SN8P2735

USER'S MANUAL

Version 1.1

SN8P2735
SN8P2734
SN8P2733
SN8P2732

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 0.1	Dec. 2008	First issue.
VER 0.2	Mar. 2009	Major Modification: 1. IHRC frequency is modified to 16MHz and related sections. 2. Fcpu is modified to Fhosc/2~Fhosc/16 and related sections. 3. OTP programming pin chapter.
VER 1.0	Sep. 2009	1. Add SN8P2734/2733/2732 items, programming pin table and package information. 2. Fix typing error. 3. Modify IDE version to M2IDE_V120.
VER 1.1	Apr. 2011	1. Update PWM1/2 clock source description are Fhosc, Fcpu. 2. Modify SN8P2734/33/32 package type at “PIN ASSIGNMENT”. 3. Add SN8P2734/33/32 to ‘ELECTRICAL CHARACTERISTIC” description. 4. Add “DEVELOPMENT TOOL” description. 5. Modify “MARKING DEFINITION” contents

Table of Content

1 PRODUCT OVERVIEW

1.1 FEATURES

- **Memory configuration**
  ROM size: 6K * 16 bits.
  RAM size: 256 * 8 bits.

- **8 levels stack buffer.**

- **10 interrupt sources**
  7 internal interrupts: T0, T1, TC0, ADC, CM0, CM1, CM2
  3 external interrupt: INT0, INT1, INT2

- **I/O pin configuration**
  Bi-directional: P0, P1, P4, P5.
  Wakeup: P0, P1 level change.
  Pull-up resisters: P0, P1, P4, P5.
  Op-amp/Comparator pins: P1.0~P1.7, P5.0.
  ADC input pin: P4.0~P4.7.

- **3-Level LVD.**
  Reset system and power monitor.

- **Fcpu (Instruction cycle)**
  Fcpu = Fosc/1, Fosc/2, Fosc/4, Fosc/8, Fosc/16.

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

- **One 8-bit basic timer. (T0).**
- **One 8-bit timer with external event counter, Buzzer and PWM. (TC0).**
- **One 16-bit capture timer. (T1).**
- **6-channel 8/10/12 bits PWM.**
- **1-channel 8/10/12 bits PWM with dead-band and inverse output.**
- **8-channel 12-bit SAR ADC.**
- **3-set rail-to-rail OP-amp.**
- **3-set rail-to-rail comparator.**
- **On chip watchdog timer and clock source is Internal low clock RC type (16KHz @3V, 32KHz @5V).**

- **Four system clocks**
  External high clock: RC type up to 10 MHz
  External high clock: Crystal type up to 16 MHz
  Internal high clock: RC type 16MHz
  Internal low clock: RC type 16KHz(3V), 32KHz(5V)

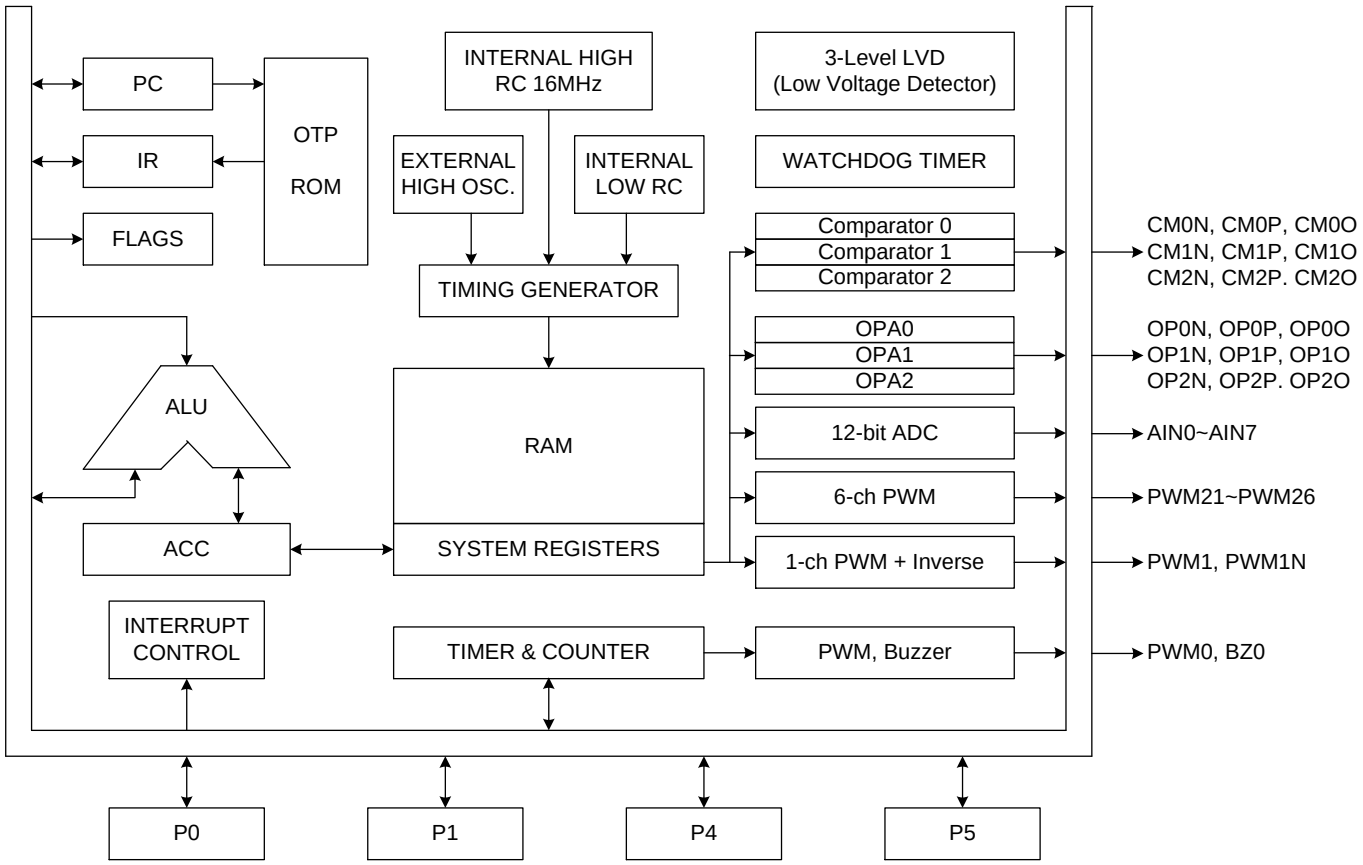
- **Four operating modes**
  Normal mode: Both high and low clock active
  Slow mode: Low clock only
  Sleep mode: Both high and low clock stop
  Green mode: Periodical wakeup by timer

- **Package (Chip form support)**
  PDIP 32 pin
  LQFP 32 pin

☞ **Features Selection Table**

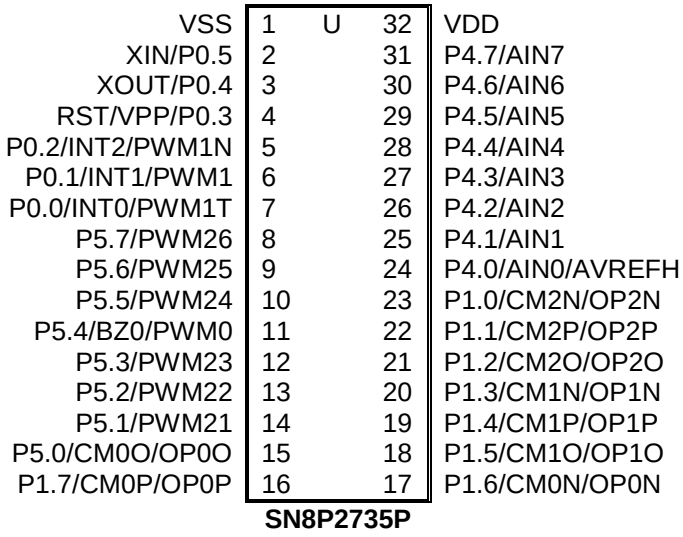
CHIP	ROM	RAM	Stack	Timer			I/O	PWM		ADC	OP-amp	Comparator	Wake-up Pin No.	Package
				T0	TC0	T1		8Bit	8~12 Bit					
SN8P2735	6K*16	256	8	V	V	V	30	1	7	8-ch	3	3	14	PDIP32/ LQFP32
SN8P2734	6K*16	256	8	V	V	V	26	1	3	8-ch	3	3	14	SKDIP28/ SOP28
SN8P2733	6K*16	256	8	V	V	V	22	1	2	8-ch	2	2	11	SKDIP24/ SOP24
SN8P2732	6K*16	256	8	V	V	V	18	1	2	7-ch	1	1	8	DIP20/ SOP20

1.2 SYSTEM BLOCK DIAGRAM

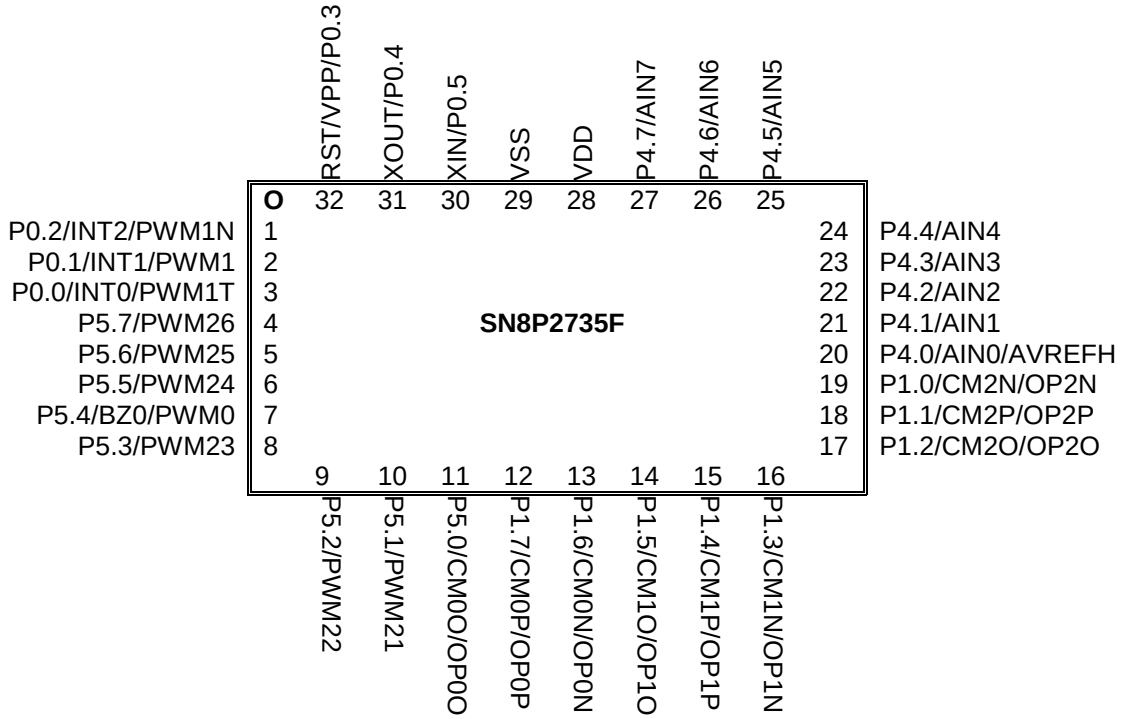


1.3 PIN ASSIGNMENT

SN8P2735P (PDIP 32 pins)



SN8P2735F (LQFP 32 pins)



SN8P2734P (SK-DIP 28 pins)
SN8P2734S (SOP 28 pins)

VSS	1	U	28	VDD
XIN/P0.5	2		27	P4.7/AIN7
XOUT/P0.4	3		26	P4.6/AIN6
RST/VPP/P0.3	4		25	P4.5/AIN5
P0.2/INT2/PWM1N	5		24	P4.4/AIN4
P0.1/INT1/PWM1	6		23	P4.3/AIN3
P0.0/INT0/PWM1T	7		22	P4.2/AIN2
P5.4/BZ0/PWM0	8		21	P4.1/AIN1
P5.2/PWM22	9		20	P4.0/AIN0/AVREFH
P5.1/PWM21	10		19	P1.0/CM2N/OP2N
P5.0/CM0O/OP0O	11		18	P1.1/CM2P/OP2P
P1.7/CM0P/OP0P	12		17	P1.2/CM2O/OP2O
P1.6/CM0N/OP0N	13		16	P1.3/CM1N/OP1N
P1.5/CM1O/OP1O	14		15	P1.4/CM1P/OP1P

SN8P2734

SN8P2733K (SK-DIP 24 pins)
SN8P2733S (SOP 24 pins)

VSS	1	U	24	VDD
XIN/P0.5	2		23	P4.7/AIN7
XOUT/P0.4	3		22	P4.6/AIN6
RST/VPP/P0.3	4		21	P4.5/AIN5
P0.2/INT2/PWM1N	5		20	P4.4/AIN4
P0.1/INT1/PWM1	6		18	P4.3/AIN3
P0.0/INT0/PWM1T	7		19	P4.2/AIN2
P5.4/BZ0/PWM0	8		17	P4.1/AIN1
P5.1/PWM21	9		16	P4.0/AIN0/AVREFH
P5.0/CM0O/OP0O	10		15	P1.3/CM1N/OP1N
P1.7/CM0P/OP0P	11		14	P1.4/CM1P/OP1P
P1.6/CM0N/OP0N	12		13	P1.5/CM1O/OP1O

SN8P2733

SN8P2732P (PDIP 20 pins)
SN8P2732S (SOP 20 pins)

VSS	1	U	20	VDD
XIN/P0.5	2		19	P4.6/AIN6
XOUT/P0.4	3		18	P4.5/AIN5
RST/VPP/P0.3	4		17	P4.4/AIN4
P0.2/INT2/PWM1N	5		16	P4.3/AIN3
P0.1/INT1/PWM1	6		15	P4.2/AIN2
P0.0/INT0/PWM1T	7		14	P4.1/AIN1
P5.4/BZ0/PWM0	8		13	P4.0/AIN0/AVREFH
P5.1/PWM21	9		12	P1.6/CM0N/OP0N
P5.0/CM0O/OP0O	10		11	P1.7/CM0P/OP0P

SN8P2732

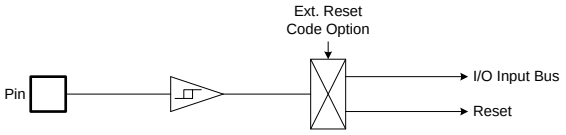
1.4 PIN DESCRIPTIONS

PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital and analog circuit.
P0.3/RST/VPP	I, P	RST: System external reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”.
		VPP: OTP 12.3V power input pin in programming mode.
		P0.3: Input only pin with Schmitt trigger structure and no pull-up resistor.
XIN/P0.5	I/O	XIN: Oscillator input pin while external oscillator enable (crystal and RC).
		P0.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
XOUT/P0.4	I/O	XOUT: Oscillator output pin while external crystal enable.
		P0.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
P0.0/INT0/PWM1T	I/O	P0.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		INT0: External interrupt 0 input pin.
		TC0 event counter input pin.
		PWM1T: PWM1 external trigger source.
P0.1/INT1/PWM1	I/O	P0.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		INT1: External interrupt 1 input pin.
		PWM1: 4-phase speed-up PWM output pin.
P0.2/INT2/PWM1N	I/O	P0.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		INT2: External interrupt 2 input pin.
		PWM1N: PWM1 negative output pin.
P1.0/CM2N/OP2N	I/O	P1.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM2N: The negative input pin of comparator.
		OP2N: The negative input pin of OP amp.
P1.1/CM2P/OP2P	I/O	P1.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM2P: The positive input pin of comparator.
		OP2P: The positive input pin of OP amp.
P1.2/CM2O/OP2O	I/O	P1.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM2O: The output pin of comparator.
		OP2O: The output pin of OP amp.
P1.3/CM1N/OP1N	I/O	P1.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM1N: The negative input pin of comparator.
		OP1N: The negative input pin of OP amp.
P1.4/CM1P/OP1P	I/O	P1.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM1P: The positive input pin of comparator.
		OP1P: The positive input pin of OP amp.
P1.5/CM1O/OP1O	I/O	P1.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM1O: The output pin of comparator.
		OP1O: The output pin of OP amp.
P1.6/CM0N/OP0N	I/O	P1.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM0N: The negative input pin of comparator.
		OP0N: The negative input pin of OP amp.
P1.7/CM0P/OP0P	I/O	P1.7: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM0P: The positive input pin of comparator.
		OP0P: The positive input pin of OP amp.
P5.0/CM0O/OP0O	I/O	P5.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM0O: The output pin of comparator.
		OP0O: The output pin of OP amp.
P5.1/PWM21	I/O	P5.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		PWM21: Channel 1 of 4-phase speed-up PWM2 output pin.
P5.2/ PWM22	I/O	P5.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		PWM22: Channel 2 of 4-phase speed-up PWM2 output pin.
P5.3/ PWM23	I/O	P5.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		PWM23: Channel 3 of 4-phase speed-up PWM2 output pin.
P5.4/BZ0/PWM0	I/O	P5.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.

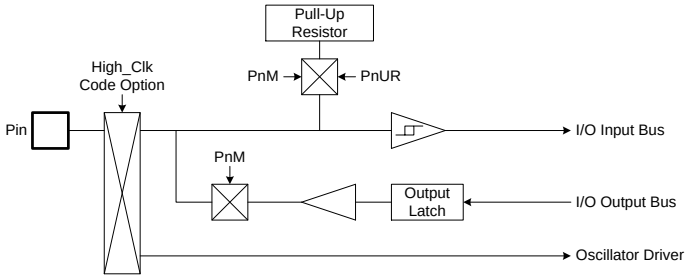
		BZ0: Programmable buzzer output pin from TC0/2 signal.
		PWM0: Programmable PWM output pin from TC0.
P5.5/PWM24	I/O	P5.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		PWM24: Channel 4 of 4-phase speed-up PWM2 output pin.
P5.6/PWM25	I/O	P5.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		PWM25: Channel 5 of 4-phase speed-up PWM2 output pin.
P5.7/PWM26	I/O	P5.7: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		PWM26: Channel 6 of 4-phase speed-up PWM2 output pin.
P4.0/AIN0/AVREFH	I/O	P4.0: Bi-direction pin. No Schmitt trigger structure. Built-in pull-up resisters.
		AIN0: ADC analog input pin.
		AVREFH: ADC reference high voltage input pin.
P4.1/AIN1	I/O	P4.1: Bi-direction pin. No Schmitt trigger structure. Built-in pull-up resisters.
		AIN1: ADC analog input pin.
P4.2/AIN2	I/O	P4.2: Bi-direction pin. No Schmitt trigger structure. Built-in pull-up resisters.
		AIN2: ADC analog input pin.
P4.3/AIN3	I/O	P4.3: Bi-direction pin. No Schmitt trigger structure. Built-in pull-up resisters.
		AIN3: ADC analog input pin.
P4.4/AIN4	I/O	P4.4: Bi-direction pin. No Schmitt trigger structure. Built-in pull-up resisters.
		AIN4: ADC analog input pin.
P4.5/AIN5	I/O	P4.5: Bi-direction pin. No Schmitt trigger structure. Built-in pull-up resisters.
		AIN5: ADC analog input pin.
P4.6/AIN6	I/O	P4.6: Bi-direction pin. No Schmitt trigger structure. Built-in pull-up resisters.
		AIN6: ADC analog input pin.
P4.7/AIN7	I/O	P4.7: Bi-direction pin. No Schmitt trigger structure. Built-in pull-up resisters.
		AIN7: ADC analog input pin. Refer to ADC section.

1.5 PIN CIRCUIT DIAGRAMS

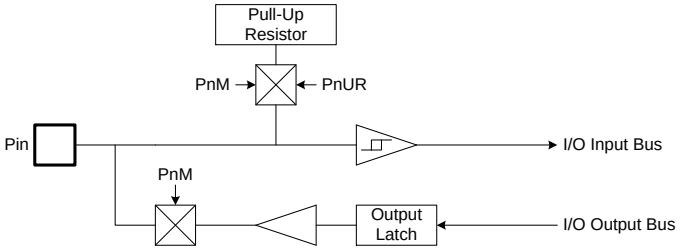
- **P0.3 structure:**



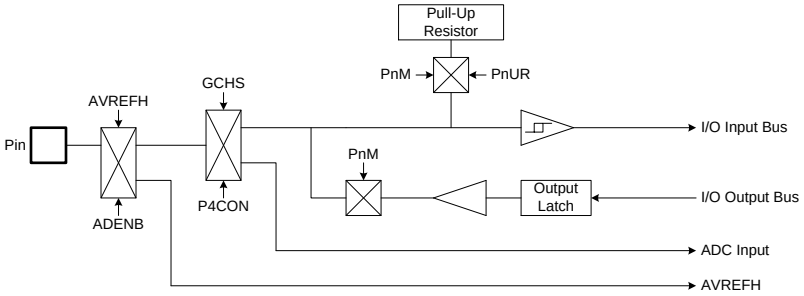
- **P0.4, P0.5 structure:**



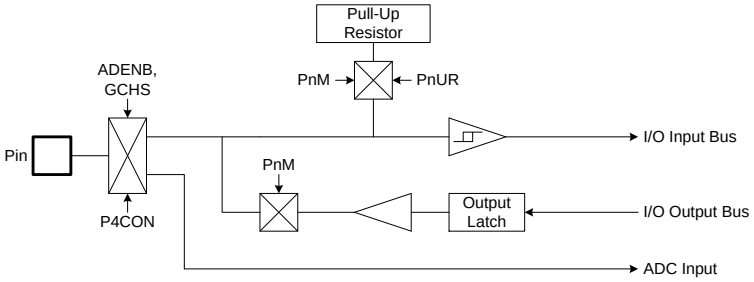
- **P0.0~P0.2, P5.1~P5.7 structure:**



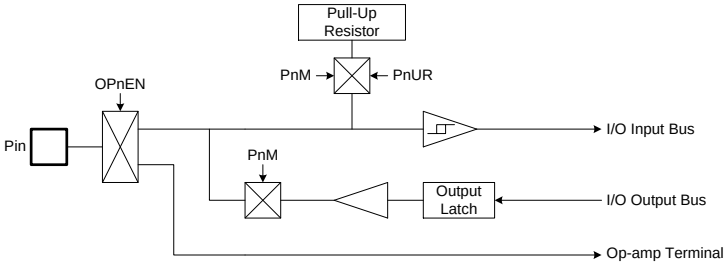
- **P4.0 structure:**



- **P4.1~P4.7 structure:**

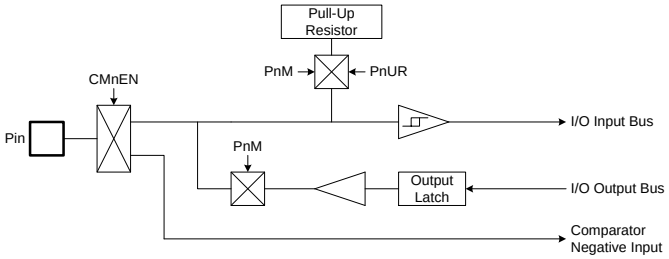


- **P1.0~P1.7, P5.0 OP-amp shared pins structure:**

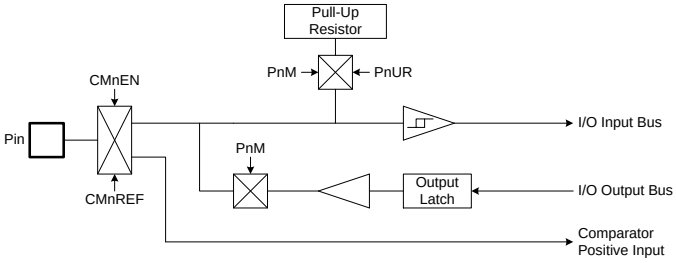


- **P1.0~P1.7, P5.0 Comparator shared pins structure:**

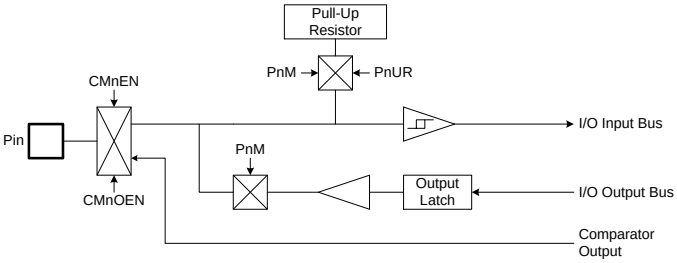
Comparator Negative Pin:



Comparator Positive Pin:



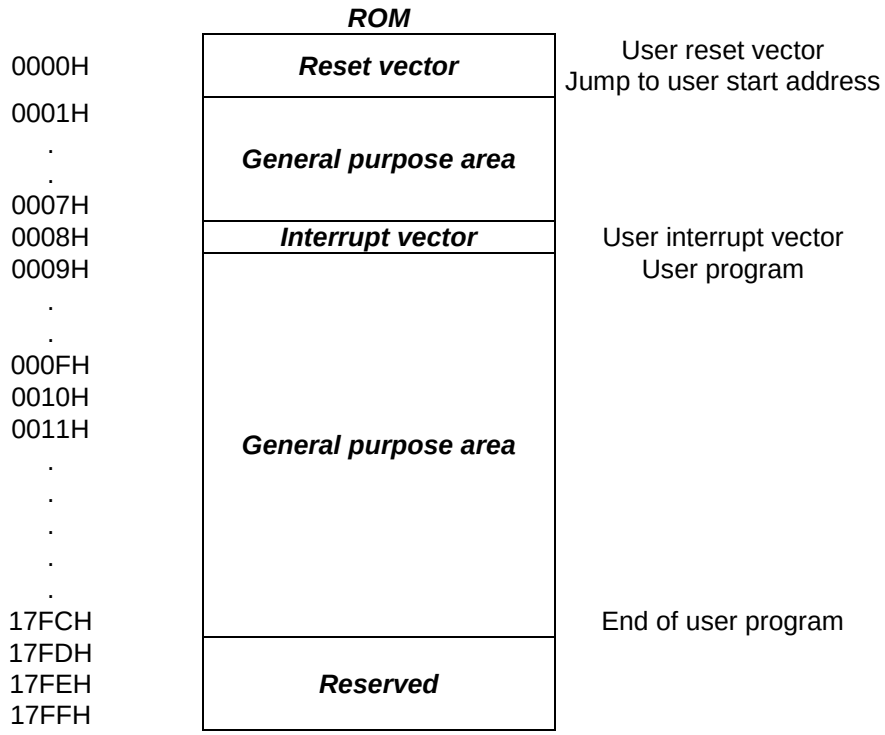
Comparator Output Pin:



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (ROM)

☞ ***6K words ROM***



The ROM includes Reset vector, Interrupt vector, General purpose area and Reserved area. The Reset vector is program beginning address. The Interrupt vector is the head of interrupt service routine when any interrupt occurring. The General purpose area is main program area including main loop, sub-routines and data table.

2.1.1 RESET VECTOR (0000H)

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

- ☞ **Power On Reset (NT0=1, NPD=0).**
- ☞ **Watchdog Reset (NT0=0, NPD=0).**
- ☞ **External Reset (NT0=1, NPD=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 NT0, NPD 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     10H
START:                              ; 0010H, The head of user program.
                …                   ; User program
                …


                ENDP                ; End of program
```

2.1.2 INTERRUPT VECTOR (0008H)

A 1-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 of program memory to execute the vectored interrupt. Users have to define the interrupt vector. The following example shows the way to define the interrupt vector in the program memory.

✸ ***Note: "PUSH", "POP" instructions save and load ACC/PFLAG without (NT0, NPD). PUSH/POP buffer is a unique buffer and only one level.***

➢ **Example: Defining Interrupt Vector. The interrupt service routine is following ORG 8.**

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

                ORG     8               ; Interrupt vector.
                PUSH                    ; Save ACC and PFLAG register to buffers.
                …
                …
                POP                     ; Load ACC and PFLAG register from buffers.
                RETI                    ; End of interrupt service routine
                …

START:                                  ; The head of user program.
                …                       ; User program
                …
                JMP     START           ; End of user program
                …

                ENDP                    ; End of program
```

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

```
.CODE
        ORG     0           ; 0000H
        JMP     START       ; Jump to user program address.
        …
        ORG     8           ; Interrupt vector.
        JMP     MY_IRQ      ; 0008H, Jump to interrupt service routine address.

        ORG     10H
START:                      ; 0010H, The head of user program.
        …                   ; User program.
        …
        …
        JMP     START       ; End of user program.
        …
MY_IRQ:                     ;The head of interrupt service routine.
        PUSH                ; Save ACC and PFLAG register to buffers.
        …
        …
        POP                 ; Load ACC and PFLAG 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 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 are 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
                B0BSET  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)

☞ ***256 X 8-bit RAM***

	Address	RAM Location	
BANK 0	000h “ “ “ 07Fh	***General Purpose Area***	RAM Bank 0
	080h “ “ “ 0FFh	***System Register***	080h~0FFh of Bank 0 store system registers (128 bytes). End of Bank 0
BANK 1	100h “ “ “ 17Fh	***General Purpose Area***	RAM Bank 1 End of Bank 1

The 256-byte general purpose RAM is separated into Bank 0 and Bank 1. Accessing the two banks' RAM is controlled by “RBANK” register. When RBANK = 0, the program controls Bank 0 RAM directly. When RBANK = 1, the program controls Bank 1 RAM directly. Under one bank condition and need to access the other bank RAM, setup the RBANK register is necessary. Sonix provides “Bank 0” type instructions (e.g. b0mov, b0add, b0bts1, b0bset…) to control Bank 0 RAM in non-zero RAM bank condition directly.

➢ **Example: Access Bank 0 RAM in Bank 1 condition. Move Bank 0 RAM (WK00) value to Bank 1 RAM (WK01).**

```
; Bank 1 (RBANK = 1)
        B0MOV   A, WK00     ; Use Bank 0 type instruction to access Bank 0 RAM.
        MOV     WK01,A
```

✸ ***Note: For multi-bank RAM program, it is not easy to control RAM Bank selection. Users have to take care the RBANK condition very carefully, especially for interrupt service routine. The system won't switch RAM bank to Bank 0, so these controls must be through program. It is a good to use Bank 0 type instruction to process the situations.***

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	-	PFLAG	RBANK	-	-	-	-	-	-	-	-
9	PW1NM	-	-	PW1M	PW2M	PW2CHS	PW1RH	PW1RL	PW2RH	PW2RL	-	-	CM0M	CM1M	CM2M	OPM
A	T1M	T1CL	T1CH	-	-	-	-	-	-	-	-	-	-	-	P4CON	-
B	-	ADM	ADB	ADR	ADT	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	-	-	P4M	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	-	-	P4	P5	-	-	T0M	T0C	TC0M	TC0C	-	-	-	STKP
E	P0UR	P1UR	-	-	P4UR	P5UR	@HL	@YZ	-	-	-	-	-	-	-	-
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.
RBANK = RAM bank select register.
PW1M = PWM 1 mode register.
PW2CHS = PWM 2 output channel select register.
PW2RH, L = PWM 2 reload buffers.
CM1M = Comparator 1 mode register.
OPM = OP amp 0~2 mode register.
T1CH, L = T1 counting registers.
ADM = ADC mode register.
ADR = ADC resolution select register.
PEDGE = P0.0, P0.1, P0.2 edge direction register.
INTEN = Interrupt enable register.
PnM = Port n input/output mode register.
PnUR = Port n pull-up resister control register.
PCH, PCL = Program counter.
T0C = T0 counting register.
TC0C = TC0 counting register.
@HL = RAM HL indirect addressing index pointer.
STKP = Stack pointer buffer.

Y, Z = Working, @YZ and ROM addressing register.
PFLAG = Special flag register.
PW1NM = Inverse PWM 1 mode register.
PW2M = PWM 2 mode register.
PW1RH, L = PWM 1 reload buffers.
CM0M = Comparator 0 mode register.
CM2M = Comparator 2 mode register.
T1M = T1 mode register.
P4CON = P4 configuration register.
ADB = ADC data buffer.
ADT = ADC offset calibration register.
INTRQ = Interrupt request register.
WDTR = Watchdog timer clear register.
Pn = Port n data buffer.
OSCM = Oscillator mode register.
T0M = T0 mode register.
TC0M = TC0 mode register.
TC0R = TC0 auto-reload data buffer.
@YZ = RAM YZ indirect addressing index pointer.
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
086H	NT0	NPD	LVD36	LVD24		C	DC	Z	R/W	PFLAG
087H								RBNKS0	R/W	RBANK
090H	PW1NEN	PW1D2	PW1D1	PW1D0	PW1DEN	PW1NV			R/W	PW1NM
093H	PW1EN	PW1rate2	PW1rate1	PW1rate0P	PW1CKS	PW1LN1	PW1LN0	PW1S	R/W	PW1M
094H	PW2EN	PW2rate2	PW2rate1	PW2rate0	PW2CKS	PW2LN1	PW2LN0		R/W	PW2M
095H			PW2CH6	PW2CH5	PW2CH4	PW2CH3	PW2CH2	PW2CH1	R/W	PW2CHS
096H	PW1GS	PW1GEN	PW1GD		PW1R11	PW1R10	PW1R9	PW1R8	R/W	PW1RH
097H	PW1R7	PW1R6	PW1R5	PW1R4	PW1R3	PW1R2	PW1R1	PW1R0	W	PW1RL
098H					PW2R11	PW2R10	PW2R9	PW2R8	R/W	PW2RH
099H	PW2R7	PW2R6	PW2R5	PW2R4	PW2R3	PW2R2	PW2R1	PW2R0	W	PW2RL
09CH	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CM0G1	CM0G0	R/W	CM0M
09DH	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	CM1G1	CM1G0	R/W	CM1M
09EH	CM2EN	CM2IEN	CM2IRQ	CM2OEN	CM2REF	CM2OUT	CM2G1	CM2G0	R/W	CM2M
09FH						OP2EN	OP1EN	OP0EN	R/W	OPM
0A0H	T1ENB	T1rate2	T1rate1	T1rate0	CPTCKS	CPTStart	CPTG1	CPTG0	R/W	T1M
0A1H	T1CL7	T1CL6	T1CL5	T1CL4	T1CL3	T1CL2	T1CL1	T1CL0	R/W	T1CL
0A2H	T1CH7	T1CH6	T1CH5	T1CH4	T1CH3	T1CH2	T1CH1	T1CH0	R/W	T1CH
0AEH	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	W	P4CON
0B1H	ADENB	ADS	EOC	GCHS	AVREFH	CHS2	CHS1	CHS0	R/W	ADM
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0B3H		ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0B4H	ADTS1	ADTS0		ADT4	ADT3	ADT2	ADT1	ADT0	R/W	ADT
0B8H			P05M	P04M	-	P02M	P01M	P00M	R/W	P0M
0BFH			P02G1	P02G0	P01G1	P01G0	P00G1	P00G0	R/W	PEDGE
0C0H	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C5H	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H	ADCIRQ	T1IRQ	TC0IRQ	T0IRQ		P02IRQ	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	T1IEN	TC0IEN	T0IEN		P02IEN	P01IEN	P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH				PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H			P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D4H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4
0D5H	P57	P56	P55	P54	P53	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0				T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H			P05R	P04R	-	P02R	P01R	P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H				S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H				S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H				S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H				S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H				S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L

0FBH				S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH				S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH				S0PC12	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
; or
                B0MOV       A, BUF
```

The system doesn't store ACC and PFLAG value when interrupt executed. ACC and PFLAG data must be saved to other data memories. "PUSH", "POP" save and load ACC, PFLAG data into buffers.

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

```
INT_SERVICE:
                PUSH                ; Save ACC and PFLAG to buffers.
                …
                …
                POP                 ; Load ACC and PFLAG from buffers.

                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. NT0, NPD 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, LVD36 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	NT0	NPD	LVD36	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

Bit [7:6] **NT0, NPD:** Reset status flag.

NT0	NPD	Reset Status
0	0	Watch-dog time out
0	1	Reserved
1	0	Reset by LVD
1	1	Reset by external Reset Pin

Bit 5 **LVD36:** LVD 3.6V operating flag and only support LVD code option is LVD_H.
0 = Inactive (VDD > 3.6V).
1 = Active (VDD ≦ 3.6V).

Bit 4 **LVD24:** LVD 2.4V operating flag and only support LVD code option is LVD_M.
0 = Inactive (VDD > 2.4V).
1 = Active (VDD ≦ 2.4V).

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.

✸ ***Note: Refer to instruction set table for detailed information of C, DC and Z flags.***

2.2.4 PROGRAM COUNTER

The program counter (PC) is a 13-bit binary counter separated into the high-byte 5 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 12.

	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	-	-	-	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
	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.***

2.3 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 0x80~0x87 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.
```

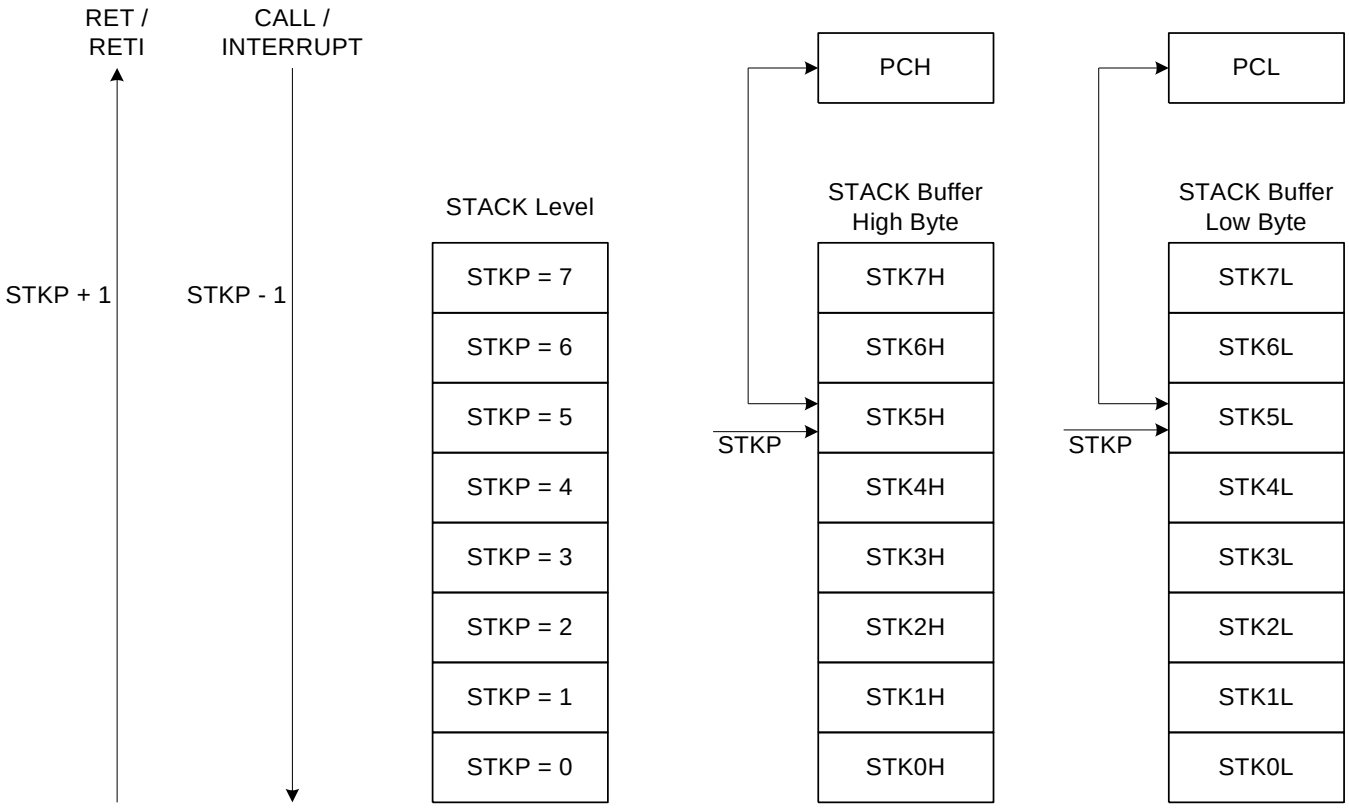
- **Example: Indirectly addressing mode with @YZ register**

```
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 REGISTERS

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.

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.

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

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

Bit 7 **GIE:** Global interrupt control bit.
0 = Disable.
1 = Enable. Please refer to the interrupt chapter.

➢ **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
```

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	-	SnPC12	SnPC11	SnPC10	SnPC9	SnPC8
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	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)

2.4.3 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*		*Description*
	STKPB2	**STKPB1**	**STKPB0**	**High Byte**	**Low Byte**	
0	1	1	1	Free	Free	-
1	1	1	0	STK0H	STK0L	-
2	1	0	1	STK1H	STK1L	-
3	1	0	0	STK2H	STK2L	-
4	0	1	1	STK3H	STK3L	-
5	0	1	0	STK4H	STK4L	-
6	0	0	1	STK5H	STK5L	-
7	0	0	0	STK6H	STK6L	-
8	1	1	1	STK7H	STK7L	-
> 8	1	1	0	-	-	**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*		*Description*
	STKPB2	**STKPB1**	**STKPB0**	**High Byte**	**Low Byte**	
8	1	1	1	STK7H	STK7L	-
7	0	0	0	STK6H	STK6L	-
6	0	0	1	STK5H	STK5L	-
5	0	1	0	STK4H	STK4L	-
4	0	1	1	STK3H	STK3L	-
3	1	0	0	STK2H	STK2L	-
2	1	0	1	STK1H	STK1L	-
1	1	1	0	STK0H	STK0L	-
0	1	1	1	Free	Free	-

2.5 CODE OPTION TABLE

The code option is the system hardware configurations including oscillator type, noise filter option, watchdog timer operation, LVD option, reset pin option and OTP 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.
	IHRC_RTC	High speed internal 16MHz RC. XIN/XOUT pins are connected to external 32768Hz crystal.
	RC	Low cost RC for external high clock oscillator. XIN pin is connected to RC oscillator. XOUT pin is bi-direction GPIO mode.
	32K X'tal	Low frequency, power saving crystal (e.g. 32.768KHz) for external high clock oscillator.
	12M X'tal	High speed crystal /resonator (e.g. 12MHz) for external high clock oscillator.
	4M X'tal	Standard crystal /resonator (e.g. 4M) for external high clock oscillator.
Fcpu	Fhosc/1	Instruction cycle is 1 oscillator clocks. **Notice: In Fosc/1, Noise Filter must be disabled.**
	Fhosc/2	Instruction cycle is 2 oscillator clocks. **Notice: In Fosc/2, Noise Filter must be disabled.**
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
	Fhosc/16	Instruction cycle is 16 oscillator clocks.
Noise_Filter	Enable	Enable Noise Filter and the Fcpu is Fosc/4~Fosc/16.
	Disable	Disable Noise Filter and the Fcpu is Fosc/2~Fosc/16.
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.
	P03	Enable P0.3 input only without pull-up resister.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
LVD	LVD_L	LVD will reset chip if VDD is below 2.0V
	LVD_M	LVD will reset chip if VDD is below 2.0V Enable LVD24 bit of PFLAG register for 2.4V low voltage indicator.
	LVD_H	LVD will reset chip if VDD is below 2.4V Enable LVD36 bit of PFLAG register for 3.6V low voltage indicator.
	LVD_MAX	LVD will reset chip if VDD is below 3.6V

2.5.1 Fcpu code option

Fcpu means instruction cycle of normal mode (high clock). In slow mode, the system clock source is internal low speed RC oscillator. The Fcpu of slow mode isn't controlled by Fcpu code option and fixed Flosc/4 (16KHz/4 @3V, 32KHz/4 @5V).

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.
- **P03:** Set reset pin to general input only pin (P0.3). The external reset function is disabled and the pin is input pin.

2.5.3 Security code option

Security code option is OTP 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, Fcpu is limited below Fhosc/1 and Fhosc/2. The fast Fcpu rate is Fhosc/4. If noise filter disable, the Fhosc/1 and Fhosc/2 options are released. 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)

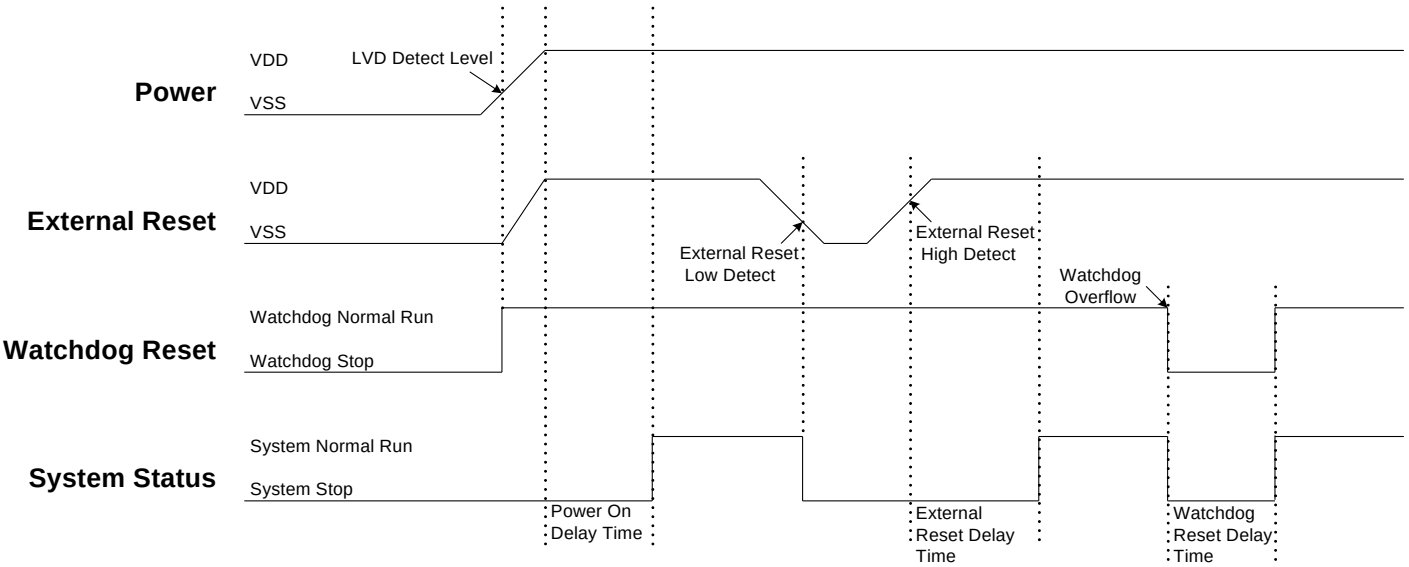
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status 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	NT0	NPD	LVD36	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

Bit [7:6] **NT0, NPD:** Reset status flag.

NT0	NPD	Condition	Description
0	0	Watchdog reset	Watchdog timer overflow.
0	1	Reserved	-
1	0	Power on reset and LVD reset.	Power voltage is lower than LVD detecting level.
1	1	External reset	External reset pin detect low level status.

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.

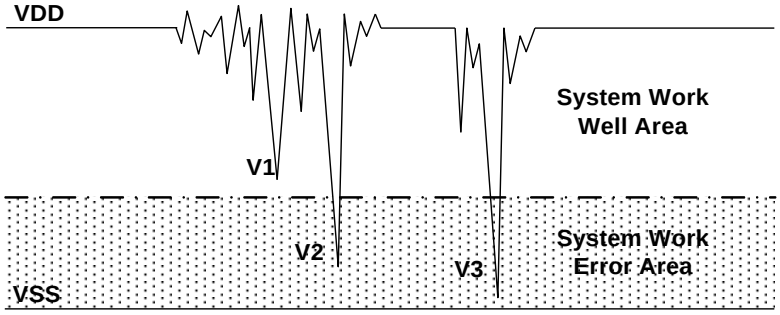
Watchdog timer application note is as following.

- 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.

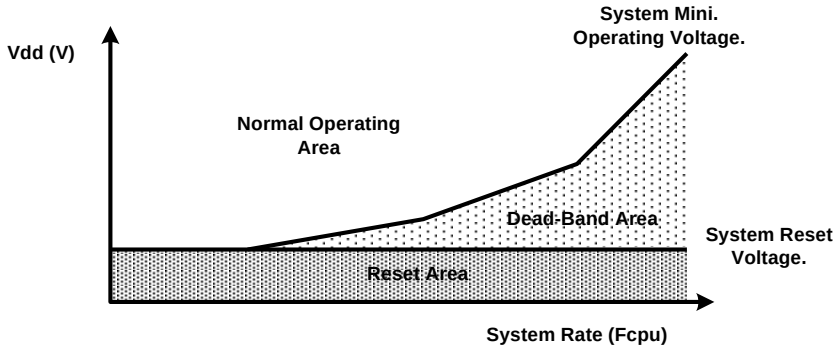
AC application:

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.

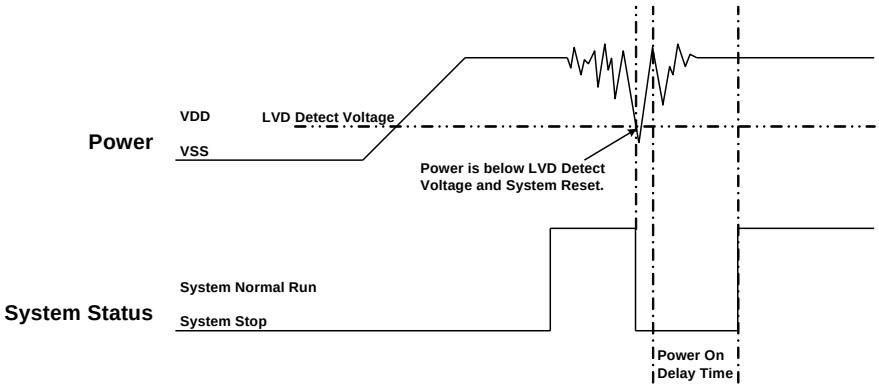
3.4.1 THE SYSTEM OPERATING VOLTAGE

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 (2.0V/2.4V/3.6V) and controlled by LVD code option. The 2.0V 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 3.6V includes flag function to indicate VDD status. LVD flag function can be an **easy low battery detector**. LVD24, LVD36 flags indicate VDD voltage level. For low battery detect application, only checking LVD24, LVD36 status to be battery status. This is a cheap and easy solution.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

Bit 5 **LVD36:** LVD 3.6V operating flag and only support LVD code option is LVD_H.
0 = Inactive (VDD > 3.6V).
1 = Active (VDD ≦ 3.6V).

Bit 4 **LVD24:** LVD 2.4V operating flag and only support LVD code option is LVD_M.
0 = Inactive (VDD > 2.4V).
1 = Active (VDD ≦ 2.4V).

LVD	LVD Code Option			
	LVD_L	LVD_M	LVD_H	LVD_MAX
2.0V Reset	Available	Available	Available	Available
2.4V Flag	-	Available	-	-
2.4V Reset	-	-	Available	-
3.6V Flag	-	-	Available	-
3.6V Reset	-	-	-	Available

LVD_L
If VDD < 2.0V, system will be reset.
Disable LVD24 and LVD36 bit of PFLAG register.
LVD_M
If VDD < 2.0V, 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 LVD36 bit of PFLAG register.
LVD_H
If VDD < 2.4V, 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”.
Enable LVD36 bit of PFLAG register. If VDD > 3.6V, LVD36 is “0”. If VDD ≦3.6V, LVD36 flag is “1”.
LVD_MAX
If VDD < 3.6V, system will be reset.

✸ ***Note:***
1. ***After any LVD reset, LVD24, LVD36 flags are cleared.***
2. ***The voltage level of LVD 2.4V or 3.6V is for design reference only. Don’t use the LVD indicator as precision VDD measurement.***

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

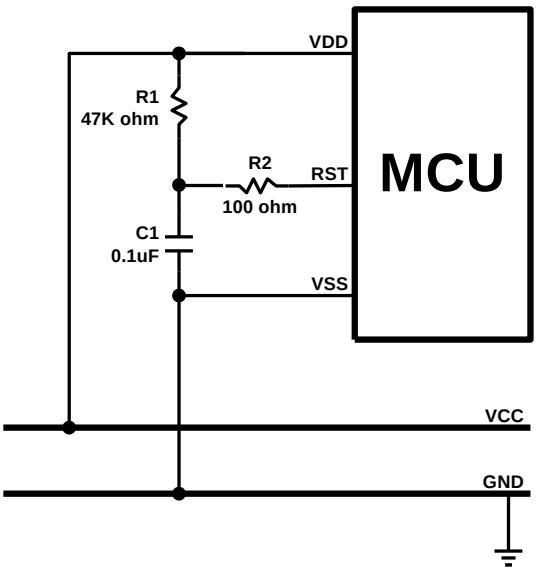
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the 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

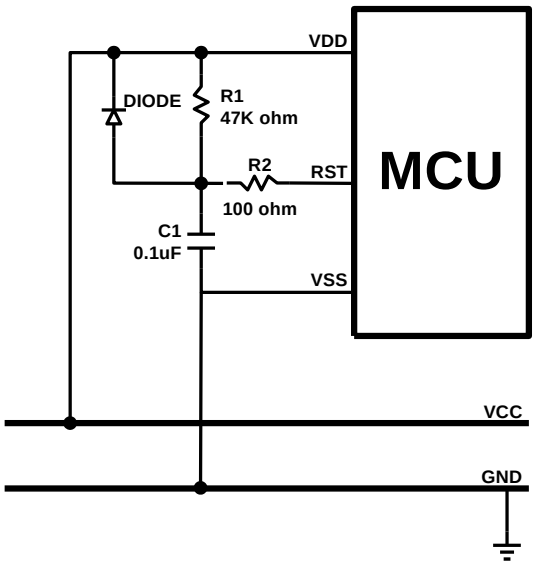
3.6.1 Simply RC Reset Circuit



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.***

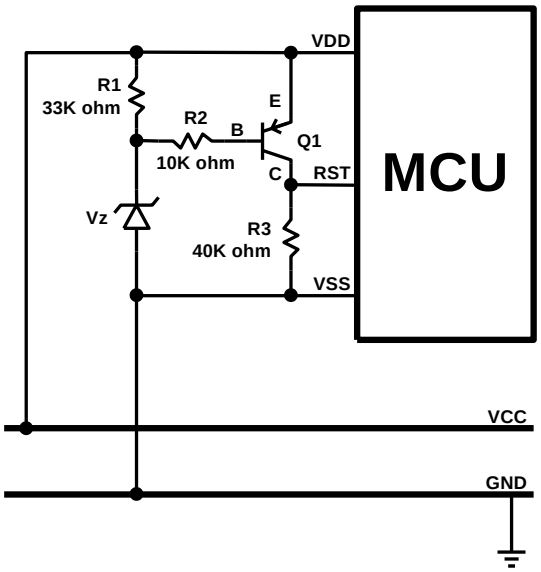
3.6.2 Diode & RC Reset Circuit



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.

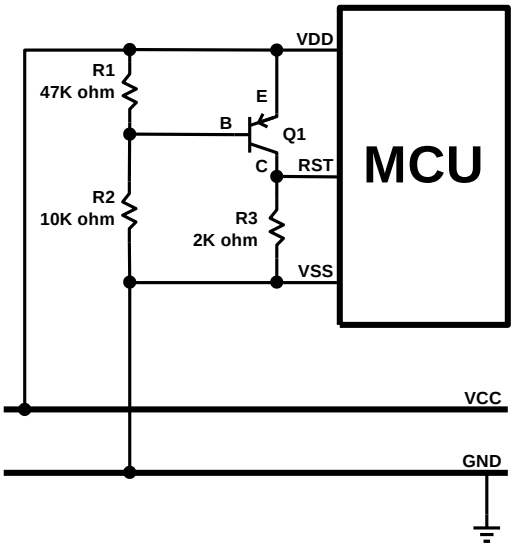
✷ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

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.

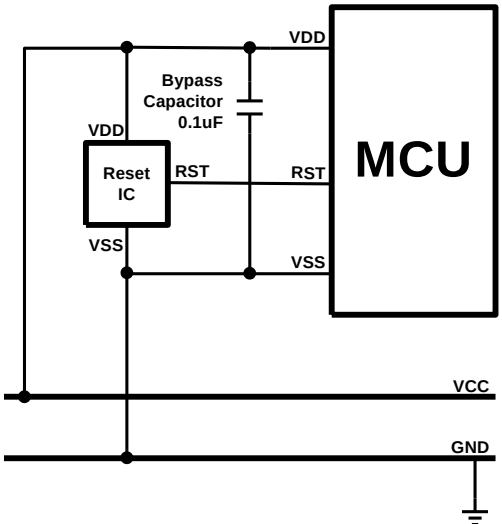
3.6.4 Voltage Bias Reset Circuit



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.

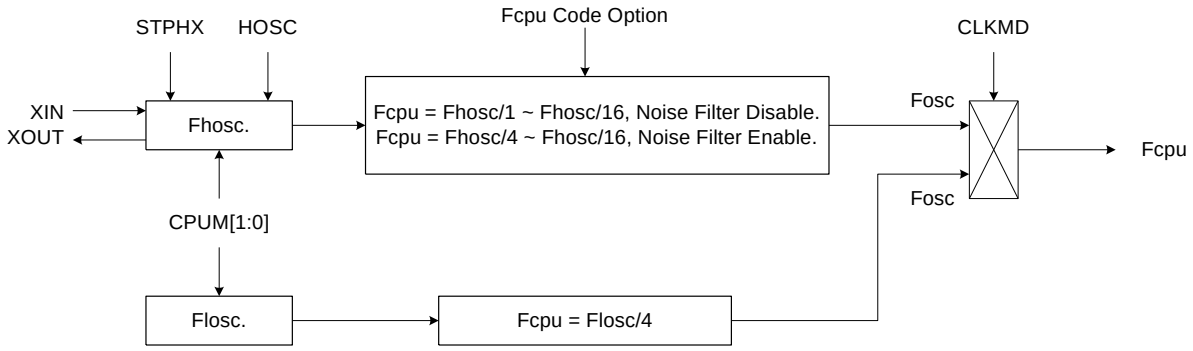
- **High-speed oscillator**

Internal high-speed oscillator is 16MHz RC type called “**IHRC**”.
External high-speed oscillator includes crystal/ceramic (4MHz, 12MHz, 32KHz) and RC type.

- **Low-speed oscillator**

Internal low-speed oscillator is 16KHz @3V, 32KHz @5V 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 16KHz@3V, 32KHz@5V).
- Fosc: System clock source.
- Fcpu: Instruction cycle.

SONIX provides a **“Noise Filter”** controlled by code option. In high noisy situation, the noise filter can isolate noise outside and protect system works well. The minimum Fcpu of high clock is limited at **Fhosc/4** when noise filter enable.

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 Fcpu code option and the range is **Fhosc/1~Fhosc/16** under system normal mode. If the system high clock source is external 4MHz crystal, and the Fcpu code option is Fhosc/4, the Fcpu frequency is 4MHz/4 = 1MHz. Under system slow mode, the Fcpu is fixed Flosc/4, 16KHz/4=4KHz @3V, 32KHz/4=8KHz @5V.
The Fcpu range is limited by high clock and noise filter code option. If high clock code option is RC, 12M X’tal, 4M X’tal or 32K X’tal, the Fcpu range is Fhosc/1~Fhosc/16. If high clock code option is IHRC_16M or IHRC_RTC, the Fcpu range is Fhosc/2~Fhosc/16. If noise filter code option is disabled, the Fcpu range is depended on high clock code option. If noise filter code option is enabled, the Fcpu range is Fhosc/4~Fhosc/16 to reduce noise effect.

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, 32KHz crystal/ceramic and RC type. These high-speed oscillators are selected by “**High_CLK**” code option. The internal high-speed clock supports real time clock (RTC) function. Under “IHRC_RTC” mode, the internal high-speed clock and external 32KHz oscillator active. The internal high-speed clock is the system clock source, and the external 32KHz oscillator is the RTC clock source to supply a accurately real time clock rate.

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, IHRC_RTC, RC, 32K X’tal, 12M X’tal and 4M X’tal. 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.
- **IHRC_RTC:** The system high-speed clock source is internal high-speed 16MHz RC type oscillator. The RTC clock source is external low-speed 32768Hz crystal. The XIN and XOUT pins are defined to drive external 32768Hz crystal and disables GPIO function.
- **RC:** The system high-speed clock source is external low cost RC type oscillator. The RC oscillator circuit only connects to XIN pin, and the XOUT pin is bi-direction GPIO mode.
- **32K X’tal:** The system high-speed clock source is external low-speed 32768Hz crystal. The option only supports 32768Hz crystal and the RTC function is workable.
- **12M X’tal:** The system high-speed clock source is external high-speed crystal/ceramic. The oscillator bandwidth is 10MHz~16MHz.
- **4M X’tal:** The system high-speed clock source is external high-speed crystal/resonator. The oscillator bandwidth is 1MHz~10MHz.

For power consumption under “**IHRC_RTC**” mode, the internal high-speed oscillator and internal low–speed oscillator stops and only external 32KHz crystal actives under green mode. The condition is the watchdog timer can’t be “**Always_On**” option, or the internal low-speed oscillator actives.

4.4.2 INTERNAL HIGH-SPEED OSCILLATOR RC TYPE (IHRC)

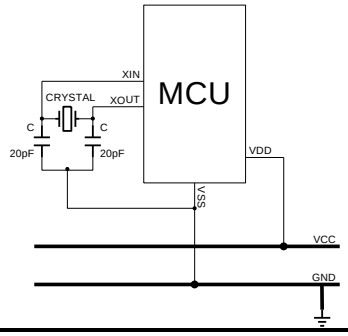
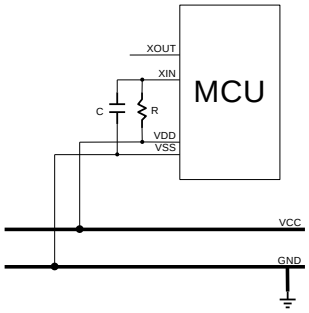
The internal high-speed oscillator is 16MHz RC type. The accuracy is ±2% under commercial condition. When the “High_CLK” code option is “IHRC_16M” or “IHRC_RTC”, the internal high-speed oscillator is enabled.

- **IHRC_16M:** The system high-speed clock is internal 16MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.
- **IHRC_RTC:** The system high-speed clock is internal 16MHz oscillator RC type, and the real time clock is external 32768Hz crystal. XIN/XOUT pins connect with external 32768Hz crystal.

4.4.3 EXTERNAL HIGH-SPEED OSCILLATOR

The external high-speed oscillator includes 4MHz, 12MHz, 32KHz and RC type. The 4MHz, 12MHz and 32KHz 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.

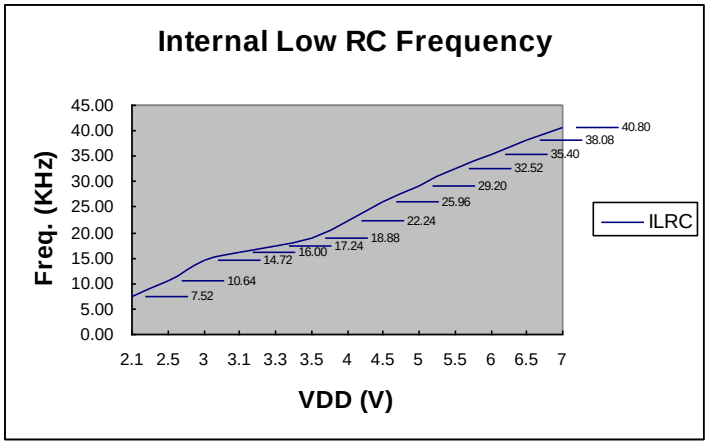
4.4.4 EXTERNAL OSCILLATOR APPLICATION CIRCUIT

CRYSTAL/CERAMIC	RC Type
	

✸ ***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 16KHz at 3V and 32KHz at 5V. The relation between the RC frequency and voltage is as the following figure.



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 16KHz @3V, 32KHz @5V).***
- ***Slow mode Fcpu = Flosc / 4***

There are two conditions to stop internal low RC. One is power down mode, and the other is green mode of 32K mode and watchdog disable. If system is in 32K mode and watchdog disable, only 32K oscillator actives and system is under low power consumption.

➢ **Example: Stop internal low-speed oscillator by power down mode.**

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

✸ ***Note: The internal low-speed clock can’t be turned off individually. It is controlled by CPUM0, CPUM1 (32K, watchdog disable) bits of OSCM register.***

4.6 OSCM REGISTER

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

0CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	0	0	0	CPUM1	CPUM0	CLKMD	STPHX	0
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.

- **IHRC_16M: "STPHX=1" disables internal high speed RC type oscillator.**
- **IHRC_RTC: "STPHX=1" disables internal high speed RC type oscillator and external 32768Hz crystal.**
- **RC, 4M, 12M, 32K: "STPHX=1" disables external oscillator.**

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.**

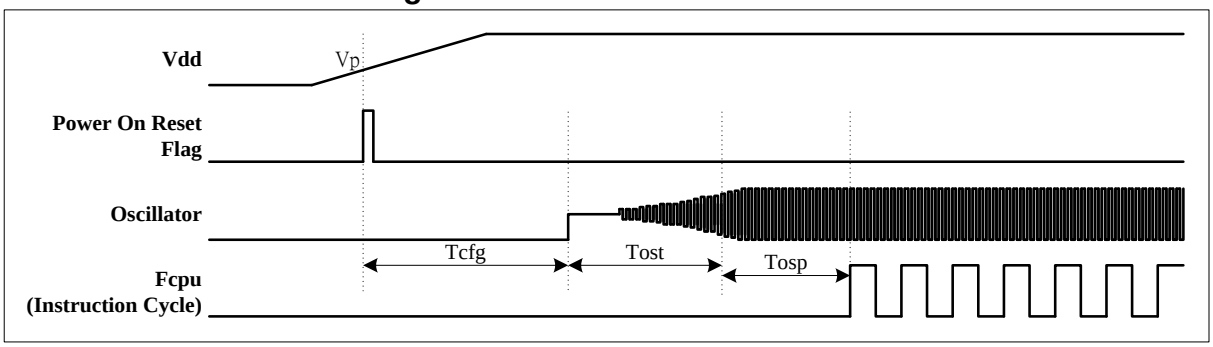
```
        B0BSET    P0M.0     ; Set P0.0 to be output mode for outputting Fcpu toggle signal.
@@:
        B0BSET    P0.0      ; Output Fcpu toggle signal in low-speed clock mode.
        B0BCLR    P0.0      ; Measure the Fcpu frequency by oscilloscope.
        JMP       @B
```

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

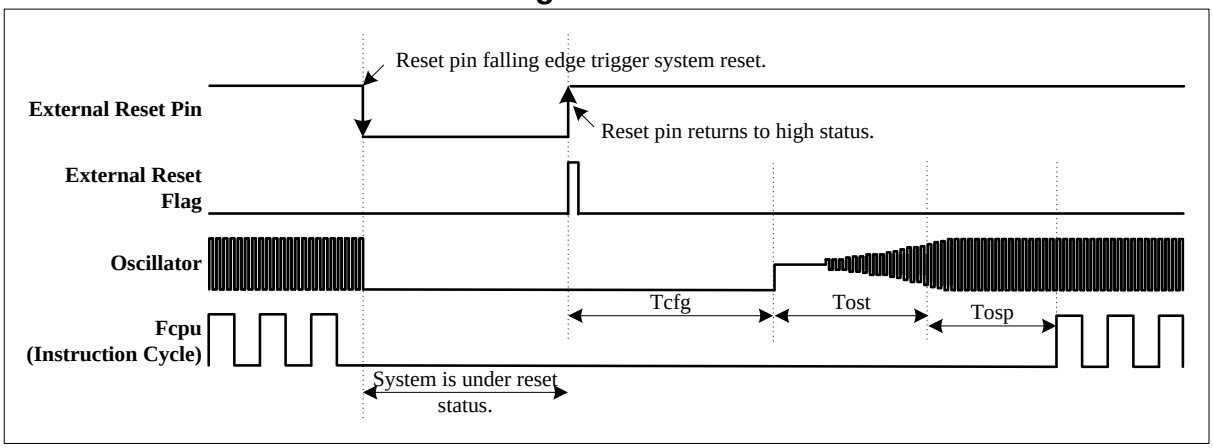
4.8 SYSTEM CLOCK TIMING

Parameter	Symbol	Description	Typical
Hardware configuration time	Tcfg	2048*F_{ILRC}	64ms @ F_{ILRC} = 32KHz 128ms @ F_{ILRC} = 16KHz
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.)	64ms @ F_{hosc} = 32KHz 512us @ F_{hosc} = 4MHz 128us @ F_{hosc} = 16MHz
		Oscillator warm-up time of power down mode wake-up condition. 2048*F_{hosc} ……Crystal/resonator type oscillator, e.g. 32768Hz crystal, 4MHz crystal, 16MHz crystal… 32*F_{hosc}……RC type oscillator, e.g. external RC type oscillator, internal high-speed RC type oscillator.	X'tal: 64ms @ F_{hosc} = 32KHz 512us @ F_{hosc} = 4MHz 128us @ F_{hosc} = 16MHz RC: 8us @ F_{hosc} = 4MHz 2us @ F_{hosc} = 16MHz

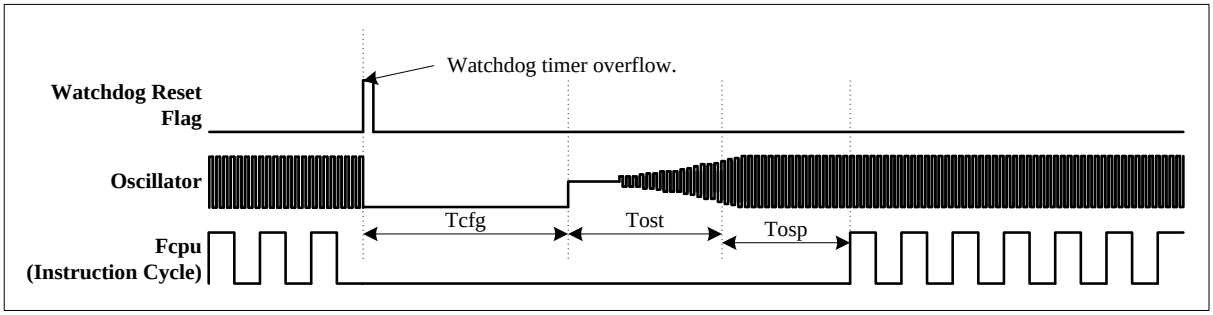
- **Power On Reset Timing**



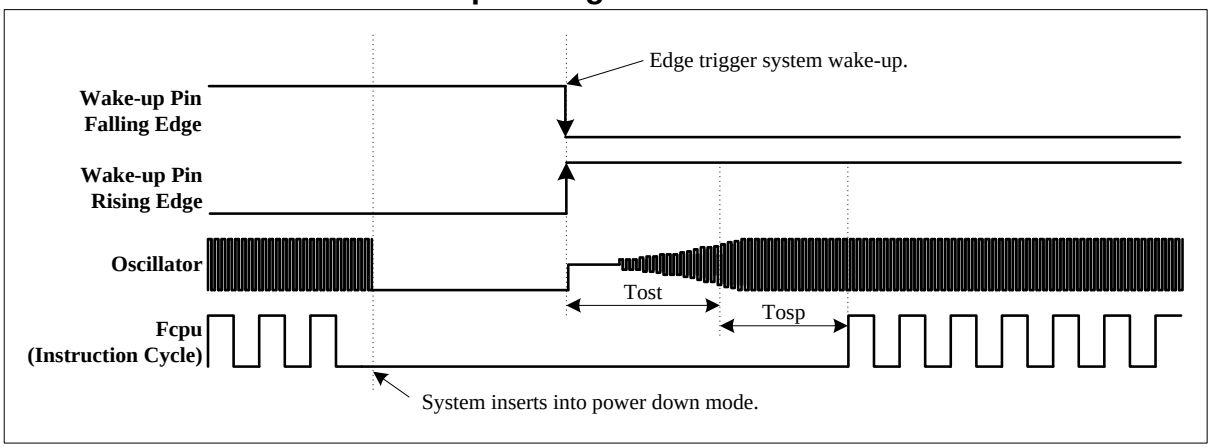
- **External Reset Pin Reset Timing**



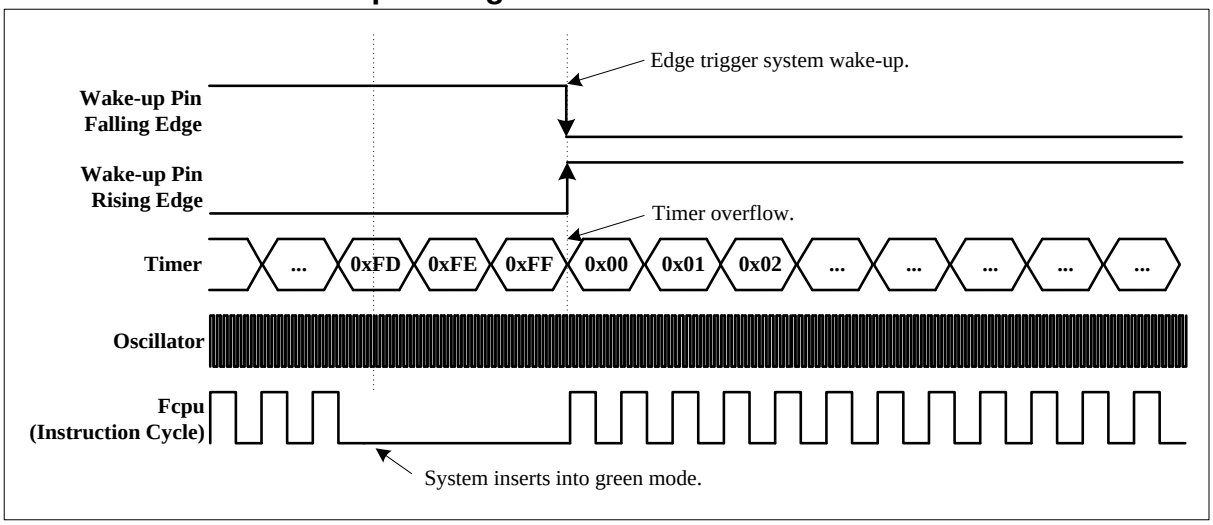
Watchdog Reset Timing



Power Down Mode Wake-up Timing

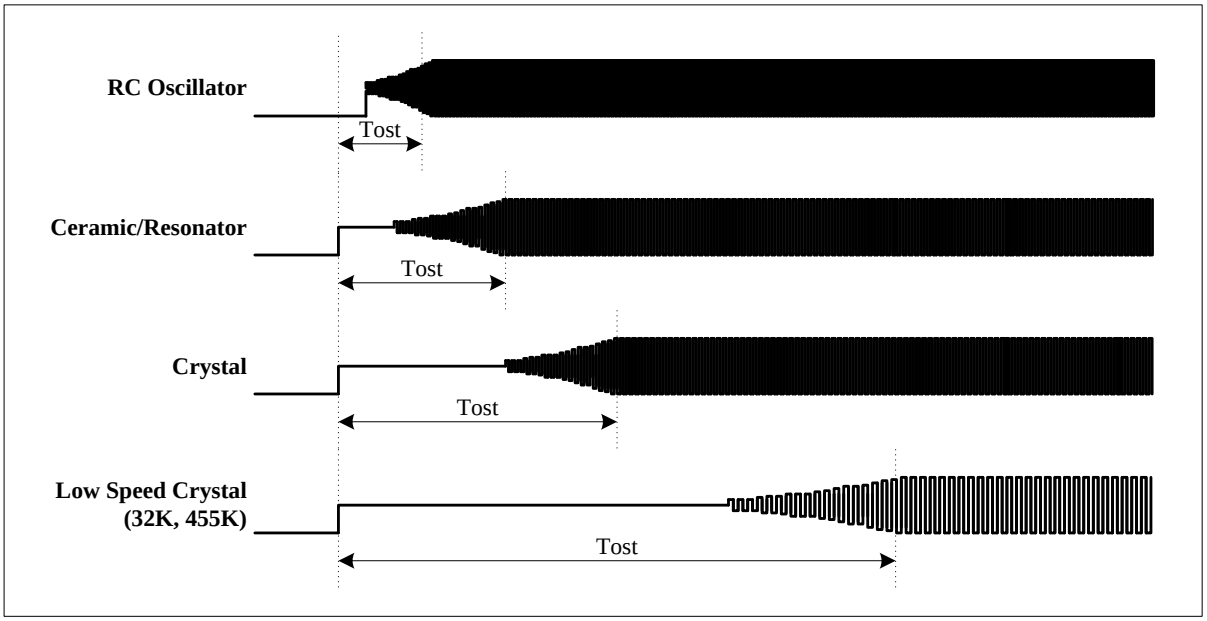


Green Mode Wake-up Timing



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.



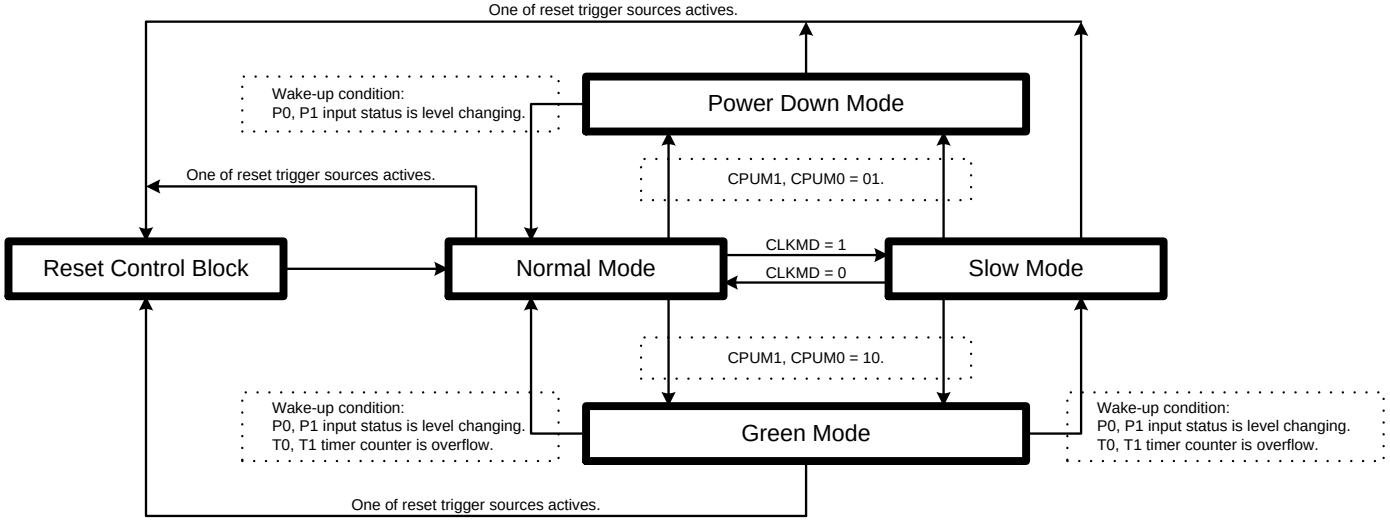
5 SYSTEM OPERATION MODE

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
EHOSC	Running	By STPHX	By STPHX	Stop
IHRC	Running	By STPHX	By STPHX	Stop
ILRC	Running	Running	Running	Stop
EHOSC with **RTC**	Running	By STPHX	Running	Stop
IHRC with **RTC**	Running	By STPHX	Stop	Stop
ILRC with **RTC**	Running	Running	Stop	Stop
CPU instruction	Executing	Executing	Stop	Stop
T0 timer	By T0ENB	By T0ENB	By T0ENB	Inactive
TC0 timer	By TC0ENB	By TC0ENB	By TC0ENB (PWM/Buzzer active)	Inactive
T1 timer	By T1ENB	By T1ENB	By T1ENB	Inactive
PWM1, PWM2	By PWnEN	By PWnEN	By PWnEN	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	T0, T1	All inactive
External interrupt	All active	All active	All active	All inactive
Wakeup source	-	-	P0, P1, T0, T1, Reset	P0, P1, Reset

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

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 rate is fixed Flosc/4 (Flosc is internal low speed RC type oscillator frequency).

- The program is executed, and full functions are controllable.
- The system rate is low speed (Flosc/4).
- 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. Whole chip is under low power consumption status under 1uA. The power down mode is waked up by P0, 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.

- 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 power consumption is under 1uA.
- The system inserts into normal mode after wake-up from power down mode.
- The power down mode wake-up source is P0 and P1 level change trigger.

✷ ***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 P0, 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 P0, 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.

- 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 P0, P1 level change trigger and unique time overflow.
- PWM and buzzer output functions active in green mode, but the timer can't wake-up the system as overflow.

✸ ***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.
```

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

```
    CLR         T0C             ; Clear T0 counter.
    B0BSET      FT0TB           ; Enable T0 RTC function.
    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 (P0/P1 level change) and internal trigger (T0/T1 timer overflow).

- Power down mode is waked up to normal mode. The wakeup trigger is only external trigger (P0/P1 level change)
- Green mode is waked up to last mode (normal mode or slow mode). The wakeup triggers are external trigger (P0/P1 level change) and internal trigger (T0/T1 timer overflow).

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 32 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 = 0.512 ms (Fosc = 4MHz)
The total wakeup time = 0.512 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 * 32 (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 * 32 = 2 us (Fhosc = 16MHz)

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

5.7.3 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. The Port 0 and Port 1 have wakeup function. Port 0 wakeup function always enables, but the Port 1 is controlled by the P1W register.

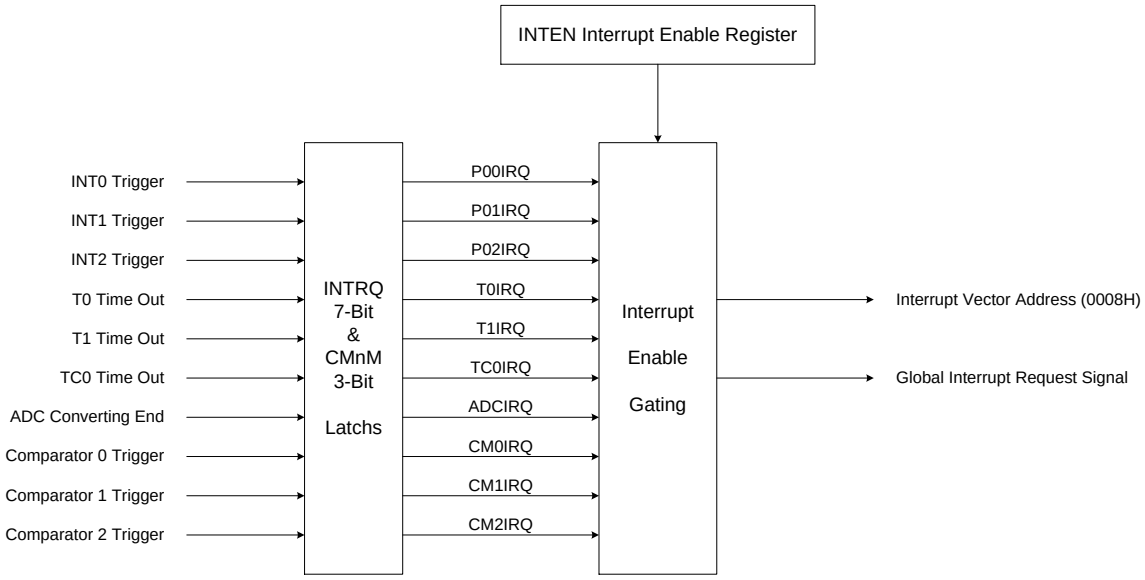
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 10 interrupt sources, including 7 internal interrupt (T0/T1/TC0/CM0/CM1/CM2/ADC) and 3 external interrupt (INT0/INT1/INT2). 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. Most of the interrupt request signals are stored in INTRQ register, but comparator interrupt request flags are stored in CMnM registers.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

INTEN is the interrupt request control register including four internal interrupts, three 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 to execute interrupt service routines. The program exits the interrupt service routine when the returning interrupt service routine instruction (RETI) is executed.

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	ADCIEN	T1IEN	TC0IEN	T0IEN	-	P02IEN	P01IEN	P00IEN
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 **P00IEN:** External P0.0 interrupt (INT0) control bit.
0 = Disable INT0 interrupt function.
1 = Enable INT0 interrupt function.

Bit 1 **P01IEN:** External P0.1 interrupt (INT1) control bit.
0 = Disable INT1 interrupt function.
1 = Enable INT1 interrupt function.

Bit 2 **P02IEN:** External P0.2 interrupt (INT2) control bit.
0 = Disable INT2 interrupt function.
1 = Enable INT2 interrupt function.

Bit 4 **T0IEN:** T0 timer interrupt control bit.
0 = Disable T0 interrupt function.
1 = Enable T0 interrupt function.

Bit 5 **TC0IEN:** TC0 timer interrupt control bit.
0 = Disable TC0 interrupt function.
1 = Enable TC0 interrupt function.

Bit 6 **T1IEN:** T1 timer interrupt control bit.
0 = Disable T1 interrupt function.
1 = Enable T1 interrupt function.

Bit 7 **ADCIEN:** ADC interrupt control bit.
0 = Disable ADC interrupt function.
1 = Enable ADC interrupt function.

CM0IEN (CM0M’s bit 6): Comparator 0 interrupt control bit.
0 = Disable comparator 0 interrupt function.
1 = Enable comparator 0 interrupt function.

CM1IEN (CM1M’s bit 6): Comparator 1 interrupt control bit.
0 = Disable comparator 1 interrupt function.
1 = Enable comparator 1 interrupt function.

CM2IEN (CM2M’s bit 6): Comparator 2 interrupt control bit.
0 = Disable comparator 2 interrupt function.
1 = Enable comparator 2 interrupt function.

6.3 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.

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	ADCIRQ	T1IRQ	TC0IRQ	T0IRQ	-	P02IRQ	P01IRQ	P00IRQ
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 **P00IRQ:** External P0.0 interrupt (INT0) request flag.
0 = None INT0 interrupt request.
1 = INT0 interrupt request.

Bit 1 **P01IRQ:** External P0.1 interrupt (INT1) request flag.
0 = None INT1 interrupt request.
1 = INT1 interrupt request.

Bit 2 **P02IRQ:** External P0.2 interrupt (INT2) request flag.
0 = None INT2 interrupt request.
1 = INT2 interrupt request.

Bit 4 **T0IRQ:** T0 timer interrupt request flag.
0 = None T0 interrupt request.
1 = T0 interrupt request.

Bit 5 **TC0IRQ:** TC0 timer interrupt request flag.
0 = None TC0 interrupt request.
1 = TC0 interrupt request.

Bit 6 **T1IRQ:** T1 timer interrupt request flag.
0 = None T1 interrupt request.
1 = T1 interrupt request.

Bit 7 **ADCIRQ:** ADC interrupt request flag.
0 = None ADC interrupt request.
1 = ADC interrupt request.

CM0IRQ (CM0M’s bit 5): Comparator 0 interrupt request flag.
0 = None comparator 0 interrupt request.
1 = Comparator 0 interrupt request.

CM1IRQ (CM1M’s bit 5): Comparator 1 interrupt request flag.
0 = None comparator 1 interrupt request.
1 = Comparator 1 interrupt request.

CM2IRQ (CM2M’s bit 5): Comparator 2 interrupt request flag.
0 = None comparator 2 interrupt request.
1 = Comparator 2 interrupt request.

6.4 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) and the stack add 1 level.

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0
Read/Write	R/W	-	-	-	-	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.***

6.5 PUSH, POP ROUTINE

When any interrupt occurs, system will jump to ORG 8 and execute interrupt service routine. It is necessary to save ACC, PFLAG data. The chip includes “PUSH”, “POP” for in/out interrupt service routine. The two instructions save and load **ACC**, **PFLAG** data into buffers and avoid main routine error after interrupt service routine finishing.

✷ ***Note: ”PUSH”, “POP” instructions save and load ACC/PFLAG without (NT0, NPD). PUSH/POP buffer is an unique buffer and only one level.***

➢ **Example: Store ACC and PAFLG data by PUSH, POP instructions when interrupt service routine executed.**

```
                ORG     0
                JMP     START

                ORG     8
                JMP     INT_SERVICE

                ORG     10H
START:
                …

INT_SERVICE:
                PUSH                    ; Save ACC and PFLAG to buffers.
                …
                …
                POP                     ; Load ACC and PFLAG from buffers.

                RETI                    ; Exit interrupt service vector
                …
                ENDP
```

6.6 EXTERNAL INTERRUPT OPERATION (INT0~INT2)

Sonix provides 3 sets external interrupt sources in the micro-controller. INT0, INT1 and INT2 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 8) 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 (P0.0, P0.1 or P0.2), 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.

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	P02G1	P02G0	P01G1	P01G0	P00G1	P00G0
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit[5:4] **P02G[1:0]:** INT2 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

Bit[3:2] **P01G[1:0]:** INT1 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

Bit[1:0] **P00G[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  FP00IEN         ; Enable INT0 interrupt service
        B0BCLR  FP00IRQ         ; Clear INT0 interrupt request flag
        B0BSET  FGIE            ; Enable GIE
```

Example: INT0 interrupt service routine.

```
        ORG     8               ; Interrupt vector
        JMP     INT_SERVICE
INT_SERVICE:
        …                       ; Push routine to save ACC and PFLAG to buffers.

        B0BTS1  FP00IRQ         ; Check P00IRQ
        JMP     EXIT_INT        ; P00IRQ = 0, exit interrupt vector

        B0BCLR  FP00IRQ         ; Reset P00IRQ
        …                       ; INT0 interrupt service routine
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.
        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. Fcpu = 4MHz / 4.**

```
        B0BCLR      FT0IEN      ; Disable T0 interrupt service
        B0BCLR      FT0ENB      ; Disable T0 timer
        MOV         A, #20H     ;
        B0MOV       T0M, A      ; Set T0 clock = Fcpu / 64
        MOV         A, #64H     ; Set T0C initial value = 64H
        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         8           ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

        …                       ; Push routine to save ACC and PFLAG to buffers.

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

        B0BCLR      FT0IRQ      ; Reset T0IRQ
        MOV         A, #64H
        B0MOV       T0C, A      ; Reset T0C.
        …                       ; T0 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.


        RETI                    ; Exit interrupt vector
```

6.8 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. Fcpu = 16MHz / 16.**

```
        B0BCLR      FTC0IEN         ; Disable TC0 interrupt service
        B0BCLR      FTC0ENB         ; Disable TC0 timer
        MOV         A, #20H         ;
        B0MOV       TC0M, A         ; Set TC0 clock = Fcpu / 64
        MOV         A, #64H         ; Set TC0C initial value = 64H
        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         8               ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; Push routine to save ACC and PFLAG to buffers.

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

        B0BCLR      FTC0IRQ         ; Reset TC0IRQ
        MOV         A, #64H
        B0MOV       TC0C, A         ; Reset TC0C.
        …                           ; TC0 interrupt service routine
        …
EXIT_INT:
        …                           ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

6.9 T1 INTERRUPT OPERATION

When the T1C (T1CH, T1CL) 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 / 32.
        CLR         T1CH
        CLR         T1CL

        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         8           ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

        PUSH                    ; Push routine to save ACC and PFLAG to buffers.

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

        B0BCLR      FT1IRQ      ; Reset T1IRQ
        B0MOV       A, T1CH
        B0MOV       T1CHBUF, A
        B0MOV       A, T1CL
        B0MOV       T1CLBUF, A  ; Save pulse width.
        CLR         T1CH
        CLR         T1CL
        …                       ; T1 interrupt service routine
        …
EXIT_INT:
        POP                     ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.10 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 = Fcpu/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         8               ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; Push routine to save ACC and PFLAG to buffers.

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

        B0BCLR      FADCIRQ         ; Reset ADCIRQ
        …                           ; ADC interrupt service routine
        …
EXIT_INT:
        …                           ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

6.11 COMPARATOR INTERRUPT OPERATION (CMP0~CMP2)

Sonix provides 3 sets comparator with interrupt function in the micro-controller. The comparator interrupt trigger edge direction is controlled by comparator register. CM0G[1:0] of CM0M is control comparator 0 interrupt trigger edge direction. CM1G[1:0] of CM1M is control comparator 1 interrupt trigger edge direction. CM2G[1:0] of CM2M is control comparator 2 interrupt trigger edge direction. When the comparator output status transition occurs, the comparator interrupt request flag will be set to “1” no matter the comparator interrupt control bit status. The comparator interrupt flag doesn’t active only when comparator control bit is disabled. When comparator interrupt control bit is enabled and comparator interrupt edge trigger is occurring, the program counter will jump to the interrupt vector (ORG 8) and execute interrupt service routine.

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CM0G1	CM0G0
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 6 **CM0IEN:** Comparator 0 interrupt function control bit.
0 = Disable.
1 = Enable.

Bit 5 **CM0IRQ:** Comparator 0 interrupt request bit.
0 = Non comparator interrupt request.
1 = Announce comparator interrupt request.

Bit [1:0] **CM0G[1:0]:** Comparator interrupt trigger direction control bit.
00 = Reserved.
01 = Rising edge trigger. CM0P > CM0N or comparator internal reference voltage.
10 = Falling edge trigger. CM0P < CM0N or comparator internal reference voltage.
11 = Both rising and falling edge trigger (Level change trigger).

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM1M	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	CM1G1	CM1G0
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 6 **CM1IEN:** Comparator 1 interrupt function control bit.
0 = Disable.
1 = Enable.

Bit 5 **CM1IRQ:** Comparator 1 interrupt request bit.
0 = Non comparator interrupt request.
1 = Announce comparator interrupt request.

Bit [1:0] **CM1G[1:0]:** Comparator interrupt trigger direction control bit.
00 = Reserved.
01 = Rising edge trigger. CM1P > CM1N or comparator internal reference voltage.
10 = Falling edge trigger. CM1P < CM1N or comparator internal reference voltage.
11 = Both rising and falling edge trigger (Level change trigger).

09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM2M	CM2EN	CM2IEN	CM2IRQ	CM2OEN	CM2REF	CM2OUT	CM2G1	CM2G0
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 6 **CM2IEN:** Comparator 2 interrupt function control bit.
0 = Disable.
1 = Enable.

Bit 5 **CM2IRQ:** Comparator 2 interrupt request bit.
0 = Non comparator interrupt request.
1 = Announce comparator interrupt request.

Bit [1:0] **CM2G[1:0]:** Comparator interrupt trigger direction control bit.
00 = Reserved.
01 = Rising edge trigger. CM2P > CM2N or comparator internal reference voltage.
10 = Falling edge trigger. CM2P < CM2N or comparator internal reference voltage.
11 = Both rising and falling edge trigger (Level change trigger).

Example: Setup comparator 0 interrupt request and bi-direction edge trigger.

```
        MOV     A, #03H
        B0MOV   CM0M, A         ; Set comparator 0 interrupt trigger as bi-direction edge.

        B0BSET  FCM0IEN         ; Enable comparator 0 interrupt service
        B0BCLR  FCM0IRQ         ; Clear comparator 0 interrupt request flag
        B0BSET  FCM0EN          ; Enable comparator 0.
        B0BSET  FGIE            ; Enable GIE
```

Example: Comparator 0 interrupt service routine.

```
        ORG     8               ; Interrupt vector
        JMP     INT_SERVICE
INT_SERVICE:
        …                       ; Push routine to save ACC and PFLAG to buffers.

        B0BTS1  FCM0IRQ         ; Check CM0IRQ
        JMP     EXIT_INT        ; CM0IRQ = 0, exit interrupt vector

        B0BCLR  FCM0IRQ         ; Reset CM0IRQ
        …                       ; Comparator 0 interrupt service routine
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.
        RETI                    ; Exit interrupt vector
```

6.12 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*
P00IRQ	P0.0 trigger controlled by PEDGE
P01IRQ	P0.1 trigger controlled by PEDGE
P02IRQ	P0.2 trigger controlled by PEDGE
T0IRQ	T0C overflow
T1IRQ	T1CH, T1CL overflow
TC0IRQ	TC0C overflow
ADCIRQ	ADC converting end.
CM0IRQ	Comparator 0 output level transition.
CM1IRQ	Comparator 1 output level transition.
CM2IRQ	Comparator 2 output level transition.

For multi-interrupt conditions, two things need to be taking care of. One is to set the priority for these interrupt requests. Two is using IEN and IRQ flags to decide which interrupt to be executed. Users have to check interrupt control bit and interrupt request flag in interrupt routine.

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

```
                ORG         8               ; Interrupt vector
                JMP         INT_SERVICE
INT_SERVICE:
                …                           ; Push routine to save ACC and PFLAG to buffers.
INTP00CHK:                                  ; Check INT0 interrupt request
                B0BTS1      FP00IEN         ; Check P00IEN
                JMP         INTT0CHK        ; Jump check to next interrupt
                B0BTS0      FP00IRQ         ; Check P00IRQ
                JMP         INTP00
INTT0CHK:                                   ; Check T0 interrupt request
                B0BTS1      FT0IEN          ; Check T0IEN
                JMP         INTTC0CHK       ; Jump check to next interrupt
                B0BTS0      FT0IRQ          ; Check T0IRQ
                JMP         INTT0           ; Jump to T0 interrupt service routine
INTTC0CHK:                                  ; Check TC0 interrupt request
                B0BTS1      FTC0IEN         ; Check TC0IEN
                JMP         INTADCHK        ; Jump check to next interrupt
                B0BTS0      FTC0IRQ         ; Check TC0IRQ
                JMP         INTTC0          ; Jump to TC0 interrupt service routine
INTADCHK:                                   ; Check ADC interrupt request
                B0BTS1      FADCIEN         ; Check ADCIEN
                JMP         …               ; Jump check to next interrupt
                B0BTS0      FADCIRQ         ; Check ADCIRQ
                JMP         INTADC          ; Jump to ADC interrupt service routine
                …
                …
INT_EXIT:
                …                           ; Pop routine to load ACC and PFLAG from buffers.
                RETI                        ; Exit interrupt vector
```

7 I/O PORT

7.1 OVERVIEW

The micro-controller builds in 30 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	INT0	DC	P00IEN=1
		PWM1T	DC	PW1EN=1, PW1GEN=1, PW1GS=0
P0.1	I/O	INT1	DC	P01IEN=1
		PWM1	DC	PW1EN=1.
P0.2	I/O	INT2	DC	P02IEN=1
		PWM1N	DC	PW1EN=1, PW1NEN=1
P0.3	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P0.4	I/O	XOUT	AC	High_CLK code option = IHRC_RTC, 32K, 4M, 12M
P0.5	I/O	XIN	AC	High_CLK code option = IHRC_RTC, RC, 32K, 4M, 12M
P1.0	I/O	CM2N	AC	CM2EN=1
		OP2N	AC	OP2EN=1
P1.1	I/O	CM2P	AC	CM2EN=1, CM2REF=0
		OP2P	AC	OP2EN=1
P1.2	I/O	CM2O	AC	CM2EN=1, CM2OEN=1
		OP2O	AC	OP2EN=1
P1.3	I/O	CM1N	AC	CM1EN=1
		OP1N	AC	OP1EN=1
P1.4	I/O	CM1P	AC	CM1EN=1, CM1REF=0
		OP1P	AC	OP1EN=1
P1.5	I/O	CM1O	AC	CM1EN=1, CM1OEN=1
		OP1O	AC	OP1EN=1
P1.6	I/O	CM0N	AC	CM0EN=1
		OP0N	AC	OP0EN=1
P1.7	I/O	CM0P	AC	CM0EN=1, CM0REF=0
		OP0P	AC	OP0EN=1
P5.0	I/O	CM0O	AC	CM0EN=1, CM0OEN=1
		OP0O	AC	OP0EN=1
P5.1	I/O	PWM21	DC	PW2EN=1, PW2CH1=1
P5.2	I/O	PWM22	DC	PW2EN=1, PW2CH2=1
P5.3	I/O	PWM23	DC	PW2EN=1, PW2CH3=1
P5.4	I/O	BZ0/PWM0	DC	TC0ENB=1, TC0OUT=1 or PWM0OUT=1
P5.5	I/O	PWM24	DC	PW2EN=1, PW2CH4=1
P5.6	I/O	PWM25	DC	PW2EN=1, PW2CH5=1
P5.7	I/O	PWM26	DC	PW2EN=1, PW2CH6=1
P4.0	I/O	AIN0	AC	ADENB=1, GCHS=1, CHS[2:1] = 000b
		AVREFH	AC	ADENB=1, AVREFH=1
P4[7:1]	I/O	AIN[7:1]	AC	ADENB=1, GCHS=1, CHS[2:1] = 001b~111b

*** 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.

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	-	-	P05M	P04M	-	P02M	P01M	P00M
Read/Write	-	-	R/W	R/W	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

0C1H	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

0C4H	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

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M
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] **PnM[7:0]:** Pn mode control bits. (n = 0~5).
0 = Pn is input mode.
1 = Pn is output mode.

✸ ***Note:***
1. ***Users can program them by bit control instructions (B0BSET, B0BCLR).***
2. ***P0.3 input only pin, and the P0M.3 is undefined.***

➢ **Example: I/O mode selecting**

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

MOV     A, #0FFH    ; Set all ports to be output mode.
B0MOV   P0M, A
B0MOV   P4M,A
B0MOV   P5M, A

B0BCLR  P4M.0       ; Set P4.0 to be input mode.

B0BSET  P4M.0       ; Set P4.0 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.

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	-	P05R	P04R	-	P02R	P01R	P00R
Read/Write	-	-	W	W	-	W	W	W
After reset	-	-	0	0	-	0	0	0

0E1H	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	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0E4H	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	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

✸ ***Note: P0.3 is input only pin and without pull-up resister. The P0UR.3 is undefined.***

➢ **Example: I/O Pull up Register**

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

7.4 I/O PORT DATA REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	-	-	P05	P04	P03	P02	P01	P00
Read/Write	-	-	R/W	R/W	R	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

0D1H	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

0D4H	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

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	P57	P56	P55	P54	P53	P52	P51	P50
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

✸ ***Note: The P03 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, 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   P4, A
B0MOV   P5, A
```

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

```
B0BSET  P4.0        ; Set P4.0 and P5.3 to be "1".
B0BSET  P5.3

B0BCLR  P4.0        ; Set P4.0 and P5.3 to be "0".
B0BCLR  P5.3
```

7.5 PORT 4 ADC SHARE PIN

The Port 4 is shared with ADC input function and no Schmitt trigger structure. Only one pin of port 4 can be configured as ADC input in the same time by ADM register. The other pins of port 4 are 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 port 4 will encounter above current leakage situation. P4CON is Port4 Configuration register. Write "1" into P4CON.n will configure related port 4 pin as pure analog input pin to avoid current leakage.

0AEH	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	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[4:0] **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.

✸ ***Note: When Port 4.n is general I/O port not ADC channel, P4CON.n must set to "0" or the Port 4.n digital I/O signal would be isolated.***

Port 4 ADC analog input is controlled by GCHS and CHSn bits of ADM register. If GCHS = 0, P4.n is general purpose bi-direction I/O port. If GCHS = 1, P4.n pointed by CHSn is ADC analog signal input pin.

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	AVREFH	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 4 **GCHS:** Global channel select bit.
0 = Disable AIN channel.
1 = Enable AIN channel.

Bit 3 **AVREFH:** ADC external high reference voltage input pin control bit.
0 = ADC high reference voltage is from internal Vdd. P4.0 is GPIO or AIN0 pin.
1 = Enable ADC external high reference voltage input pin from P4.0.

Bit[2:0] **CHS[2:0]:** ADC input channels select bit.
000 = AIN0, 001 = AIN1, 010 = AIN2, 011 = AIN3, 100 = AIN4, 101 = AIN5, 110 = AIN6, 111 = AIN7.

✸ ***Note: For P4.n general purpose I/O function, users should make sure of P4.n's ADC channel is disabled, or P4.n is automatically set as ADC analog input when GCHS = 1 and CHS[2:0] point to P4.n.***

➢ **Example: Set P4.1 to be general purpose input mode. P4CON.1 must be set as "0".**

```
; Check GCHS and CHS[2:0] status.
        B0BCLR      FGCHS       ;If CHS[2:0] point to P4.1 (CHS[2:0] = 001B), set GCHS=0
                                ;If CHS[2:0] don't point to P4.1 (CHS[2:0] ≠ 001B), don't
                                care GCHS status.

; Clear P4CON.
        B0BCLR      P4CON.1     ; Enable P4.1 digital function.

; Enable P4.1 input mode.
        B0BCLR      P4M.1       ; Set P4.1 as input mode.
```

➢ **Example: Set P4.1 to be general purpose output. P4CON.1 must be set as "0".**

```
; Check GCHS and CHS[2:0] status.
        B0BCLR      FGCHS       ;If CHS[2:0] point to P4.1 (CHS[2:0] = 001B), set GCHS=0.
                                ;If CHS[2:0] don't point to P4.1 (CHS[2:0] ≠ 001B), don't
                                care GCHS status.

; Clear P4CON.
        B0BCLR      P4CON.1     ; Enable P4.1 digital function.

; Set P4.1 output buffer to avoid glitch.
        B0BSET      P4.1        ; Set P4.1 buffer as "1".
; or
        B0BCLR      P4.1        ; Set P4.1 buffer as "0".

; Enable P4.1 output mode.
        B0BSET      P4M.1       ; Set P4.1 as input mode.
```

P4.0 is shared with general purpose I/O, ADC input (AIN0) and ADC external high reference voltage input. AVREFH flag of ADM register is external ADC high reference voltage input control bit. If AVREFH is enabled, P4.0 general purpose I/O and ADC analog input (AIN0) functions are disabled. P4.0 pin is connected to ADC high reference voltage directly.

✸ ***Note:*** **For P4.0 general purpose I/O and AIN0 functions, AVREFH must be set as "0".**

➢ **Example: Set P4.0 to be general purpose input mode. AVREFH and P4CON.0 bits must be set as "0".**

```
; Check AVREFH status.
        B0BTS0      FAVREFH     ; Check AVREFH = 0.
        B0BCLR      FAVREFH     ; AVREFH = 1, clear it to disable external ADC high
                                reference input.
                                ; AVREFH = 0, execute next routine.

; Check GCHS and CHS[2:0] status.
        B0BCLR      FGCHS       ;If CHS[2:0] point to P4.0 (CHS[2:0] = 000B), set GCHS=0
                                ;If CHS[2:0] don't point to P4.0 (CHS[2:0] ≠ 000B), don't
                                care GCHS status.

; Clear P4CON.
        B0BCLR      P4CON.0     ; Enable P4.0 digital function.

; Enable P4.0 input mode.
        B0BCLR      P4M.0       ; Set P4.0 as input mode.
```

➢ **Example: Set P4.0 to be general purpose output. EVHENB and P4CON.0 bits must be set as "0".**

```
; Check AVREFH status.
        B0BTS0      FAVREFH     ; Check AVREFH = 0.
        B0BCLR      FAVREFH     ; AVREFH = 1, clear it to disable external ADC high reference input.
                                ; AVREFH = 0, execute next routine.

; Check GCHS and CHS[2:0] status.
        B0BCLR      FGCHS       ;If CHS[2:0] point to P4.0 (CHS[2:0] = 000B), set GCHS=0
                                ;If CHS[2:0] don't point to P4.0 (CHS[2:0] ≠ 000B), don't care GCHS status.

; Clear P4CON.
        B0BCLR      P4CON.0     ; Enable P4.0 digital function.

; Set P4.0 output buffer to avoid glitch.
        B0BSET      P4.0        ; Set P4.0 buffer as "1".
; or
        B0BCLR      P4.0        ; Set P4.0 buffer as "0".

; Enable P4.0 output mode.
        B0BSET      P4M.0       ; Set P4.0 as input mode.
```

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, WDT overflow signal raises and resets MCU. Watchdog clock controlled by code option and the clock source is internal low-speed oscillator.

Watchdog overflow time = 8192 / Internal Low-Speed oscillator (sec).

VDD	*Internal Low RC Freq.*	*Watchdog Overflow Time*
3V	16KHz	512ms
5V	32KHz	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.

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.

0CCH	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.**

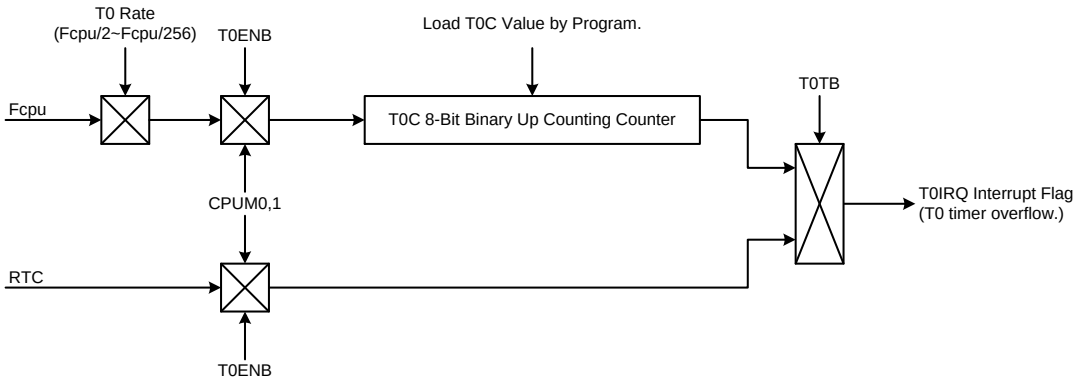
```
Main:
                …                               ; Check I/O.
                …                               ; Check RAM
Err:            JMP $                           ; I/O or RAM error. Program jump here and don't
                                                ; clear watchdog. Wait watchdog 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 and supports RTC function. 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.

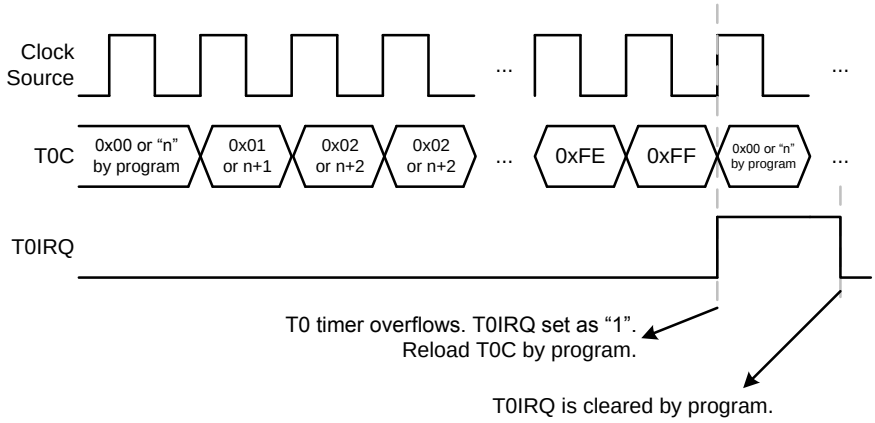
- **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** T0 timer function supports interrupt function. When T0 timer occurs overflow, the T0IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **RTC function:** T0 supports RTC function. The RTC clock source is from external low speed 32K oscillator when T0TB=1. **RTC function is only available in High_Clk code option = "IHRC_RTC".**
- **Green mode function:** T0 timer keeps running in green mode and wakes up system when T0 timer overflows.



✸ ***Note: In RTC mode, the T0 interval time is fixed at 0.5 sec and T0C is 256 counts.***

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 8) 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-scaler 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=4MHz, Fcpu=Fhosc/4		IHRC_RTC mode	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)	max. (sec)	Unit (ms)
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	-	-
-	32768Hz/64	-	-	-	-	0.5	1.953

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.

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB
Read/Write	R/W	R/W	R/W	R/W	-	-	-	R/W
After reset	0	0	0	0	-	-	-	0

Bit 0 **T0TB:** RTC clock source control bit.
0 = Disable RTC (T0 clock source from Fcpu).
1 = Enable RTC.

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.

✸ ***Note: T0RATE is not available in RTC mode. The T0 interval time is fixed at 0.5 sec.***

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.

0D9H	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: To calculation T0C to obtain 10ms T0 interval time. T0 clock source is Fcpu = 4MHz/4 = 1MHz. Select T0RATE=001 (Fcpu/128).**
T0 interval time = 10ms. T0 clock rate = 4MHz/4/128

*T0C initial value = 256 - (T0 interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 128)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 128)*
= B2H

✸ ***Note: In RTC mode, T0C is 256 counts and generatesT0 0.5 sec interval time. Don't change T0C value in RTC mode.***

8.2.5 T0 TIMER OPERATION EXPLAME

- **T0 TIMER CONFIGURATION:**

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

; Set T0 clock source and T0 rate.
        MOV     A, #0nnn00000b
        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.
```

- **T0 works in RTC mode:**

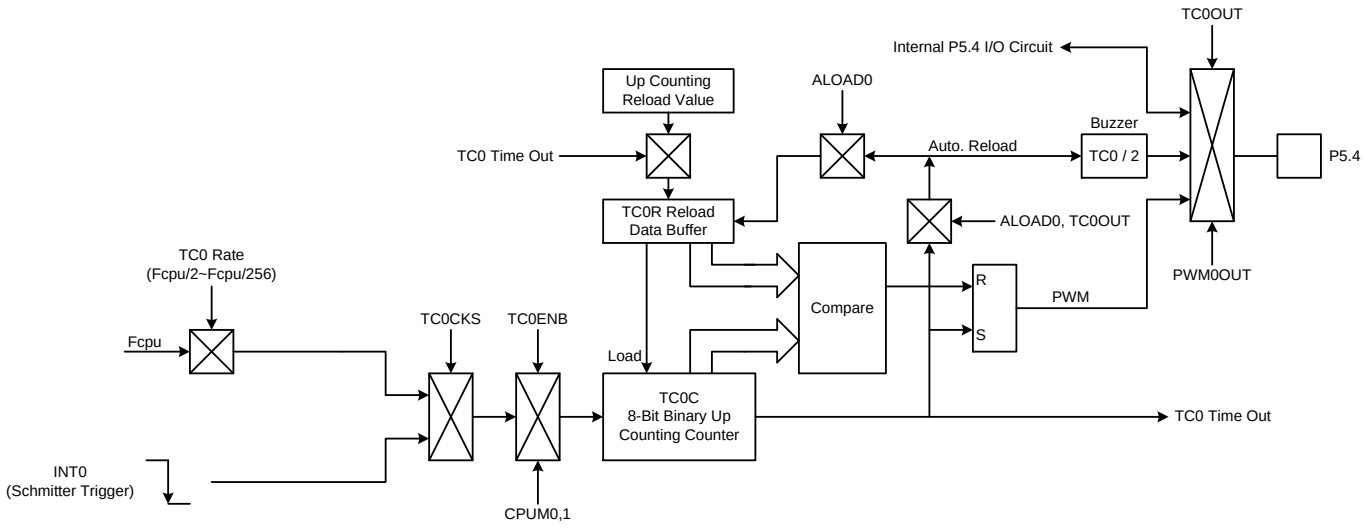
```
; Reset T0 timer.
        MOV     A, #0x00        ; Clear T0M register.
        B0MOV   T0M, A

; Set T0 RTC function.
        B0BSET  FT0TB

; Clear T0C.
        CLR     T0C

; Clear T0IRQ
        B0BCLR  FT0IRQ

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

8.3 TC0 8-BIT TIMER/COUNTER

8.3.1 OVERVIEW

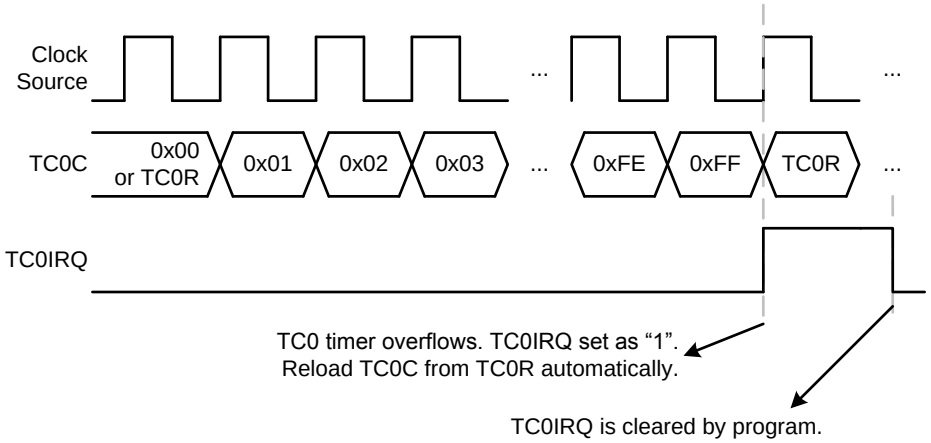
The TC0 timer is an 8-bit binary up timer with basic timer, event counter, buzzer and PWM functions. The basic timer function supports flag indicator (TC0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TC0M, TC0C, TC0R registers. The event counter is changing TC0 clock source from system clock (Fcpu) to external clock like signal (e.g. continuous pulse, R/C type oscillating signal…). TC0 becomes a counter to count external clock number to implement measure application. TC0 also builds in buzzer and PWM functions. The cycle/resolution of buzzer and PWM are controlled by TC0 timer clock rate and TC0R registers, so the buzzer and PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster…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.
- **PWM output:** The PWM is duty/cycle programmable controlled by T0rate and TC0R registers.
- **Buzzer output:** The Buzzer output signal is 1/2 cycle of TC0 interval time.
- **Green mode function:** All TC0 functions (timer, PWM, Buzzer, event counter, auto-reload) keep running in green mode and no wake-up function.



8.3.2 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 controlled by ALOAD0 bit in timer/counter mode, and enabled automatically in PWM mode 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 8) 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. But in green mode, TC0 keep counting, set TC0IRQ and outputs PWM, but can't wake-up system.



TC0 provides different clock sources to implement different applications and configurations. TC0 clock source includes Fcpu (instruction cycle) and external input pin (P0.0) controlled by TC0CKS bits. TC0CKS bit selects the clock source is from Fcpu or external input pin. If TC0CKS=0, TC0 clock source is Fcpu through TC0rate[2:0] pre-scaler to decide Fcpu/2~Fcpu/256. If TC0CKS=1, TC0 clock source is external input pin that means to enable event counter function. TC0rate[2:0] pre-scaler is unless when TC0CKS=1 condition. TC0 length is 8-bit (256 steps) when PWM disabled, and the one count period is each cycle of input clock.

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)
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

8.3.3 TC0M MODE REGISTER

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

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	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 P5.4 is GPIO mode.
1 = Enable PWM output function, and P5.4 outputs PWM signal. PWM duty controlled by TC0OUT, ALOAD0 bits.

Bit 1 **TC0OUT:** TC0 time out toggle signal output control bit. **Only valid when PWM0OUT = 0.**
0 = Disable, P5.4 is I/O function.
1 = Enable, P5.4 is output TC0OUT signal.

Bit 2 **ALOAD0:** Auto-reload control bit. **Only valid when PWM0OUT = 0.**
0 = Disable TC0 auto-reload function.
1 = Enable TC0 auto-reload function.

Bit 3 **TC0CKS:** TC0 clock source select bit.
0 = Internal clock (Fcpu).
1 = External input pin (P0.0/INT0) and enable event counter function. **TC0rate[2:0] bits are useless.**

Bit [6:4] **TC0RATE[2:0]:** TC0 internal clock 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 **TC0ENB:** TC0 counter control bit.
0 = Disable TC0 timer.
1 = Enable TC0 timer.

3. ***Note: When TC0CKS=1, TC0 became an external event counter and TC0RATE is useless. No more P0.0 interrupt request will be raised. (P0.0IRQ will be always 0).***

8.3.4 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.

0DBH	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 = N - (TC0 interrupt interval time * TC0 clock rate)

N is TC0 overflow boundary number. TC0 timer overflow time has five types (TC0 timer, TC0 event counter, TC0 Fcpu clock source, PWM mode and no PWM mode). These parameters decide TC0 overflow time and valid value as follow table.

TC0CKS	PWM0	ALOAD0	TC0OUT	N	TC0C valid value	TC0C value binary type	Remark
0	0	x	x	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
	1	0	0	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
	1	0	1	64	0x00~0x3F	xx000000b~xx111111b	Overflow per 64 count
	1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b	Overflow per 32 count
	1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b	Overflow per 16 count
1	-	-	-	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count

8.3.5 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 effect the correctness of TC0 interval time and PWM output signal.

0CDH	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)

N is TC0 overflow boundary number. TC0 timer overflow time has five types (TC0 timer, TC0 event counter, TC0 Fcpu clock source, PWM mode and no PWM mode). These parameters decide TC0 overflow time and valid value as follow table.

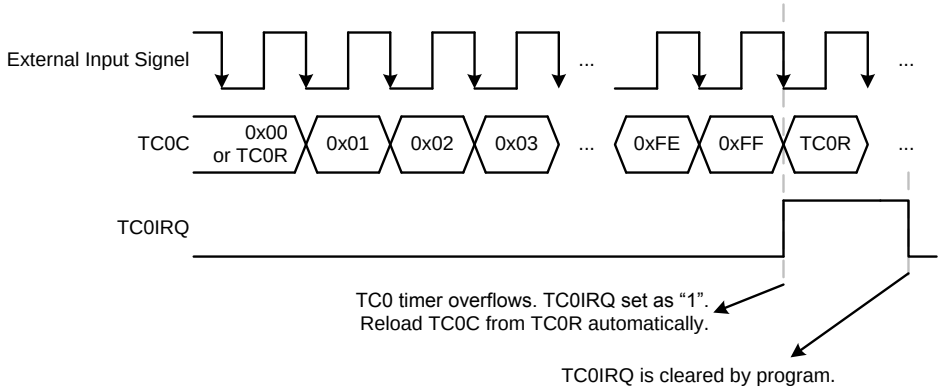
TC0CKS	PWM0	ALOAD0	TC0OUT	N	TC0R valid value	TC0R value binary type
0	0	x	x	256	0x00~0xFF	00000000b~11111111b
	1	0	0	256	0x00~0xFF	00000000b~11111111b
	1	0	1	64	0x00~0x3F	xx000000b~xx111111b
	1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
	1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
1	-	-	-	256	0x00~0xFF	00000000b~11111111b

- **Example: To calculation TC0C and TC0R value to obtain 10ms TC0 interval time. TC0 clock source is Fcpu = 4MHz/4 = 1MHz. Select TC0RATE=001 (Fcpu/128).**
  TC0 interval time = 10ms. TC0 clock rate = 4MHz/4/128

  *TC0C/TC0R initial value = 256 - (TC0 interval time * input clock)*
  *= 256 - (10ms * 4MHz / 4 / 128)*
  *= 256 - (10^{-2} * 4 * 10^{6} / 4 / 128)*
  = B2H

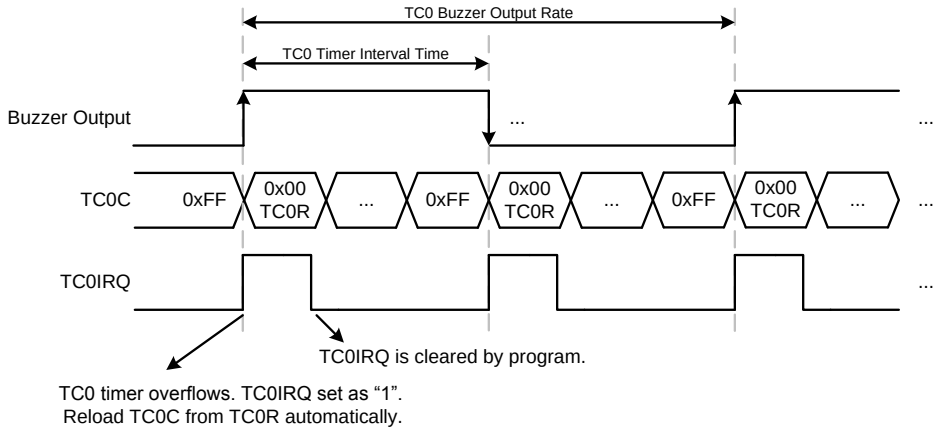
8.3.6 TC0 EVENT COUNTER

TC0 event counter is set the TC0 clock source from external input pin (P0.0). When TC0CKS1=1, TC0 clock source is switch to external input pin (P0.0). 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 P00IRQ 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.

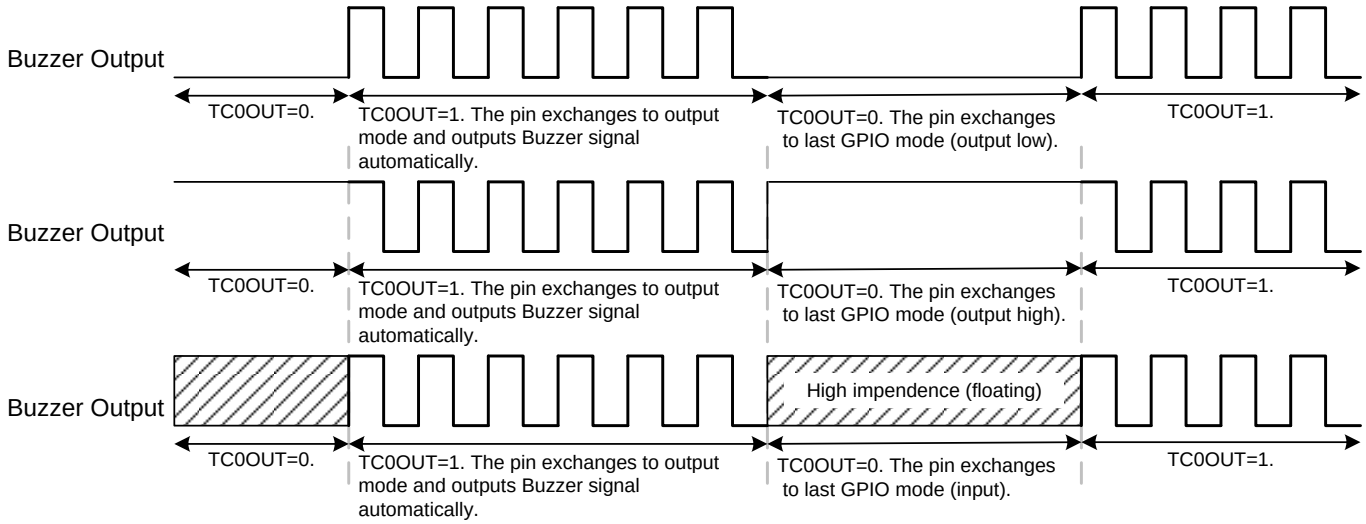


8.3.7 TC0 BUZZER OUTPUT

The buzzer output is a simple 1/2 duty signal output function. The buzzer signal is generated from TC0 timer. When TC0 timer overflows, the buzzer output exchanges status, and generates a square waveform. The frequency of buzzer output is 1/2 of TC0 interval time. The TC0 clock has many combinations and easily to make difference frequency. The buzzer output waveform is as following.



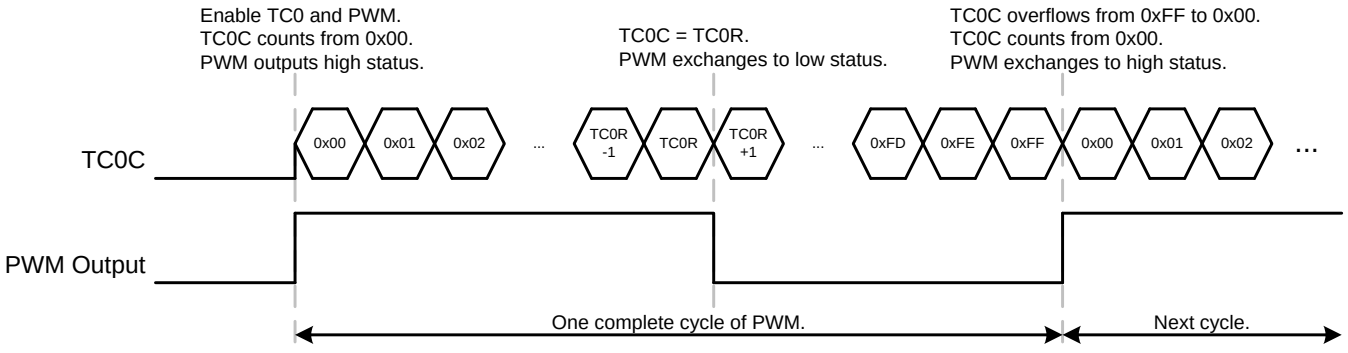
When buzzer outputs, TC0IRQ still actives as TC0 overflows, and TC0 interrupt function actives as TC0IEN = 1. But strongly recommend be careful to use buzzer and TC0 timer together, and make sure both functions work well.
The buzzer output pin is shared with GPIO and switch to output buzzer signal as TC0OUT=1 automatically. If TC0OUT bit is cleared to disable buzzer signal, the output pin exchanges to last GPIO mode automatically. It easily to implement carry signal on/off operation, not to control TC0ENB bit.



4. ***Note: Because the TC0OUT decides the PWM cycle in PWM mode. The PWM0OUT bit must be "0" when buzzer output function works.***

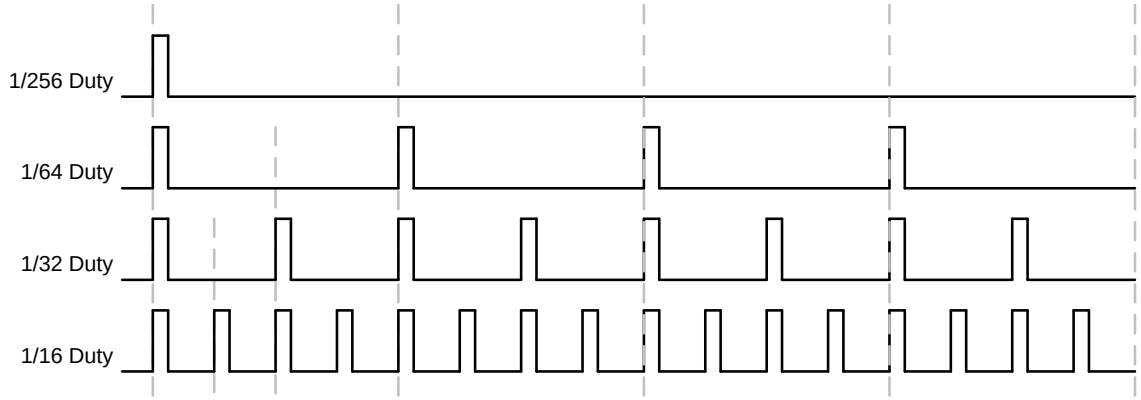
8.3.8 PULSE WIDTH MODULATION (PWM)

The PWM is duty/cycle programmable design to offer various PWM signals. When TC0 timer enables and PWM0OUT bit sets as "1" (enable PWM output), the PWM output pin (P5.4) outputs PWM signal. One cycle of PWM signal is high pulse first, and then low pulse outputs. TC0rate[2:0] bits control the cycle of PWM, ALOAD0 and TC0OUT bits decides the resolution of PWM, and TC0R decides the duty (high pulse width length) of PWM. TC0C initial value is zero when TC0 timer enables and TC0 timer overflows. When TC0C count is equal to TC0R, 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 TC0R when TC0 overflows and the end of PWM's cycle, to keeps PWM continuity. If modify the PWM duty by program as PWM outputting, the new duty occurs at next cycle when TC0R loaded from the reload buffer.

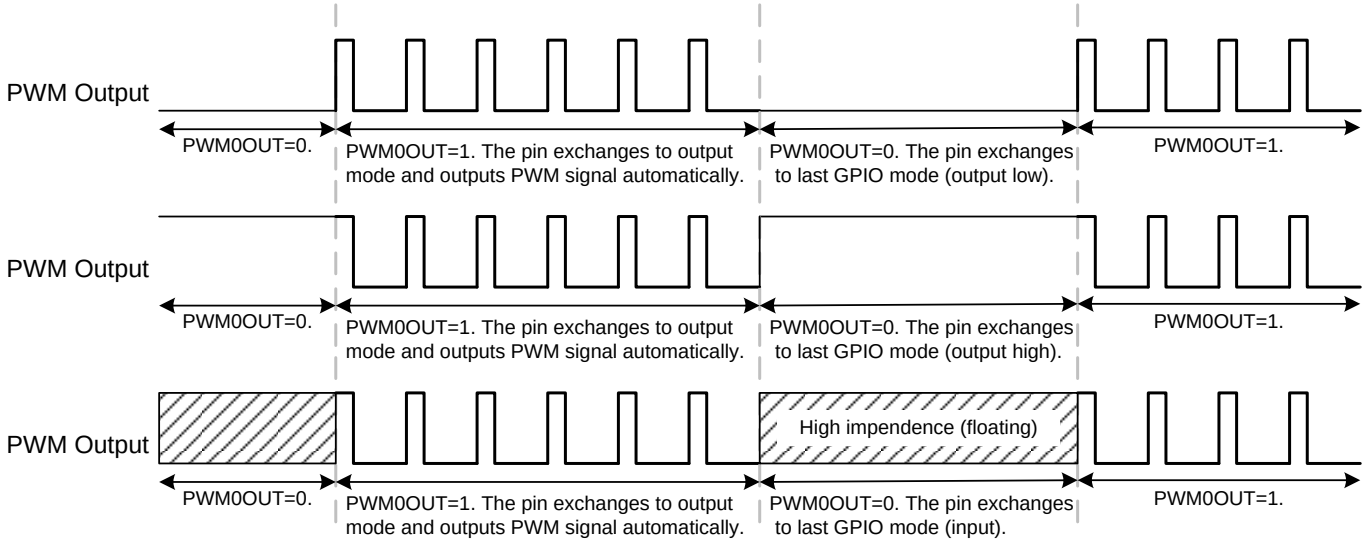


The resolution of PWM includes 1/256, 1/64, 1/32, 1/16 controlled by ALOAD0 and TC0OUT bits to implement high speed PWM signal. ALOAD0, TC0OUT = 00, the PWM resolution is 1/256. ALOAD0, TC0OUT = 01, the PWM resolution is 1/64. ALOAD0, TC0OUT = 10, the PWM resolution is 1/32. ALOAD0, TC0OUT = 11, the PWM resolution is 1/16. If modify the PWM resolution, the TC0R PWM duty control range must be modified to meet resolution. 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.

ALOAD0	TC0OUT	PWM Resolution	TC0R valid value	TC0R value binary type
0	0	256	0x00~0xFF	00000000b~11111111b
0	1	64	0x00~0x3F	xx000000b~xx111111b
1	0	32	0x00~0x1F	xxx00000b~xxx11111b
1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b



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.



8.3.9 TC0 TIMER OPERATION EXPLAME

- **TC0 TIMER CONFIGURATION:**

```
; Reset TC0 timer.
        MOV     A, #0x00        ; Clear TC0M register.
        B0MOV   TC0M, A

; Set TC0 rate and auto-reload function.
        MOV     A, #0nnn0000b   ; TC0rate[2:0] bits.
        B0MOV   TC0M, A
        B0BSET  FALOAD0

; 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.
        MOV     A, #0x00        ; Clear TC0M register.
        B0MOV   TC0M, A

; Set TC0 auto-reload function.
        B0BSET  FALOAD0

; Enable TC0 event counter.
        B0BSET  FTC0CKS         ; Set TC0 clock source from external input pin (P0.0).

; 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 BUZZER OUTPUT CONFIGURATION:**

```
; Reset TC0 timer.
        MOV     A, #0x00        ; Clear TC0M register.
        B0MOV   TC0M, A

; Set TC0 rate and auto-reload function.
        MOV     A, #0nnn0000b   ; TC0rate[2:0] bits.
        B0MOV   TC0M, A
        B0BSET  FALOAD0

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

; Enable TC0 timer and buzzer output function.
        B0BSET  FTC0ENB         ; Enable TC0 timer.
        B0BSET  FTC0OUT         ; Enable TC0 buzzer output function.
```

- **TC0 PWM CONFIGURATION:**

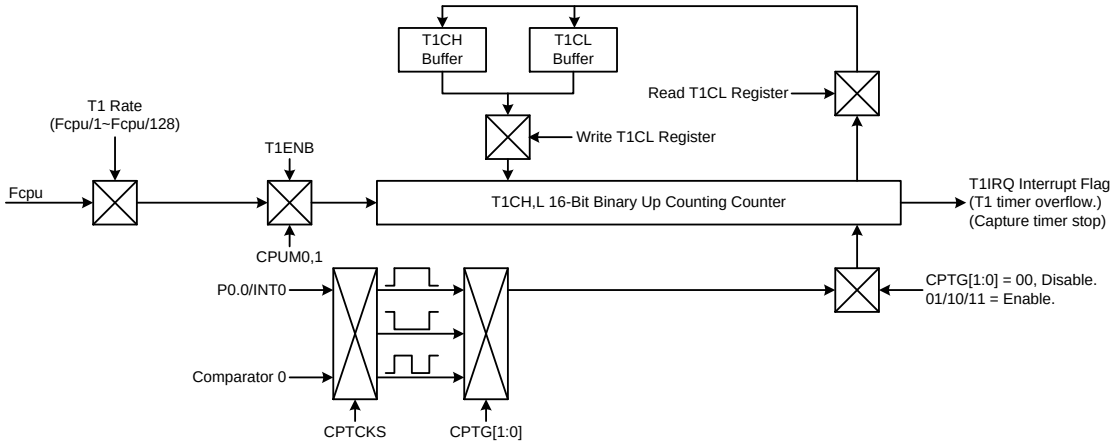
```
; Reset TC0 timer.
        MOV     A, #0x00        ; Clear TC0M register.
        B0MOV   TC0M, A

; Set TC0 rate for PWM cycle.
        MOV     A, #0nnn0000b   ; TC0rate[2:0] bits.
        B0MOV   TC0M, A

; Set PWM resolution.
        MOV     A, #00000nn0b   ; ALOAD0 and TC0OUT bits.
        OR      TC0M, A

; Set TC0R register for PWM duty.
        MOV     A, #value
        B0MOV   TC0R, A

; Clear TC0C as initial value.
        CLR     TC0C

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

8.4 T1 16-BIT TIMER/COUNTER

8.4.1 OVERVIEW

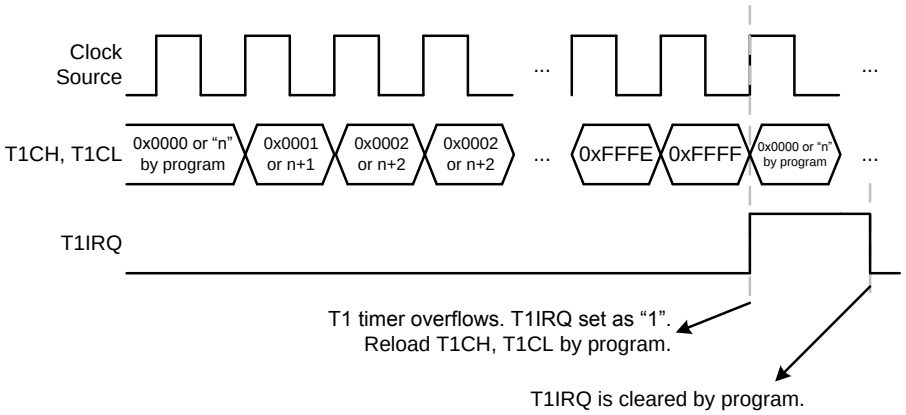
The T1 timer is a 16-bit binary up timer with basic timer and capture timer functions. The basic timer function supports flag indicator (T1IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T1M, T1CH/T1CL 16-bit counter registers. The capture timer supports high pulse width measurement, low pulse width measurement and cycle measurement from P0.0 and comparator output oscillating like signal (e.g. continuous pulse, R/C type oscillating signal…). T1 becomes a timer meter to count external signal time parameters to implement measure application. The main purposes of the T1 timer are as following.

- **16-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** T1 timer function supports interrupt function. When T1 timer occurs overflow, the T1IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **16-bit capture timer:** Measure the input signal pulse width and cycle depend on the T1 clock time base to decide the capture timer's resolution. The capture timer builds in programmable trigger edge selection to decide the start-stop trigger event.
- **Green mode function:** T1 timer keeps running in green mode and wakes up system when T1 timer overflows and issue T1IRQ=1.



8.4.2 T1 TIMER OPERATION

T1 timer is controlled by T1ENB bit. When T1ENB=0, T1 timer stops. When T1ENB=1, T1 timer starts to count. Before enabling T1 timer, setup T1 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…T1 16-bit counter (T1CH, T1CL) increases "1" by timer clock source. When T1 overflow event occurs, T1IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is T1CH, T1CL count from full scale (0xFFFF) to zero scale (0x0000). T1 doesn't build in double buffer, so load T1CH, T1CL by program when T1 timer overflows to fix the correct interval time. If T1 timer interrupt function is enabled (T1IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 8) and executes interrupt service routine after T1 overflow occurrence. Clear T1IRQ by program is necessary in interrupt procedure. T1 timer can works in normal mode, slow mode and green mode. In green mode, T1 keeps counting, set T1IRQ and wakes up system when T1 timer overflows.



T1 clock source is Fcpu (instruction cycle) through T1rate[2:0] pre-scaler to decide Fcpu/1~Fcpu/128. T1 length is 16-bit (65536 steps), and the one count period is each cycle of input clock.

T1rate[2:0]	T1 Clock	T1 Interval Time			
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)
000b	Fcpu/128	2097.152	32	8388.608	128
001b	Fcpu/64	1048.576	16	4194.304	64
010b	Fcpu/32	524.288	8	2097.152	32
011b	Fcpu/16	262.144	4	1048.576	16
100b	Fcpu/8	131.072	2	524.288	8
101b	Fcpu/4	65.536	1	262.144	4
110b	Fcpu/2	32.768	0.5	131.072	2
111b	Fcpu/1	16.384	0.25	65.536	1

8.4.3 T1M MODE REGISTER

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

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1rate2	T1rate1	T1rate0	CPTCKS	CPTStart	CPTG1	CPTG0
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 [1:0] **CPTG[1:0]:** T1 capture timer function control bit.
00 = Disable capture timer function.
01 = High pulse width measurement.
10 = Low pulse width measurement
11 = Cycle measurement.

Bit 2 **CPTStart:** T1 capture timer operating control bit.
0 = Process end.
1 = Start to count and processing.

Bit 3 **CPTCKS:** T1 capture timer input source select bit.
0 = External input pin (P0.0/INT0) and enable capture timer function.
1 = Comparator output terminal and enable capture timer function.

Bit [6:4] **T1RATE[2:0]:** T1 timer clock source select bits.
000 = Fcpu/128, 001 = Fcpu/64, 010 = Fcpu/32, 011 = Fcpu/16, 100 = Fcpu/8, 101 = Fcpu/4, 110 = Fcpu/2, 111 = Fcpu/1.

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

8.4.4 T1CH, T1CL 16-bit COUNTING REGISTERS

T1 counter is 16-bit counter combined with T1CH and T1CL registers. When T1 timer overflow occurs, the T1IRQ flag is set as "1" and cleared by program. The T1CH, T1CL decide T1 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T1CH and T1CL registers, and then enable T1 timer to make sure the fist cycle correct. After one T1 overflow occurs, the T1CH and T1CL registers are loaded correct values by program.

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CL	T1CL7	T1CL6	T1CL5	T1CL4	T1CL3	T1CL2	T1CL1	T1CL0
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
T1CH	T1CH7	T1CH6	T1CH5	T1CH4	T1CH3	T1CH2	T1CH1	T1CH0
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 T1 timer counter length is 16-bit and points to T1CH and T1CL registers. The timer counter is double buffer design. The core bus is 8-bit, so access 16-bit data needs a latch flag to avoid the transient status affect the 16-bit data mistake occurrence. Under write mode, the write T1CL is the latch control flag. Under read mode, the read T1CL is the latch control flag. So, write T1 16-bit counter is to write T1CH first, and then write T1CL. The 16-bit data is written to 16-bit counter buffer after executing writing T1CL. Read T1 16-bit counter is to read T1CL first, and then read T1CH. The 16-bit data is dumped to T1CH/T1CL after executing reading T1CL.

- **Read T1 counter buffer sequence is to read T1CL first, and then read T1CH.**
- **Write T1 counter buffer sequence is to write T1CH first, and then write T1CL.**

The equation of T1 16-bit counter (T1CH, T1CL) initial value is as following.

T1CH, T1CL initial value = 65536 - (T1 interrupt interval time * T1 clock rate)

Example: To calculation T1CH and T1CL values to obtain 500ms T1 interval time. T1 clock source is Fcpu = 16MHz/16 = 1MHz. Select T1RATE=000 (Fcpu/128).

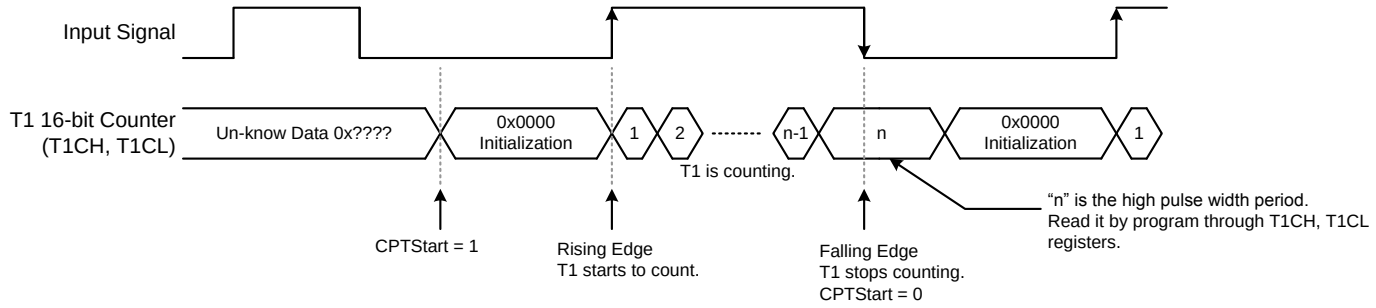
T1 interval time = 500ms. T1 clock rate = 16MHz/16/128

*T1 16-bit counter initial value = 65536 - (T1 interval time * input clock)*
*= 65536 - (500ms * 16MHz / 16 / 128)*
*= 65536 - (500*10^{-3} * 16 * 10^{6} / 16 / 128)*
= F0BDH ***(T1CH = F0H, T1CL = BDH)***

8.4.5 T1 CPATURE TIMER

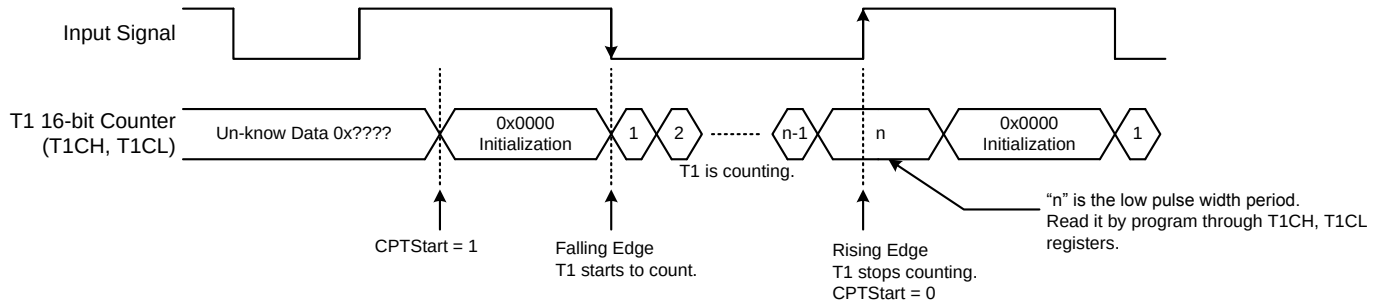
The 16-bit capture timer purpose is to measure input signal pulse width and cycle. The measurement is through T1 timer by trigger selection. The concept is using T1 to measure input signal like a meter. When trigger condition exists, the T1 timer starts and stops following signal condition. The capture timer is controlled by CPTG[1:0] bits. When CPTG[1:0] = 00, the capture timer is disabled. When CPTG[1:0] = 01/10/11, the capture timer is enabled, but the T1ENB must be enabled first. The capture timer can measure input high pulse width, input low pulse width and the cycle of input signal controlled by CPTG[1:0]. CPTG[1:0] = 01, measure input high pulse width. CPTG[1:0] = 10, measure input low pulse width. CPTG[1:0] = 11, measure the cycle of input signal. The CPTG[1:0] only selects the capture timer function, not execute the capture timer. CPTStart bit is capture timer execution control bit. When CPTStart is set as “1”, the capture timer waits the right trigger edge to active 16-bit counter. If the trigger edge finds, the T1 16-bit counter starts to count which clock source is T1. When the second right edge finds, the 16-counter stops, CPTStart is cleared and the T1IRQ actives. Before setting CPTStart bit, the T1 16-bit counter is cleared for capture timer initialization automatically.

- **High Pulse Width Measurement (T1ENB = 1. CPTG[1:0] = 01.)**



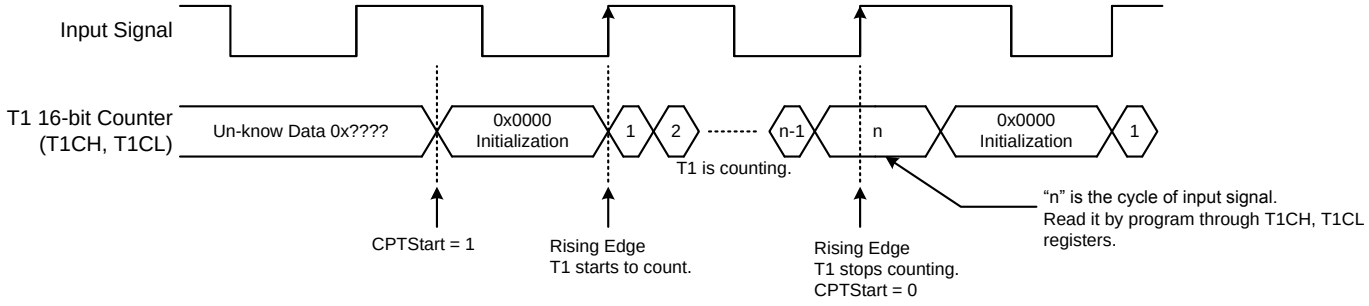
The high pulse width measurement is using rising edge to trigger T1 timer counting and falling edge to stop T1 timer. If set CPTStart bit at high pulse duration, the capture timer will measure next high pulse until the rising edge occurrence. When the falling edge occurs and T1 timer stops, the T1CH, T1CL 16-bit counter is the period of high pulse width.

- **Low Pulse Width Measurement (T1ENB = 1. CPTG[1:0] = 10.)**



The low pulse width measurement is using falling edge to start T1 timer counting and rising edge to stop T1 timer. If set CPTStart bit at low pulse duration, the capture timer will measure next low pulse until the falling edge occurrence. When the rising edge occurs and T1 timer stops, the T1CH, T1CL 16-bit counter is the period of low pulse width.

- **Input Cycle Measurement (T1ENB = 1. CPTG[1:0] = 11.)**



The cycle measurement is using rising edge to start and stop T1 timer. If set CPTStart bit at high or low pulse duration, the capture timer will measure next cycle until the rising edge occurrence. When the rising edge occurs and T1 timer stops, the T1CH, T1CL 16-bit counter is the cycle width.

8.4.6 T1 TIMER OPERATION EXPLAME

- **T1 TIMER CONFIGURATION:**

```
; Reset T1 timer.
        MOV     A, #0x00        ; Clear T1M register.
        B0MOV   T1M, A

; Set T1 clock rate.
        MOV     A, #0nnn0000b   ; T1rate[2:0] bits.
        B0MOV   T1M, A

; Set T1CH, T1CL registers for T1 Interval time.
        MOV     A, #value1      ; Set high byte first.
        B0MOV   T1CH, A
        MOV     A, #value2      ; Set low byte.
        B0MOV   T1CL, A

; Clear T1IRQ
        B0BCLR  FT1IRQ

; Enable T1 timer and interrupt function.
        B0BSET  FT1IEN          ; Enable T1 interrupt function.
        B0BSET  FT1ENB          ; Enable T1 timer.
```

- **T1 CAPTURE TIMER FOR SINGLE CYCLE MEASUREMENT CONFIGURATION:**

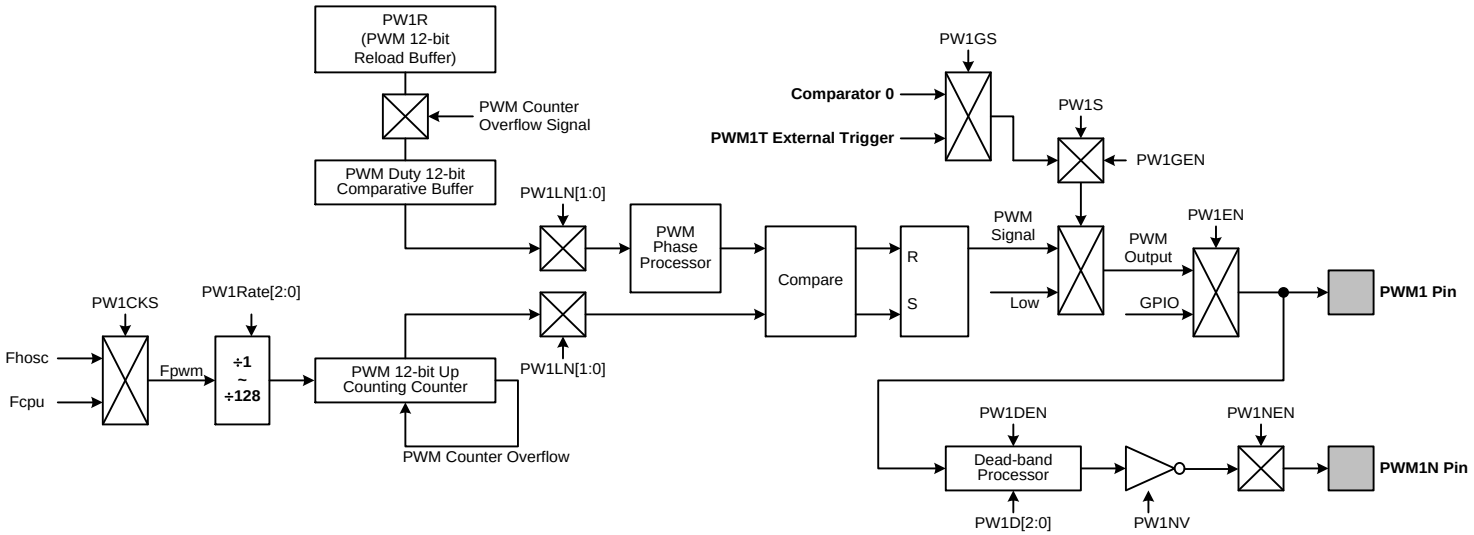
```
; Reset T1 timer.
        MOV     A, #0x00        ; Clear T1M register.
        B0MOV   T1M, A

; Set T1 clock rate, select input source, and select/enable T1 capture timer.
        MOV     A, #0nnn00mmb   ; “nnn” is T1rate[2:0] for T1 clock rate selection.
        B0MOV   T1M, A          ; “mm” is CPTG[1:0] for T1 capture timer selection.
                                ; CPTG[1:0] = 00b, disable T1 capture timer.
                                ; CPTG[1:0] = 01b, high pulse width measurement.
                                ; CPTG[1:0] = 10b, low pulse width measurement.
                                ; CPTG[1:0] = 11b, cycle measurement.

; Select T1 capture source.
        B0BCLR  FCPTCKS         ; Capture source is P0.0.
; or
        B0BSET  FCPTCKS         ; Capture source is comparator output.

; Clear T1CH, T1CL.
        CLR     T1CH            ; Clear high byte first.
        CLR     T1CL            ; Clear low byte.

; Clear T1IRQ
        B0BCLR  FT1IRQ

; Enable T1 timer and interrupt function.
        B0BSET  FT1IEN          ; Enable T1 interrupt function.
        B0BSET  FT1ENB          ; Enable T1 timer.

; Set capture timer start bit.
        B0BSET  FCPTStart
```

9 MULTI-PURPOSE PULSE WIDTH MODULATION (PWM1)

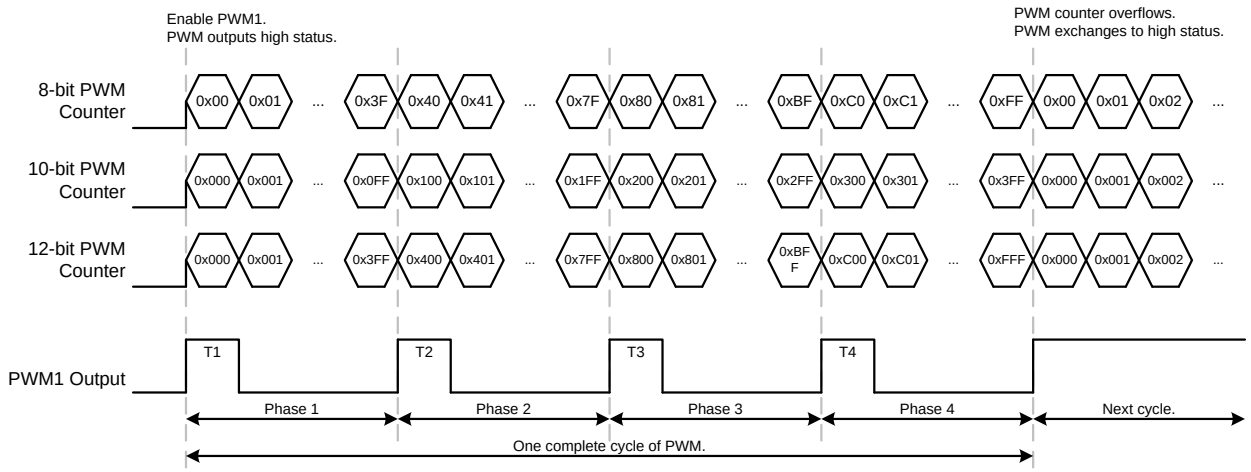
9.1 OVERVIEW

The multi-purpose pulse width modulation (PWM) is a high performance design including programmable high resolution, duty/cycle programmable, high-speed frequency, synchronous trigger function and inverse output function. The PWM resolution is programmable including 8-bit, 10-bit and 12-bit. The PWM cycle is decides by PWM clock source, PWM clock rate selection and PWM resolution. The PWM clock source includes Fcpu and Fhosc. The PWM clock rate includes Fhosc/1~Fhosc/128 and Fcpu/1~Fcpu/128. The PWM uses new technology to implement PWM high speed frequency and reduce the ripple effect of output signal. The inverse output function outputs a complete inverse waveform of normal PWM signal, and also builds in programmable dead-band function. The PWM builds in synchronous trigger function to trigger PWM output. The inverse PWM output with dead-band and synchronous trigger function can implement AC zero-cross processing application. The PWM output pins are shared with GPIO pin. The PWM main functions are as following.

- **High speed-up PWM.**
- **8/10/12-bits programmable resolution with auto reload buffers.**
- **Multi clock source including Fhosc, Fcpu.**
- **Inverse output pin with programmable dead-band function.**
- **Non-inverse output pin with programmable dead-band function.**
- **Synchronous trigger function is to control PWM output. The trigger source is from comparator or external pin. The trigger edge is programmable including rising, falling and bi-direction.**



9.2 PWM1 COMMON OPERATION



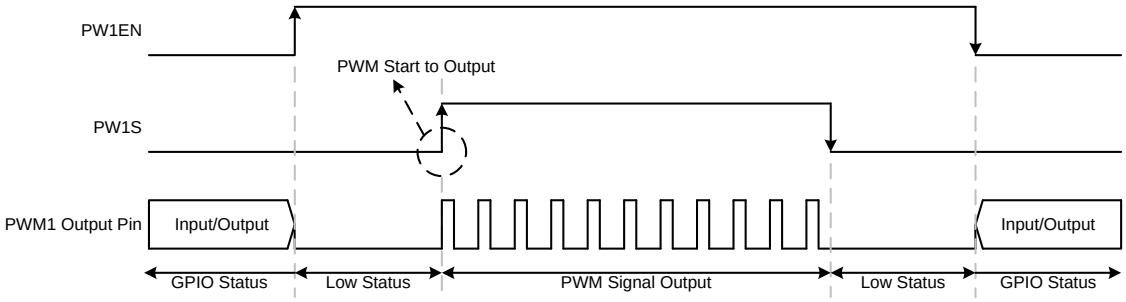
The PWM is four phases design. One cycle of PWM is combined from four sub-cycle signals. PWM signal keeps original cycle parameter, but the frequency is faster. The PWM frequency is equal to sub-cycle's frequency. The duty of PWM is placed in each of sub-cycle. The cycle of the above PWM waveform is "(T1+T2+T3+T4) / (PWM cycle)". The duty output sequence of the four phases is a special design and more balance.

The cycle of PWM is controlled by PWM clock source and PWM clock rate options. PW1CKS bit selects PWM clock source (F_{PWM}) from Fhosc and Fcpu. PW1rate[2:0] bits selects PWM clock rate from $F_{PWM}/1$~$F_{PWM}/128$. So the PWM cycle selection is very flexible.

The PWM reload buffers (PW1RH, PW1RL) decide the duty of PWM and through PWM phase processor to allot the duty to each phase. The PWM designs auto-reload function. If modify the PWM duty by program as PWM outputting, the new duty occurs at next cycle and not change immediately. The methods can avoid the transitional duty occurrence.

The PWM resolution includes 8-bit, 10-bit and 12-bit controlled by PW1LN[1:0] bit. PW1LN[1:0]=00 selects 8-bit PWM. PW1LN[1:0]=01 selects 10-bit PWM. PW1LN[1:0]=10 selects 12-bit PWM. The PWM resolution decision depends on application requirement.

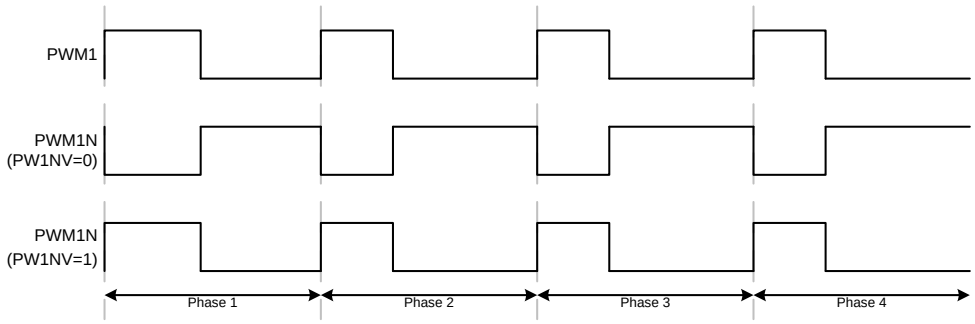
The PWM output is controlled by PW1EN and PW1S bits. PW1EN bit controls PWM1 pin to be GPIO or PWM output pin purpose. When PW1EN = 0, PWM1 pin is GPIO mode. When PW1EN = 1, PWM1 pin changes to PWM output pin and low status for initial status. PW1S bit controls to output PWM signal and useable as PW1EN = 1. When PW1S = 0, disable PWM signal output, and PWM1 pin keeps initial status. When PW1S = 1, PWM signal outputs to PWM1 pin.



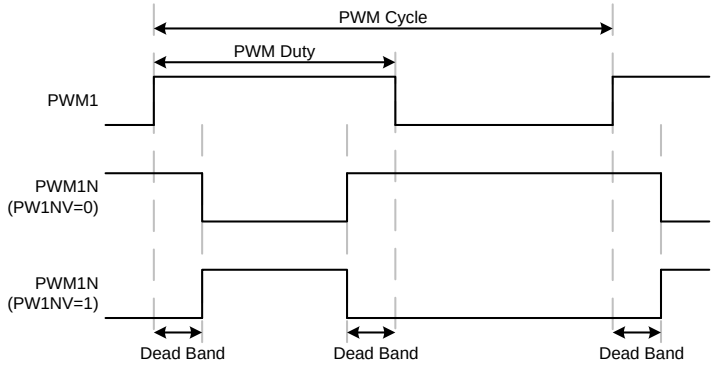
✷ ***Note: When PW1S=1 and PWM starts to output, the first PWM cycle is a complete cycle, and no any delay time or error PWM signal at PW1S rising edge.***

9.3 INVERSE PWM1 OUTPUT WITH DEAD-BAND FUNCTION

The PWM1 builds in inverse output function controlled by PW1NEN bit. When PW1NEN = 0, PWM1N pin is GPIO mode. When PW1NEN = 1, PWM1N pin changes to inverse PWM output pin and used PW1NV bit to select PWM output phase. The PWM signal from PWM1N pin is decided by PW1NV bit as PW1NEN = 1. When PW1NV = 0, the PWM1N pin outputs the inverse PWM signal of PWM1. When PWM1NV = 1, the PWM1N pin outputs the non-inverse PWM signal of PWM1.



The inverse PWM builds in “Dead-Band” function. The inverse PWM signal has a delay time of PWM1 signal.

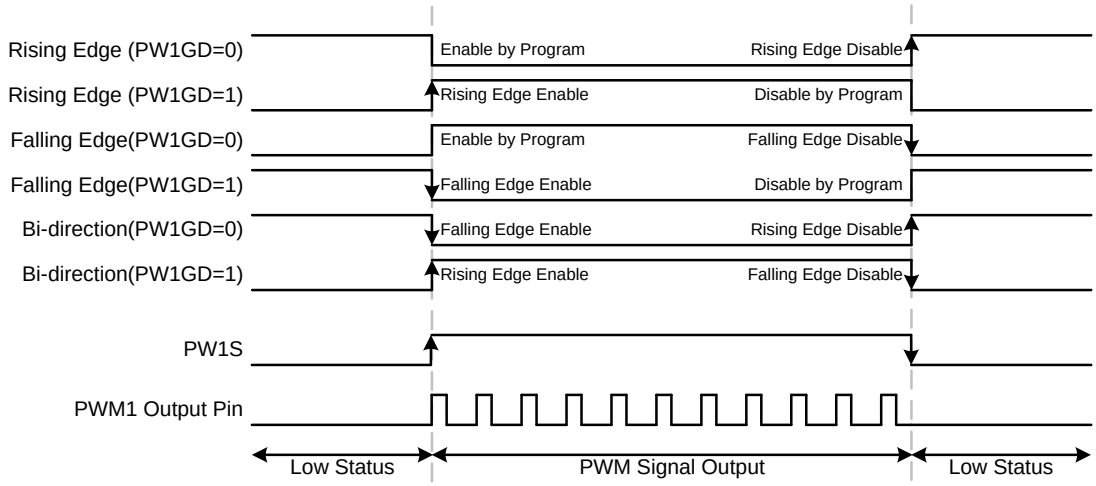


The PWM dead-band function is controlled by PW1DEN bit. When PW1DEN = 0, the PWM1 dead-band function is disabled. When PW1DEN = 1, enable PWM1 dead-ban function. The PW1DEN is only usable when PWM inverse function enables (PW1NEN = 1). The inverse PWM dead-band occurs in PWM1 high pulse width, and the dead-band period is programmable. The dead- band period is symmetrical at left-right terminal of PWM1 high pulse width. If the dead-band period is set as 1*Fpwm, 1*Fpwm dead-band is in the left side of PWM high pulse, and the other side also includes one dead-band duration. So the total dead-band period is 2*Fpwm under 1*Fpwm dead-band configuration. The dead-band period is controlled by PW1D[2:0] bits (000 = 1*Fpwm clock. 001 = 2* Fpwm clock. 010 = 3* Fpwm clock. 011 = 4* Fpwm clock. 100 = 5* Fpwm clock. 101 = 6* Fpwm clock. 110 = 7* Fpwm clock. 111 = 8* Fpwm clock). To take care the PWM high pulse width with dead-bane function is necessary. Recommend the dead-band period less than PWM high pulse width, or the PWM high pulse width disappears.

✸ ***Note: If the bead band period is longer than PWM duty, the PWM1N is no output.***

9.4 PWM SYNCHRONOUS TRIGGER FUNCTION

The PWM1 builds in synchronous trigger function. The trigger sources include MCU's comparator 0 output signal and PW1T external trigger pin. Use the trigger to decide PWM signal output or not. Actually the PWM synchronous trigger function is to control PW1S bit through the trigger source, not program. The PWM synchronous trigger function is controlled by PW1GEN bit. When PW1GEN = 0, the PWM synchronous trigger function is disabled. When PW1GEN = 1, enable PWM synchronous trigger function.



The PWM1 synchronous trigger operation includes 6 types decided through PW1GD bit and the direction selection of trigger source. PW1GD bit selects PWM operation as trigger occurrence. When PW1GED = 0, disable PWM output as trigger occurrence. When PW1GD = 1, enable PWM output as trigger occurrence. If the trigger edge direction is rising and PW1GD = 0, the PWM operation is stop PWM output as rising edge trigger occurrence. The trigger sources builds in trigger edge options including rising edge, falling edge and bi-direction edge. The rising edge and falling edge are single trigger function. When the trigger occurrence, PW1S bit is controlled by trigger source and must be set or cleared by program for the other operation. The bi-direction edge enables and disables PW1S bit. When edge trigger occurrence, PW1S bit is enabled/disable by first edge, and disabled/enabled by second edge automatically. The edge direction is follow trigger source edge configuration. The PW1S is controlled by PWM1 synchronous trigger as below table.

PW1GEN	PW1GD	CM0G[1:0] PW1GS=1	P00G[1:0] PW1GS=0	Edge Direction	PWM Synchronous Trigger Operation	PW1S (PWM Start Bit)
1	0	00	00	No edge.	No trigger source.	By Program
	1					
	0	01	01	Rising edge.	PWM rising edge disable.	0: Edge. 1: Program
	1				PWM rising edge enable.	0: Program 1: Edge
	0	10	10	Falling edge.	PWM falling edge disable.	0: Edge 1: Program
	1				PWM falling edge enable.	0: Program 1: Edge
	0	11	11	Bi-direction.	PWM rising edge disable and falling edge enable.	0/1: Edge.
	1				PWM falling edge disable and rising edge enable.	0/1: Edge.
0	-	-	-	-	Disable PWM synchronous trigger function.	By Program

9.5 PWM1 MODE REGISTER

PW1M and PW1RH registers PWM1 mode control registers to configure PWM operating mode including PWM pre-scaler, clock source, PWM resolution, PWM synchronous trigger function…These configurations must be setup completely before enabling PWM1 function.

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1M	PW1EN	PW1rate2	PW1rate1	PW1rate0	PW1CKS	PW1LN1	PW1LN0	PW1S
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 **PW1EN:** PWM1 control bit.
0 = Disable. PWM1 (P0.1) and PWM1N (P0.2) pins are GPIO mode.
1 = Enable. PWM1 (P0.1) is PWM output pin and low status.

Bit [6:4] **PW1rate[2:0]:** PWM1 clock divide control bit. **Note: Fpwm is PWM1 clock source.**
000 = Fpwm/128, 001 = Fpwm/64, 010 = Fpwm/32, 011 = Fpwm/16, 100 = Fpwm/8, 101 = Fpwm/4, 110 = Fpwm/2, 111 = Fpwm/1.

Bit 3 **PW1CKS:** PWM1 clock source select bit.
0 = Fhosc.
1 = Fcpu.

Bit [2:1] **PW1LN[1:0]:** PWM1 resolution select bit.
00 = 8-bit. 01 = 10-bit. 10 = 12-bit. 11 = Reserved.

Bit 0 **PW1S:** PWM1 start to output control bit. It is controlled by program or PWM synchronous trigger edge.
0 = Disable PWM output. PWM1 output pin is low status.
1 = PWM outputting.

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1RH	PW1GS	PW1GEN	PW1GD	-	PW1R11	PW1R10	PW1R9	PW1R8
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 **PW1GS:** PWM1 synchronous trigger source select bit. The function is workable when PWM synchronous trigger function enabled.
0 = PWM1T pin (P0.0).
1 = Comparator 0 output signal.

Bit 6 **PW1GEN:** PWM1 synchronous trigger function control bit.
0 = Disable.
1 = Enable.

Bit 5 **PW1GD:** PWM1 synchronous trigger operation control bit.
0 = Disable PWM output.
1 = Enable PWM output.

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1NM	PW1NEN	PW1D2	PW1D1	PW1D0	PW1DEN	PW1NV	-	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	-	-
After Reset	0	0	0	0	0	0	-	-

Bit 7 **PW1NEN:** PWM1 invert output function control bit.
0 = Disable.PWM1N pin is P0.2 GPIO mode.
1 = Enable when PW1EN = 1. PWM1N pin is PWM1 invert output pin and isolate P0.2 GPIO function.

Bit [6:4] **PW1D[2:0]:** PWM1 invert output dead-band period select bit.
000 = 1*Fpwm clock. 001 = 2* Fpwm clock. 010 = 3* Fpwm clock. 011 = 4* Fpwm clock.
100 = 5* Fpwm clock. 101 = 6* Fpwm clock. 110 = 7* Fpwm clock. 111 = 8* Fpwm clock.

Bit 3 **PW1DEN:** PWM1 invert output dead-band control bit.
0 = Disable.
1 = Enable.

Bit 2 **PW1NV:** PWM1 invert output control bit.
0 = PW1N pin outputs the inverse signal with dead-band of PWM1.
1 = PW1N pin output signal direction is equal to PWM1 and with dead-band.

9.6 PWM1 DUTY REGISTER

The duty of PWM1 is decided by PW1RH, PW1RL registers. The length of PWM duty registers is 12-bit. The operation is base on PWM counter value and PWM phase processor to generate PWM output signal. The PW1RH, PW1RL duty registers build in auto-reload function. If modify the PWM duty by program as PWM outputting, the new duty occurs at next cycle and not change immediately. The methods can avoid the transitional duty occurrence.

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1RH	PW1GS	PW1GEN	PW1GD	-	PW1R11	PW1R10	PW1R9	PW1R8
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 [3:0] **PW1R [11:8]** = PWM1 duty configuration bit [11:8].

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1RL	PW1R7	PW1R6	PW1R5	PW1R4	PW1R3	PW1R2	PW1R1	PW1R0
Read/Write	W	W	W	W	W	W	W	W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **PW1R [7:0]** = PWM1 duty configuration bit [7:0].

The PWM duty registers length is 12-bit and points to PW1RH and PW1RL registers. The timer counter is double buffer design. The core bus is 8-bit, so access 12-bit data needs a latch flag to avoid the transient status affect the 12-bit data mistake occurrence. Under write mode, the write PW1RL is the latch control flag. Write the 12-bit buffer is to write PW1RH first, and then write PW1RL. The 12-bit data is written to 12-bit buffer after executing writing PW1RL.

➢ **Write PWM 12-bit duty register sequence is to write PW1RH first, and then write PW1RL.**

The PWM duty length is controlled by PWM resolution including 8-bit, 10-bit and 12-bit. If different PWM resolutions, the PWM duty register usable bits are different.

- **8-bit resolution: PWM duty buffer is PW1R0~PW1R7. PW1R8~PW1R11 are set as "0".**
- **10-bit resolution: PWM duty buffer is PW1R0~PW1R9. PW1R10~PW1R11 are set as "0".**
- **12-bit resolution: PWM duty buffer is PW1R0~PW1R11.**

✸ ***Note: PW1R[11:0] written sequence is write PW1R[11:8] first, and then write PW1R[7:0]. End of writing PW1R[7:0] signal means the end of PW1R written.***

9.7 PWM1 OPERATION EXPLAME

- **PWM1 CONFIGURATION:**

```
; Reset PWM1.
        CLR     PW1M            ; Clear PWM1 mode registers.

; Set PWM1 clock source, clock rate and resolution.
        MOV     A, #0mmmnll0b   ; “mmm” is PWM clock rate selection.
        B0MOV   PW1M, A         ; “n” is PWM clock source selection.
                                ; “ll” is PWM resolution selection.

; Set the duty of PWM1.
        MOV     A, #value1      ; Set high byte first.
        B0MOV   PW1RH, A
        MOV     A, #value2      ; Set low byte.
        B0MOV   PW1RL, A

; Enable PWM1 function.
        B0BSET  FPW1EN

; Output PWM signal.
        B0BSET  FPW1S
```

- **ENABLE PWM1 SYNCHRONOUS TRIGGER FUNCTION:**
  (Configure PW1RH[5:7] bits for PWM1 synchronous trigger function before enabling PWM1.)

```
; PWM1 configuration is referred to above example.
        …

; Set PWM1 synchronous trigger source and trigger operation.
        MOV     A, #n0m000000b  ; “n” is PWM1 synchronous trigger source selection.
        OR      PW1RH, A        ; “m” is PWM1 synchronous trigger operation selection.

; Enable PWM1 synchronous trigger function.
        B0BSET  FPW1GEN

; Enable PWM1 function.
        B0BSET  FPW1EN

; Output PWM signal.
        B0BSET  FPW1S
```

- **ENABLE PWM1 INVERSE OUTPUT FUNCTION:**
  (Configure PW1NM mode register for PWM1 inverse output function before enabling PWM1.)

```
; PWM1 configuration is referred to above example.
        …

; Reset PW1NM register.
        CLR     PW1NM

; Select PWM1N pin output signal type.
        B0BCLR  FPW1NV      ; PWM1N pin outputs inverse signal of PWM1.
or
        B0BSET  FPW1NV      ; PWM1N pin outputs non-inverse signal of PWM1.

; Enable PWM1 inverse output function.
        B0BSET  FPW1NEN

; Enable PWM1 function.
        B0BSET  FPW1EN

; Output PWM signal.
        B0BSET  FPW1S
```

- **ENABLE PWM1 DEAD-BAND FUNCTION:**
  (Configure PW1NM mode register for PWM1 dead-band function before enabling PWM1.)

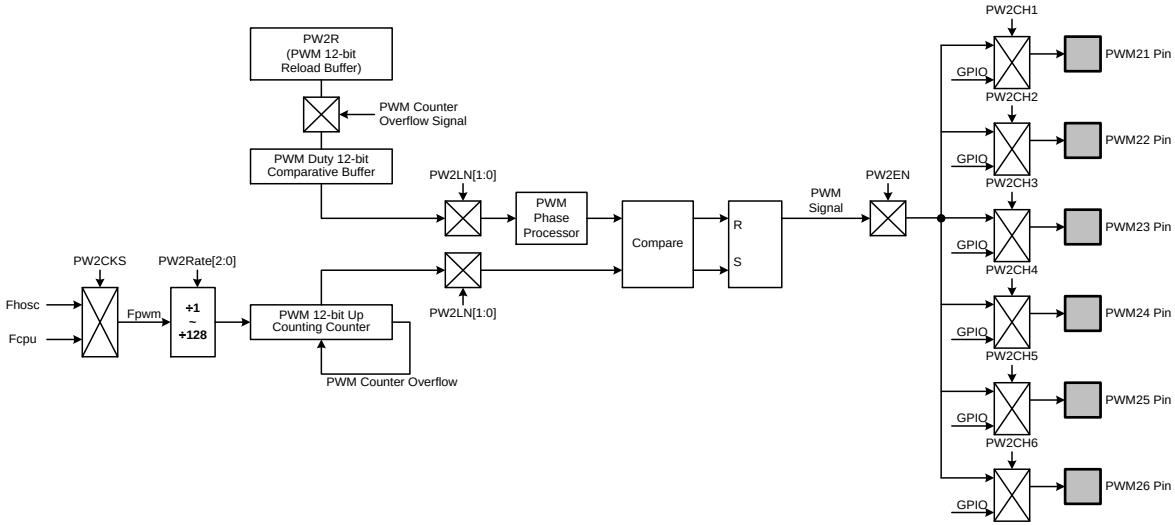
```
; PWM1 configuration is referred to above example.
        …

; PWM1 inverse output function configuration is referred to above example.
        …

; Set PWM1 dead-band period.
        MOV     A, #0nnn0000b   ; "nnn" is PW1D[2:0] dead-band period selection.
        OR      PW1NM, A

; Enable PWM1 dead-band function.
        B0BSET  FPW1DEN

; Enable PWM1 inverse output function.
        B0BSET  FPW1NEN

; Enable PWM1 function.
        B0BSET  FPW1EN

; Output PWM signal.
        B0BSET  FPW1S
```

10 6-CHANNEL PULSE WIDTH MODULATION (PWM2)

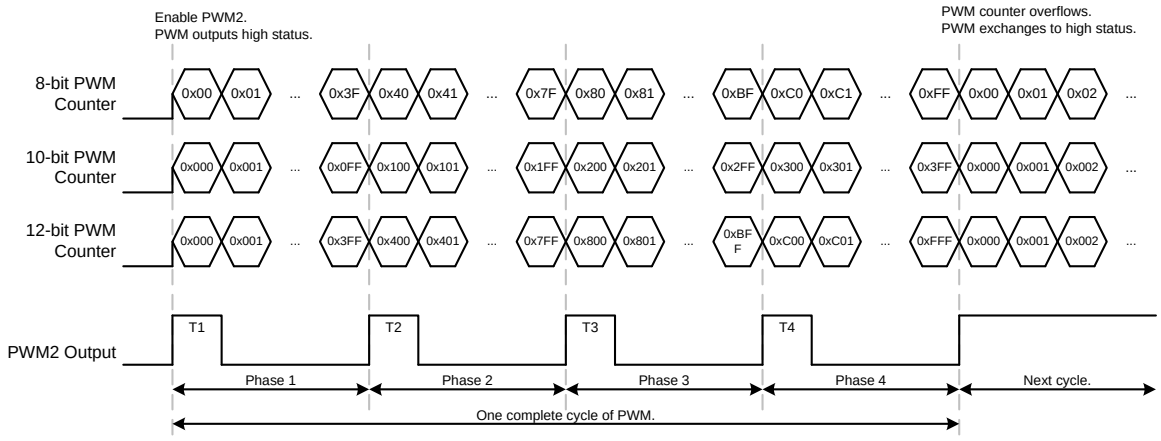
10.1 OVERVIEW

The 6-channel pulse width modulation (PWM) is a high performance design including programmable high resolution, duty/cycle programmable and high-speed frequency. The PWM resolution is programmable including 8-bit, 10-bit and 12-bit. The PWM cycle is decides by PWM clock source, PWM clock rate selection and PWM resolution. The PWM clock source includes Fcpu and Fhosc. The PWM clock rate includes Fhosc/1~Fhosc/128 and Fcpu/1~Fcpu/128. The PWM uses new technology to implement PWM high speed frequency and reduce the ripple effect of output signal. The PWM output pins are shared with GPIO pin controlled by PW2CHS register. If the bit of PW2CHS register enables, the PWM channel exchanges from GPIO to PWM output. When the bit of PW2CHS register disables, the PWM channel returns to GPIO mode and last status. The 6-channel easily implements 3-phase DC motor control, e.g. BLDC motor…The PWM main functions are as following.

- **High speed-up PWM.**
- **8/10/12-bits programmable resolution with auto reload buffers.**
- **Multi clock source including Fhosc, Fcpu.**
- **6-channel individual output pins.**



10.2 PWM2 COMMON OPERATION



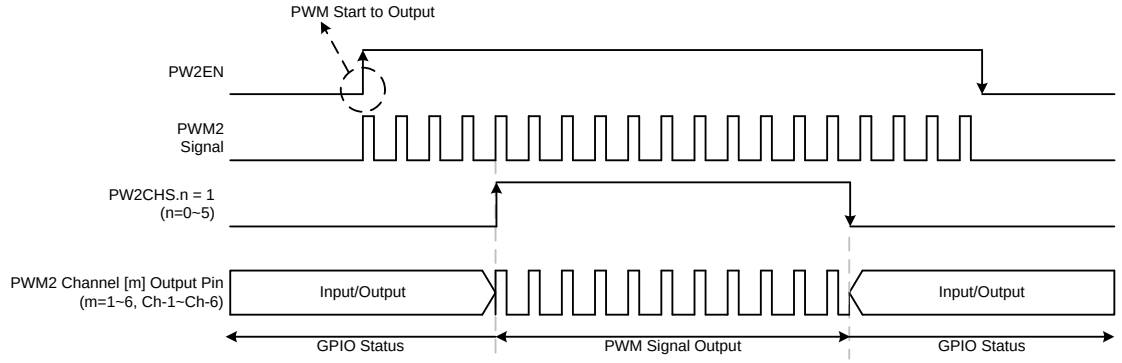
The PWM is four phases design. One cycle of PWM is combined from four sub-cycle signals. PWM signal keeps original cycle parameter, but the frequency is faster. The PWM frequency is equal to sub-cycle's frequency. The duty of PWM is placed in each of sub-cycle. The cycle of the above PWM waveform is "(T1+T2+T3+T4) / (PWM cycle)". The duty output sequence of the four phases is a special design and more balance.

The cycle of PWM is controlled by PWM clock source and PWM clock rate options. PW2CKS bit selects PWM clock source (F_{PWM}) from Fhosc and Fcpu. PW2rate[2:0] bits selects PWM clock rate from $F_{PWM}/1$~$F_{PWM}/128$. So the PWM cycle selection is very flexible.

The PWM reload buffers (PW2RH, PW2RL) decide the duty of PWM and through PWM phase processor to allot the duty to each phase. The PWM designs auto-reload function. If modify the PWM duty by program as PWM outputting, the new duty occurs at next cycle and not change immediately. The methods can avoid the transitional duty occurrence.

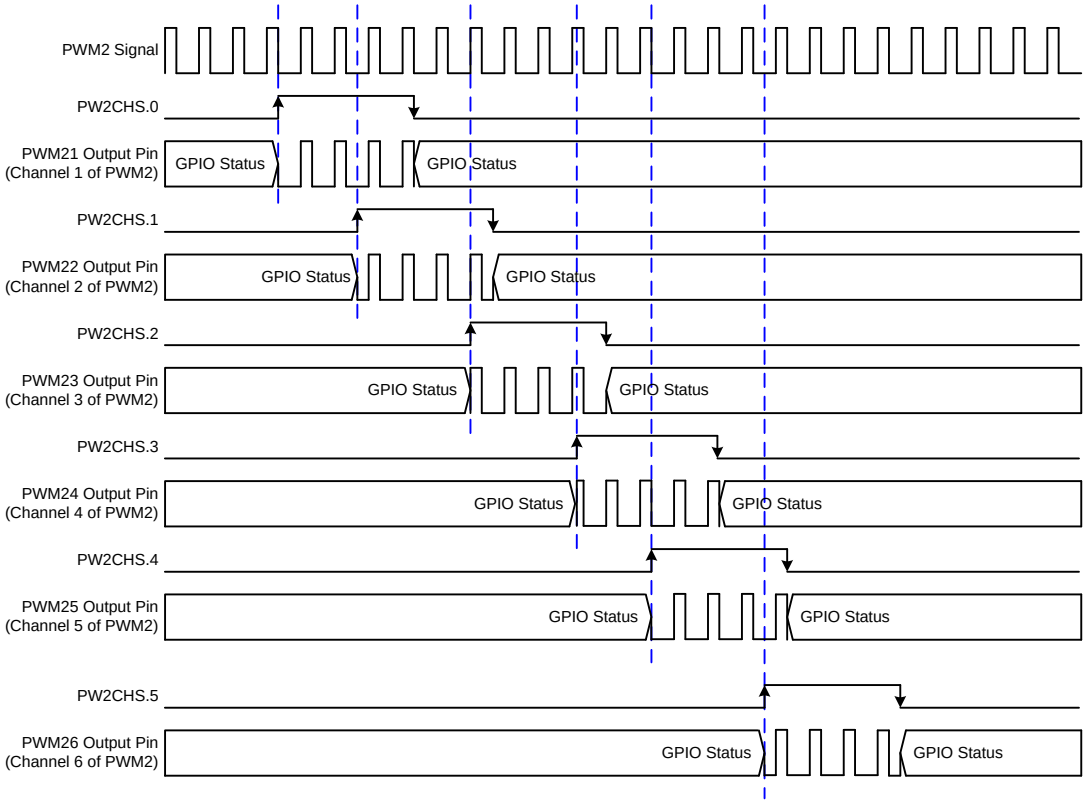
The PWM resolution includes 8-bit, 10-bit and 12-bit controlled by PW2LN[1:0] bit. PW2LN[1:0]=00 selects 8-bit PWM. PW2LN[1:0]=01 selects 10-bit PWM. PW2LN[1:0]=10 selects 12-bit PWM. The PWM resolution decision depends on application requirement.

The PWM output is controlled by PW2EN bit. PW2EN bit controls PWM2 function. When PW2EN = 0, the PWM2 is disabled. When PW2EN = 1, the PWM2 outputs PWM signal. PW2CHS bits control PWM output pin to be GPIO or PWM output pin purpose. PW2CHS bit 0~6 point to each channel of 6-channel PWM (PWM21~PWM26). When PW2CHS.n = 0, the channel "n" of PWM2 pins is GPIO mode. When PW2CHS.n = 1, the channel "n" of PWM2 pins changes to PWM output pin. PW2CHS register is useable as PW2EN = 1.



The GPIO mode of PWM2 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's 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 (PWM21~PWM26) through PWM2 channel selections. The PWM signal of internal source and external pins are the same. The channel selections only switch PWM2 channels and not process the phase of PWM signal.



10.3 PWM2 MODE REGISTER

PW2M register is PWM2 mode control registers to configure PWM operating mode including PWM pre-scaler, clock source, PWM resolution…These configurations must be setup completely before enabling PWM2 function.

094H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW2M	PW2EN	PW2rate2	PW2rate1	PW2rate0	PW2CKS	PW2LN1	PW2LN0	-
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 **PW2EN:** PWM2 control bit.
0 = Disable.
1 = Enable.

Bit [6:4] **PW2rate[2:0]:** PWM2 clock divide control bit. **Note: Fpwm is PWM2 clock source.**
000 = Fpwm/128, 001 = Fpwm/64, 010 = Fpwm/32, 011 = Fpwm/16, 100 = Fpwm/8, 101 = Fpwm/4, 110 = Fpwm/2, 111 = Fpwm/1.

Bit 3 **PW2CKS:** PWM2 clock source select bit.
0 = Fhosc.
1 = Fcpu.

Bit [2:1] **PW2LN[1:0]:** PWM2 resolution select bit.
00 = 8-bit. 01 = 10-bit. 10 = 12-bit. 11 = Reserved.

10.4 PWM2 CHANNEL SELECTION REGISTER

PWM2 includes 6 channels selected through PW2CHS register. If the related bit of PW2CHS register is enabled, the related pin outputs PWM2 signal. If the bit is disabled, the pin returns to last GPIO mode.

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW2CHS	-	-	PW2CH6	PW2CH5	PW2CH4	PW2CH3	PW2CH2	PW2CH1
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	0	0	0	0	0	0

Bit 5 **PW2CH6:** PWM2 channel 6 control bit.
0 = Disable PWM output. PWM26 pins is P5.7 GPIO status.
1 = PWM26 outputs PWM signal and isolate GPIO mode.

Bit 4 **PW2CH5:** PWM2 channel 5 control bit.
0 = Disable PWM output. PWM25 pins is P5.6 GPIO status.
1 = PWM25 outputs PWM signal and isolate GPIO mode.

Bit 3 **PW2CH4:** PWM2 channel 4 control bit.
0 = Disable PWM output. PWM24 pins is P5.5 GPIO status.
1 = PWM24 outputs PWM signal and isolate GPIO mode.

Bit 2 **PW2CH3:** PWM2 channel 3 control bit.
0 = Disable PWM output. PWM23 pins is P5.3 GPIO status.
1 = PWM23 outputs PWM signal and isolate GPIO mode.

Bit 1 **PW2CH2:** PWM2 channel 2 control bit.
0 = Disable PWM output. PWM22 pins is P5.2 GPIO status.
1 = PWM22 outputs PWM signal and isolate GPIO mode.

Bit 0 **PW2CH1:** PWM2 channel 1 control bit.
0 = Disable PWM output. PWM21 pins is P5.1 GPIO status.
1 = PWM21 outputs PWM signal and isolate GPIO mode.

✸ ***Note: PW2CHS register is workable only when PW2EN = 1, or the PWM output pins are GPIO mode.***

10.5 PWM2 DUTY REGISTER

The duty of PWM2 is decided by PW2RH, PW2RL registers. The length of PWM duty registers is 12-bit. The operation is base on PWM counter value and PWM phase processor to generate PWM output signal. The PW2RH, PW2RL duty registers build in auto-reload function. If modify the PWM duty by program as PWM outputting, the new duty occurs at next cycle and not change immediately. The methods can avoid the transitional duty occurrence.

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW2RH	-	-	-	-	PW2R11	PW2R10	PW2R9	PW2R8
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After Reset	-	-	-	-	0	0	0	0

Bit [3:0] **PW2R [11:8]** = PWM2 duty configuration bit [11:8].

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW2RL	PW2R7	PW2R6	PW2R5	PW2R4	PW2R3	PW2R2	PW2R1	PW2R0
Read/Write	W	W	W	W	W	W	W	W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **PW2R [7:0]** = PWM2 duty configuration bit [7:0].

The PWM duty registers length is 12-bit and points to PW2RH and PW2RL registers. The timer counter is double buffer design. The core bus is 8-bit, so access 12-bit data needs a latch flag to avoid the transient status affect the 12-bit data mistake occurrence. Under write mode, the write PW2RL is the latch control flag. Write the 12-bit buffer is to write PW2RH first, and then write PW2RL. The 12-bit data is written to 12-bit buffer after executing writing PW2RL.

➢ **Write PWM 12-bit duty register sequence is to write PW2RH first, and then write PW2RL.**

The PWM duty length is controlled by PWM resolution including 8-bit, 10-bit and 12-bit. If different PWM resolutions, the PWM duty register usable bits are different.

- **8-bit resolution: PWM duty buffer is PW2R0~PW2R7. PW2R8~PW2R11 are set as "0".**
- **10-bit resolution: PWM duty buffer is PW2R0~PW2R9. PW2R10~PW2R11 are set as "0".**
- **12-bit resolution: PWM duty buffer is PW2R0~PW2R11.**

✸ ***Note: PW2R[11:0] written sequence is write PW2R[11:8] first, and then write PW2R[7:0]. End of writing PW2R[7:0] signal means the end of PW2R written.***

10.6 PWM2 OPERATION EXPLAME

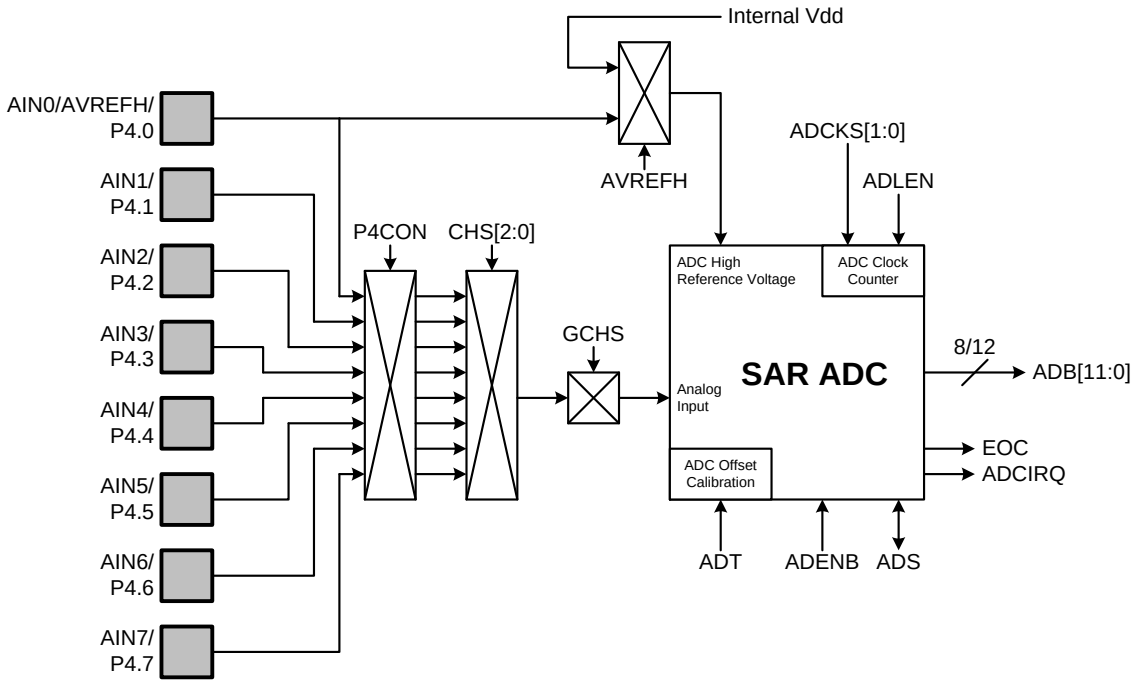
- **PWM2 CONFIGURATION:**

```
; Reset PWM2.
        CLR     PW2M            ; Clear PWM2 mode registers.

; Set PWM2 clock source, clock rate and resolution.
        MOV     A, #0mmmnll0b   ; "mmm" is PWM clock rate selection.
        B0MOV   PW2M, A         ; "n" is PWM clock source selection.
                                ; "ll" is PWM resolution selection.

; Set the duty of PWM2.
        MOV     A, #value1      ; Set high byte first.
        B0MOV   PW2RH, A
        MOV     A, #value2      ; Set low byte.
        B0MOV   PW2RL, A

; Set PWM's idle status.
        MOV     A, #11101110b   ; PWM21~PWM26 high idle status.
        OR      P5, A
        MOV     A, #11101110b
        OR      P5M, A
or
        MOV     A, #00010001b   ; PWM21~PWM26 low idle status.
        AND     P5, A
        MOV     A, #11101110b
        OR      P5M, A

; Enable PWM2 function.
        B0BSET  FPW2EN

; Output PWM signal.
        B0BSET  FPW2CH1         ; Enable PWM21 output.
        B0BSET  FPW2CH2         ; Enable PWM22 output.
        B0BSET  FPW2CH3         ; Enable PWM23 output.
        B0BSET  FPW2CH4         ; Enable PWM24 output.
        B0BSET  FPW2CH5         ; Enable PWM25 output.
        B0BSET  FPW2CH6         ; Enable PWM26 output.
```

11 8 CHANNEL ANALOG TO DIGITAL CONVERTER (ADC)

11.1 OVERVIEW

The analog to digital converter (ADC) is SAR structure with 8-input sources and up to 4096-step resolution to transfer analog signal into 12-bits digital buffers. The ADC builds in 8-channel input source (AIN0~AIN7) to measure 8 different analog signal sources controlled by CHS[2:0] 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 high voltage includes two sources controlled by AVREFH bit. One is internal Vdd (AVREFH=0), and the other one is external reference voltage input pin from P4.0 pin (AVREFH=1). The ADC builds in P4CON register to set pure analog input pin. It is necessary to set P4 as input mode with 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 8) and executes interrupt service routine after finishing ADC converting. Clear ADCIRQ by program is necessary in interrupt procedure.



11.2 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.

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	AVREFH	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 **AVREFH:** ADC high reference voltage source control bit.
0 = Internal Vdd. P4.0 is GPIO or AIN0 pin.
1 = Enable external reference voltage from P4.0.

Bit [2:0] **CHS[2:0]:** ADC input channel select bit.
000 = AIN0, 001 = AIN1, 010 = AIN2, 011 = AIN3, 100 = AIN4, 101 = AIN5, 110 = AIN6, 111 = AIN7

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.

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	-	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	-	R/W	R/W	R/W	R	R	R	R
After reset	-	0	0	0	-	-	-	-

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

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

11.3 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.**

0B2H	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.

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	-	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	-	R/W	R/W	R/W	R	R	R	R
After reset	-	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.***

11.4 ADC OPERATION DESCRIPTION AND NOTIC

11.4.1 ADC SIGNAL FORMAT

ADC sampling voltage range is limited by high/low reference voltage. The ADC low reference voltage is Vss and not changeable. The ADC high reference voltage includes internal Vdd and external reference voltage source from P4.0/AVREFH pin controlled by AVREFH bit. If AVREFH=0, ADC reference voltage is from internal Vdd (MCU power voltage). If AVREFH=1, ADC reference voltage is from external voltage source (P4.0/AVREFH). ADC reference voltage range limitation is **"(ADC high reference voltage – low reference voltage) ≧ 2V"**. ADC low reference voltage is Vss = 0V. So **ADC high reference voltage range is 2V~Vdd**. The range is ADC external high reference voltage range.

- **ADC Internal Low Reference Voltage = 0V.**
- **ADC Internal High Reference Voltage = Vdd. (AVREFH=0)**
- **ADC External High Reference Voltage = 2V~Vdd. (AVREFH=1)**

ADC sampled input signal voltage must be from ADC low reference voltage to ADC high reference. If the ADC input signal voltage is over the range, the ADC converting result is error (full scale or zero).

- **ADC Low Reference Voltage ≦ ADC Sampled Input Voltage ≦ ADC High Reference Voltage**

11.4.2 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 Fcpu and includes Fcpu/1, Fcpu/2, Fcpu/8 and Fcpu/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	Fcpu=4MHz		Fcpu=16MHz	
			ADC Converting time	ADC Converting Rate	ADC Converting time	ADC Converting Rate
1 (12-bit)	00	Fcpu/16	1/(4MHz/16/4)*16 = 256 us	3.906KHz	1/(16MHz/16/4)*16 = 64 us	15.625KHz
	01	Fcpu/8	1/(4MHz/8/4)*16 = 128 us	7.813KHz	1/(16MHz/8/4)*16 = 32 us	31.25KHz
	10	Fcpu	1/(4MHz/4)*16 = 16 us	62.5KHz	1/(16MHz/4)*16 = 4 us	250KHz
	11	Fcpu/2	1/(4MHz/2/4)*16 = 32 us	31.25KHz	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	Fcpu=4MHz		Fcpu=16MHz	
			ADC Converting time	ADC Converting Rate	ADC Converting time	ADC Converting Rate
0 (8-bit)	00	Fcpu/16	1/(4MHz/16/4)*12 = 192 us	5.208KHz	1/(16MHz/16/4)*12 = 48 us	20.833KHz
	01	Fcpu/8	1/(4MHz/8/4)*12 = 96 us	10.416KHz	1/(16MHz/8/4)*12 = 24 us	41.667KHz
	10	Fcpu	1/(4MHz/4)*12 = 12 us	83.333KHz	1/(16MHz/4)*12 = 3 us	333.333KHz
	11	Fcpu/2	1/(4MHz/2/4)*12 = 24 us	41.667KHz	1/(16MHz/2/4)*12 = 6 us	166.667KHz

11.4.3 ADC PIN CONFIGURATION

ADC input channels are shared with Port4. ADC channel selection is through ADCHS[2:0] bit. ADCHS[2:0] value points to the ADC input channel directly. ADCHS[2:0]=000 selects AIN0. ADCHS[2:0]=001 selects AIN1……Only one pin of port 4 can be configured as ADC input in the same time. The pins of Port4 configured as ADC input channel must be set input mode, disable internal pull-up and enable P4CON first by program. After selecting ADC input channel through ADCHS[2:0], 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.**
- **P4CON bits of ADC input channel must be set.**

The P4.0/AIN0 can be ADC external high reference voltage input pin when AVREFH=1. In the condition, P4.0 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 port 4 will encounter above current leakage situation. 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.

0AEH	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[4:0] **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.

✷ ***Note: When Port 4.n is general I/O port not ADC channel, P4CON.n must set to "0" or the Port 4.n digital I/O signal would be isolated.***

11.4.4 ADC OPERATION EXAMLPE

- **ADC CONFIGURATION:**

```
; Reset ADC.
        CLR     ADM             ; Clear TC0M 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.
        B0BCLR  FAVREFH         ; Internal Vdd.
or
        B0BSET  FAVREFH         ; 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

; Enable ADC.
        B0BSET  FADCENB

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

; Select ADC input channel.
        MOV     A, #value       ; Set ADCHS[2:0] 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.

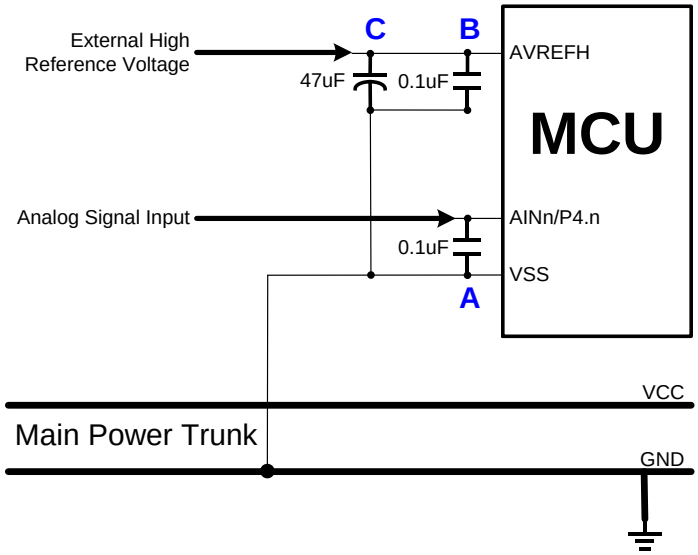
```
@@:
        B0BTS1      FEOC        ; Check ADC processing flag.
        JMP         @B          ; EOC=0: ADC is processing.
        B0MOV       A, ADB      ; EOC=1: End of ADC processing. Process ADC result.
        B0MOV       BUF1,A
        MOV         A, #00001111b
        AND         A, ADR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

; ADC Interrupt enable mode.

```
        ORG 8                   ; Interrupt vector.
INT_SR:                         ; Interrupt service routine.
        PUSH
        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:
        POP
        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 ADS bit by program.***

11.5 ADC APPLICATION CIRCUIT

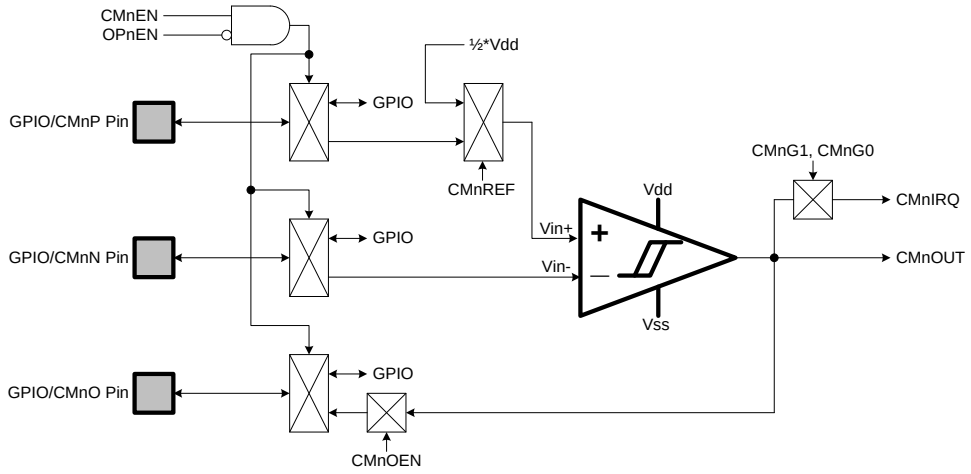


The analog signal is inputted to ADC input pin “AINn/P4.n”. The ADC input signal must be through a 0.1uF capacitor “A”. The 0.1uF capacitor is set between ADC input pin and VSS pin, and must be on the side of the ADC input pin as possible. Don’t connect the capacitor’s ground pin to ground plain directly, 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 (P4.0). 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.

12 RAIL TO RAIL ANALOG COMPARAOTR

12.1 OVERVIEW

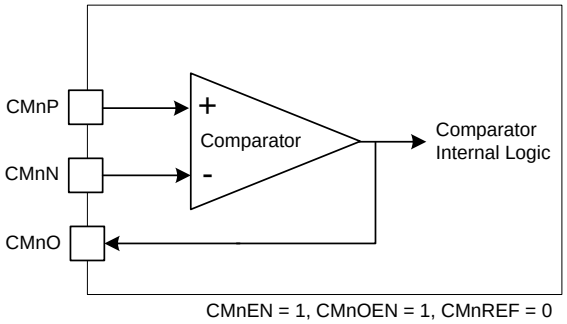


The micro-controller builds in 3 sets comparators. The comparators are Rail-to-Rail structure. That means the input/output voltage is real from Vdd~Vss. When the positive input voltage is greater than the negative input voltage, the comparator output is high. When the positive input voltage is smaller than the negative input voltage, the comparator output is low.

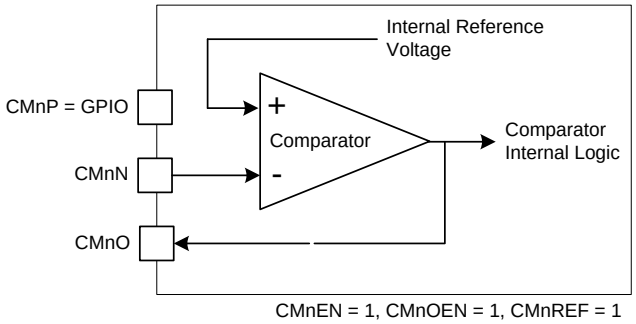
The CMnOUT and CMnIRQ bits indicate the comparator result. The CMnOUT shows the comparator result immediately, but the CMnIRQ only indicates the even of the comparator result. The even condition is controlled by register and includes rising edge (CMnOUT changes from low to high), falling edge (CMnOUT changes from high to low) and bi-direction (Any CMnOUT transition occurrence). The CMnIRQ = 1 condition makes the comparator interrupt service executed when CMnIEN (comparator interrupt control bit) set.
The comparator builds in internal reference to replace comparator external positive input source and controlled by CMnREF bits. The internal reference voltage is 1/2*Vdd. When CMnREF = 0, the comparator positive is from external voltage source and CMnP pin. When CMnREF = 1, the comparator positive is from internal 1/2*Vdd and CMnP pin is GPIO function.

✸ ***Note: CMnOUT is comparator raw output without latch. It varies depend on the comparator process result. But the CMnIRQ is latch comparator output result. It must be cleared by program.***

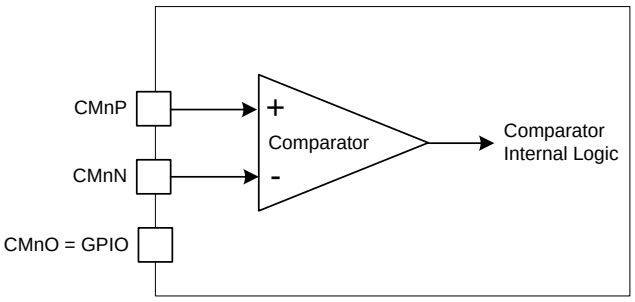
The comparator pins are shared with GPIO controlled by CMnEN bit. When CMnEN=1, CMnN pin is enabled connected to Comparator negative terminal. CMnOEN controls Comparator output connected to GPIO or not. When CMnOEN=1, Comparator output terminal is connected to GPIO pins and isolate GPIO function. CMnREF controls Comparator positive source from internal 1/2*Vdd voltage or external input. When CMnREF=1, Comparator positive voltage is from internal 1/2*Vdd voltage, and enable GPIO function of CMnP pin. Comparator pins selection table is as following.



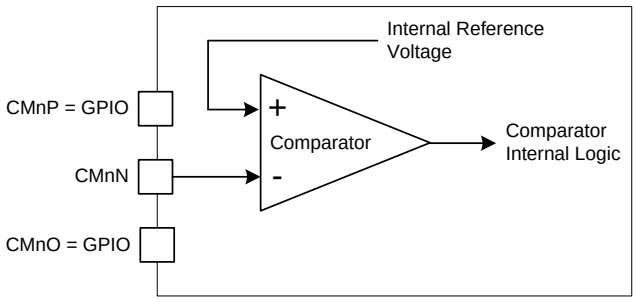
CMnEN = 1, CMnOEN = 1, CMnREF = 0



CMnEN = 1, CMnOEN = 1, CMnREF = 1



CMnEN = 1, CMnOEN = 0, CMnREF = 0



CMnEN = 1, CMnOEN = 0, CMnREF = 1

Because the pins of comparator and OP are the same, comparator control signal and OP amp control signal needs some condition to avoid comparator and OP amp enabled at the same time. CMnEN and OPnEN are the same (00 or 11), the pins are GPIO mode and disable comparator and OP. CMnEN=1 and OPnEN=0, enable comparator and disable OP amp. OPnEN=1 and CMnEN=0, enable OP amp and disable comparator.

Comparator No.	CMnEN	OPnEN	Comparator Negative Pin	Comparator Positive Pin		Comparator Output Pin	
				CMnREF=0	CMnREF=1	CMnOEN=0	CMnOEN=1
CMP0	CM0EN=0	OP0EN=0	All pins are GPIO mode. Comparator and OP amp are disabled.				
		OP0EN=1	OP Amp 0 enable.				
	CM0EN=1	OP0EN=0	CM0N	CM0P	P1.7 GPIO	P5.0 GPIO	CM0O
		OP0EN=1	All pins are GPIO mode. Comparator and OP amp are disabled.				
CMP1	CM1EN=0	OP1EN=0	All pins are GPIO mode. Comparator and OP amp are disabled.				
		OP1EN=1	OP Amp 1 enable.				
	CM1EN=1	OP1EN=0	CM1N	CM1P	P1.4 GPIO	P1.5 GPIO	CM1O
		OP1EN=1	All pins are GPIO mode. Comparator and OP amp are disabled.				
CMP2	CM2EN=0	OP2EN=0	All pins are GPIO mode. Comparator and OP amp are disabled.				
		OP2EN=1	OP Amp 2 enable.				
	CM2EN=1	OP2EN=0	CM2N	CM2P	P1.1 GPIO	P1.2 GPIO	CM2O
		OP2EN=1	All pins are GPIO mode. Comparator and OP amp are disabled.				

✸ ***Note: The comparator enable condition is fixed CMnEN=1 and OPnEN=0, or the comparator pins are GPIO mode and comparators are disabled.***

12.2 COMPARATOR MODE REGISTER

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMP0M	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CM0G1	CM0G0
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 7 **CM0EN:** Comparator 0 control bit.
0 = Disable. P1.6, P1.7, P5.0 are GPIO mode.
1 = Enable. P1.6 is CM0N pin.

Bit 6 **CM0IEN:** Comparator 0 interrupt function control bit.
0 = Disable.
1 = Enable.

Bit 5 **CM0IRQ:** Comparator 0 interrupt request bit.
0 = Non comparator interrupt request.
1 = Announce comparator interrupt request.

Bit 4 **CM0OEN:** Comparator 0 output pin control bit.
0 = Disable. CM0O is P5.0 GPIO mode.
1 = Enable. CM0O is comparator output pin and isolate P5.0 GPIO function.

Bit 3 **CM0REF:** Comparator 0 internal reference voltage source control bit.
0 = Disable. CM0P input pin is comparator positive input pin, and isolate P1.7 GPIO function.
1 = Enable. CM0P pin is P1.7 GPIO mode.

Bit 2 **CM0OUT:** Comparator 0 output flag bit.
0 = CM0P voltage or comparator internal reference voltage is less than CM0N voltage.
1 = CM0P voltage or comparator internal reference voltage is larger than CM0N voltage.

Bit [1:0] **CM0G[1:0]:** Comparator interrupt trigger direction control bit.
00 = Reserved.
01 = Rising edge trigger. CM0P > CM0N or comparator internal reference voltage.
10 = Falling edge trigger. CM0P < CM0N or comparator internal reference voltage.
11 = Both rising and falling edge trigger (Level change trigger).

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMP1M	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	CM1G1	CM1G0
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 7 **CM1EN:** Comparator 1 control bit.
0 = Disable. P1.3, P1.4, P1.5 are GPIO mode.
1 = Enable. P1.3 is CM1N pin.

Bit 6 **CM1IEN:** Comparator 1 interrupt function control bit.
0 = Disable.
1 = Enable.

Bit 5 **CM1IRQ:** Comparator 1 interrupt request bit.
0 = Non comparator interrupt request.
1 = Announce comparator interrupt request.

Bit 4 **CM1OEN:** Comparator 1 output pin control bit.
0 = Disable. CM1O is P1.5 GPIO mode.
1 = Enable. CM1O is comparator output pin and isolate P1.5 GPIO function.

Bit 3 **CM1REF:** Comparator 1 internal reference voltage source control bit.
0 = Disable. CM1P input pin is comparator positive input pin, and isolate P1.4 GPIO function.
1 = Enable. CM1P pin is P1.4 GPIO mode.

Bit 2 **CM1OUT:** Comparator 1 output flag bit.
0 = CM1P voltage or comparator internal reference voltage is less than CM1N voltage.
1 = CM1P voltage or comparator internal reference voltage is larger than CM1N voltage.

Bit [1:0] **CM1G[1:0]:** Comparator interrupt trigger direction control bit.
00 = Reserved.
01 = Rising edge trigger. CM1P > CM1N or comparator internal reference voltage.
10 = Falling edge trigger. CM1P < CM1N or comparator internal reference voltage.
11 = Both rising and falling edge trigger (Level change trigger).

09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMP2M	CM2EN	CM2IEN	CM2IRQ	CM2OEN	CM2REF	CM2OUT	CM2G1	CM2G0
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 7 **CM2EN:** Comparator 2 control bit.
0 = Disable. P1.0, P1.1, P1.2 are GPIO mode.
1 = Enable. P1.0 is CM2N pin.

Bit 6 **CM2IEN:** Comparator 2 interrupt function control bit.
0 = Disable.
1 = Enable.

Bit 5 **CM2IRQ:** Comparator 2 interrupt request bit.
0 = Non comparator interrupt request.
1 = Announce comparator interrupt request.

Bit 4 **CM2OEN:** Comparator 2 output pin control bit.
0 = Disable. CM2O is P1.2 GPIO mode.
1 = Enable. CM2O is comparator output pin and isolate P1.2 GPIO function.

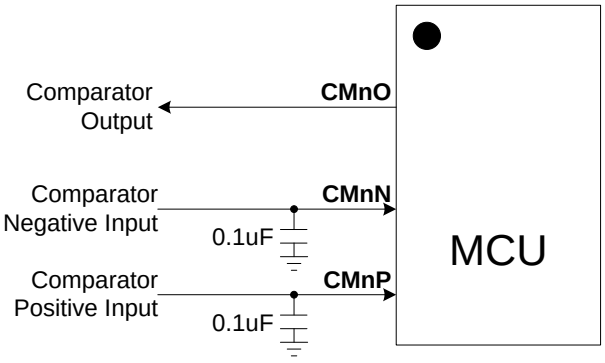
Bit 3 **CM2REF:** Comparator 2 internal reference voltage source control bit.
0 = Disable. CM2P input pin is comparator positive input pin, and isolate P1.1 GPIO function.
1 = Enable. CM2P pin is P1.1 GPIO mode.

Bit 2 **CM2OUT:** Comparator 2 output flag bit.
0 = CM2P voltage or comparator internal reference voltage is less than CM2N voltage.
1 = CM2P voltage or comparator internal reference voltage is larger than CM2N voltage.

Bit [1:0] **CM2G[1:0]:** Comparator interrupt trigger direction control bit.
00 = Reserved.
01 = Rising edge trigger. CM2P > CM2N or comparator internal reference voltage.
10 = Falling edge trigger. CM2P < CM2N or comparator internal reference voltage.
11 = Both rising and falling edge trigger (Level change trigger).

12.3 COMPARATOR APPLICATION NOTICE

The comparator is to compares the positive voltage and negative voltage to output result. The positive and negative sources are analog signal. In hardware application circuit, the comparator input pins must be connected a 0.1uF comparator to reduce power noise and make the input signal more stable. The application circuit is as following.

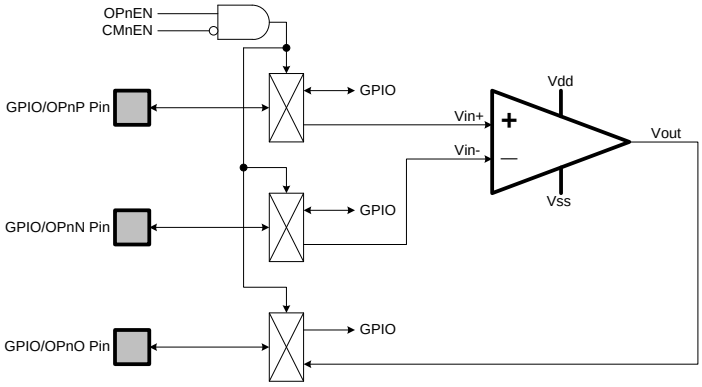


➢ **Example: Use comparator 0 to measure the external analog signal. When the analog signal is smaller than 1/2*Vdd, to execute interrupt service routine.**
In the case, use comparator interface 1/2*Vdd reference to be the comparator positive voltage source. The interrupt trigger condition is comparator output from low to high rising edge.

```
; The comparator 0 initialize.
        MOV     A, #01001001b   ; CM0REF=1, enable comparator internal 1/2*Vdd reference
        B0MOV   CM0M, A         ; voltage to be comparator positive source.
                                ; CM0G1, CM0G0=01, set comparator interrupt request
                                ; condition to be rising edge.
                                ; CM0IEN=1, enable comparator 0 interrupt function.
                                ; CM0IRQ=0, clear comparator 0 interrupt request flag.

        B0BSET  FCM0EN          ; Enable comparator 0.

Main:                           ; Main loop.
        …
        …
        JMP     MAIN
        …
        …

; Interrupt service routine. Jump from interrupt vector (ORG 8).
ISR:
        PUSH                    ; Save ACC and PFLAG.
        B0BTS1  FCM0IRQ         ; Check comparator 0 interrupt request flag.
        JMP     ISR_EXIT        ; No interrupt request, exit interrupt service routine.

        B0BCLR  FCM0IRQ         ; Clear comparator 0 interrupt request flag.
        …                       ; Execute comparator 0 interrupt service routine.
        …
        …
        JMP     ISR_EXIT        ; End of comparator 0 interrupt service routine.

ISR_EXIT:                       ; Exit interrupt service routine.
        POP                     ; Reload ACC and PFLAG.
        RETI                    ; Return to main loop.
```

13 RAIL to RAIL OP AMPLIFER

13.1 OVERVIEW



The micro-controller builds in 3 sets OP AMP. The comparators are Rail-to-Rail structure. That means the input/output voltage is real from Vdd~Vss.

The Rail-to-Rail OP AMP pins are shared with GPIO controlled by OPnEN bit. When OPnEN=1, GPIO pins switch to OP AMP and isolate GPIO path. OP pins selection table is as following.
Because the pins of comparator and OP are the same, comparator control signal and OP amp control signal needs some condition to avoid comparator and OP amp enabled at the same time. CMnEN and OPnEN are the same (00 or 11), the pins are GPIO mode and disable comparator and OP. CMnEN=1 and OPnEN=0, enable comparator and disable OP amp. OPnEN=1 and CMnEN=0, enable OP amp and disable comparator.

OP No.	OPnEN	CMnEN	OP Positive Pin	OP Negative Pin	OP Output Pin
OP0	OP0EN=0	CM0EN=0	All pins are GPIO mode. Comparator and OP amp are disabled.		
		CM0EN=1	Comparator 0 enable.		
	OP0EN=1	CM0EN=0	OP0P (Vin+)	OP0N (Vin-)	OP0O (Vout)
		CM0EN=1	All pins are GPIO mode. Comparator and OP amp are disabled.		
OP1	OP1EN=0	CM1EN=0	All pins are GPIO mode. Comparator and OP amp are disabled.		
		CM1EN=1	Comparator 1 enable.		
	OP1EN=1	CM1EN=0	OP1P (Vin+)	OP1N (Vin-)	OP1O (Vout)
		CM1EN=1	All pins are GPIO mode. Comparator and OP amp are disabled.		
OP2	OP2EN=0	CM2EN=0	All pins are GPIO mode. Comparator and OP amp are disabled.		
		CM2EN=1	Comparator 2 enable.		
	OP2EN=1	CM2EN=0	OP2P (Vin+)	OP2N (Vin-)	OP2O (Vout)
		CM2EN=1	All pins are GPIO mode. Comparator and OP amp are disabled.		

✸ ***Note: The OP AMP enable condition is fixed OPnEN=1 and CMnEN=0, or the OP AMP pins are GPIO mode and comparators are disabled.***

13.2 OP AMP REGISTER

09FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPM	-	-	-	-	-	OP2EN	OP1EN	OP0EN
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After Reset	-	-	-	-	-	0	0	0

Bit 2 **OP2EN:** OP Amp 2 control bit.
0 = Disable. P1.0~P1.2 are GPIO mode or comparator mode.
1 = Enable. P1.0~P1.2 are OP Amp pins.

Bit 1 **OP1EN:** OP Amp 1 control bit.
0 = Disable. P1.3~P1.5 are GPIO mode or comparator mode.
1 = Enable. P1.3~P1.5 are OP Amp pins.

Bit 0 **OP0EN:** OP Amp 0 control bit.
0 = Disable. P1.6, P1.7, P5.0 are GPIO mode or comparator mode.
1 = Enable. P1.6, P1.7, P5.0 are OP Amp pins.

14 INSTRUCTION TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I, "M" only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z…)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
ARITHMETIC	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1+N
	ADD	A,M	A ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
	ADD	M,A	M ← A + M, if occur carry, then C=1, else C=0	√	√	√	1+N
	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
	ADD	A,I	A ← A + I, if occur carry, then C=1, else C=0	√	√	√	1
	SBC	A,M	A ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
	SUB	A,M	A ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
	SUB	M,A	M ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1+N
	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
LOGIC	AND	A,M	A ← A and M	-	-	√	1
	AND	M,A	M ← A and M	-	-	√	1+N
	AND	A,I	A ← A and I	-	-	√	1
	OR	A,M	A ← A or M	-	-	√	1
	OR	M,A	M ← A or M	-	-	√	1+N
	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
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1+N
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1+N
	CLR	M	M ← 0	-	-	-	1
	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
BRANCH	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1+ S
	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
	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
	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
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
MISC	RET		PC ← Stack	-	-	-	2
	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
	PUSH		To push ACC and PFLAG (except NT0, NPD bit) into buffers.	-	-	-	1
	POP		To pop ACC and PFLAG (except NT0, NPD bit) from buffers.	√	√	√	1
	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)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2735P, SN8P2735F, SN8P2734K, SN8P2734S, SN8P2733K, SN8P2733S, SN8P2732P, SN8P2732S ..... 0°C ~ + 70°C
SN8P2735PD, SN8P2735FD, SN8P2734KD, SN8P2734SD, SN8P2733KD, SN8P2733SD, SN8P2732PD, SN8P2732SD .... –40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

15.2 ELECTRICAL CHARACTERISTIC

- **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	Normal mode, Vpp = Vdd, *25℃, Fpu = 1MHz*		1.8	-	5.5	V
		Normal mode, Vpp = Vdd, *-40℃~85℃*		2.4	-	5.5	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	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
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 = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	150	
I/O output source current	IoH	Vop = Vdd – 0.5V		8	-	-	mA
sink current	IoL	Vop = Vss + 0.5V		8	-	-	
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current (Disable ADC, OP-amp, Comparator)	Idd1	Run Mode (No loading)	Vdd= 3V, Fcpu = 8MHz	-	5	-	mA
			Vdd= 5V, Fcpu = 8MHz	-	7	-	mA
			Vdd= 3V, Fcpu = 4MHz	-	2	-	mA
			Vdd= 5V, Fcpu = 4MHz	-	4	-	mA
			Vdd= 3V, Fcpu = 1MHz	-	1.5	-	mA
			Vdd= 5V, Fcpu = 1MHz	-	3	-	mA
			Vdd= 3V, Fcpu = 32KHz	-	20	-	uA
			Vdd= 5V, Fcpu = 32KHz	-	45	-	uA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd= 3V, ILRC=16KHz	-	3.5	-	uA
			Vdd= 5V, ILRC=32KHz	-	10	-	uA
	Idd3	Sleep Mode	Vdd= 5V/3V	-	1	2	uA
	Idd4	Green Mode (No loading, Watchdog Disable)	Vdd= 5V, IHRC=8MHz	-	0.35	-	mA
			Vdd= 3V, IHRC=8MHz	-	0.55	-	mA
			Vdd= 3V, Ext. 32KHz X'tal	-	6	-	uA
			Vdd= 5V, Ext. 32KHz X'tal	-	18	-	uA
			Vdd= 3V, ILRC=16KHz	-	3	-	uA
			Vdd= 5V, ILRC=32KHz	-	5.5	-	uA
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	*25℃*, Vdd=2.2V~ 5.5V Fcpu=Fhosc/2~Fhosc/16	15.68	16	16.32	MHz
			-40℃~85℃,Vdd=2.4V~ 5.5V Fcpu=Fhosc/2~Fhosc/16	15.2	16	16.8	MHz
LVD Voltage	Vdet0	Low voltage reset level. *25℃*		1.9	2.0	2.1	V
		Low voltage reset level. *-40℃~85℃*		1.8	2.0	2.3	V
	Vdet1	Low voltage reset/indicator level. *25℃*		2.3	2.4	2.5	V
		Low voltage reset/indicator level. *-40℃~85℃*		2.2	2.4	2.7	V
	Vdet2	Low voltage reset/indicator level. *25℃*		3.5	3.6	3.7	V
		Low voltage reset/indicator level. *-40℃~85℃*		3.3	3.6	3.9	V

" *" These parameters are for design reference, not tested.

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
AIN0 ~ AIN7 input voltage	Vani	Vdd = 5.0V	0	-	Avrefh	V
ADC reference Voltage	Vref		2	-	-	V
*ADC enable time	Tast	Ready to start convert after set ADENB = "1"	100	-	-	us
*ADC current consumption	I_{ADC}	Vdd=5.0V	-	0.6	-	mA
		Vdd=3.0V	-	0.4	-	mA
ADC Clock Frequency	F_{ADCLK}	VDD=5.0V	-	-	8M	Hz
		VDD=3.0V	-	-	5M	Hz
ADC Conversion Cycle Time	F_{ADCYL}	VDD=2.4V~5.5V	64	-	-	$1/F_{ADCLK}$
ADC Sampling Rate (Set FADS=1 Frequency)	F_{ADSMP}	VDD=5.0V	-	-	125	K/sec
		VDD=3.0V	-	-	80	K/sec
Differential Nonlinearity	DNL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	±1	-	-	LSB
Integral Nonlinearity	INL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	±2	-	-	LSB
No Missing Code	NMC	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	10	11	12	Bits
ADC offset Voltage	$V_{ADCoffset}$	Non-trimmed	-10	0	+10	mV
		Trimmed	-2	0	+2	mV

" *" These parameters are for design reference, not tested.

OP AMP 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
*Supply Current	Iop	Iout=0, Vdd= 3V	-	130	-	uA
		Iout=0, Vdd= 5V	-	150	-	uA
*Quiescent Current	Iq	OP-amp disable.	-	-	1	uA
Common Mode Input Voltage Range	Vcmr	Vdd=5.0V	Vss-0.3	-	Vdd+0.3	V
Input Offset Voltage	Vos	Vcm=Vss	-3	-	+3	mV
Power Supply Rejection Ratio	PSRR	Vcm=Vss	50	-	70	dB
Common Mode Rejection Ratio	CMRR	Vcm=-0.3V~2.5V. Vdd=5V.	50	-	80	dB
Open-Loop Gain (Large Signal)	Aol	Vout=0.2V~Vdd-0.2V. Vcm=Vss.	90	110	-	dB
Maximum Output Voltage Swing	Vol, Voh	0.5V output overdrive.	Vss+15		Vdd-15	V
Output Short Current	Isc	Unit Gain Buffer. Vo = Vss~Vdd, Vdd=3V	-15	-	+15	mA
		Unit Gain Buffer. Vo = Vss~Vdd, Vdd=5V	-40	-	+40	mA
Output Slew Rate	Tosr	Vo = Vss to Vdd, Vdd to Vss.	3	-	5	us

COMPARATOR CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

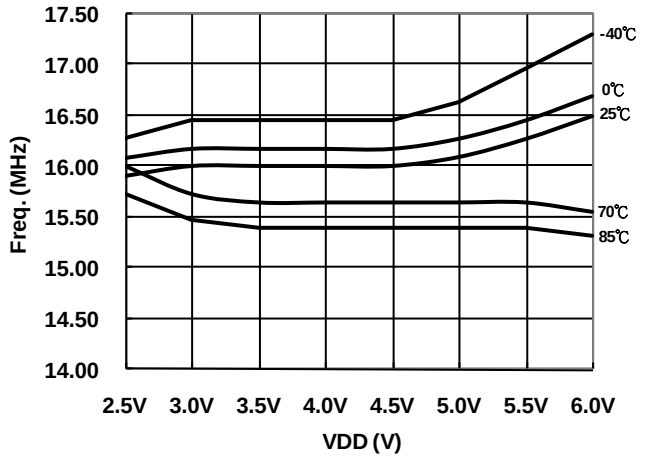
PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
*Supply Current	Icm	Iout=0, Vdd= 3V	-	130	-	uA
		Iout=0, Vdd= 5V	-	150	-	uA
*Quiescent Current	Iq	OP-amp disable.	-	-	1	uA
Input Offset Voltage	Vos	Vcm=Vss	-3		+3	mV
Common Mode Input Voltage Range	Vcmr	Vdd=5.0V	Vss-0.3		Vdd+0.3	V
Output Slew Rate	Tosr	Vo = Vss to Vdd, Vdd to Vss.	1	-	2	us

" *" These parameters are for design reference, not tested.

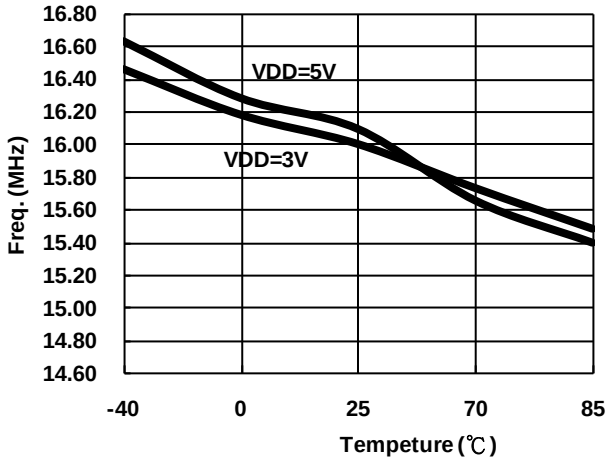
15.3 CHARACTERISTIC GRAPHS

The Graphs in this section are for design guidance, not tested or guaranteed. In some graphs, the data presented are outside specified operating range. This is for information only and devices are guaranteed to operate properly only within the specified range.

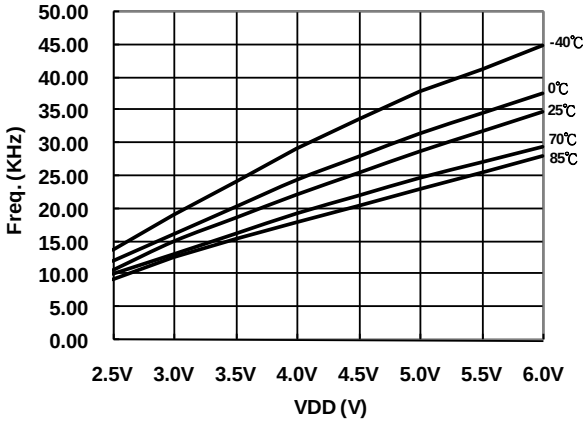
Internal High RC Oscillator (MHz)
(Fcpu = IHRC/2~IHRC/16)



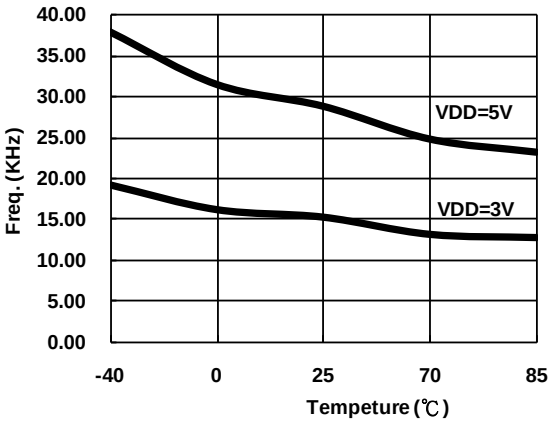
Internal High RC Oscillator (MHz)
(Fcpu=IHRC/2~IHRC/16)



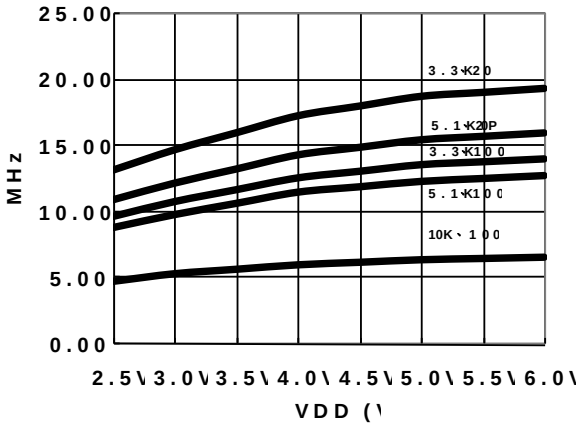
Internal Low RC Oscillator (KHz)



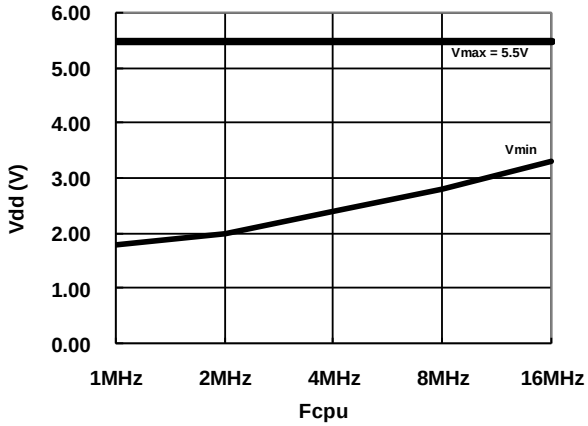
Internal Low RC Oscillator (KHz)



External RC Oscillator



System Minmum Operating Voltage



16 DEVELOPMENT TOOL

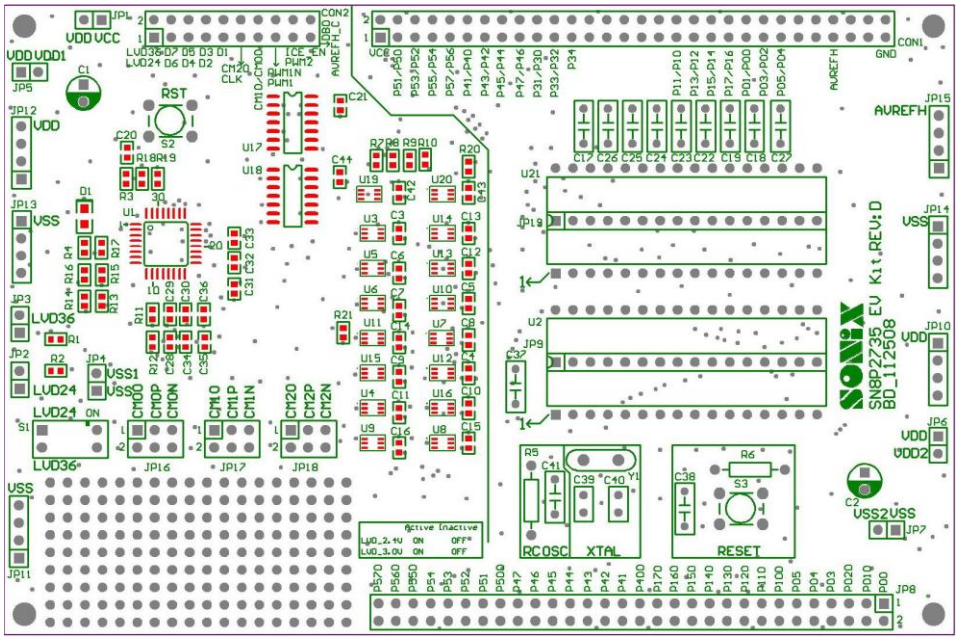
SONIX provides ICE (in circuit emulation), IDE (Integrated Development Environment) and EV-kit for SN8P2735 development. ICE and EV-kit are external hardware devices, and IDE is a friendly user interface for firmware development and emulation. These development tools' version is as following.

- **ICE: SN8ICE2K Plus II. (Please install 16MHz crystal in ICE to implement IHRC emulation.).**
- **ICE emulation speed maximum: 8 MIPS @ 5V (e.g. 16Mhz crystal, Fcpu = Fosc/2).**
- **EV-kit: SN8P2735_EV kit Rev. D.**
- **IDE: SONiX IDE M2IDE_V120.**
- **Writer: MPIII writer.**
- **Writer transition board: SN8P2735/SN8P2734/SN8P2733/SN8P2732**

16.1 SN8P2735 EV-KIT

SONIX provides SN8P2735 MCU which includes multi-PWM, ADC, Comparator and OP analog functions. These functions aren't built in SN8ICE2K Plus. To emulate the functions must be through SN8P2735 real chip. The real chip provides an EV-KIT to achieve Multi-PWM and the analog functions emulations. For SN8P2735 ICE emulation, the EV-Kit includes OP / Comparator / PWM / ADC / LVD2.4V / 3.6V and switch circuits.

SN8P2735 EV-kit PCB Outline:



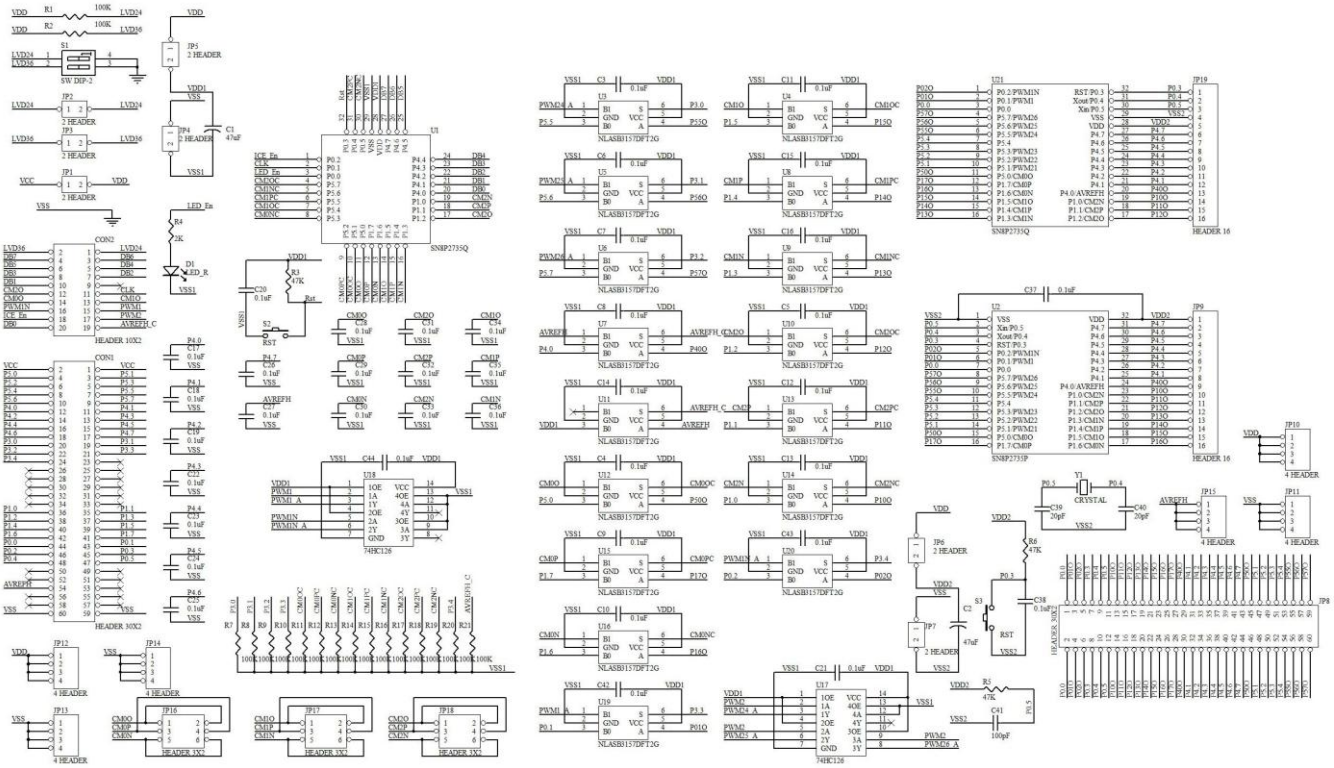
- CON1: Connect to SN8ICE 2K Plus CON1 (includes GPIO, EV-KIT control signal, and the others).
- CON2: Connect to SN8ICE 2K Plus JP3 (EV-KIT communication bus with ICE, control signal, and the others).
- S1: LVD24V / LVD36V control switch. To emulate LVD2.4V flag / reset function and LVD3.6V / flag function

Switch No.	ON	OFF
LVD24	LVD 2.4V Active	LVD 2.4V Inactive
LVD36	LVD 3.6V Active	LVD 3.6V Inactive

- JP8: GPIO connector.
- U1: SN8P2735 EV-chip for analog functions emulation.
- U2: SN8P2735 DIP form connector for connecting to user's target board.
- U21: SN8P2735 LQFP form connector for connecting to user's target board.
- JP15: Using ADC function before, SN8ICE 2K Plus AVREFH/VDD jumper pin must be removed. If ADC external reference voltage function enable, JP15 (AVREFH) or P40O is external reference voltage input.
- C17~C19,C22~C26: Connect 0.1uF capacitors to AIN0~AIN7 input which are ADC channel 0~7 bypass capacitors.

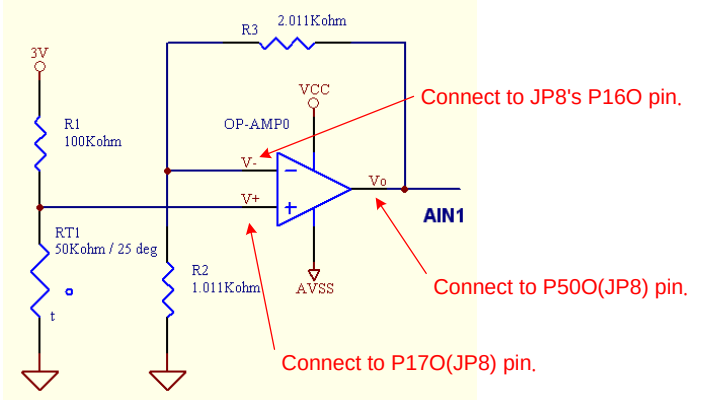
- C27: Connect 0.1uF capacitors to AVREFH input which are ADC reference voltage bypass capacitors.
- S2: Reset key. If EV-KIT active fail, press S2 to reset EV-KIT Real Chip (U1).
- JP16 ~ JP18: Observe CMP0~CMP2 / OP-Amp0~OP-Amp2 input/output voltage.
- C28~C36: Comparator or OP-Amp bypass capacitors

SN8P2735 EV-kit schematic:



16.2 ICE AND EV-KIT APPLICATION NOTIC

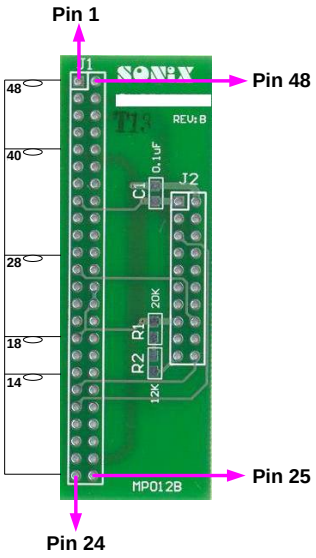
1. SN8ICE2K Plus power switch must be turned off before you connect the SN8P2735 EV-KIT to SN8ICE2K Plus.
2. Connect EV-KIT's CON1/CON2 to ICE's CON1/JP3.
3. SN8ICE2K Plus's AVREFH/VDD jumper pin must be removed.
4. Turn on SN8ICE2K Plus power switch after user had finished step 1~3.
5. **It is necessary to connect 16MHz crystal in ICE for IHRC_8M mode emulation.**
6. When ADC function enable. The ADM's bit 3 (FAVREFH) is set as High. P40 or JP15 (AVREFH) will be external reference voltage input pin.
7. When ADC function enable. The ADM's bit 3 (FAVREFH) is set as Low. P40 will be analog signal input pin. JP15 (AVREFH) do not connect power device.
8. Observe ADC internal or external reference voltage is JP15(AVREFH).
9. The OP-Amp0 of SN8P2735 application circuit is as the below figure (OP-AMP connect).



10. When SN8P2735's OP-AMP0 function enable as the above figure. P50O (JP8) is OP-AMP0's output. P17O (JP8) is OP-AMP0's non-inverse. P16O (JP8) is OP-AMP0's inverse.
11. If user want to measure OP-AMP0 V+ / V- / Vo voltage as the above figure, the real OP-AMP0's inverse voltage is CM0N (JP16). The real OP-AMP0's non-inverse voltage is CM0P (JP16). The real OP-AMP0's output voltage is CM0O (JP16).
12. If user use OP-AMP1 to replace OP-AMP0 as the above figure circuit, P15O (JP8) is OP-AMP1's output. P14O (JP8) is OP-AMP1's non-inverse. P13O (JP8) is OP-AMP1's inverse.
13. If user want to measure OP-AMP1 V+ / V- / Vo voltage as the above figure, the real OP-AMP1's inverse voltage is CM1N (JP17). The real OP-AMP1's non-inverse voltage is CM1P (JP17). The real OP-AMP1's output voltage is CM1O (JP17).
14. If user use OP-AMP2 to replace OP-AMP0 as the above figure circuit, P12O (JP8) is OP-AMP2's output. P11O (JP8) is OP-AMP2's non-inverse. P10O (JP8) is OP-AMP2's inverse.
15. If user want to measure OP-AMP2 V+ / V- / Vo voltage as the above figure, the real OP-AMP2's inverse voltage is CM2N (JP18). The real OP-AMP2's non-inverse voltage is CM2P (JP18). The real OP-AMP2's output voltage is CM2O (JP18).
16. Why OP-AMP connecting is different with measurement, because OP-AMP series connection with analog switch internal resistor (Ron).

17 OTP PROGRAMMING PIN

17.1 WRITER TRANSITION BOARD SOCKET PIN ASSIGNMENT



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	VSS
CLK/PGCLK	3	4	CE
PGM/OTPCLK	5	6	OE/ShiftDat
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 PROGRAMMING PIN MAPPING:

Programming Pin Information of SN8P2735 Series							
Chip Name		SN8P2735P(P-DIP)			SN8P2735F(LQFP)		
Writer Connector		IC and JP3 48-pin text tool Pin Assignment					
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number
1	VDD	32	VDD	40	28	VDD	36
2	GND	1	VSS	9	29	VSS	37
3	CLK	24	P4.0	32	20	P4.0	28
4	CE	-	-	-	-	-	-
5	PGM	28	P4.4	36	24	P4.4	32
6	OE	25	P4.1	33	21	P4.1	29
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	4	RST	12	32	RST	40
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	3	P0.4	11	31	P0.4	39

Programming Pin Information of SN8P2735 Series							
Chip Name		SN8P2734K/S(SKDIP/SOP)			SN8P2733K/S(SKDIP/SOP)		
Writer Connector		IC and JP3 48-pin text tool Pin Assignment					
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number
1	VDD	28	VDD	38	24	VDD	36
2	GND	1	VSS	11	1	VSS	13
3	CLK	20	P4.0	30	16	P4.0	28
4	CE	-	-	-	-	-	-
5	PGM	24	P4.4	34	20	P4.4	32
6	OE	21	P4.1	31	17	P4.1	29
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	4	RST	14	4	RST	16
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	3	P0.4	13	3	P0.4	15

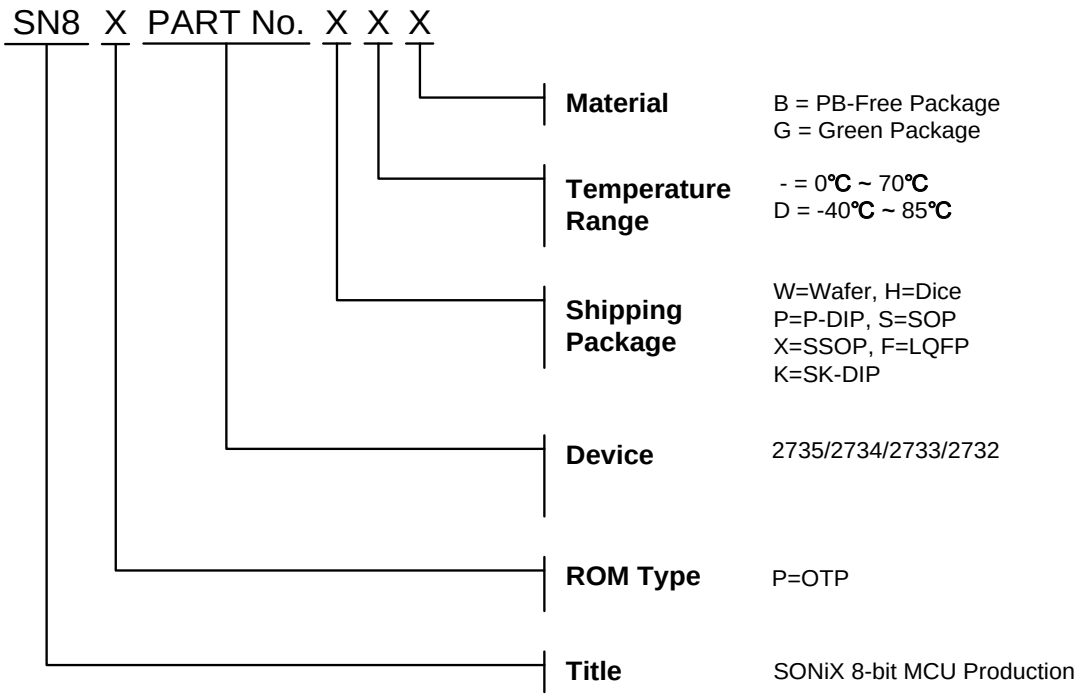
Programming Pin Information of SN8P2735 Series							
Chip Name		SN8P2732P/S(P-DIP/SOP)					
Writer Connector		IC and JP3 48-pin text tool Pin Assignment					
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number
1	VDD	20	VDD	34			
2	GND	1	VSS	15			
3	CLK	13	P4.0	27			
4	CE	-	-	-			
5	PGM	17	P4.4	31			
6	OE	14	P4.1	28			
7	D1	-	-	-			
8	D0	-	-	-			
9	D3	-	-	-			
10	D2	-	-	-			
11	D5	-	-	-			
12	D4	-	-	-			
13	D7	-	-	-			
14	D6	-	-	-			
15	VDD	-	-	-			
16	VPP	4	RST	18			
17	HLS	-	-	-			
18	RST	-	-	-			
19	-	-	-	-			
20	ALSB/PDB	3	P0.4	17			

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 OTP MCU.

18.2 MARKING INDETIFICATION SYSTEM



18.3 MARKING EXAMPLE

- **Wafer, Dice:**

Name	ROM Type	Device	Package	Temperature	Material
S8P2735W	OTP	2735	Wafer	0℃~70℃	-
SN8P2735H	OTP	2735	Dice	0℃~70℃	-

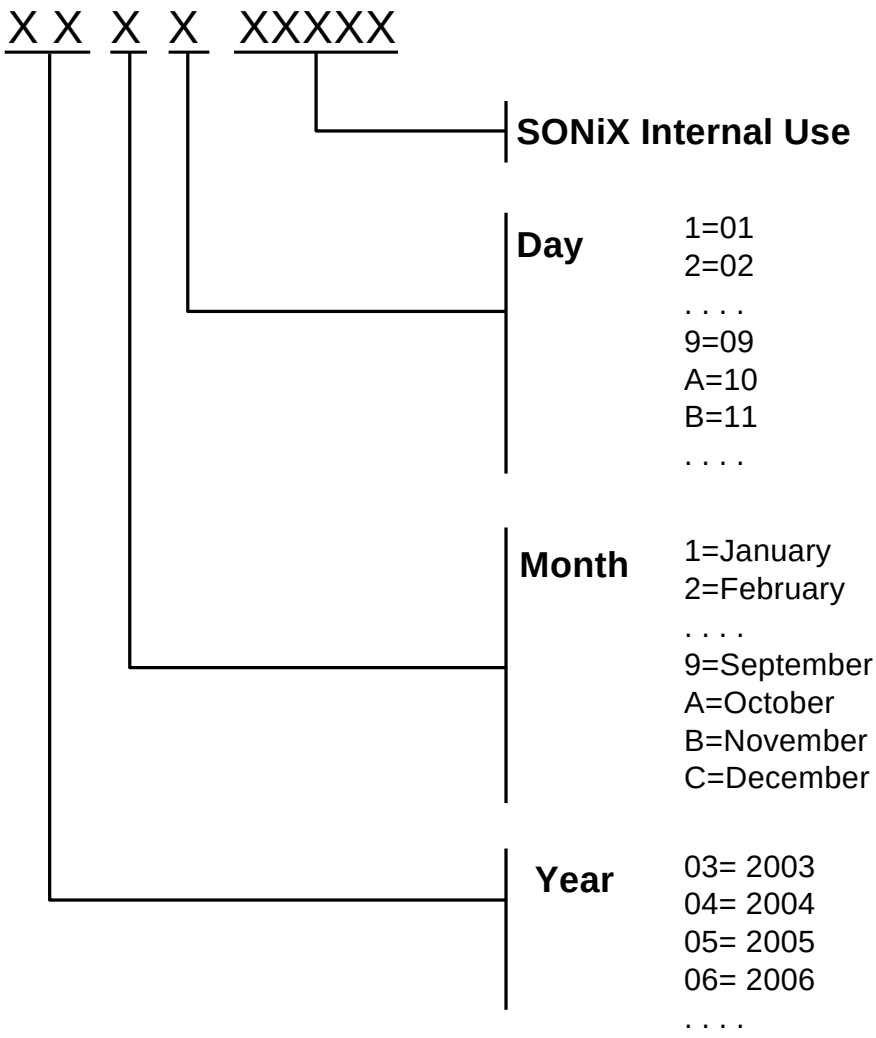
- **Green Package:**

Name	ROM Type	Device	Package	Temperature	Material
SN8P2735PG	OTP	2735	P-DIP	0℃~70℃	Green Package
SN8P2735FG	OTP	2735	LQFP	0℃~70℃	Green Package
SN8P2734KG	OTP	2735	SK-DIP	0℃~70℃	Green Package
SN8P2734SG	OTP	2735	SOP	0℃~70℃	Green Package
SN8P2733KG	OTP	2735	SK-DIP	0℃~70℃	Green Package
SN8P2733SG	OTP	2735	SOP	0℃~70℃	Green Package
SN8P2732PG	OTP	2735	P-DIP	0℃~70℃	Green Package
SN8P2732SG	OTP	2735	SOP	0℃~70℃	Green Package
SN8P2735PDG	OTP	2735	P-DIP	-40℃~85℃	Green Package
SN8P2735FDG	OTP	2735	LQFP	-40℃~85℃	Green Package
SN8P2734KDG	OTP	2735	SK-DIP	-40℃~85℃	Green Package
SN8P2734SDG	OTP	2735	SOP	-40℃~85℃	Green Package
SN8P2733KDG	OTP	2735	SK-DIP	-40℃~85℃	Green Package
SN8P2733SDG	OTP	2735	SOP	-40℃~85℃	Green Package
SN8P2732PDG	OTP	2735	P-DIP	-40℃~85℃	Green Package
SN8P2732SDG	OTP	2735	SOP	-40℃~85℃	Green Package

- **PB-Free Package:**

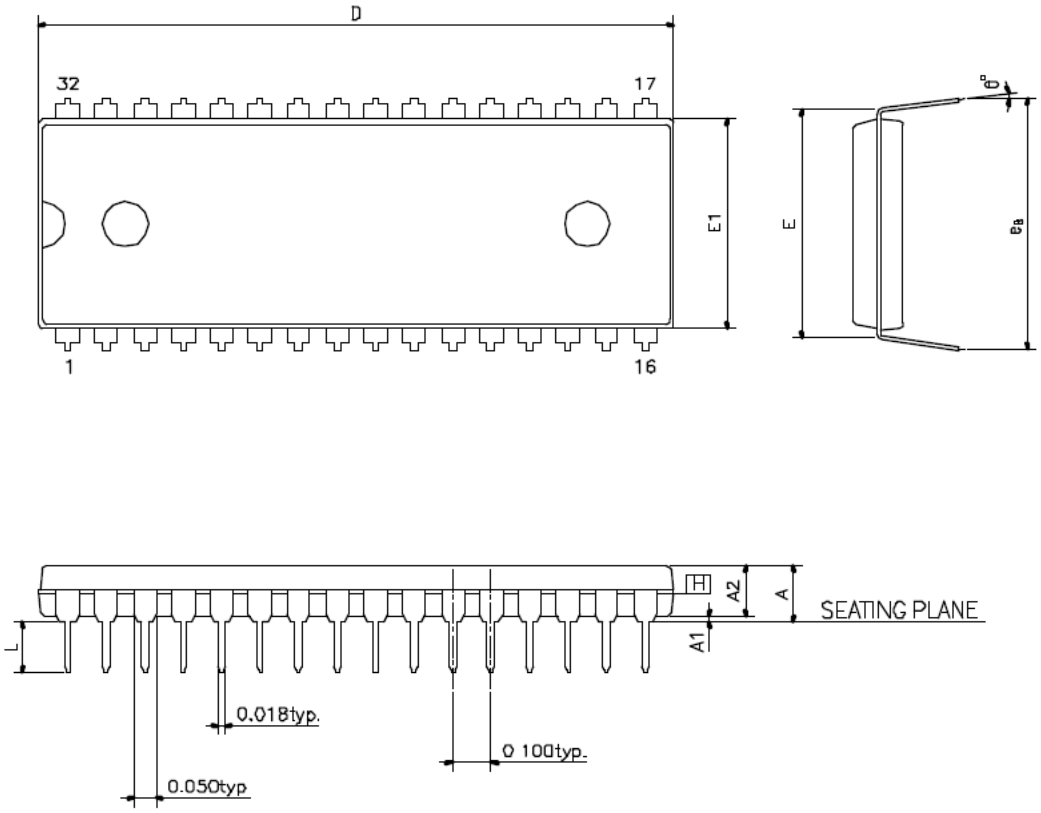
Name	ROM Type	Device	Package	Temperature	Material
SN8P2735PB	OTP	2735	P-DIP	0℃~70℃	PB-Free Package
SN8P2735FB	OTP	2735	LQFP	0℃~70℃	PB-Free Package
SN8P2734KB	OTP	2735	SK-DIP	0℃~70℃	PB-Free Package
SN8P2734SB	OTP	2735	SOP	0℃~70℃	PB-Free Package
SN8P2733KB	OTP	2735	SK-DIP	0℃~70℃	PB-Free Package
SN8P2733SB	OTP	2735	SOP	0℃~70℃	PB-Free Package
SN8P2732PB	OTP	2735	P-DIP	0℃~70℃	PB-Free Package
SN8P2732SB	OTP	2735	SOP	0℃~70℃	PB-Free Package
SN8P2735PDB	OTP	2735	P-DIP	-40℃~85℃	PB-Free Package
SN8P2735FDB	OTP	2735	LQFP	-40℃~85℃	PB-Free Package
SN8P2734KDB	OTP	2735	SK-DIP	-40℃~85℃	PB-Free Package
SN8P2734SDB	OTP	2735	SOP	-40℃~85℃	PB-Free Package
SN8P2733KDB	OTP	2735	SK-DIP	-40℃~85℃	PB-Free Package
SN8P2733SDB	OTP	2735	SOP	-40℃~85℃	PB-Free Package
SN8P2732PDB	OTP	2735	P-DIP	-40℃~85℃	PB-Free Package
SN8P2732SDB	OTP	2735	SOP	-40℃~85℃	PB-Free Package

18.4 DATECODE SYSTEM



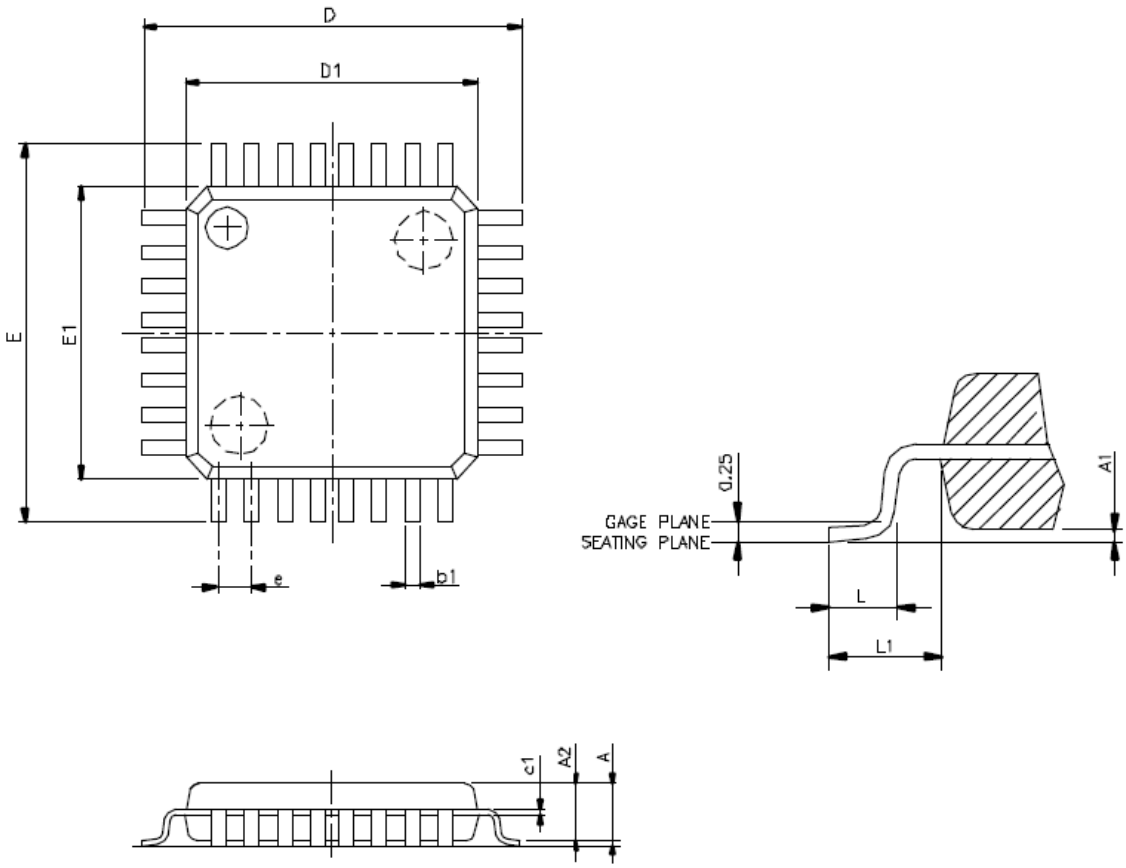
19 PACKAGE INFORMATION

19.1 P-DIP 32 PIN



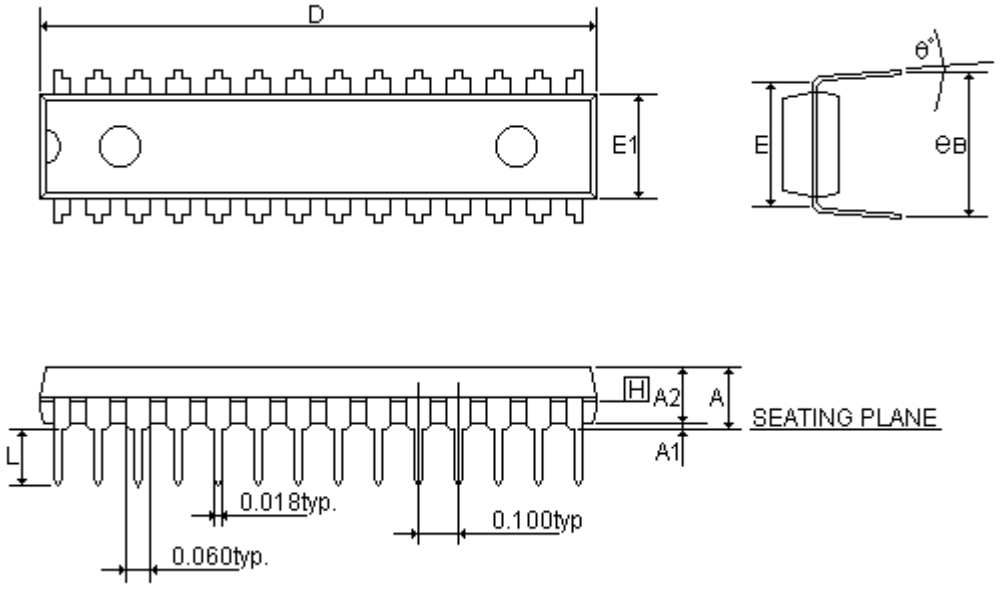
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.220	-	-	5.588
A1	0.015	-	-	0.381	-	-
A2	0.150	0.155	0.160	3.81	3.937	4.064
D	1.645	1.650	1.660	41.783	41.91	42.164
E	0.600 BSC			15.24 BSC		
E1	0.540	0.545	0.550	13.716	13.843	13.97
L	0.115	0.130	0.150	2.921	3.302	3.81
e_B	0.630	0.650	0.670	16.002	16.51	17.018
θ°	0°	7°	15°	0°	7°	15°

19.2 LQFP 32 PIN



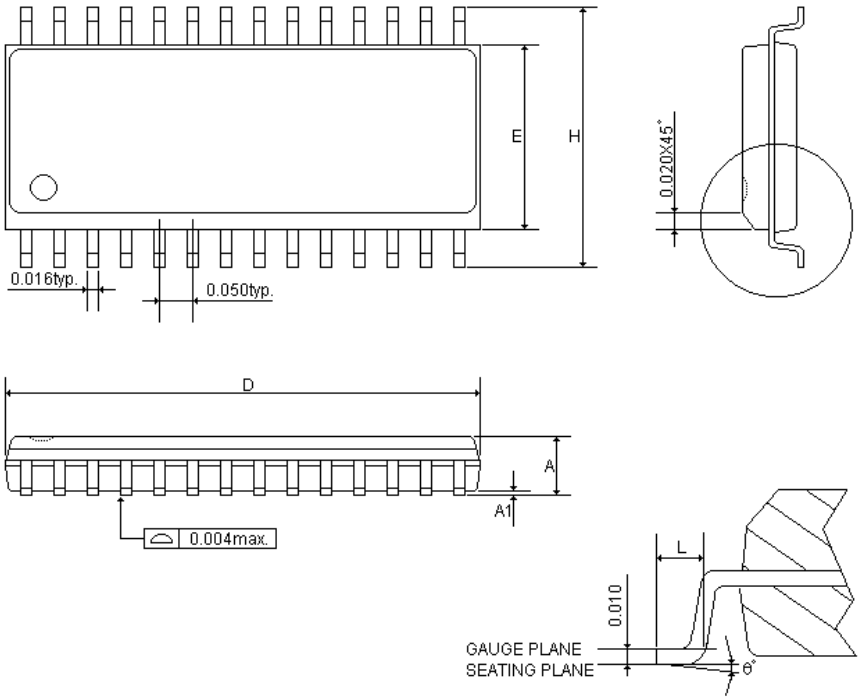
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.063	-	-	1.6
A1	0.002	0.004	0.006	0.05	0.1	0.15
A2	0.053	0.055	0.057	1.35	1.4	1.45
c1	0.004	0.005	0.006	0.09	0.125	0.16
D	0.354 BSC			9 BSC		
D1	0.276 BSC			7 BSC		
BSC E	0.354 BSC			9 BSC		
E1	0.276 BSC			7 BSC		
e	0.031 BSC			0.8 BSC		
b	0.012	0.015	0.018	0.3	0.375	0.45
L	0.018	0.024	0.030	0.45	0.6	0.75
L1	0.039 REF			1 REF		

19.3 SK-DIP 28 PIN



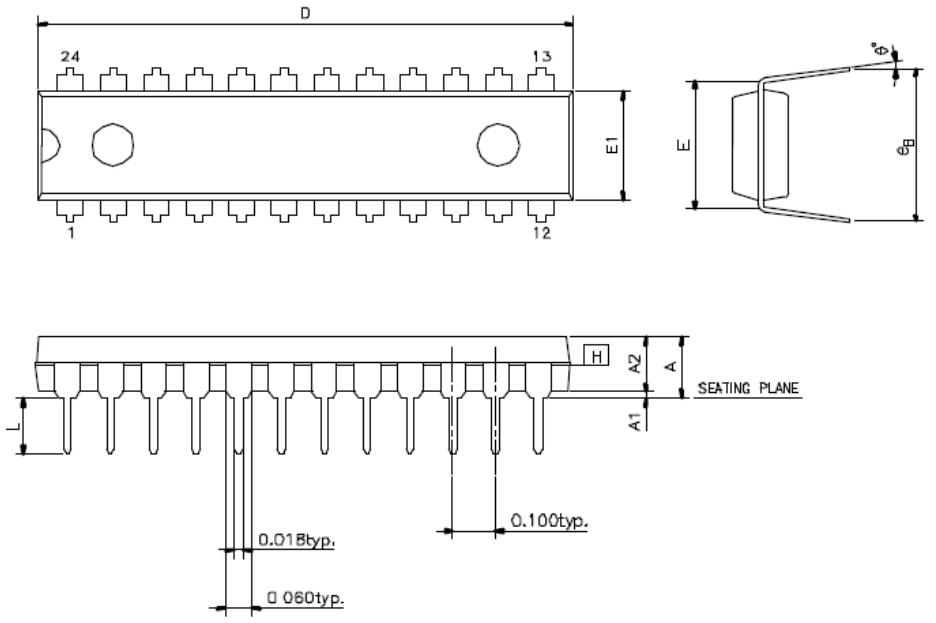
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.114	0.130	0.135	2.896	3.302	3.429
D	1.390	1.390	1.400	35.306	35.306	35.560
E		0.310			7.874	
E1	0.283	0.288	0.293	7.188	7.315	7.442
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.330	0.350	0.370	8.382	8.890	9.398
θ°	0°	7°	15°	0°	7°	15°

19.4 SOP 28 PIN



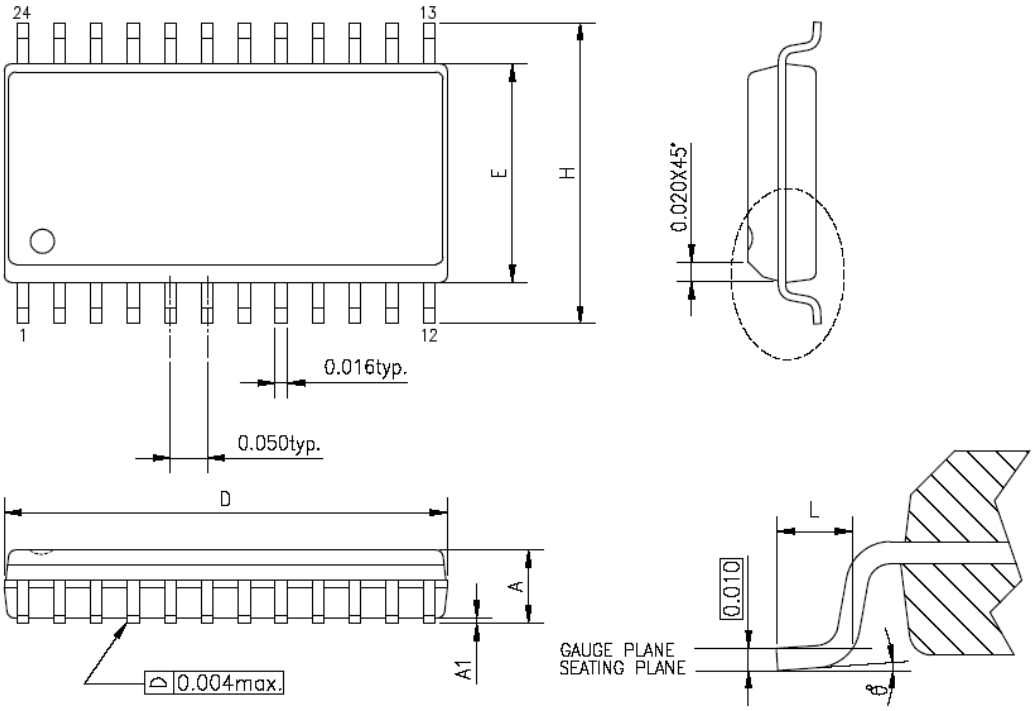
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.697	0.705	0.713	17.704	17.907	18.110
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
θ°	0°	4°	8°	0°	4°	8°

19.5 SK-DIP 24 PIN



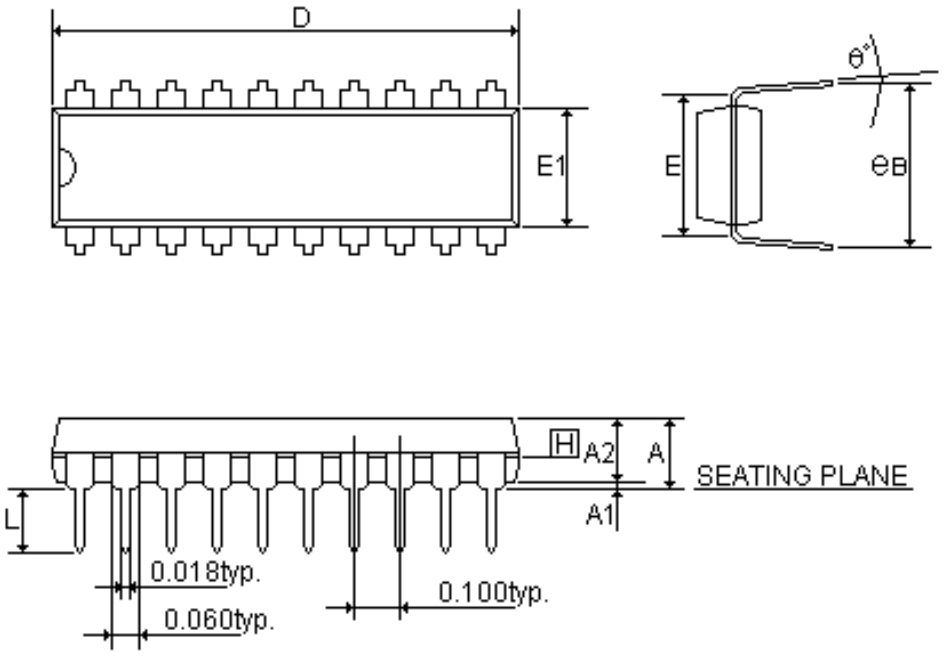
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-		0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.735	0.755	0.775	18.669	19.177	19.685
E	0.30 BSC			7.620 BSC		
E1	0.253	0.258	0.263	6.426	6.553	6.680
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

19.6 SOP 24 PIN



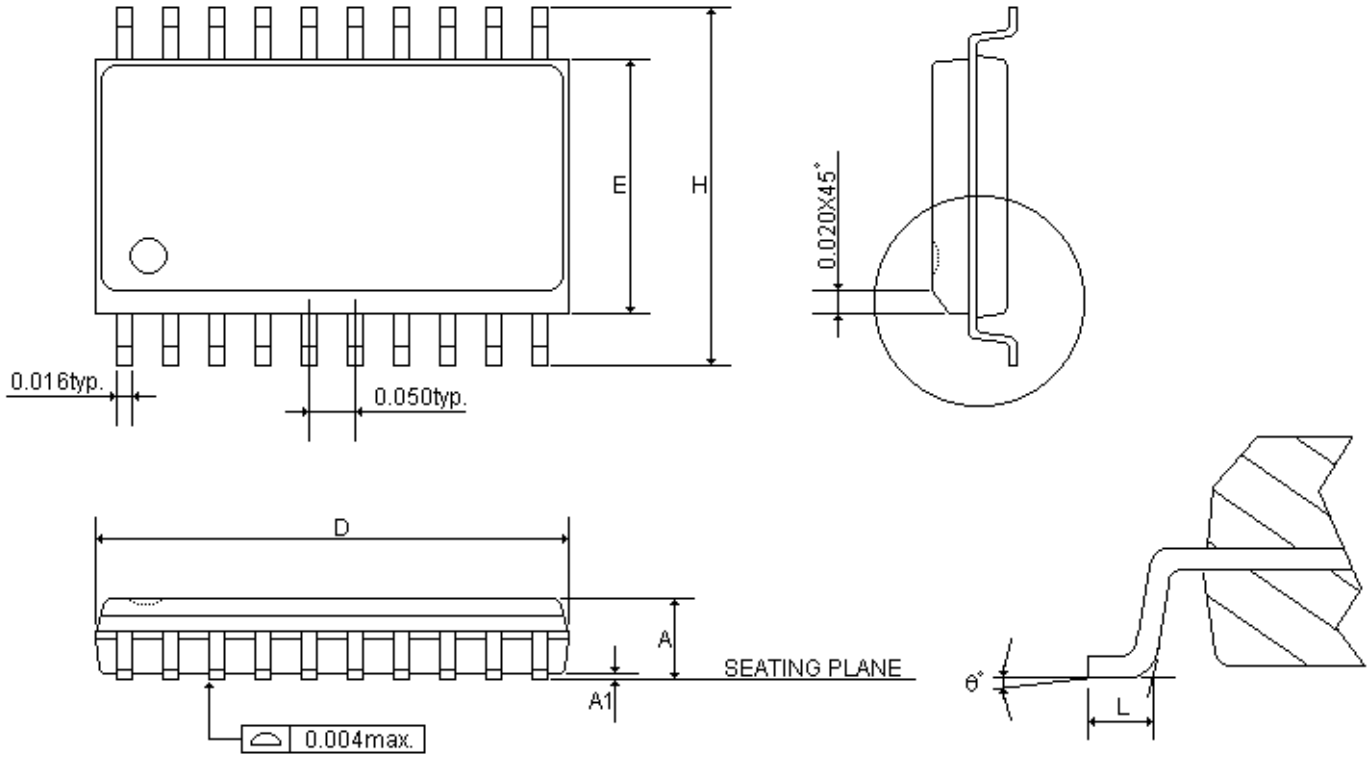
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.069	-	-	1.753
A1	0.004	-	0.010	0.102	-	0.254
D	0.612	0.618	0.624	15.545	15.697	15.850
E	0.292	0.296	0.299	7.417	7.518	7.595
H	0.405	0.412	0.419	10.287	10.465	10.643
L	0.021	0.031	0.041	0.533	0.787	1.041
θ°	0°	4°	8°	0°	4°	8°

19.7 P-DIP 20 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.980	1.030	1.060	24.892	26.162	26.924
E		0.300			7.620	
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

19.8 SOP 20 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	0.496	0.502	0.508	12.598	12.751	12.903
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°

Main Office:
Address: 10F-1, NO.36, Taiyuan Street, Chupei City, Hsinchu, Taiwan R.O.C.
Tel: 886-3-560 0888
Fax: 886-3-560 0889

Taipei Office:
Address: 15F-2, NO.171, Song Ted Road, Taipei, Taiwan R.O.C.
Tel: 886-2-2759 1980
Fax: 886-2-2759 8180

Hong Kong Office:
Unit 1519, Chevalier Commercial Centre, NO.8 Wang Hoi Road, Kowloon Bay, Hong Kong.
Tel: 852-2723-8086
Fax: 852-2723-9179

Technical Support by Email:
Sn8fae@sonix.com.tw