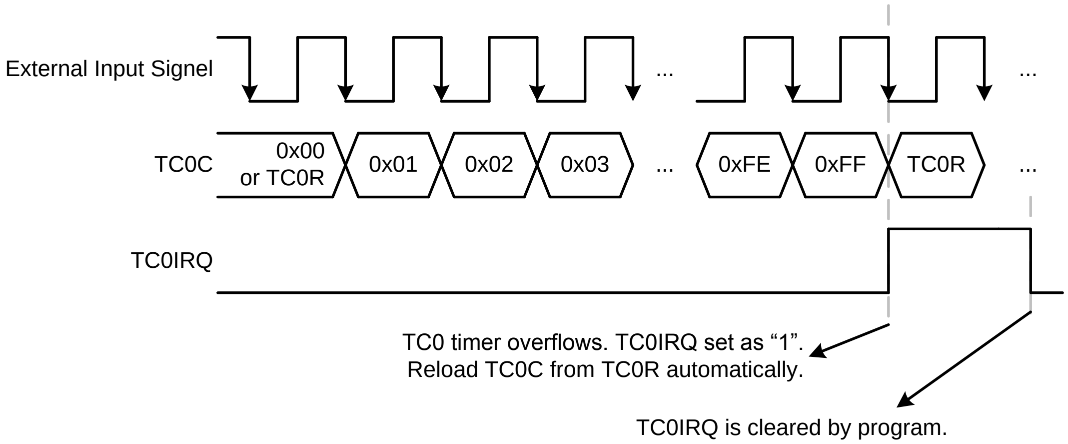
TC0D initial value = TC0R + (PWM high pulse width period / TC0 clock rate)

☞ **Example: To calculate TC0D value to obtain 1/3 duty PWM signal. The TC0 clock source is Fhosc = 16MHz. Select TC0RATE=000 (Fosc/128).**
TC0R = 9CH. TC0 interval time = 800us. So the PWM cycle is 1.25KHz. In 1/3 duty condition, the high pulse width is about 267us.

TC0D initial value = 9CH + (PWM high pulse width period / TC0 clock rate)
*= 9CH + (267us * 16MHz / 128)*
= 9CH + 21H
= BDH

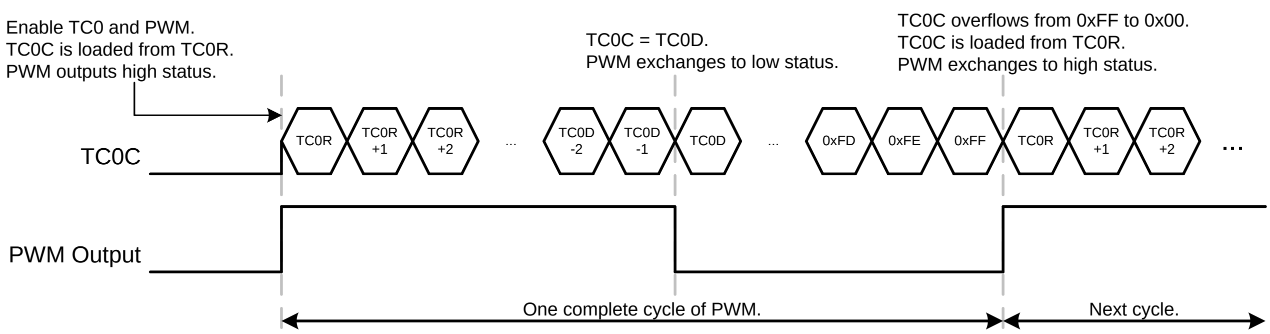
8.4.7 TC0 EVENT COUNTER

TC0 event counter is set the TC0 clock source from external input pin (P1.2). When TC0CKS1=1, TC0 clock source is switch to external input pin (P1.2). TC0 event counter trigger direction is falling edge. When one falling edge occurs, TC0C will up one count. When TC0C counts from 0xFF to 0x00, TC0 triggers overflow event. The external event counter input pin's wake-up function of GPIO mode is disabled when TC0 event counter function enabled to avoid event counter signal trigger system wake-up and not keep in power saving mode. The external event counter input pin's external interrupt function is also disabled when TC0 event counter function enabled, and the P12IRQ bit keeps "0" status. The event counter usually is used to measure external continuous signal rate, e.g. continuous pulse, R/C type oscillating signal…These signal phase don't synchronize with MCU's main clock. Use TC0 event to measure it and calculate the signal rate in program for different applications.



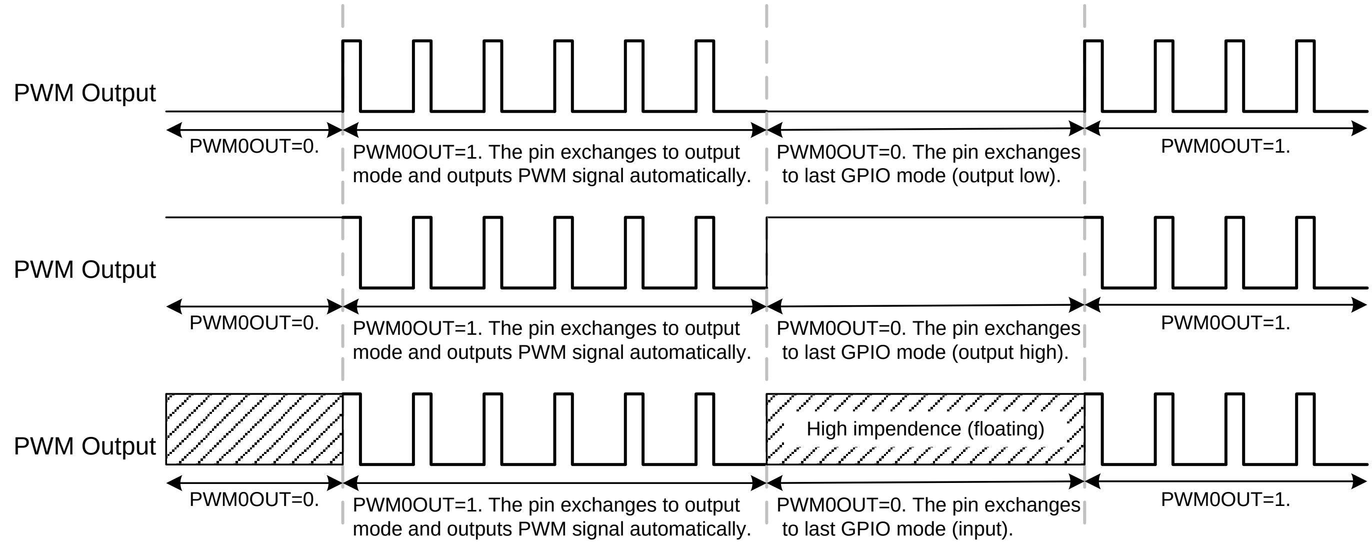
8.4.8 PULSE WIDTH MODULATION (PWM)

The PWM is duty/cycle programmable design to offer various PWM signals. When TC0 timer enables before, PWM0OUT bit sets as "1" (enable PWM output), and select PWM output pin (PWCH[1:0]), the PWM output pin (P4.1, P1.0) outputs PWM signal. When TC0ENB = 0 or PWM0OUT = 0, the PWM channel returns to GPIO mode and last status. One cycle of PWM signal is high pulse first, and then low pulse outputs. TC0R register controls the cycle of PWM, and TC0D decides the duty (high pulse width length) of PWM. TC0C initial value is TC0R reloaded when TC0 timer enables and TC0 timer overflows. When TC0C count is equal to TC0D, the PWM high pulse finishes and exchanges to low level. When TC0 overflows (TC0C counts from 0xFF to 0x00), one complete PWM cycle finishes. The PWM exchanges to high level for next cycle. The PWM is auto-reload design to load TC0C from TC0R automatically when TC0 overflows and the end of PWM's cycle, to keeps PWM continuity. If modify the PWM duty/cycle by program as PWM outputting, the new cycle occurs at next cycle when TC0C loaded from TC0R. The PWM channels selected by PWCH[1:0] bits. PWM00, PWM01 output pin is P4.1, P1.0. The PWM00, PWM01 output pins are shared with GPIO pin controlled by PWCH[1:0] bits.



The resolution of PWM is decided by TC0R. TC0R range is from 0x00~0xFF. If TC0R = 0x00, PWM's resolution is 1/256. If TC0R = 0x80, PWM's resolution is 1/128. TC0D controls the high pulse width of PWM for PWM's duty. When TC0C = TC0D, PWM output exchanges to low status. TC0D must be greater than TC0R, or the PWM signal keeps low status. When PWM outputs, TC0IRQ still actives as TC0 overflows, and TC0 interrupt function actives as TC0IEN = 1. But strongly recommend be careful to use PWM and TC0 timer together, and make sure both functions work well.
The PWM output pin is shared with GPIO and switch to output PWM signal as PWM0OUT=1 automatically. If PWM0OUT bit is cleared to disable PWM, the output pin exchanges to last GPIO mode automatically. It easily to

implement carry signal on/off operation, not to control TC0ENB bit.



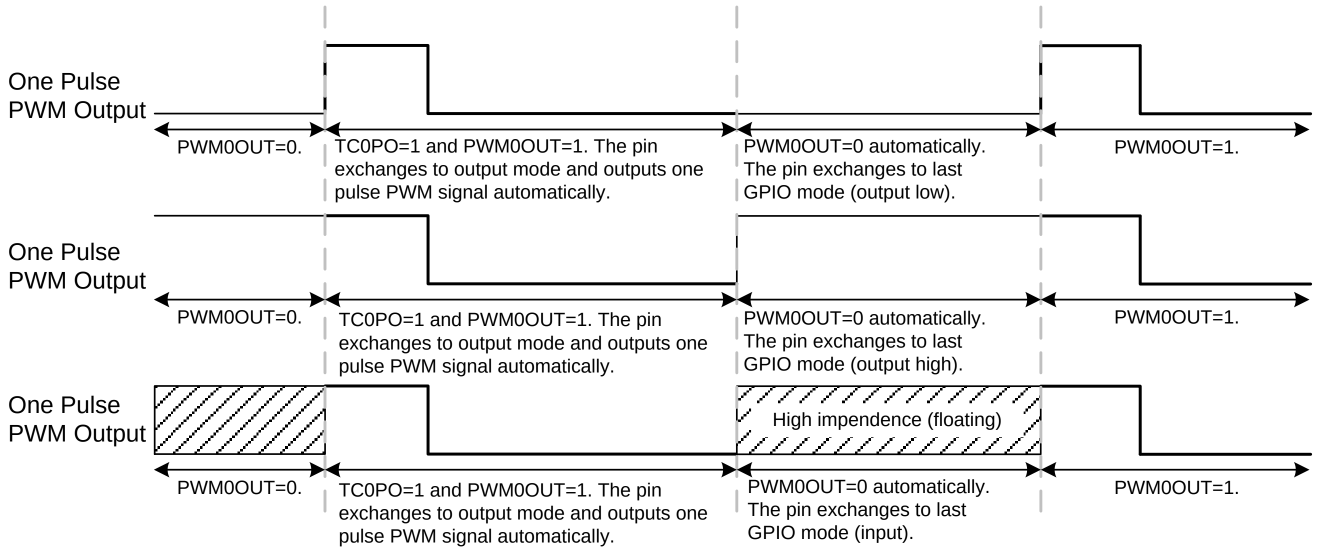
C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWCH	-	-	PWCH20	PWCH21	PWCH11	PWCH10	PWCH01	PWCH00
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	0	0	0	0	0	0

Bit 1 **PWCH01**: PWM11 control bit.
0 = P1.0 pin GPIO mode.
1 = PWM11 output.

Bit 0 **PWCH00**: PWM10 control bit.
0 = P4.1 pin GPIO mode.
1 = PWM10 output.

8.4.9 One Pulse PWM

When TC0PO = 0, TC0 is normal timer mode or PWM function mode. When TC0PO = 1 and PWM0OUT=1, TC0 will output one pulse PWM function. When one pulse PWM output signal finishes, the PWM channel exchanges from GPIO to PWM output. When one pulse PWM output finishes, PWM0OUT bit is cleared automatically, the PWM channel returns to GPIO mode and last status. TC0IRQ is issued as TC0 counter overflow. To output next pulse is to set PWM0OUT bit by program again.



8.4.10 TC0 TIMER OPERATION EXAMPLE

- **TC0 TIMER CONFIGURATION:**

```
; Reset TC0 timer.
        CLR       TC0M             ; Clear TC0M register.

; Set TC0 clock source and TC0 rate.
        MOV       A, #0nnnn000b
        B0MOV     TC0M, A

; Set TC0C and TC0R register for TC0 Interval time.
        MOV       A, #value        ; TC0C must be equal to TC0R.
        B0MOV     TC0C, A
        B0MOV     TC0R, A

; Clear TC0IRQ
        B0BCLR    FTC0IRQ

; Enable TC0 timer and interrupt function.
        B0BSET    FTC0IEN          ; Enable TC0 interrupt function.
        B0BSET    FTC0ENB          ; Enable TC0 timer.
```

- **TC0 EVENT COUNTER CONFIGURATION:**

```
; Reset TC0 timer.
        CLR       TC0M             ; Clear TC0M register.

; Enable TC0 event counter.
        B0BSET    FTC0CKS1         ; Set TC0 clock source from external input pin (P4.0).

; Set TC0C and TC0R register for TC0 Interval time.
        MOV       A, #value        ; TC0C must be equal to TC0R.
        B0MOV     TC0C, A
        B0MOV     TC0R, A

; Clear TC0IRQ
        B0BCLR    FTC0IRQ

; Enable TC0 timer and interrupt function.
        B0BSET    FTC0IEN          ; Enable TC0 interrupt function.
        B0BSET    FTC0ENB          ; Enable TC0 timer.
```

- **TC0 PWM CONFIGURATION:**

```
; Reset TC0 timer.
            CLR         TC0M            ; Clear TC0M register.

; Set TC0 clock source and TC0 rate.
            MOV         A, #0nnnn000b
            B0MOV       TC0M, A

; Set TC0C and TC0R register for PWM cycle.
            MOV         A, #value1      ; TC0C must be equal to TC0R.
            B0MOV       TC0C, A
            B0MOV       TC0R, A

; Set TC0D register for PWM duty.
            MOV         A, #value2      ; TC0D must be greater than TC0R.
            B0MOV       TC0D, A

; Set PWM channel.
            MOV         A, #000000nnb
            B0MOV       PWCH, A

; Enable PWM and TC0 timer.
            B0BSET      FTC0ENB         ; Enable TC0 timer.
            B0BSET      FPWM0OUT        ; Enable PWM.
```

- **TC0 One Pulse PWM CONFIGURATION:**

```
; Reset TC0 timer.
            CLR         TC0M            ; Clear TC0M register.

; Set TC0 clock source and TC0 rate.
            MOV         A, #0nnnn000b
            B0MOV       TC0M, A

; Set TC0C and TC0R register for PWM cycle.
            MOV         A, #value1      ; TC0C must be equal to TC0R.
            B0MOV       TC0C, A
            B0MOV       TC0R, A

; Set TC0D register for PWM duty.
            MOV         A, #value2      ; TC0D must be greater than TC0R.
            B0MOV       TC0D, A

; Set PWM channel.
            MOV         A, #000000nnb
            B0MOV       PWCH, A

; Enable PWM and TC0 timer.
            B0BSET      FTC0PO          ; Enable One Pulse.
            B0BSET      FTC0ENB         ; Enable TC0 timer.
            B0BSET      FPWM0OUT        ; Enable PWM.
```

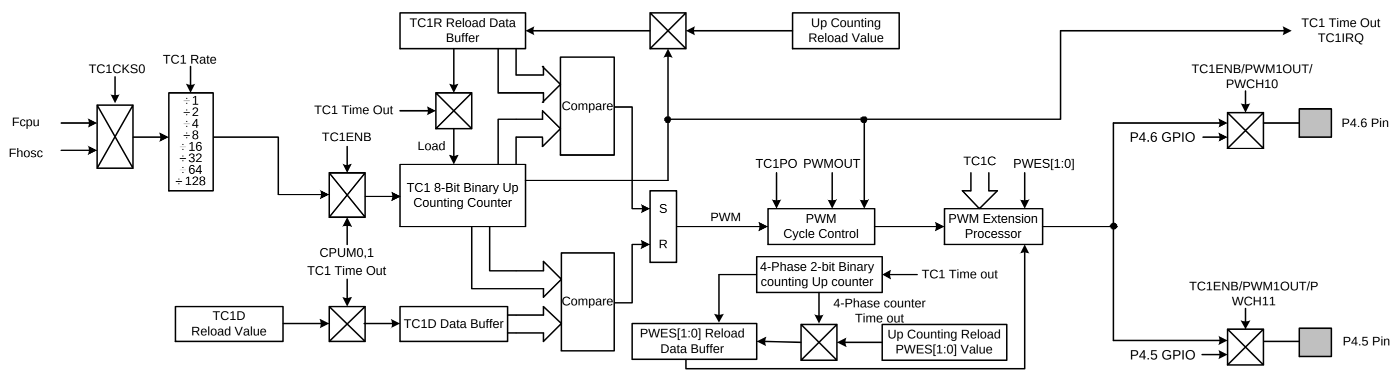
8.5 TC1 8-BIT TIMER/COUNTER

8.5.1 OVERVIEW

The TC1 timer is an 8-bit binary up timer with basic timer, PWM, One Pulse PWM and PWM with extension functions. The basic timer function supports flag indicator (TC1IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TC1M, TC1C, TC1R registers. The event counter is changing TC1 clock source from system clock (Fhosc/Fcpu) to external clock like signal (e.g. continuous pulse, R/C type oscillating signal…). TC1 becomes a counter to count external clock number to implement measure application. TC1 also builds in duty/cycle programmable PWM. The PWM cycle and resolution are controlled by TC1 timer clock rate, TC1R and TC1D registers. So the PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster…. TC1 PWM function also supports one pulse output signal that means only output one cycle PWM signal, not continuous. The PWM has two programmable channels shared with GPIO pins and controlled by PWCH[3:2] bit. The output operation must be through enabled each bit/channel of PWCH[3:2] bits. The enabled PWM channel exchanges from GPIO to PWM output. When the PWCH[3:2] bits disables, the PWM channel returns to GPIO mode and last status. And support the 2-channel PWM output with extension selected by PWES[1:0] bits. TC1 counter supports auto-reload function which always enabled. When TC1 timer overflow occurs, the TC1C will be reloaded from TC1R automatically. The auto-reload function is always enabled. The TC1 doesn't build in green mode wake-up function.

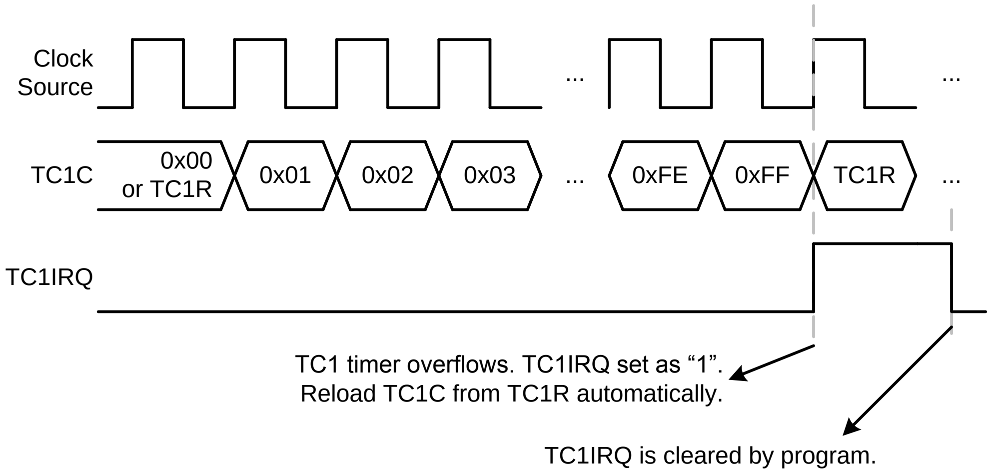
The main purposes of the TC1 timer are as following.

- **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** TC1 timer function supports interrupt function. When TC1 timer occurs overflow, the TC1IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by TC1R and TC1D registers.
- **One Pulse PWM:** The one pulse PWM is controlled by TC1PO bit. When TC1PO=0, TC1 is normal timer mode or PWM function mode. When TC1PO=1, TC1 is one pulse PWM function. When PWMOUT=1, one pulse PWM outputs and the TC1IRQ is issued as TC1 counter overflow, PWMOUT bit is cleared automatically, the PWM channel returns to GPIO mode and last status. **One Pulse PWM doesn't support extension function (PWES[1:0]).**
- **2-Channel PWM output:** The 2-channel PWM output are controlled by PWCH[3:2] bits.
- **PWM output with Extension function: The PWM with extension function is four phases design.** The PWM output with extension function are controlled by PWES[1:0] bits.
- **Green mode function:** All TC1 functions (timer, PWM, event counter, auto-reload, extension) keeps running in green mode, but no wake-up function. Timer IRQ actives as any IRQ trigger occurrence, e.g. timer overflow…



8.5.2 TC1 TIMER OPERATION

TC1 timer is controlled by TC1ENB bit. When TC1ENB=0, TC1 timer stops. When TC1ENB=1, TC1 timer starts to count. Before enabling TC1 timer, setup TC1 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…TC1C increases "1" by timer clock source. When TC1 overflow event occurs, TC1IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is TC1C count from full scale (0xFF) to zero scale (0x00). In difference function modes, TC1C value relates to operation. If TC1C value changing effects operation, the transition of operations would make timer function error. So TC1 builds in double buffer to avoid these situations happen. The double buffer concept is to flash TC1C during TC1 counting, to set the new value to TC1R (reload buffer), and the new value will be loaded from TC1R to TC1C after TC1 overflow occurrence automatically. In the next cycle, the TC1 timer runs under new conditions, and no any transitions occur. The auto-reload function is no any control interface and always actives as TC1 enables. If TC1 timer interrupt function is enabled (TC1IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 0012H) and executes interrupt service routine after TC1 overflow occurrence. Clear TC1IRQ by program is necessary in interrupt procedure. TC1 timer can works in normal mode, slow mode and green mode. Under green mode, TC1 keep counting, set TC1IRQ and outputs PWM, can't wake-up system.



TC1 provides different clock sources to implement different applications and configurations. TC1 clock source includes Fcpu (instruction cycle), Fhosc (high speed oscillator) controlled by TC1CKS0 bit. TC1CKS0 bit selects the clock source is from Fcpu or Fhosc. If TC1CKS0=0, TC1 clock source is Fcpu through TC1rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. If TC1CKS0=1, TC1 clock source is Fhosc through TC1rate[2:0] pre-scalar to decide Fhosc/1~Fhosc/128. TC1 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

TC1CKS0	TC1rate[2:0]	TC1 Clock	TC1 Interval Time			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	8.192	32	32.768	128
0	001b	Fcpu/64	4.096	16	16.384	64
0	010b	Fcpu/32	2.048	8	8.192	32
0	011b	Fcpu/16	1.024	4	4.096	16
0	100b	Fcpu/8	0.512	2	2.048	8
0	101b	Fcpu/4	0.256	1	1.024	4
0	110b	Fcpu/2	0.128	0.5	0.512	2
0	111b	Fcpu/1	0.064	0.25	0.256	1
1	000b	Fhosc/128	2.048	8	8.192	32
1	001b	Fhosc/64	1.024	4	4.096	16
1	010b	Fhosc/32	0.512	2	2.048	8
1	011b	Fhosc/16	0.256	1	1.024	4
1	100b	Fhosc/8	0.128	0.5	0.512	2
1	101b	Fhosc/4	0.064	0.25	0.256	1
1	110b	Fhosc/2	0.032	0.125	0.128	0.5
1	111b	Fhosc/1	0.016	0.0625	0.064	0.25

8.5.3 TC1M MODE REGISTER

TC1M is TC1 timer mode control register to configure TC1 operating mode including TC1 pre-scalar, clock source, PWM function…These configurations must be setup completely before enabling TC1 timer.

0BCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1M	TC1ENB	TC1rate2	TC1rate1	TC1rate0	-	-	TC1PO	PWM1OUT
Read/Write	R/W	R/W	R/W	R/W	-	-	R/W	R/W
After reset	0	0	0	0	-	-	0	0

Bit 0 **PWM1OUT:** PWM output control bit.
0 = Disable PWM output function, and P4.6, P4.5 (controlled by PWCH[3:2] bits) are GPIO mode.
1 = Enable PWM output function, and P4.6, P4.5 (controlled by PWCH[3:2] bits) output PWM signal.

Bit 1 **TC1PO:** TC1 pulse output function control bit.
0 = Disable TC1 pulse output function.
1 = Enable TC1 pulse output function.

Bit [6:4] **TC1RATE[2:0]:** TC1 timer clock source select bits.
TC1CKS0=0 -> 000 = Fcpu/128, 001 = Fcpu/64, 010 = Fcpu/32, 011 = Fcpu/16, 100 = Fcpu/8, 101 = Fcpu/4, 110 = Fcpu/2,111 = Fcpu/1.
TC1CKS0=1 -> 000 = Fhosc/128, 001 = Fhosc/64, 010 = Fhosc/32, 011 = Fhosc/16, 100 = Fhosc/8, 101 = Fhosc/4, 110 = Fhosc/2,111 = Fhosc/1.

Bit 7 **TC1ENB:** TC1 timer control bit.
0 = Disable TC1 timer.
1 = Enable TC1 timer.

8.5.4 TC1C COUNTING REGISTER

TC1C is TC1 8-bit counter. When TC1C overflow occurs, the TC1IRQ flag is set as “1” and cleared by program. The TC1C decides TC1 interval time through below equation to calculate a correct value. It is necessary to write the correct value to TC1C register and TC1R register first time, and then enable TC1 timer to make sure the fist cycle correct. After one TC1 overflow occurs, the TC1C register is loaded a correct value from TC1R register automatically, not program.

0BDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1C	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of TC1C initial value is as following.

TC1C initial value = 256 - (TC1 interrupt interval time * TC1 clock rate)

8.5.5 TC1R AUTO-RELOAD REGISTER

TC1 timer builds in auto-reload function, and TC1R register stores reload data. When TC1C overflow occurs, TC1C register is loaded data from TC1R register automatically. Under TC1 timer counting status, to modify TC1 interval time is to modify TC1R register, not TC1C register. New TC1C data of TC1 interval time will be updated after TC1 timer overflow occurrence, TC1R loads new value to TC1C register. But at the first time to setup TC1M, TC1C and TC1R must be set the same value before enabling TC1 timer. TC1 is double buffer design. If new TC1R value is set by program, the new value is stored in 1st buffer. Until TC1 overflow occurs, the new value moves to real TC1R buffer. This way can avoid any transitional condition to affect the correctness of TC1 interval time and PWM output signal.

0BEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1R	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of TC1R initial value is as following.

TC1R initial value = 256 - (TC1 interrupt interval time * TC1 clock rate)

☞ **Example: To calculation TC1C and TC1R value to obtain 100us TC1 interval time. TC1 clock source is Fhosc = 16MHz. Select TC1RATE = 011 (Fosc/16).**
TC1 interval time = 100us. TC1 clock rate = 16MHz/16

*TC1C/TC1R initial value = 256 - (TC1 interval time * input clock)*
*= 256 - (100us * 16MHz / 16)*
*= 256 - (100 * 10^{-6} * 16 * 10^{6} / 16)*
= 9CH

8.5.6 TC1D PWM DUTY REGISTER

TC1D register's purpose is to decide PWM duty. In PWM mode, TC1R controls PWM's cycle, and TC1D controls the duty of PWM. The operation is base on timer counter value. When TC1C = TC1D, the PWM high duty finished and exchange to low level. It is easy to configure TC1D to choose the right PWM's duty for application.

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1D	TC1D7	TC1D6	TC1D5	TC1D4	TC1D3	TC1D2	TC1D1	TC1D0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

The equation of TC1D initial value is as following.

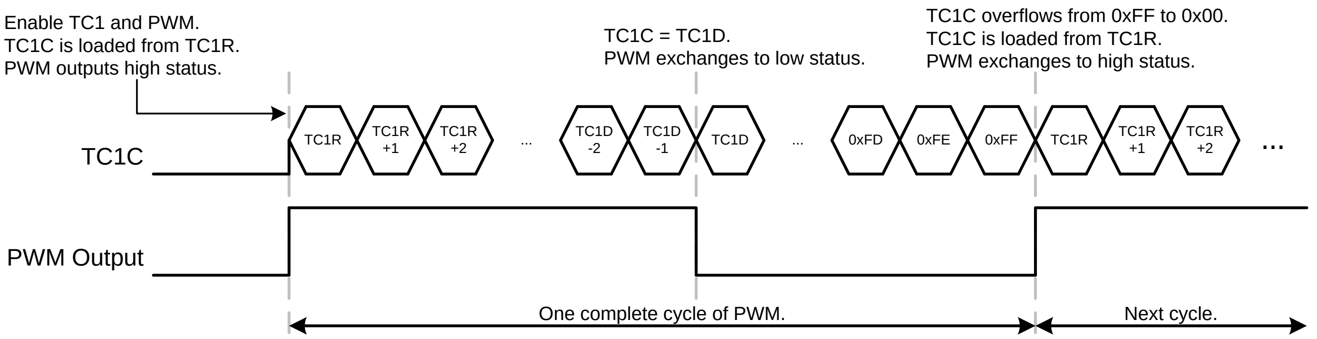
TC1D initial value = TC1R + (PWM high pulse width period / TC1 clock rate)

☞ **Example: To calculate TC1D value to obtain 1/3 duty PWM signal. The TC1 clock source is Fhosc = 16MHz. Select TC1RATE=000 (Fosc/128).**
TC1R = 9CH. TC1 interval time = 800us. So the PWM cycle is 1.25KHz. In 1/3 duty condition, the high pulse width is about 267us.

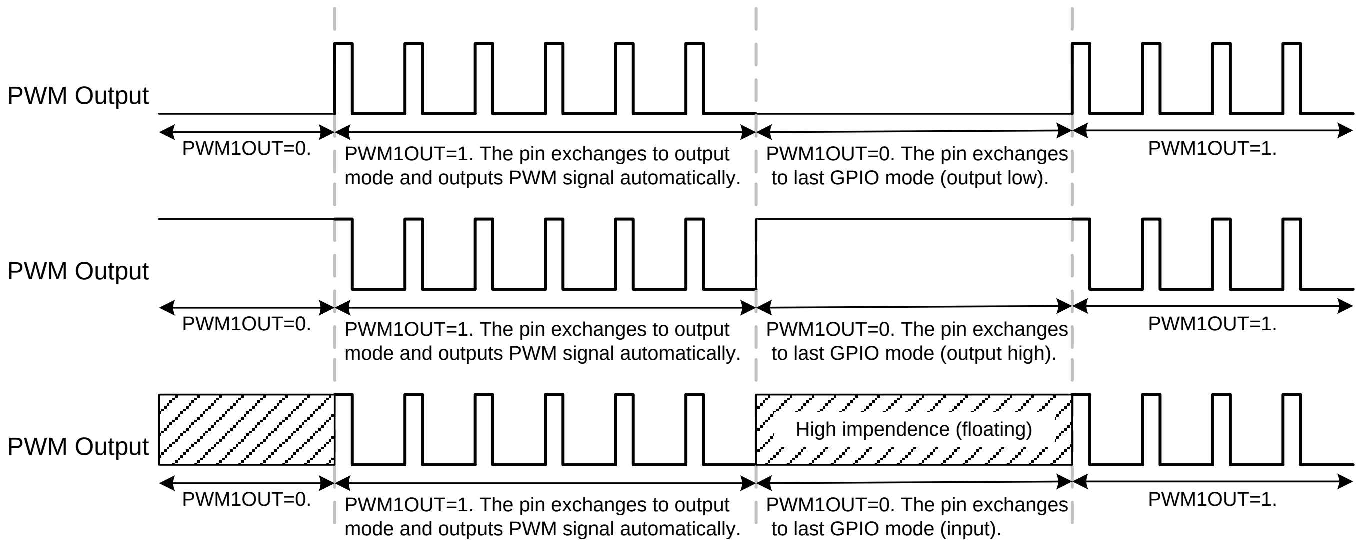
TC1D initial value = 9CH + (PWM high pulse width period / TC1 clock rate)
*= 9CH + (267us * 16MHz / 128)*
= 9CH + 21H
= BDH

8.5.7 PULSE WIDTH MODULATION (PWM)

The PWM is duty/cycle programmable design to offer various PWM signals. When TC1 timer enables before, PWM1OUT bit sets as “1” (enable PWM output) and select PWM output pin (PWCH[3:2]), the PWM output pin (P4.6, P4.5) outputs PWM signal. When TC1ENB = 0 or PWM1OUT = 0, the PWM channel returns to GPIO mode and last status. One cycle of PWM signal is high pulse first, and then low pulse outputs. TC1R register controls the cycle of PWM, and TC1D decides the duty (high pulse width length) of PWM. TC1C initial value is TC1R reloaded when TC1 timer enables and TC1 timer overflows. When TC1C count is equal to TC1D, the PWM high pulse finishes and exchanges to low level. When TC1 overflows (TC1C counts from 0xFF to 0x00), one complete PWM cycle finishes. The PWM exchanges to high level for next cycle. The PWM is auto-reload design to load TC1C from TC1R automatically when TC1 overflows and the end of PWM’s cycle, to keeps PWM continuity. If modify the PWM duty/cycle by program as PWM outputting, the new cycle occurs at next cycle when TC1C loaded from TC1R. The PWM channels selected by PWCH[3:2] bits. PWM10, PWM11 output pin is P4.6, P4.5. The PWM10, PWM11 output pins are shared with GPIO pin controlled by PWCH[3:2] bits.



The resolution of PWM is decided by TC1R. TC1R range is from 0x00~0xFF. If TC1R = 0x00, PWM’s resolution is 1/256. If TC1R = 0x80, PWM’s resolution is 1/128. TC1D controls the high pulse width of PWM for PWM’s duty. When TC1C = TC1D, PWM output exchanges to low status. TC1D must be greater than TC1R, or the PWM signal keeps low status. When PWM outputs, TC1IRQ still actives as TC1 overflows, and TC1 interrupt function actives as TC1IEN = 1. But strongly recommend be careful to use PWM and TC1 timer together, and make sure both functions work well.
The PWM output pin is shared with GPIO and switch to output PWM signal as PWM1OUT=1 automatically. If PWM1OUT bit is cleared to disable PWM, the output pin exchanges to last GPIO mode automatically. It easily to implement carry signal on/off operation, not to control TC1ENB bit.



C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWCH	-	-	PWCH20	PWCH21	PWCH11	PWCH10	PWCH01	PWCH00
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	0	0	0	0	0	0

Bit 3 **PWCH11**: PWM11 control bit.
0 = P4.5 pin GPIO mode.
1 = PWM11 output.

Bit 2 **PWCH10**: PWM10 control bit.
0 = P4.6 pin GPIO mode.
1 = PWM10 output.

C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWES	-	-	-	-	PW2ES1	PW2ES0	PW1ES1	PW1ES0
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After Reset	-	-	-	-	0	0	0	0

Bit [1:0] **PWnES[1:0], n=1~2:** The PWMn output with extension control bits **(One pulse PWM function doesn't support)**.
00: None compensation.
01: Compensation 1 count at phase 1 of TC1 PWM output.
10: Compensation 1 count at phase 1/3 of TC1 PWM output.
11: Compensation 1 count at phase 1/2/3 of TC1 PWM output.

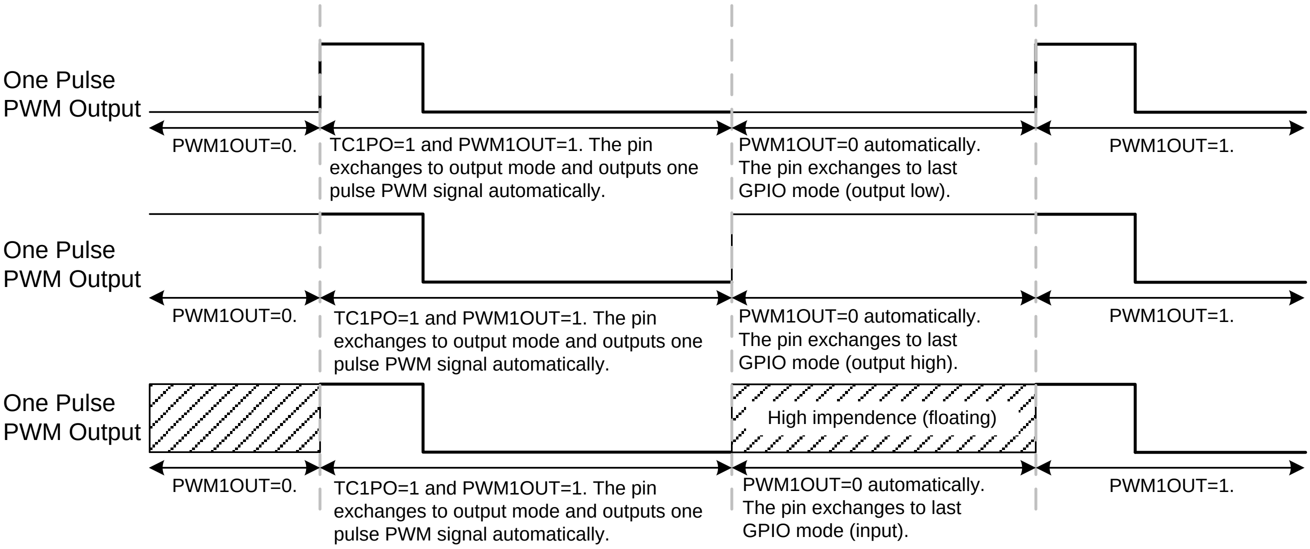
- **If TC1ENB = PWM1OUT = 1 and PWCH[3:2] = 0, PWM doesn't output and P4.6, P4.5 still GPIO mode.**
- **PWM includes 2-channel selected through PWCH[3:2] bits. If the related bits of PWCH[3:2] are enabled, the related pin outputs PWM signal. If the bit is disabled, the pin returns to last GPIO mode.**

8.5.8 2-Channel PWM

The GPIO mode of 2-channel PWM output pins can be the idle status of PWM signal. PWM high idle status is GPIO output high mode. PWM low idle status is GPIO output low mode. PWM high impendence idle status is GPIO input mode. Select a right "PWM" idle status is very important for loading control as PWM disable. The PWM signal is generated from internal PWM processor and outputs to external pin (P4.6, P4.5) through PWCH[3:2] bits channel selections. **If the related bits of PWCH[3:2] are enabled, the related pin outputs PWM signal. If the bit is disabled, the pin returns to last GPIO mode.** The PWM signal of internal source and external pins are the same. The channel selections only switch PWM channels and not process the phase of PWM signal.

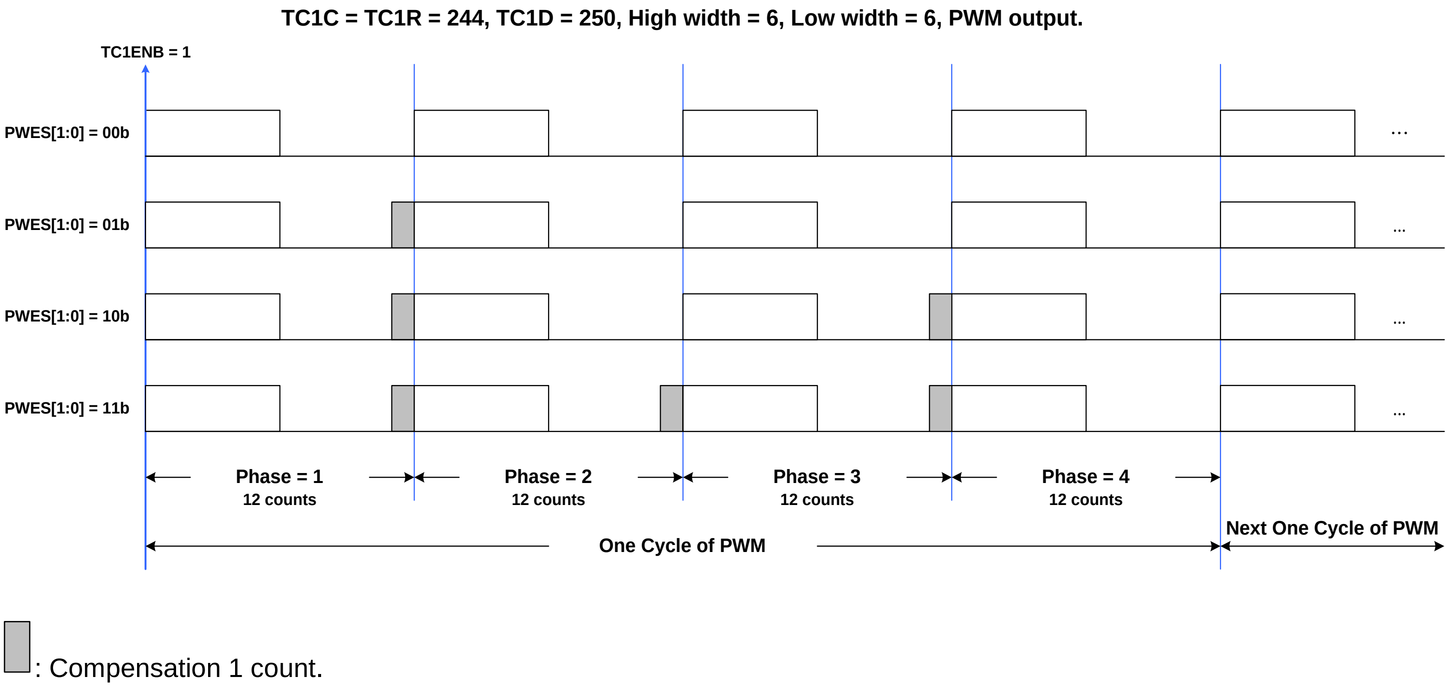
8.5.9 One Pulse PWM

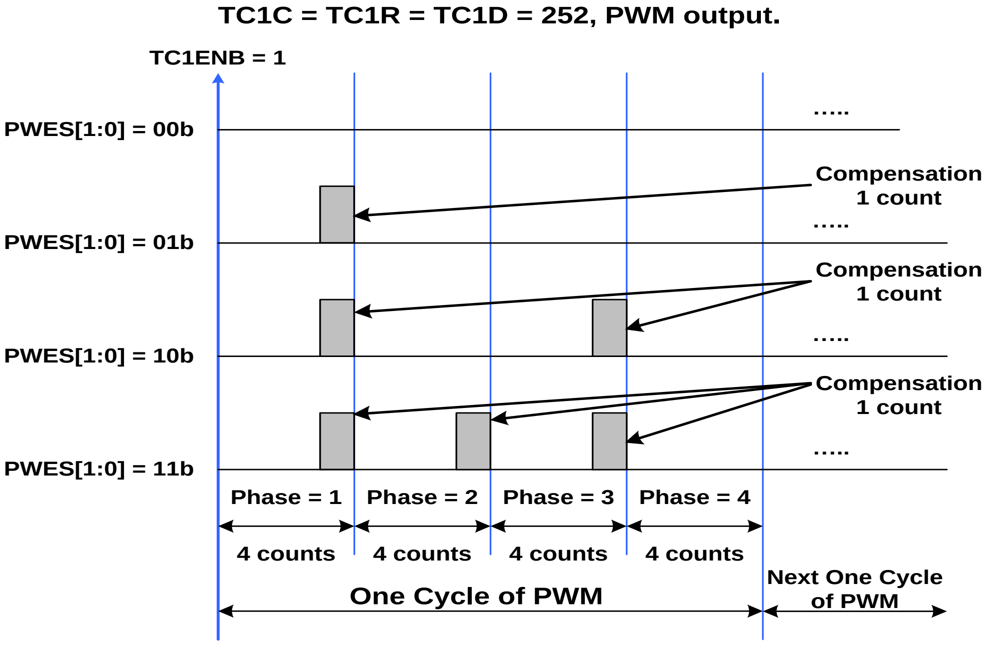
When TC1PO = 0, TC1 is normal timer mode or PWM function mode. When TC1PO = 1 and PWM1OUT=1, TC1 will output one pulse PWM function. When one pulse PWM output signal, the PWM channel exchanges from GPIO to PWM output. When one pulse PWM output finishes, PWM1OUT bit is cleared automatically, the PWM channel returns to GPIO mode and last status. TC1IRQ is issued as TC1 counter overflow. To output next pulse is to set PWM1OUT bit by program again. One pulse PWM channels selected by PWCH[3:2] bits. PWM10, PWM11 output pin is P4.6, P4.5. The PWM10, PWM11 output pins are shared with GPIO pin controlled by PWCH[3:2] bits.



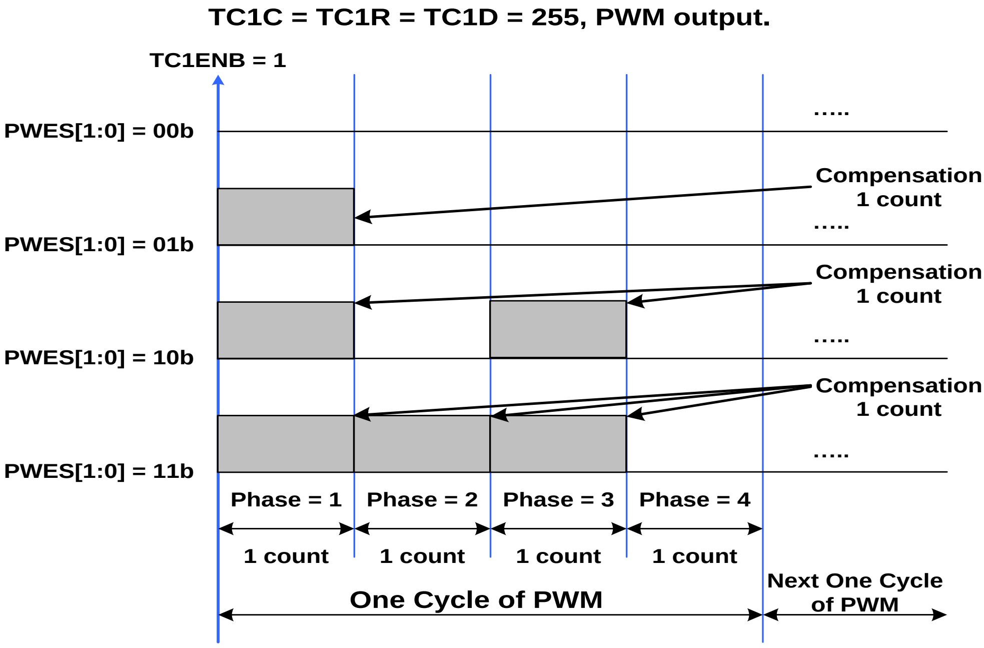
8.5.10 PWM output with Extension function

The PWM extension function supports TC1 PWM function only. One pulse PWM function doesn't support PWM extension function. The PWM extension function controlled by PWES[3:2] bits. If PWES[3:2] = 00, PWM extension function is none compensation. The PWM extension function compensates high level status period at low width of duty cycle. **The PWM with extension function is four phases design. One cycle of PWM with extension function is combined from four sub-cycle signals.** The duty of PWM with extension function is placed in each of sub-cycle. The duty output sequence of the four phases is a special design and through PWM phase processor to allot the duty to each phase. The PWM output with extension function designs auto-reload function. If modify the PWES[3:2]by program as PWM outputting, the new PWM with extension function occurs at next cycle and not change immediately. The methods can avoid the transitional duty occurrence. The PWM output with extension function compensate phase1~phase3 at TC1C from 254 to 255 count instantaneous. The compensate phase time is one count (TC1C is from 255 to overflow: 1 count period time). If user modify PWES[3:2] bits, the new PWM will be change after current 4-phase PWM cycle finished**.**





: Compensation 1 count.



: Compensation 1 count.

8.5.11 TC1 TIMER OPERATION EXAMPLE

- **TC1 TIMER CONFIGURATION:**

```
; Reset TC1 timer.
        CLR       TC1M             ; Clear TC1M register.

; Set TC1 clock source and TC1 rate.
        MOV       A, #0nnn0000b
        B0MOV     TC1M, A

; Set TC1C and TC1R register for TC1 Interval time.
        MOV       A, #value        ; TC1C must be equal to TC1R.
        B0MOV     TC1C, A
        B0MOV     TC1R, A

; Clear TC1IRQ
        B0BCLR    FTC1IRQ

; Enable TC1 timer and interrupt function.
        B0BSET    FTC1IEN          ; Enable TC1 interrupt function.
        B0BSET    FTC1ENB          ; Enable TC1 timer.
```

- **TC1 PWM CONFIGURATION:**

```
; Reset TC1 timer.
        CLR       TC1M             ; Clear TC1M register.

; Set TC1 clock source and TC1 rate.
        MOV       A, #0nnn0000b
        B0MOV     TC1M, A

; Set TC1C and TC1R register for PWM cycle.
        MOV       A, #value1       ; TC1C must be equal to TC1R.
        B0MOV     TC1C, A
        B0MOV     TC1R, A

; Set TC1D register for PWM duty.
        MOV       A, #value2       ; TC1D must be greater than TC1R.
        B0MOV     TC1D, A

; Set PWM channel.
        MOV       A, #0000nn00b
        B0MOV     PWCH, A

; Enable PWM and TC1 timer.
        B0BSET    FTC1ENB          ; Enable TC1 timer.
        B0BSET    FPWM1OUT         ; Enable PWM.
```

- **TC1 One Pulse PWM CONFIGURATION:**

```
; Reset TC1 timer.
        CLR         TC1M            ; Clear TC1M register.

; Set TC1 clock source and TC1 rate.
        MOV         A, #0nnn0000b
        B0MOV       TC1M, A

; Set TC1C and TC1R register for PWM cycle.
        MOV         A, #value1      ; TC1C must be equal to TC1R.
        B0MOV       TC1C, A
        B0MOV       TC1R, A

; Set TC1D register for PWM duty.
        MOV         A, #value2      ; TC1D must be greater than TC1R.
        B0MOV       TC1D, A

; Set PWM channel.
        MOV         A, #0000nn00b
        B0MOV       PWCH, A

; Enable PWM and TC1 timer.
        B0BSET      FTC1PO          ; Enable One Pulse.
        B0BSET      FTC1ENB         ; Enable TC1 timer.
        B0BSET      FPWM1OUT        ; Enable PWM.
```

- **TC1 PWM WITH EXTENSION CONFIGURATION:**

```
; Reset TC1 timer.
        CLR         TC1M            ; Clear TC1M register.

; Set TC1 clock source and TC1 rate.
        MOV         A, #0nnn0000b
        B0MOV       TC1M, A

; Set TC1C and TC1R register for PWM cycle.
        MOV         A, #value1      ; TC1C must be equal to TC1R.
        B0MOV       TC1C, A
        B0MOV       TC1R, A

; Set TC1D register for PWM duty.
        MOV         A, #value2      ; TC1D must be greater than TC1R.
        B0MOV       TC1D, A

; Set PWM channel and Extension.
        MOV         A, #0000nn00b
        B0MOV       PWCH, A
        MOV         A, #000000nnb
        B0MOV       PWES, A

; Enable PWM and TC1 timer.
        B0BSET      FTC1ENB         ; Enable TC1 timer.
        B0BSET      FPWM1OUT        ; Enable PWM.
```

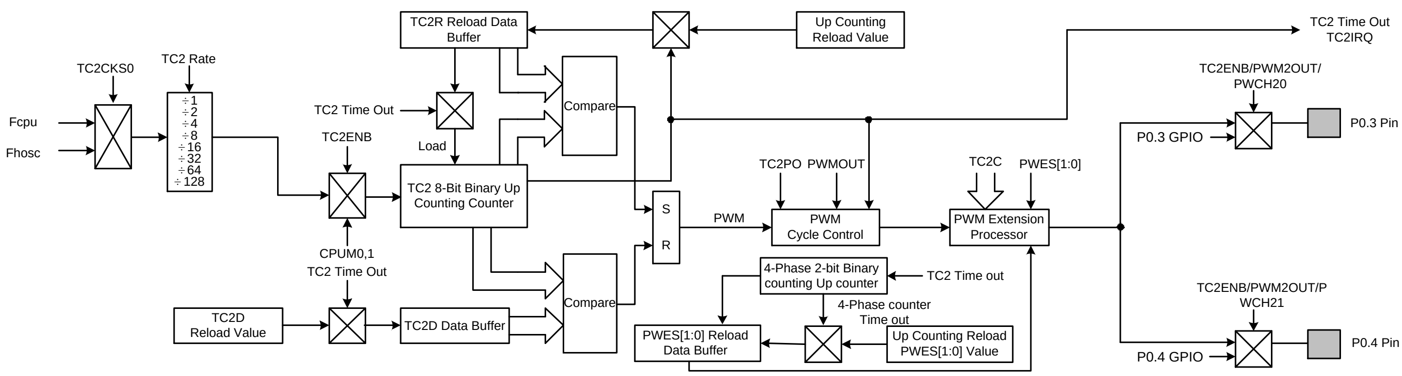
8.6 TC2 8-BIT TIMER/COUNTER

8.6.1 OVERVIEW

The TC2 timer is an 8-bit binary up timer with basic timer, PWM, One Pulse PWM and PWM with extension functions. The basic timer function supports flag indicator (TC2IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TC2M, TC2C, TC2R registers. The event counter is changing TC2 clock source from system clock (Fhosc/Fcpu) to external clock like signal (e.g. continuous pulse, R/C type oscillating signal…). TC2 becomes a counter to count external clock number to implement measure application. TC2 also builds in duty/cycle programmable PWM. The PWM cycle and resolution are controlled by TC2 timer clock rate, TC2R and TC2D registers. So the PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster…. TC2 PWM function also supports one pulse output signal that means only output one cycle PWM signal, not continuous. The PWM has two programmable channels shared with GPIO pins and controlled by PWCH[5:4] bit. The output operation must be through enabled each bit/channel of PWCH[5:4] bits. The enabled PWM channel exchanges from GPIO to PWM output. When the PWCH[5:4] bits disables, the PWM channel returns to GPIO mode and last status. And support the 2-channel PWM output with extension selected by PWES[3:2] bits. TC2 counter supports auto-reload function which always enabled. When TC2 timer overflow occurs, the TC2C will be reloaded from TC2R automatically. The auto-reload function is always enabled. The TC2 doesn't build in green mode wake-up function.

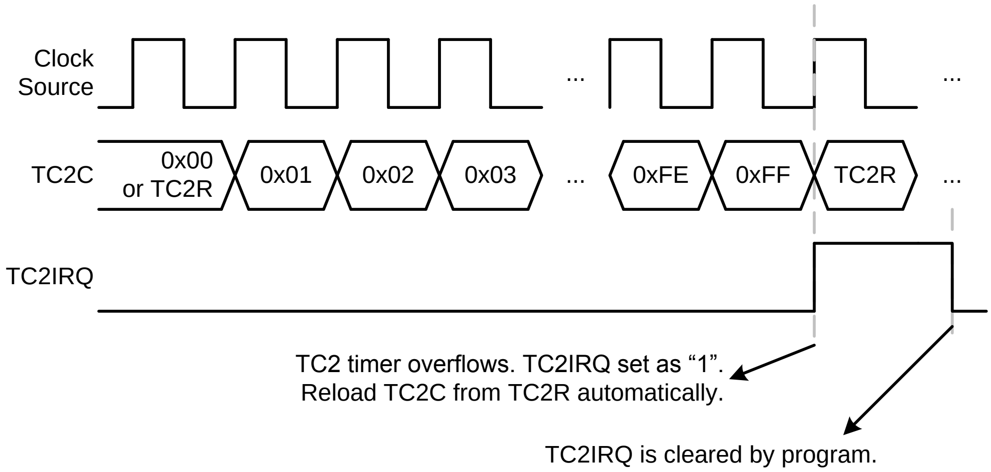
The main purposes of the TC2 timer are as following.

- **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** TC2 timer function supports interrupt function. When TC2 timer occurs overflow, the TC2IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by TC2R and TC2D registers.
- **One Pulse PWM:** The one pulse PWM is controlled by TC2PO bit. When TC2PO=0, TC2 is normal timer mode or PWM function mode. When TC2PO=1, TC2 is one pulse PWM function. When PWMOUT=1, one pulse PWM outputs and the TC2IRQ is issued as TC2 counter overflow, PWMOUT bit is cleared automatically, the PWM channel returns to GPIO mode and last status. **One Pulse PWM doesn't support extension function (PWES[3:2]).**
- **2-Channel PWM output:** The 2-channel PWM output are controlled by PWCH[5:4] bits.
- **PWM output with Extension function: The PWM with extension function is four phases design.** The PWM output with extension function are controlled by PWES[3:2] bits.
- **Green mode function:** All TC2 functions (timer, PWM, event counter, auto-reload, extension) keeps running in green mode, but no wake-up function. Timer IRQ actives as any IRQ trigger occurrence, e.g. timer overflow…



8.6.2 TC2 TIMER OPERATION

TC2 timer is controlled by TC2ENB bit. When TC2ENB=0, TC2 timer stops. When TC2ENB=1, TC2 timer starts to count. Before enabling TC2 timer, setup TC2 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…TC2C increases "1" by timer clock source. When TC2 overflow event occurs, TC2IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is TC2C count from full scale (0xFF) to zero scale (0x00). In difference function modes, TC2C value relates to operation. If TC2C value changing effects operation, the transition of operations would make timer function error. So TC2 builds in double buffer to avoid these situations happen. The double buffer concept is to flash TC2C during TC2 counting, to set the new value to TC2R (reload buffer), and the new value will be loaded from TC2R to TC2C after TC2 overflow occurrence automatically. In the next cycle, the TC2 timer runs under new conditions, and no any transitions occur. The auto-reload function is no any control interface and always actives as TC2 enables. If TC2 timer interrupt function is enabled (TC2IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 0013H) and executes interrupt service routine after TC2 overflow occurrence. Clear TC2IRQ by program is necessary in interrupt procedure. TC2 timer can works in normal mode, slow mode and green mode. Under green mode, TC2 keep counting, set TC2IRQ and outputs PWM, can't wake-up system.



TC2 provides different clock sources to implement different applications and configurations. TC2 clock source includes Fcpu (instruction cycle), Fhosc (high speed oscillator) controlled by TC2CKS0 bit. TC2CKS0 bit selects the clock source is from Fcpu or Fhosc. If TC2CKS0=0, TC2 clock source is Fcpu through TC2rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. If TC2CKS0=1, TC2 clock source is Fhosc through TC2rate[2:0] pre-scalar to decide Fhosc/1~Fhosc/128. TC2 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

TC2CKS0	TC2rate[2:0]	TC2 Clock	TC2 Interval Time			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	8.192	32	32.768	128
0	001b	Fcpu/64	4.096	16	16.384	64
0	010b	Fcpu/32	2.048	8	8.192	32
0	011b	Fcpu/16	1.024	4	4.096	16
0	100b	Fcpu/8	0.512	2	2.048	8
0	101b	Fcpu/4	0.256	1	1.024	4
0	110b	Fcpu/2	0.128	0.5	0.512	2
0	111b	Fcpu/1	0.064	0.25	0.256	1
1	000b	Fhosc/128	2.048	8	8.192	32
1	001b	Fhosc/64	1.024	4	4.096	16
1	010b	Fhosc/32	0.512	2	2.048	8
1	011b	Fhosc/16	0.256	1	1.024	4
1	100b	Fhosc/8	0.128	0.5	0.512	2
1	101b	Fhosc/4	0.064	0.25	0.256	1
1	110b	Fhosc/2	0.032	0.125	0.128	0.5
1	111b	Fhosc/1	0.016	0.0625	0.064	0.25

8.6.3 TC2M MODE REGISTER

TC2M is TC2 timer mode control register to configure TC2 operating mode including TC2 pre-scalar, clock source, PWM function…These configurations must be setup completely before enabling TC2 timer.

0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2M	TC2ENB	TC2rate2	TC2rate1	TC2rate0	-	-	TC2PO	PWM2OUT
Read/Write	R/W	R/W	R/W	R/W	-	-	R/W	R/W
After reset	0	0	0	0	-	-	0	0

Bit 0 **PWM2OUT:** PWM output control bit.
0 = Disable PWM output function, and P0.3, P0.4 (controlled by PWCH[5:4] bits) are GPIO mode.
1 = Enable PWM output function, and P0.3, P0.4 (controlled by PWCH[5:4] bits) output PWM signal.

Bit 1 **TC2PO:** TC2 pulse output function control bit.
0 = Disable TC2 pulse output function.
1 = Enable TC2 pulse output function.

Bit [6:4] **TC2RATE[2:0]:** TC2 timer clock source select bits.
TC2CKS0=0 -> 000 = Fcpu/128, 001 = Fcpu/64, 010 = Fcpu/32, 011 = Fcpu/16, 100 = Fcpu/8, 101 = Fcpu/4, 110 = Fcpu/2,111 = Fcpu/1.
TC2CKS0=1 -> 000 = Fhosc/128, 001 = Fhosc/64, 010 = Fhosc/32, 011 = Fhosc/16, 100 = Fhosc/8, 101 = Fhosc/4, 110 = Fhosc/2,111 = Fhosc/1.

Bit 7 **TC2ENB:** TC2 timer control bit.
0 = Disable TC2 timer.
1 = Enable TC2 timer.

8.6.4 TC2C COUNTING REGISTER

TC2C is TC2 8-bit counter. When TC2C overflow occurs, the TC2IRQ flag is set as "1" and cleared by program. The TC2C decides TC2 interval time through below equation to calculate a correct value. It is necessary to write the correct value to TC2C register and TC2R register first time, and then enable TC2 timer to make sure the fist cycle correct. After one TC2 overflow occurs, the TC2C register is loaded a correct value from TC2R register automatically, not program.

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2C	TC2C7	TC2C6	TC2C5	TC2C4	TC2C3	TC2C2	TC2C1	TC2C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of TC2C initial value is as following.

TC2C initial value = 256 - (TC2 interrupt interval time * TC2 clock rate)

8.6.5 TC2R AUTO-RELOAD REGISTER

TC2 timer builds in auto-reload function, and TC2R register stores reload data. When TC2C overflow occurs, TC2C register is loaded data from TC2R register automatically. Under TC2 timer counting status, to modify TC2 interval time is to modify TC2R register, not TC2C register. New TC2C data of TC2 interval time will be updated after TC2 timer overflow occurrence, TC2R loads new value to TC2C register. But at the first time to setup TC2M, TC2C and TC2R must be set the same value before enabling TC2 timer. TC2 is double buffer design. If new TC2R value is set by program, the new value is stored in 1st buffer. Until TC2 overflow occurs, the new value moves to real TC2R buffer. This way can avoid any transitional condition to affect the correctness of TC2 interval time and PWM output signal.

0C2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2R	TC2R7	TC2R6	TC2R5	TC2R4	TC2R3	TC2R2	TC2R1	TC2R0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of TC2R initial value is as following.

TC2R initial value = 256 - (TC2 interrupt interval time * TC2 clock rate)

☞ **Example: To calculation TC2C and TC2R value to obtain 100us TC2 interval time. TC2 clock source is Fhosc = 16MHz. Select TC2RATE = 011 (Fosc/16).**
TC2 interval time = 100us. TC2 clock rate = 16MHz/16

*TC2C/TC2R initial value = 256 - (TC2 interval time * input clock)*
*= 256 - (100us * 16MHz / 16)*
*= 256 - (100 * 10^{-6} * 16 * 10^{6} / 16)*
= 9CH

8.6.6 TC2D PWM DUTY REGISTER

TC2D register's purpose is to decide PWM duty. In PWM mode, TC2R controls PWM's cycle, and TC2D controls the duty of PWM. The operation is base on timer counter value. When TC2C = TC2D, the PWM high duty finished and exchange to low level. It is easy to configure TC2D to choose the right PWM's duty for application.

0C3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2D	TC2D7	TC2D6	TC2D5	TC2D4	TC2D3	TC2D2	TC2D1	TC2D0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

The equation of TC2D initial value is as following.

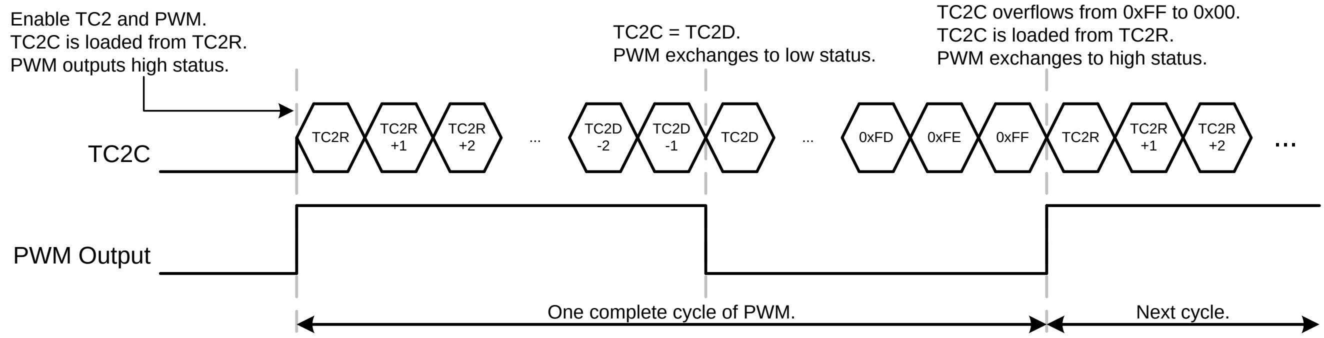
TC2D initial value = TC2R + (PWM high pulse width period / TC2 clock rate)

☞ **Example: To calculate TC2D value to obtain 1/3 duty PWM signal. The TC2 clock source is Fhosc = 16MHz. Select TC2RATE=000 (Fosc/128).**
TC2R = 9CH. TC2 interval time = 800us. So the PWM cycle is 1.25KHz. In 1/3 duty condition, the high pulse width is about 267us.

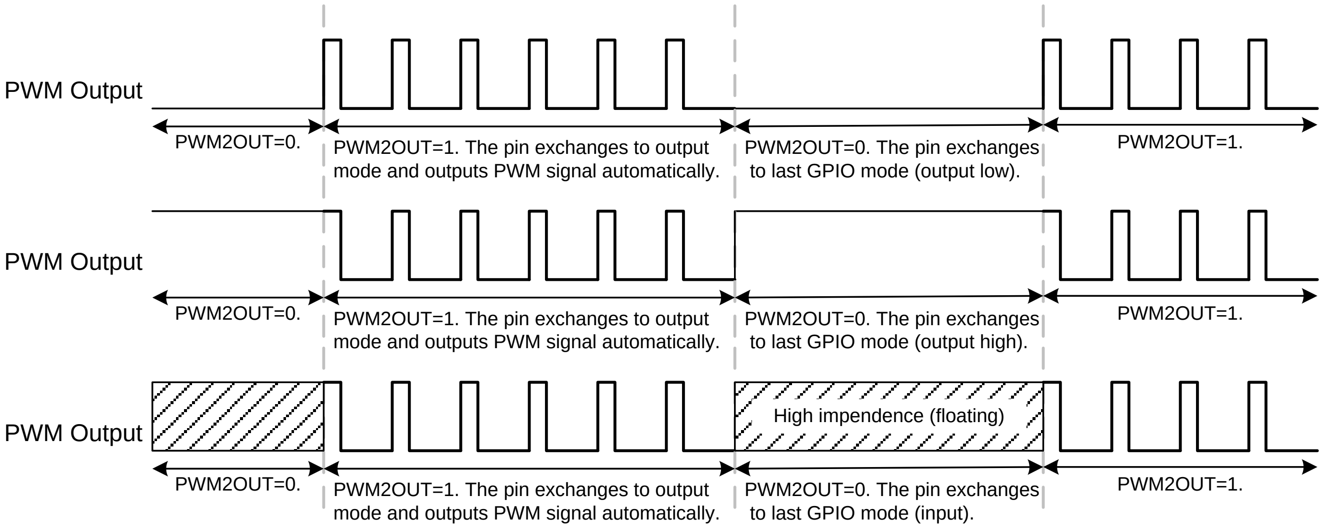
TC2D initial value = 9CH + (PWM high pulse width period / TC2 clock rate)
*= 9CH + (267us * 16MHz / 128)*
= 9CH + 21H
= BDH

8.6.7 PULSE WIDTH MODULATION (PWM)

The PWM is duty/cycle programmable design to offer various PWM signals. When TC2 timer enables before, PWM2OUT bit sets as “1” (enable PWM output) and select PWM output pin (PWCH[5:4]), the PWM output pin (P0.3, P0.4) outputs PWM signal. When TC2ENB = 0 or PWM2OUT = 0, the PWM channel returns to GPIO mode and last status. One cycle of PWM signal is high pulse first, and then low pulse outputs. TC2R register controls the cycle of PWM, and TC2D decides the duty (high pulse width length) of PWM. TC2C initial value is TC2R reloaded when TC2 timer enables and TC2 timer overflows. When TC2C count is equal to TC2D, the PWM high pulse finishes and exchanges to low level. When TC2 overflows (TC2C counts from 0xFF to 0x00), one complete PWM cycle finishes. The PWM exchanges to high level for next cycle. The PWM is auto-reload design to load TC2C from TC2R automatically when TC2 overflows and the end of PWM’s cycle, to keeps PWM continuity. If modify the PWM duty/cycle by program as PWM outputting, the new cycle occurs at next cycle when TC2C loaded from TC2R. The PWM channels selected by PWCH[5:4] bits. PWM20, PWM21 output pin is P0.3, P0.4. The PWM20, PWM21 output pins are shared with GPIO pin controlled by PWCH[5:4] bits.



The resolution of PWM is decided by TC2R. TC2R range is from 0x00~0xFF. If TC2R = 0x00, PWM’s resolution is 1/256. If TC2R = 0x80, PWM’s resolution is 1/128. TC2D controls the high pulse width of PWM for PWM’s duty. When TC2C = TC2D, PWM output exchanges to low status. TC2D must be greater than TC2R, or the PWM signal keeps low status. When PWM outputs, TC2IRQ still actives as TC2 overflows, and TC2 interrupt function actives as TC2IEN = 1. But strongly recommend be careful to use PWM and TC2 timer together, and make sure both functions work well.
The PWM output pin is shared with GPIO and switch to output PWM signal as PWM2OUT=1 automatically. If PWM2OUT bit is cleared to disable PWM, the output pin exchanges to last GPIO mode automatically. It easily to implement carry signal on/off operation, not to control TC2ENB bit.



C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWCH	-	-	PWCH20	PWCH21	PWCH11	PWCH10	PWCH01	PWCH00
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	0	0	0	0	0	0

Bit 4 **PWCH20**: PWM20 control bit.
0 = P0.3 pin GPIO mode.
1 = PWM20 output.

Bit 5 **PWCH21**: PWM21 control bit.
0 = P0.4 pin GPIO mode.
1 = PWM21 output.

C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWES	-	-	-	-	PW2ES1	PW2ES0	PW1ES1	PW1ES0
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After Reset	-	-	-	-	0	0	0	0

Bit [3:2] **PW2ES[1:0]:** The PWM2 output with extension control bits **(One pulse PWM function doesn't support)**.
00: None compensation.
01: Compensation 1 count at phase 1 of TC2 PWM output.
10: Compensation 1 count at phase 1/3 of TC2 PWM output.
11: Compensation 1 count at phase 1/2/3 of TC2 PWM output.

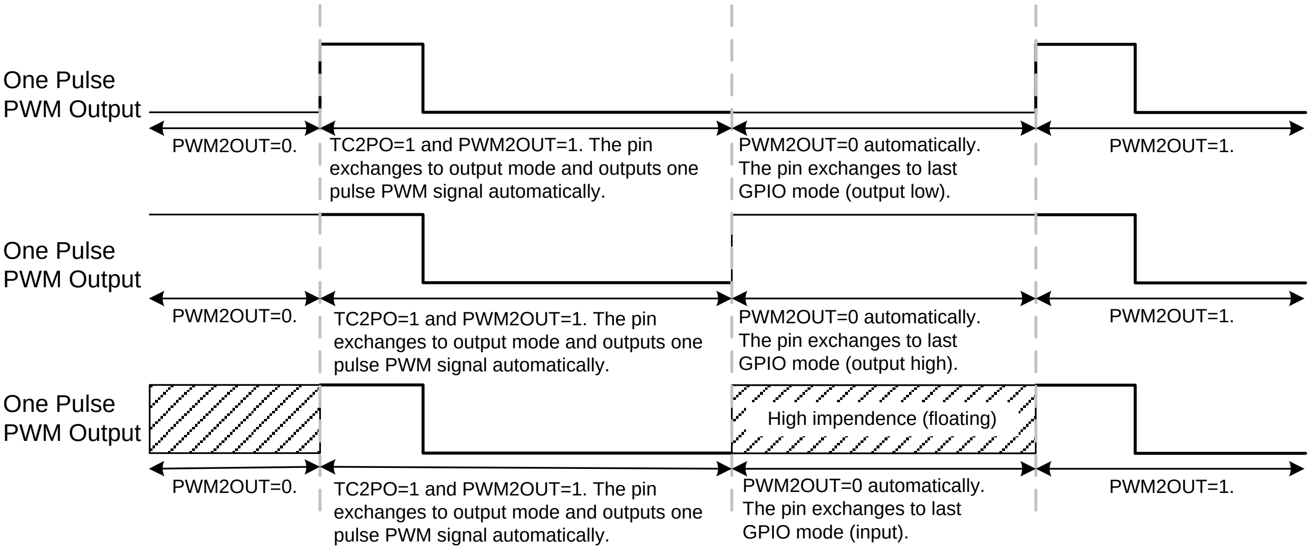
- **If TC2ENB = PWM2OUT = 1 and PWCH[5:4] = 0, PWM doesn't output and P0.3, P0.4 still GPIO mode.**
- **PWM includes 2-channel selected through PWCH [5:4] bits. If the related bits of PWCH[5:4] are enabled, the related pin outputs PWM signal. If the bit is disabled, the pin returns to last GPIO mode.**

8.6.8 2-Channel PWM

The GPIO mode of 2-channel PWM output pins can be the idle status of PWM signal. PWM high idle status is GPIO output high mode. PWM low idle status is GPIO output low mode. PWM high impendence idle status is GPIO input mode. Select a right "PWM" idle status is very important for loading control as PWM disable. The PWM signal is generated from internal PWM processor and outputs to external pin (P0.3, P0.4) through PWCH[5:4] bits channel selections. **If the related bits of PWCH[5:4] are enabled, the related pin outputs PWM signal. If the bit is disabled, the pin returns to last GPIO mode.** The PWM signal of internal source and external pins are the same. The channel selections only switch PWM channels and not process the phase of PWM signal.

8.6.9 One Pulse PWM

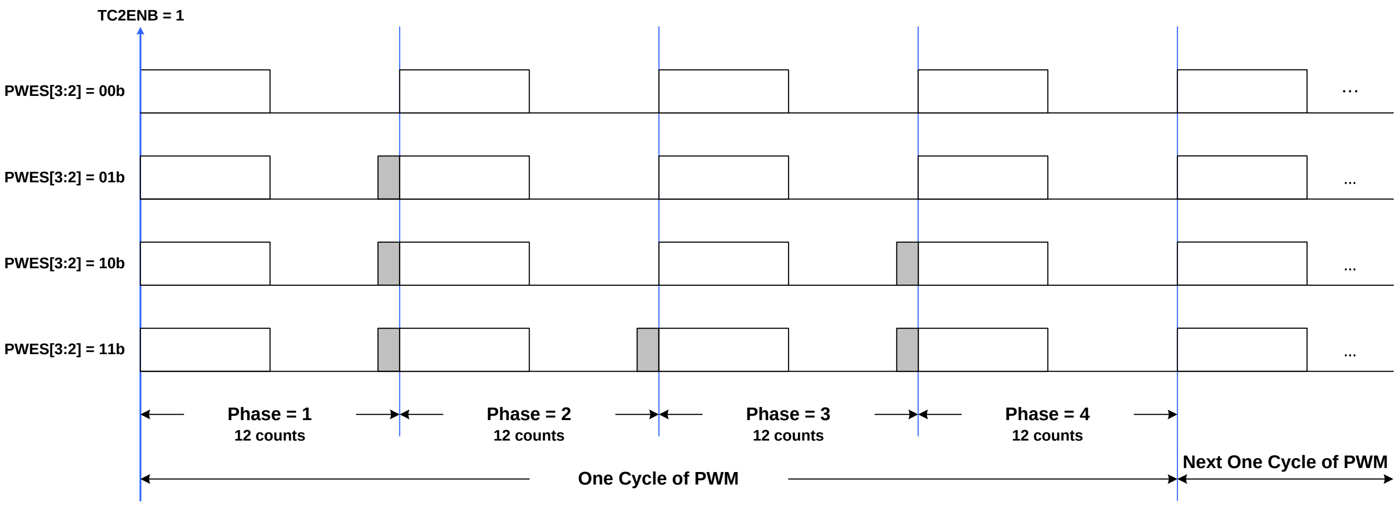
When TC2PO = 0, TC2 is normal timer mode or PWM function mode. When TC2PO = 1 and PWM2OUT=1, TC2 will output one pulse PWM function. When one pulse PWM output signal, the PWM channel exchanges from GPIO to PWM output. When one pulse PWM output finishes, PWM2OUT bit is cleared automatically, the PWM channel returns to GPIO mode and last status. TC2IRQ is issued as TC2 counter overflow. To output next pulse is to set PWM2OUT bit by program again. One pulse PWM channels selected by PWCH[5:4] bits. PWM20, PWM21 output pin is P0.3, P0.4. The PWM20, PWM21 output pins are shared with GPIO pin controlled by PWCH[5:4] bits.



8.6.10 PWM output with Extension function

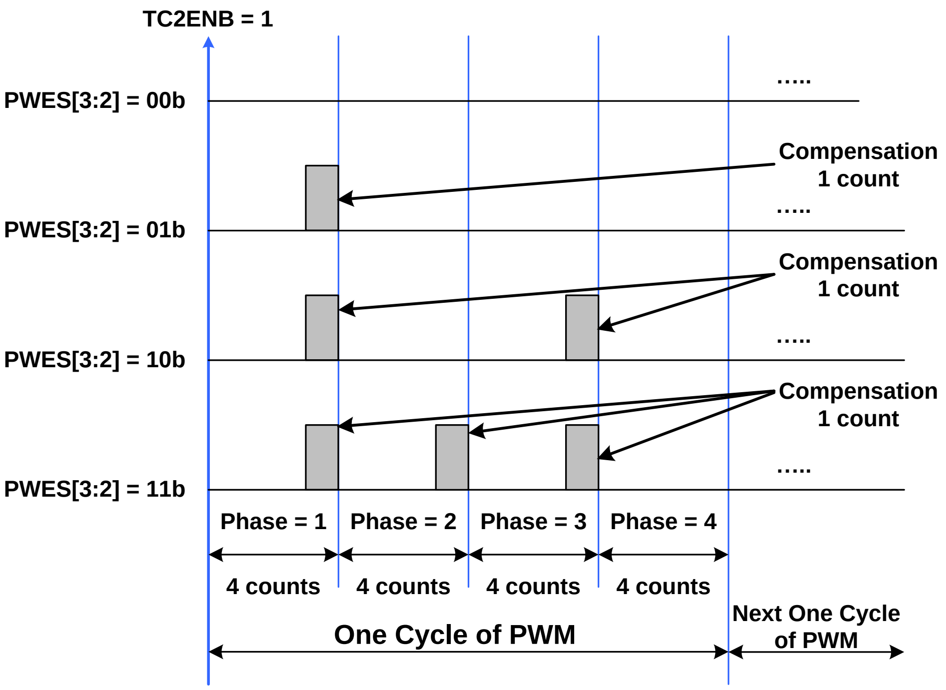
The PWM extension function supports TC2 PWM function only. One pulse PWM function doesn't support PWM extension function. The PWM extension function controlled by PWES[3:2] bits. If PWES[3:2] = 00, PWM extension function is none compensation. The PWM extension function compensates high level status period at low width of duty cycle. **The PWM with extension function is four phases design. One cycle of PWM with extension function is combined from four sub-cycle signals.** The duty of PWM with extension function is placed in each of sub-cycle. The duty output sequence of the four phases is a special design and through PWM phase processor to allot the duty to each phase. The PWM output with extension function designs auto-reload function. If modify the PWES[3:2]by program as PWM outputting, the new PWM with extension function occurs at next cycle and not change immediately. The methods can avoid the transitional duty occurrence. The PWM output with extension function compensate phase1~phase3 at TC2C from 254 to 255 count instantaneous. The compensate phase time is one count (TC2C is from 255 to overflow: 1 count period time). If user modify PWES[3:2] bits, the new PWM will be change after current 4-phase PWM cycle finished**.**

TC2C = TC2R = 244, TC2D = 250, High width = 6, Low width = 6, PWM output.



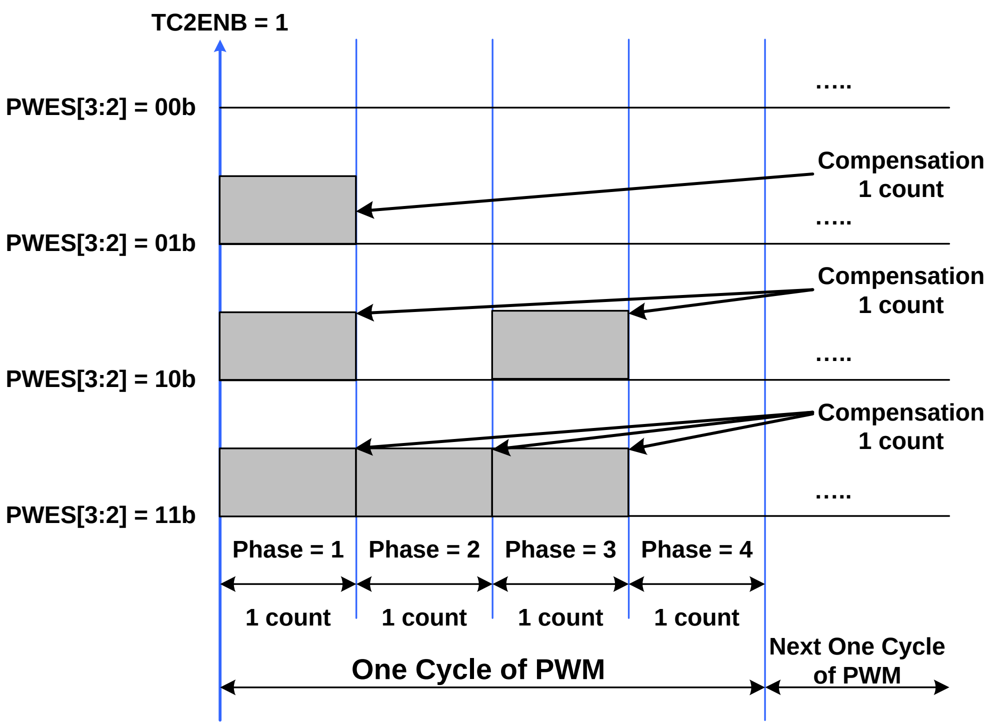
: Compensation 1 count.

TC2C = TC2R = TC2D = 252, PWM output.



: Compensation 1 count.

TC2C = TC2R = TC2D = 255, PWM output.



: Compensation 1 count.

8.6.11 TC2 TIMER OPERATION EXAMPLE

- **TC2 TIMER CONFIGURATION:**

```
; Reset TC2 timer.
        CLR         TC2M            ; Clear TC2M register.

; Set TC2 clock source and TC2 rate.
        MOV         A, #0nnn0000b
        B0MOV       TC2M, A

; Set TC2C and TC2R register for TC2 Interval time.
        MOV         A, #value       ; TC2C must be equal to TC2R.
        B0MOV       TC2C, A
        B0MOV       TC2R, A

; Clear TC2IRQ
        B0BCLR      FTC2IRQ

; Enable TC2 timer and interrupt function.
        B0BSET      FTC2IEN         ; Enable TC2 interrupt function.
        B0BSET      FTC2ENB         ; Enable TC2 timer.
```

- **TC2 PWM CONFIGURATION:**

```
; Reset TC2 timer.
        CLR         TC2M            ; Clear TC2M register.

; Set TC2 clock source and TC2 rate.
        MOV         A, #0nnn0000b
        B0MOV       TC2M, A

; Set TC2C and TC2R register for PWM cycle.
        MOV         A, #value1      ; TC2C must be equal to TC2R.
        B0MOV       TC2C, A
        B0MOV       TC2R, A

; Set TC2D register for PWM duty.
        MOV         A, #value2      ; TC2D must be greater than TC2R.
        B0MOV       TC2D, A

; Set PWM channel.
        MOV         A, #00nn0000b
        B0MOV       PWCH, A

; Enable PWM and TC2 timer.
        B0BSET      FTC2ENB         ; Enable TC2 timer.
        B0BSET      FPWM2OUT        ; Enable PWM.
```

- **TC2 One Pulse PWM CONFIGURATION:**

```
; Reset TC2 timer.
		CLR		TC2M			; Clear TC2M register.

; Set TC2 clock source and TC2 rate.
		MOV		A, #0nnn0000b
		B0MOV		TC2M, A

; Set TC2C and TC2R register for PWM cycle.
		MOV		A, #value1		; TC2C must be equal to TC2R.
		B0MOV		TC2C, A
		B0MOV		TC2R, A

; Set TC2D register for PWM duty.
		MOV		A, #value2		; TC2D must be greater than TC2R.
		B0MOV		TC2D, A

; Set PWM channel.
		MOV		A, #00nn0000b
		B0MOV		PWCH, A

; Enable PWM and TC2 timer.
		B0BSET		FTC2PO			; Enable One Pulse.
		B0BSET		FTC2ENB			; Enable TC2 timer.
		B0BSET		FPWM2OUT		; Enable PWM.
```

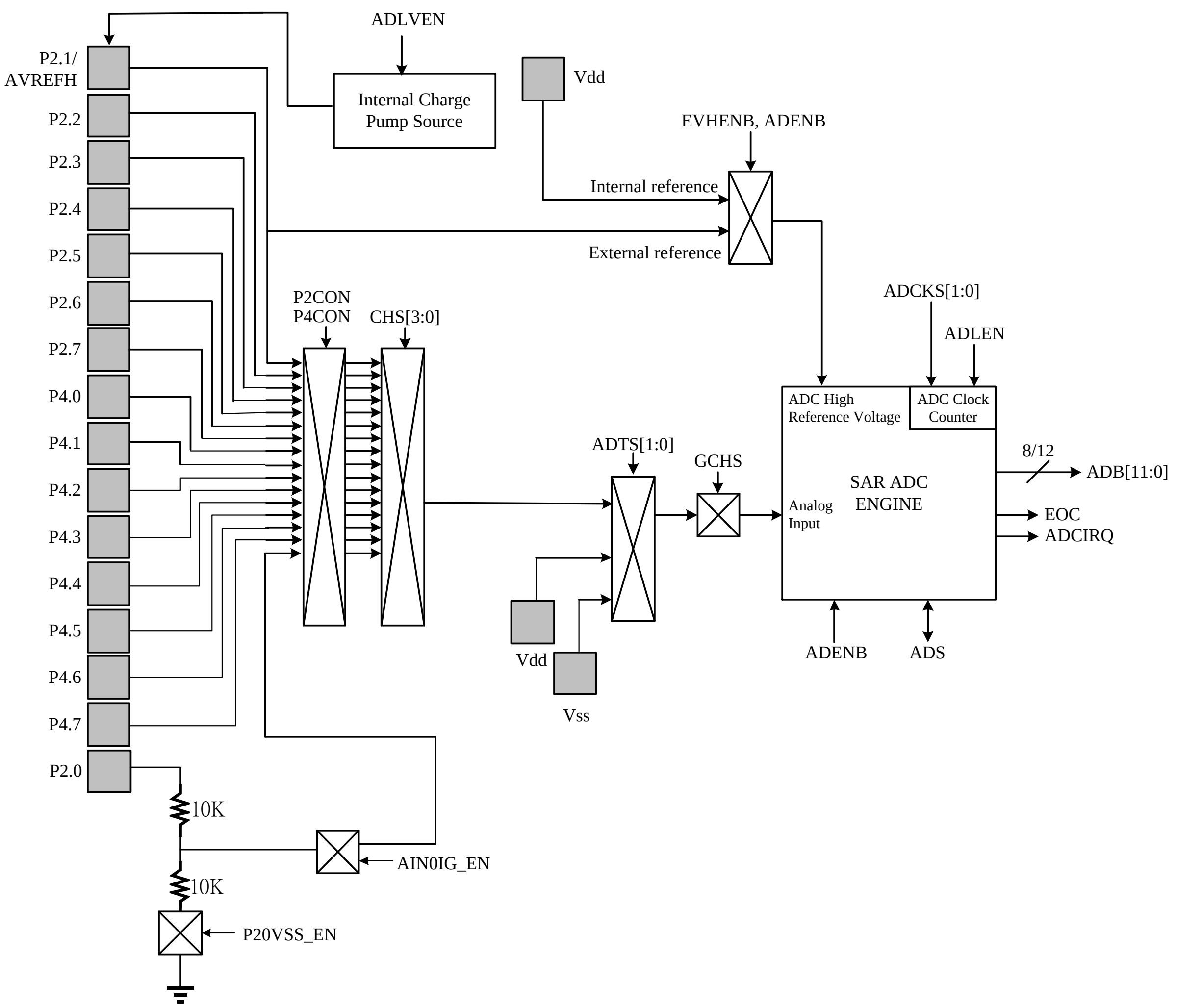
- **TC2 PWM WITH EXTENSION CONFIGURATION:**

```
; Reset TC2 timer.
		CLR		TC2M			; Clear TC2M register.

; Set TC2 clock source and TC2 rate.
		MOV		A, #0nnn0000b
		B0MOV		TC2M, A

; Set TC2C and TC2R register for PWM cycle.
		MOV		A, #value1		; TC2C must be equal to TC2R.
		B0MOV		TC2C, A
		B0MOV		TC2R, A

; Set TC2D register for PWM duty.
		MOV		A, #value2		; TC2D must be greater than TC2R.
		B0MOV		TC2D, A

; Set PWM channel and Extension.
		MOV		A, #00nn0000b
		B0MOV		PWCH, A
		MOV		A, #0000nn00b
		B0MOV		PWES, A

; Enable PWM and TC2 timer.
		B0BSET		FTC2ENB			; Enable TC2 timer.
		B0BSET		FPWM2OUT		; Enable PWM.
```

9 16 CHANNEL ANALOG TO DIGITAL CONVERTER (ADC)

9.1 OVERVIEW

The analog to digital converter (ADC) is SAR structure with 16-input sources and up to 4096-step resolution to transfer analog signal into 12-bits digital buffers. The ADC builds in 16-channel input source (AIN0~AIN15) to measure 16 different analog signal sources controlled by CHS[3:0] and GCHS bits. The ADC resolution can be selected 8-bit and 12-bit resolutions through ADLEN bit. The ADC converting rate can be selected by ADCKS[1:0] bits to decide ADC converting time. The ADC reference voltage input pin from P2.1 pin. The ADC builds in P2CON/P4CON registers to set pure analog input pin. It is necessary to set ADC input pin as input mode without pull-up resistor by program. After setup ADENB and ADS bits, the ADC starts to convert analog signal to digital data. When the conversion is complete, the ADC circuit will set EOC and ADCIRQ bits to “1” and the digital data outputs in ADB and ADR registers. If the ADCIEN = 1, the ADC interrupt request occurs and executes interrupt service routine when ADCIRQ = 1 after ADC converting. If ADC interrupt function is enabled (ADCIEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 15H) and executes interrupt service routine after ADC converting. Clear ADCIRQ by program is necessary in interrupt procedure. ADC can work in green mode. After ADC operating, the system would be waked up from green mode to last mode (normal mode/slow mode).



9.2 ADC MODE REGISTER

ADM is ADC mode control register to configure ADC configurations including ADC start, ADC channel selection, ADC high reference voltage source and ADC processing indicator…These configurations must be setup completely before starting ADC converting.

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **ADENB:** ADC control bit. **In power saving mode, disable ADC to reduce power consumption.**
0 = Disable ADC function.
1 = Enable ADC function.

Bit 6 **ADS:** ADC start control bit. **ADS bit is cleared after ADC processing automatically.**
0 = ADC converting stops.
1 = Start to execute ADC converting.

Bit 5 **EOC:** ADC status bit.
0 = ADC progressing.
1 = End of converting and reset ADS bit.

Bit 4 **GCHS:** ADC global channel select bit.
0 = Disable AIN channel.
1 = Enable AIN channel.

Bit [3:0] **CHS[3:0]:** ADC input channel select bit.
0000 = AIN0, 0001 = AIN1, 0010 = AIN2, 0011 = AIN3, 0100 = AIN4, 0101 = AIN5, 0110 = AIN6,
0111 = AIN7, 1000 = AIN8, 1001 = AIN9, 1010 = AIN10, 1011 = AIN11, 1100 = AIN12, 1101 = AIN13,
1110 = AIN14, 1111 = AIN15.

ADR register includes ADC mode control and ADC low-nibble data buffer. ADC configurations including ADC clock rate and ADC resolution. These configurations must be setup completely before starting ADC converting.

0CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	AIN0IG_EN	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
After reset	1	0	0	0	-	-	-	-

Bit 7 **AIN0IG_EN:** AIN0's high voltage gating function control bit.
0 = Disable high voltage gating.
1 = Enable high voltage gating.

AIN0IG_EN	P2CON0	P20VSS_EN	Description
0	0	0	I/O input mode. (Input voltage must be equal or under VDDIO voltage)
0	0	1	I/O input mode with 20k pull-down. (Input voltage must be equal or under VDDIO)
0	1	0	ADC mode. Input range: 0v~AVDD
0	1	1	ADC mode with high voltage detection. Input range: 0v~Vbat. Vbat cannot exceed 2*AVDD, and the maximum absolute voltage cannot exceed 8v.

1	0	0	I/O input mode.
1	0	1	I/O input mode with 20k pull-down.
1	1	0	ADC mode. (No ADC function is available)
1	1	1	ADC mode. (No ADC function is available).

*Under battery direct detection mode (high voltage detection), user should follow the sequence to measure the battery voltage:

1. Enable P20VSS_EN.
2. Enable P2CON0.
3. Disable AIN0IG_EN.

*Under battery direct detection mode (high voltage detection), user should follow the sequence to disable the measurement for power saving.

1. Enable AIN0IG_EN.
2. Disable P20VSS_EN.
3. Disable P2CON0.

*Under ADC mode (no high voltage detection), user should follow the sequence to measure the external input voltage.

1. Disable AIN0IG_EN.
2. Disable P20VSS_EN.
3. Enable P2CON0.

Bit 6,4 **ADCKS [1:0]:** ADC's clock rate select bit.
00 = Fhosc/16, 01 = Fhosc/8, 10 = Fhosc/1, 11 = Fhosc/2

Bit 5 **ADLEN:** ADC's resolution select bits.
0 = 8-bit.
1 = 12-bit.

9.3 ADC DATA BUFFER REGISTERS

ADC data buffer is 12-bit length to store ADC converter result. The high byte is ADB register, and the low-nibble is ADR[3:0] bits. The ADB register is only 8-bit register including bit 4~bit11 ADC data. To combine ADB register and the low-nibble of ADR will get full 12-bit ADC data buffer. The ADC data buffer is a read-only register and the initial status is unknown after system reset.

- **ADB[11:4]: In 8-bit ADC mode, the ADC data is stored in ADB register.**
- **ADB[11:0]: In 12-bit ADC mode, the ADC data is stored in ADB and ADR registers.**

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
Read/Write	R	R	R	R	R	R	R	R
After reset	-	-	-	-	-	-	-	-

Bit[7:0] **ADB[7:0]:** 8-bit ADC data buffer and the high-byte data buffer of 12-bit ADC.

0CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	AIN0IG_EN	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
After reset	0	0	0	0	-	-	-	-

Bit [3:0] **ADB [3:0]:** 12-bit low-nibble ADC data buffer.

The AIN input voltage v.s. ADB output data

AIN n	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
0/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	0
1/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	1
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
4094/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	0
4095/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	1

For different applications, users maybe need more than 8-bit resolution but less than 12-bit. To process the ADB and ADR data can make the job well. First, the ADC resolution must be set 12-bit mode and then to execute ADC converter routine. Then delete the LSB of ADC data and get the new resolution result. The table is as following.

ADC Resolution	ADB								ADR			
	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
8-bit	O	O	O	O	O	O	O	O	x	x	x	x
9-bit	O	O	O	O	O	O	O	O	O	x	x	x
10-bit	O	O	O	O	O	O	O	O	O	O	x	x
11-bit	O	O	O	O	O	O	O	O	O	O	O	x
12-bit	O	O	O	O	O	O	O	O	O	O	O	O
O = Selected. X = Useless.												

✸ ***Note: The initial status of ADC data buffer including ADB register and ADR low-nibble after the system reset is unknown.***

9.3.1 ADC CONVERTING TIME

The ADC converting time is from ADS=1 (Start to ADC convert) to EOC=1 (End of ADC convert). The converting time duration is depend on ADC resolution and ADC clock rate. 12-bit ADC's converting time is 1/(ADC clock /4)*16 sec, and the 8-bit ADC converting time is 1/(ADC clock /4)*12 sec. ADC clock source is Fhosc and includes Fhosc/1, Fhosc/2, Fhosc/8 and Fhosc/16 controlled by ADCKS[1:0] bits.
The ADC converting time affects ADC performance. If input high rate analog signal, it is necessary to select a high ADC converting rate. If the ADC converting time is slower than analog signal variation rate, the ADC result would be error. So to select a correct ADC clock rate and ADC resolution to decide a right ADC converting rate is very important.

12-bit ADC conversion time = 1/(ADC clock rate/4)*16 sec

ADLEN	ADCKS1, ADCKS0	ADC Clock Rate	Fhosc=16MHz	
			ADC Converting time	ADC Converting Rate
1 (12-bit)	00	Fhosc/16	1/(16MHz/16/4)*16 = 64 us	15.625KHz
	01	Fhosc/8	1/(16MHz/8/4)*16 = 32 us	31.25KHz
	10	Fhosc	1/(16MHz/4)*16 = 4 us	250KHz
	11	Fhosc/2	1/(16MHz/2/4)*16 = 8 us	125KHz

8-bit ADC conversion time = 1/(ADC clock rate/4)*12 sec

ADLEN	ADCKS1, ADCKS0	ADC Clock Rate	Fhosc=16MHz	
			ADC Converting time	ADC Converting Rate
0 (8-bit)	00	Fhosc/16	1/(16MHz/16/4)*12 = 48 us	20.833KHz
	01	Fhosc/8	1/(16MHz/8/4)*12 = 24 us	41.667KHz
	10	Fhosc	1/(16MHz/4)*12 = 3 us	333.333KHz
	11	Fhosc/2	1/(16MHz/2/4)*12 = 6 us	166.667KHz

9.3.2 ADC PIN CONFIGURATION

ADC input channels are shared wit Port2 and Port4. ADC channel selection is through CHS[3:0] bit. CHS[3:0] value points to the ADC input channel directly. CHS[3:0]=0000 selects AIN0. CHS[3:0]=0001 selects AIN1…… Only one pin of port4, port2 can be configured as ADC input in the same time. The pins of Port2 and Port4 configured as ADC input channel must be set input mode, disable internal pull-up and enable P4CON/P2CON first by program. After selecting ADC input channel through CHS[3:0], set GCHS bit as "1" to enable ADC channel function.

- **The GPIO mode of ADC input channels must be set as input mode.**
- **The internal pull-up resistor of ADC input channels must be disabled.**
- **P2CON and P4CON bits of ADC input channel must be set.**

The P2.1/AIN1 can be ADC external high reference voltage input pin when EVHENB=1. In the condition, P2.1 GPIO mode must be set as input mode and disable internal pull-up resistor.

- **The GPIO mode of ADC external high reference voltage input pin must be set as input mode.**
- **The internal pull-up resistor of ADC external high reference voltage input pin must be disabled.**

ADC input pins are shared with digital I/O pins. Connect an analog signal to COMS digital input pin, especially, the analog signal level is about 1/2 VDD will cause extra current leakage. In the power down mode, the above leakage current will be a big problem. Unfortunately, if users connect more than one analog input signal to port4, port2 will encounter above current leakage situation. P2CON is Port2 configuration register. Write "1" into P2CON [7:0] will configure relate port2 pin as pure analog input pin to avoid current leakage. P4CON is Port4 configuration register. Write "1" into P4CON [7:0] will configure related port 4 pin as pure analog input pin to avoid current leakage.

0C7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Bit[7:0] **P4CON[7:0]:** P4.n configuration control bits.
0 = P4.n can be an analog input (ADC input) or digital I/O pins.
1 = P4.n is pure analog input, can't be a digital I/O pin.

0C6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2CON	P2CON7	P2CON6	P2CON5	P2CON4	P2CON3	P2CON2	P2CON1	P2CON0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Bit[7:0] **P2CON[7:0]:** P2.n configuration control bits.
0 = P2.n can be an analog input (ADC input) or digital I/O pins.
1 = P2.n is pure analog input, can't be a digital I/O pin.

> ✷ ***Note: When ADC pin is general I/O mode, the bit of P1CON and P4CON must be set to "0", or the digital I/O signal would be isolated.***

9.4 ADC REFERENCE VOLTQAGE REGISTERS

ADC reference voltage source controlled through ADT register. There are external voltage source and internal voltage source. When EVHENB bit is “1”, ADC reference voltage is external voltage source from P2.1. If EVHENB bit is “0”, ADC reference voltage is from internal VDD source.

0CBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADT	-	-	EVHENB	-	-	-	-	-
Read/Write	-	-	R/W	-	-	-	-	-
After reset	-	-	0	-	-	-	-	-

Bit 5 **EVHENB:** ADC status bit.
0 = Enable ADC internal VREFH function. P2.1/AIN1/AVREFH pin is P2.1/AIN1 pin
1 = Disable ADC internal VREFH function. P2.1/AIN1/AVREFH pin is external AVREFH input pin.

9.5 ADC OPERATION EXAMPLE

- **ADC CONFIGURATION:**

```
; Reset ADC.
            CLR         ADM             ; Clear ADM register.

; Set ADC clock rate and ADC resolution.
            MOV         A, #0nmn0000b   ; nn: ADCKS[1:0] for ADC clock rate.
            B0MOV       ADR, A          ; m: ADLEN for ADC reolution.

; Set ADC high reference voltage source.
            B0BSET      FEVHENB         ; External reference voltage.

; Set ADC input channel configuration.
            MOV         A, #value1      ; Set P4CON for ADC input channel.
            B0MOV       P4CON, A
            MOV         A, #value2      ; Set ADC input channel as input mode.
            B0MOV       P4M, A
            MOV         A, #value3      ; Disable ADC input channel's internal pull-up resistor.
            B0MOV       P4UR, A

            MOV         A, #value4      ; Set P2CON for ADC input channel.
            B0MOV       P2CON, A
            MOV         A, #value5      ; Set ADC input channel as input mode.
            B0MOV       P2M, A
            MOV         A, #value6      ; Disable ADC input channel's internal pull-up resistor.
            B0MOV       P2UR, A

; Enable ADC.
            B0BSET      FADENB

; Execute ADC 100us warm-up time delay loop.
            CALL        100usDLY        ; 100us delay loop.

; Select ADC input channel.
            MOV         A, #value       ; Set CHS[3:0] for ADC input channel selection.
            OR          ADM, A

; Enable ADC input channel.
            B0BSET      FGCHS

; Enable ADC interrupt function.
            B0BCLR      FADCIRQ         ; Clear ADC interrupt flag.
            B0BSET      FADCIEN         ; Enable ADC interrupt function.

; Start to execute ADC converting.
            B0BSET      FADS
```

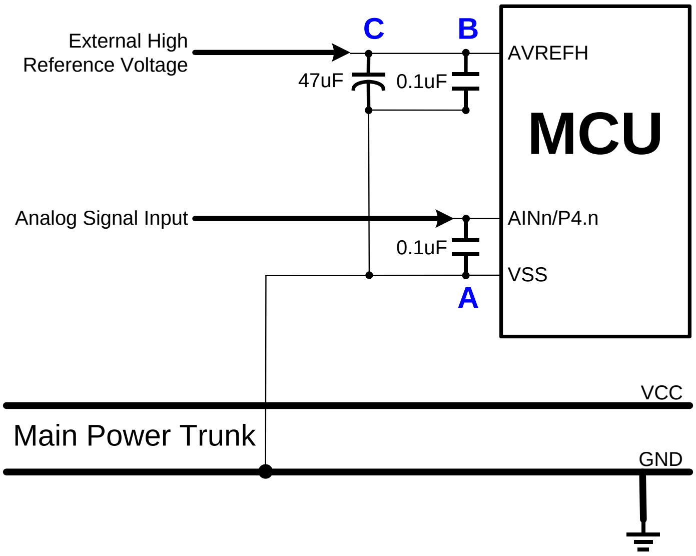
> ✸ ***Note:***
> 1. ***When ADENB is enabled, the system must be delay 100us to be the ADC warm-up time by program, and then set ADS to do ADC converting. The 100us delay time is necessary after ADENB setting (not ADS setting), or the ADC converting result would be error. Normally, the ADENB is set one time when the system under normal run condition, and do the delay time only one time.***
> 2. ***In power saving situation like power down mode and green mode, and not using ADC function, to disable ADC by program is necessary to reduce power consumption.***

- **ADC CONVERTING OPERATION:**

```
; ADC Interrupt disable mode.
@@:
            B0BTS1      FEOC            ; Check ADC processing flag.
            JMP         @B              ; EOC=0: ADC is processing.
            B0MOV       A, ADB          ; EOC=1: End of ADC processing. Process ADC result.
            B0MOV       BUF1,A
            MOV         A, #00001111b
            AND         A, ADR
            B0MOV       BUF2,A
            …                           ; End of processing ADC result.
            CLR         FEOC            ; Clear ADC processing flag for next ADC converting.

; ADC Interrupt enable mode.
            ORG 11H                     ; Interrupt vector.
INT_SR:                                 ; Interrupt service routine.

            B0BTS1      FADCIRQ         ; Check ADC interrupt flag.
            JMP         EXIT_INT        ; ADCIRQ=0: Not ADC interrupt request.
            B0MOV       A, ADB          ; ADCIRQ=1: End of ADC processing. Process ADC result.
            B0MOV       BUF1,A
            MOV         A, #00001111b
            AND         A, ADR
            B0MOV       BUF2,A
            …                           ; End of processing ADC result.
            CLR         FEOC            ; Clear ADC processing flag for next ADC converting.
            JMP         INT_EXIT

INT_EXIT:

            RETI                        ; Exit interrupt service routine.
```

✸ ***Note: ADS is cleared when the end of ADC converting automatically. EOC bit indicates ADC processing status immediately and is cleared when ADS = 1. Users needn't to clear it by program.***

9.6 ADC APPLICATION CIRCUIT



The analog signal is inputted to ADC input pin “AINn/P4.n” or “AINn/P2.n”. The ADC input signal must be through a 0.1uF capacitor “A”. The 0.1uF capacitor is set between ADC input pin and VSS pin, and must be on the side of the ADC input pin as possible. Don’t connect the capacitor’s ground pin to ground plain directly, and must be through VSS pin. The capacitor can reduce the power noise effective coupled with the analog signal.

If the ADC high reference voltage is from external voltage source, the external high reference is connected to AVREFH pin (P2.1). The external high reference source must be through a 47uF ”C” capacitor first, and then 0.1uF capacitor “B”. These capacitors are set between AVREFH pin and VSS pin, and must be on the side of the AVREFH pin as possible. Don’t connect the capacitor’s ground pin to ground plain directly, and must be through VSS pin.

10 SERIAL INPUT/OUTPUT TRANSCEIVER (SIO)

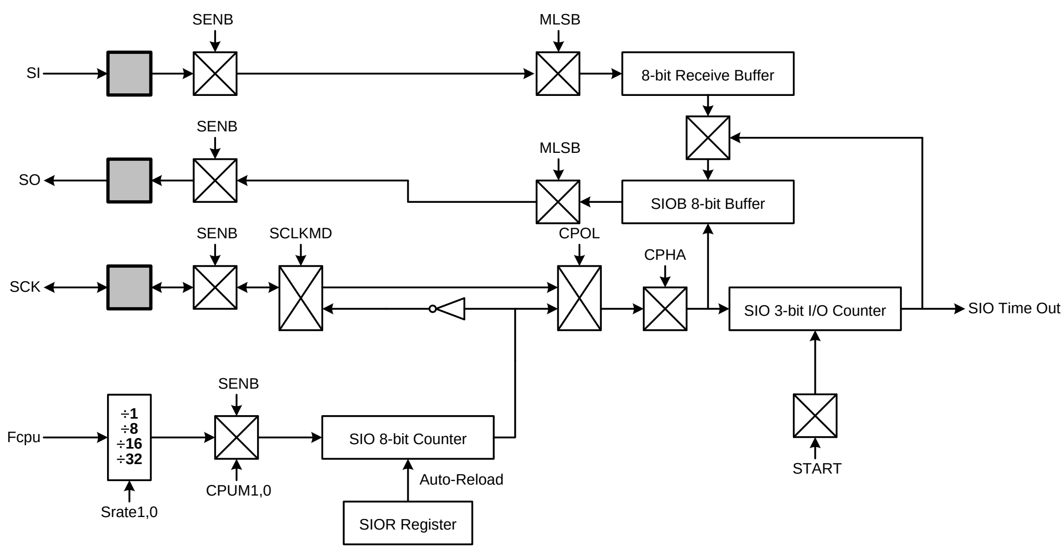
10.1 OVERVIEW

The SIO (serial input/output) transceiver is a serial communicate interface for data exchanging from one MCU to one MCU or other hardware peripherals. It is a simple 8-bit interface without a major definition of protocol, packet or control bits. The SIO transceiver includes three pins, clock (SCK), data input (SI) and data output (SO) to send data between master and slaver terminals. The SIO interface builds in 8-mode which are the clock idle status, the clock phases and data fist bit direction. The 8-bit mode supports most of SIO/SPI communicate format.
The SIO features include the following:

- **Full-duplex, 3-wire synchronous data transfer.**
- **Master (SCK is clock output) or Slave (SCK is clock input) operation.**
- **MSB/LSB first data transfer.**
- **The start phase of data sampling location selection is 1st-phase or 2nd-phase controlled register.**
- **Two programmable bit rates (Only in master mode).**
- **End-of-Transfer interrupt.**

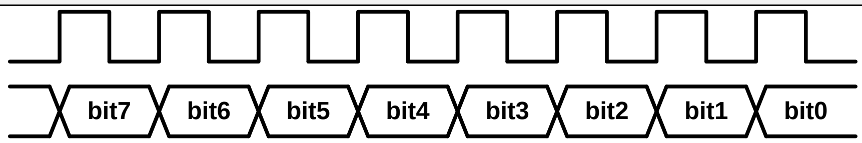
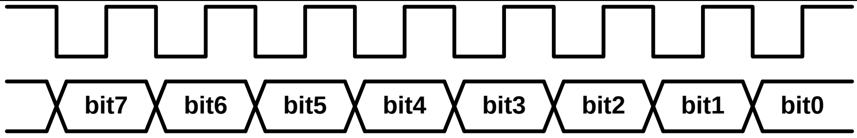
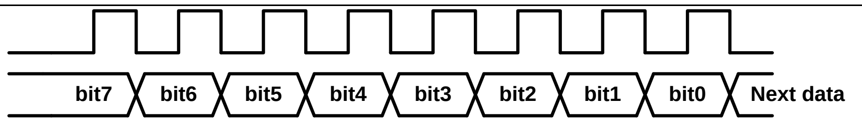
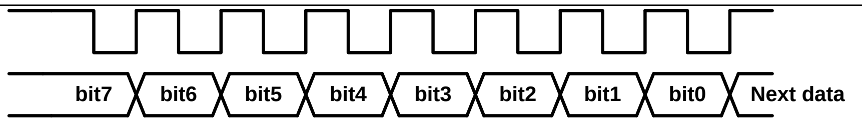
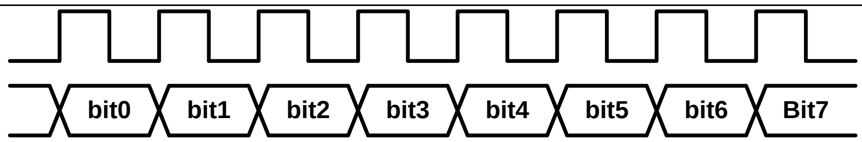
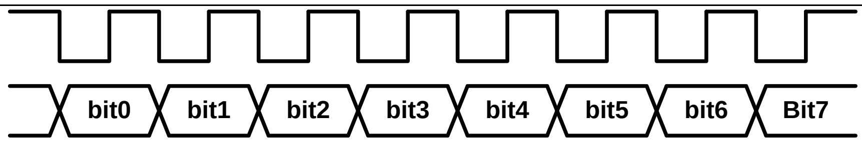
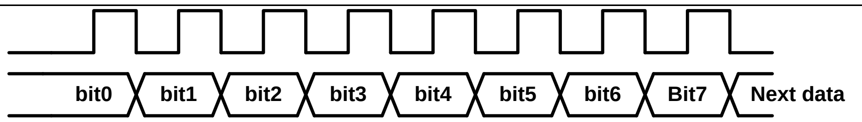
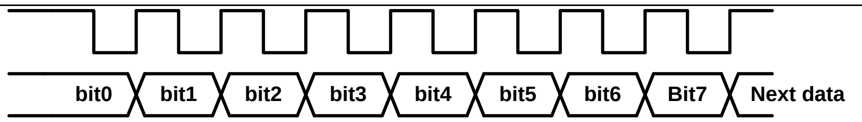
10.2 SIO OPERATION

The SIOM register can control SIO operating function, such as: transmit/receive, clock rate, data transfer direction, SIO clock idle status and clock control phase and starting this circuit. This SIO circuit will transmit or receive 8-bit data automatically by setting SENB and START bits in SIOM register. The SIO data buffer is double buffer design. When the SIO operating, the SIOB register stores transfer data and one internal buffer stores receive data. When SIO operation is successfully, the internal buffer reloads into SIOB register automatically. The SIO 8-bit counter and SIOR register are designed to generate SIO's clock source with auto-reload function. The 3-bit I/O counter can monitor the operation of SIO and announce an interrupt request after transmitting/ receiving 8-bit data. After transferring 8-bit data, this circuit will be disabled automatically and re-transfer data by programming SIOM register. CPOL bit is designed to control SIO clock idle status. CPHA bit is designed to control the clock edge direction of data receive. CPOL and CPHA bits decide the SIO format. The SIO data transfer direction is controlled by MLSB bit to decide MSB first or LSB first.



SIO Interface Structure Diagram

The SIO supports 8-mode format controlled by MLSB, CPOL and CPHA bits. The edge direction is “Data Transfer Edge”. When setting rising edge that means to receive and transmit one bit data at SCK rising edge, and data transition is at SCK falling edge. When setting falling edge, that means to receive and transmit one bit data at SCK falling edge, and data transition is at SCK rising edge.
“CPHA” is the clock phase bit controls the phase of the clock on which data is sampled. When CPHA=1, the SCK first edge is for data transition, and receive and transmit data is at SCK 2nd edge. When CPHA=0, the 1st bit is fixed already, and the SCK first edge is to receive and transmit data. The SIO data transfer timing as following figure:

MLSB	CPOL	CPHA	Diagrams	Description
0	0	1		SCK idle status = Low. The transfer first bit = MSB. SCK data transfer edge = Falling edge.
0	1	1		SCK idle status = High. The transfer first bit = MSB. SCK data transfer edge = Rising edge.
0	0	0		SCK idle status = Low. The transfer first bit = MSB. SCK data transfer edge = Rising edge.
0	1	0		SCK idle status = High. The transfer first bit = MSB. SCK data transfer edge = Falling edge.
1	0	1		SCK idle status = Low. The transfer first bit = LSB. SCK data transfer edge = Falling edge.
1	1	1		SCK idle status = High. The transfer first bit = LSB. SCK data transfer edge = Rising edge.
1	0	0		SCK idle status = Low. The transfer first bit = LSB. SCK data transfer edge = Rising edge.
1	1	0		SCK idle status = High. The transfer first bit = LSB. SCK data transfer edge = Falling edge.

SIO Data Transfer Timing

The SIO supports interrupt function. SIOIEN is SIO interrupt function control bit. SIOIEN=0, disable SIO interrupt function. SIOIEN=1, enable SIO interrupt function. When SIO interrupt function enable, the program counter points to interrupt vector (ORG 0015H) to do SIO interrupt service routine after SIO operating. SIOIRQ is SIO interrupt request flag, and also to be the SIO operating status indicator when SIOIEN = 0, but cleared by program. When SIO operation finished, the SIOIRQ would be set to “1”, and the operation is the inverse status of SIO “START” control bit.
The SIOIRQ and SIO START bit indicating the end status of SIO operation is after one 8-bit data transferring. The duration from SIO transfer end to SIOIRQ/START active is about **“1/2*SIO clock”**, means the SIO end indicator doesn’t active immediately.

> ✸ ***Note: The first step of SIO operation is to setup the SIO pins’ mode. Enable SENB, select CPOL and CPHA bits. These bits control SIO pins’ mode.***

10.3 SIOM MODE REGISTER

0B0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOM	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **SENB:** SIO function control bit.
0 = Disable SIO function. SIO pins are GPIO.
1 = Enable SIO function. GPIO pins are SIO pins.

Bit 6 **START:** SIO progress control bit.
0 = End of transfer.
1 = SIO transmitting.

Bit [5:4] **SRATE1,0:** SIO’s transfer rate select bit. **These 2-bits are workless when SCKMD=1.**
00 = fcpu. 01 = fcpu/32. 10 = fcpu/16. 11 = fcpu/8.

Bit 3 **MLSB:** MSB/LSB transfer first.
0 = MSB transmit first.
1 = LSB transmit first.

Bit 2 **SCKMD:** SIO’s clock mode select bit.
0 = Internal. (Master mode)
1 = External. (Slave mode)

Bit 1 **CPOL:** SCK idle status control bit.
0 = SCK idle status is low status.
1 = SCK idle status is high status.

Bit 0 **CPHA:** The Clock Phase bit controls the phase of the clock on which data is sampled.
0 = Data receive at the first clock phase.
1 = Data receive at the second clock phase.

Because SIO function is shared with GPIO. The following table shows the SIO pin mode mode behavior and setting when SIO function enable and disable.

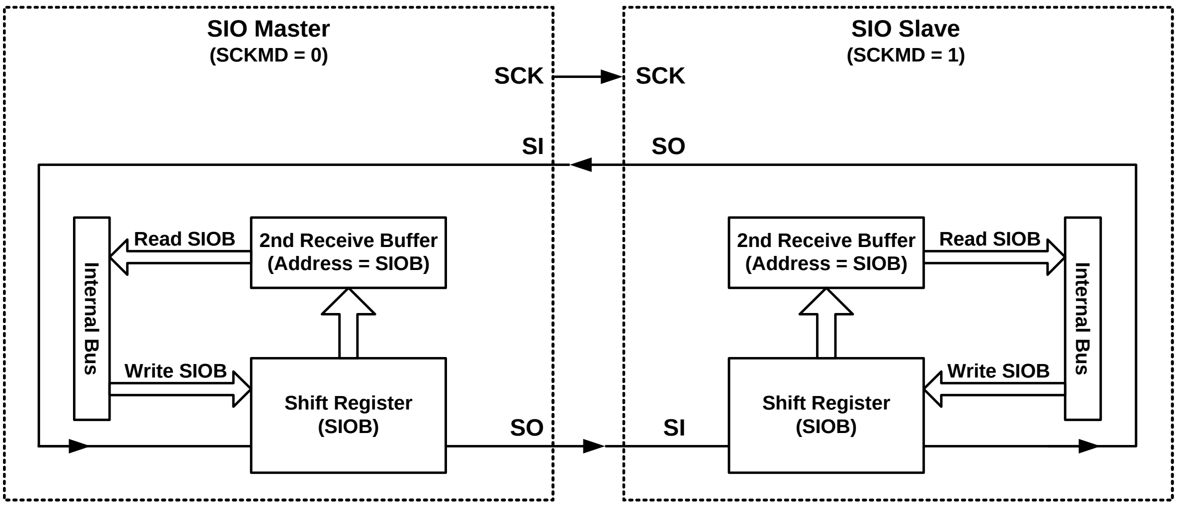
SENB=1 (SIO Function Enable)		
SCK	SCKMD=1 SIO source = External clock	GPIO will change to Input mode automatically, no matter what PnM setting.
	SCKMD=0 SIO source = Internal clock	GPIO will change to Output mode automatically, no matter what PnM setting.
SI	GPIO must be set as Input mode in PnM ,or the SIO function will be abnormal	
SO	SIO = Transmitter/Receiver	GPIO will change to Output mode automatically, no matter what PnM setting.
SENB=0 (SIO Function Disable)		
GPIO	GPIO I/O mode are fully controlled by PnM when SIO function Disable	

> ✸ ***Note:***
> 1. ***If SCKMD=1 for external clock, the SIO is in SLAVE mode. If SCKMD=0 for internal clock, the SIO is in MASTER mode.***
> 2. ***Don't set SENB and START bits in the same time. That makes the SIO function error.***
> 3. ***Except START bit, all other bit registers in SIOM must be set or cleared by b0mov/mov instructions. Only START bit can be enabled or disabled by b0bset/b0bclr/bset/bclr instructions.***

10.4 SIOB DATA BUFFER

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOB	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

SIOB is the SIO data buffer register. It stores serial I/O transmit and receive data. The system is single-buffered in the transmit direction and double-buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SIOB Data Register before the entire shift cycle is completed. When receiving data, however, a received byte must be read from the SIOB Data Register before the next byte has been completely shifted in. Otherwise, the first byte is lost. Following figure shows a typical SIO transfer between two micro-controllers. Master MCU sends SCK for initial the data transfer. Both master and slave MCU must work in the same clock edge direction, and then both controllers would send and receive data at the same time.



SIO Data Transfer Diagram

10.5 SIOR REGISTER DESCRIPTION

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOR	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The SIOR is designed for the SIO counter to reload the counted value when end of counting. It is like a post-scalar of SIO clock source and let SIO has more flexible to setting SCK range. Users can set the SIOR value to setup SIO transfer time. To setup SIOR value equation to desire transfer time is as following.

SCK frequency = (SIO rate / (256 - SIOR))/2

SIOR = 256 - (1 / (2 * SCK frequency) * SIO rate)

- **Example: Setup the SIO clock to be 5KHz. Fhosc = 16MHz, SIO's rate = Fcpu/16, Fcpu = Fhosc/1= 16MHz.**

SIOR = 256 – (1/(2*5KHz) * 16MHz/16)
= 256 – (0.0001*1000000)
= 256 – 100
= 156

> ✸ ***Note:***
> 1. ***If SCKMD=1 for external clock, the SIO is in SLAVE mode. If SCKMD=0 for internal clock, the SIO is in MASTER mode.***
> 2. ***Don't set SENB and START bits in the same time. That makes the SIO function error.***
> 3. ***Except START bit, all other bit registers in SIOM must be set or cleared by b0mov/mov instructions. Only START bit can be enabled or disabled by b0bset/b0bclr/bset/bclr instructions.***

Example: Master, duplex transfer and transmit data on rising edge

```
            MOV         A,TXDATA          ; Load transmitted data into SIOB register.
            B0MOV       SIOB,A
            MOV         A,#0FFH           ; Set SIO clock
            B0MOV       SIOR,A
            MOV         A,#10000000B      ; Setup SIOM and enable SIO function.
            B0MOV       SIOM,A            ; Do not use b0bset FSENB to enable SIO
            B0BSET      FSTART            ; Start transfer and receiving SIO data.
CHK_END:
            B0BTS0      FSTART            ; Wait the end of SIO operation.
            JMP         CHK_END
            B0MOV       A,SIOB            ; Save SIOB data into RXDATA buffer.
            MOV         RXDATA,A
```

Example: Slave, duplex transfer and transmit data on rising edge

```
            MOV         A,TXDATA          ; Load transfer data into SIOB register.
            B0MOV       SIOB,A
            MOV         A,# 10000100B     ; Setup SIOM and enable SIO function.
            B0MOV       SIOM,A            ; Do not use b0bset FSENB to enable SIO
            B0BSET      FSTART            ; Start transfer and receiving SIO data.
CHK_END:
            B0BTS0      FSTART            ; Wait the end of SIO operation.
            JMP         CHK_END
            B0MOV       A,SIOB            ; Save SIOB data into RXDATA buffer.
            MOV         RXDATA,A
```

11 UNIVERSAL SERIAL BUS (USB)

11.1 OVERVIEW

The USB is the answer to connectivity for the PC architecture. A fast, bi-directional interrupt pipe, low-cost, dynamically attachable serial interface is consistent with the requirements of the PC platform of today and tomorrow. The SONIX USB microcontrollers are optimized for human-interface computer peripherals such as a mouse, keyboard, joystick, and game pad.

- **Conforms to USB specifications, Version 2.0.**
- **Supports 1 Full-speed USB device address.**
- **Supports 1 control endpoint and 6 configurable endpoints for isochronous/interrupt/bulk transfer.**
- **Integrated USB transceiver.**
- **5V to 3.3V regulator output for D+ 1.5K ohm internal resistor pull up.**

11.2 USB REGISTERS

11.2.1 USB DEVICE ADDRESS REGISTER

The USB Device Address Register (UDA) contains a 7-bit USB device address and one bit to enable the USB function. This register is cleared during a reset, setting the USB device address to zero and disable the USB function.

CCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UDA	UDE	UDA6	UDA5	UDA4	UDA3	UDA2	UDA1	UDA0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **UDE:** Device Function Enable.
0 = Disable USB device functions.
1 = Enable USB device functions.

Bit [6:0] **UDA[6:0]**: These bits must be set by firmware during the USB enumeration process (i.e., SetAddress) to the non-zero address assigned by the USB host.

11.2.2 USB STATUS REGISTER

The USB status register indicates the status of USB.

CDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USTATUS	ERR_TIMEOUT	ERR_SETUP	EP0_PRESETUP	EP0_OUT_STALL	EP0_IN_STALL	EP0SETUP	EP0IN	EP0OUT
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **ERR_TIMEOUT:** Timeout Status.
0 = No time out. Clear by firmware.
1 = Bus no any response more than 18 bits time.

Bit 6 **ERR_SETUP:** Wrong Setup data received.
0= Normal 8-byte Setup DATA0 is received. Clear by firmware.
1 = Setup data is not 8-byte or is not DATA0.

Bit5 **EP0_PRESETUP:** EP0 Setup Token packet Flag
0 = No EP0 Setup token packet. Clear by firmware.

1 = EP0 Setup token packet is received.

Bit 4 **EP0_OUT_STALL:** EP0 OUT STALL transaction.
0 = No EP0 OUT STALL transaction. Clear by firmware.
1 = EP0 OUT STALL transaction is completed.

Bit 3 **EP0_IN_STALL:** EP0 IN STALL Transaction is completed.
0 = No EP0 IN STALL transaction. Clear by firmware.
1 = EP0 IN STALL transaction is completed.

Bit 2 **EP0SETUP:** EP0 Setup Transaction Flag.
0 = No EP0 Setup transaction. Clear by firmware.
1 = EP0 Setup transaction is completed.

Bit 1 **EP0IN:** EP0 IN ACK Transaction Flag.
0 = No EP0 IN Transaction. Clear by firmware
1 = EP0 IN Transaction is completed.

Bit 0 **EP0OUT:** USB device address enable.
0 = No EP0 OUT ACK transaction. Clear by firmware.
1 = EP0 OUT ACK transaction is completed.

CEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USTATUS1	NDT2	NDT1	-	SOF	BUS_RESUME	BUS_WAKEUP	BUS_RST	SUSPEND
Read/Write	R	R	-	R/W	R/W	R/W	R/W	R/W
After reset	0	0	-	0	0	0	0	0

Bit 7 **NDT2:** Noise detected flag.
0 = No NDT2 noise is detected.
1 = NDT2 noise is detected.

Bit 6 **NDT1:** Noise detected flag.
0 = No NDT1 noise is detected.
1 = NDT1 noise is detected.

Bit 4 **SOF:** USB SOF packet received flag
0 = No USB SOF packet. Clear by firmware.
1 = USB SOF packet is received.

Bit 3 **BUS_RESUME:** USB Bus Resume signal flag.
0 = No bus resume signal is detected. Clear by firmware.
1 = Bus resume signal from suspend mode is detected.

Bit 2 **BUS_WAKEUP:** Bus Wake Up flag.
0 = No wakeup from suspend mode. Clear by firmware.
1 = Wakeup from suspend.

Bit 1 **BUS_RST:** USB Bus Reset signal (>2.5us SE0) flag
0 = No bus reset signal is detected. Clear by firmware.
1 = Bus reset signal is detected.

Bit 0 **SUSPEND:** USB Bus Suspend signal (>3ms idle state) flag
0 = No bus suspend is detected.
1 = Bus suspend is detected.

11.2.3 USB INTERRUPT ENABLE REGISTER

CFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USB_INT_EN	-	-	SOF_IE	EPN_ACK_IE	EPN_NAK_IE	BUS_IE	BUSWK_IE	USB_IE
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit 5 **SOF_IE:** USB SOF Interrupt Enable
0 = Disable USB SOF interrupt.
1 = Enable USB SOF interrupt.

Bit 4 **EPN_ACK_IE:** EPN ACK Interrupt Enable.
0 = Disable EPN ACK interrupt function.
1 = Enable EPN ACK interrupt function.

Bit 3 **EPN_NAK_IE:** EPN NAK Interrupt Enable.
0 = Disable EPN NAK interrupt function.
1 = Enable EPN NAK interrupt function.

Bit 2 **BUS_IE:** Bus Event Interrupt Enable
0 = Disable BUS event interrupt..
1 = Enable Bus event interrupt..

Bit 1 **BUSWK_IE:** Bus Wake Up Interrupt Enable
0 = Disable Wake Up event interrupt.
1 = Enable Wake Up event interrupt.

Bit 0 **USB_IE:** USB Interrupt Enable
0 = Disable USB event interrupt.
1 = Enable USB event interrupt.

11.2.4 USB ENDPOINT'S ACK HANDSHAKING FLAG REGISTER

The status of endpoint's ACK transaction

D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP_ACK	-	-	EP6_ACK	EP5_ACK	EP4_ACK	EP3_ACK	EP2_ACK	EP1_ACK
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit [5:0] **EPn_ACK [5:0], n=1~6:** The bit is set whenever the endpoint that completes with an ACK received.
0 = the endpoint (interrupt pipe) doesn't complete with an ACK.
1 = the endpoint (interrupt pipe) complete with an ACK.

✸ ***Note: If endpoint1~6 in ISO mode, EPn_ACK will be set 1 by H/W once ISO transaction completes.***

11.2.5 USB ENDPOINT'S NAK HANDSHAKING FLAG REGISTER

The status of endpoint's ACK transaction

D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP_NAK	-	-	EP6_NAK	EP5_NAK	EP4_NAK	EP3_NAK	EP2_NAK	EP1_NAK
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit [5:0] **EPn_NAK [5:0], n=1~6:** The bit is set whenever the endpoint that completes with an NAK received.
0 = the endpoint (interrupt pipe) doesn't complete with an NAK.
1 = the endpoint (interrupt pipe) complete with an NAK.

11.2.6 USB CONFIGURATION REGISTER

D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UCFG	ULVD36_EN	VREG33_EN	PHY_EN	DPPU_EN	ESD_EN	USGIG_EN	VREG33DIS_EN	USBRAM_EN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	1	0	0	0	0	0	0

Bit6 **ULVD36_EN:** Enable the USB VDD's LVD36 detect function.
0 = Disable.
1 = Enable.

✸ ***Note: Before enabling ULVD36_EN bit, user must enable UPDET_EN first.***

Bit6 **VREG33_EN:** Enable the internal VREG33 output.
0 = Disable.
1 = Enable.

✸ ***Note: If VREG33 is disabled, the VREG33 will be switched to USB_VDD.***
✸ ***Note: Under PS/2 mode, VREG33 must be disabled.***

Bit5 **PHY_EN:** PHY Transceiver Function Enable
0 = Disable PHY transceiver function.
1 = Enable PHY transceiver function.

✷ ***Note: If PHY_EN = 0, the D+/D- state can still be read from UPID[1:0].***

Bit4 **DPPU_EN:** Enable internal D+ 1.5k pull-up resistor
0 = Disable internal D+ pull-up resistor.
1 = Enable internal D+ pull-up resistor.

Bit3 **ESD_EN:** Enable USB anti-ESD protection.
0 = Disable anti-ESD protection.
1 = Enable anti-ESD protection.

Bit2 **USGIG_EN:** Enable the input gating function for USB related signals.
0 = Disable.
1 = Enable.

Bit1 **VREG33DIS_EN:** Enable the discharge path function for VREG33, D+, and D- pins.
0 = Disable.
1 = Enable.

Bit0 **USBRAM_EN:** Enable USB 256-byte RAM function.
0 = Disable.
1 = Enable.

✷ ***Note: When USBRAM_EN = 0, USB RAM access is unavailable.***

✷ ***Note: USBRAM_EN can be set as '0' when leaving normal mode (High clock mode) or USB power off for power saving.***

D3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UEPDIR	-	-	UE6D	UE5D	UE4D	UE3D	UE2D	UE1D
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit [5:0] **UEnD [5:0], n=1~6:** Endpoint n IN/OUT direction setting.
0 = EPn only handshakes to IN token packet.
1 = EPn only handshakes to OUT token packet.

D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UEPISO	-	-	UE6ISO	UE5ISO	UE4ISO	UE3ISO	UE2ISO	UE1ISO
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit [5:0] **UEnISO [5:0], n=1~6:** Endpoint n ISO mode setting.
0 = Enable interrupt/bulk mode.
1 = Enable ISO mode.

11.2.7 USB ENDPOINT 0 ENABLE REGISTER

An endpoint 0 (EP0) is used to initialize and control the USB device. EP0 is bi-directional (Control pipe), as the device, can both receive and transmit data, which provides to access the device configuration information and allows generic USB status and control accesses.

D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE0R	UE0E	UE0M1	UE0M0	OUT_STALL_EN	IN_STALL_EN	-	-	-
Read/Write	R/W	R/W	R/W	R/W	R/W	-	-	-
After reset	0	0	0	0	0	-	-	-

Bit 7 **UE0E:** EP0 Enable bit.
0 = EP0 disable (No handshake for any EP0 USB packet).
1 = EP0 enable.

Bit [6:5] **UE0M [1:0]:** The endpoint 0 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 0. For example, if the endpoint 0's mode bit is set to 00 that is NAK IN/OUT mode as shown in Table, The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 0.The bit 5 UE0M0 will auto reset to zero when the ACK transaction complete.

UE0M1	UE0M0	IN/OUT Token handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

Bit4 **OUT_STALL_EN:** EP0 to handshake STALL to EP0 OUT transaction.
0 = Disable EP0 OUT STALL function.
1 = Enable EP0 OUT STALL function. If this function is enable, EP0 OUT token always handshakes STALL. The EP0 IN token handshakes depend on UE0R. This flag will be auto-cleared by HW at next SETUP token.

Bit 3 **IN_STALL_EN:** EP0 to handshake STALL to EP0 IN transaction.
0 = Disable EP0 IN STALL function.
1 = Enable EP0 IN STALL function. If this function is enable, EP0 IN token always handshakes STALL. The EP0 OUT token handshakes depend on UE0R. This flag will be auto-cleared by HW at next SETUP token.

D6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE0R_C	-	UE0C6	UE0C5	UE0C4	UE0C3	UE0C2	UE0C1	UE0C0
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	0	0	0	0	0	0	0

Bit [6:0] **UE0Cn [6:0], n=0~6:** Indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint 0 FIFO.

11.2.8 USB ENDPOINT 1 ENABLE REGISTER

An endpoint 1 (EP1) is used to initialize and control the USB device. EP1 is bi-directional (Control pipe), as the device, can both receive and transmit data, which provides to access the device configuration information and allows generic USB status and control accesses.

D7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE1R	UE1E	UE1M1	UE1M0	-	-	-	-	-
Read/Write	R/W	R/W	R/W	-	-	-	-	-
After reset	0	0	0	-	-	-	-	-

Bit 7 **UE1E:** EP1 Enable bit.
0 = EP1 disable (No handshake for any EP1 USB packet).
1 = EP1 enable.

Bit [6:5] **UE1M [1:0]:** The endpoint 1 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 1. For example, if the endpoint 1's mode bit is set to 00 that is NAK IN/OUT mode as shown in Table, The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 1.The bit 5 UE1M0 will auto reset to zero when the ACK transaction complete.

UE1M1	UE1M0	IN/OUT Token handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE1R_C	UE1C7	UE1C6	UE1C5	UE1C4	UE1C3	UE1C2	UE1C1	UE1C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **UE1Cn [7:0], n=0~7:** Indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint 1 FIFO.

11.2.9 USB ENDPOINT 2 ENABLE REGISTER

An endpoint 2 (EP2) is used to initialize and control the USB device. EP2 is bi-directional (Control pipe), as the device, can both receive and transmit data, which provides to access the device configuration information and allows generic USB status and control accesses.

D9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE2R	UE2E	UE2M1	UE2M0	-	-	-	-	-
Read/Write	R/W	R/W	R/W	-	-	-	-	-
After reset	0	0	0	-	-	-	-	-

Bit 7 **UE2E:** EP2 Enable bit.
0 = EP2 disable (No handshake for any EP2 USB packet).
1 = EP2 enable.

Bit [6:5] **UE2M [1:0]:** The endpoint 2 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 2. For example, if the endpoint 2's mode bit is set to 00 that is NAK IN/OUT mode as shown in Table, The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 1.The bit 5 UE2M0 will auto reset to zero when the ACK transaction complete.

UE2M1	UE2M0	IN/OUT Token handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE2R_C	UE2C7	UE2C6	UE2C5	UE2C4	UE2C3	UE2C2	UE2C1	UE2C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **UE2Cn [7:0], n=0~7:** Indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint 2 FIFO.

11.2.10 USB ENDPOINT 3 ENABLE REGISTER

An endpoint 3 (EP3) is used to initialize and control the USB device. EP3 is bi-directional (Control pipe), as the device, can both receive and transmit data, which provides to access the device configuration information and allows generic USB status and control accesses.

DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE3R	UE3E	UE3M1	UE3M0	-	-	-	-	-
Read/Write	R/W	R/W	R/W	-	-	-	-	-
After reset	0	0	0	-	-	-	-	-

Bit 7 **UE3E:** EP3 Enable bit.
0 = EP3 disable (No handshake for any EP3 USB packet).
1 = EP3 enable.

Bit [6:5] **UE3M [1:0]:** The endpoint 3 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 3. For example, if the endpoint 3's mode bit is set to 00 that is NAK IN/OUT mode as shown in Table, The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 3.The bit 5 UE3M0 will auto reset to zero when the ACK transaction complete.

UE3M1	UE3M0	IN/OUT Token handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE3R_C	UE3C7	UE3C6	UE3C5	UE3C4	UE3C3	UE3C2	UE3C1	UE3C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **UE3Cn [7:0], n=0~7:** Indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint 3 FIFO.

11.2.11 USB ENDPOINT 4 ENABLE REGISTER

An endpoint 4 (EP4) is used to initialize and control the USB device. EP4 is bi-directional (Control pipe), as the device, can both receive and transmit data, which provides to access the device configuration information and allows generic USB status and control accesses.

DDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE4R	UE4E	UE4M1	UE4M0	-	-	-	-	-
Read/Write	R/W	R/W	R/W	-	-	-	-	-
After reset	0	0	0	-	-	-	-	-

Bit 7 **UE4E:** EP0 Enable bit.
0 = EP4 disable (No handshake for any EP4 USB packet).
1 = EP4 enable.

Bit [6:5] **UE4M [1:0]:** The endpoint 4 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 4. For example, if the endpoint 4's mode bit is set to 00 that is NAK IN/OUT mode as shown in Table, The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 4.The bit 5 UE4M0 will auto reset to zero when the ACK transaction complete.

UE4M1	UE4M0	IN/OUT Token handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

DEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE4R_C	UE4C7	UE4C6	UE4C5	UE4C4	UE4C3	UE4C2	UE4C1	UE4C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **UE4Cn [7:0], n=0~7:** Indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint 4 FIFO.

11.2.12 USB ENDPOINT 5 ENABLE REGISTER

An endpoint 5 (EP5) is used to initialize and control the USB device. EP5 is bi-directional (Control pipe), as the device, can both receive and transmit data, which provides to access the device configuration information and allows generic USB status and control accesses.

DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE5R	UE5E	UE5M1	UE5M0	-	-	-	-	-
Read/Write	R/W	R/W	R/W	-	-	-	-	-
After reset	0	0	0	-	-	-	-	-

Bit 7 **UE5E:** EP5 Enable bit.
0 = EP5 disable (No handshake for any EP5 USB packet).
1 = EP5 enable.

Bit [6:5] **UE5M [1:0]:** The endpoint 5 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 5. For example, if the endpoint 5's mode bit is set to 00 that is NAK IN/OUT mode as shown in Table, The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 5.The bit 5 UE5M0 will auto reset to zero when the ACK transaction complete.

UE5M1	UE5M0	IN/OUT Token handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE5R_C	UE5C7	UE5C6	UE5C5	UE5C4	UE5C3	UE5C2	UE5C1	UE5C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **UE5Cn [7:0], n=0~7:** Indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint 5 FIFO.

11.2.13 USB ENDPOINT 6 ENABLE REGISTER

An endpoint 6 (EP6) is used to initialize and control the USB device. EP6 is bi-directional (Control pipe), as the device, can both receive and transmit data, which provides to access the device configuration information and allows generic USB status and control accesses.

E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE6R	UE6E	UE6M1	UE6M0	-	-	-	-	-
Read/Write	R/W	R/W	R/W	-	-	-	-	-
After reset	0	0	0	-	-	-	-	-

Bit 7 **UE6E:** EP6 Enable bit.
0 = EP6 disable (No handshake for any EP6 USB packet).
1 = EP6 enable.

Bit [6:5] **UE6M [1:0]:** The endpoint 1 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 6. For example, if the endpoint 6's mode bit is set to 00 that is NAK IN/OUT mode as shown in Table, The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 6.The bit 5 UE6M0 will auto reset to zero when the ACK transaction complete.

UE6M1	UE6M0	IN/OUT Token handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE6R_C	UE6C7	UE6C6	UE6C5	UE6C4	UE6C3	UE6C2	UE6C1	UE6C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **UE6Cn [7:0], n=0~7:** Indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint 6 FIFO.

11.2.14 UPID REGISTER

Forcing bits allow firmware to directly drive the D+ and D– pins.

E3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UPID	-	-	-	-	-	UBDE	DDP	DDN
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

Bit 2 **UBDE:** Enable direct drive USB D+ and D- bus.
0 = Disable the ability of driving D+ and D- signals on the USB bus.
1 = Enable.

Bit 1 **DDP:** Drive D+ on the USB bus.
0 = drive D+ low.
1 = drive D+ high.

Bit 0 **DDN:** Drive D- on the USB bus.
0 = drive D- low.
1 = drive D- high.

11.2.15 USB Frame Number Register

The 11-bit frame number of the Start-Of-Frame(SOF) packet. This number is updated by H/W automatically when SOF packet is received.

E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UFRMNO_L	FRMNO7	FRMNO6	FRMNO5	FRMNO4	FRMNO3	FRMNO2	FRMNO1	FRMNO0
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0

E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UFRMNO_H	-	-	-	-	-	FRMNO10	FRMNO9	FRMNO8
Read/Write	-	-	-	-	-	R	R	R
After reset	-	-	-	-	-	0	0	0

11.2.16 USB PHY Register

E6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PHYPRM	PHYPRM_EN	-	-	-	-	-	-	-
Read/Write	R/W	-	-	-	-	-	-	-
After reset	0	-	-	-	-	-	-	-

Bit7 **PHYPRM_EN:** USB PHY control function
0 = Disable
1 = PHY control function enable.

When the “High_CLK” code option setting is “IHRC_16M” and USB function is used, users must add the function fnUSB_CheckPhyram before enable USB function. The example is shown below.

Example: How to use fnUSB_CheckPhyram function.

```
; Initializing system.
INITIAL:
            …
            …

; Call fnUSB_CheckPhyram.
            CALL        fnUSB_CheckPhyram

; Enalbe USB function.
            B0BCLR      FUE0E               ; Disable EP0
            B0BCLR      FUSBIRQ             ; Clear USB IRQ flag
            B0BSET      FUSBIEN             ; Enable USB IRQ
            B0BSET      FUSB_IE             ; Enable USB event interrupt
            B0BSET      FBUS_IE             ; Enable Bus event interrupt
            B0BSET      FBUSWK_IE           ; Enable Wake Up event interrupt
            B0BSET      UDA.7               ; Enable USB
            CALL        fnUSB_EnablePhyRam  ; Enable D+ PU, VREG33, PHY, USB RAM
            B0BSET      FGIE                ; Global interrupt enable
            JMP         MAIN

MAIN:
            …
            …

; fnUSB_CheckPhyram function's code.
fnUSB_CheckPhyram:
            B0MOV       Y,#0x00
            B0MOV       Z,#0xE7
            B0MOV       A,@YZ
            B0MOV       L,A
            B0MOV       Z,#0xE8
            B0MOV       A,@YZ
            B0MOV       H,A
            B0MOV       Y,#0x3F
            B0MOV       Z,#0xFD
            MOVC
            CMPRS       A,L
            JMP         fnUSB_CheckPhyram_80
            RLCM        R
            RLCM        R
            SWAP        R
            XOR         A,H
            AND         A,#0xC0
            B0BTS0      FZ
            JMP         fnUSB_CheckPhyram_90
fnUSB_CheckPhyram_80:
            B0BCLR      PHYPRM.7
            B0BSET      PHYPRM.7
            JMP         fnUSB_CheckPhyram
fnUSB_CheckPhyram_90:
            RET
```

✷ ***Note: If the following both conditions are satisfied, user must follow demo programs given above :***

1. ***High_CLK" code option setting is "IHRC_16M.***
2. ***User's system with USB function.***

11.2.17 ENDPOINT TOGGLE BIT CONTROL REGISTER

E8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UTOGGLE	-	-	EP6_DATA01	EP5_DATA01	EP4_DATA01	EP3_DATA01	EP2_DATA01	EP1_DATA01
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit [5:0] **EPn_DATA01, n=1~6:**
0 = clearing endpoint n's toggle bit to DATA0
1 = hardware set toggle bit automatically.

11.2.18 USB ENDPOINT FIFO ADDRESS SETTING REGISTER

- ***Note: EP1FIFO's start address is fixed at 0x540.***
- ***Note: EP2~EP6 FIFO's high address is always mapped to 0x500.***

E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP2FIFO_ADDR	EP2FIFO7	EP2FIFO6	EP2FIFO5	EP2FIFO4	EP2FIFO3	EP2FIFO2	EP2FIFO1	EP2FIFO0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **EP2FIFO_ADDR [7:0]:** EP2 FIFO start address.

EAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP3FIFO_ADDR	EP3FIFO7	EP3FIFO6	EP3FIFO5	EP3FIFO4	EP3FIFO3	EP3FIFO2	EP3FIFO1	EP3FIFO0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **EP3FIFO_ADDR [7:0]:** EP3 FIFO start address.

EBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP4FIFO_ADDR	EP4FIFO7	EP4FIFO6	EP4FIFO5	EP4FIFO4	EP4FIFO3	EP4FIFO2	EP4FIFO1	EP4FIFO0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **EP4FIFO_ADDR [7:0]:** EP4 FIFO start address.

ECH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP5FIFO_ADDR	EP5FIFO7	EP5FIFO6	EP5FIFO5	EP5FIFO4	EP5FIFO3	EP5FIFO2	EP5FIFO1	EP5FIFO0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **EP5FIFO_ADDR [7:0]:** EP5 FIFO start address.

EDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP6FIFO_ADDR	EP6FIFO7	EP6FIFO6	EP6FIFO5	EP6FIFO4	EP6FIFO3	EP6FIFO2	EP6FIFO1	EP6FIFO0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **EP6FIFO_ADDR [7:0]:** EP6 FIFO start address.

12 PS/2 INTERFACE

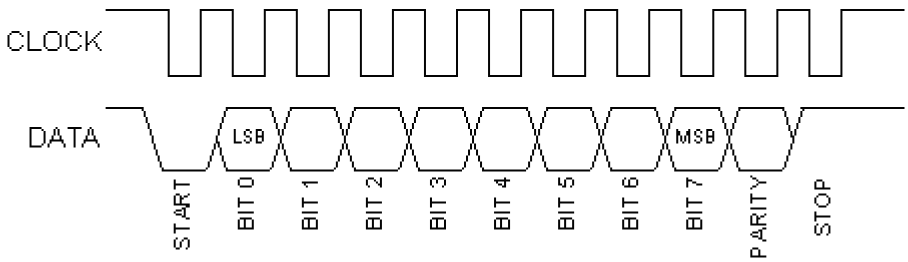
12.1 OVERVIEW

PS/2 interface is built in the microcontroller. There are SCK, SDA pins and includes internal 5K pull-up resistors. Use the firmware to achieve PS/2 communication.

12.2 PS/2 OPERATION

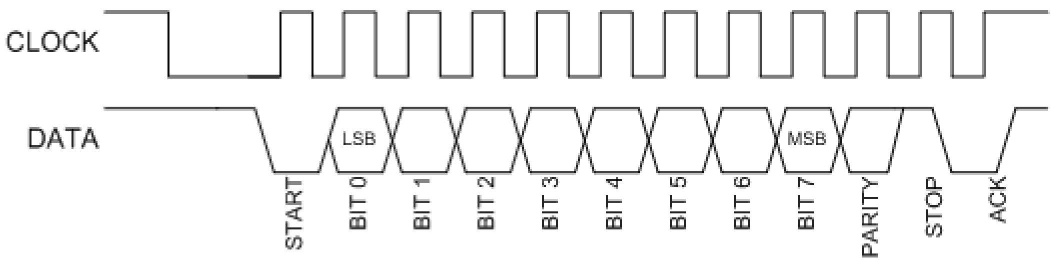
PS/2 is a kind of serial interface to control PC's peripheral devices. This interface extensively applies to mouse and keyboard. PS/2 waveform is as following.

Device to Host:



One pocket includes start bit (Clock and data pins are low status.), one byte data (LSB to MSB), parity bit (odd parity) and stop bit (Clock is low status and data is high status.). The clock typical frequency is 15KHz.

Host to Device:

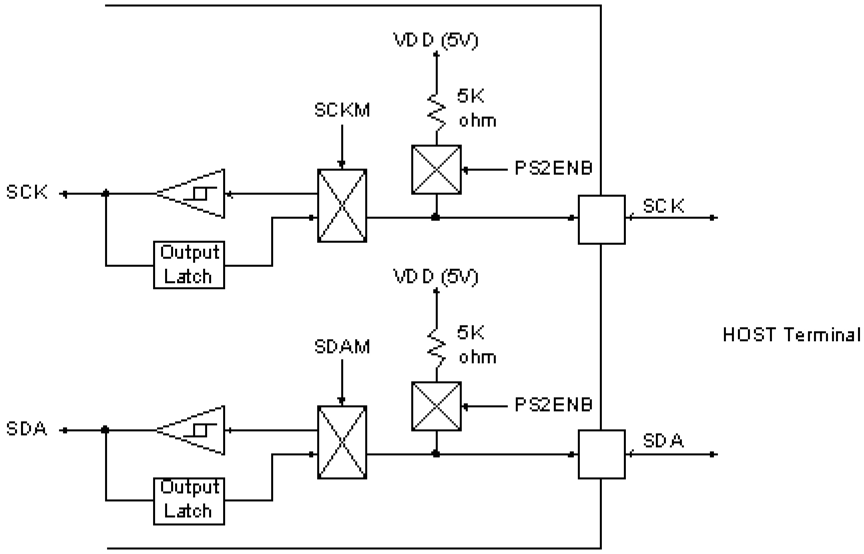


One pocket includes a period silence time (Clock falling to first clock rising. The period time is ≤ 15ms.), start bit (Clock and data pins are low status.), one byte data (LSB to MSB), parity bit (odd parity), stop bit (Clock is low status and data is high status.) and ACK bit (Device set data pin to low status.). The clock typical frequency is 15KHz.

Firmware must include clock 15KHz signal generator, odd parity calculate, start/stop/ack bit routine to get a basic PS/2 protocol signal, and follow PS/2 protocol specification to define PC's peripheral device (mouse or keyboard) transmitting data form and contents.

12.3 PS/2 DESCRIPITON

PS/2 interface is from SCK and SDA pins which open-drain structure. SCKM, SDAM bits control SCK, SDA direction. PS/2 builds in internal 5K ohm pull-up resistors controlled by PS2ENB bit of PS2M register. PS2ENB=0, internal pull-up resistors disable and SCK, SDA are open-drain without pull-up resistor. PS2ENB=1, internal pull-up resistor enable. PS/2 communication is controlled by firmware.



12.4 PS/2 REGISTER

0EEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PS2M	PS2ENB	-	-	-	SDA	SCK	SDAM	SCKM
Read/Write	R/W	-	-	-	R/W	R/W	R/W	R/W
After reset	0	-	-	-	0	0	0	0

Bit 7 **PS2ENB:** PS2 internal 5K ohm pull-up resistor control bit.
0 = Disable.
1 = Enable.

Bit [3:2] **SDA, SCK:** SDA, SCK data buffer.0 = End of transfer.
0 = Data 0.
1 = Data 1.

Bit [1:0] **SDAM, SCKM:** SDA, SCK mode control bit.
0 = Input mode.
1 = Output mode.

> ✸ ***Note: Use the PS/2 communication, the USB must be disable (UDE=0) and VREG33 must be disabled.***

13 IN SYSTEM PROGRAM FLASH ROM

13.1 OVERVIEW

The SN8F22E80B MCU integrated device feature in-system programmable (ISP) FLASH memory for convenient, upgradeable code storage. The FLASH memory may be programmed via the SONiX 8 bit MCU programming interface or by application code. The SN8F22E80B provides security options at the disposal of the designer to prevent unauthorized access to information stored in FLASH memory. ISP Flash ROM provided user an easy way to storage data into Flash ROM. The ISP concept is memory mapping idea that is to move RAM buffer to flash ROM. Choice ROM/RAM address and executing ROM programming command – PECMD, and L/H data will be programmed into address Z/Y.

RAM Address	RAM (byte) bit7 ~ bit0
X	DATA0
X+1	DATA1
X+2	DATA2
X+3	DATA3
…	
…	
X+N	DATAN

=>

ROM Address	Flash ROM (word) bit15 ~ bit8	bit7 ~ bit0
Y	DATA1	DATA0
Y+1	DATA3	DATA2
Y+2		
Y+3		
…		
…		
Y+M	DATAN	DATAN-1

During Flash program or erase operation, the MCU is stalled, although peripherals (Timers, WDT, I/O, PWM, etc.) remain active. When PECMD register is set to execute ISP program and erase operations, the program counter stops, op-code can't be dumped from flash ROM, instruction stops operating, and program execution is hold not to active. At this time hardware depends on ISP operation configuration to do flash ROM erasing and flash ROM programming automatically. After ISP operation is finished, hardware releases system clock to make program counter running, system returns to last operating mode, and the next instruction is executed. Recommend to add two "NOP" instructions after ISP operations.

- **ISP flash ROM erase time = 25ms……1-page, 256-word.**
- **ISP flash ROM program time = 28us……1-word.**
  ISP flash ROM program time = 56us……2-word.
  …
  ISP flash ROM program time = 448us……16-word.
  …
  ISP flash ROM program time = 896us……32-word.

> ✵ ***Note:***
> 1. ***Watch dog timer should be clear before the Flash write (program) or erase operation, or watchdog timer would overflow and reset system during ISP operating.***
> 2. ***Besides program execution, all functions keep operating during ISP operating, e.g. timer, ADC, SIO, UART... All interrupt events still active and latch interrupt flags automatically. If any interrupt request occurs during ISP operating, the interrupt request will be process by program after ISP finishing.***

13.2 ISP FLASH ROM ERASE OPERATION

ISP flash ROM erase operation is to clear flash ROM contents to blank status "1". Erasing ROM length is 256-word and has ROM page limitation. ISP flash ROM erase ROM map is as following:

<table>
<tr><th colspan="2" rowspan="2">ISP ROM MAP</th><th colspan="16">ROM address bit0~bit6 (hex)</th></tr>
<tr><td>0000</td><td>0001</td><td>0002</td><td>…</td><td>0010</td><td>0011</td><td>…</td><td>0080</td><td>0081</td><td>…</td><td>00A0</td><td>00A1</td><td>…</td><td>00FE</td><td>00FF</td></tr>
<tr><td rowspan="12">ROM address bit7~bit15 (hex)</td><td>0000</td><td colspan="15">This page includes reset vector and interrupt sector. We strongly recommend to reserve the area not to do ISP erase.</td></tr>
<tr><td>0100</td><td colspan="15">One ISP Erase Page</td></tr>
<tr><td>0200</td><td colspan="15">One ISP Erase Page</td></tr>
<tr><td>0300</td><td colspan="15">One ISP Erase Page</td></tr>
<tr><td>0400</td><td colspan="15">One ISP Erase Page</td></tr>
<tr><td>0500</td><td colspan="15">One ISP Erase Page</td></tr>
<tr><td>…</td><td colspan="15">One ISP Erase Page</td></tr>
<tr><td>3E00</td><td colspan="15">One ISP Erase Page</td></tr>
<tr><td>3D00</td><td colspan="15">One ISP Erase Page</td></tr>
<tr><td>3E00</td><td colspan="15">One ISP Erase Page</td></tr>
<tr><td>3F00</td><td colspan="15">This page includes ROM reserved area. We strongly recommend to reserve the area not to do ISP erase.</td></tr>
</table>

ISP flash ROM erase density is 256-word which limits erase page boundary. The first 256-word of flash ROM (0x0000~0x0100) includes reset vector and interrupt vectors related essential program operation, and the last page 256-word of flash ROM (0x3F00~0x3FFF) includes system reserved ROM area, we strongly recommend do not execute ISP flash ROM erase operation in the two pages. Flash ROM area 0x0000~0x3FFF includes 64-page for ISP flash ROM erase operation.

The first step to do ISP flash ROM erase is to address ROM-page location. The address must be the head location of a page area, e.g. 0x0100, 0x0200, 0x0300…0x0D00, 0x0E00 and 0x0F00. Z [7:0] and Y [7:0] define the target starting address [15:0] of flash ROM. Write the start address into PEROML and Y registers, set PECMD register to "0xC3", and the system start to execute ISP flash ROM erase operation.

1 Example: Use ISP flash ROM erase to clear 0x0100~0x01FF contents of flash ROM.

```
; Set erased start address 0x0100.
        MOV       A, #0x00
        B0MOV     Z, A            ; Move low byte address 0x00 to Z
        MOV       A, #0x01
        B0MOV     Y, A            ;Move high byte address 0x01 to Y

; Clear watchdog timer.
        MOV       A,#0X5A
        B0MOV     WDTR,A

; Start to execute ISP flash ROM erase operation.
        B0BSET    FIHRC_EN        ; Enable IHRC.
        MOV       A,#0XC3         ; Start to page erase.
        B0MOV     PECMD, A
        NOP                       ; NOP Delay
        NOP
        NOP
        B0BCLR    FIHRC_EN        ; Disable IHRC.
                                  ; The end of ISP flash ROM erase operation.
```

The two "NOP" instructions make a short delay to let system stable after ISP flash ROM erase operation.

> ✸ ***Note:***
> 1. ***Don't execute ISP flash ROM erase operation for the first page and the last page, or affect program operation.***
> 2. ***IHRC must enable before executing ISP flash ROM erase operation if "High_CLK" code option setting is "X'TAL" and PLL clock source is not "IHRC_16M".***

13.3 ISP FLASH ROM PROGRAM OPERATION

ISP flash ROM program operation is to write data into flash ROM by program. Program ROM doesn't limit written ROM address and length, but limits 32-word density of one page. The number of ISP flash ROM program operation can be 1-word ~ 32-word at one time, but these words must be in the same page. ISP flash ROM program ROM map is as following:

<table>
<tr><th colspan="2" rowspan="2">ISP ROM MAP</th><th colspan="9">ROM address bit0~bit4 (hex)</th></tr>
<tr><td>0000</td><td>0001</td><td>0002</td><td>…</td><td>000F</td><td>0010</td><td>…</td><td>001E</td><td>001F</td></tr>
<tr><td rowspan="13">ROM address bit5~bit15 (hex)</td><td>0000</td><td colspan="9">This page includes reset vector and interrupt sector. We strongly recommend to reserve the area not to do ISP erase.</td></tr>
<tr><td>0020</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>0040</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>0060</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>0080</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>00A0</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>00C0</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>00E0</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>…</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>3F00</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>3F20</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>…</td><td colspan="9">One ISP Program Page</td></tr>
<tr><td>3F80</td><td colspan="9">This page includes ROM reserved area. We strongly recommend to reserve the area not to do ISP erase.</td></tr>
</table>

ISP flash ROM program page density is 32-word which limits program page boundary. The first 32-word of flash ROM (0x0000~0x001F) includes reset vector and interrupt vectors related essential program operation, and the last page 32-word of flash ROM (0x3FD0~0x3FFF) includes system reserved ROM area, we strongly recommend do not execute ISP flash ROM program operation in the two pages. Flash ROM area 0x0000~0x3FFF includes 512-page for ISP flash ROM program operation.

ISP flash ROM program operation is a simple memory mapping operation. The first step is to plan a RAM area to store programmed data and keeps the RAM address for IS RAM addressing. The second step is to plan a ROM area will be programmed from RAM area in ISP flash ROM program operation. The RAM addressing is through L[7:0] and H[2:0] 11-bit buffer to configure the start RAM address. The RAM data storage sequence is down-up structure. The first RAM data is the low byte data of the first word of flash ROM. The second RAM data is the high byte data of the first word of ROM, and so on.

Before ISP ROM programming execution, set the length by program. Z [7:0] and Y [7:0] define the target starting address [15:0] of flash ROM. Write the start address into Z and Y registers, set PECMD register to "0x5A", and the system start to execute ISP flash ROM program operation. If the programming length is over ISP flash ROM program page boundary, the hardware immediately stops programming flash ROM after finishing programming the last word of the ROM page. So it is very important to plan right ROM address and programming length.

- **Case 1:** 32-word ISP program. RAM buffer length is 64-byte and RAM address is X ~ X+63. The page address of flash ROM is Y ~ Y+31. The Y is the start address and set to PEROML, PEROMH registers.

RAM (byte)

RAM Address 64-byte	bit7 ~ bit0
X	**DATA0**
X+1	**DATA1**
X+2	**DATA2**
X+3	**DATA3**
…	**…**
X+62	**DATA62**
X+63	**DATA63**

=>

Flash ROM (word)

ROM Address 32-word	bit15 ~ bit8	bit7 ~ bit0	
Y	**DATA1**	**DATA0**	**The head of the page. The start address of ISP.**
Y+1	**DATA3**	**DATA2**	
Y+2	**…**	**…**	
Y+3	**…**	**…**	
…	**…**	**…**	
…	**…**	**…**	
Y+31	**DATA63**	**DATA62**	**The end of the page. The end address of ISP.**

Example: Use ISP flash ROM program to program 32-word data to flash ROM as case 1. Set RAM buffer start address is 0x010. Set flash ROM programmed start address is 0x0020.

```
; Load data into 64-byte RAM buffer.
                …
                …

; Set RAM start address of 64-byte buffer.
                MOV             A, #0x10
                B0MOV           L, A                    ; Set L[7:0] to 0x10.
                MOV             A, #0x00
                B0MOV           H, A                    ; Set H[2:0] to 000b.

; Set programmed start address of flash ROM to 0x0020.
                MOV             A, #0x20
                B0MOV           Z, A                    ; Move low byte address 0x20 to Z.
                MOV             A, #0x00
                B0MOV           Y, A                    ;Move high byte address 0x00 to Y

; Clear watchdog timer.
                MOV             A,#0X5A
                B0MOV           WDTR,A

; Start to execute ISP flash ROM program operation.
                B0BSET          FIHRC_EN                ; Enable IHRC.
                MOV             A,#0X5A                 ; Start to program flash ROM.
                B0MOV           PECMD, A
                NOP                                     ; NOP Delay
                NOP
                NOP
                B0BCLR          FIHRC_EN                ; Disable IHRC.
                                                        ; The end of ISP flash ROM program operation.
```

The two "NOP" instructions make a short delay to let system stable after ISP flash ROM program operation.

> ✸ ***Note:***
> 1. ***Don't execute ISP flash ROM program operation for the first page and the last page, or affect program operation.***
> 2. ***IHRC must enable before executing ISP flash ROM program operation if "High_CLK" code option setting is "X'TAL" and PLL clock source is not "IHRC_16M".***

13.4 ISP PROGRAM/ERASE CONTROL REGISTER

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PECMD	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **PECMD [7:0]:** ISP operation control register.
0x5A: Page Program (32 words / page).
0xC3: Page Erase (256 words / page).
Others: Reserved.

✸ ***Note: Before executing ISP program and erase operations, clear PECMD register is necessary. After ISP configuration, set ISP operation code in "MOV A,I" and "B0MOV M,A" instructions to start ISP operations.***

13.5 ISP ROM ADDRESS REGISTER

ISP ROM address length is 16-bit and separated into Z and Y[5:0] registers. Before ISP execution, set the head address of ISP ROM by program.

083H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Z	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

Bit [7:0] **Z[7:0]:** The low byte buffer of ISP ROM address.

084H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Y	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

Bit [5:0] **Y[5:0]:** The high byte buffer of ISP ROM address.

13.6 ISP RAM ADDRESS REGISTER

ISP RAM address length is 11-bit and separated into L register and H[2:0] bits. Before ISP execution, set the head address of ISP RAM by program.

080H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
L	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

Bit [7:0] **L[7:0]:** ISP RAM address [7:0].

081H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

Bit [2:0] **H[2:0]:** ISP RAM address [10:8].

13.7 ISP ROM PROGRAMMING LENGTH REGISTER

ISP programming length is 1-word ~ 32-word. ISP ROM programming length is controlled by H[7:3] bits which is 5-bit format. Before ISP ROM programming execution, set the length by program

Bit [7:0] **L[7:0]:** ISP RAM address [7:0].

081H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

Bit [7:3] **H[7:3]:** ISP ROM programming length control register.

H[7:3]=0: ISP programming length is 1-word.
H[7:3]=1: ISP programming length is 2-word.
…
…
H[7:3]=30: ISP programming length is 31-word.
H[7:3]=31: ISP programming length is 32-word.

14 INSTRUCTION TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
	MOV	A,M	A ← M	-	-	√	1
M	MOV	M,A	M ← A	-	-	-	1
O	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
V	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
E	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I, "M" only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z…)	-	-	-	1
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
A	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1+N
R	ADD	A,M	A ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
I	ADD	M,A	M ← A + M, if occur carry, then C=1, else C=0	√	√	√	1+N
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
H	ADD	A,I	A ← A + I, if occur carry, then C=1, else C=0	√	√	√	1
M	SBC	A,M	A ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
E	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
T	SUB	A,M	A ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
I	SUB	M,A	M ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1+N
C	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
	DAA		To adjust ACC's data format from HEX to DEC.	√	-	-	1
	MUL	A,M	R, A ← A * M, The LB of product stored in Acc and HB stored in R register. ZF affected by Acc.	-	-	√	2
	AND	A,M	A ← A and M	-	-	√	1
L	AND	M,A	M ← A and M	-	-	√	1+N
O	AND	A,I	A ← A and I	-	-	√	1
G	OR	A,M	A ← A or M	-	-	√	1
I	OR	M,A	M ← A or M	-	-	√	1+N
C	OR	A,I	A ← A or I	-	-	√	1
	XOR	A,M	A ← A xor M	-	-	√	1
	XOR	M,A	M ← A xor M	-	-	√	1+N
	XOR	A,I	A ← A xor I	-	-	√	1
	COM	M	A ← M (1's complement).	-	-	√	1
	COMM	M	M ← M (1's complement).	-	-	√	1
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
R	RRC	M	A ← RRC M	√	-	-	1
O	RRCM	M	M ← RRC M	√	-	-	1+N
C	RLC	M	A ← RLC M	√	-	-	1
E	RLCM	M	M ← RLC M	√	-	-	1+N
S	CLR	M	M ← 0	-	-	-	1
S	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
B	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
R	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1+ S
A	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
N	INC	M	A ← M + 1.	-	-	√	1
C	INCM	M	M ← M + 1.	-	-	√	1+N
H	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1+ S
	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1+N+S
	DEC	M	A ← M – 1.	-	-	√	1
	DECM	M	M ← M – 1.	-	-	√	1+N
	BTS0	M.b	If M.b = 0, then skip next instruction	-	-	-	1 + S
	BTS1	M.b	If M.b = 1, then skip next instruction	-	-	-	1 + S
	B0BTS0	M.b	If M(bank 0).b = 0, then skip next instruction	-	-	-	1 + S
	B0BTS1	M.b	If M(bank 0).b = 1, then skip next instruction	-	-	-	1 + S
	TS0M	M	If M = 0, Z = 1. Else Z = 0.	-	-	√	1
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALLHL		Stack ← PC15~PC0, PC15~PC8 ← H register, PC7~PC0 ← L register	-	-	-	2

	CALLYZ	Stack ← PC15~PC0, PC15~PC8 ← Y register, PC7~PC0 ← Z register	-	-	-	2
M	RET	PC ← Stack	-	-	-	2
I	RETI	PC ← Stack, and to enable global interrupt	-	-	-	2
S	RETLW I	PC ← Stack, and load I to ACC.	-	-	-	2
C	NOP	No operation	-	-	-	1

Note: 1. "M" is system register or RAM. If "M" is system registers then "N" = 0, otherwise "N" = 1.
2. If branch condition is true then "S = 1", otherwise "S = 0".

15 ELECTRICAL CHARACTERISTIC

15.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd,USBVdd)……………………………………………………………………………… 1.8V ~ 6.0V
Input in voltage (Vin)……………………………………………………………………………… Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8F22E80B……………………………………………………………………………………… -40°C ~ + 85°C
Storage ambient temperature (Tstor) ……………………………………………………………… –40°C ~ + 125°C

15.2 ELECTRICAL CHARACTERISTIC

- **SN8F22E80B DC CHARACTERISTIC**

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	*Fcpu = 16MHz*		1.8	-	6.0	V
Operating voltage	USBVdd	*Fcpu = 16MHz*		3.6	-	6.0	V
RAM Data Retention voltage	Vdr				1.5-	-	V
Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3*Vdd	V
	ViL2	Reset pin		Vss	-	0.3*Vdd	V
Input High Voltage	ViH1	All input ports		0.7*Vdd	-	Vdd	V
	ViH2	Reset pin		0.7*Vdd	-	Vdd	V
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 1.8V. P0/P1/P2.1~P2.7/P4/P5 pins		180	360	540	KΩ
		Vin = Vss , Vdd = 3V. P0/P1/ P2.1~P2.7/P4/P5 pins		60	120	180	
		Vin = Vss , Vdd = 5V. P0/P1/P2.1~P2.7/P4/P5 pins		40	75	120	
I/O output source current	IoH	Vop = Vdd – 0.5V		6	10	-	mA
sink current	IoL	Vop = Vss + 0.5V		6	10	-	
INTn trigger pulse width	Tint0	INT0,1 interrupt request pulse width		2/fcpu	-	-	cycle
VREG33 Regulator current	IVREG33	VREG33 Max Regulator Output Current, Vcc > 4.35 volt with 10uF to GND		-	-	60	mA
Supply Current (Disable ADC)	Idd1	Normal Mode (USB PLL OFF)	Vdd= 3V, Fcpu = 16MHz	-	4.5	-	mA
			Vdd= 5V, Fcpu = 16MHz	-	4.6	-	mA
		Normal Mode (USB PLL ON)	Vdd= 3V, Fcpu = 16MHz	-	10.0	-	mA
			Vdd= 5V, Fcpu = 16MHz	-	10.2	-	mA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd= 3V, ILRC=32KHz	-	125	-	uA
			Vdd= 5V, ILRC=32KHz	-	135	-	uA
	Idd3	Sleep Mode	Vdd= 5V	-	5	15	uA
	Idd4	Green Mode (No loading, Watchdog Disable)	Vdd= 3V, IHRC=16MHz	-	700	-	uA
			Vdd= 5V, IHRC=16MHz	-	700		uA
			Vdd= 3V, ILRC=32KHz	-	115		uA
			Vdd= 5V, ILRC=32KHz	-	120		uA
Internal High Oscillator Freq.	Fihrc	Internal High RC (IHRC)	*25℃,* Vdd=6.0V Fcpu = 1mips~16mips	15.92	16	16.08	MHz
LVD Voltage	LVD18	Low voltage reset level. *25℃*		1.78	-	-	V
	LVD24	Low voltage reset/indicator level. *25℃*		2.3	2.4	2.5	V
	LVD33	Low voltage reset/indicator level. *25℃*		3.2	3.3	3.4	V

● ADC CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

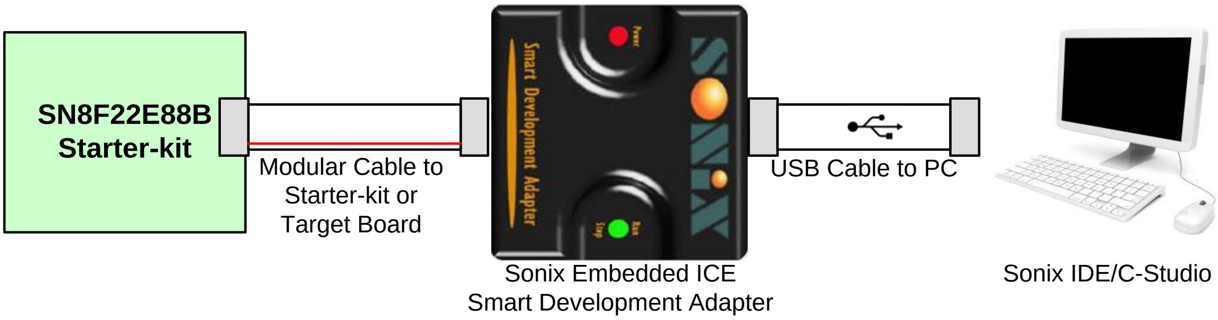
PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
ADC Operating Voltage	V_{ADC}		1.8	-	6.0	V
ADC current consumption	I_{ADC}	Vdd=3V			250	uA
		Vdd=5V			350	uA
Internal Reference Voltage power consumption	I_{INTREF}	Vdd=3V			25	uA
		Vdd=5V			50	uA
Resolution	Nr	No missing code.	10	11	12	bit
Integral Nonlinearity	INL	Vrefh= Vdd			< ±2	LSB
Differential Nonlinearity	DNL	Vrefh= Vdd			< ±2	LSB
Missing Code	MMC	Vrefh= Vdd			< ±1	LSB
Full Scale Error	Efs	Vrefh= Vdd. Vin = Vdd.			< ±1	LSB
Offset Error	Eoff	Vrefh= Vdd. Vin = 0V.			< ±1	LSB
Reference Voltage	Vrefh1	External reference voltage, Vdd=5V.	2.0		Vdd	V
*ADC current consumption	I_{ADC}	Vdd =5.0V	-	0.3	-	mA
		Vdd =3.0V	-	0.25	-	mA
Reference Voltage	Vrefh1	External reference voltage, Vdd=5V.	2		Vdd	V
	Vrefh2	Internal VDD reference voltage, Vdd=5V		Vdd		V
Analogy Input Voltage	Vain		0		Vrefh	V
Offset Calibration Range	Voff		-16		+16	mV
ADC enable time	Tast	Ready to start convert after set ADENB = “1”	100			us
ADC Clock Frequency	F_{ADCLK}	VDD=5.0V	32K		8M	Hz
		VDD=3.0V	32K		5M	Hz
ADC Conversion Cycle Time	F_{ADCYL}	VDD=1.8V~6.0V	64			1/FADCLK
ADC offset Voltage	$V_{ADCoffset}$	Non-trimmed	-10	0	+10	mV
		Trimmed	-2	0	+2	mV

“ *” These parameters are for design reference, not tested.

16 DEVELOPMENT TOOL

SONIX provides an Embedded ICE emulator system to offer SN8F22E88B firmware development. The platform is a in-circuit debugger and controlled by SONIX M2IDE software on Microsoft Windows platform. The platform includes Smart Development Adapter, SN8F22E88B Starter-kit and M2IDE software to build a high-speed, low cost, powerful and multi-task development environment including emulator, debugger and programmer. To execute emulation is like run real chip because the emulator circuit integrated in SN8F22E88B to offer a real development environment.

SN8F22E88B Embedded ICE Emulator System:



SN8F22E88B Embedded ICE Emulator includes:

- Smart Development Adapter.
- USB cable to provide communications between the Smart Development Adapter and a PC.
- SN8F22E88B Starter-Kit.
- Modular cable to connect the Smart Development Adapter and SN8F22E88B Starter-Kit or target board.
- CD-ROM with M2IDE software (M2IDE V139 or greater).

SN8F22E88B Embedded ICE Emulator Feature:

- Target's Operating Voltage: 1.8V~6.0V.
- Up to 6 hardware break points.
- System clock rate up to 16MHz (Fcpu=16mips).
- Oscillator supports internal high speed RC, internal low speed RC, external crystal/resonator and external RC.

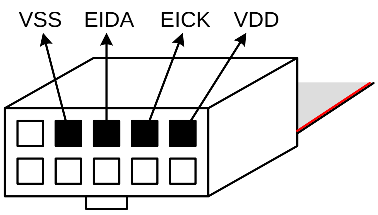
SN8F22E88B Embedded ICE Emulator Limitation:

- EIDA and EICK pins are shared with GPIO pins. In embedded ICE mode, the shared GPI function can't work. We strongly recommend planning the two pins as simple function which can be verified without debugger platform.

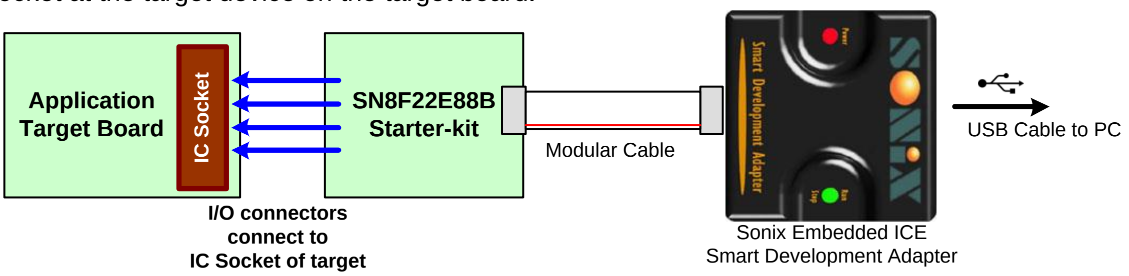
16.1 SMART DEVELOPMENT ADAPTER

Smart Development Adapter is a high speed emulator for Sonix Embedded ICE type flash MCU. It debugs and programs Sonix flash MCU and transfers MCU's system status, RAM data and system register between M2IDE and Sonix flash MCU through USB interface. The other terminal connected to SN8F22E88B Starter-kit or Target board is a 4-wire serial interface. In addition to debugger functions, the Smart Starter-Kit system also may be used as a programmer to load firmware from PC to MCU for engineering production, even mass production.

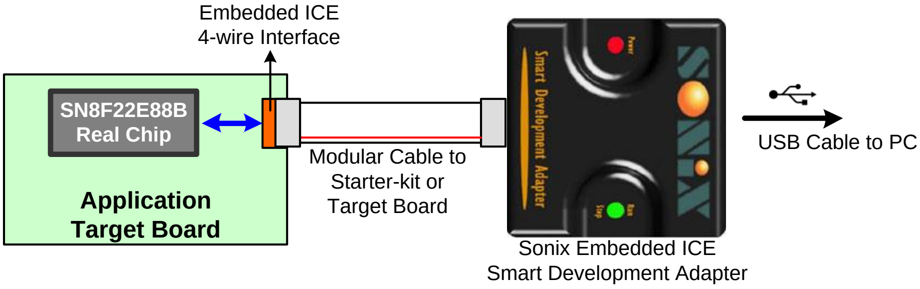
Smart Development Adapter communication with SN8F22E88B flash MCU is through a 4-wire bus. The pin definition of the Modular cable is as following:



The modular cable can be inserted into SN8F22E88B Starter-Kit plugged into the target board or inserted into a matching socket at the target device on the target board.



If the target board of application is designed and ready, the modular cable can be inserted into the target directly to replace SN8F22E88B Starter-Kit. Design the 4-wire interface connected with SN8F22E88B IC to build a real application environment. In the mode, set SN8F22E88B IC on the target is necessary, or the emulation would be error without MCU.



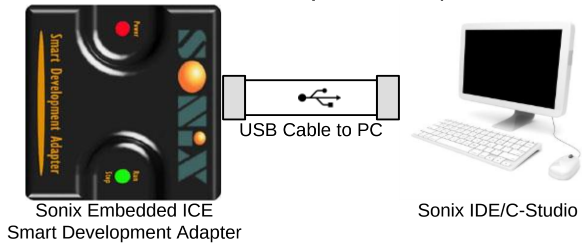
EIDA and EICK share with P5.5/P5.4 GPIO. In emulation mode, EIDA and EICK are Embedded ICE interface and not execute GPIO functions. The P5.5/P5.4 GPIO status still display on M2IDE window to simulate P5.5/P5.4 program execution.

16.2 SN8F22E88B STARTER-KIT

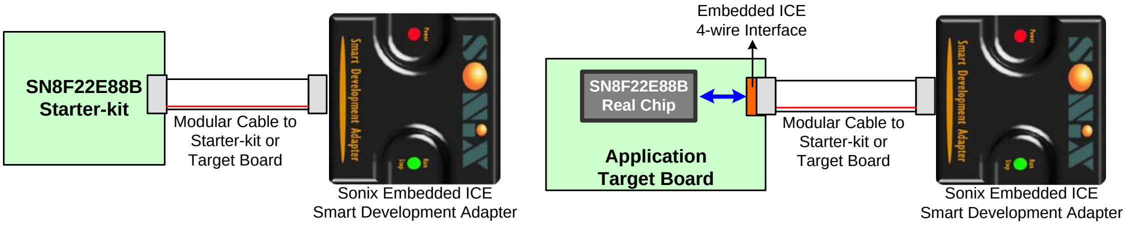
SN8F22E88B Starter-kit is an easy-development platform. It includes SN8F22E88B real chip and I/O connectors to input signal or drive extra device of user's application. It is a simple platform to develop application as target board not ready. The starter-kit can be replaced by target board, because SN8F22E88B integrates embedded ICE in-circuit debugger circuitry. The schematic and outline of SN8F22E88B Starter-Kit is as following:

16.3 EMULATOR/DEBUGGER INSTALLATION

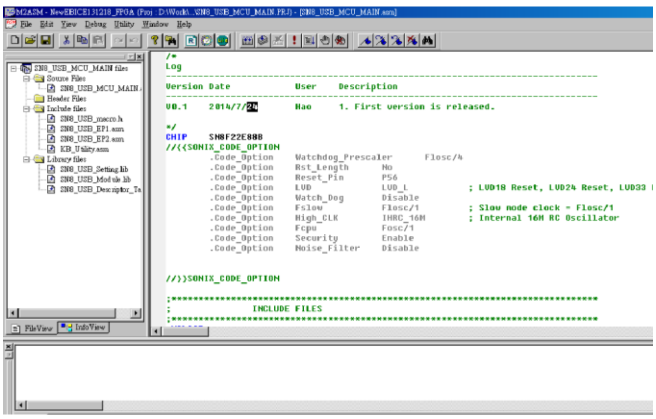
- Install the M2IDE Software (V139 or greater).
- Connect Smart Development Adapter with PC plugging in USB cable.



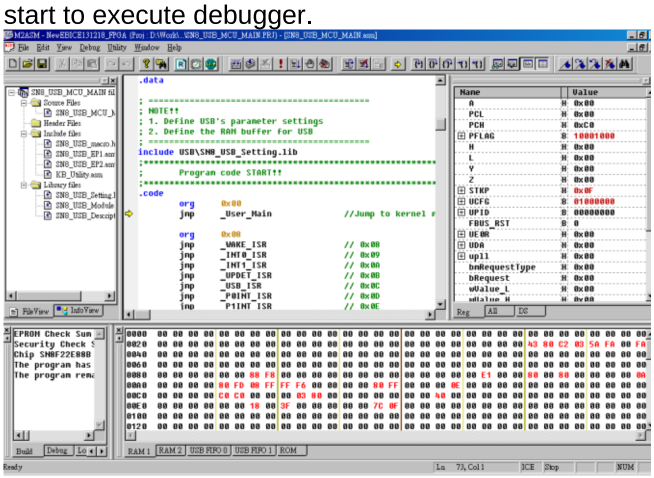
- Attach the modular cable between Smart Development Adapter and SN8F22E88B Starter-kit or target.



- Connect the power supplier to SN8F22E88B Starter-kit or target, and turn off the power.

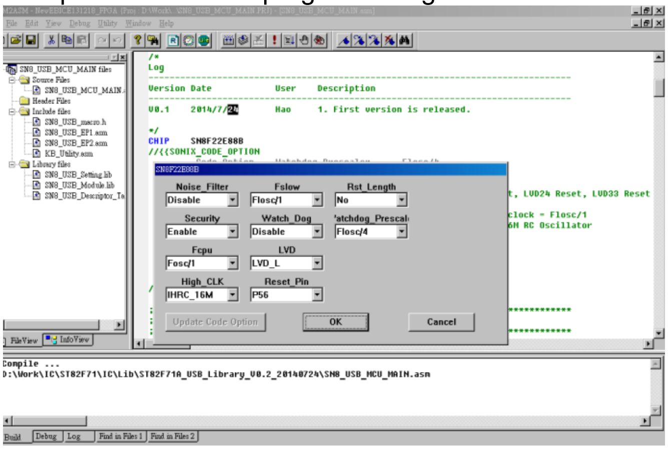


- Open M2IDE software and load firmware program (A project or a ".ASM" file).
- Turn on the power switch of SN8F22E88B Starter-kit or target.
- Embedded ICE emulator platform is installed, and start to execute debugger.

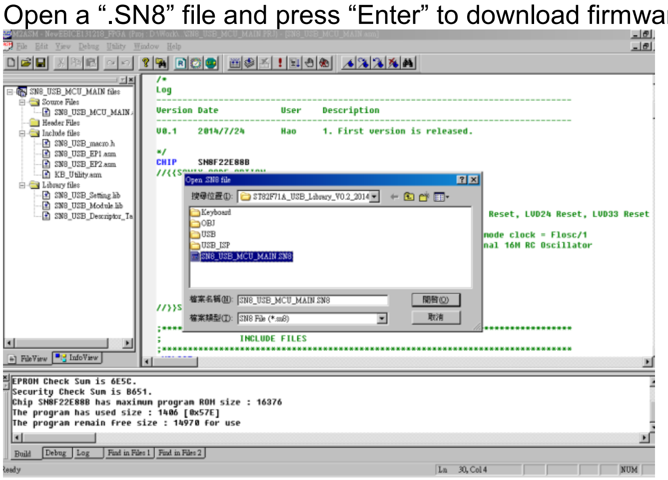


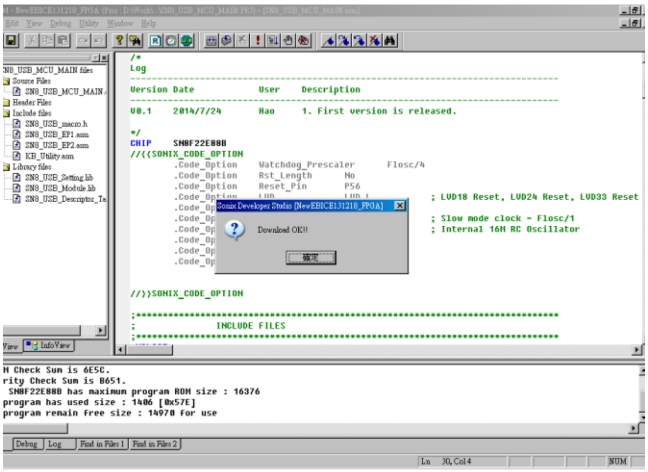
16.4 PROGRAMMER INSTALLATION

- Setup emulator/debugger environment first.
- Compile the firmware program and generate a ".SN8" file.



- Execute download (F8) function of M2IDE.
- Open a ".SN8" file and press "Enter" to download firmware to SN8F22E88B Starter-kit or target.





- Turn off the power of SN8F22E88B Starter-kit or target.
- Disconnect SN8F22E80B Starter-kit or target from Smart Development Adapter.
- Turn on the power of SN8F22E88B Starter-kit or target, and MCU works independently.

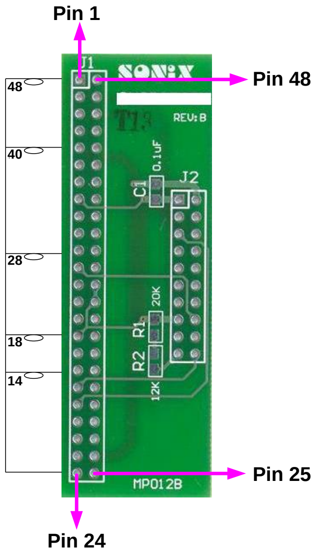
17 ROM PROGRAMMING PIN

SN8F22E80B Series MCUs Flash ROM erase/program/verify support SDA, MP-Pro writer.

- SDA: Embedded ICE interface.
- writer: Plug on SN8F22E80B Series MCUs directly.

17.1 WRITER TRANSITION BOARD SOCKET PIN ASSIGNMENT

MP-PRO Writer Transition Board:



JP3 (Mapping to 48-pin text tool):

DIP 1	1	48	DIP48
DIP 2	2	47	DIP47
DIP 3	3	46	DIP46
DIP 4	4	45	DIP45
DIP 5	5	44	DIP44
DIP 6	6	43	DIP43
DIP 7	7	42	DIP42
DIP 8	8	41	DIP41
DIP 9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP37
DIP13	13	36	DIP36
DIP14	14	35	DIP35
DIP15	15	34	DIP34
DIP16	16	33	DIP33
DIP17	17	32	DIP32
DIP18	18	31	DIP31
DIP19	19	30	DIP30
DIP20	20	29	DIP29
DIP21	21	28	DIP28
DIP22	22	27	DIP27
DIP23	23	26	DIP26
DIP24	24	25	DIP25

Writer JP1/JP2:

VDD	1	2	GND
CLK	3	4	CE
PGM	5	6	OE
D1	7	8	D0
D3	9	10	D2
D5	11	12	D4
D7	13	14	D6
VDD	15	16	VPP
HLS	17	18	RST
-	19	20	ALSB/PDB

JP1 for Writer transition board
JP2 for dice and >48 pin package

17.2 WRITER PROGRAMMING PIN MAPPING

Programming Pin Information of SN8F22E80B Series											
Chip Name		**SN8F22E88BF**		**SN8F22E87BJ**		**SN8F22E84BS**		**SN8F22E83BJ**		**SN8F22E831BX**	
MP Writer Connector		**IC and JP3 48-pin text tool Pin Assignment**									
Pin Number	**Pin Name**	**IC Pin Number**	**IC Pin Name**	**IC Pin Number**	**IC Pin Name**	**IC Pin Number**	**IC Pin Name**	**IC Pin Number**	**IC Pin Name**	**IC Pin Number**	**IC Pin Name**
1	VDD	2, 30, 31	VDD	1, 27, 28	VDD	3, 23	VDD	23	VDD	2	VDD
2	GND	26	VSS	23	VSS	19	VSS	19	VSS	22	VSS
3	CLK	21	P5.4	18	P5.4	16	P5.4	16	P5.4	19	P5.4
4	CE	-	-	-		-					
5	PGM	20	P5.5	17	P5.5	15	P5.5	15	P5.5	18	P5.5
6	OE		-	-		-					
7	D1	-	-	-		-					
8	D0	-	-	-		-					
9	D3	-	-	-		-					
10	D2	-	-	-		-					
11	D5	-	-	-		-					
12	D4	-	-	-		-					
13	D7	-	-	-		-					
14	D6	-	-	-		-					
15	VDD	-	-	-		-					
16	VPP	-	-	-		-					
17	HLS	-	-	-		-					
18	RST	-	-	-		-					
19	-	-	-	-		-					
20	ALSB/PDB	11	P0.0	10	P0.0	12	P0.0	13	P0.0	15	P0.0

18 Marking Definition

18.1 INTRODUCTION

There are many different types in Sonix 8-bit MCU production line. This note listed the production definition of all 8-bit MCU for order or obtain information. This definition is only for Blank Flash ROM MCU.

18.2 MARKING INDETIFICATION SYSTEM

SN8 X PART No. X X X

Material	B = PB-Free Package G = Green Package
Temperature Range	- = 0℃ ~ 70℃
Shipping Package	W=Wafer, H=Dice P=DIP S=SOP X=SSOP
Device	22E88B
ROM Type	F = Flash
Title	SONiX 8-bit MCU Production

18.3 MARKING EXAMPLE

- **Wafer, Dice:**

Name	ROM Type	Device	Package	Temperature	Material
S8F22E80BW	FLASH	22E80B	Wafer	-40℃~85℃	-
SN8F22E80BH	FLASH	22E80B	Dice	-40℃~85℃	-

- **Green Package:**

Name	ROM Type	Device	Package	Temperature	Material
SN8F22E88BF	FLASH	22E80B	LQGP	-40℃~85℃	Green Package
SN8F22E87BJ	FLASH	22E80B	QFN	-40℃~85℃	Green Package
SN8F22E84BS	FLASH	22E80B	SOP	-40℃~85℃	Green Package
SN8F22E83BJ	FLASH	22E80B	QFN	-40℃~85℃	Green Package
SN8F22E831BX	FLASH	22E80B	SSOP	-40℃~85℃	Green Package

18.4 DATECODE SYSTEM

XX X X XXXXX

SONiX Internal Use

Day

1=01
2=02
. . . .
9=09
A=10
B=11
. . . .

Month

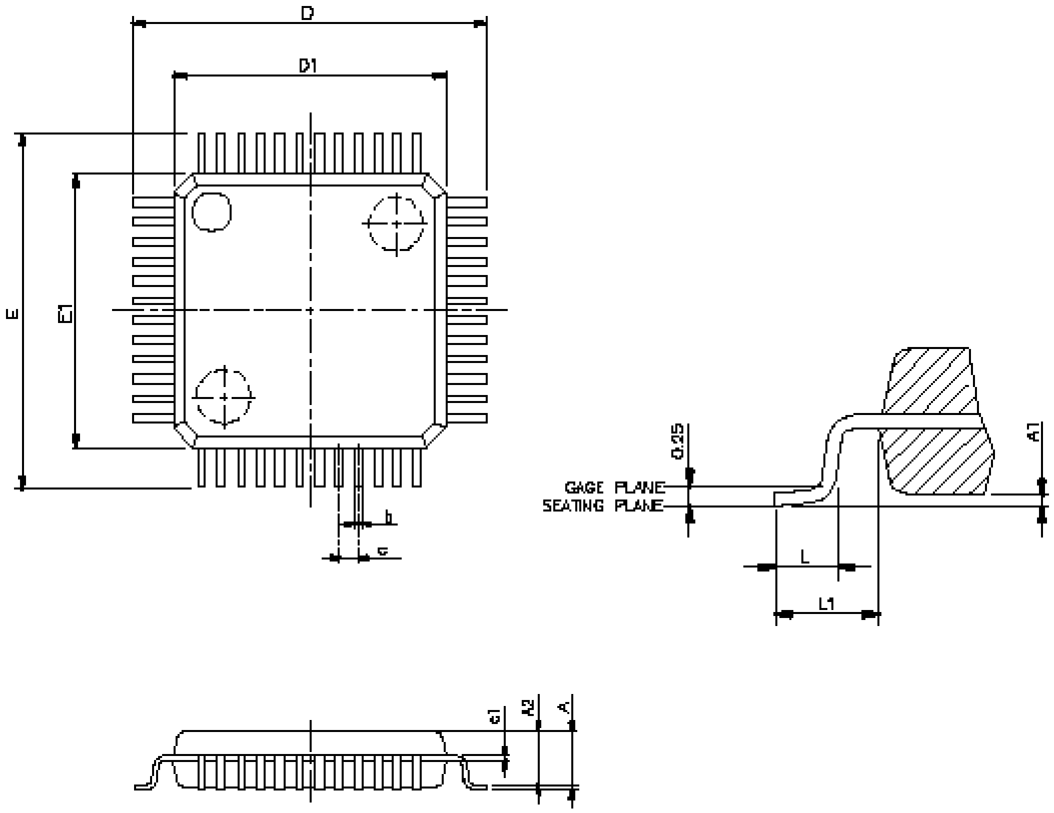
1=January
2=February
. . . .
9=September
A=October
B=November
C=December

Year

03= 2003
04= 2004
05= 2005
06= 2006
. . . .

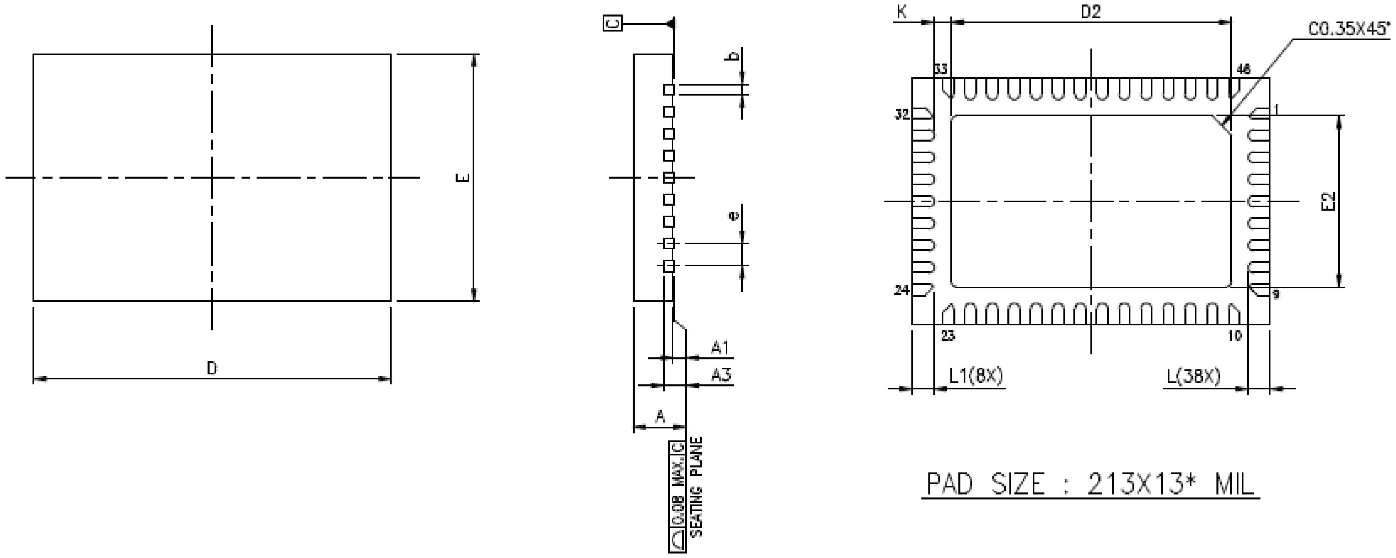
19 PACKAGE INFORMATION

19.1 LQFP 48 PIN



SYMBOLS	MIN	NOR	MAX
	(mm)		
A	-	-	1.6
A1	0.05	-	0.15
A2	1.35	-	1.45
c1	0.09	-	0.16
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
e	0.5 BSC		
B	0.17	-	0.27
L	0.45	-	0.75
L1	1 REF		

19.2 QFN 46 PIN



NOTES :

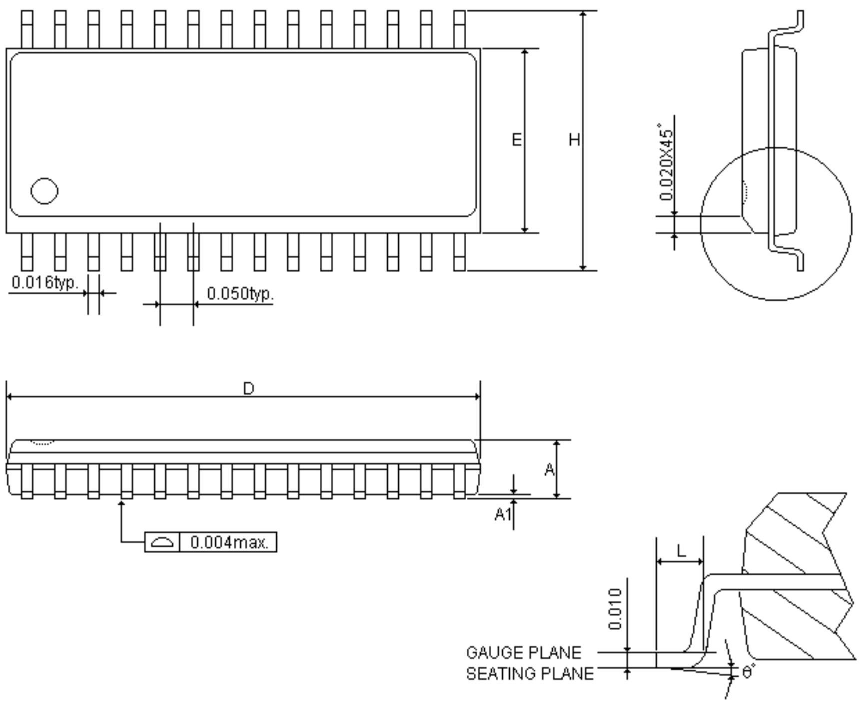
1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. DIMENSION b APPLIES TO METALLIZED TERMINAL AND IS MEASURED BETWEEN 0.15mm AND 0.30mm FROM THE TERMINAL TIP. IF THE TERMINAL HAS THE OPTIONAL RADIUS ON THE OTHER END OF THE TERMINAL, THE DIMENSION b SHOULD NOT BE MEASURED IN THAT RADIUS AREA.
3. BILATERAL COPLANARITY ZONE APPLIES TO THE EXPOSED HEAT SINK SLUG AS WELL AS THE TERMINALS.

	PACKAGE TYPE		
JEDEC OUTLINE	N/A		
SYMBOLS	MIN.	NOM.	MAX.
A	0.70	0.80	0.90
A1	0.00	0.02	0.05
A3	0.203 REF.		
b	0.15	0.20	0.25
D	6.50 BSC		
E	4.50 BSC		
e	0.40 BSC		
K	0.20	—	—

PAD SIZE	D2			E2			L			L1			LEAD FINISH		JEDEC CODE
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	Pure Tin	PPF	
⚠ 212X13* MIL	5.05	5.10	5.15	3.05	3.10	3.15	0.35	0.40	0.45	0.33	0.38	0.43	V	X	N/A

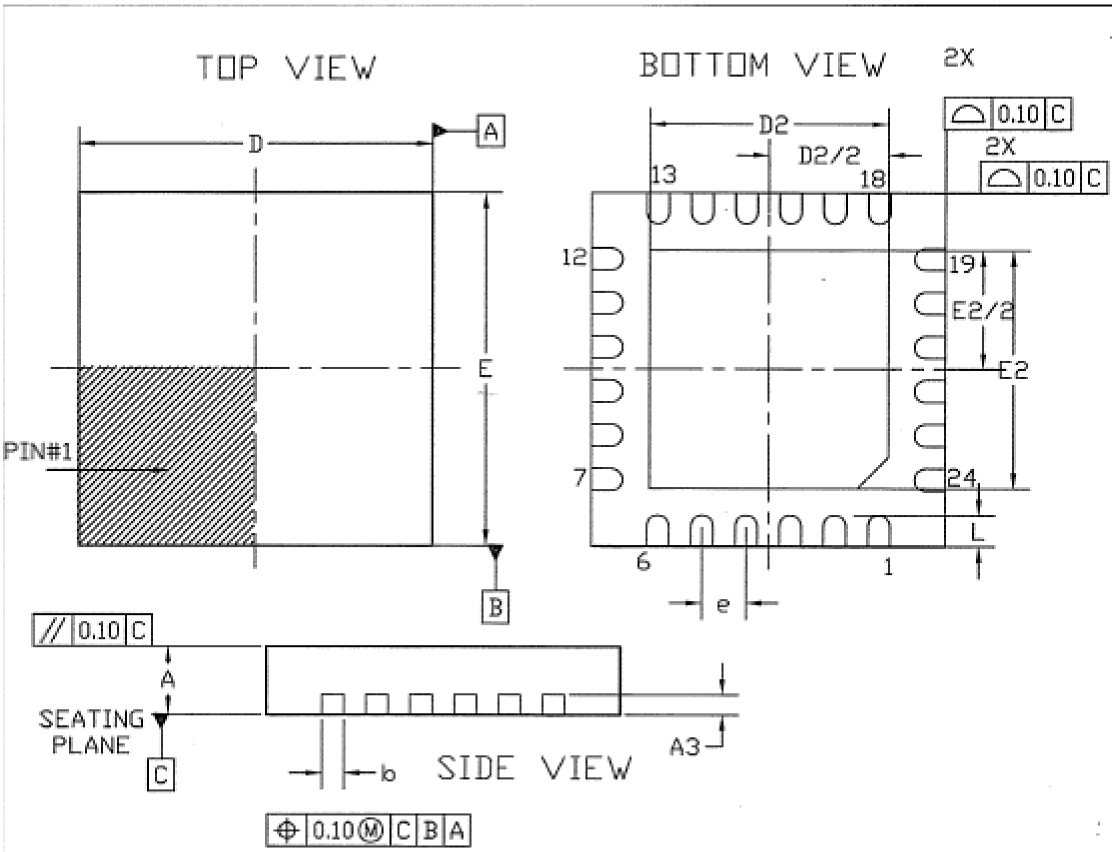
⚠ "*"表示汎用字元,此汎用字元可能被其它不同字元所取代,實際的字元請參照bonding diagram所示.
"*" is an universal character, which means maybe replaced by specific character, the actual character please refers to the bonding diagram.

19.3 SOP 28 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.697	0.705	0.713	17.704	17.907	18.110
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
θ°	0°	4°	8°	0°	4°	8°

19.4 QFN 24 PIN



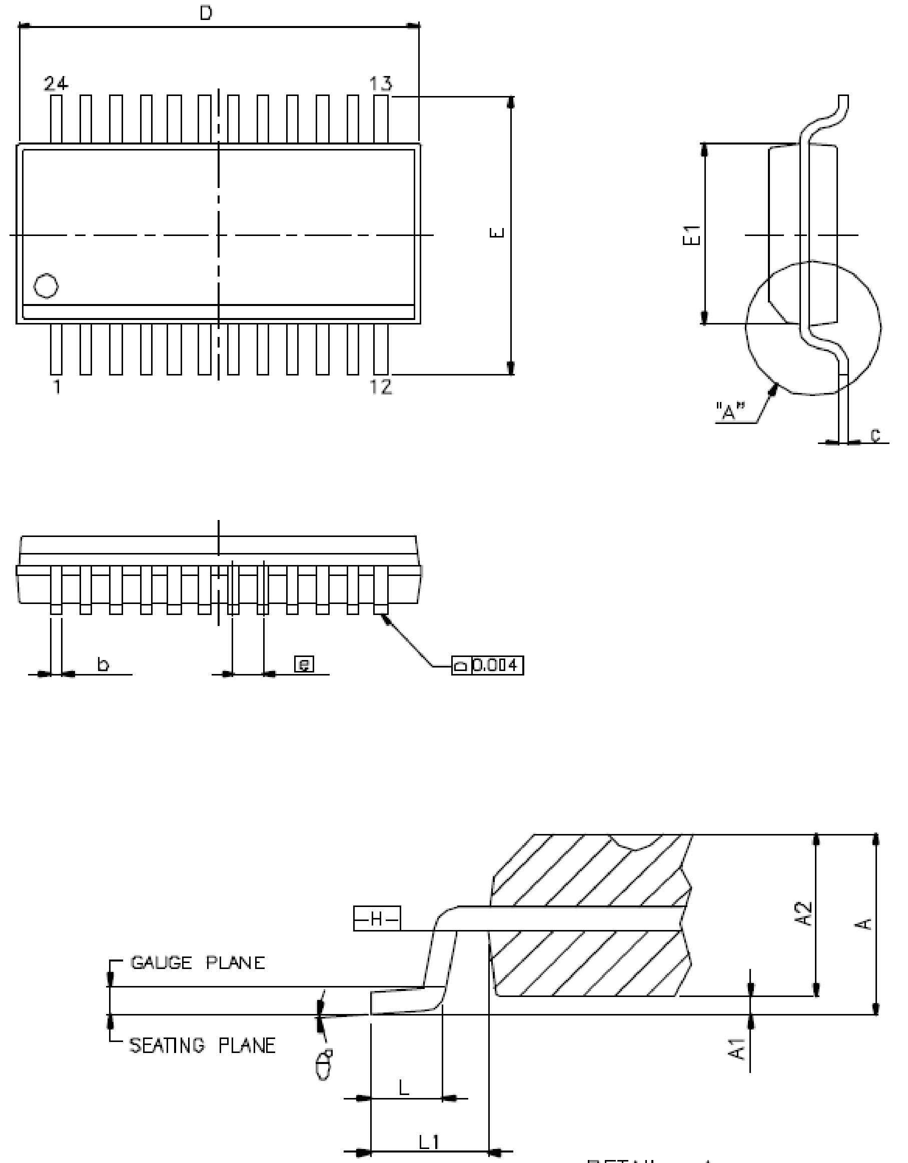
SYMBOL	COMMON					
	DIMENSIONS MILLIMETER			DIMENSIONS INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	SEE VARIATION					
A3	0.195	0.203	0.211	0.0077	0.008	0.0083
b	0.180	0.230	0.300	0.007	0.009	0.012
D	3.925	4.0	4.075	0.154	0.157	0.160
E	3.925	4.0	4.075	0.154	0.157	0.160
e	0.50 BSC			0.020 BSC		
L	SEE VARIATION					

VARIATION " A "					
DIMENSIONS MILLIMETER			DIMENSIONS INCH		
MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
0.70	0.75	0.80	0.027	0.029	0.031

PAD SIZE	VARIATION " L "						REF
	DIMENSIONS MILLIMETER			DIMENSIONS INCH			
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	
118x118	0.30	0.35	0.40	0.012	0.014	0.016	CUSTOMS A,B

PAD SIZE	D2/E2			D2/E2			REF
	DIMENSIONS MILLIMETER			DIMENSIONS INCH			
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	
118x118	2.50/2.50	2.65/2.65	2.80/2.80	0.098/0.098	0.104/0.104	0.110/0.110	VGGD-6, WGGD-6

19.5 SSOP 24 PIN



DETAIL : A

SYMBOLS	MIN.	NOM.	MAX.
A	0.053	0.064	0.069
A1	0.004	0.006	0.010
A2	–	–	0.059
D	0.337	0.341	0.344
E	0.228	0.236	0.244
E1	0.150	0.154	0.157
b	0.008	–	0.012
C	0.007	–	0.010
e	0.025 BASIC		
L	0.016	0.025	0.050
2 L1	0.041 BASIC		
θ°	0°	–	8°

UNIT : INCH

NOTES:

1.JEDEC OUTLINE : MO-137 AE

2.DIMENSION D DOES NOT INCLUDE MOLD PROTRUSIONS OR GATE BURRS. MOLD PROTRUSIONS AND GATE BURRS SHALL NOT EXCEED 0.006" PER SIDE. DIMENSION E1 DOES NOT INCLUDE INTERLEAD MOLD PROTRUSIONS. INTERLEAD MOLD PROTRUSIONS SHALL NOT EXCEED 0.010" PER SIDE.

3.DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION/INTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.004" TOTAL IN EXCESS OF b DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR INTRUSION SHALL NOT REDUCE DIMENSION b BY MORE THAN 0.002" AT LEAST.

Corporate Headquarters:
10F-1, No.36, Taiyuan Street, Chupei City, Hsinchu, Taiwan
TEL :(886)3-5600-888 FAX :(886)3-5600-889
Taipei Sales Office:
15F-2, No.171 Song Ted Road, Taipei, Taiwan
TEL:(886)2-2759-1980 FAX:(886)2-2759-8180
mkt@sonix.com.tw | sales@sonix.com.tw
Hong Kong Sales Office:
Unit 2603, 26/F CCT Telecom Building, No. 11 Wo Shing Street,
Fo Tan, New Territories, Hong Kong
TEL:(852)2723-8086 FAX:(852)2723-9179
hk@sonix.com.tw
Shenzhen Contact Office:
High Tech Industrial Park, Shenzhen, China
TEL:(86)755-2671-9666 FAX:(86)755-2671-9786
mkt@sonix.com.tw | sales@sonix.com.tw
Sonix Japan Office:
Kobayashi bldg. 2F, 4-8-27,Kudanminami, Chiyodaku,Tokyo,
102-0074, Japan
TEL:(81)3-6272-6070 FAX:(81)3-6272-6165
jpsales@sonix.com.tw
FAE Support via Email:
8-bit Microcontroller Products: sa1fae@sonix.com.tw
All Products: fae@sonix.com.tw