SN8P2761

USER'S MANUAL

Version 0.1

SN8P2761

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 0.1	Apr. 2015	First issue.

Table of Content

1 PRODUCT OVERVIEW

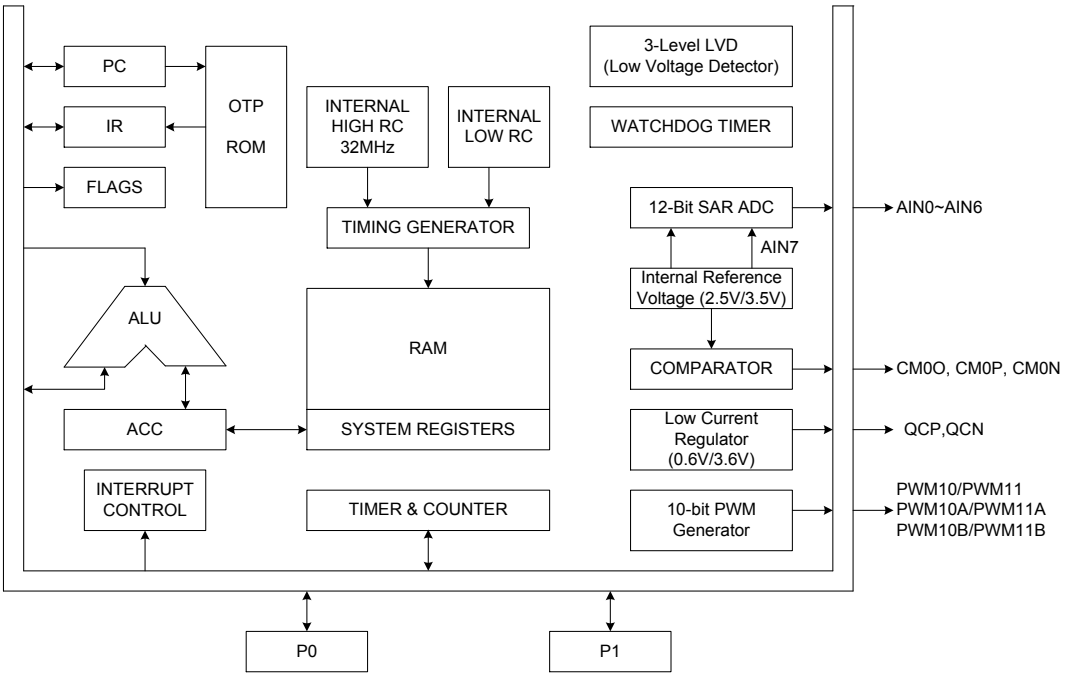
1.1 FEATURES

- **Memory configuration**
  ROM size: 1K * 16 bits.
  RAM size: 64 * 8 bits.
- **4 levels stack buffer.**
- **5 interrupt sources**
  4 internal interrupts: T0, PWM1, ADC, CM0
  1 external interrupt: INT0
- **I/O pin configuration**
  Bi-directional: P0, P1.
  Wakeup: P0, P1 level change.
  Pull-up resisters: P0, P1.
  Pull-down resisters: P0.0, P0.1, P0.5, P0.6
  External interrupt: P0.0.
  ADC input pin: AIN0~AIN6.
  Reset/Open-Drain: P0.4
- **Fcpu (Instruction cycle)**
  Fcpu = Fosc/2, Fosc/4, Fosc/8, Fosc/16, Fosc/32, Fosc/64, Fosc/128, Fosc/256.
- **On chip watchdog timer**
- **2.0V/2.4V/3.6V 3-level LVD.**
- **Powerful instructions**
  Instruction's length is one word.
  Most of instructions are one cycle only.
  All ROM area JMP instruction.
  All ROM area lookup table function (MOVC).
- **One 8-bit timer. (T0).**
  T0: Basic timer.
- **1-set 10-bit duty/cycle programmable PWM generator**
  3 pairs PWM output pins. Each pair has one normal PWM and one dead-band PWM.
- **7+1 channel 12-bit SAR ADC with 3-level Int. Ref.**
  Seven external ADC input channels.
  One internal input to measure internal AD reference voltage for power measurement.
  Internal AD reference voltage (VDD, 3.5V, 2.5V).
- **1-set comparator with internal reference.**
- **Two low current regulator outputs: 0.6V, 3.6V with trim.**
- **2 system clocks**
  Internal high clock: RC type 32MHz
  Internal low clock: RC type 16KHz(3V), 32KHz(5V)
- **4 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 T0 timer
- **Package (Chip form support)**
  PDIP 16 pins
  SOP 16 pins
  SSOP 16 pins

☞ **Features Selection Table**

CHIP	ROM	RAM	Stack	Timer	I/O	10-bit PWM	ADC	Int. Ref.	Reg. Out	Interrupt		Wake-up Pin No.	Package
				T0						Int	Ext		
SN8P2761	1K*16	64*8	4	V	14	6-ch (3 pairs)	12-ch	2.5V/ 3.5V	0.6V/ 3.6V	4	1	14	DIP16/ SOP16/ SSOP16

1.2 SYSTEM BLOCK DIAGRAM



1.3 PIN ASSIGNMENT

SN8P2761P (PDIP 16 pins)
SN8P2761S (SOP 16 pins)

VSS	1	U	16	VDD
P0.2/PWM10A/QCP	2		15	P1.0/AIN0/AVREFH
P0.3/PWM11A/QCN	3		14	P1.1/AIN1
P0.4/RST/VPP	4		13	P1.2/AIN2
P0.5/PWM10	5		12	P1.3/AIN3
P0.6/PWM11	6		11	P1.4/AIN4/CM0O
P0.1	7		10	P1.5/PWM10B/AIN5/CM0P
P0.0/INT0	8		9	P1.6/PWM11B/AIN6/CM0N

SN8P2761X (SSOP 16 pins)

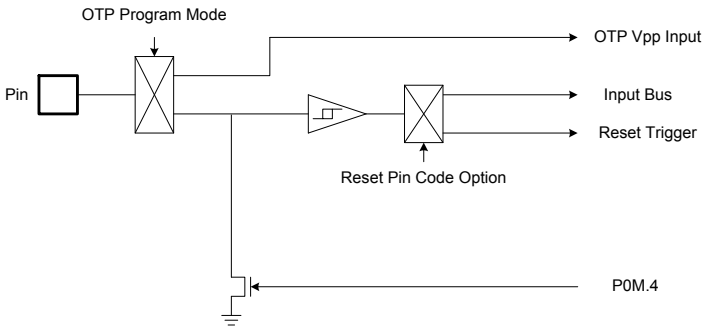
P0.5/PWM10	1	U	16	P0.4/RST/VPP
P0.6/PWM11	2		15	P0.3/PWM11A/QCN
P0.1	3		14	P0.2/PWM10A/QCP
P0.0/INT0	4		13	VSS
P1.6/PWM11B/AIN6/CM0N	5		12	VDD
P1.5/PWM10B/AIN5/CM0P	6		11	P1.0/AIN0/AVREFH
P1.4/AIN4/CM0O	7		10	P1.1/AIN1
P1.3/AIN3	8		9	P1.2/AIN2

1.4 PIN DESCRIPTIONS

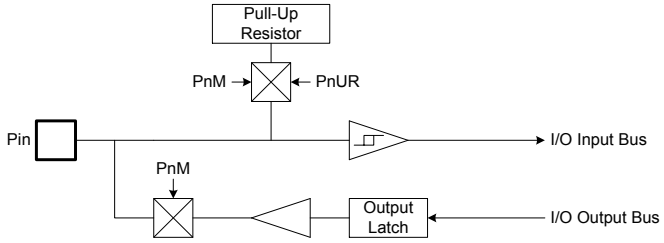
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital and analog circuit.
P0.4/RST/VPP	I/O, P	RST: System external reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”.
		VPP: OTP 12.3V power input pin in programming mode.
		P0.4: Bi-direction pin. Schmitt trigger structure as input mode and open-drain structure as output mode. No built-in pull-up/pull-down resisters. Level change wake-up.
P0.0/INT0	I/O	P0.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up/pull-down resisters. Level change wake-up.
		INT0: External interrupt 0 input pin.
P0.1	I/O	P0.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up/pull-down resisters. Level change wake-up.
P0.2/PWM10A/QCP	I/O	P0.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resister. Level change wake-up.
		PWM10A: PWM A pair output pin.
		QCP: Low current regulator output pin.
P0.3/PWM11A/QCN	I/O	P0.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resister. Level change wake-up.
		PWM11A: PWM A pair dead-band output pin.
		QCN: Low current regulator output pin.
P0.5/PWM10	I/O	P0.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up/pull-down resisters. Level change wake-up.
		PWM10: PWM output pin.
P0.6/PWM11	I/O	P0.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up/pull-down resisters. Level change wake-up.
		PWM11: PWM dead-band output pin.
P1.0/AIN0/AVREFH	I/O	P1.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resister.
		AIN0: ADC analog input pin.
		AVREFH: ADC reference high voltage input pin.
P1.[3:1]/AIN [3:1]	I/O	P1.[3:1]: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		AIN [3:1]: ADC analog input pin.
P1.4/AIN4/CM0O	I/O	P1.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resister.
		AIN4: ADC analog input pin.
		CM0O: The output pin of comparator 0.
P1.5/PWM10B/ AIN5/CM0P	I/O	P1.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resister.
		AIN5: ADC analog input pin.
		PWM10B: PWM B pair output pin.
		CM0P: The positive input pin of comparator 0.
P1.6/PWM11B/ AIN6/CM0N	I/O	P1.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resister.
		PWM11B: PWM B pair dead-band output pin.
		AIN6: ADC analog input pin.
		CM0N: The negative input pin of comparator 0.

1.5 PIN CIRCUIT DIAGRAMS

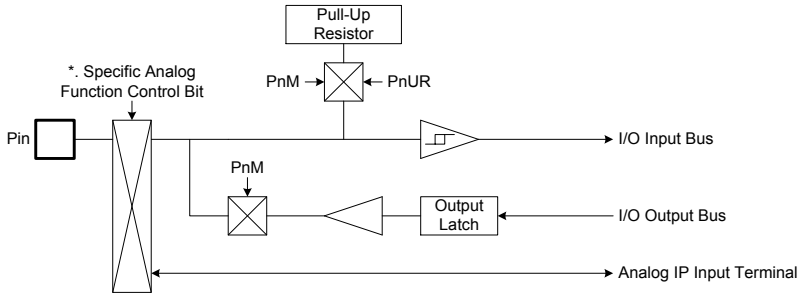
- **Reset shared pin structure:**



- **Normal bi-direction pin structure:**

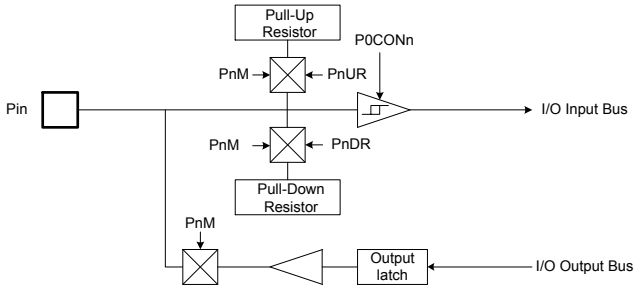


- **Bi-direction shared pin structure with specific analog input function (e.g. CM0P, CM0N):**

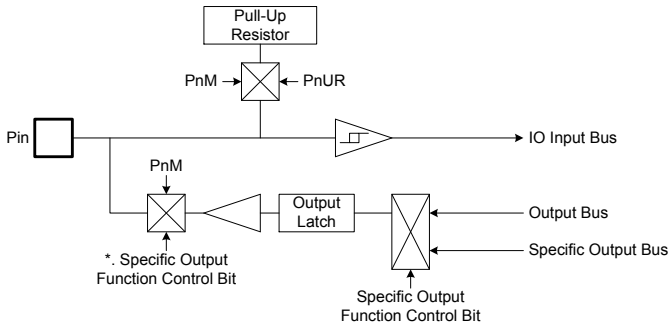


***. Some specific functions switch I/O direction directly, not through PnM register.**

- **Bi-direction shared pin structure with pull-up/pull-down resistor:**



- **Bi-direction shared pin structure with specific digital output function (e.g. PWM, …..):**

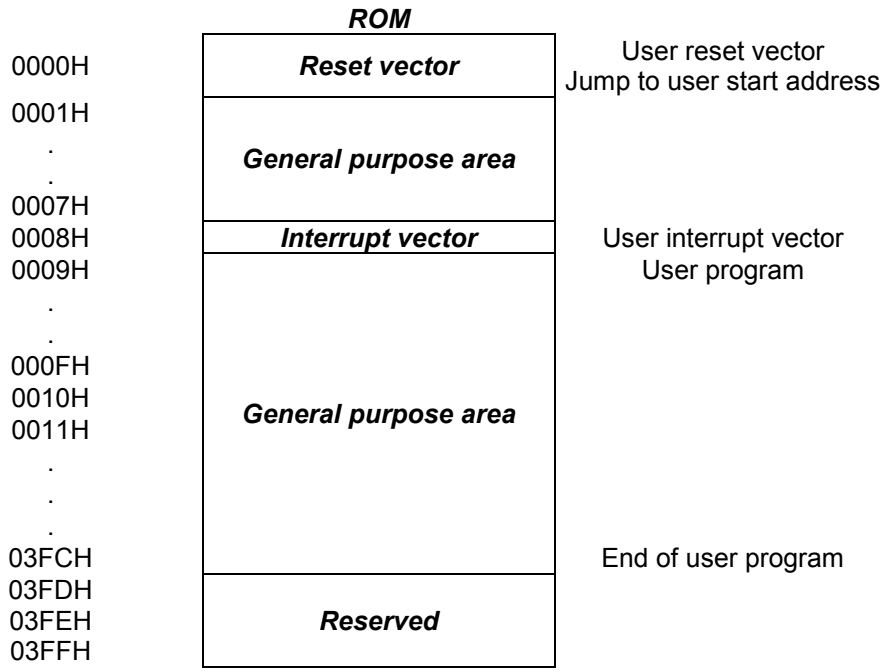


***. Some specific functions switch I/O direction directly, not through PnM register.**

2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (ROM)

☞ ***1K 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

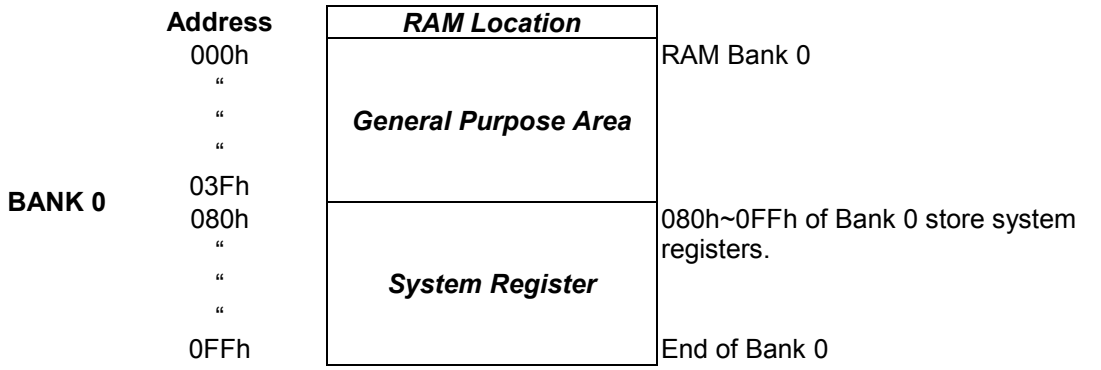
The last ROM addresses are reserved area. User should avoid these addresses (last address) when calculate the Checksum value.

➢ **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)

☞ ***64 X 8-bit RAM***



The 64-byte general purpose RAM is separated into Bank 0. Sonix provides “Bank 0” type instructions (e.g. b0mov, b0add, b0bts1, b0bset…) to control Bank 0 RAM directly.

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	-	-	R	Z	Y	-	PFLAG	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	CM0M	-	-	-
A	PW1M	PWCH	PW1YL	PW1YH	PW1BL	PW1BH	PW1DL	PW1DH	PW1A	-	-	-	-	-	-	P1CON
B	VREFH	ADM	ADB	ADR	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	-	-	-	-	-	-	INTRQ	INTEN	OSCM	-	WDTR	-	PCL	PCH
D	P0	P1	-	-	-	-	P0DR	-	T0M	T0C	-	-	-	-	-	STKP
E	P0UR	P1UR	-	-	-	-	-	@YZ	-	-	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 SYSTEM REGISTER DESCRIPTION

R = Working register and ROM look-up data buffer.
PFLAG = Special flag register.
PW1M = PWM mode register.
PW1YL = PWM cycle control register (Low byte).
PW1BL = PWM dead-band control register (Low byte).
PW1DL = PWM duty control register (Low byte).
PW1A = PWM dead-band control register.
VREFH = ADC reference voltage control register.
ADB = ADC data buffer.
Pn = Port n data buffer.
P1W = P1 wake-up control register.
INTRQ = Interrupt request register.
OSCM = Oscillator mode register.
PCH, PCL = Program counter.
T0M = T0 mode register.
STKP = Stack pointer buffer.
@YZ = RAM YZ indirect addressing index pointer.

Y, Z = Working, @YZ and ROM addressing register.
CM0M = Comparator 0 mode register.
PWCH = PWM output channel register.
PW1YH = PWM cycle control register (High byte).
PW1BH = PWM dead band control register (High byte).
PW1DH = PWM duty control register (High byte).
P1CON = P1 configuration register.
ADM = ADC mode register.
ADR = ADC resolution select register.
PEDGE = P0.0 edge direction register.
PnM = Port n input/output mode register.
INTEN = Interrupt enable register.
WDTR = Watchdog timer clear register.
P0DR = Port0 pull-down resister/Schmitt-trigger control register.
T0C = T0 counting register.
PnUR = Port n pull-up resister control register.
STK0~STK3 = Stack 0 ~ stack 3 buffer.

2.2.1.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
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
09CH	CM0EN	CM0T	CM0S1	CM0S0	CM0OEN	CM0OUT	CM0G1	CM0G0	R/W	CM0M
0A0H	PW1EN	PW1rate2	PW1rate1	PW1rate0	PWNV11	PWNV10		PW1PO	R/W	PW1M
0A1H	-	-	PWCH11B	PWCH10B	PWCH11A	PWCH10A	PWCH11	PWCH10	R/W	PWCH
0A2H	PW1Y7	PW1Y6	PW1Y5	PW1Y4	PW1Y3	PW1Y2	PW1Y1	PW1Y0	R/W	PW1YL
0A3H	-	-	-	-	-	-	PW1Y9	PW1Y8	R/W	PW1YH
0A4H	PW1B7	PW1B6	PW1B5	PW1B4	PW1B3	PW1B2	PW1B1	PW1B0	R/W	PW1BL
0A5H	-	-	-	-	-	-	PW1B9	PW1B8	R/W	PW1BH
0A6H	PW1D7	PW1D6	PW1D5	PW1D4	PW1D3	PW1D2	PW1D1	PW1D0	R/W	PW1DL
0A7H	-	-	-	-	-	-	PW1D9	PW1D8	R/W	PW1DH
0A8H	PW1A7	PW1A6	PW1A5	PW1A4	PW1A3	PW1A2	PW1A1	PW1A0	R/W	PW1A
0AFH	-	P1CON6	P1CON5	P1CON4	P1CON3	P1CON2	P1CON1	P1CON0	R/W	P1CON
0B0H	EVHENB	-	QCEN	QCM1	QCM0	VHS2	VHS1	VHS0	R/W	VREFH
0B1H	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0	R/W	ADM
0B2H	ADB11	ADB	ADB	ADB	ADB	ADB	ADB	ADB	R/W	ADB
0B3H	-	-	-	ADCKS0	ADB	ADB	ADB	ADB	R/W	ADR
0B8H	-	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0BFH	-	-	-	P00G1	P00G0	-	-	-	R/W	PEDGE
0C0H	-	P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H	-	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C8H	ADCIRQ	CM0IRQ	PW1IRQ	T0IRQ	-	-	-	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	CM0IEN	PW1IEN	T0IEN	-	-	-	P00IEN	R/W	INTEN
0CAH	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	-	-	-	PC9	PC8	R/W	PCH
0D0H	-	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	-	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D6H	P0CON6	P0CON5	P0CON1	P0CON0	P06DR	P05DR	P01DR	P00DR	R/W	P0DR
0D8H	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	-	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DFH	GIE	-	-	-	-	-	STKPB1	STKPB0	R/W	STKP
0E0H	-	P06R	P05R	-	P03R	P02R	P01R	P00R	W	P0UR
0E1H	-	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	-	-	-	-	-	-	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	-	-	-	-	-	-	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	-	-	-	-	-	-	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	-	-	-	-	-	-	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 10-bit binary counter separated into the high-byte 2 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 9.

	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	-	-	-	-	-	-	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	-	-	-	-	-	-	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 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
        …
```

2.2.6 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 (Y/Z).

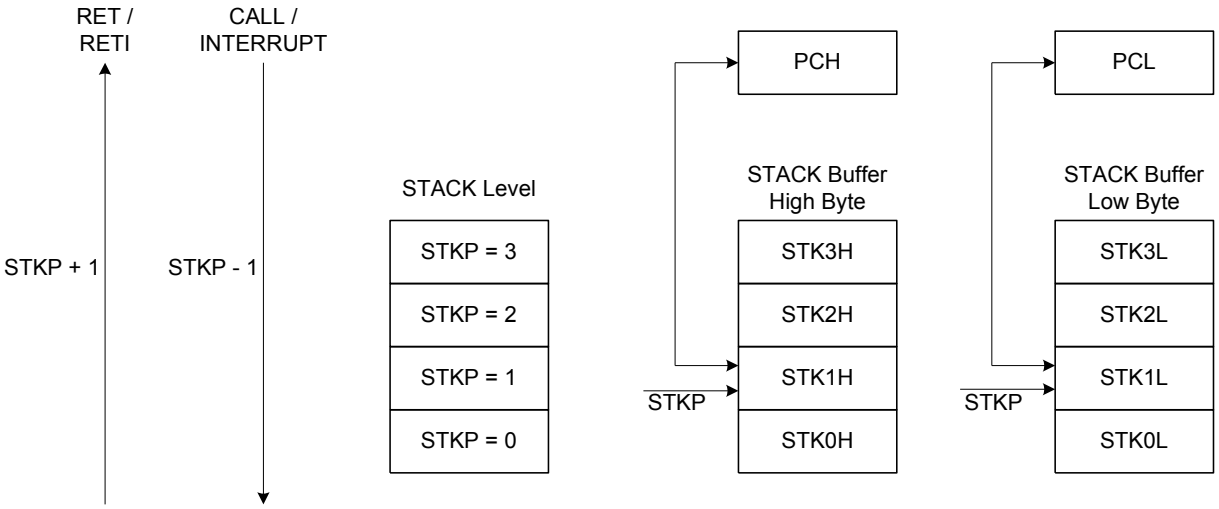
- **Example: Indirectly addressing mode with @YZ register.**

```
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 4-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 2-bit register to store the address used to access the stack buffer, 10-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	-	-	-	-	-	STKPB1	STKPB0
Read/Write	R/W	-	-	-	-	-	R/W	R/W
After reset	0	-	-	-	-	-	1	1

Bit[1:0] **STKPBn:** Stack pointer (n = 0 ~ 1)

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, #00000011B
B0MOV   STKP, A
```

0F0H~0F8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	-	-	-	-	SnPC9	SnPC8
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

0F0H~0F8H	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 = 3 ~ 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***
	STKPB1	**STKPB0**	**High Byte**	**Low Byte**	
0	1	1	Free	Free	-
1	1	0	STK0H	STK0L	-
2	0	1	STK1H	STK1L	-
3	0	0	STK2H	STK2L	-
4	1	1	STK3H	STK3L	-
> 4	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***
	STKPB1	**STKPB0**	**High Byte**	**Low Byte**	
4	1	1	STK3H	STK3L	-
3	0	0	STK2H	STK2L	-
2	0	1	STK1H	STK1L	-
1	1	0	STK0H	STK0L	-
0	1	1	Free	Free	-

2.5 CODE OPTION TABLE

The code option is the system hardware configurations including system clock option (Fcpu), watchdog timer operation, reset pin option and OTP ROM security control. The code option items are as following table:

Code Option	Content	Function Description
Fcpu	Fhosc/2	Instruction cycle is 2 oscillator clocks.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
	Fhosc/16	Instruction cycle is 16 oscillator clocks.
	Fhosc/32	Instruction cycle is 32 oscillator clocks.
	Fhosc/64	Instruction cycle is 64 oscillator clocks.
	Fhosc/128	Instruction cycle is 128 oscillator clocks.
	Fhosc/256	Instruction cycle is 256 oscillator clocks.
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.
	P0.4	Enable P0.4 bi-direction without pull-up resister (Open-drain structure as output mode).
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). In high noisy environment, reduce system clock speed (Fcpu), enable watchdog timer and select a good LVD level can make whole system work well and avoid error event occurrence.

2.5.2 Reset_Pin code option

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

- **Reset:** The reset pin is external reset function. When Reset pin falling edge trigger is occurring, the system will be reset.
- **P0.4:** Set reset pin to general input only pin (P0.4). The external reset function is disabled and the pin is bi-direction pin. Open-drain structure as output mode.

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.

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)

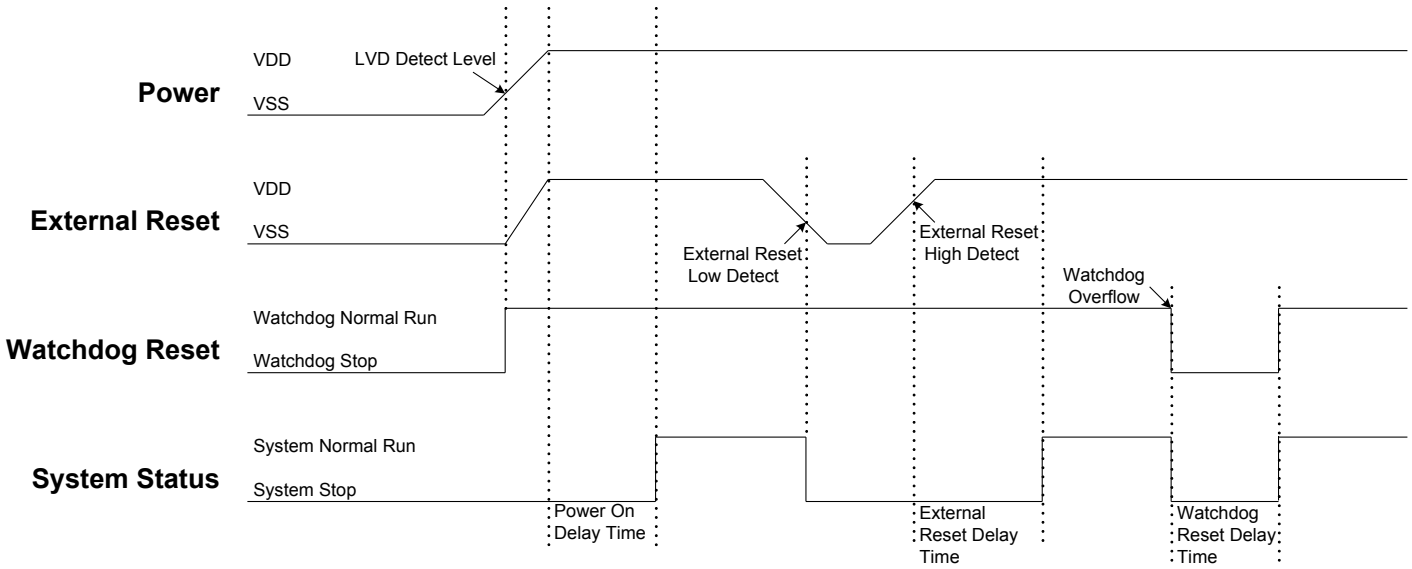
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status 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 of 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 on 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.

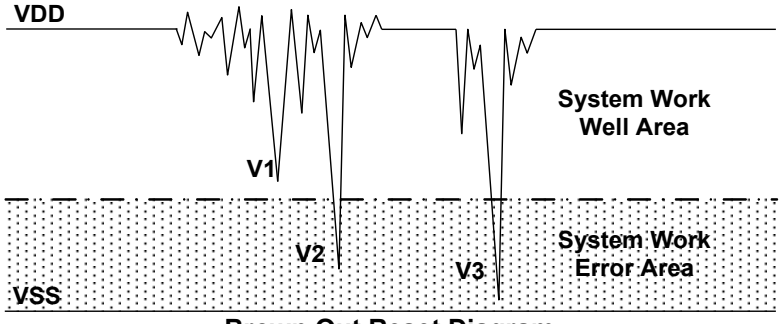
Watchdog timer application note is as following.

- 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

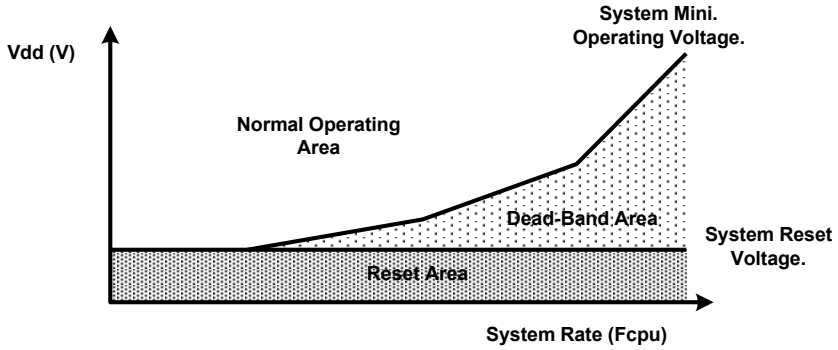
The power dropping might through the voltage range that's the system dead-band. The dead-band means the power range can't offer the system minimum operation power requirement. The above diagram is a typical brown out reset diagram. There is a serious noise under the VDD, and VDD voltage drops very deep. 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 the system operation is well. 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.

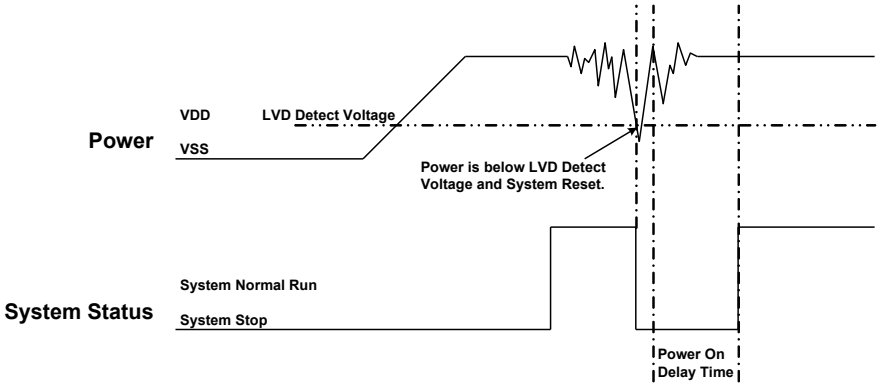
3.5 THE SYSTEM OPERATING VOLTAGE

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.

3.6 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 LVD includes LVD reset function and flag function to indicate VDD status function. 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	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". Or 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". Or VDD ≦2.4V, LVD24 flag is "1".
Enable LVD36 bit of PFLAG register. If VDD > 3.6V, LVD36 is "0". Or VDD ≦3.6V, LVD36 flag is "1".
LVD_MAX
If VDD < 3.6V, system will be reset.
Disable LVD24 and LVD36 bit of PFLAG register.

✷ ***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.***

3.7 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 16MHz/16 (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.

3.8 EXTERNAL RESET

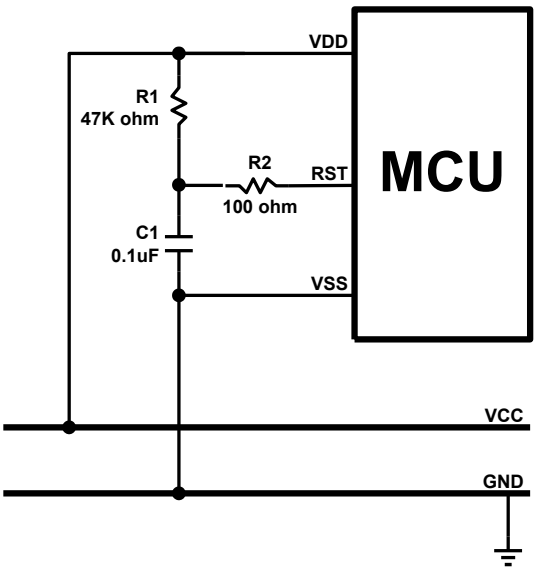
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

3.9 EXTERNAL RESET CIRCUIT

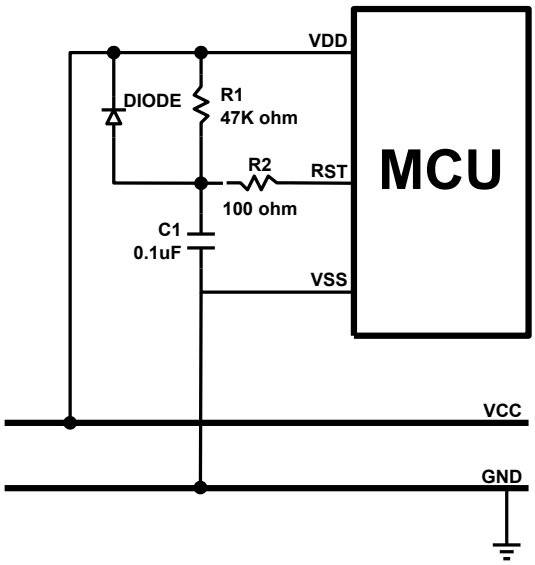
3.9.1 Simply RC Reset Circuit



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 occur to a power on signal from the timing difference.

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

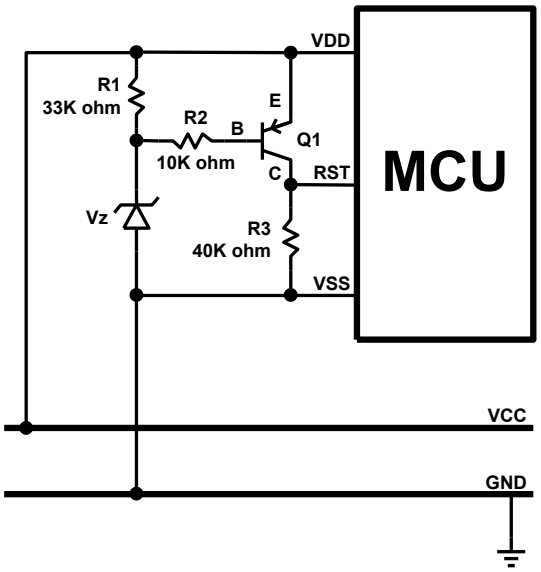
3.9.2 Diode & RC Reset Circuit



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.

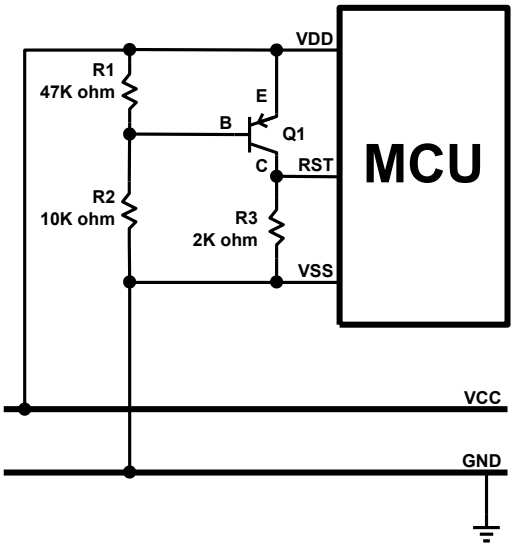
✷ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.9.3 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.

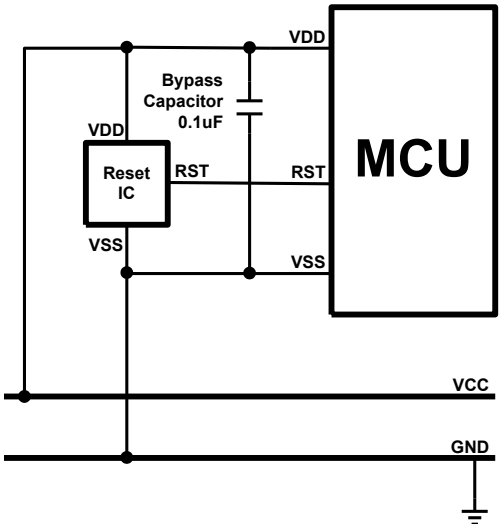
3.9.4 Voltage Bias Reset Circuit



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.***

3.9.5 External Reset IC



The external reset circuit is also to 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 is internal high-speed oscillator. The low-speed clock is 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.

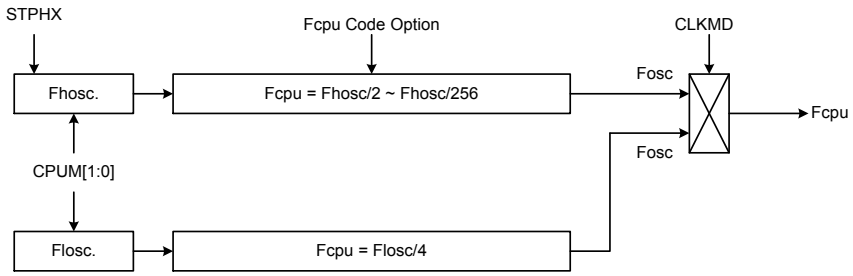
- **High-speed oscillator**

Internal high-speed oscillator is 32MHz RC type called "**IHRC**".

- **Low-speed oscillator**

Internal low-speed oscillator is 16KHz @3V, 32KHz @5V RC type called "**ILRC**".

- **System clock block diagram**



- Fhosc: Internal high-speed RC clock.
- Flosc: Internal low-speed RC clock (about 16KHz@3V and @5V).
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.2 FCPU (INSTRUCTION CYCLE)

The system clock rate is instruction cycle called "**Fcpu**" which is divided from the system clock source and decides the system operating rate. Fcpu rate is selected by Fcpu code option and the range is **Fhosc/2~Fhosc/256** under system normal mode. If the Fcpu code option is Fhosc/4, the Fcpu frequency is 32MHz/4 = 8MHz. Under system slow mode, the Fcpu is fixed Flosc/4, 16KHz/4=4KHz @3V, 32KHz/4=8KHz @5V.

4.3 SYSTEM HIGH-SPEED CLOCK

The internal high-speed oscillator is 32MHz RC type. The accuracy is ±2% under commercial condition.

4.4 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 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.***

4.5 OSCM REGISTER

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

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

Bit 1 **STPHX:** High-speed oscillator control bit.
0 = The high-speed oscillator free run.
1 = The high-speed oscillator stops. 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 operation. When "STPHX=0", internal high speed RC type oscillator active. When "STPHX=1", the internal high speed RC type oscillator is disabled.

- **"STPHX=1" disables internal high speed RC type oscillator.**

4.6 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.

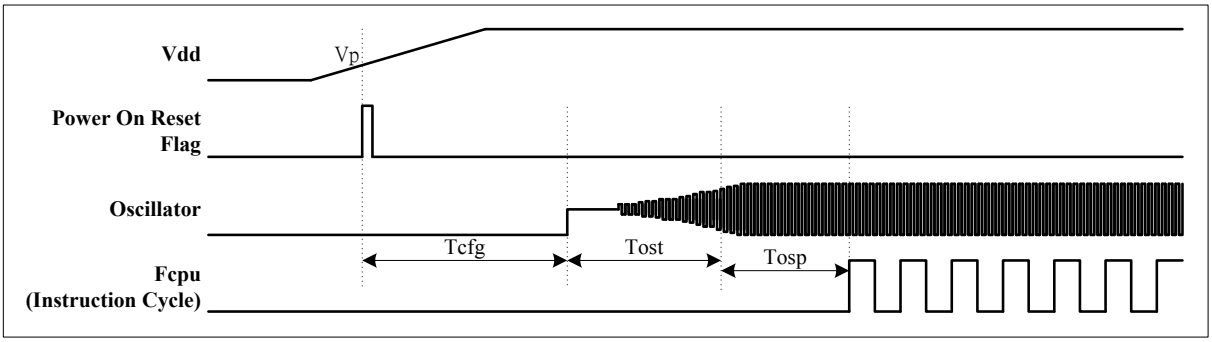
➢ **Example: Fcpu instruction cycle of internal oscillator.**

```
        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
```

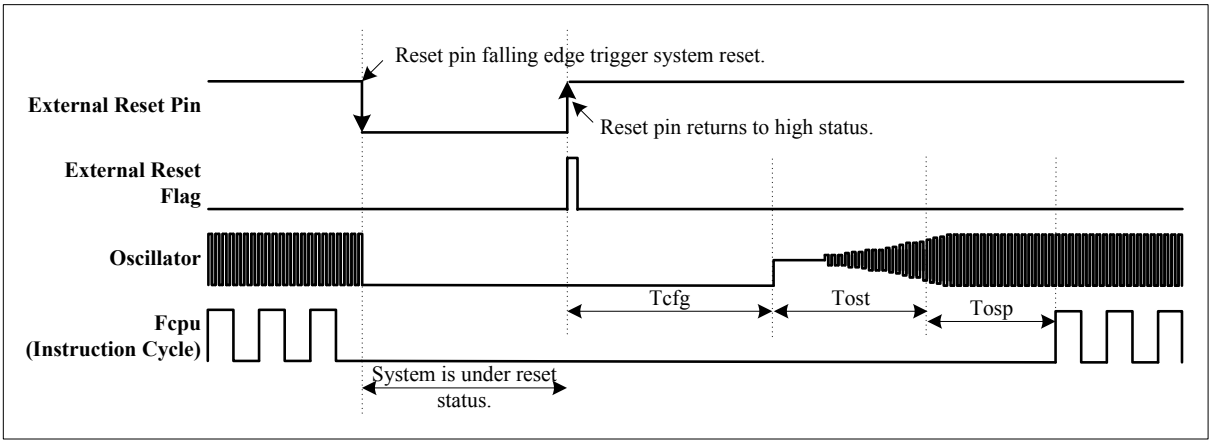
4.7 SYSTEM CLOCK TIMING

Parameter	Symbol	Description	Typical
Hardware configuration time	Tcfg	64*F_{ILRC}	2ms @ F_{ILRC} = 32KHz 4ms @ F_{ILRC} = 16KHz
Oscillator start up time	Tost	The start-up time is depended on oscillator's material, factory and architecture. The internal high speed RC type oscillator's start-up time is very short and ignored.	-
Oscillator warm-up time	Tosp	Oscillator warm-up time of reset condition. 1024*F_{hosc} (Power on reset, LVD reset, watchdog reset, external reset pin active.)	32us @ F_{hosc} = 32MHz
		Oscillator warm-up time of power down mode wake-up condition. 64*F_{hosc}……Internal high-speed RC type oscillator.	2us @ F_{hosc} = 32MHz

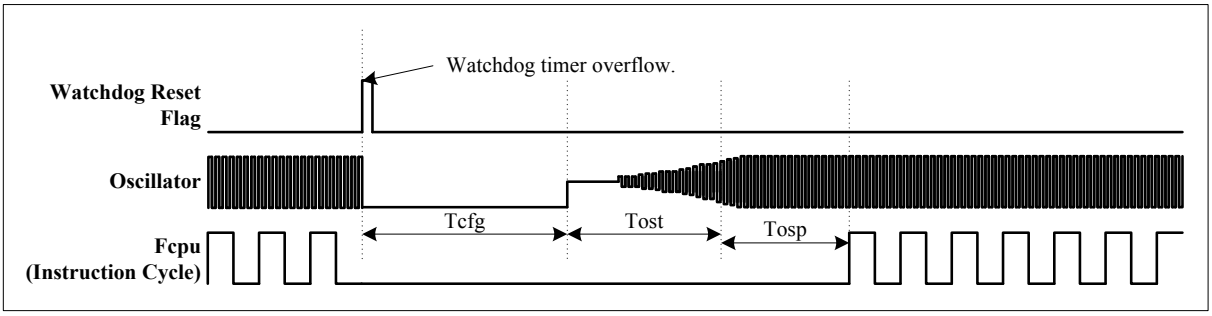
- **Power On Reset Timing**



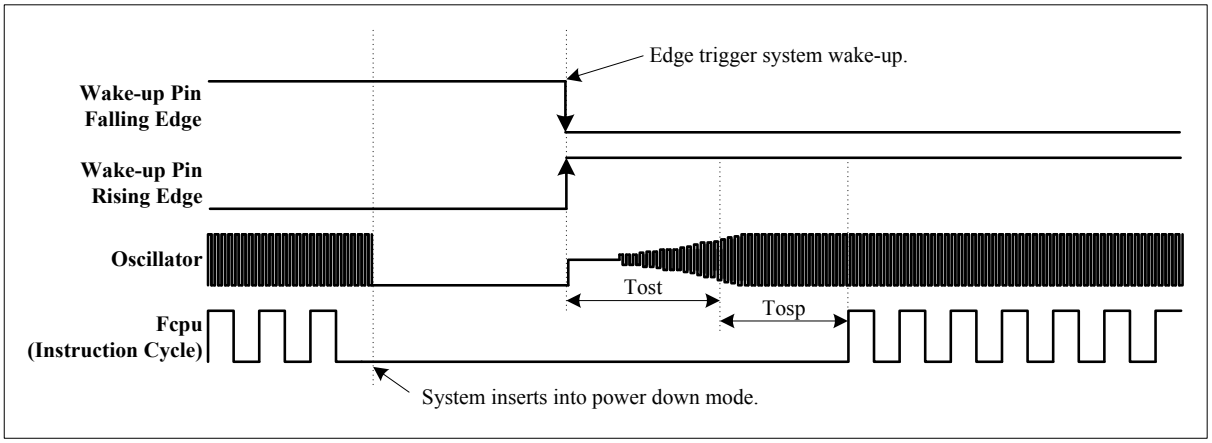
- **External Reset Pin Reset Timing**



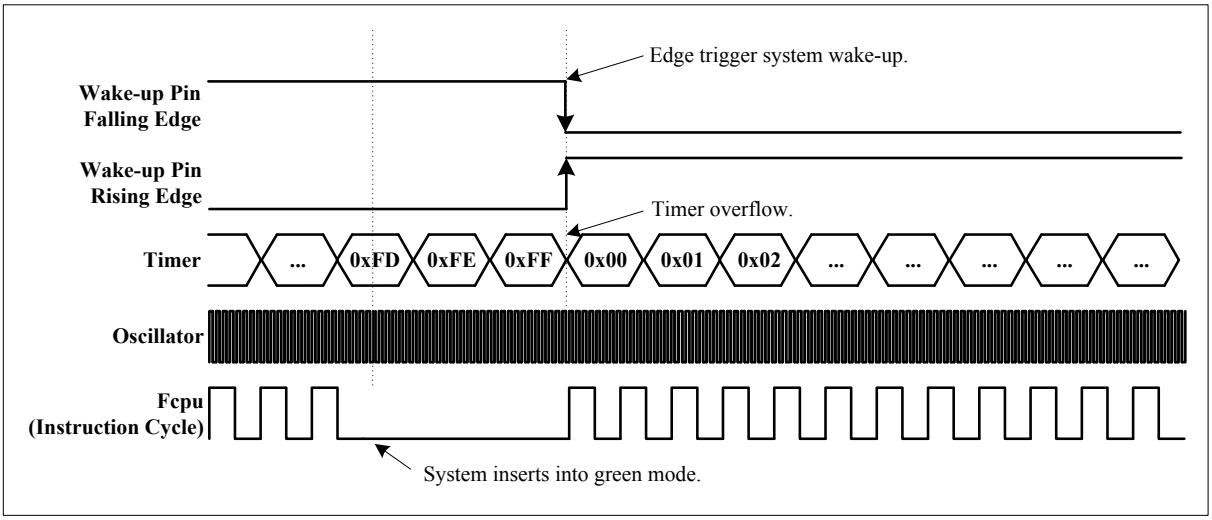
- **Watchdog Reset Timing**



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

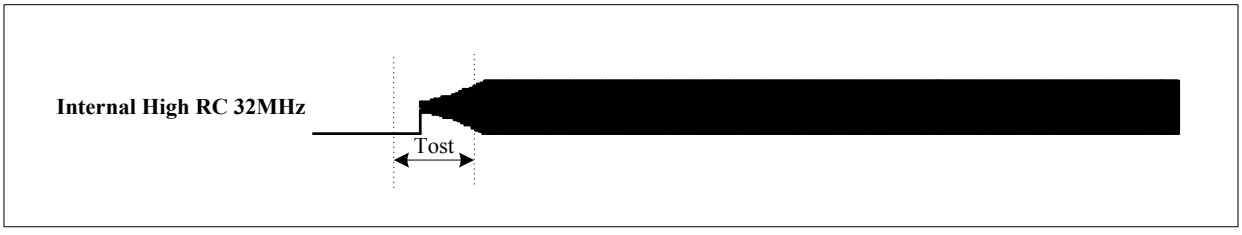


- **Green Mode Wake-up Timing**



- **Oscillator Start-up Time**

The start-up time is depended on oscillator's material, factory and architecture. The internal high speed RC type oscillator's start-up time is very short and ignored.



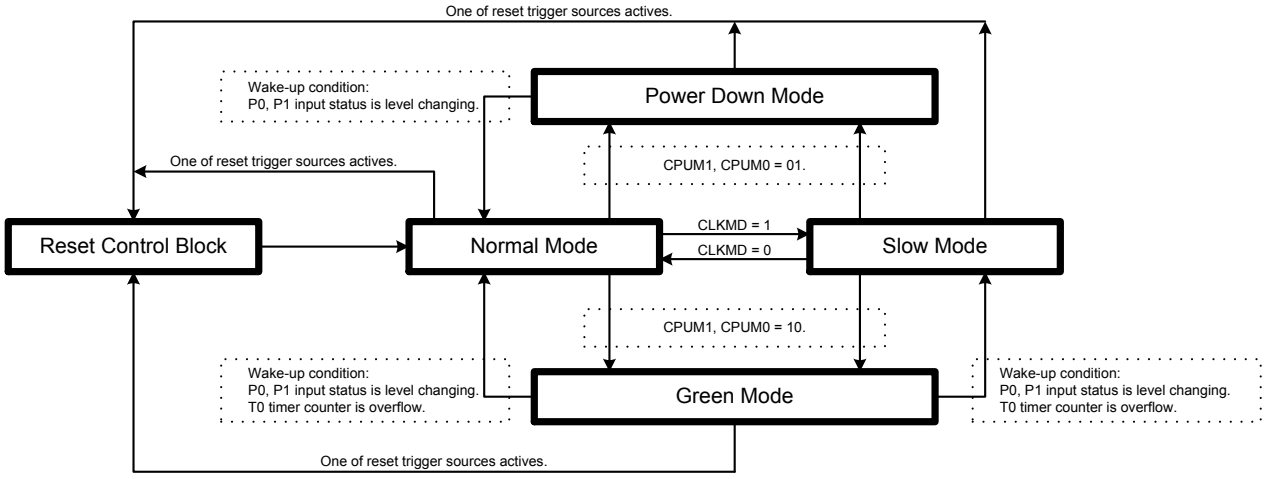
5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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

Operating Mode Control Block



Operating Mode Clock Control Table

Operating Mode	Normal Mode	Slow Mode	Green Mode	Power Down Mode
IHRC	Running	By STPHX	By STPHX	Stop
ILRC	Running	Running	Running	Stop
CPU instruction	Executing	Executing	Stop	Stop
T0 timer	By T0ENB	By T0ENB	By T0ENB	Inactive
PWM Generator	By PW1EN	By PW1EN	By PW1EN (PWM active)	Inactive
ADC	By ADENB/ADS	By ADENB/ADS	By ADENB/ADS	Inactive
Comparator 0	By CM0EN	By CM0EN	By CM0EN	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	All inactive
External interrupt	All active	All active	All active	All inactive
Wakeup source	-	-	P0, P1, T0 Reset	P0, P1 Reset

- **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 MODE

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 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 are special function with wake-up function (T0 timer occurring overflow, ADC converting finish, Comparator 0 output trigger). That's mean users can setup one fix period to timer, and the system is waked up until the time out. User can setup ADC register's ADS bit as 1. ADC is converting and waiting for ADC converting finished. User can set Comparator 0 enable, comparator 0 output trigger.
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.
- The green mode wake-up source is ADC converting finished trigger.
- The green mode wake-up source is comparator 0 output trigger.
- PWM 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 high-speed oscillator stops).**

```
    Slow2Normal                 ; Declare “Slow2Normal” macro directly.
```

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

```
    GreenMode                   ; Declare “GreenMode” macro directly.
```

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

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

; Go into green mode
    GreenMode                   ; Declare “GreenMode” macro directly.
```

5.7 WAKEUP

5.7.1 OVERVIEW

Under power down mode (sleep mode) or green mode, program doesn't execute. The wakeup trigger can wake the system up to normal mode or slow mode. The wakeup trigger sources are external trigger (P0/P1 level change) and internal trigger (T0 overflow / Comparator 0 output trigger / ADC converting finish).

- 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 overflow / Comparator 0 output trigger / ADC converting finish).

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

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

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

The wakeup time = 1/Fosc * 64 = 2 us (Fhosc = 32MHz)

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.

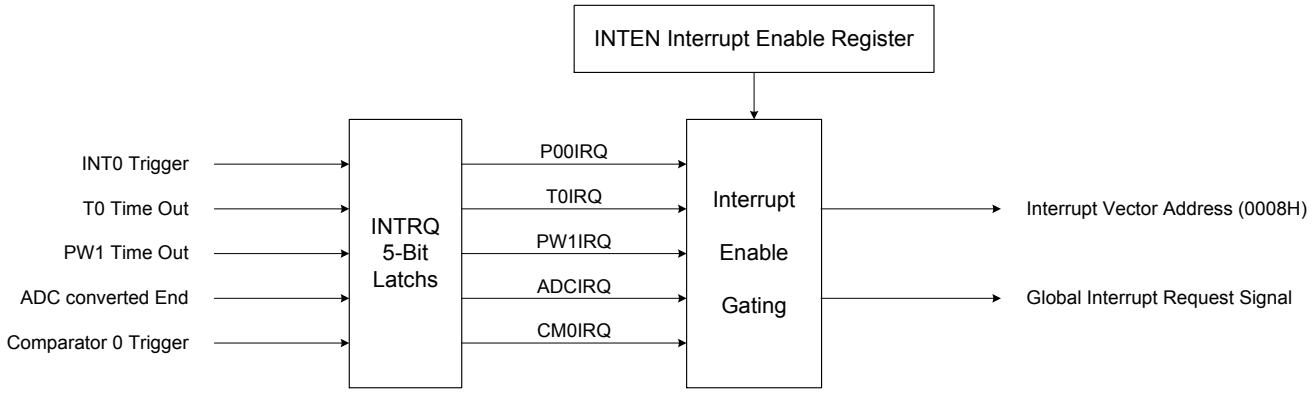
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	-	P16W	P15W	P14W	P13W	P12W	P11W	P10W
Read/Write	-	W	W	W	W	W	W	W
After reset	-	0	0	0	0	0	0	0

Bit[6:0] **P10W~P16W:** Port 1 wakeup function control bits.
0 = Disable P1n wakeup function.
1 = Enable P1n wakeup function.

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 5 interrupt sources, including 4 internal interrupt (T0/PW1/CM0/ADC) and 1 external interrupt (INT0). The external interrupt can wake up the chip while the system is switched from power down mode to high-speed normal mode, and interrupt request is latched until return to normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to "0" for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to "1" to accept the next interrupts' request. The interrupt request signals are stored in INTRQ register.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

INTEN is the interrupt request control register including three internal interrupts, two external interrupts enable control bits. One of the register to be set "1" is to enable the interrupt request function. Once of the interrupt occur, the stack is incremented and program jump to ORG 8 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	CM0IEN	PW1IEN	T0IEN	-	-	-	P00IEN
Read/Write	R/W	R/W	R/W	R/W	-	-	-	R/W
After reset	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 4 **T0IEN:** T0 timer interrupt control bit.
0 = Disable T0 interrupt function.
1 = Enable T0 interrupt function.

Bit 5 **PW1IEN:** PWM interrupt control bit.
0 = Disable PWM interrupt function.
1 = Enable PWM interrupt function.

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

Bit 7 **ADCIEN:** ADC interrupt control bit.
0 = Disable ADC interrupt function.
1 = Enable ADC 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	CM0IRQ	PW1IRQ	T0IRQ	-	-	-	P00IRQ
Read/Write	R/W	R/W	R/W	R/W	-	-	-	R/W
After reset	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 4 **T0IRQ:** T0 timer interrupt request flag.
0 = None T0 interrupt request.
1 = T0 interrupt request.

Bit5 **PW1IRQ:** PWM interrupt request flag.
0 = None PWM interrupt request.
1 = PWM interrupt request.

Bit 6 **CM0IRQ:** Comparator 0 interrupt request flag.
0 = None comparator 0 interrupt request.
1 = Comparator 0 interrupt request.

Bit 7 **ADCIRQ:** ADC interrupt request flag.
0 = None ADC interrupt request.
1 = ADC 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	-	-	-	-	-	STKPB1	STKPB0
Read/Write	R/W	-	-	-	-	-	R/W	R/W
After reset	0	-	-	-	-	-	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
```

✸ ***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)

INT0 is external interrupt trigger source and builds in edge trigger configuration function. When the external edge trigger occurs, the external interrupt request flag will be set to "1" no matter the external interrupt control bit enabled or disable. 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), 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	-	-	-	P00G1	P00G0	-	-	-
Read/Write	-	-	-	R/W	R/W	-	-	-
After reset	-	-	-	0	0	-	-	-

Bit[4:3] **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, #18H
        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.**

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

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

        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: T0 interrupt service routine.**

```
        ORG         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, #74H
        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 PWM INTERRUPT OPERATION

When the PWM counter overflows, the PW1IRQ will be set to "1" no matter the PW1IEN is enable or disable. If the PW1IEN and the trigger event PW1IRQ is set to be "1". As the result, the system will execute the interrupt vector. If the PW1IEN = 0, the trigger event PW1IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the PW1IEN is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: PWM interrupt request setup. Fhosc32 = 32MHz / 32.**

```
        B0BCLR      FPW1IEN             ; Disable PWM interrupt service
        B0BCLR      FPW1ENB             ; Disable PWM function
        MOV         A, #00100000B       ;
        B0MOV       PW1M, A             ; Set PWM clock = Fhosc32 / 32
        MOV         A, #9CH             ; Set PW1L/PW1H initial value = 9CH
        B0MOV       PW1YL, A            ; Set PWM interval timer = 100 us
        MOV         A, #03H
        B0MOV       PW1YH, A

        B0BSET      FPW1IEN             ; Enable PWM interrupt service
        B0BCLR      FPW1IRQ             ; Clear PWM interrupt request flag
        B0BSET      FPW1ENB             ; Enable PWM function

        B0BSET      FGIE                ; Enable GIE
```

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

```
        ORG         8                   ; Interrupt vector.
        JMP         INT_SERVICE         ; executed by hardware automatically.
INT_SERVICE:
        PUSH                            ; Push routine to save ACC and PFLAG to buffers.
        …

        B0BTS1      FPW1IRQ             ; Check PW1IRQ
        JMP         EXIT_INT            ; PW1IRQ = 0, exit interrupt vector

        B0BCLR      FPW1IRQ             ; Reset PW1IRQ
        …                               ; PWM interrupt service routine
        …
EXIT_INT:
        …
        POP                             ; Pop routine to load ACC and PFLAG from buffers.
        RETI                            ; Exit interrupt vector.
```

6.9 ADC INTERRUPT OPERATION

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

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

```
        B0BCLR      FADCIEN         ; Disable ADC interrupt service

        MOV         A, #10110000B   ;
        B0MOV       ADM, A          ; Enable P4.0 ADC input and ADC function.
        MOV         A, #00000000B   ; Set ADC converting rate = Fhosc/4
        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
```

6.10 COMPARATOR INTERRUPT OPERATION (CMP0)

Sonix provides one comparator with interrupt function in the micro-controller. The comparator interrupt trigger edge direction depends on CM0G [1:0]. 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 08H) and execute interrupt service routine.

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M	CM0EN	CM0T	CM0S1	CM0S0	CM0OEN	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 [1:0] **CM0G [1:0]:** Comparator 0 interrupt trigger direction control bit.
00 = Reserved
01 = Rising edge trigger. Comparator output status is from low to high as CM0P > CM0N.
10 = Falling edge trigger. Comparator output status is from high to low as CM0P < CM0N.
11 = Both rising and falling edge trigger. Comparator output status are from low to high as CM0P > CM0N and from high to low as CM0P < CM0N.

➢ **Example: Setup comparator 0 interrupt request.**

```
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       08H              ; Interrupt vector
          JMP       INT_SERVICE
INT_SERVICE:
          PUSH                       ; 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                        ; Pop routine to load ACC and PFLAG from buffers.
          RETI                       ; Exit interrupt vector
```

6.11 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.
T0IRQ	T0C overflow.
PW1IRQ	PWM counter overflow.
CM0IRQ	Comparator 0 output level transition.
ADCIRQ	ADC converting end.

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. Saving ACC and PFLAG to buffers
        JMP     INT_SERVICE     ; executed by hardware automatically.
INT_SERVICE:
        PUSH                    ; 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     INTPW1CHK       ; Jump check to next interrupt
        B0BTS0  FT0IRQ          ; Check T0IRQ
        JMP     INTT0           ; Jump to T0 interrupt service routine
INTPW1CHK:                      ; Check PWM interrupt request
        B0BTS1  FPW1IEN         ; Check PW1IEN
        JMP     INTCM0CHK       ; Jump check to next interrupt
        B0BTS0  FPW1IRQ         ; Check PW1IRQ
        JMP     INTPW1          ; Jump to PW1 interrupt service routine
INTCM0CHK:                      ; Check Comparator 0 interrupt request
        B0BTS1  FCM0IEN         ; Check CM0IEN
        JMP     INTADCCHK       ; Jump check to next interrupt
        B0BTS0  FCM0IRQ         ; Check CM0IRQ
        JMP     INTCM0          ; Jump to CM0 interrupt service routine
INTADCCHK:                      ; 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                     ; 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 14 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.
P0.2	I/O	PWM10A	DC	PW1EN=1 and PWCH10A=1.
		QCP	AC	QCEN=1.
P0.3	I/O	PWM11A	DC	PW1EN=1 and PWCH11A=1.
		QCN	AC	QCEN=1.
P0.4	I/O	RST	DC	Reset_Pin code option = Reset.
		VPP	HV	OTP Programming.
P0.5	I/O	PWM10	DC	PW1EN=1 and PWCH10=1.
P0.6	I/O	PWM11	DC	PW1EN=1 and PWCH11=1.
P1.0	I/O	AIN0	AC	ADENB=1, CHS[2:0] = 000b.
	I/O	AVREFH	AC	ADENB=1, EVHENB = 1.
P1.1	I/O	AIN1	AC	ADENB=1, CHS [2:0] = 001b.
P1.2	I/O	AIN2	AC	ADENB=1, CHS [2:0] = 010b.
P1.3	I/O	AIN3	AC	ADENB=1, CHS [2:0] = 011b.
P1.4	I/O	AIN4	AC	ADENB=1, CHS [2:0] = 100b.
		CM0O	DC	CM0EN=1, CM0OEN=1.
P1.5	I/O	PWM10B	DC	PW1EN=1 and PWCH10B=1
		AIN5	AC	ADENB=1, CHS [2:0] = 101b.
		CM0P	AC	CM0EN=1, CM0S [1:0] = 00b.
P1.4	I/O	PWM11B	DC	PW1EN=1 and PWCH11B=1.
		AIN6	AC	ADENB=1, CHS [2:0] = 110b.
		CM0N	AC	CM0EN=1.

*** 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	-	P06M	P05M	P04M	P03M	P02M	P01M	P00M
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

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	-	P16M	P15M	P14M	P13M	P12M	P11M	P10M
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	0	0	0	0	0	0	0

Bit[6:0] **PnM[6:0]:** Pn mode control bits. (n = 0~1).
0 = Pn is input mode.
1 = Pn is output mode.

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

➢ **Example: I/O mode selecting**

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

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

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

7.3 I/O PULL UP REGISTER

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

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	P06R	P05R	-	P03R	P02R	P01R	P00R
Read/Write	-	W	W	-	W	W	W	W
After reset	-	0	0	-	0	0	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	-	P16R	P15R	P14R	P13R	P12R	P11R	P10R
Read/Write	-	W	W	W	W	W	W	W
After reset	-	0	0	0	0	0	0	0

✸ ***Note: P0.4 pin is without pull-up resister. The P0UR.4 is undefined.***

➢ **Example: I/O Pull up Register**

```
MOV     A, #06FH        ; Enable Port0,1 Pull-up register,
B0MOV   P0UR, A         ;
B0MOV   P1UR,A
```

7.1 I/O PULL DOWN REGISTER

The P0.0, P0.1, P0.5, P0.6 I/O pins build in internal pull-down resistors and only support I/O input mode. The port internal pull-down resistor is programmed by P0DR register. When the bit of P0DR register is "0", the I/O pin's pull-down is disabled. When the bit of P0DR register is "1", the I/O pin's pull-down is enabled

0D6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0DR	P0CON6	P0CON5	P0CON1	P0CON0	P06DR	P05DR	P01DR	P00DR
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Bit 3 **P06DR:** P0.6 I/O port pull-down resistor control bit.
0 = Disable P0.6 pull-down resistor.
1 = Enable P0.6 pull-down resistor.

Bit 2 **P05DR:** P0.5 I/O port pull-down resistor control bit.
0 = Disable P0.5 pull-down resistor.
1 = Enable P0.5 pull-down resistor.

Bit 1 **P01DR:** P0.1 I/O port pull-down resistor control bit.
0 = Disable P0.1 pull-down resistor.
1 = Enable P0.1 pull-down resistor.

Bit 0 **P00DR:** P0.0 I/O port pull-down resistor control bit.
0 = Disable P0.0 pull-down resistor.
1 = Enable P0.0 pull-down resistor.

➢ **Example: I/O Pull-down Register**

```
MOV     A, #0FH     ; Enable P0.0, P0.1, P0.5, P0.6 Pull-down register,
B0MOV   P0DR, A     ;
```

7.2 PORT0 SCHMITT-TRIGGER CONTROL REGISTER

P0.0, P0.1, P0.5, P0.6 I/O pins build in pull-up and pull-down resistors. If one pin's pull-up and pull-down are enabled together, the pin is in 1/2*VDD status that makes Schmitt-trigger in half-level status and induces current. To improve the problem, P0.0, P0.1, P0.5, P0.6 build in Schmitt-trigger control circuit by P0DR [7:4] register bits. When the bit of P0DR [7:4] register is "0", the I/O pin's Schmitt-trigger is enabled. When the bit of P0DR [7:4] register is "1", the I/O pin's Schmitt-trigger is disabled. If the pull-up and pull-down resistors are enabled together, the Schmitt-trigger control bit must be set to disable Schmitt-trigger.

0D6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0DR	P0CON6	P0CON5	P0CON1	P0CON0	P06DR	P05DR	P01DR	P00DR
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Bit 7 **P0CON6:** P0.6 Schmitt-trigger control bit.
0 = Enable P0.6 Schmitt-trigger.
1 = Disable P0.6 Schmitt-trigger.

Bit 6 **P0CON5:** P0.5 Schmitt-trigger control bit.
0 = Enable P0.5 Schmitt-trigger.
1 = Disable P0.5 Schmitt-trigger.

Bit 5 **P0CON1:** P0.1 Schmitt-trigger control bit.
0 = Enable P0.1 Schmitt-trigger.
1 = Disable P0.1 Schmitt-trigger.

Bit 4 **P0CON0:** P0.0 Schmitt-trigger control bit.
0 = Enable P0.0 Schmitt-trigger.
1 = Disable P0.0 Schmitt-trigger.

➢ **Example: P0.0, P0.1, P0.5, P0.6 Schmitt-trigger Register**

```
MOV     A, #0F0H        ; Disable P0.0, P0.1, P0.5, P0.6 Schmitt-trigger.,
B0MOV   P0DR, A         ;
```

✸ ***Note: P0.0, P0.1, P0.5, P0.6 Schmitt trigger control for low power consumption under P0.0, P0.1, P0.5, P0.6 input half voltage.***

7.3 I/O PORT DATA REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	-	P06	P05	P04	P03	P02	P01	P00
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

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	-	P16	P15	P14	P13	P12	P11	P10
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	0	0	0	0	0	0	0

✸ ***Note: The P04 keeps "1" and P04 as input mode, when external reset enable by code option.***

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

```
B0MOV   A, P0       ; Read data from Port 0
B0MOV   A, P1       ; Read data from Port 1
```

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

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

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

```
B0BSET  P1.0        ; Set P1.0 to be "1".
B0BCLR  P1.0        ; Set P1.0 to be "0".
```

7.4 PORT 1 ADC SHARE PIN

The Port 1 is shared with ADC input function. ADC function and Schmitt trigger structure. Only one pin of port 1 can be configured as ADC input in the same time by ADM register. The other pins of port 1 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 1 will encounter above current leakage situation. P1CON is Port1 Configuration register. Write "1" into P1CON.n will configure related port 1 pin as pure analog input pin to avoid current leakage.

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1CON	-	P1CON6	P1CON5	P1CON4	P1CON3	P1CON2	P1CON1	P1CON0
Read/Write	-	W	W	W	W	W	W	W
After reset	-	0	0	0	0	0	0	0

Bit[6:0] **P1CON[6:0]:** P1.n configuration control bits.
0 = P1.n can be an analog input (ADC input) or digital I/O pins.
1 = P1.n is pure analog input, can't be a digital I/O pin.

✷ ***Note: When Port 1.n is general I/O port not ADC channel, P1CON.n must set to "0" or the Port 1.n digital I/O signal would be isolated.***

Port 1 ADC analog input is controlled by GCHS and CHSn bits of ADM register. If GCHS = 0, P1.n is general purpose bi-direction I/O port. If GCHS = 1, P1.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	-	CHS2	CHS1	CHS0
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 4 **GCHS:** Global channel select bit.
0 = Disable AIN channel.
1 = Enable AIN channel.

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 P1.n general purpose I/O function, users should make sure of P1.n's ADC channel is disabled and set as input floating when GCHS = 1 and CHS[2:0] point to P1.n.***

➢ **Example: Set P1.1 to be general purpose input mode. P1CON.1 must be set as "0".**

```
; Clear GCHS and CHS[2:0] status.
        B0BCLR      FGCHS       ;If CHS[2:0] point to P1.1 (CHS[2:0] = 001B), set GCHS=0
                                ;If CHS[2:0] don't point to P1.1 (CHS[2:0] ≠ 001B), don't
                                care GCHS status.

; Clear P1CON.
        MOV         A,#0nnnnn0nb
        MOV         P1CON,A     ; Enable P1.1 digital function.

; Enable P1.1 input mode.
        B0BCLR      P1M.1       ; Set P1.1 as input mode.
```

➢ **Example: Set P1.1 to be general purpose output. P1CON.1 must be set as "0".**

```
; Clear GCHS and CHS[2:0] status.
        B0BCLR      FGCHS       ;If CHS[2:0] point to P1.1 (CHS[2:0] = 001B), set GCHS=0.
                                ;If CHS[2:0] don't point to P1.1 (CHS[2:0] ≠ 001B), don't
                                care GCHS status.

; Clear P1CON.
        MOV         A,#0nnnnn0nb    ; Enable P1.1 digital function.
        MOV         P1CON,A

; Set P1.1 output buffer to avoid glitch.
        B0BSET      P1.1        ; Set P1.1 buffer as "1".
; or
        B0BCLR      P1.1        ; Set P1.1 buffer as "0".

; Enable P1.1 output mode.
        B0BSET      P1M.1       ; Set P1.1 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.**

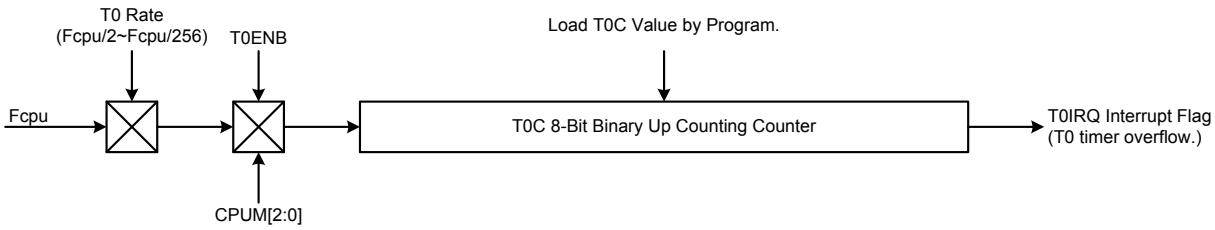
```
Main:
                …                               ; Check I/O.
                …                               ; Check RAM
Err:            JMP $                           ; I/O or RAM error. Program jump here and don't
                                                ; clear watchdog. Wait watchdog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

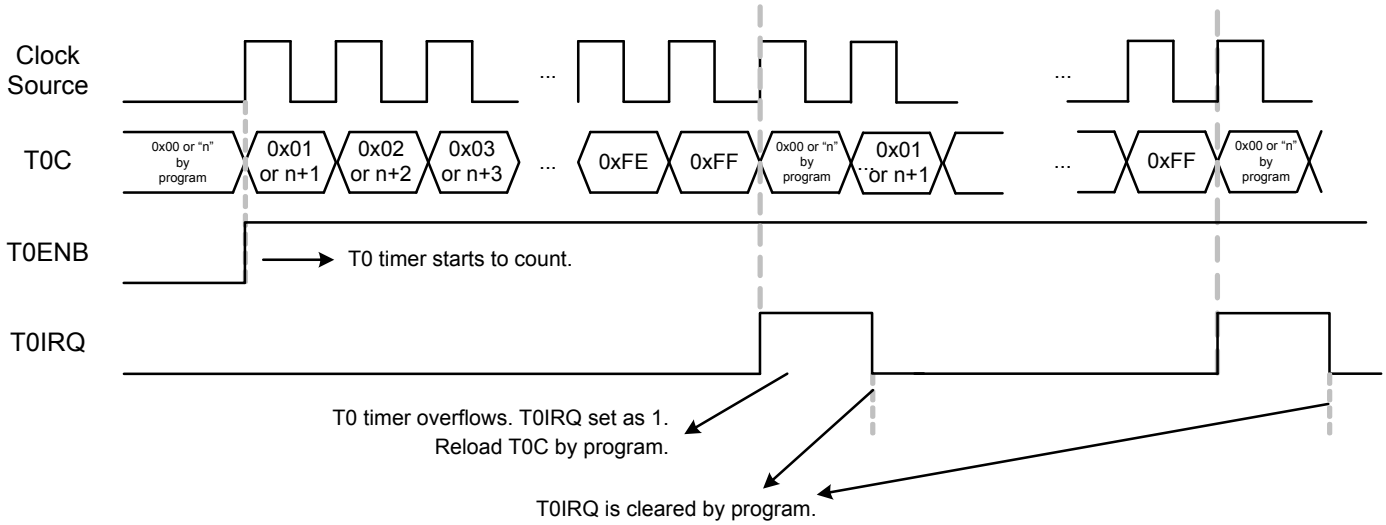
The T0 timer is an 8-bit binary up timer with basic timer function. The basic timer function supports flag indicator (T0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T0M, T0C registers. The T0 builds in green mode wake-up function (FCPUM1 = 1). When T0 timer overflow occurs under green mode, the system will be waked-up to last operating mode.

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



8.2.2 T0 TIMER OPERATION

T0 timer is controlled by T0ENB bit. When T0ENB=0, T0 timer stops. When T0ENB=1, T0 timer starts to count. T0C increases “1” by timer clock source. When T0 overflow event occurs, T0IRQ flag is set as ”1” to indicate overflow and cleared by program. The overflow condition is T0C count from full scale (0xFF) to zero scale (0x00). T0 doesn’t build in double buffer, so load T0C by program when T0 timer overflows to fix the correct interval time. If T0 timer interrupt function is enabled (T0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 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=32MHz, Fcpu=Fhosc/8		Fhosc=32MHz, Fcpu=Fhosc/32	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)
000b	Fcpu/256	16.384	64	65.536	256
001b	Fcpu/128	8.192	32	32.768	128
010b	Fcpu/64	4.096	16	16.384	64
011b	Fcpu/32	2.048	8	8.192	32
100b	Fcpu/16	1.024	4	4.096	16
101b	Fcpu/8	0.512	2	2.048	8
110b	Fcpu/4	0.256	1	1.024	4
111b	Fcpu/2	0.128	0.5	0.512	2

8.2.3 T0M MODE REGISTER

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

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

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

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

8.2.4 T0C COUNTING REGISTER

T0C is T0 8-bit counter. When T0C overflow occurs, the T0IRQ flag is set as “1” and cleared by program. The T0C decides T0 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T0C register, and then enable T0 timer to make sure the fist 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 = 32MHz/32 = 1MHz. Select T0RATE=001 (Fcpu/128).**
  T0 interval time = 10ms. T0 clock rate = 32MHz/32/128

*T0C initial value = 256 - (T0 interval time * input clock)*
*= 256 - (10ms * 32MHz / 32 / 128)*
*= 256 - (10^{-2} * 32MHz / 32 / 128)*
= B2H

8.2.5 T0 TIMER OPERATION EXPLAME

➢ **T0 TIMER CONFIGURATION:**

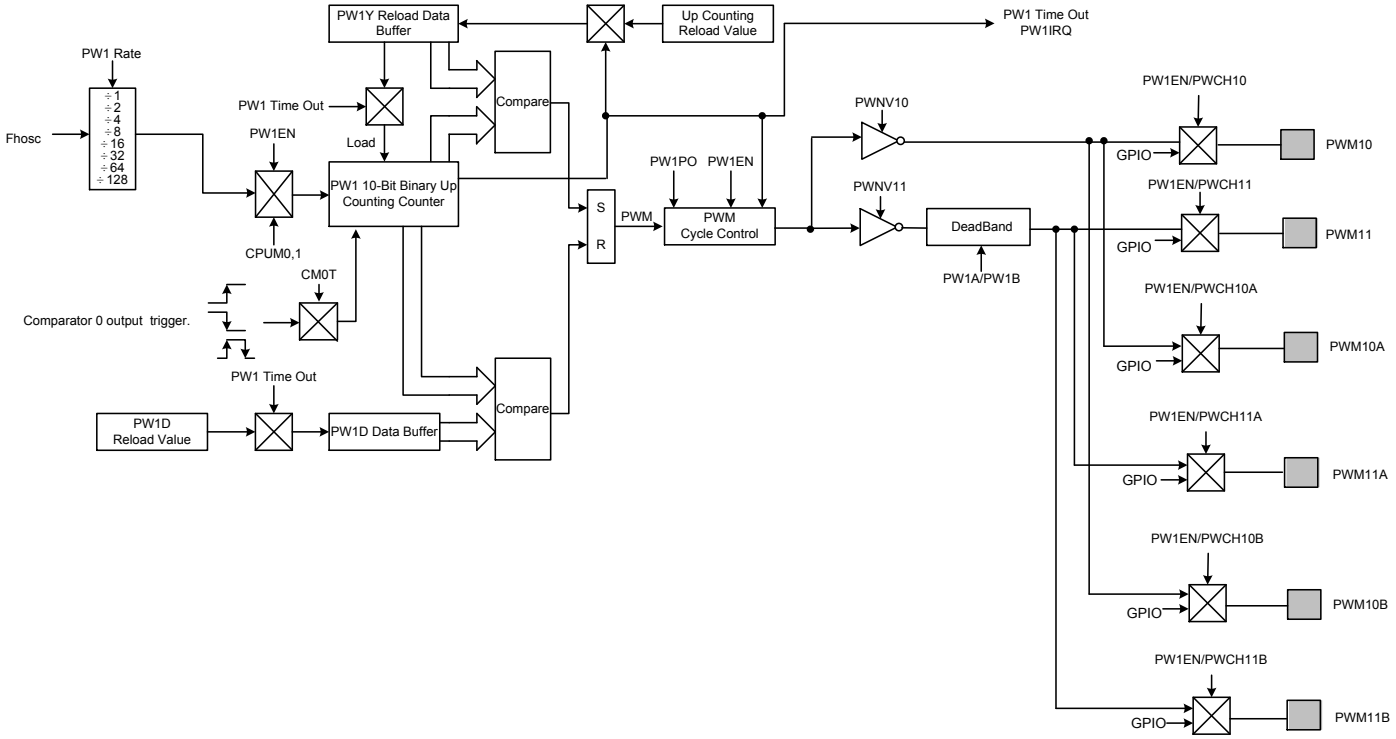
```
; Reset T0 timer.
        CLR     T0M             ; Clear T0M register.

; Set 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.
```

8.3 10-BIT PULSE WIDTH MODULATION (PWM)

8.3.1 OVERVIEW

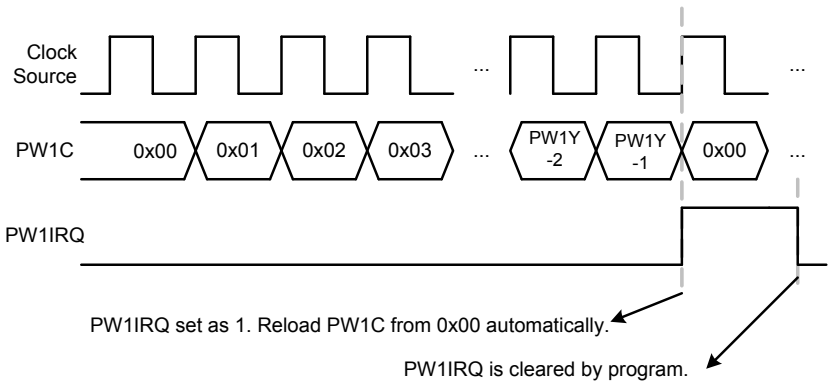
The PWM is duty/cycle programmable and 10-bit resolution. It supports continuous PWM output, one pulse output, inverse output and dead-band output functions. The PWM generator is a 10-bit binary up counter. If PMW counter occurs an overflow (from 0x00 to PWnY-1), it will continue counting and issue a time-out signal to indicate PWM time out event which support interrupt. PWM counter builds in auto-reload function which always enabled. The PWM clock source is 32MHz internal high speed clock source (Fhosc). The PWM clock rate includes Fhosc/1~Fhosc/128. The PWM has 3-pair (6-channel) programmable channels shared with GPIO pins and controlled by PWCH[5:0] bit. One channel of one pair PWM output is normal PWM output channel, and the other pin is with dead-band signal. The output operation must be through enabled each bit/channel of PWCH[5:0] bits, and PWM channels exchange from GPIO to PWM output. When the PWCH[5:0] bits disables, the PWM channel returns to GPIO mode and last status. The PWM dead-band is programmable controlled by PW1A and PW1B output. The PWM also supports comparator trigger mode which trigger PWM stopping output through comparator output changing as abnormal event occurrence. The PWM doesn't build in green mode wake-up function. The main purposes of the PWM are as following.

- **10-bit programmable up counting counter:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** PWM counter supports interrupt function. When PWM counter occurs overflow, the PW1IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by PW1Y and PW1D registers.
- **Programmable dead-band function.** The dead-band function is programmable controlled by PW1A and PW1B registers.
- **Inverse output function.** The inverse output function is controlled by PWNV10 and PWNV11 bits.
- **One Pulse PWM:** The one pulse PWM is controlled by PW1PO bit. When PW1PO=0, PWM is normal PWM function mode. When PW1PO=1, PWM is one pulse function. When PW1EN=1, one pulse outputs and the PW1IRQ is issued as PWM counter overflow, PW1EN bit is cleared automatically, the PWM channel returns to GPIO mode and last status.
- **Comparator trigger mode:** The PWM stopping output is triggered through comparator output changing.



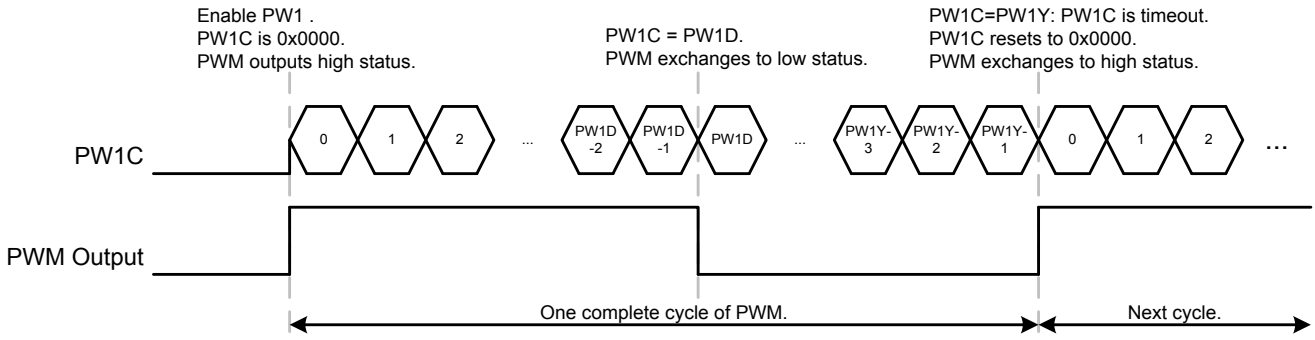
8.3.2 PWM COUNTER OPERATION

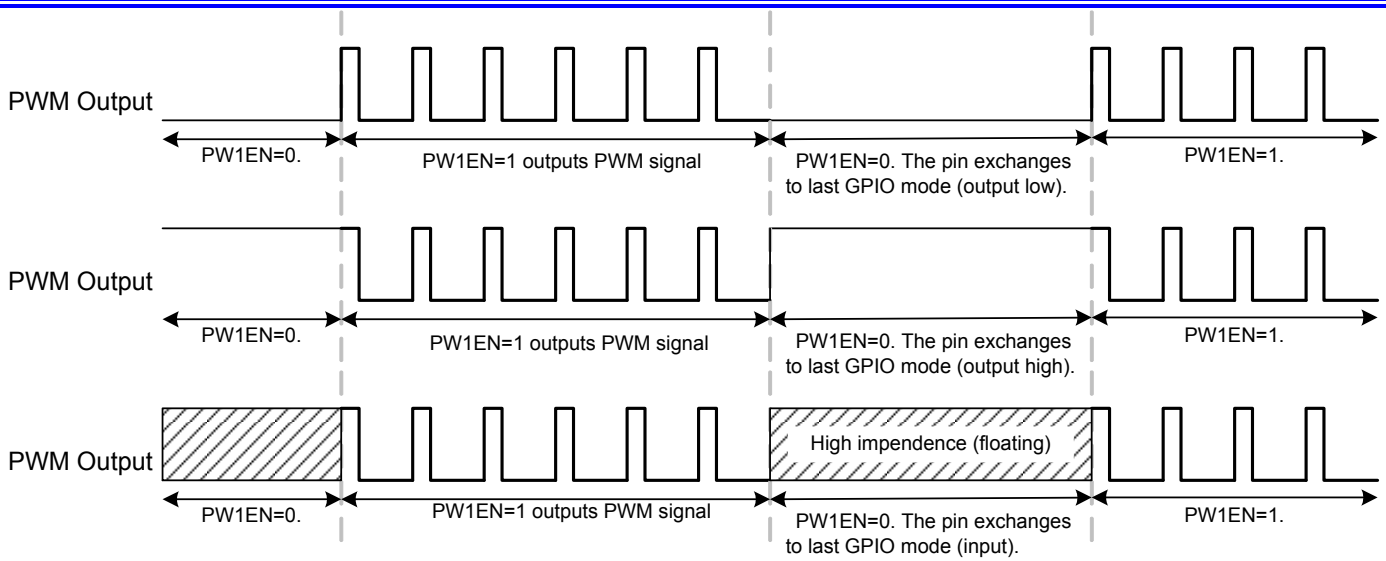
PWM counter is controlled by PW1EN bit. When PW1EN = 1, PWM counter starts to count. One count period is one clock source rate. PW1C is PWM counter and up counting when PW1EN =1. When PW1C counts from 0x0000 to PW1Y-1, PW1 overflow condition is conformed and PW1IRQ set as "1". PWM builds in auto-reload function and always enabled. When PWM counter overflow occurs, the PW1C counter buffer will be reloaded 0x0000 automatically. PWM is double buffer design. If the PW1Y/PW1D/PW1A/PW1B is changed by program, the new value will be loaded at next overflow occurrence, or the PWM interval time is error. If PWM interrupt function is enabled (PW1IEN =1), the program counter is pointed to interrupt vector to execute interrupt service routine after PWM counter overflow occurrence.



8.3.3 PWM OUTPUT FUNCTION

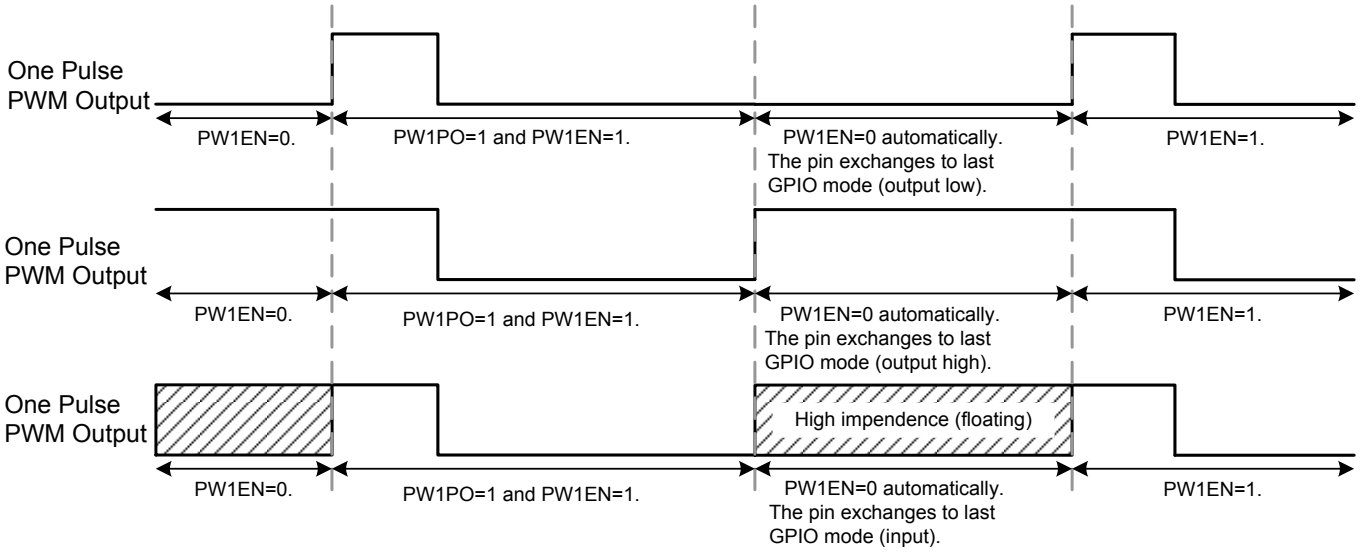
PWM output function controlled by PW1EN and PWCH[5:0] bits. The PWM output pins include PWM10, PWM11, PWM10A, PWM11A, PWM10B, PWM11B 6-pins which are shared with GPIO pins controlled by PWCH[5:0] bits. PWM10, PWM10A, PWM10B are normal PWM output signal, and PWM11, PWM11A, PWM11B are dead-band output pins. PWM output signal is generated from PWM counter controlled by PW1EN bit. When PW1EN = 0, the PWM channel is GPIO mode, so the WPM idle status is controlled by GPIO status. When PW1EN = 1, PWM output pins exchange GPIO mode to PWM output mode to output PWM signal. The PWM signal is generated from the comparison result between PW1C, PW1Y and PW1D. When PW1C starts to count from 0x0000, the PWM output high status. If PW1C=PW1D, the PWM output status exchanges to low. If PW1C=PW1Y, PWM counter is timeout, and one cycle PWM signal finishes. PW1C reset to 0x0000 automatically, and PWM output status exchanges to high status for next cycle. PW1D decides the high duty duration, and PW1Y decides the resolution and cycle of PWM. PW1D can't be larger than PW1Y, or the PWM signal is error.





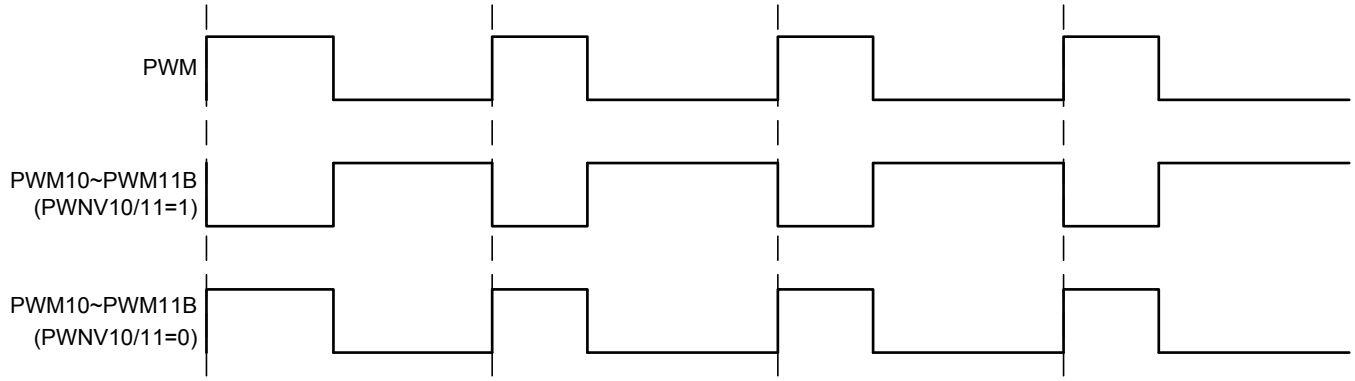
8.3.4 ONE PULSE OUTOUT FUNCTION

The PWM builds in one cycle output function controlled by PW1PO bit. When PW1PO=1 and PW1EN=1, PWM will output one pulse PWM function and the PW1IRQ is issued as PWM counter overflow. PW1EN bit is cleared automatically and pulse output pin returns to idle status. To output next pulse is to set PW1EN bit by program again.



8.3.5 PWM INVERSE OUTPUT

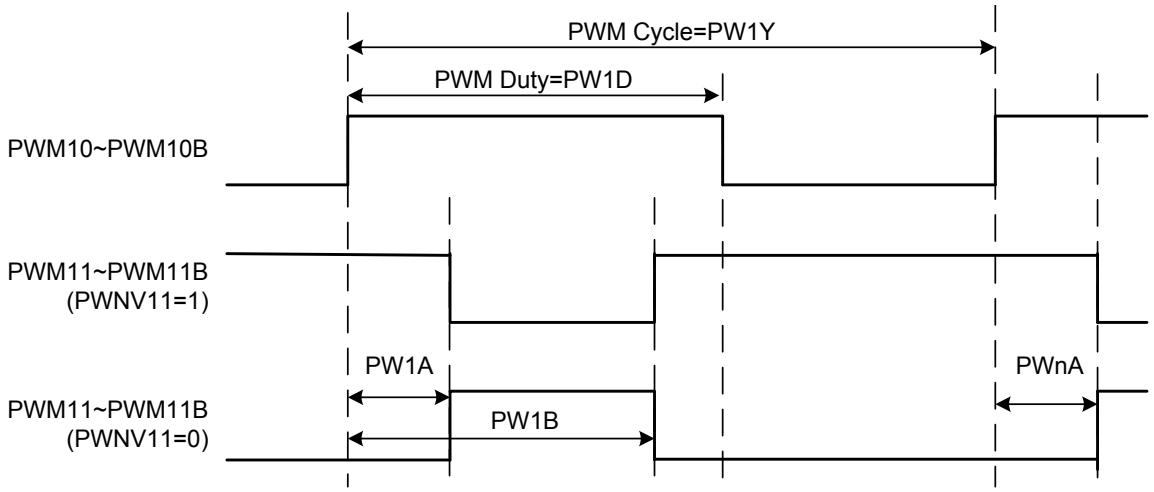
The PWM builds in inverse output function. The PWM has one inverse PWM signal as PWNV10 = 1 or PWNV11 = 1. When PWNV10 = 1 or PWNV11 = 1, the PWM outputs the inverse PWM signal. When PWNV10 = 0 or PWNV11 = 0, the PW1 outputs the non-inverse PWM signal. The inverse PWM output waveform is below diagram.



8.3.6 PWM DEADBAND FUNCTION

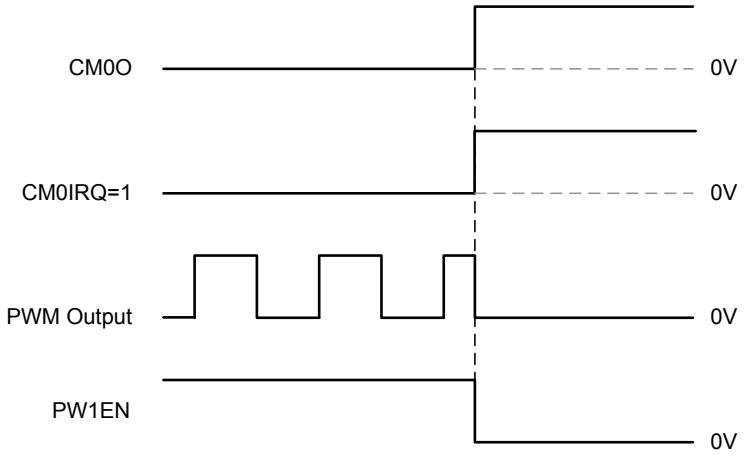
The PWM builds in dead-band output function in the PWM11, PWM11A, PWM11B pins. The dead-band time is controlled by PW1A and PW1B registers. PW1A decides the first dead-band of the starts of the PWM output pulse. PWM1B decides the second dead-band of the end of the PWM output pulse.

The start dead-band time = PW1A/PWM clock rate.
The end dead-band time = (PW1D-PW1B)/PWM clock rate.



8.3.7 COMPARATOR TRIGGER MODE

The PWM can be stop by comparator output changing automatically controlled by CM0T bit. When PW1EN=1 and CM0T=1, comparator 0 output status equals to trigger direction controlled by CM0G [1:0] bits, and the PWM is stopped, PW1EN bit is cleared automatically.



8.3.8 PWM CONTROL REGISTERS

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1M	PW1EN	PW1rate2	PW1rate1	PW1rate0	PWNV11	PWNV10	-	PW1PO
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 **PW1EN:** PWM enable control bit.
0 = Disable PWM output function, and PWM10~PWM11B are GPIO mode (controlled by PWCH[5:0]).
1 = Enable PWM output function, and PWM10~PWM11B outputs PWM signal (controlled by PWCH[5:0]).
If PW1EN = 1 and PWCH = 0, PWM doesn't output and still GPIO mode.

Bit [6:4] **PW1RATE [2:0]:** PWM clock source select bits.
000 = Fhosc/128, 001 = Fhosc/64, 010 = Fhosc/32, 011 = Fhosc/16, 100 = Fhosc/8, 101 = Fhosc/4, 110 = Fhosc/2, 111 = Fhosc/1.

Bit 3 **PWNV11:** PWM11, PWM11A, PWM11B inverse output control bit.
0 = PWM11, PWM11A, PWM11B outputs PWM non-inverse signal.
1 = PWM11, PWM11A, PWM11B outputs PWM inverse signal.

Bit 2 **PWNV10:** PWM10, PWM10A, PWM10B inverse output control bit.
0 = PWM10, PWM10A, PWM10B outputs PWM non-inverse signal.
1 = PWM10, PWM10A, PWM10B outputs PWM inverse signal.

Bit 0 **PW1PO:** PWM one pulse output function control bit.
0 = Disable PWM one pulse output function.
1 = Enable PWM one pulse output function.

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1YH	-	-	-	-	-	-	PW1Y9	PW1Y8
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1YL	PW1Y7	PW1Y6	PW1Y5	PW1Y4	PW1Y3	PW1Y2	PW1Y1	PW1Y0
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

PW1Y [9:0]: PWM cycle control buffer.
The equation of PW1Y initial value is as following.

PW1Y initial value = PWM cycle time * PWM clock rate

0A7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1DH	-	-	-	-	-	-	PW1D9	PW1D8
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

0A6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1DL	PW1D7	PW1D6	PW1D5	PW1D4	PW1D3	PW1D2	PW1D1	PW1D0
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

PW1D [9:0]: PWM duty control buffer.
The equation of PW1D initial value is as following.

PW1D initial value = PWM duty time * PWM clock rate

0A5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1BH	-	-	-	-	-	-	PW1B9	PW1B8
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1BL	PW1B7	PW1B6	PW1B5	PW1B4	PW1B3	PW1B2	PW1B1	PW1B0
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

PW1B [9:0]: PWM dead band control buffer.
The equation of PW1B initial value is as following.

PW1B initial value = (PWM duty time – PWM end dead-band time)* PWM clock rate

0A8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1A	PW1A7	PW1A6	PW1A5	PW1A4	PW1A3	PW1A2	PW1A1	PW1A0
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

PW1A [7:0]: PWM dead band control buffer.
The equation of PW1A initial value is as following.

PW1A initial value = PWM start dead-band time * PWM clock rate

8.3.9 PWM OPERATION EXPLAME

- **PWM COUNTER CONFIGURATION:**

```
; Reset PWM.
        CLR     PW1M            ; Clear PW1M register.

; Set PWM clock rate.
        MOV     A, #0nnn0000b
        B0MOV   PW1M, A

; Set PW1Y registers for PWM cycle time.
        MOV     A, #value1      ; PW1C is equal to PW1Y.
        B0MOV   PW1YL, A
        MOV     A, #value2
        B0MOV   PW1YH, A

; Clear PW1IRQ
        B0BCLR  FPW1IRQ

; Enable PWM and interrupt function.
        B0BSET  FPW1IEN         ; Enable PWM interrupt function.
        B0BSET  FPW1EN          ; Enable PWM counter.
```

- **PWM OUTPUT CONFIGURATION:**

```
; Reset PWM.
        CLR     PW1M            ; Clear PW1M register.

; Set PWM clock rate.
        MOV     A, #0nnn0000b
        B0MOV   PW1M, A

; Set PW1Y registers for PWM cycle time.
        MOV     A, #value1      ; PW1C is equal to PW1Y.
        B0MOV   PW1YL, A
        MOV     A, #value2
        B0MOV   PW1YH, A

; Set PW1D registers for PWM duty.
        MOV     A, #value3      ; PW1Y must be greater than PW1D.
        B0MOV   PW1DL, A
        MOV     A, #value4
        B0MOV   PW1DH, A

; Set PW1A and PW1B registers for dead-band control.
        MOV     A, #value5      ;Set PWM dead-band start/end width
        B0MOV   PW1BL, A
        MOV     A, #value6
        B0MOV   PW1BH, A
        MOV     A, #value7
        B0MOV   PW1A, A

; Set PWM output pin idle state.
        B0BCLR  P0.2            ; Set PWM10A share pin idle low state.
        B0BSET  P0M.2
        B0BCLR  P0.3            ; PWM11A share pin idle low state.
        B0BSET  P0M.3
```

```
        B0BCLR      P0.5        ; PWM10 share pin idle low state.
        B0BSET      P0M.5
        B0BCLR      P0.6        ; PWM11 share pin idle low state.
        B0BSET      P0M.6
        B0BCLR      P1.5        ; PWM10B share pin idle low state.
        B0BSET      P1M.5
        B0BCLR      P1.6        ; PWM11B share pin idle low state.
        B0BSET      P1M.6
; or
        B0BSET      P0.2        ; Set PWM10A share pin idle high state.
        B0BSET      P0M.2
        B0BSET      P0.3        ; PWM11A share pin idle high state.
        B0BSET      P0M.3
        B0BSET      P0.5        ; PWM10 share pin idle high state.
        B0BSET      P0M.5
        B0BSET      P0.6        ; PWM11 share pin idle high state.
        B0BSET      P0M.6
        B0BSET      P1.5        ; PWM10B share pin idle high state.
        B0BSET      P1M.5
        B0BSET      P1.6        ; PWM11B share pin idle high state.
        B0BSET      P1M.6

; Set PWM output channels.
        MOV         A, #value1  ; Set PWM10~PWM11B output channels.
        B0MOV       PWCH, A

; Set inverse PWM output function.
        B0BSET      PWNV11      ; PWM11, PWM11A, PWM11B inverse output control bit.
        B0BSET      PWNV10      ; PWM10, PWM10A, PWM10B inverse output control bit.

; Clear PW1IRQ
        B0BCLR      FPW1IRQ

; Enable PWM interrupt function.
        B0BSET      FPW1IEN     ; Enable PWM interrupt function.

; Enable global interrupt.
        B0BSET      FGIE

; Enable PWM.
        B0BSET      FPW1EN      ; Enable PWM timer.
```

- **PWM ONE PULSE FUNCTION CONFIGURATION:**

```
; Reset PWM.
        CLR     PW1M            ; Clear PW1M register.

; Set PWM clock rate.
        MOV     A, #0nnn0000b
        B0MOV   PW1M, A

; Set PW1Y registers for PWM cycle time.
        MOV     A, #value1      ; PW1C is equal to PW1Y.
        B0MOV   PW1YL, A
        MOV     A, #value2
        B0MOV   PW1YH, A

; Set PW1D registers for PWM duty.
        MOV     A, #value3      ; PW1Y must be greater than PW1D.
        B0MOV   PW1DL, A
        MOV     A, #value4
        B0MOV   PW1DH, A

; Set PW1A and PW1B registers for dead-band control.
        MOV     A, #value5      ;Set PWM dead-band front/back width
        B0MOV   PW1BL, A
        MOV     A, #value6
        B0MOV   PW1BH, A
        MOV     A, #value7
        B0MOV   PW1A, A

; Set PWM output pin idle state.
        B0BCLR  P0.2            ; Set PWM10A share pin idle low state.
        B0BSET  P0M.2
        B0BCLR  P0.3            ; PWM11A share pin idle low state.
        B0BSET  P0M.3
        B0BCLR  P0.5            ; PWM10 share pin idle low state.
        B0BSET  P0M.5
        B0BCLR  P0.6            ; PWM11 share pin idle low state.
        B0BSET  P0M.6
        B0BCLR  P1.5            ; PWM10B share pin idle low state.
        B0BSET  P1M.5
        B0BCLR  P1.6            ; PWM11B share pin idle low state.
        B0BSET  P1M.6
; or
        B0BSET  P0.2            ; Set PWM10A share pin idle high state.
        B0BSET  P0M.2
        B0BSET  P0.3            ; PWM11A share pin idle high state.
        B0BSET  P0M.3
        B0BSET  P0.5            ; PWM10 share pin idle high state.
        B0BSET  P0M.5
        B0BSET  P0.6            ; PWM11 share pin idle high state.
        B0BSET  P0M.6
        B0BSET  P1.5            ; PWM10B share pin idle high state.
        B0BSET  P1M.5
        B0BSET  P1.6            ; PWM11B share pin idle high state.
        B0BSET  P1M.6

; Set PWM output channels.
        MOV     A, #value8      ; Set PWM10~PWM11B output channels.
        B0MOV   PWCH, A
```

```
; Set inverse PWM output function.
        B0BSET      PWNV11          ; PWM11, PWM11A, PWM11B inverse output control bit.
        B0BSET      PWNV10          ; PWM10, PWM10A, PWM10B inverse output control bit.

; Enable PWM one pulse function.
        B0BSET      FPW1PO          ; Enable PWM one pulse function.

; Clear PW1IRQ
        B0BCLR      FPW1IRQ

; Enable PWM interrupt function.
        B0BSET      FPW1IEN         ; Enable PWM interrupt function.

; Enable global interrupt.
        B0BSET      FGIE

; Enable PWM.
        B0BSET      FPW1EN          ; Enable PWM.
```

- **COMPARATOR TRIGGER MODE CONFIGURATION:**

```
; Reset PWM.
        CLR         PW1M            ; Clear PW1M register.

; Set PWM clock rate.
        MOV         A, #0nnn0000b
        B0MOV       PW1M, A

; Set PW1Y registers for PWM cycle time.
        MOV         A, #value1      ; PW1C is equal to PW1Y.
        B0MOV       PW1YL, A
        MOV         A, #value2
        B0MOV       PW1YH, A

; Set PW1D registers for PWM duty.
        MOV         A, #value3      ; PW1Y must be greater than PW1D.
        B0MOV       PW1DL, A
        MOV         A, #value4
        B0MOV       PW1DH, A

; Set PW1A and PW1B registers for dead-band control.
        MOV         A, #value5      ;Set PWM dead-band front/back width
        B0MOV       PW1BL, A
        MOV         A, #value6
        B0MOV       PW1BH, A
        MOV         A, #value7
        B0MOV       PW1A, A

; Set PWM output pin idle state.
        B0BCLR      P0.2            ; Set PWM10A share pin idle low state.
        B0BSET      P0M.2
        B0BCLR      P0.3            ; PWM11A share pin idle low state.
        B0BSET      P0M.3
        B0BCLR      P0.5            ; PWM10 share pin idle low state.
        B0BSET      P0M.5
        B0BCLR      P0.6            ; PWM11 share pin idle low state.
        B0BSET      P0M.6
        B0BCLR      P1.5            ; PWM10B share pin idle low state.
        B0BSET      P1M.5
        B0BCLR      P1.6            ; PWM11B share pin idle low state.
```

```
                B0BSET      P1M.6
; or
                B0BSET      P0.2            ; Set PWM10A share pin idle high state.
                B0BSET      P0M.2
                B0BSET      P0.3            ; PWM11A share pin idle high state.
                B0BSET      P0M.3
                B0BSET      P0.5            ; PWM10 share pin idle high state.
                B0BSET      P0M.5
                B0BSET      P0.6            ; PWM11 share pin idle high state.
                B0BSET      P0M.6
                B0BSET      P1.5            ; PWM10B share pin idle high state.
                B0BSET      P1M.5
                B0BSET      P1.6            ; PWM11B share pin idle high state.
                B0BSET      P1M.6

; Set PWM output channels.
                MOV         A, #value8      ; Set PWM10~PWM11B output channels.
                B0MOV       PWCH, A



; Set inverse PWM output function.
                B0BSET      PWNV11          ; PWM11, PWM11A, PWM11B inverse output control bit.
                B0BSET      PWNV10          ; PWM10, PWM10A, PWM10B inverse output control bit.

; Reset Comparator 0.
                CLR         CM0M            ; Clear CM0M register.

; Set Comparator 0 output pin.
                B0BCLR      FCM0OEN         ; Disable comparator 0 output pin.
; or
                B0BSET      FCM0OEN         ; Enable comparator 0 output pin.

; Set Comparator 0 trigger edge.
                B0BCLR      FCM0G0          ; Rising edge.
; or
                B0BSET      FCM0G1          ; Falling edge.
; or
                B0BSET      FCM0G0          ; Rising and falling edge.
                B0BSET      FCM0G1

; Set Comparator 0 share pin configuration.
                MOV         A, #value9      ; Set P1CON[6:5] for isolate GPIO path.
                B0MOV       P1CON, A

; Set comparator input initial state.
                B0BCLR      P1M.6           ; Set CM0N share pin as input floating.

                B0BCLR      P1M.5           ; Set CM0P share pin as input floating (Comparator 0
                                            ; positive input voltage is from CM0P share pin (P1.5).

; Set Comparator 0 positive input voltage.
                B0BCLR      FCM0S0          ; Comparator 0 positive input voltage is from CM0P share
                B0BCLR      FCM0S1          ; pin (P1.5).
; or
                B0BSET      FCM0S0          ; Comparator 0 positive input voltage is from internal
                                            ; 1/4*VDD.
; or
                B0BSET      FCM0S1          ; Comparator 0 positive input voltage is from internal
                                            ; 2/4*VDD.
; or
                B0BSET      FCM0S1          ; Comparator 0 positive input voltage is from internal
```

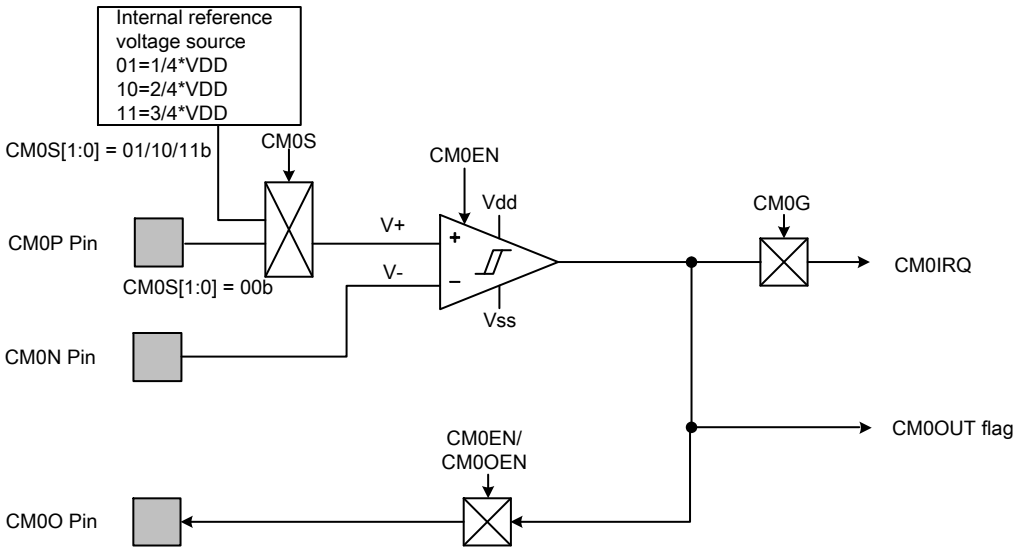
```
        B0BSET      FCM0S1          ; 3/4*VDD.

; Set comparator 0 trigger mode function.
        B0BSET      FCM0T           ; Enable comparator 0 trigger PW1 stop output function

; Clear CM0IRQ
        B0BCLR      FCM0IRQ


; Clear PW1IRQ
        B0BCLR      FPW1IRQ

; Enable Comparator 0 interrupt function.
        B0BSET      FCM0IEN         ; Enable Comparator 0 interrupt function.

; Enable PWM interrupt function.
        B0BSET      FPW1IEN         ; Enable PWM interrupt function.

; Enable global interrupt.
        B0BSET      FGIE

; Enable PWM.
        B0BSET      FPW1EN          ; Enable PWM.

; Enable Comparator 0 function.
        B0BSET      FCM0EN          ; Enable Comparator 0.
```

9 ANALOG COMPARAOTR

9.1 OVERVIEW

The analog comparator function includes analog comparator and internal reference voltage. 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. Comparator positive voltage is from internal 1/4*VDD, 2/4*VDD, 3/4*VDD, CM0P (CM0S[1:0]). There is a programmable direction function to decide comparator trigger edge for indicator function. The comparator has flag indicator, interrupt function and Green Mode weak-up function for different application. The main purposes of comparator are as following.

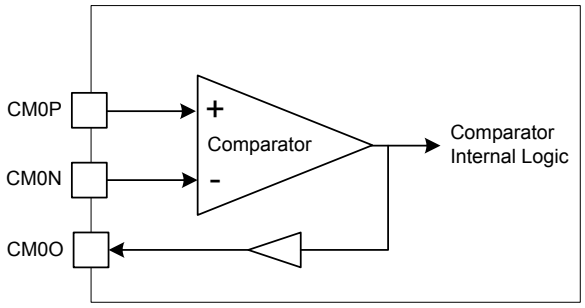
- **Normal comparator function:** General comparator mode compares the two tensions of positive input terminal and negative input terminal.
- **Interrupt function:** Comparator 0 supports interrupt function. When comparator 0 output edge direction equals to edge selection, the CM0IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **PWM stop output trigger source:** Comparator 0 can be PW1 stop output trigger source and is controlled by CM0T bit. When PW1EN = 1 and CM0T = 1, comparator 0 output status triggers PW1 stop output signal.
- **Green mode function:** Comparator 0 still actives in green mode, and wake-up function. CM0IRQ can be latched as trigger event occurrence until system wakes up and returns to normal or slow mode. After system wakes up, the comparator 0 interrupt service routine is executed by program.



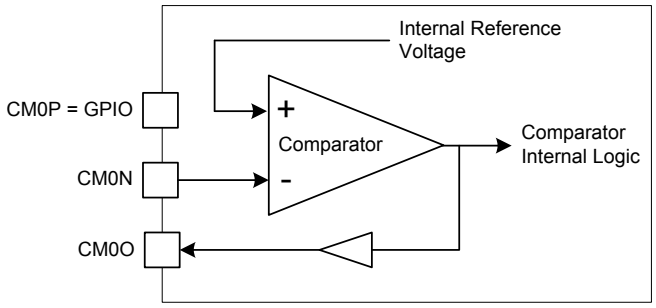
Comparator 0 Pin assignment:
CM0P: Comparator 0 positive input pin shared with P1.5. CM0P enables when CM0EN=1 and CM0S[1:0] = 00.
CM0N: Comparator 0 negative input pin shared with P1.6. CM0N enables when CM0EN=1.
CM0O: Comparator 0 output pin shared with P1.4. CM0O enables when CM0EN=1 and CM0OEN = 1.

9.2 NORMAL COMPARATOR MODE

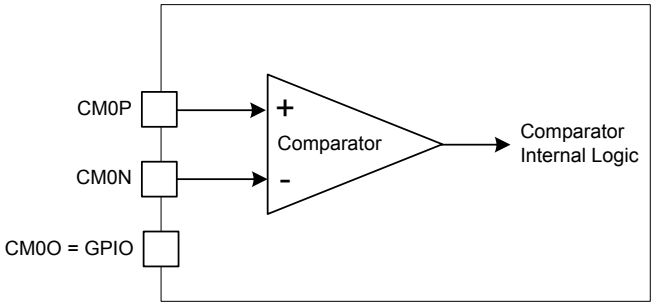
Comparator pins are shared with GPIO controlled by CM0EN bit. When CM0EN=1, CM0N pin is enabled connected to comparator negative terminal, and CM0P pin is controlled by CM0G[1:0] bits. CM0OEN controls comparator output connected to GPIO or not. When CM0OEN=1, comparator output terminal is connected to CM0O pin and isolate GPIO function. When CM0OEN=0, comparator output status can be read through CM0OUT flag and CM0O pin is GPIO mode.



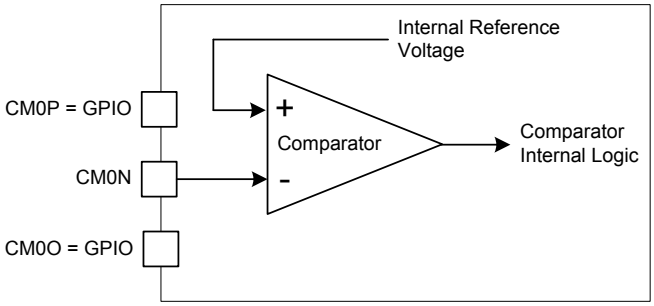
CM0EN = 1, CM1OEN = 1, CM0S[1:0] = 00b



CM0EN = 1, CM1OEN = 1, CM0S[1:0] = 01~11b



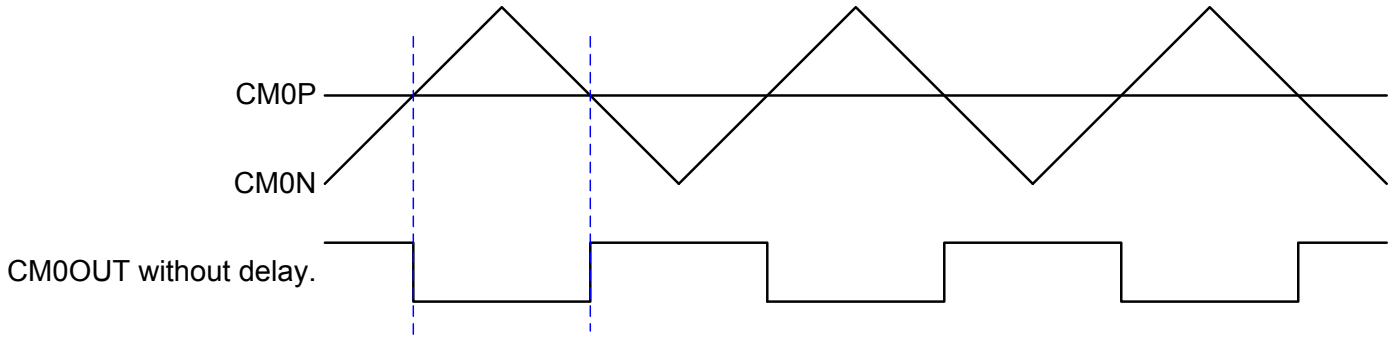
CM0EN = 1, CM1OEN = 0, CM0S[1:0] = 00b



CM0EN = 1, CM1OEN = 0, CM0S[1:0] = 01~11b

✷ ***Note:***
1. ***The comparator output pin signal is through internal buffer and not pure analog comparator output.***
2. ***The comparator negative input pin and positive input pin must be connected 0.1uF capacitor to ground and closer to comparator pins.***

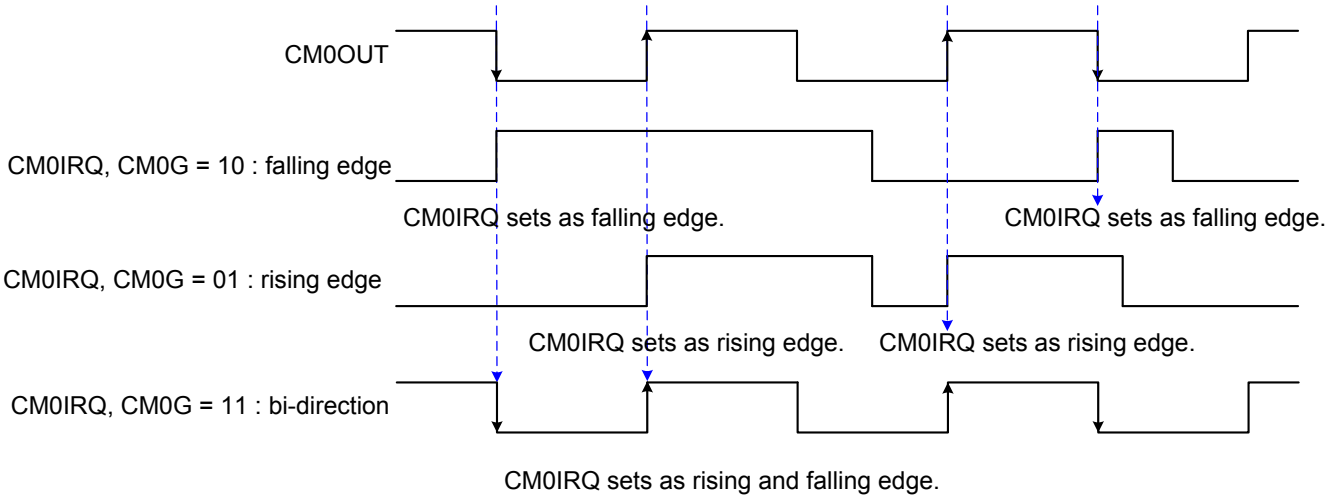
Comparator 0 compares positive terminal's voltage and negative terminal's voltage, and then output result to output pin. When V+ > V-, comparator outputs high status. When V+ < V-, comparator outputs low status.



The CM0OUT and CM0IRQ bits indicate the comparator result. The CM0OUT shows the comparator result immediately, but the CM0IRQ only indicates the event of the comparator result. The event condition is controlled by register and includes rising edge (CM0OUT changes from low to high) and falling edge (CM0OUT changes from high to low) controlled by CM0G[1:0] bits. When CM0G[1:0] = 00, the comparator 0 interrupt trigger is reserved. When CM0G[1:0] = 10, the comparator 0 interrupt trigger direction is falling edge. When CM0G[1:0] = 01, the comparator 0 interrupt trigger direction is rising edge. When CM0G[1:0] = 11, the comparator 0 interrupt trigger direction is rising and falling edge.

✷ ***Note: CM0OUT is comparator raw output without latch. It varies depend on the comparator process result. But the CM0IRQ is latch comparator output result. It must be cleared by program.***

Comparator supports interrupt function. The interrupt trigger condition can be selected through CM0G[1:0] bits including reserved, rising edge, falling edge and bi-direction. When comparator output edge event occurs and equal CM0G[1:0] condition, CM0IRQ flag is issued. If CM0IEN = 1, program counter points to interrupt vector to execute interrupt service routine.



***. CM0IRQ is cleared by program.**

9.3 COMPARATOR MODE REGISTER

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M	CM0EN	CM0T	CM0S1	CM0S0	CM0OEN	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 [1:0] **CM0G[1:0]:** Comparator output trigger direction control bit.
00 = Reserved
01 = Rising edge trigger. Comparator output status is from low to high as CM0P > CM0N.
10 = Falling edge trigger. Comparator output status is from high to low as CM0P < CM0N.
11 = Bi-direction trigger (rising and falling edge). Comparator output status is from low to high as CM0P > CM0N or is from high to low as CM0P < CM0N.

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

Bit 3 **CM0OEN:** Comparator 0 output pin control bit.
0 = Disable. CM0O is GPIO mode.
1 = Enable. CM0O is comparator output pin and isolate GPIO function.

Bit [5:4] **CM0S[1:0]:** Comparator 0 positive input voltage selection bits.

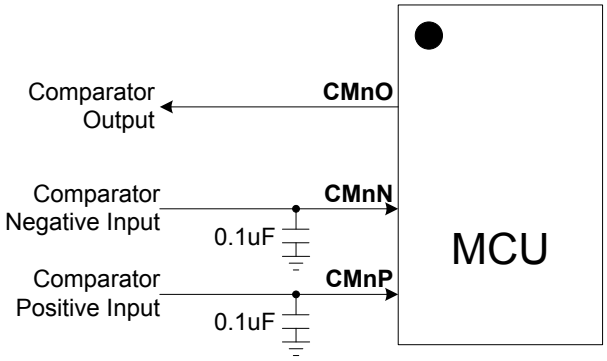
CM0S1	CM0S0	Positive input voltage
0	0	CM0P
0	1	1/4*VDD
1	0	2/4*VDD
1	1	3/4*VDD

Bit 6 **CM0T:** Comparator 0 output trigger PW1 stop control bit.
0 = Disable. Comparator 0 output trigger PW1 stop is not related.
1 = Enable. Comparator 0 outputs trigger PW1 stop control bit. When CM0IRQ = 1 => PW1 stop and clear PW1EN bit to 0.

Bit 7 **CM0EN:** Comparator 0 control bit.
0 = Disable. Comparator pins are GPIO mode.
1 = Enable. CM0N and CM0P pins are comparator mode. CM0O is controlled by CM0OEN bit.

9.4 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.



9.5 COMPARATOR 0 OPERATION EXPLAME

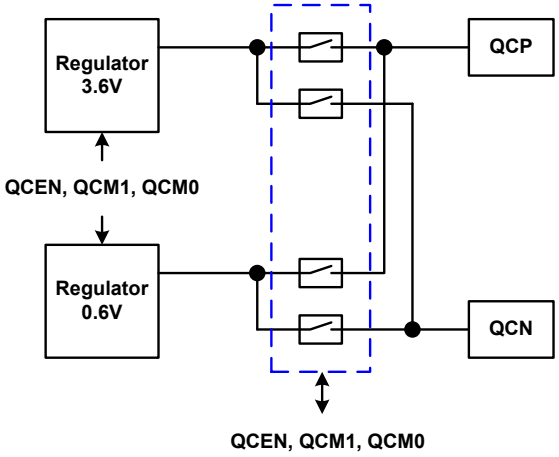
- **COMPARATOR 0 CONFIGURATION:**

```
; Reset Comparator 0.
        CLR     CM0M            ; Clear CM0M register.

; Set Comparator 0 output trigger PW1 stop control bit.
        B0BCLR  FCM0T           ; Normal comparator mode.
; or
        B0BSET  FCM0T           ; Comparator 0 outputs trigger PW1 stop control bit. When
                                  CM0IRQ = 1 => PW1 stop and clear PW1EN bit to 0.

; Set Comparator 0 output pin.
        B0BCLR  FCM0OEN         ; Disable comparator 0 output pin.
; or
        B0BSET  FCM0OEN         ; Enable comparator 0 output pin.

; Set Comparator 0 share pins are input floating.
        B0BCLR  P1M.6           ; Set CM0N share pin is input floating.
        B0BCLR  P1M.5           ; Set CM0P share pin is input floating (CM0S[1:0] = 00).

        MOV     A, #n00nnnnnb   ; Disable P1.2, P1.3 pull-up resistors
        MOV     P1UR, A

; Set Comparator 0 interrupt trigger edge.
        B0BSET  FCM0G1          ; Falling edge.
; or
        B0BSET  FCM0G0          ; Rising edge.
; or
        B0BSET  FCM0G1          ; Bi-direction (Falling edge and Rising edge).
        B0BSET  FCM0G0

; Set Comparator 0 positive input voltage source.
        B0BCLR  FCM0S0          ; Set Positive input voltage is from CM0P pin.
        B0BCLR  FCM0S1
; or
        B0BSET  FCM0S0          ; Set Positive input voltage is from internal 1/4*VDD.
; or
        B0BSET  FCM0S1          ; Set Positive input voltage is from internal 2/4*VDD.
; or
        B0BSET  FCM0S0          ; Set Positive input voltage is from internal 3/4*VDD.
        B0BSET  FCM0S1

; Clear CM0IRQ
        B0BCLR  FCM0IRQ

; Enable Comparator 0 and interrupt function.
        B0BSET  FCM0IEN         ; Enable Comparator 0 interrupt function.
        B0BSET  FCM0EN          ; Enable Comparator 0.
```

10 LOW CURRENT REGULATOR

10.1 OVERVIEW

The micro-controller builds in two low current regulators to output 0.6V and 3.6V for voltage indicator.



10.2 REGULATOR MODE CONTROL REGISTER

0B0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VREFH	EVHENB	-	QCEN	QCM1	QCM0	VHS2	VHS1	VHS0
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 5 **QCEN:** Regulator output control bit.
0 = Disable. Regulators turn off. P0.2 and P0.3 are GPIO pins.
1 = Enable. Regulators turn on and regulator voltage outputs. P0.2 and P0.3 are analog voltage.

Bit [4:3] **QCM[1:0]:** Regulators output voltage selection bits.

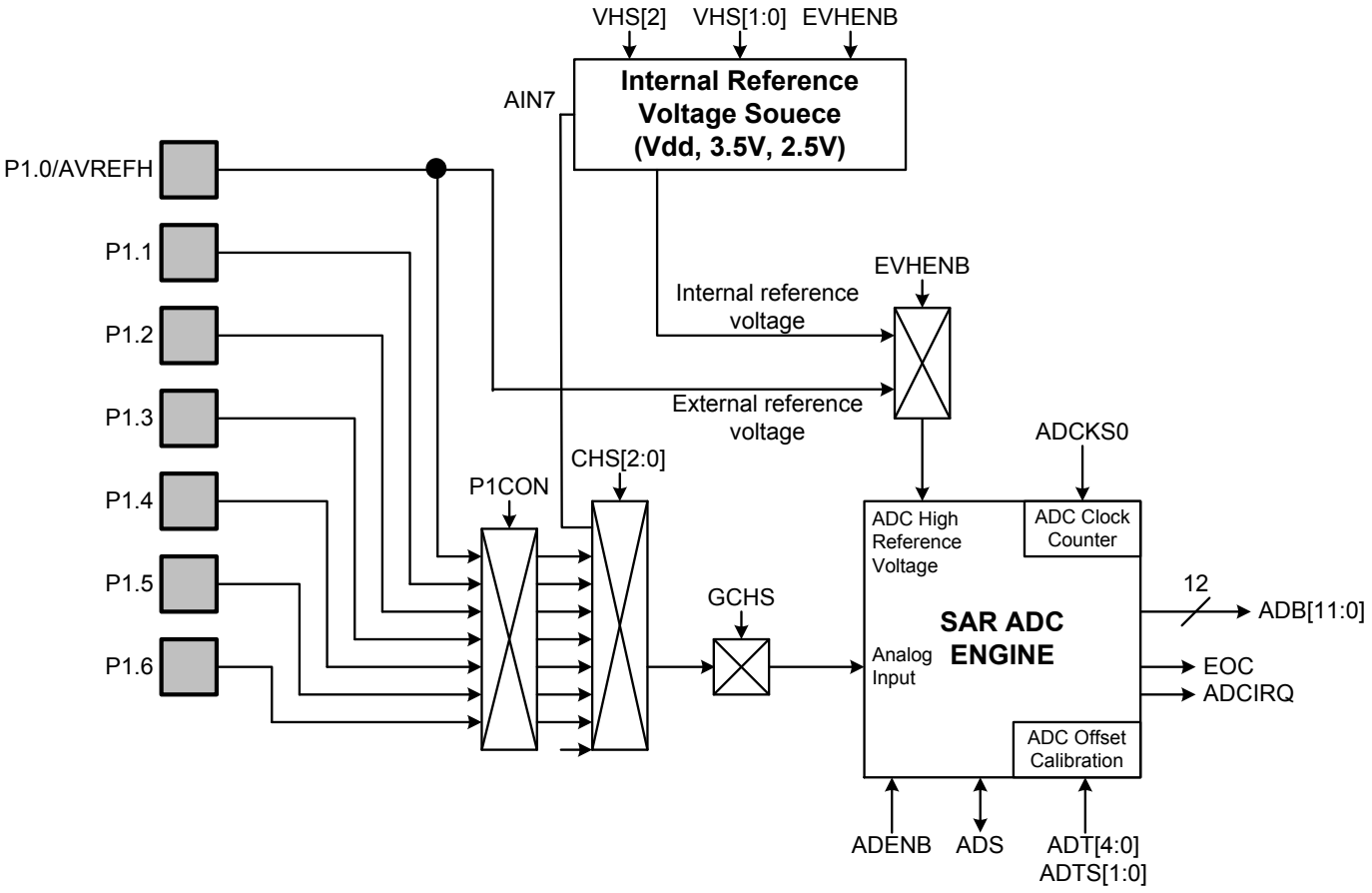
QCEN	QCM1	QCM0	0.6V Regulator	3.6V Regulator	QCP pin (P0.2)	QCN pin (P0.3)
0	-	-	Turn off	Turn off	GPIO	GPIO
1	0	0	Turn on	Turn off	0.6V	0.6V
1	0	1	Turn on	Turn on	3.6V	0.6V
1	1	0	Turn off	Turn on	3.6V	3.6V
1	1	1	Turn on	Turn off	0.6V	GND

✸ ***Note: When QCEN = 1, the regulator output share pin (P0.2/P0.3) transfer to regulator output and disable GPIO function.***

11 7+1 CHANNEL ANALOG TO DIGITAL CONVERTER (ADC)

11.1 OVERVIEW

The analog to digital converter (ADC) is SAR structure with 7+1-input sources and up to 4096-step resolution to transfer analog signal into 12-bits digital buffers. The ADC builds in 7+1-channel input source (AIN0~AIN7) to measure 8 different analog signal sources controlled by CHS[2:0] and GCHS bits. The ADC converting rate can be selected by ADCKS0 bit to decide ADC converting time. The ADC reference high voltage includes 4 sources controlled by VREFH register. Three internal power source including Vdd, 3.5V and 2.5V. The other one is external reference voltage input pin from P1.0 pin. The ADC builds in P1CON register to set pure analog input pin. It is necessary to set ADC input pin as input mode without pull-up resistor by program. After setup ADENB and ADS bits, the ADC starts to convert analog signal to digital data. When the conversion is complete, the ADC circuit will set EOC and ADCIRQ bits to “1” and the digital data outputs in ADB and ADR registers. If the ADCIEN = 1, the ADC interrupt request occurs and executes interrupt service routine when ADCIRQ = 1 after ADC converting. The interrupt procedure is system program counter points to interrupt vector (ORG 8) and executes interrupt service routine after ADC converting. Clear ADCIRQ by program is necessary in interrupt procedure. ADC can work in green mode. After ADC operating, the system would be waked up from green mode to last mode (normal mode/slow mode).



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	-	CHS2	CHS1	CHS0
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 **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 [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.

The AIN7 is internal 2.5V or 3.5V input channel. There is no any input pin from outside. In this time ADC reference voltage must be internal VDD or External voltage, not internal 2.5V or 3.5V. AIN7 can be a good battery detector for battery system. To select appropriate internal AVREFH level and compare value, a high performance and cheaper low battery detector is built in the system.

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	-	-	-	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	-	-	-	R/W	R	R	R	R
After reset	-	-	-	0	0	0	0	0

Bit 4 **ADCKS0:** ADC's clock rate select bit.
0 = Fhosc/4, 1 = Fhosc/2.

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[11:0]: 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[11:4]:** 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	-	-	-	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	-	-	-	R/W	R	R	R	R
After reset	-	-	-	0	0	0	0	0

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

The AIN input voltage v.s. ADB output data

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

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

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

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

11.4 ADC REFERENCE VOLTQAGE REGISTERS

ADC builds in four high reference voltage source controlled through VREFH register. There are one external voltage source and three internal voltage source (VDD, 3.5V, 2.5V). When EVHENB bit is “1”, ADC reference voltage is external voltage source from P1.0. It is necessary to input a voltage to be ADC high reference voltage and not below 2V. If EVHENB bit is “0”, ADC reference voltage is from internal voltage source selected by VHS[1:0] bits. If VHS[1:0] is “11” or ”10”, ADC reference voltage is VDD. If VHS[1:0] is “01”, ADC reference voltage is 3.5V. If VHS[1:0] is “00”, ADC reference voltage is 2.5V. The limitation of internal reference voltage application is VDD can’t below each of internal voltage level, or the level is equal to VDD.

0B0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VREFH	EVHENB	-	QCEN	QCM1	QCM0	VHS2	VHS1	VHS0
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 **EVHENB**: ADC reference voltage control bit.
0 = ADC reference is internal reference voltage source, and P1.0 is GPIO/AIN0.
1 = ADC reference is external reference voltage source from P1.0.

Bit [2:0] **VHS[2:0]**: AIN7 input voltage level and Internal reference voltage level selection.

EVHENB = 0

VHS2 = 0, ADC internal reference high voltage is depend on VHS[1:0]
VHS[1:0]: ADC internal reference high voltage selection bits

VHS1	VHS0	Internal reference voltage
0	0	2.5V
0	1	3.5V
1	0	VDD
1	1	VDD

VHS2 = 1, ADC internal reference high voltage is VDD.
VHS[1:0]: AIN7 input voltage selection bits.

VHS1	VHS0	AIN7 input voltage source
0	0	2.5V
0	1	3.5V
1	0	VDD
1	1	VDD

EVHENB = 1

VHS2 = 0, ADC reference is external reference voltage source from P1.0.
VHS[1:0]: Reserved.

VHS2 = 1, ADC reference is external reference voltage source from P1.0.
VHS[1:0]: AIN7 input voltage selection bits.

VHS1	VHS0	AIN7 input voltage source
0	0	2.5V
0	1	3.5V
1	0	VDD
1	1	VDD

✷ ***Note: If AIN7 channel is selected as internal 2.5V or 3.5V input channel. There is no any input pin from outside. In this time ADC reference voltage must be internal VDD or External voltage, not internal 2.5V or 3.5V.***

11.5 ADC OPERATION DESCRIPTION AND NOTIC

11.5.1 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/3.5V/2.5V and external reference voltage source from P1.0/AVREFH pin controlled by EVHENB bit. If EVHENB=0, ADC reference voltage is from internal voltage source. If EVHENB=1, ADC reference voltage is from external voltage source (P1.0/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/3.5V/2.5V. (EVHENB=0)**
- **ADC External High Reference Voltage = 2V~Vdd. (EVHENB=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**

11.5.2 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 Fhosc and includes Fhosc/1, Fhosc/2, Fhosc/8 and Fhosc/16 controlled by ADCKS[1:0] bits.
The ADC converting time affects ADC performance. If input high rate analog signal, it is necessary to select a high ADC converting rate. If the ADC converting time is slower than analog signal variation rate, the ADC result would be error. So to select a correct ADC clock rate and ADC resolution to decide a right ADC converting rate is very important.

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

ADCKS0	ADC Clock Rate	Fhosc=16MHz	
		ADC Converting time	ADC Converting Rate
0	Fhosc/4	1/(16MHz/4/4)*16 = 16 us	62.5KHz
1	Fhosc/2	1/(16MHz/2/4)*16 = 8 us	125KHz

11.5.3 ADC PIN CONFIGURATION

ADC input channels are shared with Port1. ADC channel selection is through CHS[2:0] bit. CHS[2:0] value points to the ADC input channel directly. CHS[2:0]=000 selects AIN0. CHS[2:0]=001 selects AIN1…… Only one pin of port1 can be configured as ADC input in the same time. The pins of Port1 configured as ADC input channel must be set input mode, disable internal pull-up and enable P1CON first by program. After selecting ADC input channel through CHS[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.**
- **P1CON bits of ADC input channel must be set.**

The P1.0/AIN0 can be ADC external high reference voltage input pin when EVHENB=1. In the condition, P1.0 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 port1 will encounter above current leakage situation. P1CON is Port1 configuration register. Write "1" into P1CON [6:0] will configure related port 1 pin as pure analog input pin to avoid current leakage.

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1CON	-	P1CON6	P1CON5	P1CON4	P1CON3	P1CON2	P1CON1	P1CON0
Read/Write	-	W	W	W	W	W	W	W
After reset	-	0	0	0	0	0	0	0

Bit[7:0] **P1CON[6:0]:** P1.n configuration control bits.
0 = P1.n can be an analog input (ADC input) or digital I/O pins.
1 = P1.n is pure analog input, can't be a digital I/O pin.

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

11.6 ADC OPERATION EXAMLPE

- **ADC CONFIGURATION:**

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

; Set ADC clock rate and ADC resolution.
        MOV     A, #000n0000b   ; n: ADCKS0 for ADC clock rate.
        B0MOV   ADR, A

; Set ADC high reference voltage source.
        B0BSET  FEVHENB         ; External reference voltage.
or
        MOV     A, #000000nnb   ; Internal Vdd.
        B0MOV   VREFH, A        ; “nn” select internal reference voltage level.
                                ; 11 = 10 = VDD, 01 = 3.5V, 00 = 2.5V.

; Set AIN7 input voltage source (FEVHENB = 0).
        MOV     A, #00000mnnb   ; m: Internal reference voltage is VDD.
        B0MOV   VREFH, A        ; “nn” select AIN7 input voltage source.
                                ; 11 = 10 = VDD, 01 = 3.5V, 00 = 2.5V.

; Set ADC input channel configuration.
        MOV     A, #value1      ; Set P1CON for ADC input channel.
        B0MOV   P1CON, A
        MOV     A, #value2      ; Set ADC input channel as input mode.
        B0MOV   P1M, A
        MOV     A, #value3      ; Disable ADC input channel’s internal pull-up resistor.
        B0MOV   P1UR, 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 CHS[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:***

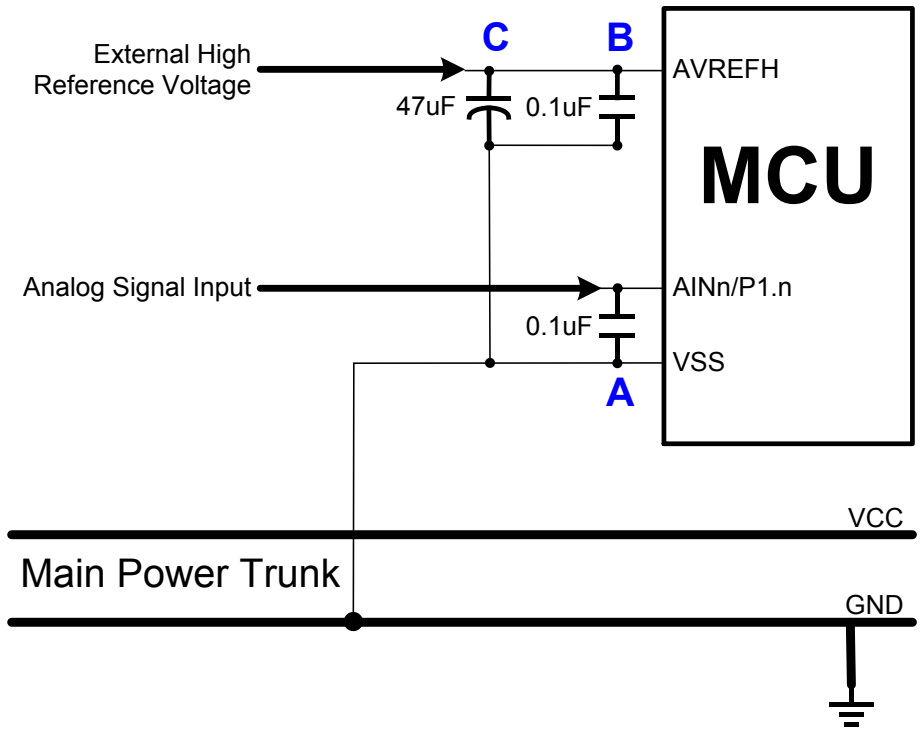
1. ***When ADENB is enabled, the system must be delay 100us to be the ADC warm-up time by program, and then set ADS to do ADC converting. The 100us delay time is necessary after ADENB setting (not ADS setting), or the ADC converting result would be error. Normally, the ADENB is set one time when the 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 it by program.***

11.7 ADC APPLICATION CIRCUIT



The analog signal is inputted to ADC input pin “AINn/P1.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 (P1.0). 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 INSTRUCTION TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
	MOV	A,M	A ← M	-	-	√	1
M	MOV	M,A	M ← A	-	-	-	1
O	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
V	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
E	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I, "M" only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z…)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
A	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1+N
R	ADD	A,M	A ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
I	ADD	M,A	M ← A + M, if occur carry, then C=1, else C=0	√	√	√	1+N
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
H	ADD	A,I	A ← A + I, if occur carry, then C=1, else C=0	√	√	√	1
M	SBC	A,M	A ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
E	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
T	SUB	A,M	A ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
I	SUB	M,A	M ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1+N
C	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
	AND	A,M	A ← A and M	-	-	√	1
L	AND	M,A	M ← A and M	-	-	√	1+N
O	AND	A,I	A ← A and I	-	-	√	1
G	OR	A,M	A ← A or M	-	-	√	1
I	OR	M,A	M ← A or M	-	-	√	1+N
C	OR	A,I	A ← A or I	-	-	√	1
	XOR	A,M	A ← A xor M	-	-	√	1
	XOR	M,A	M ← A xor M	-	-	√	1+N
	XOR	A,I	A ← A xor I	-	-	√	1
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
R	RRC	M	A ← RRC M	√	-	-	1
O	RRCM	M	M ← RRC M	√	-	-	1+N
C	RLC	M	A ← RLC M	√	-	-	1
E	RLCM	M	M ← RLC M	√	-	-	1+N
S	CLR	M	M ← 0	-	-	-	1
S	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
B	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
R	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1+ S
A	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
N	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1+ S
C	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1+N+S
H	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
M	RET		PC ← Stack	-	-	-	2
I	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
S	PUSH		To push ACC and PFLAG (except NT0, NPD bit) into buffers.	-	-	-	1
C	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".

13 ELECTRICAL CHARACTERISTIC

13.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2761P, SN8P2761S, SN8P2761X ..... 0°C ~ + 70°C

Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

13.2 ELECTRICAL CHARACTERISTIC

- **DC CHARACTERISTIC**

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 16MHz, Fcpu=1MHz, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, *25℃, Fcpu = 1MHz*		2.2	-	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	ViL	All input ports		Vss	-	0.3Vdd	V
Input High Voltage	ViH	All input ports		0.7Vdd	-	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/pull-down resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	150	
I/O output source current sink current	IoH	Vop = Vdd – 0.5V		8	15	-	mA
	IoL	Vop = Vss + 0.5V		8	15	-	
1/2*Bias Voltage	Vbias	P5.0~P5.3 pull-up/pull-down resistors enable.		2.2	2.5	2.8	V
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current	Idd1	Run Mode	Vdd= 3V, Fcpu = 16MHz	-	4	-	mA
			Vdd= 5V, Fcpu = 16MHz	-	7	-	mA
			Vdd= 3V, Fcpu = 4MHz	-	2	-	mA
			Vdd= 5V, Fcpu = 4MHz	-	4.1	-	mA
			Vdd= 3V, Fcpu = 1MHz	-	1.5	-	mA
			Vdd= 5V, Fcpu = 1MHz	-	3.5	-	mA
	Idd2	Run Mode	Vdd= 3V, Fcpu = 1MHz	-	1	-	mA
			Vdd= 5V, Fcpu = 1MHz	-	2	-	mA
	Idd3	Slow Mode (Internal low RC, high clock stop)	Vdd= 3V, ILRC=16KHz	-	3	-	uA
			Vdd= 5V, ILRC=32KHz	-	10	-	uA
	Idd4	Sleep Mode	Vdd= 5V/3V	-	1	2	uA
	Idd5	Green Mode (No loading, Watchdog Disable)	Vdd= 3V, IHRC=16MHz	-	0.45	-	mA
			Vdd= 5V, IHRC=16MHz	-	0.6	-	mA
			Vdd= 3V, ILRC=16KHz	-	2	-	uA
			Vdd= 5V, ILRC=32KHz	-	5	-	uA
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	*25℃*, Vdd=2.2V~ 5.5V Fcpu=Fhosc/1~Fhosc/128	15.68	16	16.32	MHz
			-40℃~85℃,Vdd=2.4V~ 5.5V Fcpu=Fhosc/1~Fhosc/128	15.2	16	16.8	MHz
LVD Voltage (Sleep mode)	LVD20	25℃. Trimmed. (Sleep mode)		2	2.1	2.2	V
		-40℃~+85℃. Trimmed. (Sleep mode)		1.9	2.1	2.3	
	LVD24	25℃. Trimmed. (Sleep mode)		2.3	2.4	2.5	
		-40℃~+85℃. Trimmed. (Sleep mode)		2.2	2.4	2.6	
	LVD36	25℃. Trimmed. (Sleep mode)		3.5	3.6	3.7	
		-40℃~+85℃. Trimmed. (Sleep mode)		3.4	3.6	3.8	

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

ADC CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 16MHz, 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, ADC Internal reference Voltage = 4V.	-	0.5	-	mA
		Vdd=3.0V, ADC Internal reference Voltage = 4V.	-	0.4	-	mA
ADC Clock Frequency	F_{ADCLK}	VDD=5.0V	-	-	4M	Hz
		VDD=3.0V	-	-	4M	Hz
ADC Conversion Cycle Time	F_{ADCYL}	VDD=2.2V~5.5V	64	-	-	$1/F_{ADCLK}$
ADC Sampling Rate (Set FADS=1 Frequency)	F_{ADCR}	VDD=5.0V	-	-	62.5	K/sec
		VDD=3.0V	-	-	62.5	K/sec
Differential Nonlinearity	DNL	VDD=5.0V , AVREFH = VDD = 5V, F_{ADCR} = 62.5K	±1	-	-	LSB
Integral Nonlinearity	INL	VDD=5.0V , AVREFH = VDD = 5V, F_{ADCR} = 62.5K	±2	-	-	LSB
No Missing Code	NMC	VDD=5.0V , AVREFH = VDD = 5V, F_{ADCR} = 62.5K	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.

REGULATOR CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 16MHz, Fcpu=1MHz, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
*Supply Voltage	Iop	Regulator active	4.5	-	5.5	V
		Iout=0, Vdd= 5V	-	150	-	uA
*Quiescent Current	Iq		-	200	-	uA
*Regulator Output Voltage	Vo1	*-40℃~85℃*,Vdd = 5V	0.54	0.6	0.66	V
	Vo2	*-40℃~85℃*,Vdd = 5V	3.24	3.6	3.96	V
Leakage Current	Leak	VDD = 5V.	-	1	-	uA

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

COMPARATOR 0 CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 16MHz, Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Power Supply Range	Vcmp		2.0	-	5.5	V
Supply Current	Icm	Vdd=5V. Vp > Vn. (Internal reference voltage)	-	70	-	uA
Quiescent Current	Iq	Iout=0	-	1	-	uA
Input Offset Voltage	Vos	Vcm=Vss	-15	-	+15	mV
Response Time	Trs	Positive input voltage = 1/2*Vdd. Negative input voltage transitions from Vss to Vdd.	-	-	200	ns
Output Slew Rate	Tosr	Comparator output voltage transitions from Vdd to Vss.			100	Ns
		Comparator output voltage transitions from Vss to Vdd.			100	ns
Common Mode Input Voltage Range	Vcmr	Vdd=5.0V	Vss+0.5		Vdd-0.5	V

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

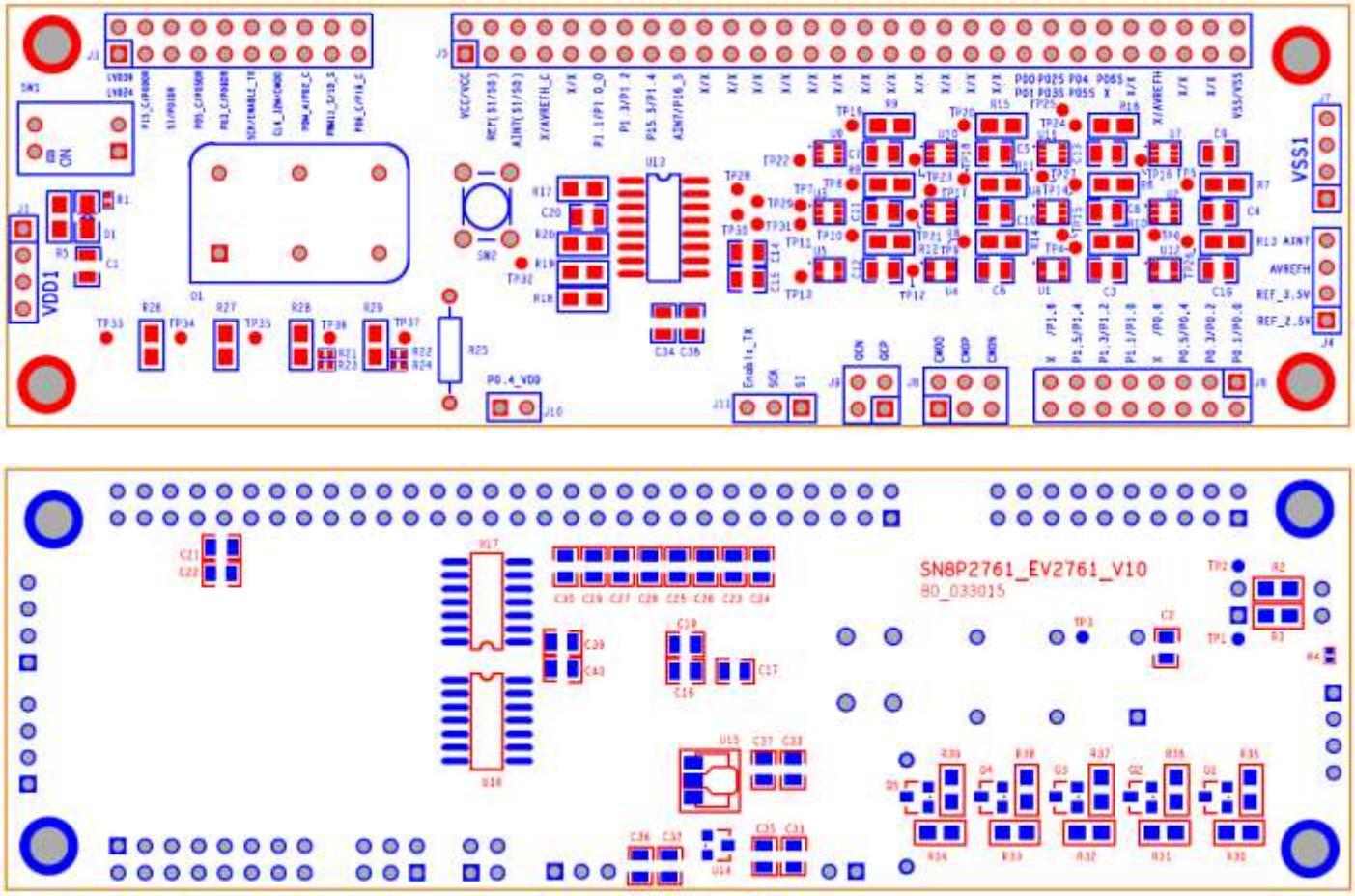
14 DEVELOPMENT TOOL

SONIX provides SN8P2761 MCU which includes Pull-down resistors, PWM, ADC, Comparator 0 and QC analog functions. These functions aren't built in SN8ICE2K Plus 2. To emulate analog functions must be through SN8P2761 real chip. The real chip provides an EV-KIT to achieve the analog functions emulations. For SN8P2761 ICE emulation, the EV-Kit includes Pull-down resistors/Comparator 0/QC/ADC or VDD Reference Voltage and switch circuits. 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: EV2761 KIT REV: V1.0.**
- **IDE: SONiX IDE M2IDE_V142 and later version.**
- **Writer: MPIII writer.**
- **Writer transition board: SN8P2761**

14.1 SN8P2761 EV-KIT

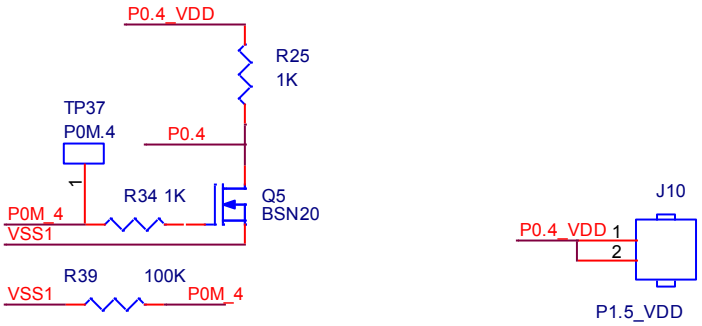
SN8P2761 EV-kit Rev. 1.0 PCB Outline:



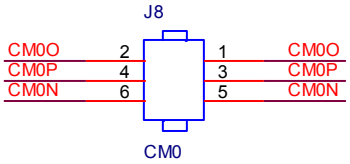
- **J5**: Connect to SN8ICE2K Plus II CON1 (includes GPIO, EV-KIT control signal, and the others).
- **J3**: Connect to SN8ICE2K Plus II JP3 (EV-KIT communication bus with ICE, control signal, and the others).
- **D1**: EV-KIT power indicator.
- **SW1**: LVD24V / LVD36V control switch. To emulate LVD2.4V 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

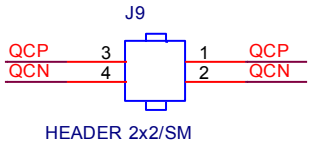
- **SW2**: Reset EV-chip function control switch.
- **J7**: EV-kit ground connector.
- **J1**: EV-kit power connector.
- **R25**: P0.4 connection's external pull-up resistor (Open-drain pull-up resistor), user define.
  J10: P0.4 external pull-up resistor and connect to power pin (P0.4_VDD).



- **J6**: Chip's I/O pin connector (But do not support CM0O/QC/Reset function.
- **R26**: Emulation P0.0 internal Pull-down resistor.
- **R27**: Emulation P0.1 internal Pull-down resistor.
- **R28**: Emulation P0.5 internal Pull-down resistor.
- **R29**: Emulation P0.6 internal Pull-down resistor.
- **U13**: EV-chip (SN8P2761) is a 16-pin SOP package. The EV-chip communication bus with ICE (Enable_TX, SCK, SI pin), support QC IP analog function, support comparator 0 IP analog function (include CM0P, 1/4*VDD, 2/4*VDD, 3/4*VDD).
- **If user source code enables PW1 function in ICE emulation and sets IDE breakpoint after PW1 function enable (PWM/plus generator), the PWM output will be finished (Fhosc still work). The PW1 pulse generator output will be back to idle status and PWM still output.**
- **J8**: Comparator 0 analog positive/negative/output pin (CM0P/CM0N/CM0O). But do not support P0.2/P0.3/P4.5 GPIO function.

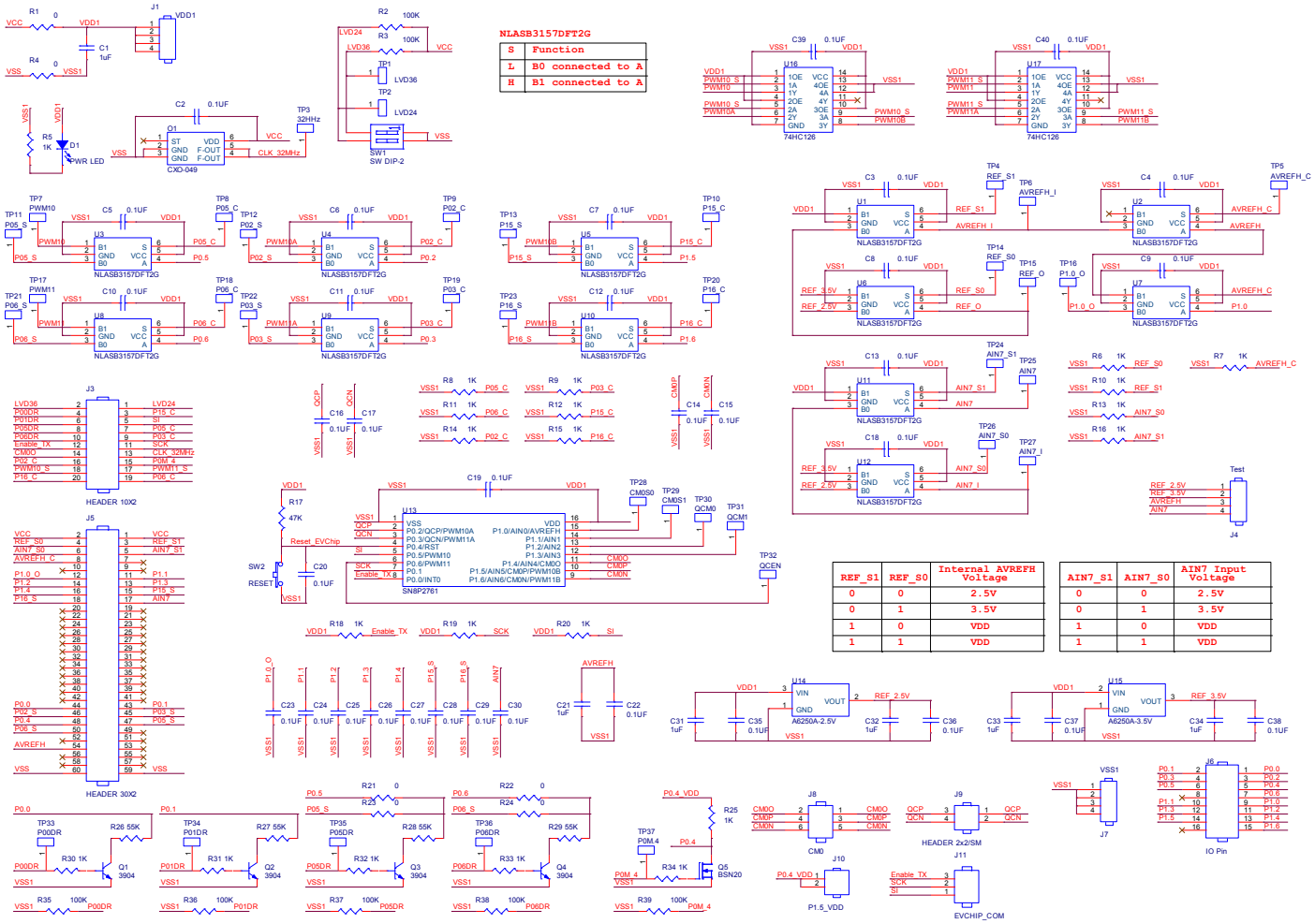


- **J12**: QC function output pin (QCP/QCN). But do not support P0.2/P0.3/PWM10A/PWM11A GPIO function.



- **C23~C29**: User **c**onnect 0.1uF capacitors to AIN0~AIN6 input which are ADC's (channel 0~6) bypass capacitors by self.
- **C30**: User **c**onnects 0.1uF capacitor to AIN7 input which are ADC's (channel 7) bypass capacitors by self.
- **C14:** Connect 0.1uF capacitors to SN8P2761 EV-chip's CM0P input pin which is Comparator 0's bypass capacitors.
- **C15:** Connect 0.1uF capacitors to SN8P2761 EV-chip's CM0N input pin which is Comparator 0's bypass capacitors.

EV2761 KIT schematic:

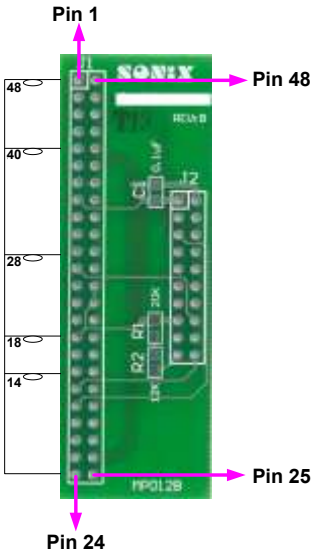


14.2 ICE AND EV-KIT APPLICATION NOTIC

1. SN8ICE2K Plus 2 power switch must be turned off before you connect the EV2761 KIT to SN8ICE2K Plus 2.
2. Connect EV-KIT's J3/J5 to ICE's JP3/CON1.
3. SN8ICE2K Plus 2's AVREFH/VDD jumper pin must be removed.
4. Turn on SN8ICE2K Plus power switch after user had finished step 1~3.
5. User observes EV-KIT's power LED (D1) is light after turn on SN8ICE2K Plus power switch. If LED (D1) is un-light, that means, user contact to SONIX's agent right now.
6. **It is necessary to connect 16MHz crystal in ICE for IHRC_16M mode emulation. SN8ICE2K Plus II doesn't support over 8-mips instruction cycle, but real chip does.**
7. **SN8P2761 ADC clock source is from Fhosc (ICE support is Fcpu). So ICE emulation ADC converted time will be incorrect**
8. **SN8P2761 ADC clock source is from Fhosc (ICE support is Fcpu). So ICE emulation ADC green mode wakeup function will be unsupported (Fcpu stop).**
9. **ICE doesn't support P1 wake up function emulation under green mode and power down mode**
10. **ICE doesn't support P1.4 emulation CM0OEN function.**
11. **When CMP0 function enable. The CM0P/CM0N (JP8) will be external analog signal input pin. The CM0O (J8) will be CMP0's output result.**
12. **When user uses Comparator 0's CM0P/CM0N/CM0O analog function, user must be connecting to J8.**
13. **J6 support P1.4/P1.5/P1.6 GPIO/PWM/ADC function, but doesn't include Comparator 0 function.**
14. **J6 support P0.2/P0.3 GPIO/PWM function, but doesn't include Regulator function.**
15. **J9 support Regulator function, but doesn't include GPIO/PWM function.**

15 OTP PROGRAMMING PIN

15.1 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

15.2 PROGRAMMING PIN MAPPING:

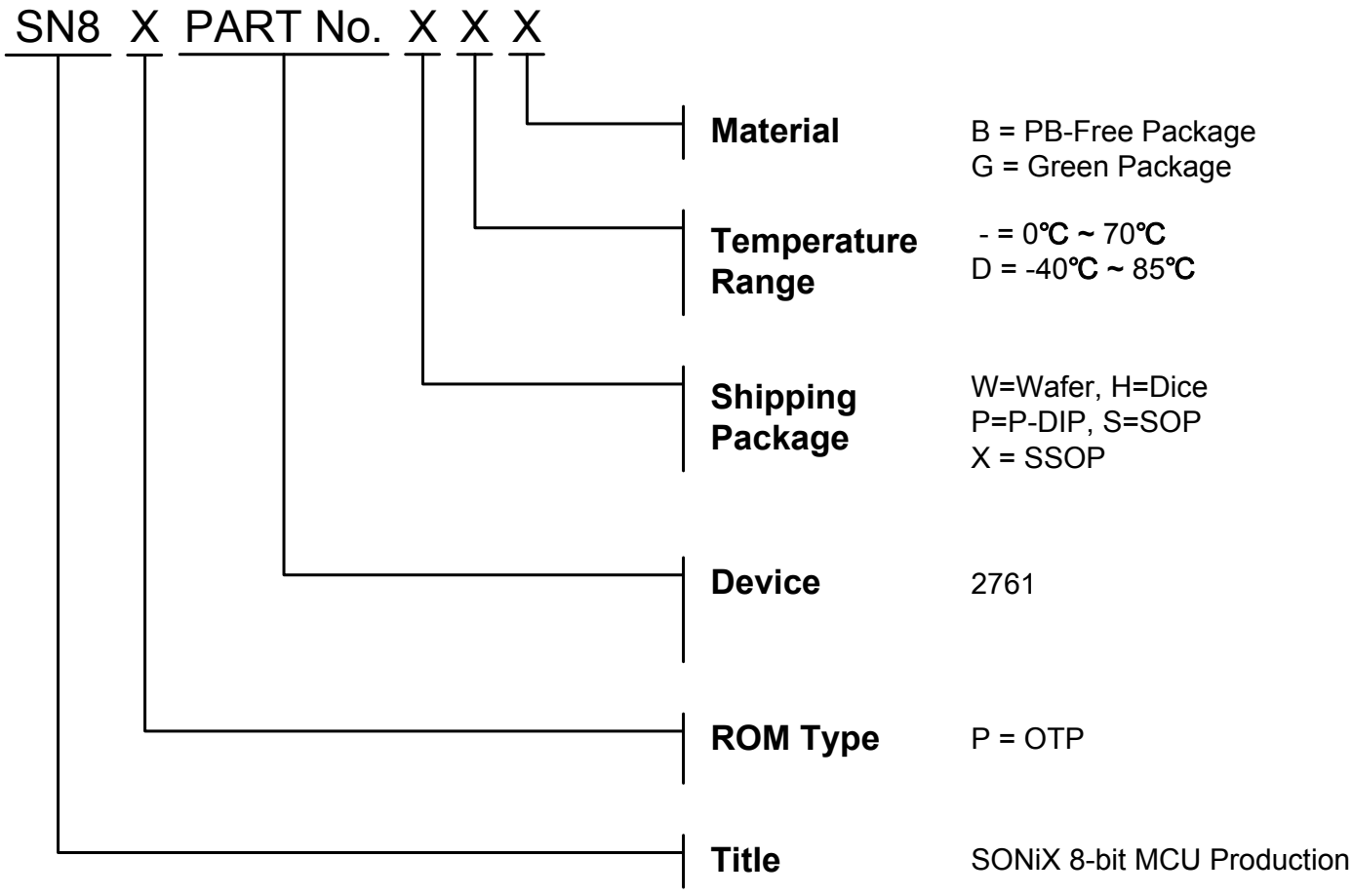
Programming Pin Information of SN8P2761 Series							
Chip Name		SN8P2761P/S(DIP/SOP)			SN8P2761X(SSOP)		
Writer Connector		IC and JP3 48-pin text tool Pin Assignment					
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number
1	VDD	16	VDD	32	12	VDD	28
2	GND	1	VSS	17	13	VSS	29
3	CLK	14	P1.1	30	10	P1.1	26
4	CE	-	-	-	-	-	-
5	PGM	13	P1.2	29	9	P1.2	25
6	OE	12	P1.3	28	8	P1.3	8
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	4	RST	20	16	RST	32
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	11	P1.4	27	7	P1.4	23

16 Marking Definition

16.1 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.

16.2 MARKING INDETIFICATION SYSTEM



16.3 MARKING EXAMPLE

- **Wafer, Dice:**

Name	ROM Type	Device	Package	Temperature	Material
S8P2761W	OTP	2761	Wafer	0℃~70℃	-
SN8P2761H	OTP	2761	Dice	0℃~70℃	-

- **Green Package:**

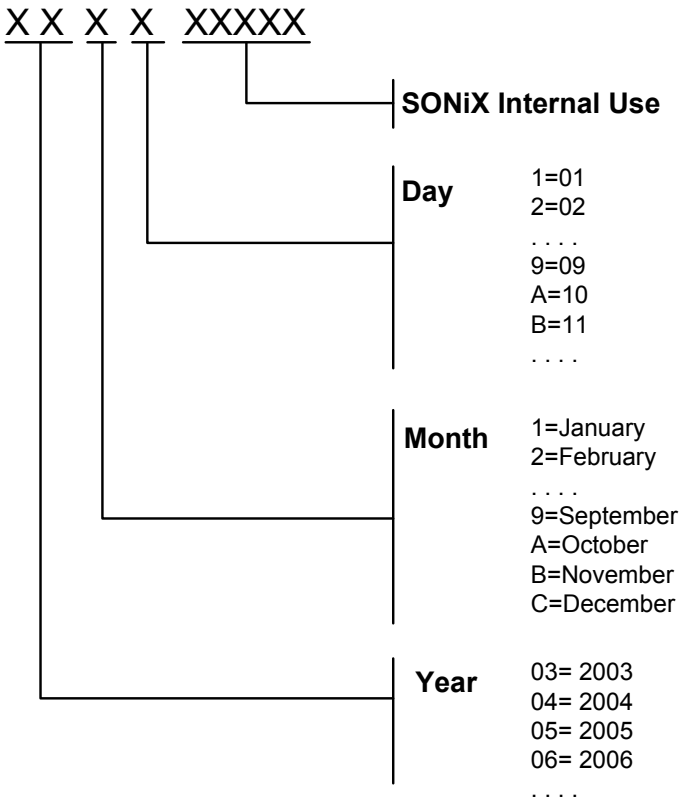
Name	ROM Type	Device	Package	Temperature	Material
SN8P2761PG	OTP	2761	P-DIP	0℃~70℃	Green Package
SN8P2761SG	OTP	2761	SOP	0℃~70℃	Green Package

SN8P2761XG	OTP	2761	SSOP	0℃~70℃	Green Package
SN8P2761PDG	OTP	2761	P-DIP	-40℃~85℃	Green Package
SN8P2761SDG	OTP	2761	SOP	-40℃~85℃	Green Package
SN8P2761XDG	OTP	2761	SSOP	-40℃~85℃	Green Package

- **PB-Free Package:**

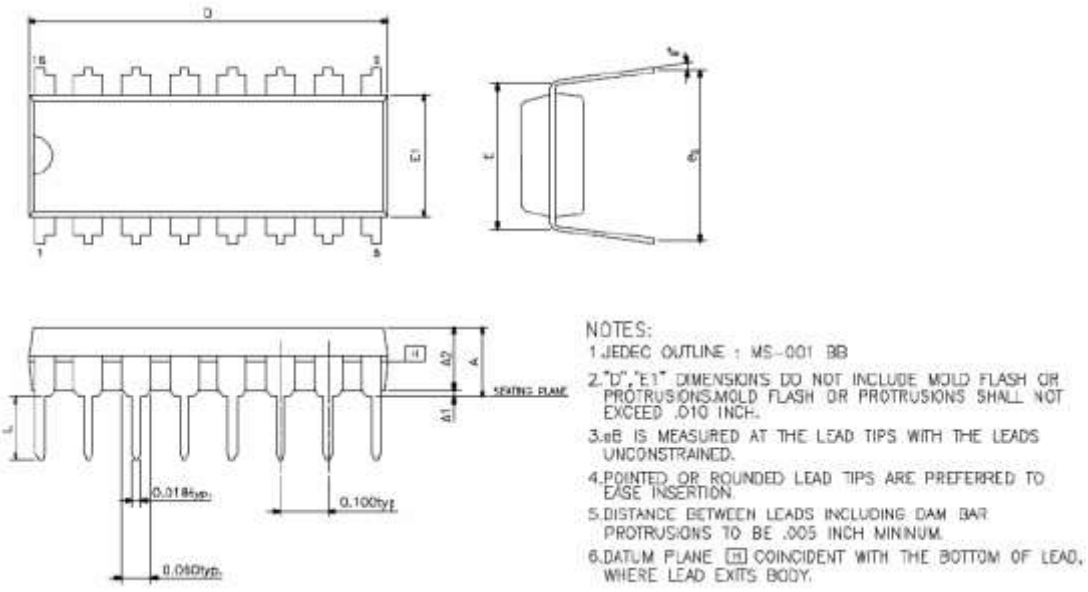
Name	ROM Type	Device	Package	Temperature	Material
SN8P2761PB	OTP	2761	P-DIP	0℃~70℃	PB-Free Package
SN8P2761SB	OTP	2761	SOP	0℃~70℃	PB-Free Package
SN8P2761XB	OTP	2761	SSOP	0℃~70℃	PB-Free Package
SN8P2761PDB	OTP	2761	P-DIP	-40℃~85℃	PB-Free Package
SN8P2761SDB	OTP	2761	SOP	-40℃~85℃	PB-Free Package
SN8P2761XDB	OTP	2761	SOP	-40℃~85℃	PB-Free Package

16.4 DATECODE SYSTEM



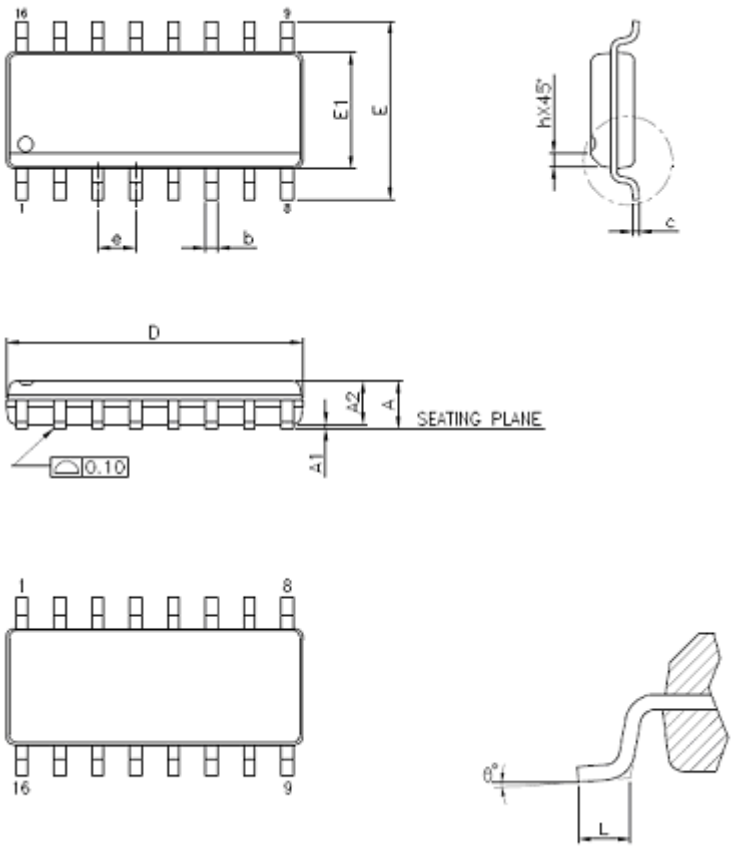
17 PACKAGE INFORMATION

17.1 P-DIP 16 PIN



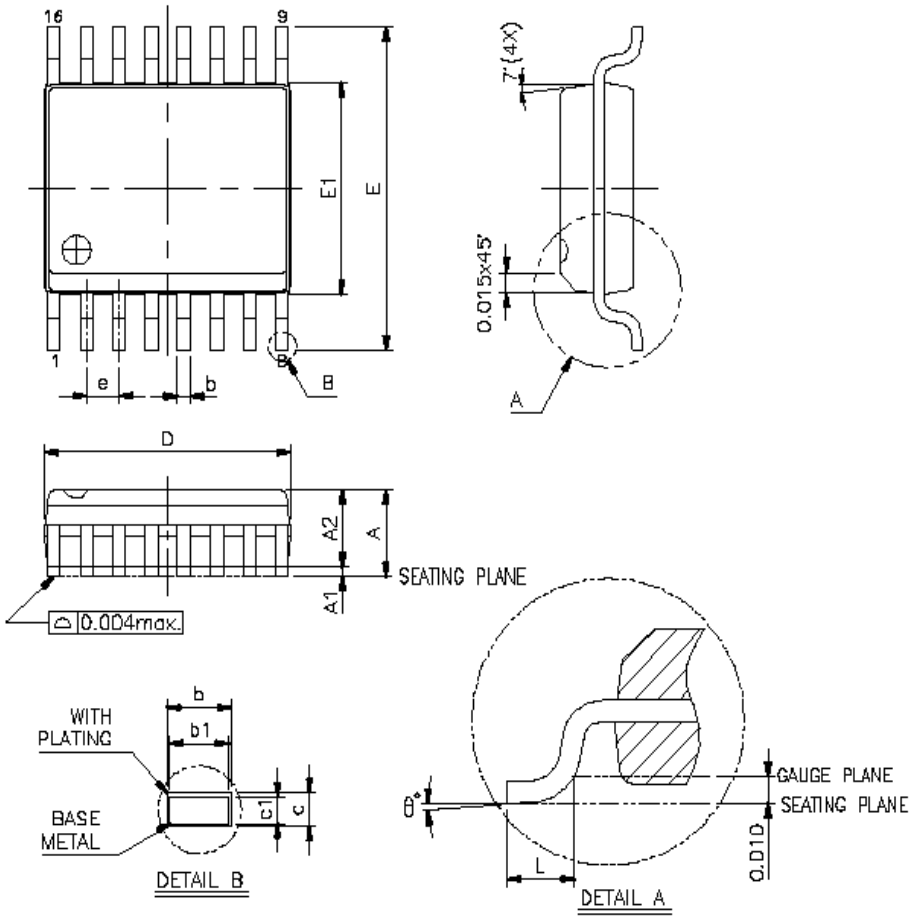
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.735	0.775	0.775	18.669	19.177	19.685
E		0.300BSC			7.620BSC	
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

17.2 SOP 16 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.069	-	-	1.75
A1	0.004	-	0.010	0.10	-	0.25
A2	0.049	-		1.25	-	-
b	0.012	-	0.020	0.31	-	0.51
c	0.004	-	0.010	0.10	-	0.25
D	9.90BSC			9.90BSC		
E	6.00BSC			6.00BSC		
E1	3.90BSC			3.90BSC		
e	1.27BSC			1.27BSC		
h	0.016	-	0.050	0.40	-	1.27
L	0.010	-	0.020	0.25	-	0.50
θ°	0°	-	8°	0°	-	8°

17.1 SSOP 16 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.053	-	0.069	1.3462	-	1.7526
A1	0.004	-	0.010	0.1016	-	0.254
A2	-	-	0.059	-	-	1.4986
b	0.008	-	0.012	0.2032	-	0.3048
b1	0.008	-	0.011	0.2032	-	0.2794
c	0.007	-	0.010	0.1778	-	0.254
c1	0.007	-	0.009	0.1778	-	0.2286
D	0.189	-	0.197	4.8006	-	5.0038
E1	0.150	-	0.157	3.81	-	3.9878
E	0.228	-	0.244	5.7912	-	6.1976
L	0.016	-	0.050	0.4064	-	1.27
e	0.025 BASIC			0.635 BASIC		
θ°	0°	-	8°	0°	-	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