SN8P2741A

USER'S MANUAL

Preliminary Version 0.4

SN8P2741A

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 0.1	Apr. 2014	First issue.
VER 0.2	May. 2014	2nd issue.
VER 0.3	Jun. 2014	Modify **ELECTRICAL CHARACTERISTIC** chapter: OP AMP CHARACTERISTIC: Input Offset Voltage Typ. = 3mV
VER 0.4	Sep. 2014	Modify **ELECTRICAL CHARACTERISTIC** chapter (OP AMP CHARACTERISTIC) : 1. Delete Maximum Output Voltage Swing description. 2. Add Output Voltage Swing description..

Table of Content

1 PRODUCT OVERVIEW

1.1 FEATURES

- **Memory configuration**
  ROM size: 4K * 16 bits.
  RAM size: 256 * 8 bits.

- **8 levels stack buffer.**

- **11 interrupt sources**
  10 internal interrupts: T0, T1, T2, TC0, ADC, CM0, CM1, CM2, UTX, URX.
  1 external interrupt: INT0

- **I/O pin configuration**
  Bi-directional: P0, P4.
  Wakeup: P0 level change.
  Pull-up resisters: P0, P4.
  Op-amp/Comparator pins: P0.2~P0.5, P4.0~P4.5.
  ADC input pin: P4.0~P4.6.
  Open-Drain output only pin: P0.1.
  Open-Drain pin: P4.7
  Reset shared pin: P4.7.

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

- **1-Level 2.1V LVD**
- **Powerful instructions**
  Instruction's length is one word.
  Most of instructions are one cycle only.
  All ROM area JMP/CALL instruction.
  All ROM area lookup table function (MOVC).

- **One 8-bit basic timer. (T0).**
- **Two 8-bit basic timers with TC0 pulse output trigger mode. (T1/T2).**
- **One 8-bit timer with duty/cycle programmable PWM and pulse generator. (TC0).**
- **One 2K/4K programmable buzzer output.**

- **One 8/9-bit UART interface.**
- **7-channel 12-bit SAR ADC.**
- **Internal 4V reference voltage for ADC/Comparator.**
- **3-set comparators.**
- **1-set rail-to-rail OP-amp.**

- **On chip watchdog timer and clock source is internal low clock RC type (16KHz @3V, 32KHz @5V).**

- **Two system clocks**
  Internal high clock: RC type 16MHz
  Internal low clock: RC type 16KHz(3V), 32KHz(5V)

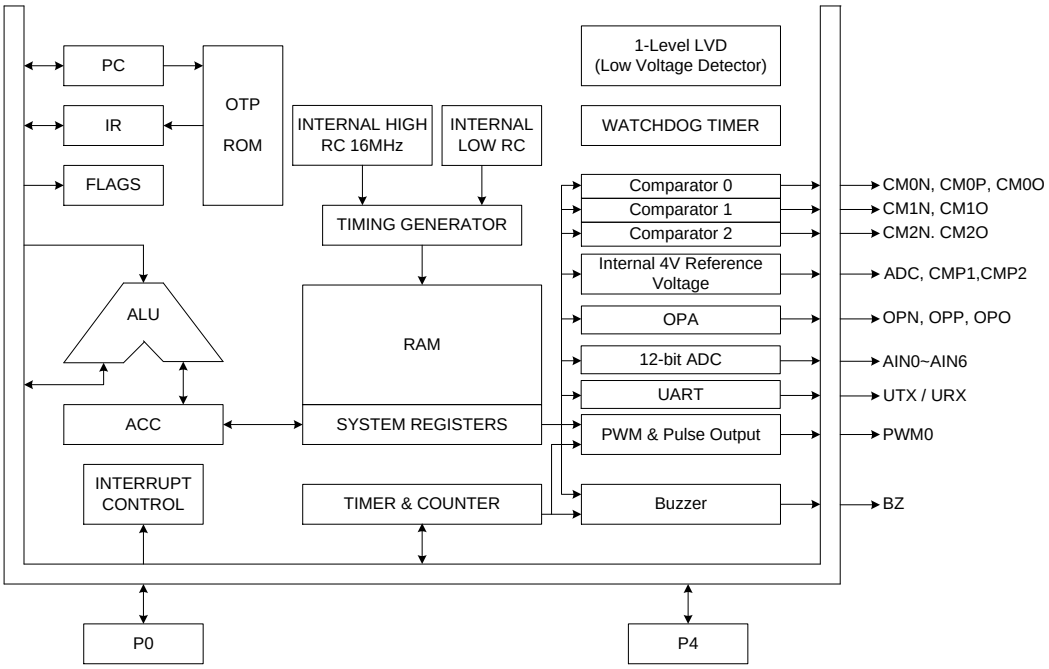
- **Four operating modes**
  Normal mode: Both high and low clock active
  Slow mode: Low clock only.
  Sleep mode: Both high and low clock stop
  Green mode: Periodical wakeup by timer

- **Package (Chip form support)**
  PDIP 16 pin
  SOP 16 pin

☞ **Features Selection Table**

CHIP	ROM	RAM	Stack	Timer				PWM	Pulse Out	BZ	Interrupt		I/O	UART	ADC	Int.Ref	OP-amp	Comparator	Package
				T0	TC0	T1	T2				Int	Ext							
SN8P2743	4K*16	128*8	8	V	V	-	-	1	1	1	6	1	22	-	8-ch	Vdd	1	3	SKDIP24 SOP24
SN8P2742	4K*16	128*8	8	V	V	-	-	1	1	1	6	1	18	-	6-ch	Vdd	1	3	PDIP20 SOP20
SN8P27411	4K*16	128*8	8	V	V	-	-	1	1	1	6	1	14	-	6-ch	Vdd	1	3	PDIP16 SOP16
SN8P2741A	4K*16	256*8	8	V	V	V	V	1	1	1	10	1	14	V	7-ch	Vdd,4V	1	3	PDIP16 SOP16

1.2 SYSTEM BLOCK DIAGRAM



1.3 PIN ASSIGNMENT

SN8P2741AP (DIP 16 pin)
SN8P2741AS (SOP 16 pin)

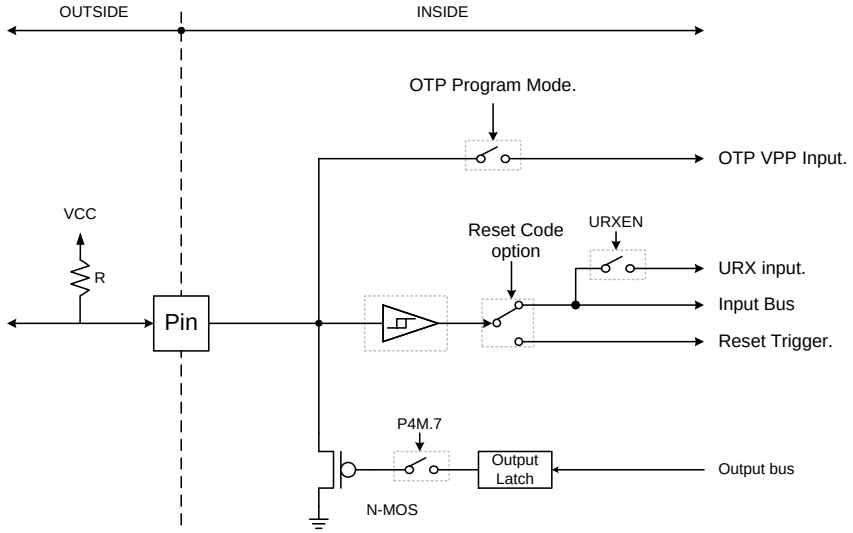
VSS	1	U	16	VDD
P0.0/INT0/BZ	2		15	P4.7/URX/RST/VPP
P0.1/PWM0	3		14	P4.6/AIN6/UTX
P0.2/CM0P	4		13	P4.5/AIN5/CM0O
P0.3/CM0N	5		12	P4.4/AIN4/CM1O
P0.4/CM1N	6		11	P4.3/AIN3/CM2O
P0.5/CM2N	7		10	P4.2/AIN2/OPP
P4.0/AIN0/OPO	8		9	P4.1/AIN1/OPN

1.4 PIN DESCRIPTIONS

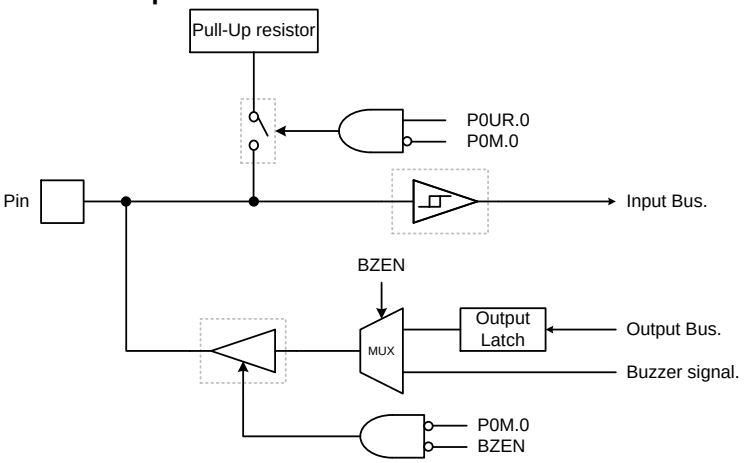
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital and analog circuit.
P4.7/RST/URX/VPP	I/O, P	RST: System external reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”.
		VPP: OTP 12.3V power input pin in programming mode.
		P4.7: Bi-direction pin. Schmitt trigger structure as input mode and without pull-up resisters. Open-drain structure as output mode.
		URX: UART receive input pin.
P0.0/INT0/BZ	I/O	P0.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		INT0: External interrupt 0 input pin.
		BZ: 2K/4K programmable buzzer output pin.
P0.1/PWM0	I/O	P0.1: Output pin with open-drain structure.
		PWM0: PWM output pin and pulse output pin.
P0.2/CM0P	I/O	P0.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		CM0P: The positive input pin of comparator 0.
P0.3/CM0N	I/O	P0.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		CM0N: The negative input pin of comparator 0.
P0.4/CM1N	I/O	P0.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		CM1N: The negative input pin of comparator 1.
P0.5/CM2N	I/O	P0.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		CM2N: The negative input pin of comparator 2.
P4.0/AIN0/OPO	I/O	P4.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		AIN0: ADC analog input pin.
		OPO: The output pin of OP amp.
P4.1/AIN1/OPN	I/O	P4.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		AIN1: ADC analog input pin.
		OPN: The negative input pin of OP amp.
P4.2/AIN2/OPP	I/O	P4.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		AIN2: ADC analog input pin.
		OPP: The positive input pin of OP amp.
P4.3/AIN3/CM2O	I/O	P4.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		AIN3: ADC analog input pin.
		CM2O: The output pin of comparator 2.
P4.4/AIN4/CM1O	I/O	P4.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		AIN4: ADC analog input pin.
		CM1O: The output pin of comparator 1.
P4.5/AIN5/CM0O	I/O	P4.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		AIN5: ADC analog input pin.
		CM0O: The output pin of comparator 0.
P4.6/AIN6/UTX	I/O	P4.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		AIN6: ADC analog input pin.
		UTX: UART transmit output pin.

1.5 PIN CIRCUIT DIAGRAMS

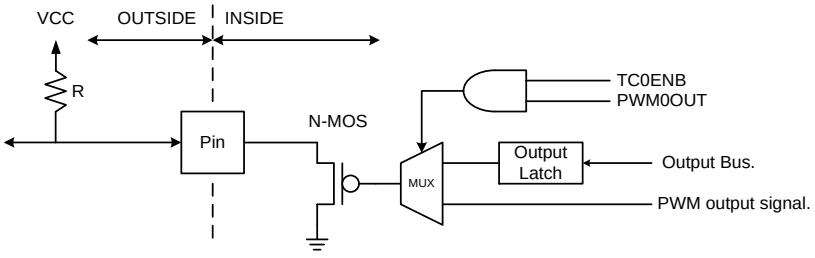
- **Reset shared pin with UART and Open-drain structure:**



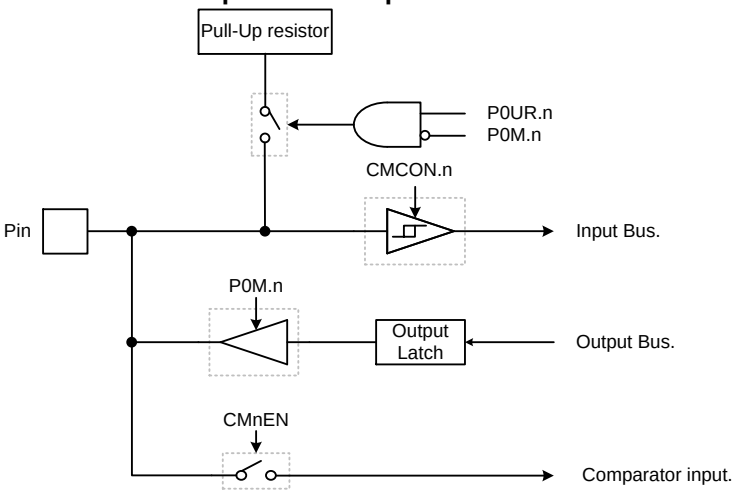
- **P0.0 shared pin structure:**



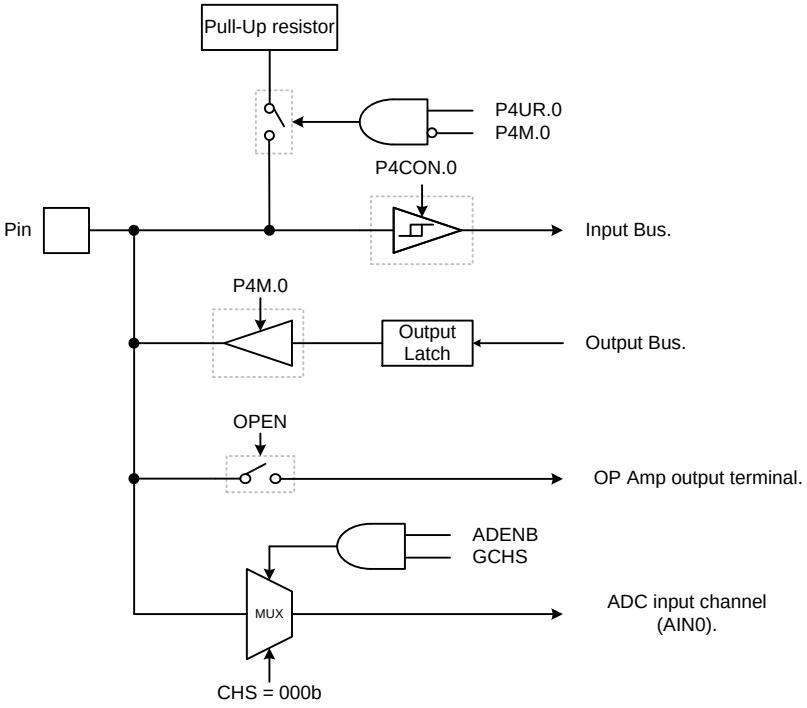
- **P0.1 Open-drain shared pin, output only:**



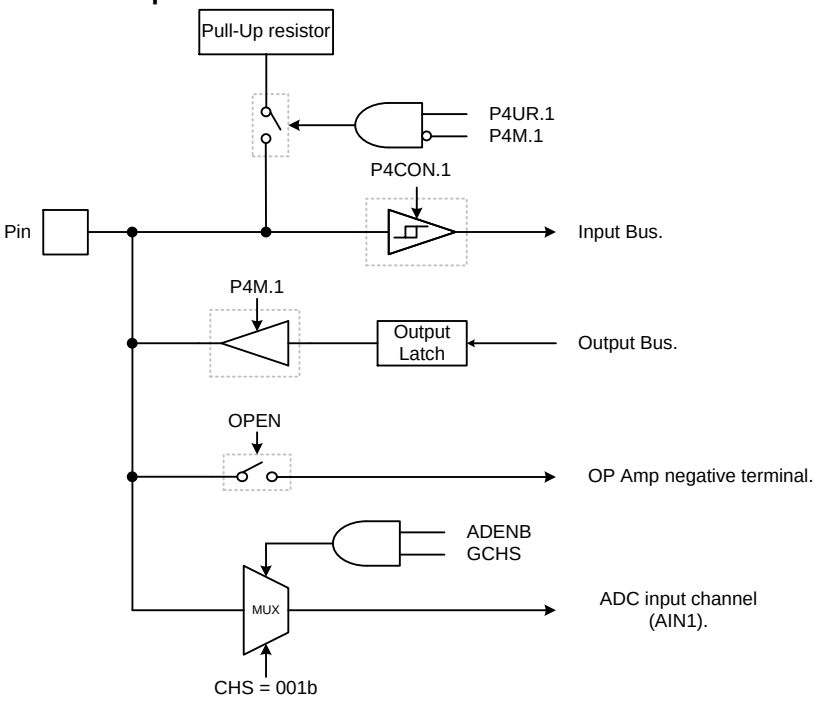
- **P0.2~P0.5 shared pin with comparator structure:**



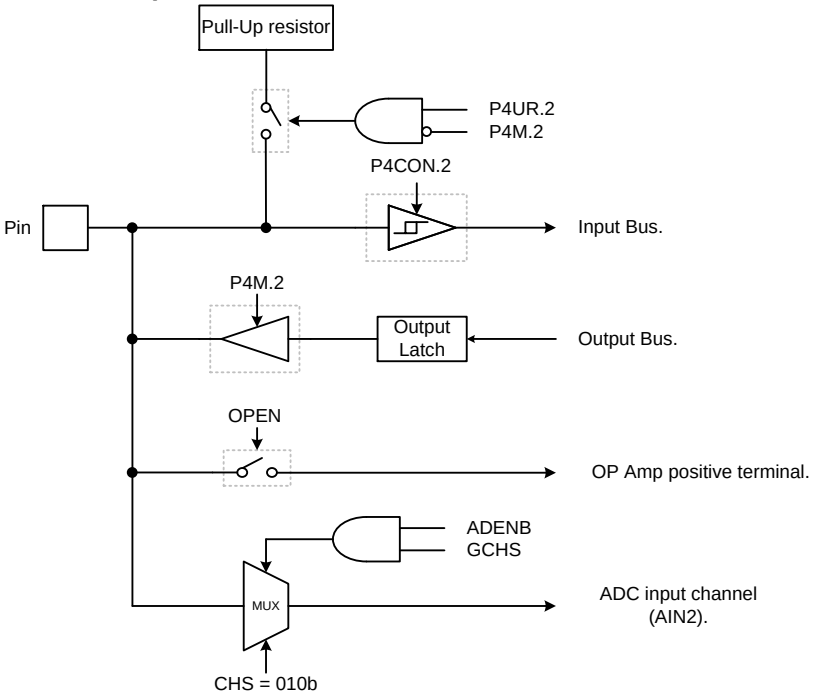
- **P4.0 shared pin with ADC/OP-AMP structure:**



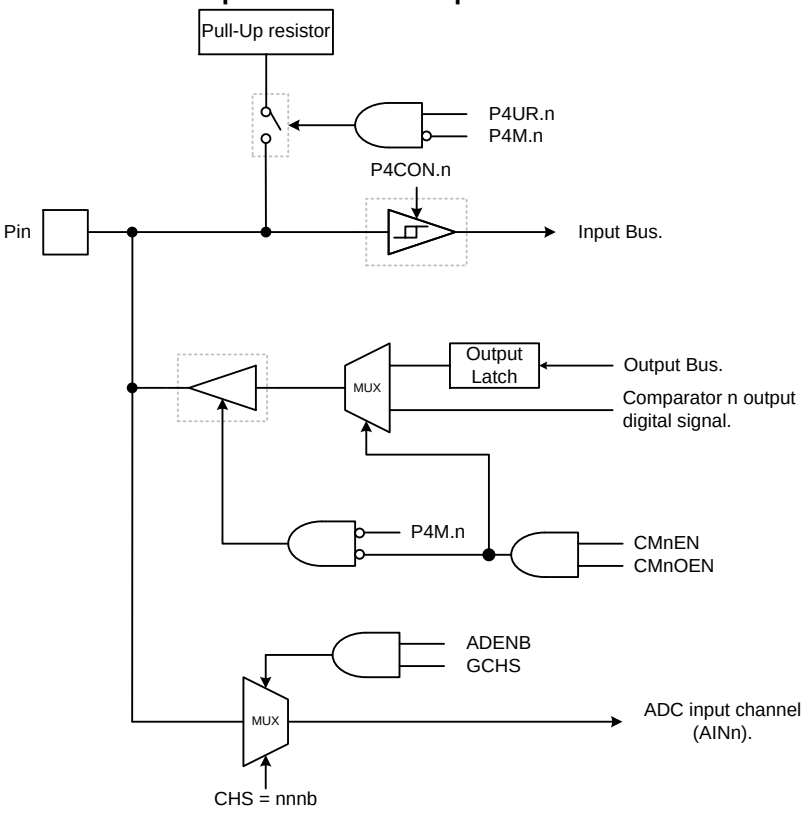
- **P4.1 shared pin with ADC/OP-AMP structure:**



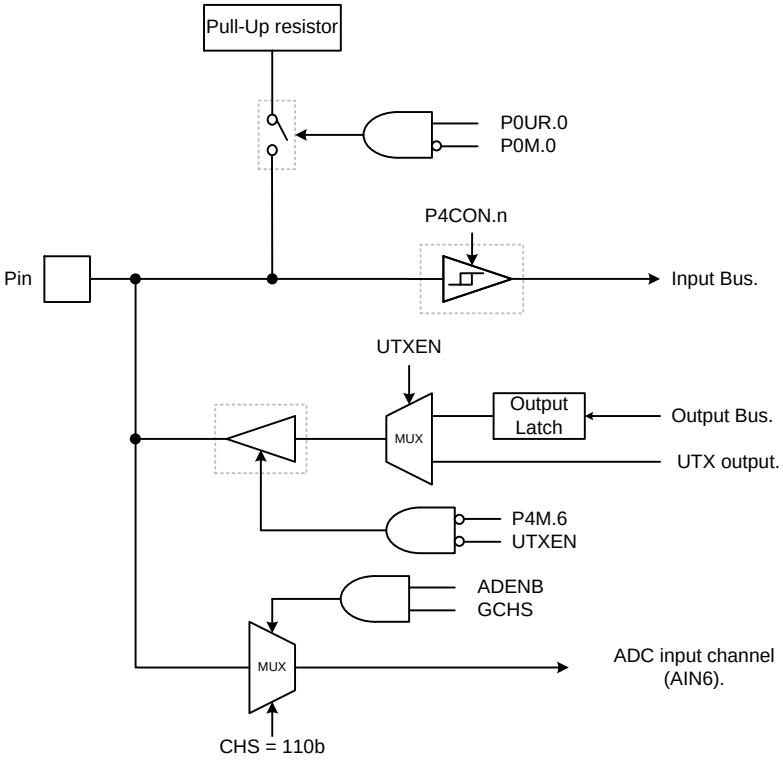
- **P4.2 shared pin with ADC/OP-AMP structure:**



- **P4.3~P4.5 shared pin with ADC/Comparator structure:**



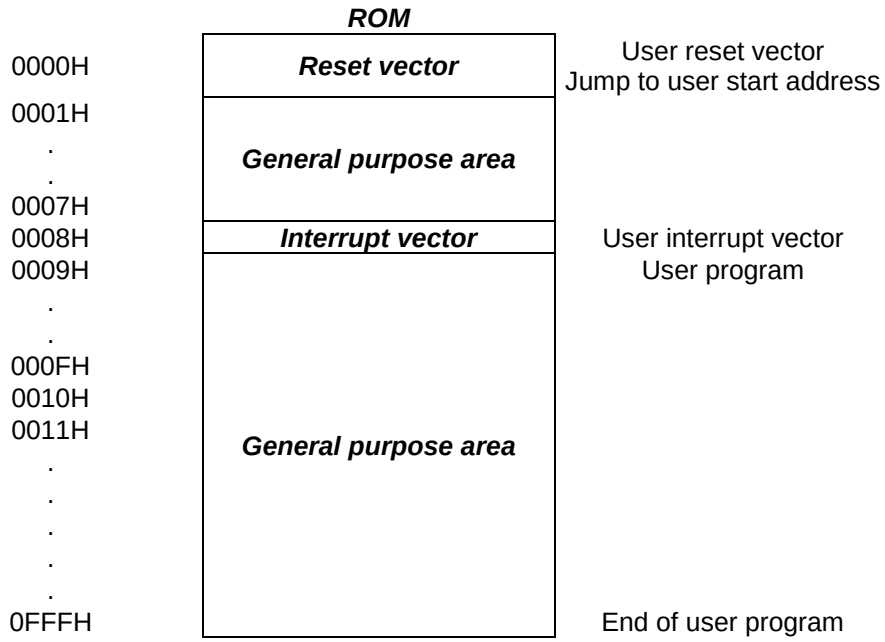
- **P4.6 shared pin with ADC/UART structure:**



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (ROM)

☞ ***4K words ROM***



The ROM includes Reset vector, Interrupt vector and General purpose 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: The "PUSH" and "POP" operations aren't through instruction (PUSH, POP) and can executed save and load ACC and PFLAG without NT0/NPD by hardware automatically.***

➢ **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. Saving ACC and PFLAG to buffers
                                        ; executed by hardware automatically.
                …
                …

                RETI                    ; Exit interrupt vector. Loading ACC and PFLAG from
                                        ; buffers executed by hardware automatically.

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. Saving ACC and PFLAG to buffers
                                    ; executed by hardware automatically.
                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.
                …
                …

                RETI                ; Exit interrupt vector. Loading ACC and PFLAG from
                                    ; buffers executed by hardware automatically.

                ENDP                ; End of program.
```

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

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

2.1.3 LOOK-UP TABLE DESCRIPTION

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

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

```
                B0MOV   Y, #TABLE1$M    ; To set lookup table1's middle address
                B0MOV   Z, #TABLE1$L    ; To set lookup table1's low address.
                MOVC                    ; To lookup data, R = 00H, ACC = 35H
                                        ;
                                        ; Increment the index address for next address.
                INCMS   Z               ; Z+1
                JMP     @F              ; Z is not overflow.
                INCMS   Y               ; Z overflow (FFH → 00), → Y=Y+1
                NOP                     ;
                                        ;
@@:             MOVC                    ; To lookup data, R = 51H, ACC = 05H.
                …                       ;
TABLE1:         DW      0035H           ; To define a word (16 bits) data.
                DW      5105H
                DW      2012H
                …
```

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

➢ **Example: INC_YZ macro.**

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

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

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

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

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

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

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

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

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

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

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

2.1.4 JUMP TABLE DESCRIPTION

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

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

➢ **Example: Jump table.**

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

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

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

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

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

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

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

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

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

➢ **Example: "@JMP_A" operation.**

; Before compiling program.

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

; After compiling program.

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

2.1.5 CHECKSUM CALCULATION

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

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

```
                MOV     A,#END_USER_CODE$L
                B0MOV   END_ADDR1, A        ; Save low end address to end_addr1
                MOV     A,#END_USER_CODE$M
                B0MOV   END_ADDR2, A        ; Save middle end address to end_addr2
                CLR     Y                   ; Set Y to 00H
                CLR     Z                   ; Set Z to 00H
@@:
                MOVC
                B0BSET  FC                  ; Clear C flag
                ADD     DATA1, A            ; Add A to Data1
                MOV     A, R
                ADC     DATA2, A            ; Add R to Data2
                JMP     END_CHECK           ; Check if the YZ address =    the end of code
AAA:
                INCMS   Z                   ; Z=Z+1
                JMP     @B                  ; If Z != 00H calculate to next address
                JMP     Y_ADD_1             ; If Z = 00H increase Y
END_CHECK:
                MOV     A, END_ADDR1
                CMPRS   A, Z                ; Check if Z = low end address
                JMP     AAA                 ; If Not jump to checksum calculate
                MOV     A, END_ADDR2
                CMPRS   A, Y                ; If Yes, check if Y = middle end address
                JMP     AAA                 ; If Not jump to checksum calculate
                JMP     CHECKSUM_END        ; If Yes checksum calculated is done.

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

2.2 DATA MEMORY (RAM)

✷ ***256 X 8-bit RAM***

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

The 256-byte general purpose RAM is in Bank 0/1. Sonix provides “Bank 0” type instructions (e.g. b0mov, b0add, b0bts1, b0bset…) to control Bank 0 RAM in non-zero RAM bank condition 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	L	H	R	Z	Y	-	PFLAG	RBANK	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	CMDB0	CMDB1	CMDB2	CM0M	CM1M	CM2M	OPM
A	T1M	T1C	T1R	T2M	T2C	T2R	-	-	-	-	-	-	-	-	P4CON	CMCON
B	-	ADM	ADB	ADR	ADT	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	-	-	-	-	P4M	-	-	-	INTRQ0	INTEN0	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	-	-	-	P4	-	INTRQ1	INTEN1	T0M	T0C	TC0M	TC0C	BZM	-	-	STKP
E	P0UR	-	-	-	P4UR	-	@HL	@YZ	TC0D	URTX	URRX	URCR	UTXD	URXD	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 SYSTEM REGISTER DESCRIPTION

H, L = Working, @HL addressing register.
R = Working register and ROM look-up data buffer.
RBANK = RAM bank select register.
CMDB1 = Comparator 1 output de-bounce control register.
CM0M = Comparator 0 mode register.
CM2M = Comparator 2 mode register.
T1M = T1 mode register.
T1R = T1 auto-reload data buffer.
T2C = T2 counting register.
P4CON = P4 configuration register.
ADM = ADC mode register.
ADR = ADC resolution select register.
PEDGE = P0.0 edge direction register.
INTENn = Interrupt n enable register.
WDTR = Watchdog timer clear register.
Pn = Port n data buffer.
BZM = 2K/4K buzzer mode register.
T0M = T0 mode register.
TC0M = TC0 mode register.
TC0R = TC0 auto-reload data buffer.
@HL = RAM HL indirect addressing index pointer.
URTX = UART transmit control register
URCR = UART baud rate control register.
URXD = UART receive data buffer
STK0~STK7 = Stack 0 ~ stack 7 buffer.

Y, Z = Working, @YZ and ROM addressing register.
PFLAG = Special flag register.
CMDB0 = Comparator 0 output de-bounce/delay control register.
CMDB2 = Comparator 2 output de-bounce control register.
CM1M = Comparator 1 mode register.
OPM = OP amp mode register.
T1C = T1 counting register.
T2M = T2 mode register.
T2R = T2 auto-reload data buffer.
CMCON = P0 configuration register.
ADB = ADC data buffer.
ADT = ADC offset calibration register.
INTRQn = Interrupt n request register.
OSCM = Oscillator mode register.
PnM = Port n input/output mode register.
PnUR = Port n pull-up resister control register.
PCH, PCL = Program counter.
T0C = T0 counting register.
TC0C = TC0 counting register.
TC0D = TC0 duty control register.
@YZ = RAM YZ indirect addressing index pointer.
URRX = UART receive control register.
UTXD = UART transmit data buffer.
STKP = Stack pointer buffer.

2.2.1.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD				C	DC	Z	R/W	PFLAG
087H								RBNKS0	R/W	RBANK
099H	CM0DY3	CM0DY2	CM0DY1	CM0DY0	CM0D3	CM0D2	CM0D1	CM0D0	R/W	CMDB0
09AH						CM1REF	CM1D1	CM1D0	R/W	CMDB1
09BH				PWS1	PWS0	CM2REF	CM2D1	CM2D0	R/W	CMDB2
09CH	CM0EN	CM0OEN	CM0OUT	CM0SF	CM0G	CM0CLK			R/W	CM0M
09DH	CM1EN	CM1OEN	CM1OUT	CM1SF	CM1G	CM1RS2	CM1RS1	CM1RS0	R/W	CM1M
09EH	CM2EN	CM2OEN	CM2OUT	CM2SF	CM2G	CM2RS2	CM2RS1	CM2RS0	R/W	CM2M
09FH							OPPSW	OPEN	R/W	OPM
0A0H	T1ENB	T1rate2	T1rate1	T1rate0				T1SF	R/W	T1M
0A1H	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0	R/W	T1C
0A2H	T1R7	T1R6	T1R5	T1R4	T1R3	T1R2	T1R1	T1R0	R	T1R
0A3H	T2ENB	T2rate2	T2rate1	T2rate0				T2SF	R/W	T2M
0A4H	T2C7	T2C6	T2C5	T2C4	T2C3	T2C2	T2C1	T2C0	R/W	T2C
0A5H	T2R7	T2R6	T2R5	T2R4	T2R3	T2R2	T2R1	T2R0	R	T2R
0AEH	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	W	P4CON
0AFH					CM2NCON	CM1NCON	CM0NCON	CM0PCON	W	CMCON
0B1H	ADENB	ADS	EOC	GCHS	ADREF	CHS2	CHS1	CHS0	R/W	ADM
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0B3H	INTR4V	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0B4H	ADTS1	ADTS0		ADT4	ADT3	ADT2	ADT1	ADT0	R/W	ADT
0B8H			P05M	P04M	P03M	P02M	-	P00M	R/W	P0M
0BFH							P00G1	P00G0	R/W	PEDGE
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C8H	ADCIRQ	T1IRQ	TC0IRQ	T0IRQ	CM2IRQ	CM1IRQ	CM0IRQ	P00IRQ	R/W	INTRQ0
0C9H	ADCIEN	T1IEN	TC0IEN	T0IEN	CM2IEN	CM1IEN	CM0IEN	P00IEN	R/W	INTEN0
0CAH	CPUM2	CPUM1	CPUM0	CLKMD1	CLKMD0	STPHX2	STPHX1	STPHX0	R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH					PC11	PC10	PC9	PC8	R/W	PCH
0D0H			P05	P04	P03	P02	P01	P00	R/W	P0
0D4H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4
0D6H						T2IRQ	UTXIRQ	URXIRQ	R/W	INTRQ1
0D7H						T2IEN	UTXIEN	URXIEN	R/W	INTEN1
0D8H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	TC0DIR	TC0PO	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	BZEN	BZRate1	BZrate0						R/W	BZM
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H			P05R	P04R	P03R	P02R	-	P00R	W	P0UR
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E8H	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0	R/W	TC0D
0E9H	UTXEN	UTXPEN	UTXPS	UARTM	URXBZ	UTXBZ	UTXD8	URXD8	R/W	URTX
0EAH	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0	R/W	URRX
0EBH	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0	R/W	URCR
0ECH	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0	R/W	UTXD
0EDH	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0	R	URXD
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H					S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H					S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H

0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H					S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H					S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H					S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH					S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH					S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH					S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ ***Note:***

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

2.2.2 ACCUMULATOR

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

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

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

                MOV         BUF, A

; Write a immediate data into ACC

                MOV         A, #0FH

; Write ACC data from BUF data memory

                MOV         A, BUF
```

The system will store ACC and PFLAG without NDT/NPD by hardware automatically when interrupt executed.

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

```
.CODE
INT_SERVICE:
                                    ; Save ACC to buffer.
                                    ; Save PFLAG without NDT/NPD to buffer.
                …           .
                …
                                    ; Load PFLAG without NDT/NPD form buffers.
                                    ; Load ACC form buffer.

                RETI                ; Exit interrupt service vector
```

2.2.3 PROGRAM FLAG

The PFLAG register contains the arithmetic status of ALU operation and system reset 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.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	-	-	-	C	DC	Z
Read/Write	R/W	R/W	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	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 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 12-bit binary counter separated into the high-byte 4 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 11.

	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	-	-	-	-	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

☞ **ONE ADDRESS SKIPPING**

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

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

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

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

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

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

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

INCS instruction:

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

INCMS instruction:

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

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

DECS instruction:

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

DECMS instruction:

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

☞ **MULTI-ADDRESS JUMPING**

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

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

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

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

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

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

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

2.2.5 H, L REGISTERS

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

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

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

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

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

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

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

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

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

2.2.6 Y, Z REGISTERS

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

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

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

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

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

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

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

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

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

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

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

2.2.7 R REGISTER

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

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

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

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

2.3 ADDRESSING MODE

2.3.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

✸ ***Note: In immediate addressing mode application, the specific RAM must be 0x80~0x87 working register.***

2.3.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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

2.3.3 INDIRECTLY ADDRESSING MODE

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

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

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

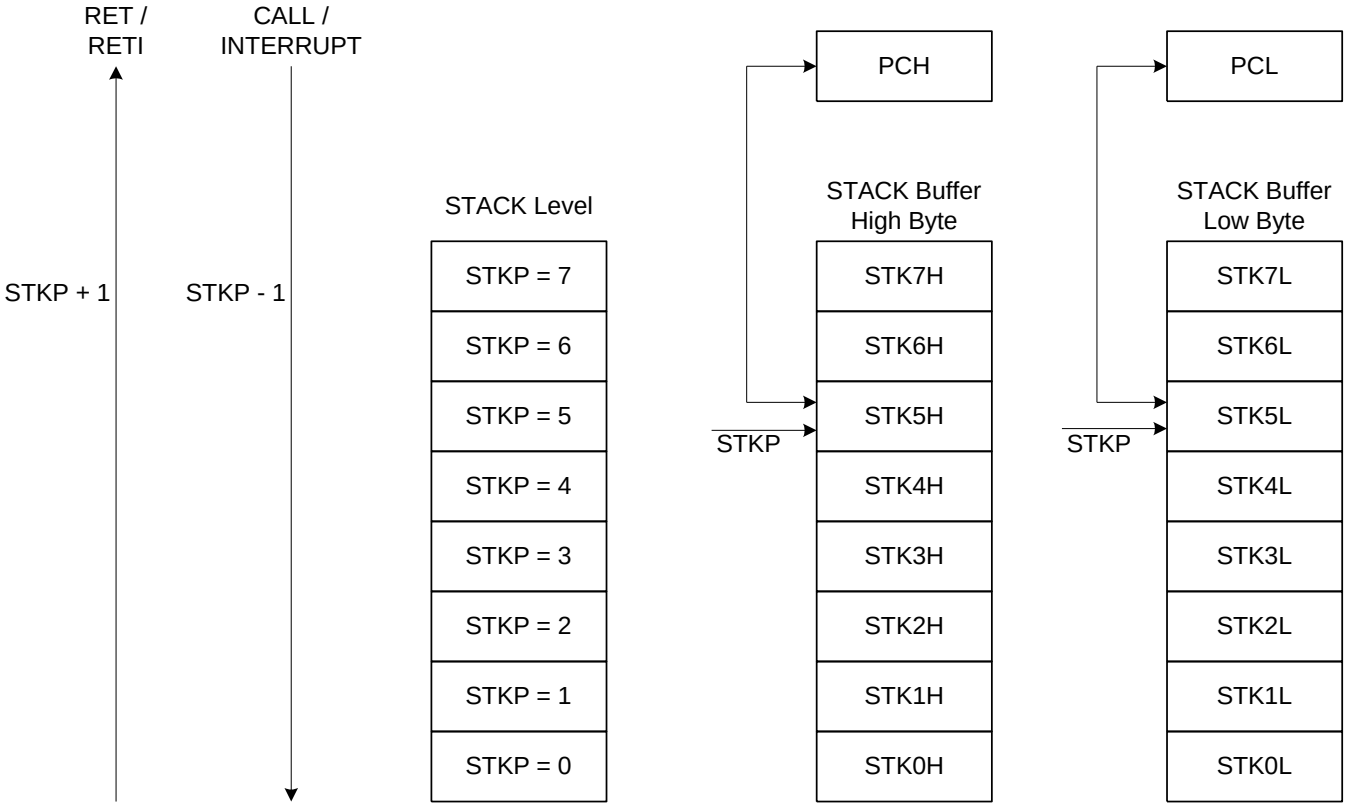
- **Example: Indirectly addressing mode with @YZ register**

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

2.4 STACK OPERATION

2.4.1 OVERVIEW

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



2.4.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 12-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

The two stack operations are writing to the top of the stack (push) and reading from the top of stack (pop). Push operation decrements the STKP and the pop operation increments each time. That makes the STKP always point to the top address of stack buffer and write the last program counter value (PC) into the stack buffer.

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

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

Bit[2:0] **STKPBn:** Stack pointer (n = 0 ~ 2)
Bit 7 **GIE:** Global interrupt control bit.
0 = Disable.
1 = Enable. Please refer to the interrupt chapter.

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

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

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

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

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

2.4.3 STACK OPERATION EXAMPLE

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

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

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

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

2.5 CODE OPTION TABLE

The code option is the system hardware configurations including 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/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
	Fhosc/16	Instruction cycle is 16 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.
	P47	Enable P47 input only without pull-up resister.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.

2.5.1 Fcpu code option

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

2.5.2 Reset_Pin code option

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

- **Reset:** The reset pin is external reset function. When falling edge trigger occurring, the system will be reset.
- **P47:** Set reset pin to general bi-direction pin (P4.7). The external reset function is disabled.

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)

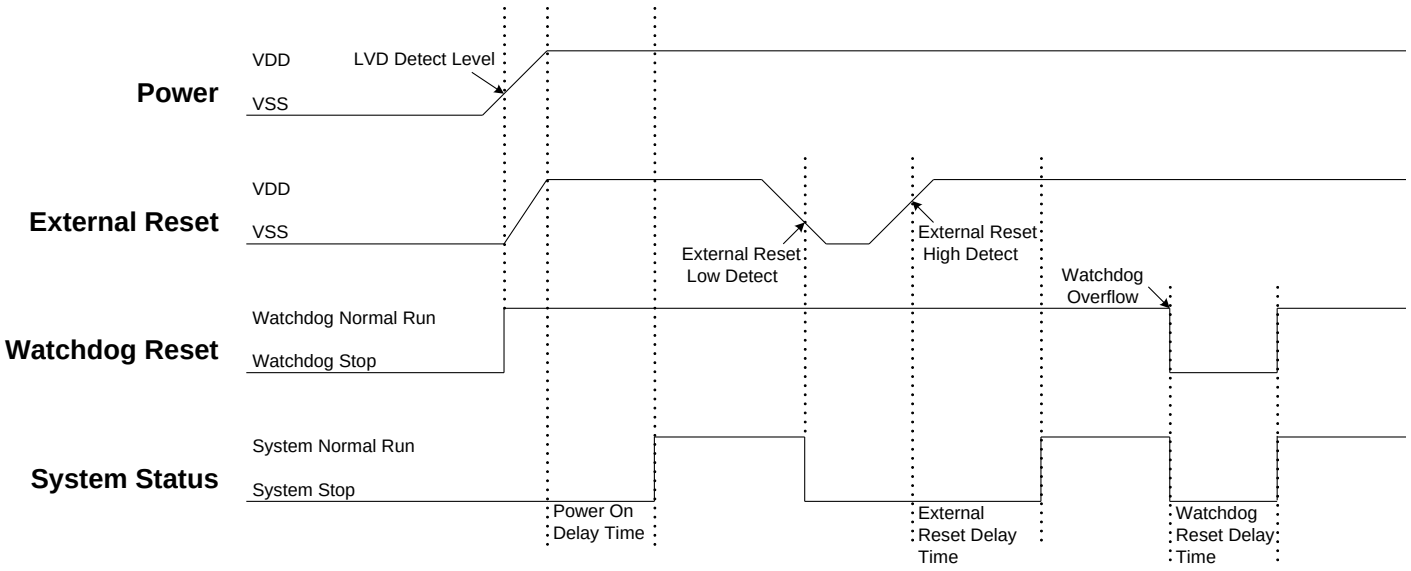
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status 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	-	-	-	C	DC	Z
Read/Write	R/W	R/W	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	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. The reset time is different which that causes the VDD rise rate and start-up time of oscillator is not fixed. Internal high speed RC type oscillator start-up time is very short. Under client terminal application, users have to take care the power on reset time for the master terminal requirement. The reset timing diagram is as following.



3.2 POWER ON RESET

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

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

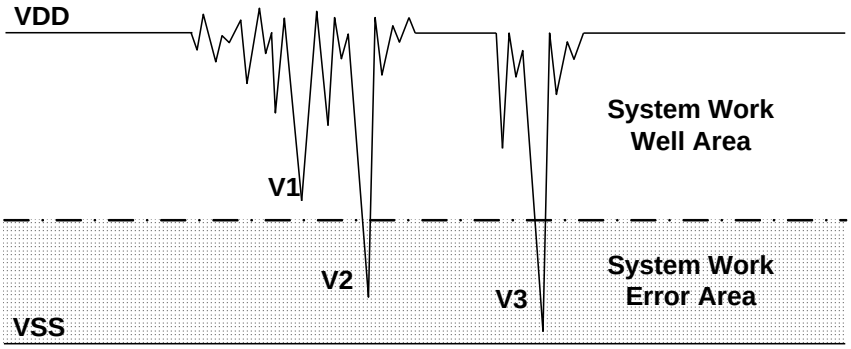
Watchdog timer application note is as following.

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

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

3.4 BROWN OUT RESET

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



Brown Out Reset Diagram

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

DC application:

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

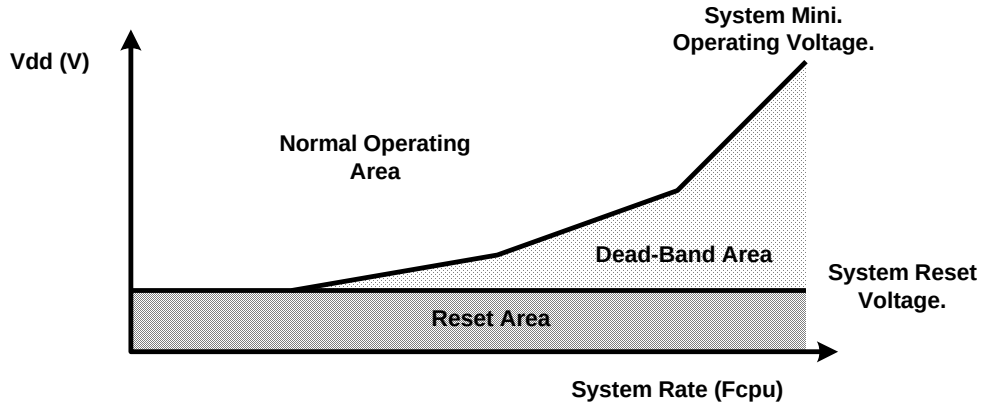
AC application:

In AC power application, the DC power is regulated from AC power source. This kind of power usually couples with AC noise that makes the DC power dirty. Or the external loading is very heavy, e.g. driving motor. The loading operating induces noise and overlaps with the DC power. VDD drops by the noise, and the system works under unstable power situation.

The power on duration and power down duration are longer in AC application. The system power on sequence protects the power on successful, but the power down situation is like DC low battery condition. When turn off the AC power, the VDD drops slowly and through the dead-band for a while.

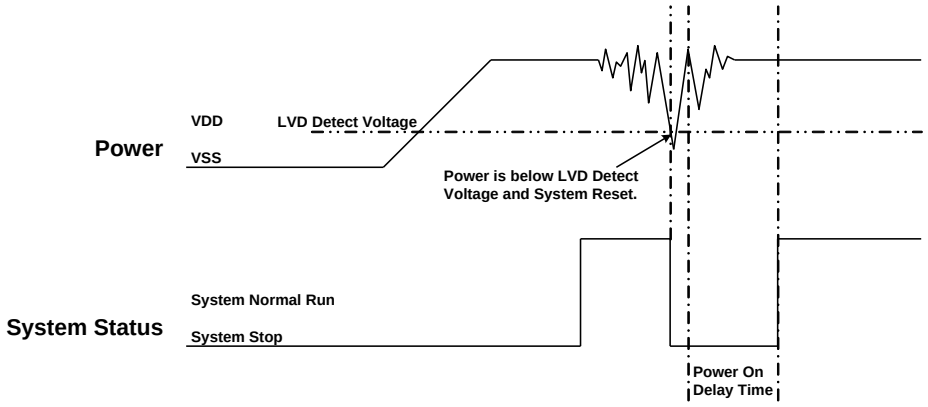
3.4.1 THE SYSTEM OPERATING VOLTAGE

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



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

3.4.2 LOW VOLTAGE DETECTOR (LVD)



The LVD (low voltage detector) is built-in Sonix 8-bit MCU to be brown out reset protection. When the VDD drops and is below LVD detect voltage, the LVD would be triggered, and the system is reset. The LVD detect level is different by each MCU. The LVD voltage level is a point of voltage and not easy to cover all dead-band range. Using LVD to improve brown out reset is depended 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.

3.4.3 BROWN OUT RESET IMPROVEMENT

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

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

✷ ***Note:***
1. ***The "Zener diode reset circuit", "Voltage bias reset circuit" and "External reset IC" can completely improve the brown out reset, DC low battery and AC slow power down conditions.***
2. ***For AC power application and enhance EFT performance, the system clock is 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.

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

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

3.5 EXTERNAL RESET

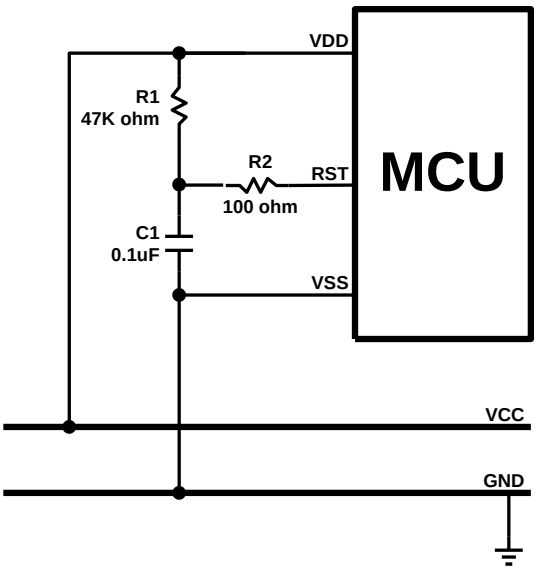
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

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

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

3.6 EXTERNAL RESET CIRCUIT

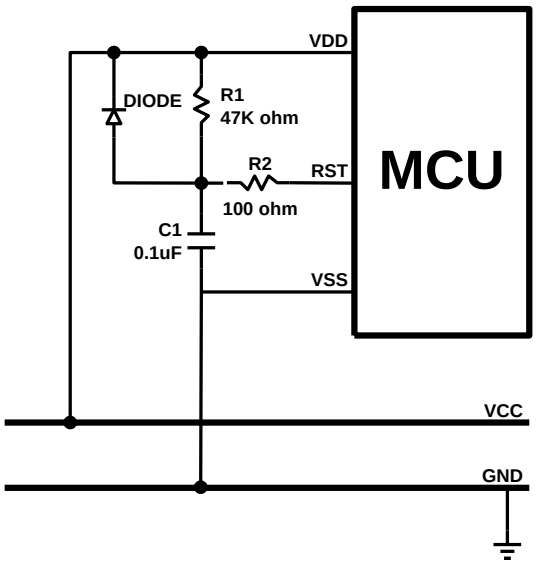
3.6.1 Simply RC Reset Circuit



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

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

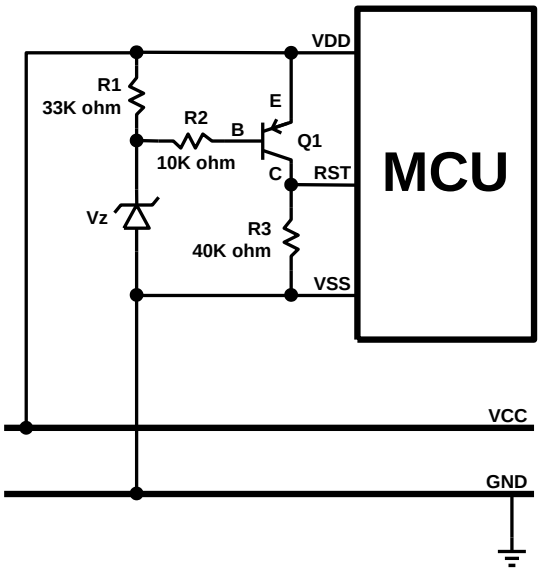
3.6.2 Diode & RC Reset Circuit



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

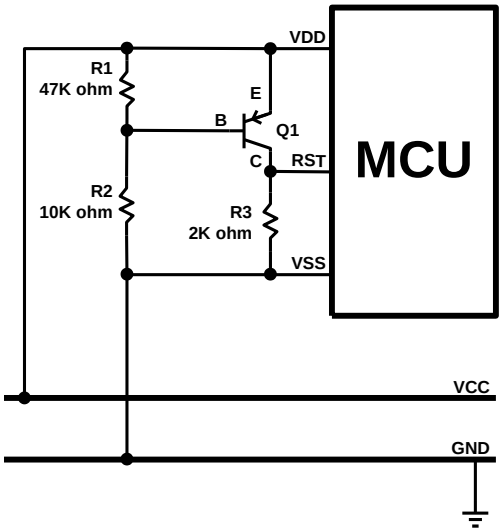
✸ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.6.3 Zener Diode Reset Circuit



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

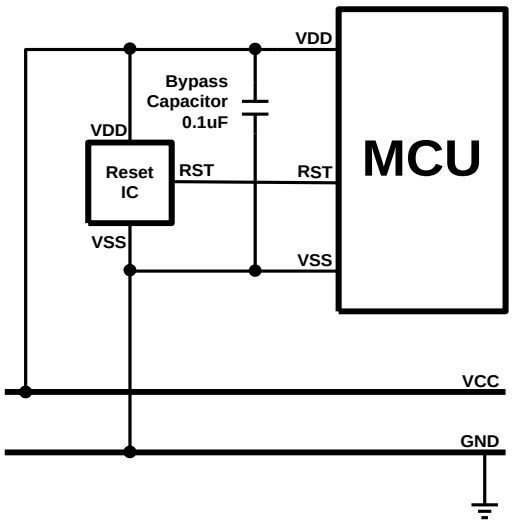
3.6.4 Voltage Bias Reset Circuit



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

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

3.6.5 External Reset IC



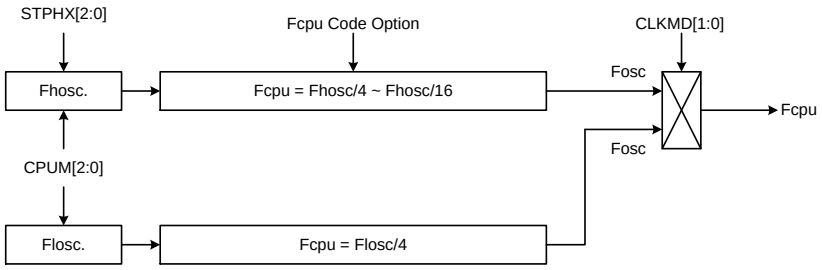
The external reset circuit also uses external reset IC to enhance MCU reset performance. This is a high cost and good effect solution. By different application and system requirement to select suitable reset IC. The reset circuit can improve all power variation.

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 from internal low-speed oscillator controlled by "CLKMD[1:0]" bits of OSCM register. Both high-speed clock and low-speed clock can be system clock source through a divider to decide the system clock rate.

- **High-speed oscillator:** Internal high-speed oscillator is 16MHz RC type called "**IHRC**".
- **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)

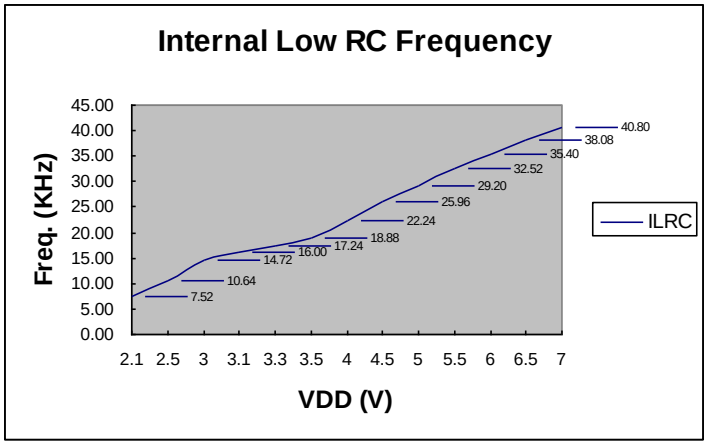
The system clock rate is instruction cycle called "**Fcpu**" which is divided from the system clock source and decides the system operating rate. Fcpu rate is selected by Fcpu code option and the range is **Fhosc/4~Fhosc/16** under system normal mode. If the system high clock source is IHRC, and the Fcpu code option is Fhosc/4, the Fcpu frequency is 16MHz/4 = 4MHz. 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 system default high-speed clock is internal high-speed 16MHz RC clock type. The accuracy is ±2% under commercial condition. The high-speed oscillator is not selected by "**High_CLK**" code option.

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 relation between the RC frequency and voltage is as the following figure.



The internal low RC supports watchdog clock source and system slow mode controlled by "CLKMD[1:0]" bits 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.**

```
MOV     A, OSCM         ; To stop internal high-speed oscillator and internal low-speed
AND     A, #00011111b   ; oscillator called power down mode (sleep mode).
OR      A, #10100000b
MOV     OSCM, A
```

✷ ***Note: The internal low-speed clock can't be turned off individually. It is controlled by CPUM[2:0] bits of OSCM register.***

4.5 SYSTEM CLOCK MEASUREMENT

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

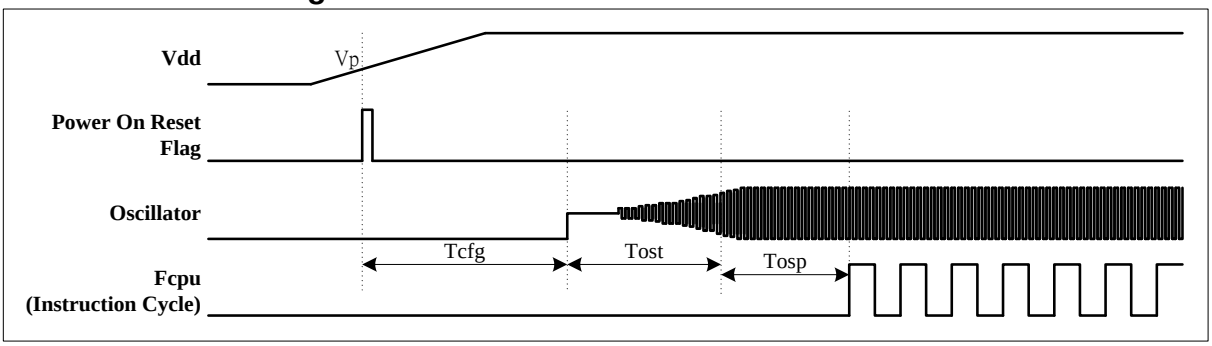
➢ **Example: Fcpu instruction cycle of external oscillator.**

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

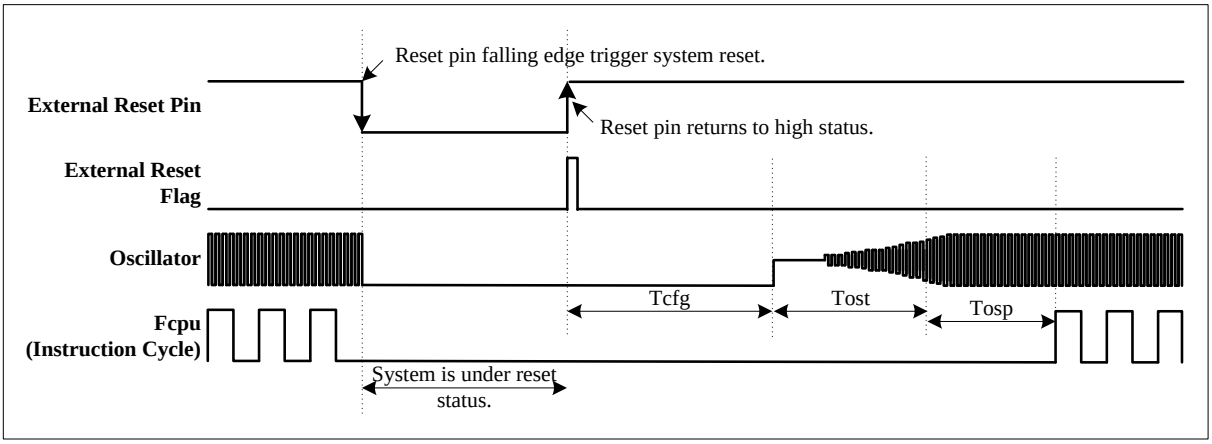
4.6 SYSTEM CLOCK TIMING

Parameter	Symbol	Description	Typical
Hardware configuration time	Tcfg	2048*F_{ILRC}	64ms @ F_{ILRC} = 32KHz 128ms @ F_{ILRC} = 16KHz
Oscillator start up time	Tost	The start-up time is depended on oscillator's material, factory and architecture. 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. 2048*F_{hosc} (Power on reset, LVD reset, watchdog reset, external reset pin active.)	128us @ F_{hosc} = 16MHz
		Oscillator warm-up time of power down mode wake-up condition. 16*F_{hosc}……Internal high-speed RC type oscillator.	2us @ F_{hosc} = 16MHz

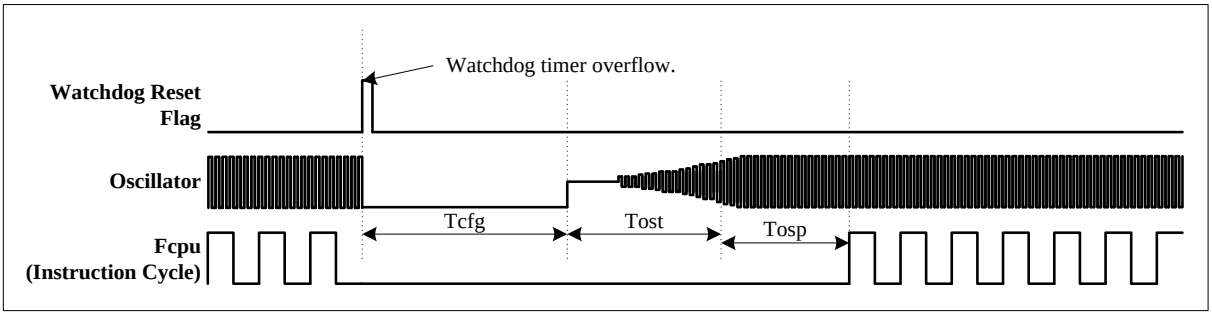
- **Power On Reset Timing**



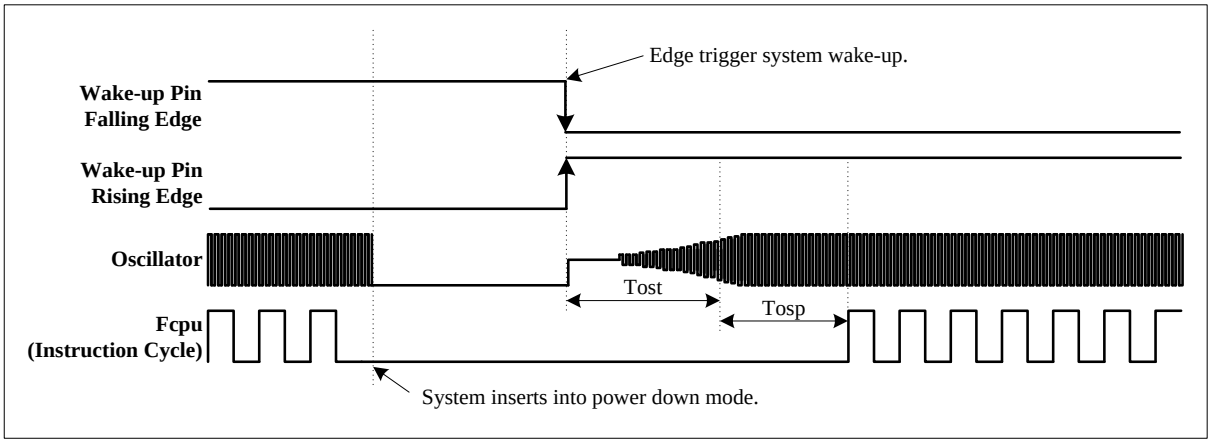
- **External Reset Pin Reset Timing**



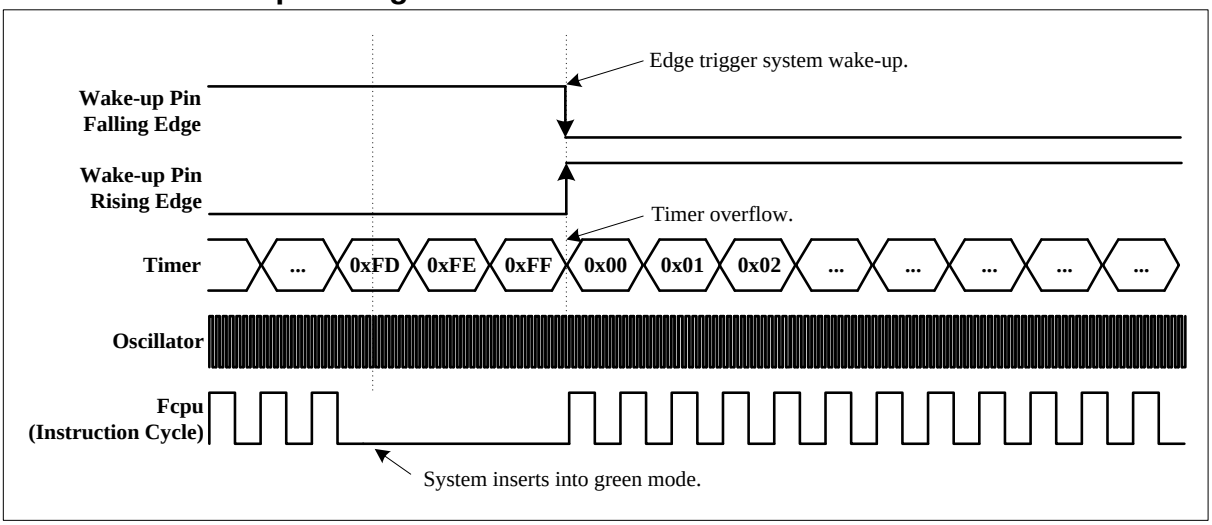
- **Watchdog Reset Timing**



Power Down Mode Wake-up Timing

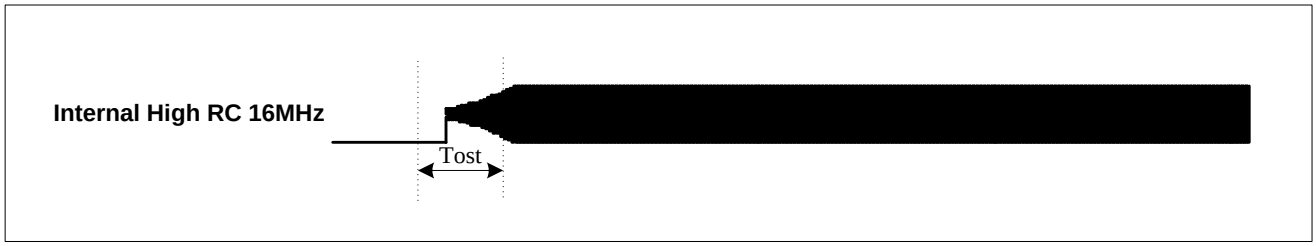


Green Mode Wake-up Timing



Oscillator Start-up Time

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



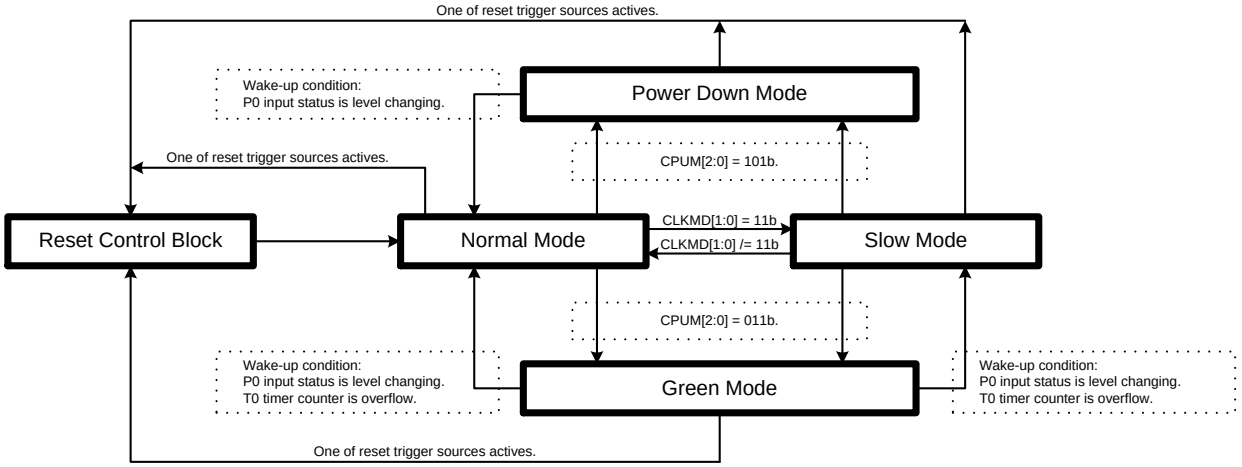
5 SYSTEM OPERATION MODE

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[2:0]	By STPHX[2:0]	Stop
ILRC	Running	Running	Running	Stop
CPU instruction	Executing	Executing	Stop	Stop
T0 timer	By T0ENB	By T0ENB	By T0ENB	Inactive
T1 timer	By T1ENB	By T1ENB	By T1ENB	Inactive
T2 timer	By T2ENB	By T2ENB	By T2ENB	Inactive
TC0 timer	By TC0ENB	By TC0ENB	By TC0ENB (PWM active)	Inactive
Buzzer	By BZEN	By BZEN	Stop	Stop
UART (RX)	By URXEN	By URXEN	Inactive	Inactive
UART (TX)	By UTXEN	By UTXEN	Inactive	Inactive
Comparator 0~2	By CM0EN/CM1EN/ CM2EN	By CM0EN/CM1EN/ CM2EN	By CM0EN/CM1EN/ CM2EN	By CM0EN/CM1EN/ CM2EN
OP	By OPEN	By OPEN	By OPEN	By OPEN
ADC	By ADENB	By ADENB	By ADENB	By ADENB
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, T0, Reset	P0, 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[1:0] bits of OSCM register. When CLKMD[1:0]≠11b, the system is in normal mode. When CLKMD[1:0]=11b, 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[2:0] bits 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[2:0]=101b. 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 hardware level change trigger. Any operating modes into power down mode, the system is waked up to normal mode. Inserting power down mode is controlled by CPUM[2:0] bits of OSCM register. When CPUM[2:0]=101b, the system inserts into power down mode. After system wake-up from power down mode, the CPUM[2:0] bits is disabled (zero status) automatically.

- The program stops executing, and full functions are disabled.
- All oscillators including external high speed oscillator, internal high speed oscillator and internal low speed oscillator stop.
- The power consumption is under 1uA.
- The system inserts into normal mode after wake-up from power down mode.
- The power down mode wake-up source is P0 level change trigger.

✷ ***Note: If the system is in normal mode, to set STPHX[2:0]=101b 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 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 level change trigger wake-up. The other one is internal timer (T0) with wake-up function occurring overflow. That's mean users can setup one fix period to timer, and the system is waked up until the time out. Inserting green mode is controlled by CPUM[2:0] bits of OSCM register. When CPUM[2:0]=011b, the system inserts into green mode. After system wake-up from green mode, the CPUM[2:0] 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 level change trigger and unique time overflow.
- PWM output functions active in green mode, but the timer can't wake-up the system as overflow.

✸ ***Note:***

1. ***Sonix provides "GreenMode" macro to control green mode operation. It is necessary to use "GreenMode" macro to control system inserting green mode.***
2. ***The macro includes six 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 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	CPUM2	CPUM1	CPUM0	CLKMD1	CLKMD0	STPHX2	STPHX1	STPHX0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:5] **CPUM [2:0]**: CPU operating mode control bit.
000b = Normal mode (default value)
101b = Power down mode (Sleep mode).
011b = Green mode.
001b, 010b, 100b, 110b, 111= Normal mode

Bit [4:3] **CLKMD [1:0]**: System high/low clock mode control bit.
00b = Normal mode. System clock is high clock (default value).
11b = Slow mode. System clock is low clock.
01b, 10b = System clock is high clock.

Bit [2:0] **STPHX [2:0]**: Internal high-speed RC (IHRC) control bit.
000b = Internal high-speed RC free runs (IHRC). Internal low-speed RC oscillator is still running (default value).
101b = Internal high-speed RC stops (IHRC). Internal low-speed RC oscillator is still running.
001b~100b, 110~111b = Internal high-speed RC free runs (IHRC). Internal low-speed RC oscillator is still running.

"STPHX[2:0]" bits controls internal high speed RC type oscillator operations. When "STPHX[2:0] ≠ 101b", the internal high speed RC type oscillator active. When "STPHX[2:0] = 101b", the internal high speed RC type oscillator and internal low speed RC type oscillator are disabled.

✸ ***Note: OSCM register is multi-bit control, recommend using "MOV" or "B0MOV" instructions to control the OSCM register, not use "bit control" type instructions (e.g. bset, bclr, b0bset, b0bclr…), or the system write fail value to OSCM register after executing instruction, the system occur function fail.***

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

```
MOV     A, OSCM
AND     A, #00011111b
OR      A, #10100000b
MOV     OSCM, A
```

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

```
MOV     A, OSCM
AND     A, #11100111b
OR      A, #00011000b
MOV     OSCM, A
```

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

```
MOV     A, OSCM
AND     A, #11100111b
OR      A, #00011000b
MOV     OSCM, A         ; Switch normal mode to slow mode

MOV     A, OSCM
AND     A, #11111000b
OR      A, #00000101b
MOV     OSCM, A         ; The internal high-speed oscillator stop
```

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

```
MOV     A, OSCM
AND     A, #11111000b
MOV     OSCM, A         ; The internal high-speed oscillator start.

NOP
NOP
NOP                     ; The internal high-speed oscillator warm-up

MOV     A, OSCM
AND     A, #11100111b
MOV     OSCM, A         ; Switch slow mode to normal mode
```

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

```
MOV     A, OSCM
AND     A, #00011111b
OR      A, #01100000b
MOV     OSCM, A
NOP
NOP
```

5.7 OPERATING MODE CONTROL MACRO

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

Macro	Length	Description
SleepMode	4-word	The system insets into Sleep Mode (Power Down Mode).
GreenMode	6-word	The system inserts into Green Mode.
SlowMode	8-word	The system inserts into Slow Mode and stops high speed oscillator.
Slow2Normal	9-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 internal high-speed oscillator stop).

```
    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.
```

5.8 WAKEUP

5.8.1 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 level change) and internal trigger (T0 timer overflow).

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

5.8.2 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 16 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 wakeup time is as the following.

The Wakeup time = 1/Fhosc * 16 (sec) + high clock start-up time

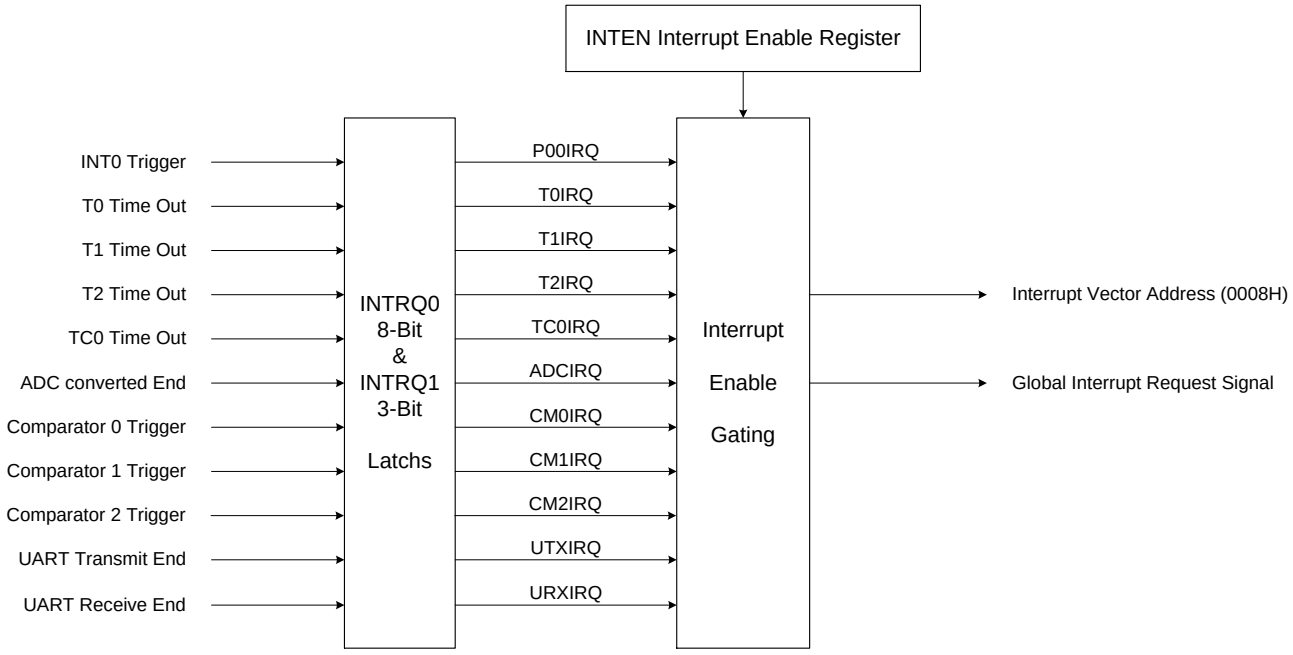
➢ **Example: In power down mode (sleep mode), the system is waked up. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.**

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

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 11 interrupt sources, including 10 internal interrupt (T0 / T1 / T2 / TC0 / CM0 / CM1 / CM2 / ADC / URRX / URTX) 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
INTEN0	ADCIEN	T1IEN	TC0IEN	T0IEN	CM2IEN	CM1IEN	CM0IEN	P00IEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **P00IEN:** External P0.0 interrupt (INT0) control bit.
0 = Disable INT0 interrupt function.
1 = Enable INT0 interrupt function.
Bit 1 **CM0IEN:** Comparator 0 interrupt control bit.
0 = Disable comparator 0 interrupt function.
1 = Enable comparator 0 interrupt function.
Bit 2 **CM1IEN:** Comparator 1 interrupt control bit.
0 = Disable comparator 1 interrupt function.
1 = Enable comparator 1 interrupt function.
Bit 3 **CM2IEN:** Comparator 2 interrupt control bit.
0 = Disable comparator 2 interrupt function.
1 = Enable comparator 2 interrupt function.
Bit 4 **T0IEN:** T0 timer interrupt control bit.
0 = Disable T0 interrupt function.
1 = Enable T0 interrupt function.
Bit 5 **TC0IEN:** TC0 timer interrupt control bit.
0 = Disable TC0 interrupt function.
1 = Enable TC0 interrupt function.
Bit 6 **T1IEN:** T1 timer interrupt control bit.
0 = Disable T1 interrupt function.
1 = Enable T1 interrupt function.
Bit 7 **ADCIEN:** ADC interrupt control bit.
0 = Disable ADC interrupt function.
1 = Enable ADC interrupt function.

0D7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN1	-	-	-	-	-	T2IEN	UTXIEN	URXIEN
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

Bit 0 **URXIEN:** UART receive interrupt control bit.
0 = Disable UART receive interrupt function.
1 = Enable UART receive interrupt function.
Bit 1 **UTXIEN:** UART transmit interrupt control bit.
0 = Disable UART transmit interrupt function.
1 = Enable UART transmit interrupt function.
Bit 2 **T2IEN:** T2 timer interrupt control bit.
0 = Disable T2 interrupt function.
1 = Enable T2 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
INTRQ0	ADCIRQ	T1IRQ	TC0IRQ	T0IRQ	CM2IRQ	CM1IRQ	CM0IRQ	P00IRQ
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **P00IRQ:** External P0.0 interrupt (INT0) request flag.
0 = None INT0 interrupt request.
1 = INT0 interrupt request.
Bit 1 **CM0IRQ:** Comparator 0 interrupt request flag.
0 = None comparator 0 interrupt request.
1 = Comparator 0 interrupt request.
Bit 2 **CM1IRQ:** Comparator 1 interrupt request flag.
0 = None comparator 1 interrupt request.
1 = Comparator 1 interrupt request.
Bit 3 **CM2IRQ:** Comparator 2 interrupt request flag.
0 = None comparator 2 interrupt request.
1 = Comparator 2 interrupt request.
Bit 4 **T0IRQ:** T0 timer interrupt request flag.
0 = None T0 interrupt request.
1 = T0 interrupt request.
Bit5 **TC0IRQ:** TC0 timer interrupt request flag.
0 = None TC0 interrupt request.
1 = TC1 interrupt request.
Bit 6 **T1IRQ:** T1 timer interrupt request flag.
0 = None T1 interrupt request.
1 = T1 interrupt request.
Bit 7 **ADCIRQ:** ADC interrupt request flag.
0 = None ADC interrupt request.
1 = ADC interrupt request.

0D6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ1	-	-	-	-	-	T2IRQ	UTXIRQ	URXIRQ
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

Bit 0 **URXIRQ:** UART receive interrupt request flag.
0 = None UART receive interrupt request.
1 = UART receive interrupt request.
Bit 1 **UTXIRQ:** UART transmit interrupt request flag.
0 = None UART transmit interrupt request.
1 = UART transmit interrupt request.
Bit 2 **T2IRQ:** T2 timer interrupt request flag.
0 = None T2 interrupt request.
1 = T1 interrupt request.

6.4 GIE GLOBAL INTERRUPT OPERATION

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

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

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

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

```
B0BSET      FGIE            ; Enable GIE
```

✸ ***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. The first procedure is "PUSH" operation. The end procedure after interrupt service routine execution is "POP" operation. **The "PUSH" and "POP" operations aren't through instruction (PUSH, POP) and executed by hardware automatically.** The **"PUSH" and "POP" operations** are saved and loaded **ACC**, **PFLAG** data into buffers and avoid main routine error after interrupt service routine finishing.

✸ ***Note: "PUSH", "POP" operations are saved and loaded ACC/PFLAG without (NT0, NPD). PUSH/POP buffer is a unique buffer and only one level.***

- **"PUSH" operation: PUSH operation saves the contents of ACC and PFLAG into hardware buffers. PUSH operation executes before program counter points to interrupt vector.**
- **"POP" operation: POP operation reloads the contents of ACC and PFLAG from hardware buffers. POP operation executes as RETI instruction executed.**

➢ **Example: Store ACC and PAFLG data when interrupt service routine executed.**

```
            ORG     0
            JMP     START

            ORG     8               ; Interrupt vector. Saving ACC and PFLAG to buffers
            JMP     INT_SERVICE     ; executed by hardware automatically.

            ORG     10H
START:
            …
INT_SERVICE:
            …
            …
            RETI                    ;Load ACC and PFLAG from buffers and exit interrupt
            …                       ; service vector.
            ENDP
```

6.6 EXTERNAL INTERRUPT OPERATION (INT0)

Sonix provides 1 external interrupt sources in the micro-controller. INT0 is external interrupt trigger sources and build in edge trigger configuration function. When the external edge trigger occurs, the external interrupt request flag will be set to “1” when the external interrupt control bit enabled. If the external interrupt control bit is disabled, the external interrupt request flag won’t active when external edge trigger occurrence. When external interrupt control bit is enabled and external interrupt edge trigger is occurring, the program counter will jump to the interrupt vector (ORG 8) and execute interrupt service routine.

The external interrupt builds in wake-up latch function. That means when the system is triggered wake-up from power down mode, the wake-up source is external interrupt source (P0.0), 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[1:0] **P00G[1:0]:** INT0 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

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

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

        B0BSET  FP00IEN         ; Enable INT0 interrupt service
        B0BCLR  FP00IRQ         ; Clear INT0 interrupt request flag
        B0BSET  FGIE            ; Enable GIE
```

➢ **Example: INT0 interrupt service routine.**

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

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

        B0BCLR  FP00IRQ         ; Reset P00IRQ
        …                       ; INT0 interrupt service routine
EXIT_INT:
        …
        RETI                    ; Exit interrupt vector. Loading ACC and PFLAG from
                                ; buffers executed by hardware automatically.
```

6.7 T0 INTERRUPT OPERATION

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

➢ **Example: T0 interrupt request setup. Fcpu = 4MHz / 4.**

```
        B0BCLR      FT0IEN          ; Disable T0 interrupt service
        B0BCLR      FT0ENB          ; Disable T0 timer
        MOV         A, #50H         ;
        B0MOV       T0M, A          ; Set T0 clock = Fosc / 4
        MOV         A, #9CH         ; Set T0C initial value = 9CH
        B0MOV       T0C, A          ; Set T0 interval = 100 us

        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. Saving ACC and PFLAG to buffers
        JMP         INT_SERVICE     ; executed by hardware automatically.
INT_SERVICE:

        …

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

        B0BCLR      FT0IRQ          ; Reset T0IRQ
        MOV         A, #9CH
        B0MOV       T0C, A          ; Reset T0C.
        …                           ; T0 interrupt service routine
        …
EXIT_INT:
        …

        RETI                        ; Exit interrupt vector. Loading ACC and PFLAG from
                                    ; buffers executed by hardware automatically.
```

6.8 T1 INTERRUPT OPERATION

When the T1C counter overflows, the T1IRQ will be set to "1" no matter the T1IEN is enable or disable. If the T1IEN and the trigger event T1IRQ is set to be "1". As the result, the system will execute the interrupt vector. If the T1IEN = 0, the trigger event T1IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the T1IEN is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: T1 interrupt request setup. Fcpu = 16MHz / 4.**

```
        B0BCLR      FT1IEN      ; Disable T1 interrupt service
        B0BCLR      FT1ENB      ; Disable T1 timer
        MOV         A, #60H     ;
        B0MOV       T1M, A      ; Set T1 clock = Fcpu / 4
        MOV         A, #9CH     ; Set T1C initial value = 9CH
        B0MOV       T1C, A      ; Set T1 interval = 100 us
        B0MOV       T1R, A      ; Set T1 reload buffer.

        B0BSET      FT1IEN      ; Enable T1 interrupt service
        B0BCLR      FT1IRQ      ; Clear T1 interrupt request flag
        B0BSET      FT1ENB      ; Enable T1 timer

        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: T1 interrupt service routine.**

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

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

        B0BCLR      FT1IRQ      ; Reset T1IRQ
        …                       ; T1 interrupt service routine
        …
EXIT_INT:

        RETI                    ; Exit interrupt vector. Loading ACC and PFLAG from
                                ; buffers executed by hardware automatically.
```

6.9 T2 INTERRUPT OPERATION

When the T2C counter overflows, the T2IRQ will be set to “1” no matter the T2IEN is enable or disable. If the T2IEN and the trigger event T2IRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the T2IEN = 0, the trigger event T2IRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the T2IEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: T2 interrupt request setup. Fcpu = 16MHz / 4.**

```
        B0BCLR      FT2IEN      ; Disable T2 interrupt service
        B0BCLR      FT2ENB      ; Disable T2 timer
        MOV         A, #60H     ;
        B0MOV       T2M, A      ; Set T2 clock = Fcpu / 4
        MOV         A, #9CH     ; Set T2C initial value = 9CH
        B0MOV       T2C, A      ; Set T2 interval = 100 us
        B0MOV       T2R, A      ; Set T2 reload buffer.

        B0BSET      FT2IEN      ; Enable T2 interrupt service
        B0BCLR      FT2IRQ      ; Clear T2 interrupt request flag
        B0BSET      FT2ENB      ; Enable T2 timer

        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: T2 interrupt service routine.**

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

        B0BTS1      FT2IRQ      ; Check T2IRQ
        JMP         EXIT_INT    ; T2IRQ = 0, exit interrupt vector

        B0BCLR      FT2IRQ      ; Reset T2IRQ
        …                       ; T2 interrupt service routine
        …
EXIT_INT:

        RETI                    ; Exit interrupt vector. Loading ACC and PFLAG from
                                ; buffers executed by hardware automatically.
```

6.10 TC0 INTERRUPT OPERATION

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

➢ **Example: TC0 interrupt request setup. Fhosc = 16MHz / 16.**

```
        B0BCLR      FTC0IEN         ; Disable TC0 interrupt service
        B0BCLR      FTC0ENB         ; Disable TC0 timer
        MOV         A, #48H         ;
        B0MOV       TC0M, A         ; Set TC0 clock = Fhosc / 16
        MOV         A, #9CH         ; Set TC0C initial value = 9CH
        B0MOV       TC0C, A         ; Set TC0 interval = 100 us
        B0MOV       TC0R, A         ; Set TC0 reload buffer.

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

        B0BSET      FGIE            ; Enable GIE
```

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

```
        ORG         8               ; Interrupt vector. Saving ACC and PFLAG to buffers
        JMP         INT_SERVICE     ; executed by hardware automatically.
INT_SERVICE:

        …

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

        B0BCLR      FTC0IRQ         ; Reset TC0IRQ
        …                           ; TC0 interrupt service routine
        …
EXIT_INT:
        …

        RETI                        ; Exit interrupt vector. Loading ACC and PFLAG from
                                    ; buffers executed by hardware automatically.
```

6.11 COMPARATOR INTERRUPT OPERATION (CMP0~CMP2)

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

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M	CM0EN	CM0OEN	CM0OUT	CM0SF	CM0G	CM0CLK	-	-
Read/Write	R/W	R/W	R	R/W	R/W	R/W	-	-
After Reset	0	0	0	0	0	0	-	-

Bit 3 **CM0G:** Comparator 0 interrupt trigger direction control bit.
0 = Falling edge trigger. Comparator output status is from high to low as CM0P < CM0N.
1 = Rising edge trigger. Comparator output status is from low to high as CM0P > CM0N.

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM1M	CM1EN	CM1OEN	CM1OUT	CM1SF	CM1G	CM1RS2	CM1RS1	CM1RS0
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 3 **CM1G:** Comparator 1 output trigger direction control bit.
0 = Falling edge trigger. Comparator output status is from high to low as CM1P < CM1N.
1 = Rising edge trigger. Comparator output status is from low to high as CM1P > CM1N.

09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM2M	CM2EN	CM2OEN	CM2OUT	CM2SF	CM2G	CM2RS2	CM2RS1	CM2RS0
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 3 **CM2G:** Comparator 2 output trigger direction control bit.
0 = Falling edge trigger. Comparator output status is from high to low as CM2P < CM2N.
1 = Rising edge trigger. Comparator output status is from low to high as CM2P > CM2N.

➢ **Example: Setup comparator 0 interrupt request and falling edge trigger.**

```
MOV     A, #00H
B0MOV   CM0M, A         ; Set comparator 0 interrupt trigger as falling edge.

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

➢ **Example: Setup comparator 1 interrupt request and falling edge trigger.**

```
MOV     A, #00H
B0MOV   CM1M, A         ; Set comparator 1 interrupt trigger as falling edge.

B0BSET  FCM1IEN         ; Enable comparator 1 interrupt service
B0BCLR  FCM1IRQ         ; Clear comparator 1 interrupt request flag
B0BSET  FCM1EN          ; Enable comparator 1.
B0BSET  FGIE            ; Enable GIE
```

➢ **Example: Setup comparator 2 interrupt request and falling edge trigger.**

```
        MOV     A, #00H
        B0MOV   CM2M, A         ; Set comparator 2 interrupt trigger as falling edge.

        B0BSET  FCM2IEN         ; Enable comparator 2 interrupt service
        B0BCLR  FCM2IRQ         ; Clear comparator 2 interrupt request flag
        B0BSET  FCM2EN          ; Enable comparator 2.
        B0BSET  FGIE            ; Enable GIE
```

➢ **Example: Comparator 0 interrupt service routine.**

```
        ORG     8               ; Interrupt vector. Saving ACC and PFLAG to buffers
        JMP     INT_SERVICE     ; executed by hardware automatically.
INT_SERVICE:

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

        B0BCLR  FCM0IRQ         ; Reset CM0IRQ
        …                       ; Comparator 0 interrupt service routine
EXIT_INT:
        …
        RETI                    ; Exit interrupt vector. Loading ACC and PFLAG from
                                ; buffers executed by hardware automatically.
```

➢ **Example: Comparator 1 interrupt service routine.**

```
        ORG     8               ; Interrupt vector. Saving ACC and PFLAG to buffers
        JMP     INT_SERVICE     ; executed by hardware automatically.
INT_SERVICE:

        …
        B0BTS1  FCM1IRQ         ; Check CM1IRQ
        JMP     EXIT_INT        ; CM1IRQ = 0, exit interrupt vector

        B0BCLR  FCM1IRQ         ; Reset CM1IRQ
        …                       ; Comparator 1 interrupt service routine
EXIT_INT:
        …
        RETI                    ; Exit interrupt vector. Loading ACC and PFLAG from
                                ; buffers executed by hardware automatically.
```

➢ **Example: Comparator 2 interrupt service routine.**

```
        ORG     8               ; Interrupt vector. Saving ACC and PFLAG to buffers
        JMP     INT_SERVICE     ; executed by hardware automatically.
INT_SERVICE:

        …
        B0BTS1  FCM2IRQ         ; Check CM2IRQ
        JMP     EXIT_INT        ; CM2IRQ = 0, exit interrupt vector

        B0BCLR  FCM2IRQ         ; Reset CM2IRQ
        …                       ; Comparator 2 interrupt service routine
EXIT_INT:
        …
        RETI                    ; Exit interrupt vector. Loading ACC and PFLAG from
                                ; buffers executed by hardware automatically.
```

6.12 ADC INTERRUPT OPERATION

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

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

```
        B0BCLR      FADCIEN         ; Disable ADC interrupt service

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

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

        B0BSET      FADS            ; Start ADC transformation
```

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

```
        ORG         8               ; Interrupt vector. Saving ACC and PFLAG to buffers
        JMP         INT_SERVICE     ; executed by hardware automatically.
INT_SERVICE:

        …

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

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

        RETI                        ; Exit interrupt vector. Loading ACC and PFLAG from
                                    ; buffers executed by hardware automatically.
```

6.13 UART INTERRUPT OPERATION

When the UART transmitter successfully, the URXIRQ/UTXIRQ will be set to “1” no matter the URXIEN/UTXIEN is enable or disable. If the URXIEN/UTXIEN and the trigger event URXIRQ/UTXIRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the URXIEN/UTXIEN = 0, the trigger event URXIRQ/UTXIRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the URXIEN/UTXIEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: UART receive and transmit interrupt request setup.**

```
        B0BSET      FURXIEN     ; Enable UART receive interrupt service
        B0BCLR      FURXIRQ     ; Clear UART receive interrupt request flag

        B0BSET      FUTXIEN     ; Enable UART transmit interrupt service
        B0BCLR      FUTXIRQ     ; Clear UART transmit interrupt request flag
        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: UART receive interrupt service routine.**

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

        B0BTS1      FURXIRQ     ; Check URXIRQ
        JMP         EXIT_INT    ; URXIRQ = 0, exit interrupt vector

        B0BCLR      FURXIRQ     ; Reset URXIRQ
        …                       ; UART receive interrupt service routine
        …
EXIT_INT:
        …

        RETI                    ; Exit interrupt vector. Loading ACC and PFLAG from
                                ; buffers executed by hardware automatically.
```

➢ **Example: UART transmit interrupt service routine.**

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

        B0BTS1      FUTXIRQ     ; Check UTXIRQ
        JMP         EXIT_INT    ; UTXIRQ = 0, exit interrupt vector

        B0BCLR      FUTXIRQ     ; Reset UTXIRQ
        …                       ; UART transmit interrupt service routine
        …
EXIT_INT:
        …

        RETI                    ; Exit interrupt vector. Loading ACC and PFLAG from
                                ; buffers executed by hardware automatically.
```

6.14 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.
CM0IRQ	Comparator 0 output level transition.
CM1IRQ	Comparator 1 output level transition.
CM2IRQ	Comparator 2 output level transition.
T0IRQ	T0C overflow.
TC0IRQ	TC0C overflow.
T1IRQ	T1C overflow.
ADCIRQ	ADC converting end.
T2IRQ	T2C overflow.
UTXIRQ	UART transmit successfully.
URXIRQ	UART receive successfully.

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:
            …
INTP00CHK:                          ; Check INT0 interrupt request
            B0BTS1  FP00IEN         ; Check P00IEN
            JMP     INTT0CHK        ; Jump check to next interrupt
            B0BTS0  FP00IRQ         ; Check P00IRQ
            JMP     INTP00
INTT0CHK:                           ; Check T0 interrupt request
            B0BTS1  FT0IEN          ; Check T0IEN
            JMP     INTTC0CHK       ; Jump check to next interrupt
            B0BTS0  FT0IRQ          ; Check T0IRQ
            JMP     INTT0           ; Jump to T0 interrupt service routine
INTTC0CHK:                          ; Check TC0 interrupt request
            B0BTS1  FTC0IEN         ; Check TC0IEN
            JMP     INTT1CHK        ; Jump check to next interrupt
            B0BTS0  FTC0IRQ         ; Check TC0IRQ
            JMP     INTTC0          ; Jump to TC0 interrupt service routine
INTT1CHK:                           ; Check T1 interrupt request
            B0BTS1  FT1IEN          ; Check T1IEN
            JMP     INTT2CHK        ; Jump check to next interrupt
            B0BTS0  FT1IRQ          ; Check T1IRQ
            JMP     INTT1           ; Jump to T1 interrupt service routine
INTT2CHK:                           ; Check T2 interrupt request
            B0BTS1  FT2IEN          ; Check T2IEN
            JMP     INTCM0CHK       ; Jump check to next interrupt
            B0BTS0  FT2IRQ          ; Check T2IRQ
            JMP     INTT2           ; Jump to T2 interrupt service routine
```

```
INTCM0CHK:                                  ; Check Comparator 0 interrupt request
            B0BTS1      FCM0IEN             ; Check CM0IEN
            JMP         INTCM1CHK           ; Jump check to next interrupt
            B0BTS0      FCM0IRQ             ; Check CM0IRQ
            JMP         INTCM0              ; Jump to CM0 interrupt service routine
INTCM1CHK:                                  ; Check Comparator 1 interrupt request
            B0BTS1      FCM1IEN             ; Check CM1IEN
            JMP         INTCM2CHK           ; Jump check to next interrupt
            B0BTS0      FCM1IRQ             ; Check CM1IRQ
            JMP         INTCM1              ; Jump to CM1 interrupt service routine
INTCM2CHK:                                  ; Check Comparator 0 interrupt request
            B0BTS1      FCM2IEN             ; Check CM0IEN
            JMP         INTCM2CHK           ; Jump check to next interrupt
            B0BTS0      FCM2IRQ             ; Check CM0IRQ
            JMP         INTCM2              ; Jump to CM0 interrupt service routine
INTUTXCHK:                                  ; Check UART UTX interrupt request
            B0BTS1      FUTXIEN             ; Check UTXIEN
            JMP         INTURXCHK           ; Jump check to next interrupt
            B0BTS0      FUTXIRQ             ; Check UTXIRQ
            JMP         INTUTX              ; Jump to UTX interrupt service routine
INTURXCHK:                                  ; Check UART URX interrupt request
            B0BTS1      FURXIEN             ; Check URXIEN
            JMP         INTADCCHK           ; Jump check to next interrupt
            B0BTS0      FURXIRQ             ; Check URXIRQ
            JMP         INTURX              ; Jump to URX 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:
            …

            RETI                            ; Exit interrupt vector. Loading ACC and PFLAG from
                                            ; buffers executed by hardware automatically.
```

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
		BZ	DC	BZEN=1
P0.1	O	PWM0	DC	PWM0OUT=1.
P0.2	I/O	CM0P	AC	CM0EN=1
P0.3	I/O	CM0N	AC	CM0EN=1
P0.4	I/O	CM1N	AC	CM1EN=1
P0.5	I/O	CM2N	AC	CM2EN=1
P4.0	I/O	OPO	AC	OPEN=1
		AIN0	AC	ADENB=1, GCHS=1, CHS[2:0] = 000b
P4.1	I/O	OPN	AC	OPEN=1
		AIN1	AC	ADENB=1, GCHS=1, CHS[2:0] = 001b
P4.2	I/O	OPP	AC	OPEN=1
		AIN2	AC	ADENB=1, GCHS=1, CHS[2:0] = 010b
P4.3	I/O	CM2O	DC	CM2EN=1, CM2OEN = 1
		AIN3	AC	ADENB=1, GCHS=1, CHS[2:0] = 011b
P4.4	I/O	CM1O	DC	CM1EN=1, CM1OEN = 1
		AIN4	AC	ADENB=1, GCHS=1, CHS[2:0] = 100b
P4.5	I/O	CM0O	DC	CM0EN=1, CM0OEN = 1
		AIN5	AC	ADENB=1, GCHS=1, CHS[2:0] = 101b
P4.6	I/O	UTX	DC	UTXEN = 1
		AIN6	AC	ADENB=1, GCHS=1, CHS[2:0] = 110b
P4.7	I/O	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
		URX	DC	URXEN = 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	-	-	P05M	P04M	P03M	P02M	-	P00M
Read/Write	-	-	R/W	R/W	R/W	R/W	-	R/W
After reset	-	-	0	0	0	0	-	0

0C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

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

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

Example: I/O mode selecting

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

MOV     A, #03DH    ; Set all pins to be output mode.
B0MOV   P0M, A
MOV     A, #0FFH    ; Set all pins to be output mode.
B0MOV   P4M,A

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

B0BSET  P4M.0       ; Set P4.0 to be output mode.
```

7.3 I/O PULL UP REGISTER

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

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	-	P05R	P04R	P03R	P02R	-	P00R
Read/Write	-	-	W	W	W	W	-	W
After reset	-	-	0	0	0	0	-	0

0E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	-	P46R	P45R	P44R	P43R	P42R	P41R	P40R
Read/Write	-	W	W	W	W	W	W	W
After reset	-	0	0	0	0	0	0	0

✸ ***Note: P0.1 pin is output only. P4.7 pin is without pull-up resister. The P0UR.1/P4UR.7 is undefined.***

Example: I/O Pull up Register

```
MOV     A, #03DH        ; Enable Port 0 Pull-up register,
B0MOV   P0UR, A         ;
MOV     A, #07FH        ; Enable Port 4 Pull-up register,
B0MOV   P4UR,A
```

7.4 I/O PORT DATA REGISTER

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

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	P47	P46	P45	P44	P43	P42	P41	P40
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

✸ ***Note:***
The P4.7 keeps "1" when external reset enable by code option.

Example: Read data from input port.

```
B0MOV   A, P0       ; Read data from Port 0
B0MOV   A, P4       ; Read data from Port 4
```

Example: Write data to output port.

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

Example: Write one bit data to output port.

```
B0BSET  P4.0        ; Set P4.0 to be "1".
B0BCLR  P4.0        ; Set P4.0 to be "0".
```

7.5 PORT 0 COMPARATOR SHARE PIN

The P0.2~P0.5 is shared with comparator input function and Schmitt trigger structure. Connect an analog signal to 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 0 will encounter above current leakage situation. CMCON is Port0 Configuration register. Write “1” into CMCON.n will configure related port 0 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
CMCON	-	-	-	-	CM2NCON	CM1NCON	CM0NCON	CM0PCON
Read/Write	-	-	-	-	W	W	W	W
After reset	-	-	-	-	0	0	0	0

Bit3 **CM2NCON:** P0.5 configuration control bits.
0 = P0.5 can be an analog input (Comparator 2 negative input pin) or digital I/O pins.
1 = P0.5 is pure analog input, can’t be a digital I/O pin.
Bit2 **CM1NCON:** P0.4 configuration control bits.
0 = P0.4 can be an analog input (Comparator 1 negative input pin) or digital I/O pins.
1 = P0.4 is pure analog input, can’t be a digital I/O pin.
Bit1 **CM0NCON:** P0.3 configuration control bits.
0 = P0.3 can be an analog input (Comparator 0 negative input pin) or digital I/O pins.
1 = P0.3 is pure analog input, can’t be a digital I/O pin.
Bit0 **CM0PCON:** P0.2 configuration control bits.
0 = P0.2 can be an analog input (Comparator 0 positive input pin) or digital I/O pins.
1 = P0.2 is pure analog input, can’t be a digital I/O pin.

● ***Note: When Port 0.n is general I/O port not comparator input pin, CMCON.n must set to “0” or the Port 0.n digital I/O signal would be isolated. P0.2~P0.5 Connect an analog signal to digital input pin, especially the analog signal level is about 1/2 VDD will cause extra current leakage.***

When CM2NS = 1, CM2N pin is GPIO function and comparator 0 positive (V+) terminal is connected to comparator 2 negative terminal (internal connect). So user has to clear CM2NCON (digital I/O pin) and CM0PCON set to “1” (analog signal input pin).

09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMDB2	-	-	PWS1	PWS0	CM2REF	CM2NS	CM2D1	CM2D0
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit2 **CM2NS: comparator** 2 negative configuration control bits.
0 = CM2N pin is comparator 2 negative input pin, and GPIO function is isolated.
1 = CM2N pin is GPIO function and comparator 0 positive (V+) terminal is connected to comparator 2 negative terminal (internal connect).

✷ ***Note:***
1. ***When Port 0.5 is general I/O port not comparator 2 negative input pin, CM2NCON must set to “0” or the Port 0.5 digital I/O signal would be isolated.***
2. ***When Port 0.2 is general I/O port not comparator 0 negative input pin, CM0PCON must set to “0” or the Port 0.2 digital I/O signal would be isolated.***
3. ***When CM2NS = 1, CM2NCON must set to “0”, CM0PCON must set to “1”, Port 0.5 is general I/O port and not comparator 2 negative input pin.***

➢ Example: Set P0.2~P0.3 to be Comparator 0 function. CM0EN and CMCON.0~CMCON.1 bits must be set as “1”.

```
; Set P0.2~P0.3 are input floating.
        B0BCLR      FP0M.2          ; clear P0.2 input mode.
        B0BCLR      FP0M.3          ; clear P0.3 input mode.
        MOV         A,#nnnn00nnb    ; Disable P0.2/P0.3 pull-up resistor.
        MOV         P0UR,A

; Set CMCON.
        MOV         A,#0000nn11b    ; Disable P0.2~P0.3 digital function.
        MOV         CMCON,A

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

➢ **Example: Set P0.4 to be Comparator 1 function. CM1EN and CMCON.2 bit must be set as “1”.**

```
; Set P0.4 is input floating.
        B0BCLR      FP0M.4          ; clear P0.4 input mode.
        MOV         A,#nnn0nnnnb    ; Disable P0.4 pull-up resistor.
        MOV         P0UR,A

; Set CMCON.
        MOV         A,#0000n1nnb    ; Disable P0.4 digital function.
        MOV         CMCON,A

; Enable Comparator 1 function.
        B0BSET      FCM1EN          ; Enable comparator 1 function.
```

➢ **Example: Set P0.5 to be Comparator 2 function. CM2EN and CMCON.3 bit must be set as “1”.**

```
; Set P0.5 is input floating.
        B0BCLR      FP0M.5          ; clear P0.5 input mode.
        MOV         A,#nn0nnnnnb    ; Disable P0.5 pull-up resistor.
        MOV         P0UR,A

; Set CMCON.
        MOV         A,#00001nnnb    ; Disable P0.5 digital function.
        MOV         CMCON,A

; Enable Comparator 2 function.
        B0BSET      FCM2EN          ; Enable comparator 2 function.
```

7.6 PORT 4 ADC SHARE PIN

The Port 4 is shared with ADC input function, ADC function and Schmitt trigger structure. Only one pin of port 4 can be configured as ADC input in the same time by ADM register. The other pins of port 4 are digital I/O pins. Connect an analog signal to COMS digital input pin, especially the analog signal level is about 1/2 VDD will cause extra current leakage. In the power down mode, the above leakage current will be a big problem. Unfortunately, if users connect more than one analog input signal to port 4 will encounter above current leakage situation. P4CON is Port4 Configuration register. Write "1" into P4CON.n will configure related port 4 pin as pure analog input pin to avoid current leakage.

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

Bit[6:0] **P4CON[6:0]:** P4.n configuration control bits.
0 = P4.n can be an analog input (ADC input) or digital I/O pins.
1 = P4.n is pure analog input, can't be a digital I/O pin.

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

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

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	ADREF	CHS2	CHS1	CHS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

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

Bit[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 = OP-Amp output terminal.

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

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

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

; Clear P4CON.
        MOV         A,#0nnnnn0nb
        MOV         P4CON,A     ; Enable P4.1 digital function.

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

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

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

; Clear P4CON.
        MOV         A,#0nnnnn0nb ; Enable P4.1 digital function.
        MOV         P4CON,A

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

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

7.7 PORT 4 OP-AMP SHARE PIN

The P4.0~P4.2 is shared with ADC input function, OP-Amp function and Schmitt trigger structure. Connect an analog signal to 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, users connect analog signals to P4.0~P4.2 will encounter above current leakage situation. P4CON is Port4 Configuration register. Write "1" into P4CON.0~P4COM.2 will configure related pin as pure analog input pin to avoid current leakage.

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

Bit[2:0] **P4CON[2:0]:** P4.2~P4.0 configuration control bits.
0 = P4.0~P4.2 can be an analog pins (OP-AMP pins) or digital I/O pins.
1 = P4.0~P4.2 is pure analog pins, can't be a digital I/O pin.

OP-AMP function is controlled by OPEN and OPPSW bits. If OPEN = 0, P4.0~P4.2 are general purpose bi-direction I/O port or ADC analog input. If OPEN = 1 and ADENB = 0, P4.0~P4.2 are OP-AMP analog pins. If OPEN = OPPSW = 1 and ADENB = 0, P4.2 is digital I/O pin, P4.0/P4.1 are OP-AMP analog pins.

09FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPM	-	-	-	-	-	-	OPPSW	OPEN
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

Bit 1 **OPPSW:** OP-AMP positive input source select bit.
0 = OPP pin (P4.2) is OP-AMP positive input pin.
1 = OPP pin (P4.2) is digital I/O or AIN2 input pin, OP-AMP positive terminal connect to internal ground.
Bit 0 **OPEN:** OP-AMP enable bit.
0 = OP-AMP function disable. OPO/OPP/OPN pin (P4.0~P4.2) are digital I/O.
1 = OP-AMP function enable. OPO/OPP/OPN pin (P4.0~P4.2) are OP-AMP input and output pins.

➢ **Example: Set P4.0~P4.2 to be OP-AMP function. OPEN and P4CON.0~P4CON.2 bits must be set as "1".**

```
; Clear GCHS status.
        B0BCLR      FGCHS       ; If CHS[2:0] don't point to P4.0~P4.2 (CHS[2:0] ≠
                                ; 000B~010B), don't care GCHS status.

; Disable P4.0~P4.2 Pull-up resistor.
        MOV         A,#0nnnn000b    ; Disable P4.0~P4.2 pull-up resistor.
        MOV         P4UR,A

; Set P4.0~P4.2 input mode.
        MOV         A,#0nnnn000b    ; Set P4.0~P4.2 input mode.
        MOV         P4M,A

; Set P4CON.
        MOV         A,#0nnnn111b    ; Disable P4.0~P4.2 digital function.
        MOV         P4CON,A

; Enable OP-AMP function.
        B0BSET      FOPEN       ; Enable OP-AMP function.
```

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.**

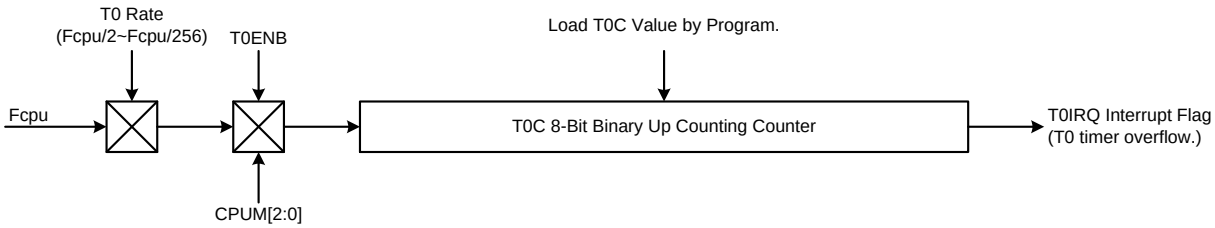
```
Main:
            …                           ; Check I/O.
            …                           ; Check RAM
Err:        JMP $                       ; I/O or RAM error. Program jump here and don't
                                        ; clear watchdog. Wait watchdog timer overflow to reset IC.

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

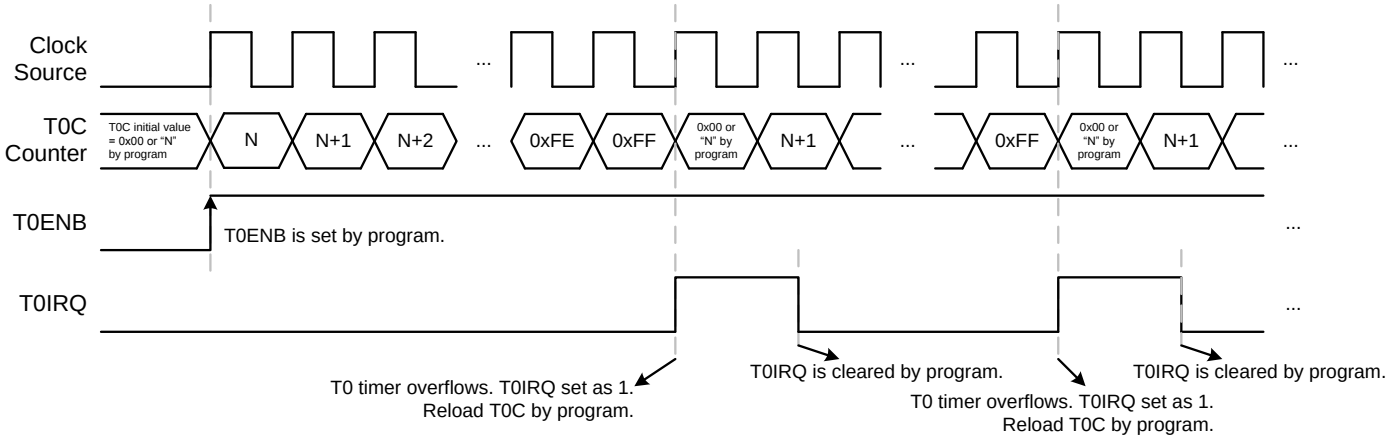
The T0 timer is an 8-bit binary up timer with basic timer function. The basic timer function supports flag indicator (T0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T0M, T0C registers. The T0 builds in green mode wake-up function (CPUM[2:0] = 011b). When 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=16MHz, Fcpu=Fhosc/4		Fhosc=16MHz, Fcpu=Fhosc/16	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)
000b	Fcpu/256	16.384	64	65.536	256
001b	Fcpu/128	8.192	32	32.768	128
010b	Fcpu/64	4.096	16	16.384	64
011b	Fcpu/32	2.048	8	8.192	32
100b	Fcpu/16	1.024	4	4.096	16
101b	Fcpu/8	0.512	2	2.048	8
110b	Fcpu/4	0.256	1	1.024	4
111b	Fcpu/2	0.128	0.5	0.512	2

8.2.3 T0M MODE REGISTER

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

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 = 16MHz/16 = 1MHz. Select T0RATE=001 (Fcpu/128).**

  T0 interval time = 10ms. T0 clock rate = 16MHz/16/128

  *T0C initial value = 256 - (T0 interval time * input clock)*
  *= 256 - (10ms * 16MHz / 16 / 128)*
  *= 256 - (10^{-2} * 16MHz / 16 / 128)*
  = B2H

8.2.5 T0 TIMER OPERATION EXPLAME

➢ **T0 TIMER CONFIGURATION:**

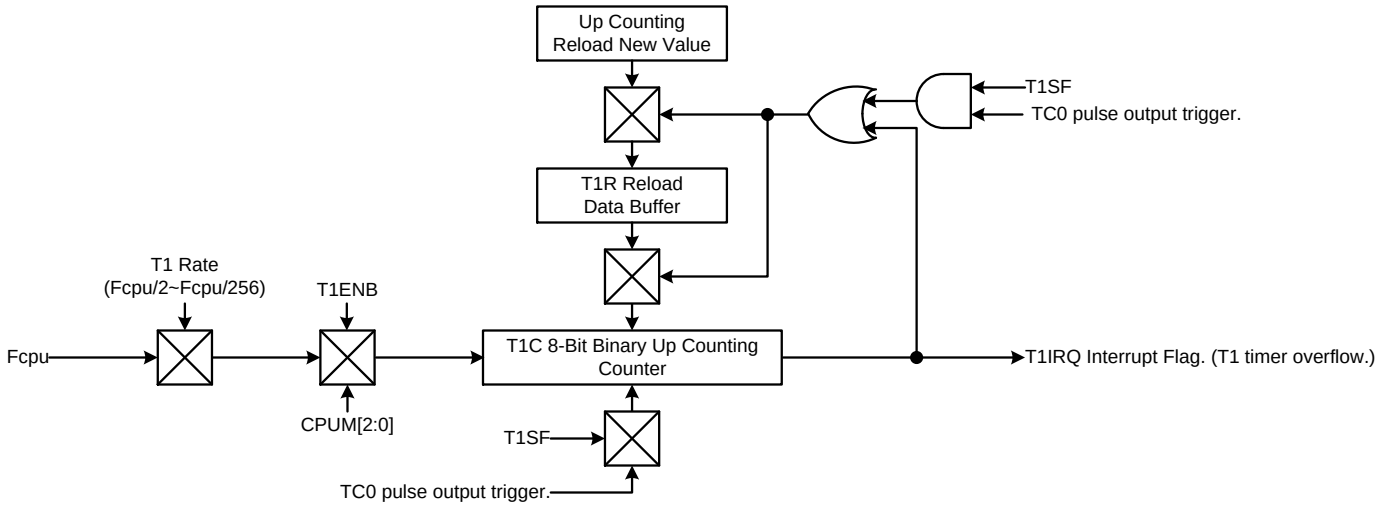
```
; Reset T0 timer.
        CLR     T0M             ; Clear T0M register.

; 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 T1 8-BIT TIMER

8.3.1 OVERVIEW

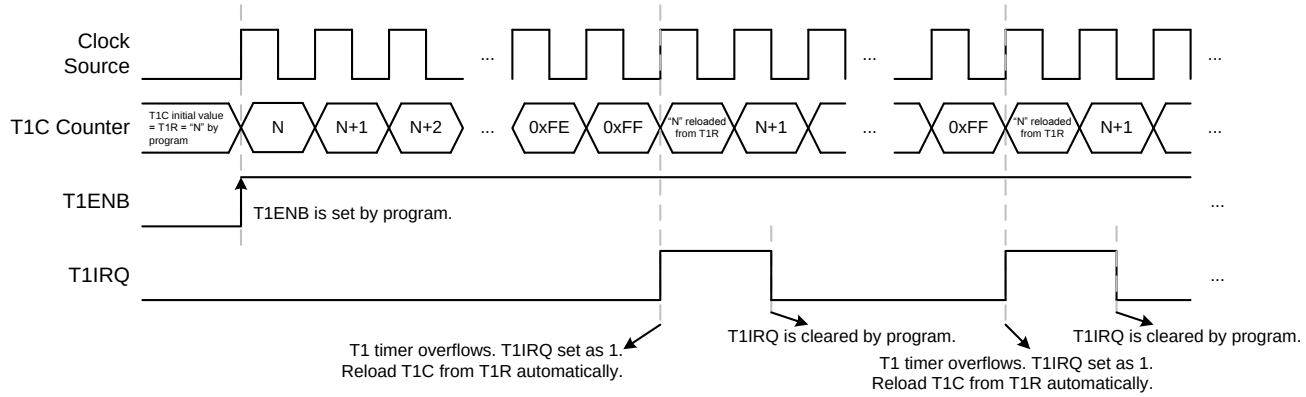
The T1 timer is an 8-bit binary up timer with basic timer function (T1SF = 0) and TC0 pulse output trigger mode (T1SF = 1). The basic timer function supports flag indicator (T1IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T1M, T1C, T1R registers. When T1 timer's TC0 pulse output trigger mode is enabled (T1SF = 1), T1 timer starting to count trigger source is TC0 pulse output signal. T1 timer supports auto-reload function which always enabled. The T1 doesn't build in green mode wake-up function. When T1 timer overflow occurs under green mode, the T1IRQ will be set as "1". The main purposes of the T1 timer are as following.

- ☞ **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- ☞ **Interrupt function:** T1 timer function supports interrupt function. When T1 timer occurs overflow, the T1IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- ☞ **TC0 pulse output trigger mode**: T1 timer with TC0 pulse output trigger mode is controlled by T1SF bit set as "1". When T1 timer's TC0 pulse output trigger mode is enabled (T1SF = 1) and TC0 pulse trigger is issued, T1 timer starts to count.
- ☞ **Green mode function:** All T1 functions (basic timer, TC0 pulse trigger mode) keeps running in green mode, but no wake-up function. Timer IRQ actives as any IRQ trigger occurrence, e.g. timer overflow.



8.3.2 T1 TIMER OPERATION

T1 timer is controlled by T1ENB bit. When T1ENB=0, T1 timer stops. When T1ENB=1, T1 timer starts to count. Before enabling T1 timer, setup T1 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…T1C increases "1" by timer clock source. When T1 overflow event occurs, T1IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is T1C count from full scale (0xFF) to zero scale (0x00). In difference function modes, T1C value relates to operation. If T1C value changing effects operation, the transition of operations would make timer function error. So T1 builds in double buffer to avoid these situations happen. The double buffer concept is to flash T1C during T1 counting, to set the new value to T1R (reload buffer), and the new value will be loaded from T1R to T1C after T1 overflow occurrence automatically. In the next cycle, the T1 timer runs under new conditions, and no any transitions occur. The auto-reload function is no any control interface and always actives as T1 enables. If T1 timer interrupt function is enabled (T1IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 0008H) and executes interrupt service routine after T1 overflow occurrence. Clear T1IRQ by program is necessary in interrupt procedure. T1 timer can works in normal mode, slow mode and green mode. Under green mode, T1 keep counting, set T1IRQ and can't wake-up system.



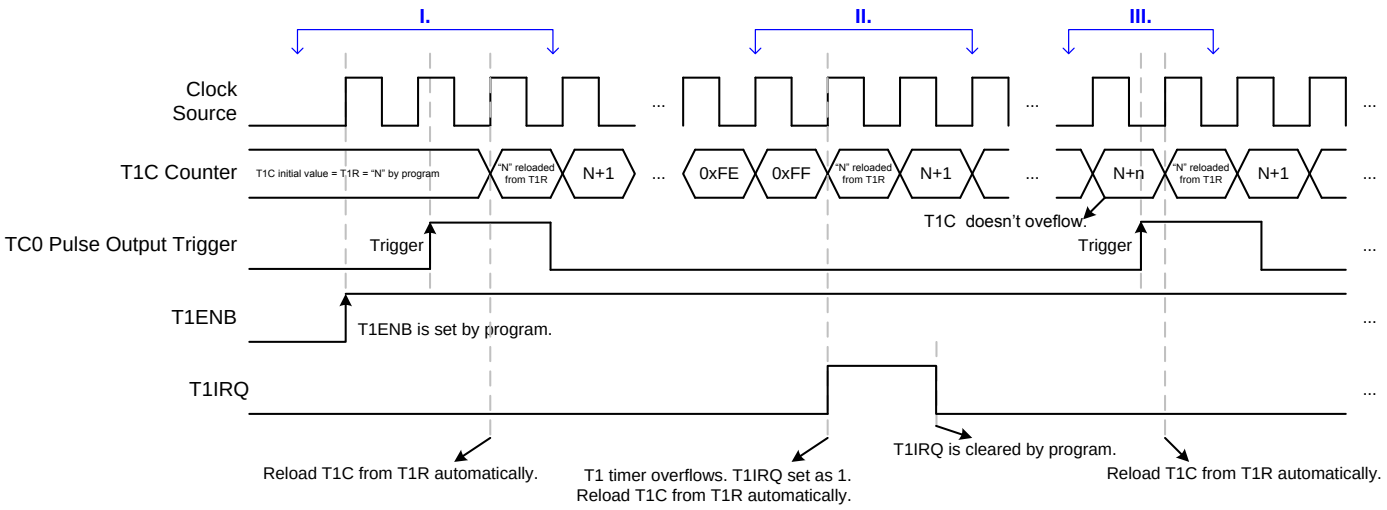
T1 clock source is Fcpu (instruction cycle) through T1rate [2:0] pre-scaler to decide Fcpu/2~Fcpu/256. T1 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

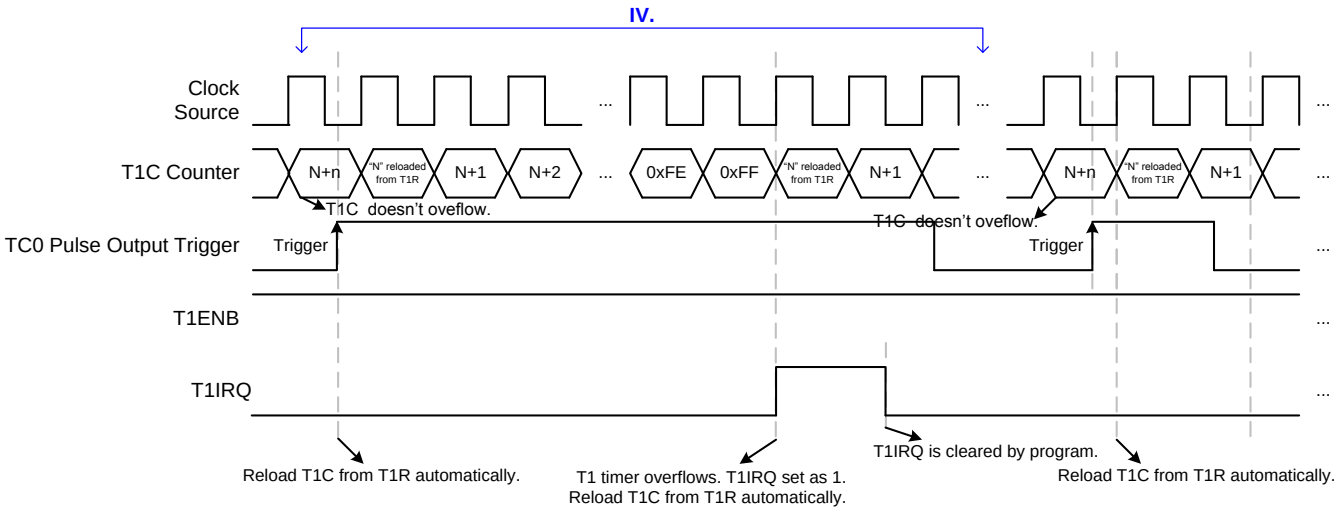
T1rate[2:0]	T1 Clock	T1 Interval Time			
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=16MHz, Fcpu=Fhosc/16	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)
000b	Fcpu/256	16.384	64	65.536	256
001b	Fcpu/128	8.192	32	32.768	128
010b	Fcpu/64	4.096	16	16.384	64
011b	Fcpu/32	2.048	8	8.192	32
100b	Fcpu/16	1.024	4	4.096	16
101b	Fcpu/8	0.512	2	2.048	8
110b	Fcpu/4	0.256	1	1.024	4
111b	Fcpu/2	0.128	0.5	0.512	2

8.3.3 T1 TIMER'S TC0 PULSE OUTPUT TRIGGER MODE

T1 timer builds in TC0 pulse trigger mode controlled by T1SF bit set as "1". When T1 timer's TC0 pulse output trigger mode is enabled (T1SF = 1) and TC0 pulse trigger is issued, T1 timer starts to count. If T1 timer doesn't overflow, and a new TC0 pulse output trigger is issued, T1C will be reloaded from T1R automatically and count again. If T1 timer overflows (from 0xFF to 0x00), T1C will be reloaded from T1R automatically, continue counting, and issue a time-out signal to indicate time out event (T1IRQ = 1).

- **T1 timer's TC0 pulse trigger mode timing diagram:**





i. After T1ENB is enabled, T1 timer doesn't count and wait the TC0 pulse trigger signal. When TC0 pulse start to output, the trigger signal will trigger T1 timer to count.
ii. When T1 timer overflows, and no TC0 pulse output trigger signal, the T1C is reloaded from T1R automatically and issues T1IRQ time-out flag.
iii. If T1 timer doesn't overflow, but TC0 pulse output trigger is issued, the T1C is reloaded from T1R automatically, but not issue T1IRQ time-out flag.
iv. If T1 timer overflows, but TC0 pulse output keeps output status, the T1C is reloaded from T1R automatically and issues T1IRQ time-out flag.

✸ ***Note:***
1. ***If TC0 pulse generator trigger source (CM0SF = 0) is PWM0OUT, system supports T1 timer with TC0 pulse trigger mode (T1SF = 1).***
2. ***If TC0 pulse generator trigger source (CM0SF = 1) is comparator 0, system doesn't support T1 timer with TC0 pulse trigger mode (T1SF = 1).***

8.3.4 T1M MODE REGISTER

T1M is T1 timer mode control register to configure T1 operating mode including T1 pre-scalar and T1 timer with TC0 pulse trigger mode. These configurations must be setup completely before enabling T1 timer.

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1rate2	T1rate1	T1rate0	-	-	-	T1SF
Read/Write	R/W	R/W	R/W	R/W	-	-	-	R/W
After reset	0	0	0	0	-	-	-	0

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

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

Bit 0 **T1SF:** T1 timer trigger function control bit.
0 = Disable (T1 is normal timer function).
1 = Enable (T1 timer with TC0 pulse trigger mode).

8.3.5 T1C COUNTING REGISTER

T1C is T1 8-bit counter. When T1C overflow occurs, the T1IRQ flag is set as "1" and cleared by program. The T1C decides T1 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T1C register and T1R register first time, and then enable T1 timer to make sure the fist cycle correct. After one T1 overflow occurs, the T1C register is loaded a correct value from T1R register automatically, not program.

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1C	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of T1C initial value is as following.

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

8.3.6 T1R AUTO-RELOAD REGISTER

T1 timer builds in auto-reload function, and T1R register stores reload data. When T1C overflow occurs, T1C register is loaded data from T1R register automatically. Under T1 timer counting status, to modify T1 interval time is to modify T1R register, not T1C register. New T1C data of T1 interval time will be updated after T1 timer overflow occurrence, T1R loads new value to T1C register. But at the first time to setup T1M, T1C and T1R must be set the same value before enabling T1 timer. T1 is double buffer design. If new T1R value is set by program, the new value is stored in 1st buffer. Until T1 overflow occurs, the new value moves to real T1R buffer. This way can avoid any transitional condition to affect the correctness of T1 interval time.

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1R	T1R7	T1R6	T1R5	T1R4	T1R3	T1R2	T1R1	T1R0
Read/Write	W	W	W	W	W	W	W	W
After Reset	0	0	0	0	0	0	0	0

The equation of T1R initial value is as following.

T1R initial value = 256 - (T1 interrupt interval time * T1 clock rate)

- **Example: To calculation T1C and T1R value to obtain 100us T1 interval time. T1 clock source is Fhosc = 16MHz. Select T1RATE = 100b (Fcpu/16).**

  T1 interval time = 100us. T1 clock rate = 16MHz/16

  *T1C/T1R initial value = 256 - (T1 interval time * input clock)*
  *= 256 - (100us * 16MHz / 16)*
  *= 256 - (100 * 10^{-6} * 16 * 10^{6} / 16)*
  = 9CH

8.3.7 T1 TIMER OPERATION EXPLAME

➢ **T1 TIMER CONFIGURATION:**

```
; Reset T1 timer.
        CLR         T1M             ; Clear T1M register.

; Set T1 clock source and T1 rate.
        MOV         A, #0nnn0000b
        B0MOV       T1M, A

; Set T1C and T1R register for T1 Interval time.
        MOV         A, #value       ; T1C must be equal to T1R.
        B0MOV       T1C, A
        B0MOV       T1R, A

; Clear T1IRQ
        B0BCLR      FT1IRQ

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

➢ **T1 TIMER WITH TC0 PULSE TRIGGER MODE:**

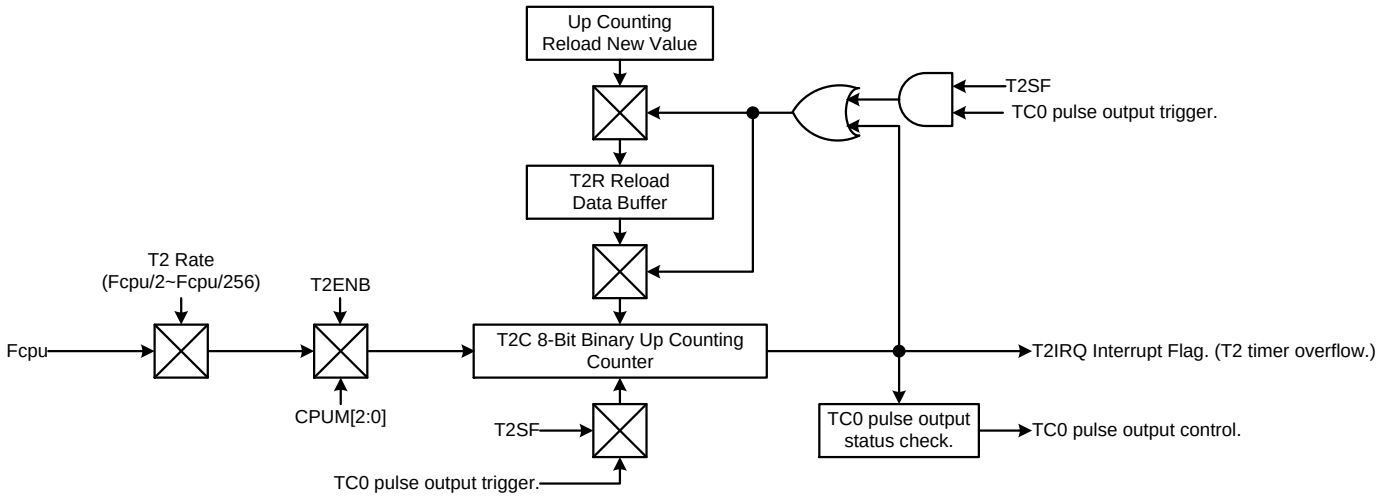
```
; Reset T1 timer.
        CLR         T1M             ; Clear T1M register.

; Set T1 clock source and T1 rate.
        MOV         A, #0nnn0000b
        B0MOV       T1M, A

; Set T1 timer with TC0 pulse trigger mode enable.
        B0BSET      FT1SF           ; Enable T1 timer with TC0 pulse trigger mode enable.

; Set T1C and T1R register for T1 Interval time.
        MOV         A, #value       ; T1C must be equal to T1R.
        B0MOV       T1C, A
        B0MOV       T1R, A

; Clear T1IRQ
        B0BCLR      FT1IRQ

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

8.4 T2 8-BIT TIMER

8.4.1 OVERVIEW

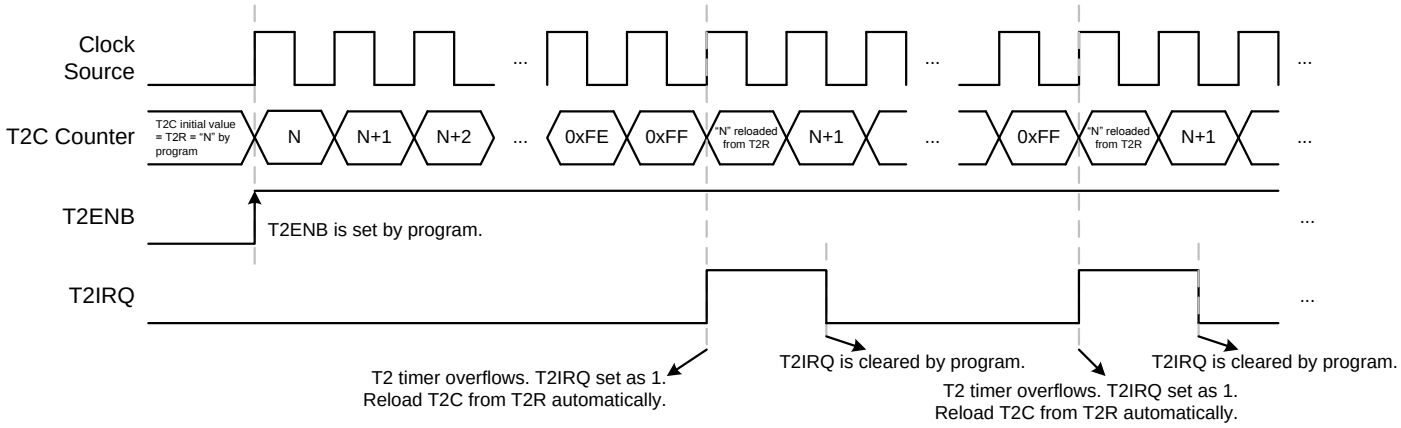
The T2 timer is an 8-bit binary up timer with basic timer function (T2SF = 0) and TC0 pulse output trigger mode (T2SF = 1). The basic timer function supports flag indicator (T2IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T2M, T2C, T2R registers. When T2 timer's TC0 pulse output trigger mode is enabled (T2SF = 1), T2 timer starting to count trigger source is TC0 pulse output signal. T2 timer supports auto-reload function which always enabled. The T2 doesn't build in green mode wake-up function. When T2 timer overflow occurs under green mode, the T2IRQ will be set as "1". The main purposes of the T2 timer are as following.

- ☞ **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- ☞ **Interrupt function:** T2 timer function supports interrupt function. When T2 timer occurs overflow, the T2IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- ☞ **TC0 pulse output trigger mode**: T2 timer with TC0 pulse output trigger mode is controlled by T2SF bit set as "1". When T2 timer's TC0 pulse output trigger mode is enabled (T2SF = 1) and TC0 pulse trigger is issued, T2 timer starts to count. When T2 timer overflows, TC0 pulse output status will be checked and controlled.
- ☞ **Green mode function:** All T2 functions (basic timer, TC0 pulse trigger mode) keeps running in green mode, but no wake-up function. Timer IRQ actives as any IRQ trigger occurrence, e.g. timer overflow.



8.4.2 T2 TIMER OPERATION

T2 timer is controlled by T2ENB bit. When T2ENB=0, T2 timer stops. When T2ENB=1, T2 timer starts to count. Before enabling T2 timer, setup T2 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…T2C increases "1" by timer clock source. When T2 overflow event occurs, T2IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is T2C count from full scale (0xFF) to zero scale (0x00). In difference function modes, T2C value relates to operation. If T2C value changing effects operation, the transition of operations would make timer function error. So T2 builds in double buffer to avoid these situations happen. The double buffer concept is to flash T2C during T2 counting, to set the new value to T2R (reload buffer), and the new value will be loaded from T2R to T2C after T2 overflow occurrence automatically. In the next cycle, the T2 timer runs under new conditions, and no any transitions occur. The auto-reload function is no any control interface and always actives as T2 enables. If T2 timer interrupt function is enabled (T2IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 0008H) and executes interrupt service routine after T2 overflow occurrence. Clear T2IRQ by program is necessary in interrupt procedure. T2 timer can works in normal mode, slow mode and green mode. Under green mode, T2 keep counting, set T2IRQ and can't wake-up system.



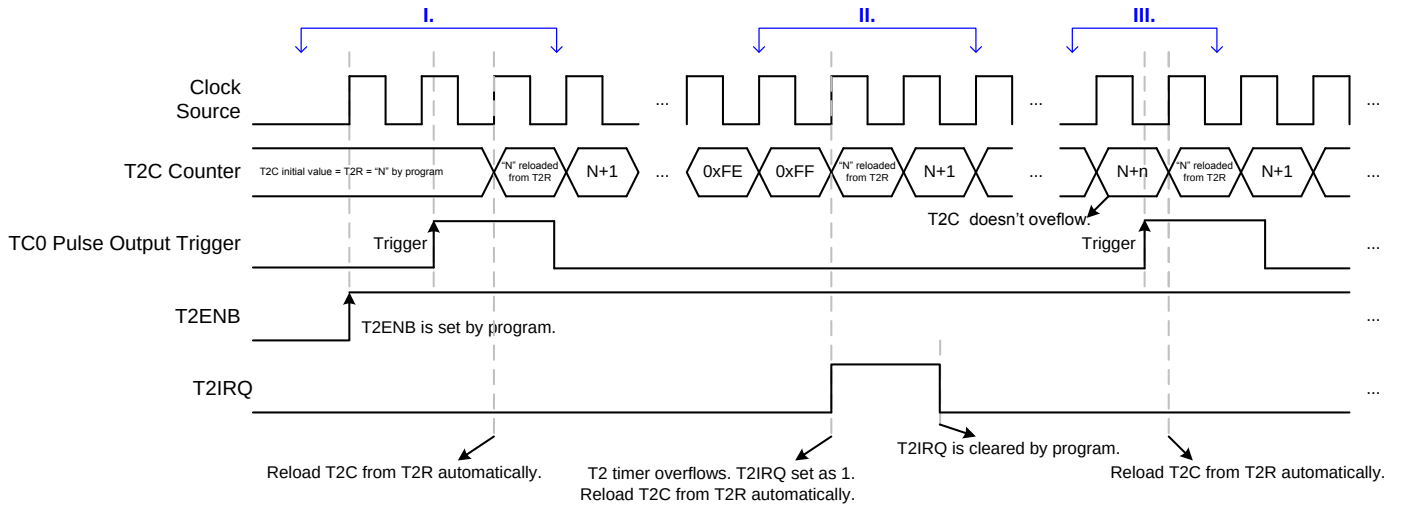
T2 clock source is Fcpu (instruction cycle) through T2rate [2:0] pre-scaler to decide Fcpu/2~Fcpu/256. T2 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

T2rate[2:0]	T2 Clock	T2 Interval Time			
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=16MHz, Fcpu=Fhosc/16	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)
000b	Fcpu/256	16.384	64	65.536	256
001b	Fcpu/128	8.192	32	32.768	128
010b	Fcpu/64	4.096	16	16.384	64
011b	Fcpu/32	2.048	8	8.192	32
100b	Fcpu/16	1.024	4	4.096	16
101b	Fcpu/8	0.512	2	2.048	8
110b	Fcpu/4	0.256	1	1.024	4
111b	Fcpu/2	0.128	0.5	0.512	2

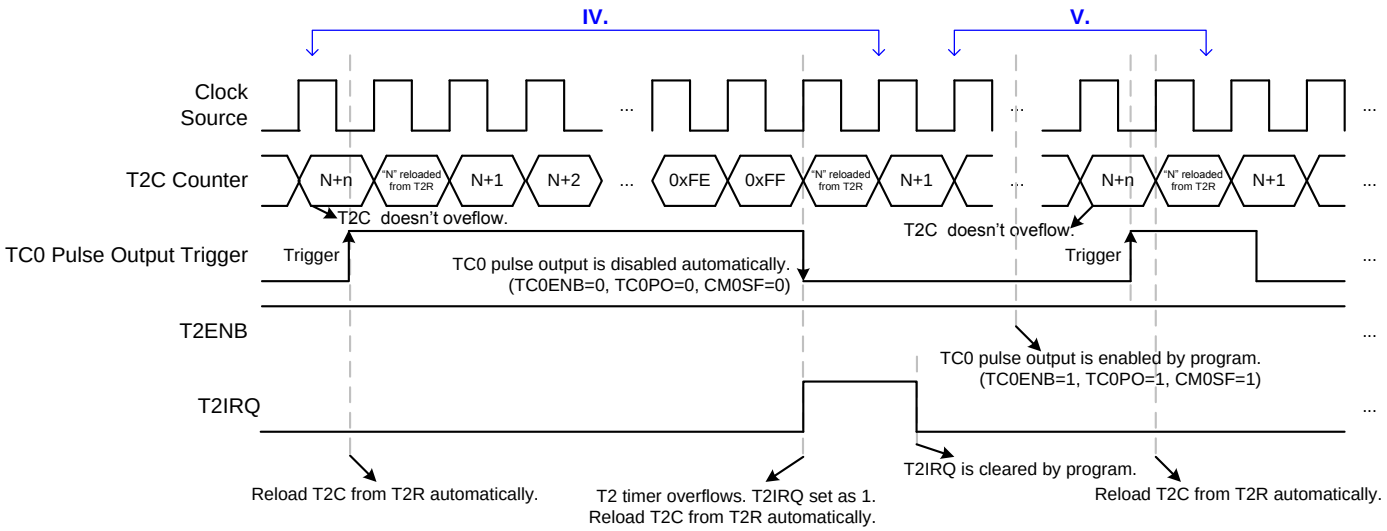
8.4.3 T2 TIMER'S TC0 PULSE OUTPUT TRIGGER MODE

T2 timer builds in TC0 pulse trigger mode controlled by T2SF bit set as "1". When T2 timer's TC0 pulse output trigger mode is enabled (T2SF = 1) and TC0 pulse trigger is issued, T2 timer starts to count. If T2 timer doesn't overflow, and then a new TC0 pulse output trigger is issued, T2C will be reloaded from T2R automatically and count again. If T2 timer overflows (from 0xFF to 0x00), and TC0 pulse output finished, T2C will be reloaded from T2R automatically, continue counting, and issue a time out signal (T2IRQ = 1). If T2 timer overflows, but TC0 pulse output doesn't finish, TC0 pulse output is disabled immediately (TC0PO/TC0ENB/CM0SF bits are cleared, and pulse output pin exchanges to GPIO mode.), and then T2C will be reloaded from T2R automatically, continue counting, and issue a time out signal (T2IRQ = 1). This is an abnormal condition, and the TC0 pulse must be enabled by program again.

- **T2 timer's TC0 pulse trigger mode timing diagram:**



- **The T2 timer overflow (from 0xFF to 0x00), hardware check TC0 pulse generator still output pulse.**



i. After T2ENB is enabled, T2 timer doesn't count and wait the TC0 pulse trigger signal. When TC0 pulse start to output, the trigger signal will trigger T2 timer to count.
ii. When T2 timer overflows, and no TC0 pulse output trigger signal, the T2C is reloaded from T2R automatically and issues T2IRQ time-out flag.
iii. If T2 timer doesn't overflow, but TC0 pulse output trigger is issued, the T2C is reloaded from T2R automatically, but not issue T2IRQ time-out flag.
iv. If T2 timer overflows, but TC0 pulse output doesn't finish, TC0 pulse output is disabled immediately (TC0PO/TC0ENB/CM0SF bits are cleared, and pulse output pin exchanges to GPIO mode.), and then T2C will be reloaded from T2R automatically, continue counting, and issue a time out signal (T2IRQ = 1).
v. TC0 pulse output function is enabled by program, and T2 timer works in TC0 pulse output trigger mode.

✸ ***Note:***
1. ***If TC0 pulse generator trigger source (CM0SF = 0) is PWM0OUT, system supports T2 timer with TC0 pulse trigger mode (T2SF = 1).***
2. ***If TC0 pulse generator trigger source (CM0SF = 1) is comparator 0, system doesn't support T2 timer with TC0 pulse trigger mode (T2SF = 1).***

8.4.4 T2M MODE REGISTER

T2M is T2 timer mode control register to configure T2 operating mode including T2 pre-scalar and T2 timer with TC0 pulse trigger mode. These configurations must be setup completely before enabling T2 timer

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2M	T2ENB	T2rate2	T2rate1	T2rate0	-	-	-	T2SF
Read/Write	R/W	R/W	R/W	R/W	-	-	-	R/W
After reset	0	0	0	0	-	-	-	0

Bit 7 **T2ENB:** T2 timer control bit.
0 = Disable T2 timer.
1 = Enable T2 timer.

Bit [6:4] **T2RATE[2:0]:** T2 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 0 **T2SF:** T2 timer trigger function control bit.
0 = Disable (T2 is normal timer function).
1 = Enable (T2 timer with TC0 pulse trigger mode).

8.4.5 T2C COUNTING REGISTER

T2C is T2 8-bit counter. When T2C overflow occurs, the T2IRQ flag is set as "1" and cleared by program. The T2C decides T2 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T2C register and T2R register first time, and then enable T2 timer to make sure the fist cycle correct. After one T2 overflow occurs, the T2C register is loaded a correct value from T2R register automatically, not program.

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2C	T2C7	T2C6	T2C5	T2C4	T2C3	T2C2	T2C1	T2C0
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 T2C initial value is as following.

T2C initial value = 256 - (T2 interrupt interval time * T2 clock rate)

8.4.6 T2R AUTO-RELOAD REGISTER

T2 timer builds in auto-reload function, and T2R register stores reload data. When T2C overflow occurs, T2C register is loaded data from T2R register automatically. Under T2 timer counting status, to modify T2 interval time is to modify T2R register, not T2C register. New T2C data of T2 interval time will be updated after T2 timer overflow occurrence, T2R loads new value to T2C register. But at the first time to setup T2M, T2C and T2R must be set the same value before enabling T2 timer. T2 is double buffer design. If new T2R value is set by program, the new value is stored in 1st buffer. Until T2 overflow occurs, the new value moves to real T2R buffer. This way can avoid any transitional condition to affect the correctness of T2 interval time.

0A5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2R	T2R7	T2R6	T2R5	T2R4	T2R3	T2R2	T2R1	T2R0
Read/Write	W	W	W	W	W	W	W	W
After Reset	0	0	0	0	0	0	0	0

The equation of T2R initial value is as following.

T2R initial value = 256 - (T2 interrupt interval time * T2 clock rate)

- **Example: To calculation T2C and T2R value to obtain 100us T2 interval time. T2 clock source is Fhosc = 16MHz. Select T2RATE = 100b (Fcpu/16).**

  T2 interval time = 100us. T2 clock rate = 16MHz/16

  *T2C/T2R initial value = 256 - (T2 interval time * input clock)*
  *= 256 - (100us * 16MHz / 16)*
  *= 256 - (100 * 10^{-6} * 16 * 10^{6} / 16)*
  = 9CH

8.4.7 T2 TIMER OPERATION EXPLAME

➢ **T2 TIMER CONFIGURATION:**

```
; Reset T2 timer.
        CLR     T2M             ; Clear T2M register.

; Set T2 clock source and T2 rate.
        MOV     A, #0nnn0000b
        B0MOV   T2M, A

; Set T2C and T2R register for T2 Interval time.
        MOV     A, #value       ; T2C must be equal to T2R.
        B0MOV   T2C, A
        B0MOV   T2R, A

; Clear T2IRQ
        B0BCLR  FT2IRQ

; Enable T2 timer and interrupt function.
        B0BSET  FT2IEN          ; Enable T2 interrupt function.
        B0BSET  FT2ENB          ; Enable T2 timer.
```

➢ **T2 TIMER WITH TC0 PULSE TRIGGER MODE:**

```
; Reset T2 timer.
        CLR     T2M             ; Clear T2M register.

; Set T2 clock source and T2 rate.
        MOV     A, #0nnn0000b
        B0MOV   T2M, A

; Set T2 timer with TC0 pulse trigger mode enable.
        B0BSET  FT2SF           ; Enable T2 timer with TC0 pulse trigger mode enable.

; Set T2C and T2R register for T2 Interval time.
        MOV     A, #value       ; T2C must be equal to T2R.
        B0MOV   T2C, A
        B0MOV   T2R, A

; Clear T2IRQ
        B0BCLR  FT2IRQ

; Enable T2 timer and interrupt function.
        B0BSET  FT2IEN          ; Enable T2 interrupt function.
        B0BSET  FT2ENB          ; Enable T2 timer.
```

➢ **T2 TIMER WITH TC0 PULSE TRIGGER MODE:**

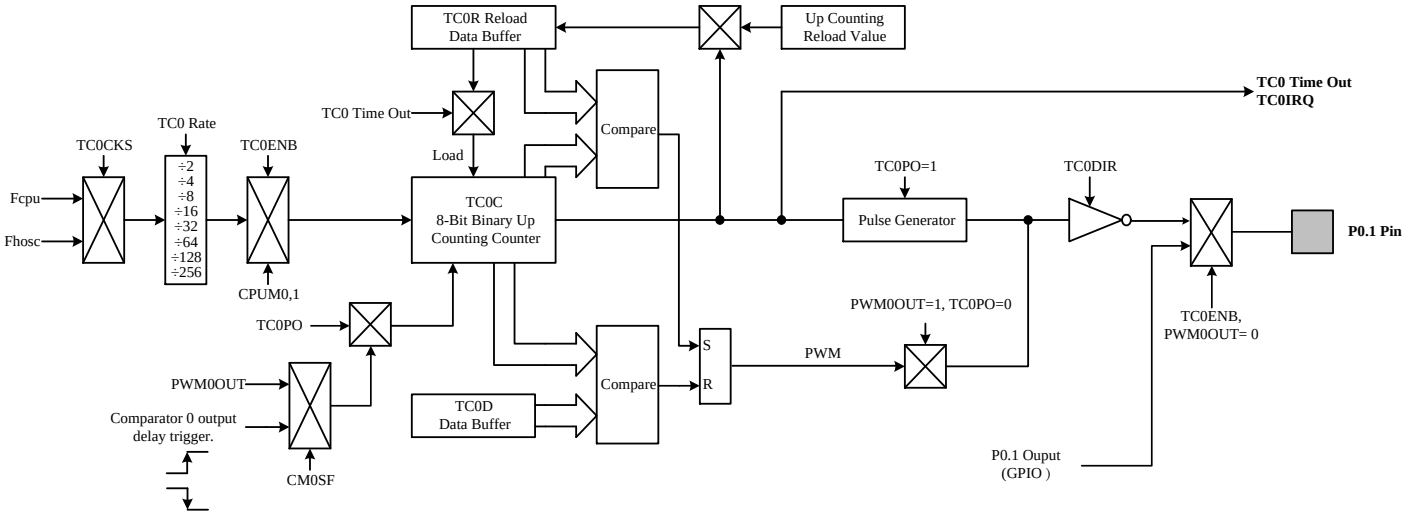
```
; Reset T2 timer.
        CLR         T2M             ; Clear T2M register.

; Set T2 clock source and T2 rate.
        MOV         A, #0nnn0000b
        B0MOV       TC0M, A

; Set T2 timer with TC0 pulse trigger mode enable.
        B0BSET      FT2SF           ; Enable T2 timer with TC0 pulse trigger mode.

; Set T2C and T2R register for T2 Interval time.
        MOV         A, #value       ; T2C must be equal to T2R.
        B0MOV       T2C, A
        B0MOV       T2R, A

; Clear T2IRQ
        B0BCLR      FT2IRQ

; Enable T2 timer and interrupt function.
        B0BSET      FT2IEN          ; Enable T2 interrupt function.
        B0BSET      FT2ENB          ; Enable T2 timer.
```

8.5 TC0 8-BIT TIMER/PWM/PULSE GENERATOR

8.5.1 OVERVIEW

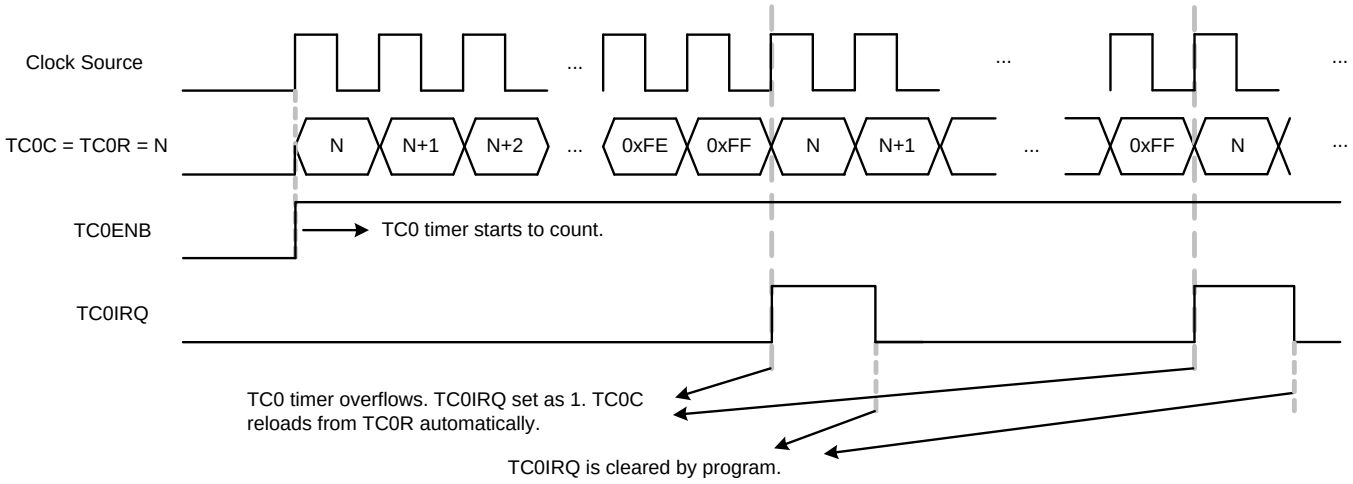
The TC0 timer is an 8-bit binary up timer with basic timer, PWM function and pulse generator. The basic timer function supports flag indicator (TC0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TC0M, TC0C, TC0R registers. TC0 builds in duty/cycle programmable PWM. The PWM cycle and resolution are controlled by TC0 timer clock rate, TC0R and TC0D registers, so the PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster…TC0 timer also builds in pulse generator function. The pulse generator function is one cycle PWM format as start trigger occurrence. The pulse output trigger source has TC0PO control bit and comparator 0 output edge controlled by register. TC0 counter supports auto-reload function which always enabled. When TC0 timer overflow occurs, the TC0C will be reloaded from TC0R automatically. The auto-reload function is always enabled. The TC0 doesn't build in green mode wake-up function. The main purposes of the TC0 timer are as following

- **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** TC0 timer function supports interrupt function. When TC0 timer occurs overflow, the TC0IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by TC0R and TC0D registers.
- **Pulse generator:** The pulse generator function is one cycle PWM format as start trigger occurrence. The pulse output trigger source has TC0PO control bit and comparator 0 output edge controlled by register. When TC0PO = 1, CM0SF = 0, the pulse generator trigger is PWM0OUT bit. When TC0PO = 1, CM0SF = 1, the pulse generator trigger source is comparator 0 output edge.
- **Green mode function:** All TC0 functions (timer, PWM, pulse generator) keeps running in green mode, but no wake up function. Timer IRQ actives as any IRQ trigger occurrence, e.g. timer overflow…



8.5.2 TC0 TIMER OPERATION

TC0 timer is controlled by TC0ENB bit. When TC0ENB=0, TC0 timer stops. When TC0ENB=1, TC0 timer starts to count. Before enabling TC0 timer, setup TC0 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…TC0C increases "1" by timer clock source. When TC0 overflow event occurs, TC0IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is TC0C count from full scale (0xFF) to zero scale (0x00). In difference function modes, TC0C value relates to operation. If TC0C value changing effects operation, the transition of operations would make timer function error. So TC0 builds in double buffer to avoid these situations happen. The double buffer concept is to flash TC0C during TC0 counting, to set the new value to TC0R (reload buffer), and the new value will be loaded from TC0R to TC0C after TC0 overflow occurrence automatically. In the next cycle, the TC0 timer runs under new conditions, and no any transitions occur. The auto-reload function is no any control interface and always actives as TC0 enables. If TC0 timer interrupt function is enabled (TC0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 000CH) and executes interrupt service routine after TC0 overflow occurrence. Clear TC0IRQ by program is necessary in interrupt procedure. TC0 timer can works in normal mode, slow mode and green mode. But in green mode, TC0 keep counting, set TC0IRQ and outputs PWM, but can't wake-up system.

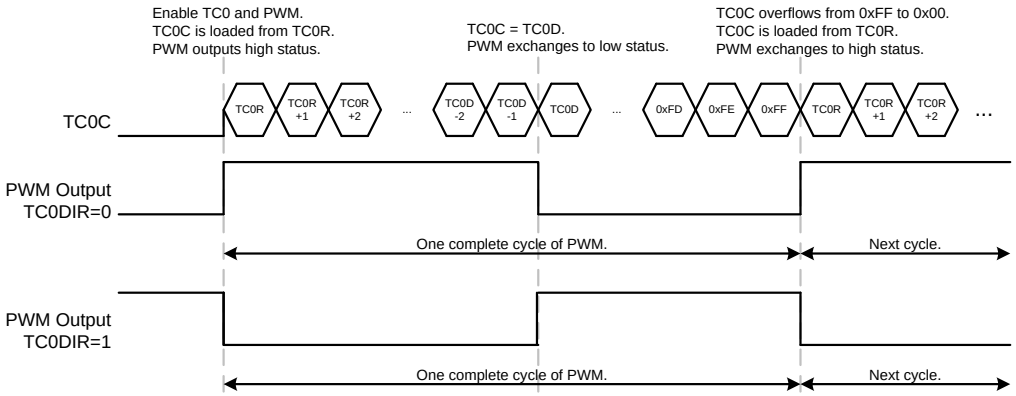


TC0 provides different clock sources to implement different applications and configurations. TC0 clock source includes Fcpu (instruction cycle) and Fhosc (high speed oscillator) controlled by TC0CKS bit. TC0CKS bit selects the clock source is from Fcpu or Fhosc. If TC0CKS=0, TC0 clock source is Fcpu through TC0rate[2:0] pre-scalar to decide Fcpu/2~Fcpu/256. If TC0CKS=1, TC0 clock source is Fhosc through TC0rate[2:0] pre-scalar to decide Fhosc/2~Fhosc/256. TC0 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

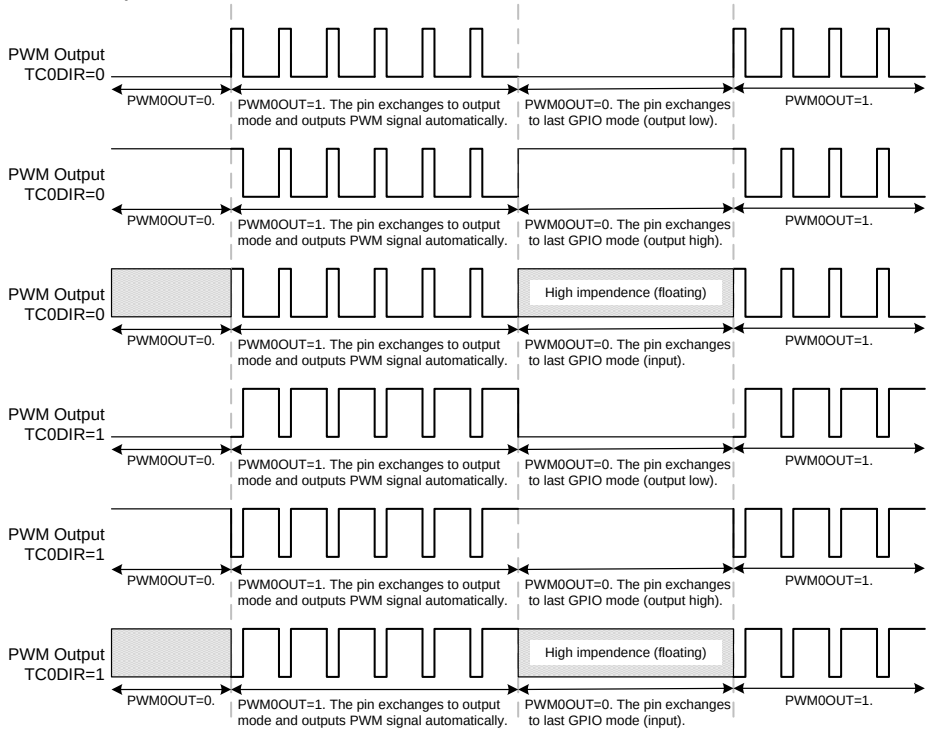
TC0CKS	TC0rate[2:0]	TC0 Clock	TC0 Interval Time			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/256	16.384	64	65.536	256
0	001b	Fcpu/128	8.192	32	32.768	128
0	010b	Fcpu/64	4.096	16	16.384	64
0	011b	Fcpu/32	2.048	8	8.192	32
0	100b	Fcpu/16	1.024	4	4.096	16
0	101b	Fcpu/8	0.512	2	2.048	8
0	110b	Fcpu/4	0.256	1	1.024	4
0	111b	Fcpu/2	0.128	0.5	0.512	2
1	000b	Fhosc/256	4.096	16	16.384	64
1	001b	Fhosc/128	2.048	8	8.192	32
1	010b	Fhosc/64	1.024	4	4.096	16
1	011b	Fhosc/32	0.512	2	2.048	8
1	100b	Fhosc/16	0.256	1	1.024	4
1	101b	Fhosc/8	0.128	0.5	0.512	2
1	110b	Fhosc/4	0.064	0.25	0.256	1
1	111b	Fhosc/2	0.032	0.125	0.128	0.5

8.5.3 PULSE WIDTH MODULATION (PWM)

TC0 timer builds in PWM function controlled by PWM0OUT bit. PWM output pin is shared with GPIO. When PWM0OUT=1, the PWM function is enabled and GPIO pin is switched from GPIO to PWM output status. When PWM0OUT=0, PWM output pin returns to GPIO last status. PWM signal is generated from the result of TC0C, TC0R and TC0D comparison. When PWM0OUT=1 or TC0C counts from 0xFF to 0x00 (overflow), the PWM outputs high status which is the PWM initial status. TC0C is loaded new data from TC0R register to decide PWM cycle and resolution. TC0C keeps counting, and the system compares TC0C and TC0D. When TC0C=TC0D, the PWM output status exchanges to low. TC0C keeps counting. When TC0 timer overflow occurs, and one cycle of PWM signal finishes. TC0C is reloaded from TC0R automatically, and PWM output status exchanges to high for next cycle. TC0D decides the high duty duration, and TC0R decides the resolution and cycle of PWM. TC0R can't be larger than TC0D, or the PWM signal is error. The PWM output phase can be selected through TC0DIR bit. When TC0DIR = 0, PWM output high pulse (from TC0C to TC0D) and PWM output low pulse (from TC0D to timer overflow). When TC0DIR = 1, PWM output low pulse (from TC0C to TC0D) and PWM output high pulse (from TC0D to timer overflow).



The resolution of PWM is decided by TC0R. TC0R range is from 0x00~0xFF. If TC0R = 0x00, PWM's resolution is 1/256. If TC0R = 0x80, PWM's resolution is 1/128. TC0D controls the high pulse width of PWM for PWM's duty. When TC0C = TC0D, PWM output exchanges to low status. TC0D must be greater than TC0R, or the PWM signal keeps low status. When PWM outputs, TC0IRQ still actives as TC0 overflows, and TC0 interrupt function actives as TC0IEN = 1. But strongly recommend be careful to use PWM and TC0 timer together, and make sure both functions work well.
The PWM output pin is shared with GPIO and switch to output PWM signal as PWM0OUT=1 automatically. If PWM0OUT bit is cleared to disable PWM, the output pin exchanges to last GPIO mode automatically. It easily to implement carry signal on/off operation, not to control TC0ENB bit.



8.5.4 TC0 PULSE GENERATOR FUNCTION

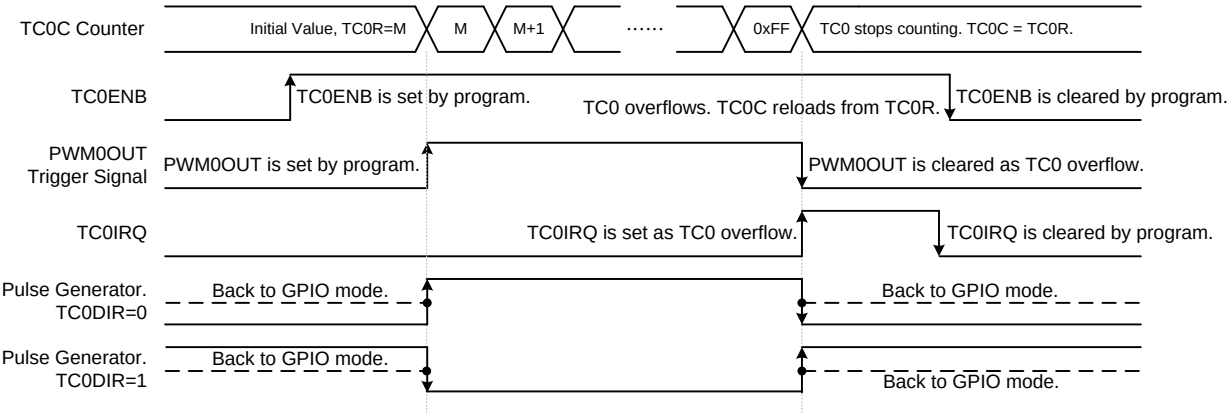
TC0 timer builds in pulse generator function. The pulse generator outputs a pulse, and the pulse width is decided by TC0 timer's interval time. The pulse generator is controlled by TC0PO bit. When TC0PO = 0, TC0 is normal timer mode or PWM function mode. When TC0PO = 1, TC0 is pulse generator mode. The pulse generator needs a start trigger signal to control TC0 counter and pulse signal output. When TC0PO is set as "1", TC0 counter keeps stopping, and TC0C/TC0R registers' value is set by program. TC0C value decides the pulse width. When the trigger event occurs, TC0 counter starts to count, and pulse output pin outputs pulse status controlled TC0DIR. When TC0 counter overflows, pulse signal finishes and changes to idle status. The TC0C stops counting and reloads new value through TC0R register. In pulse generator mode, the TC0IRQ is issued as TC0 counter overflow. During pulse generator operating, to change pulse width is through TC0R, not TC0C, or the pulse width would be error.

One pulse PWM function output signal needs time to synchronize in normal mode and slow mode so designer should be very carefully to process the synchronous timing and the first PWM duty cycle must be correct. When PWM output signal synchronize have finish, then PWM channel exchanges from GPIO to pulse generator output. When PWM output signal have finish, the PWM channel returns to GPIO mode and idle status.

TC0 Rate=Fhosc/2=8MHz @Fhosc=16MHz					
TC0C/TC0R	0x00	0x01	…	0xFE	0xFF
Pulse Width (ns)	31875	31750	…	125	0
TC0 Rate=Fhosc/4=4KHz @Fhosc=16MHz					
TC0C/TC0R	0x00	0x01	…	0xFE	0xFF
Pulse Width (us)	63750	63500	…	250	0
TC0 Rate=Fhosc/256=62.5KHz @Fhosc=16MHz					
TC0C/TC0R	0x00	0x01	…	0xFE	0xFF
Pulse Width (us)	4080	4064	…	16	0
TC0 Rate=Fcpu/2=0.5MHz @Fcpu=Fhosc/16=16MHz/16=1MHz					
TC0C/TC0R	0x00	0x01	…	0xFE	0xFF
Pulse Width (us)	510	508	…	2	0
TC0 Rate=Fcpu/256=3.90625KHz @Fcpu=Fhosc/16=16MHz/16=1MHz					
TC0C/TC0R	0x00	0x01	…	0xFE	0xFF
Pulse Width (us)	65280	65024	…	256	0

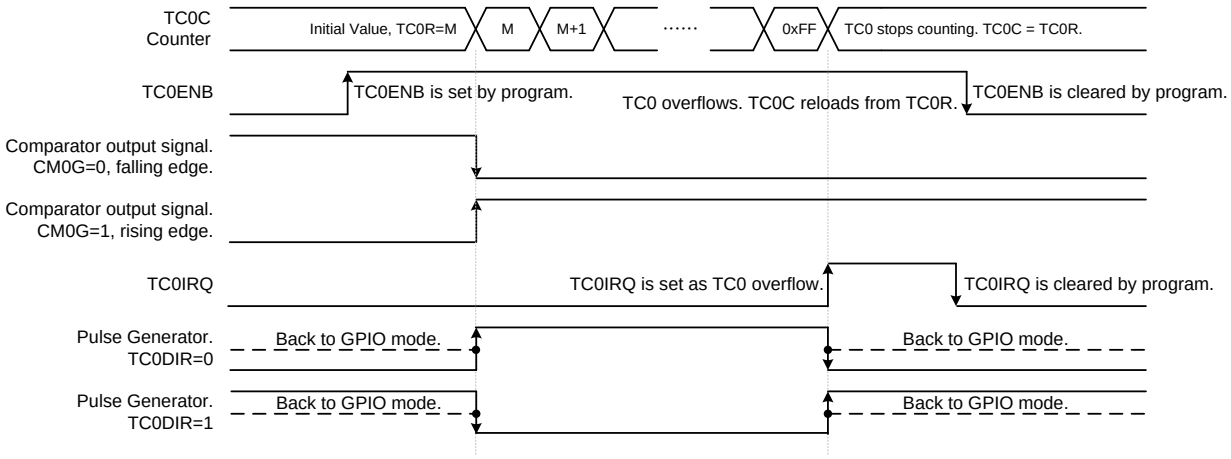
The pulse output control signal includes two trigger sources, and CM0SF bit controls TC0 pulse generator trigger source. One is PWM0OUT bit (CM0SF=0), and the other is comparator 0 output edge (CM0SF=1). If the trigger source is PWM0OUT bit, set PWM0OUT bit to output pulse signal by program, and PWM0OUT bit is cleared as TC0 counter overflow. To output next pulse is to set PWM0OUT bit by program again.

- **TC0PO=1, CM0SF=0: TC0ENB must be set as "1". TC0 8-bit binary up counter is controlled by PWM0OUT bit. If PWM0OUT bit is set as "1" by program, TC0 starts to count. If TC0 overflows, TC0 stops counting, PWM0OUT bit is cleared automatically, TC0IRQ is issued, and TC0C reloads new value from TC0R. It is necessary to set PWM0OUT = 1 by program making TC0 counts again.**



If the trigger is comparator 0 output delay edge (rising edge and falling edge controlled by comparator control register's CM0G bit), pulse starts to output as trigger edge condition occurrence. When TC0 overflows, pulse output pin returns to GPIO mode (idle status).

- **TC0PO=1, CM0SF=1: TC0ENB must be set as "1". TC0 8-bit binary up counter is controlled by comparator 0 output edge condition. The trigger edge can be selected through CM0G bit. If comparator output edge occurs, TC0 starts to count. If TC0 overflows, TC0 stops counting, TC0IRQ is issued, and TC0C reloads new value from TC0R.**



8.5.5 TC0M MODE REGISTER

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

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	TC0DIR	TC0PO	PWM0OUT
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

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

Bit [6:4] **TC0RATE [2:0]:** TC0 timer clock source select bits.
TC0CKS = 0:
000 = Fcpu/256, 001 = Fcpu/128, 010 = Fcpu/64, 011 = Fcpu/32, 100 = Fcpu/16, 101 = Fcpu/8, 110 = Fcpu/4, 111 = Fcpu/2.
TC0CKS = 1:
000 = Fhosc/256, 001 = Fhosc /128, 010 = Fhosc /64, 011 = Fhosc /32, 100 = Fhosc /16, 101 = Fhosc /8, 110 = Fhosc /4, 111 = Fhosc /2.

Bit 3 **TC0CKS:** TC0 clock source select bit.
0 = TC0 clock source is internal system clock (Fcpu).
1 = TC0 clock source is high clock source (Fhosc).

Bit 2 **TC0DIR:** PWM0 and Pulse generator output phase select bit.
0 = Normal phase. High pulse outputs and pulse output finished back to GPIO mode.
1 = Inverse phase. Low pulse outputs and pulse output finished back to GPIO mode.

Bit 1 **TC0PO:** TC0 pulse output function control bit.
0 = Disable.
1 = Enable TC0 pulse output function through P0.1 pin.

Bit 0 **PWM0OUT:** PWM0 output and pulse generator output control bit.
TC0PO = 0:
0 = Disable PWM0 output function, and P0.1 is GPIO mode.
1 = Enable PWM0 output function, and PWM0 signal outputs through P0.1 pin.
TC0PO = 1:
0 = Stop pulse output, or the end of pulse output cleared automatically.
1 = Enable pulse output.

8.5.6 TC0C COUNTING REGISTER

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

0B5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0C	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of TC0C initial value is as following.

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

8.5.7 TC0R AUTO-RELOAD REGISTER

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

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of TC0R initial value is as following.

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

- **Example: To calculation TC0C and TC0R value to obtain 10ms TC0 interval time. TC0 clock source is Fcpu = 16MHz/16 = 1MHz. Select TC0RATE=000 (Fcpu/128).**

  TC0 interval time = 10ms. TC0 clock rate = 16MHz/16/128

  *TC0C/TC0R initial value = 256 - (TC0 interval time * input clock)*
  *= 256 - (10ms * 16MHz / 16 / 128)*
  *= 256 - (10^{-2} * 16 * 10^{6} / 16 / 128)*
  = B2H

8.5.8 TC0D PWM DUTY REGISTER

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

0B7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0D	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

The equation of TC0D initial value is as following.

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

- **Example: To calculate TC0D value to obtain 1/3 duty PWM signal. The TC0 clock source is Fcpu = 16MHz/16= 1MHz. Select TC0RATE=000 (Fcpu/128). TC0R = B2H. TC0 interval time = 10ms. So the PWM cycle is 100Hz. In 1/3 duty condition, the high pulse width is about 3.33ms.**

TC0D initial value = B2H + (PWM high pulse width period / TC0 clock rate)
*= B2H + (3.33ms * 16MHz / 16 / 128)*
= B2H + 1AH
= CCH

8.5.9 TC0 TIMER OPERATION EXPLAME

➢ **TC0 TIMER CONFIGURATION:**

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

; Set TC0 rate.
        MOV     A, #0nnn0000b
        B0MOV   TC0M, A

; Set TC0 clock source.
        B0BCLR  FTC0CKS         ; TC0 clock source is Fcpu.
; or
        B0BSET  FTC0CKS         ; TC0 clock source is Fhosc.

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

; Clear TC0IRQ
        B0BCLR  FTC0IRQ

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

➢ **TC0 PWM CONFIGURATION:**

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

; Set TC0 rate.
        MOV     A, #0nnn0000b
        B0MOV   TC0M, A

; Set TC0 clock source.
        B0BCLR  FTC0CKS         ; TC0 clock source is Fcpu.
; or
        B0BSET  FTC0CKS         ; TC0 clock source is Fhosc.

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

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

; Set PWM output phase.
        B0BCLR  FTC0DIR         ; High pulse.
; or
        B0BSET  FTC0DIR         ; Low pulse.

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

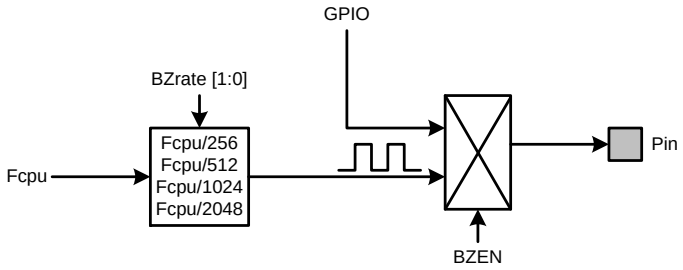
➢ **TC0 PULSE GENERATOR CONFIGURATION:**

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

; Set TC0 rate.
        MOV     A, #0nnn0000b
        B0MOV   TC0M, A

; Set TC0 clock source.
        B0BCLR  FTC0CKS         ; TC0 clock source is Fcpu.
; or
        B0BSET  FTC0CKS         ; TC0 clock source is Fhosc.

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

; Set pulse output phase.
        B0BCLR  FTC0DIR         ; High pulse.
; or
        B0BSET  FTC0DIR         ; Low pulse.

; Set pulse output trigger source.
        B0BCLR  FCM0SF          ; Pulse output trigger source is PWM0OUT bit.
; or
        B0BSET  FCM0SF          ; Pulse output trigger source is comparator 0 output edge.

; Enable pulse output and TC0 timer.
        B0BSET  FTC0PO          ; Enable pulse output function.
        B0BSET  FTC0ENB         ; Enable TC0 timer.
```

9 2K/4K BUZZER GENERATOR

9.1 OVERVIEW

The MCU builds in Buzzer generator to drive external buzzer device whose frequency is 2KHz or 4KHz. Adjusting buzzer output frequency is through BZM register. The buzzer output pin is shared with GPIO. When BZEN = 1, the pin outputs buzzer carry signal. When BZEN = 0, the pin returns to GPIO last condition (input mode, output high or output low status).



The buzzer frequency is divided from Fcpu (instruction cycle) controlled by BZrate bits, and Fcpu decides the buzzer frequency. The buzzer target frequency is 2KHz or 4KHz. It is important to choice a good Fcpu rate to obtain the correct buzzer frequency. The selection table is as following.

BZrate [1:0]	Buzzer Rate Division	Buzzer Rate		
		Fcpu = 1MHz	Fcpu = 2MHz	Fcpu = 4MHz
00	Fcpu/256	**4KHz**	8KHz	16KHz
01	Fcpu/512	**2KHz**	**4KHz**	8KHz
10	Fcpu/1024	1KHz	**2KHz**	**4KHz**
11	Fcpu/2048	0.5KHz	1KHz	**2KHz**

9.2 BZM REGISTER

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BZM	BZEN	BZrate1	BZrate0	-	-	-	-	-
Read/Write	R/W	R/W	R/W	-	-	-	-	-
After reset	0	0	0	-	-	-	-	-

Bit 7 **BZEN:** Buzzer output control bit.
0 = Disable BZ output and BZ output pin transfers to I/O last status.
1 = Enable BZ output and disable GPIO function.

Bit[6:5] **BZrate[1:0]:** Buzzer rate control bits.
00 = Fcpu/256
01 = Fcpu/512
10 = Fcpu/1024
11 = Fcpu/2048

✷ ***Note:***
1. ***If BZEN=0, the buzzer output pin is GPIO mode and returns to last status after disabling buzzer output.***
2. ***If BZEN=1, the buzzer output pin is buzzer output function and isolates the GPIO function***

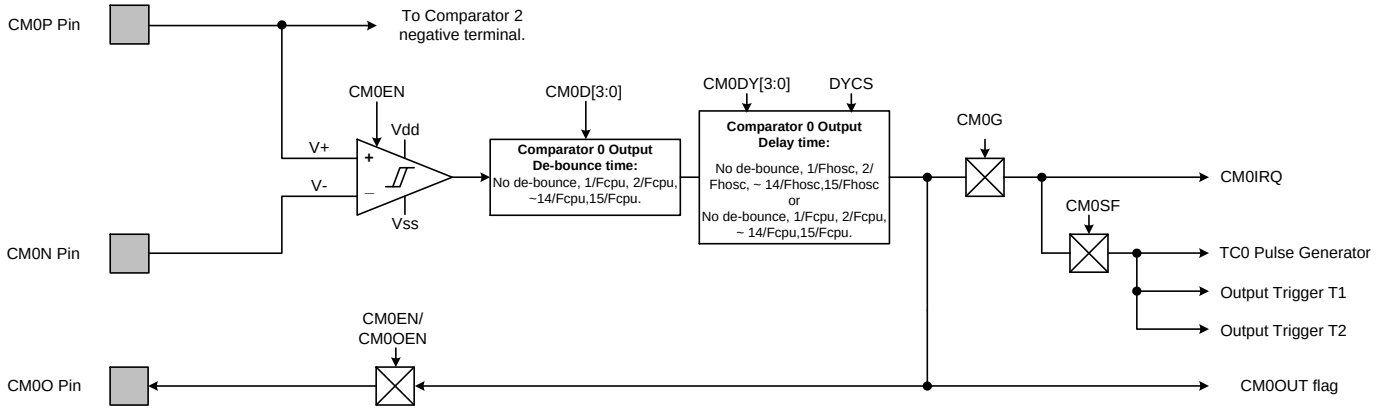
.

10 ANALOG COMPARAOTR 0

10.1 OVERVIEW

The micro-controller builds in one comparator with TC0 pulse generator trigger function. The comparator has normal comparator mode and TC0 pulse output trigger source. The comparator isn't Rail-to-Rail structure. That means the input/output voltage isn't real from Vdd~Vss (Reference to "Electrical characteristics" chapter). When the positive input voltage is greater than the negative input voltage, the comparator output is high. When the positive input voltage is smaller than the negative input voltage, the comparator output is low. The main purposes of comparator 0 are as following.

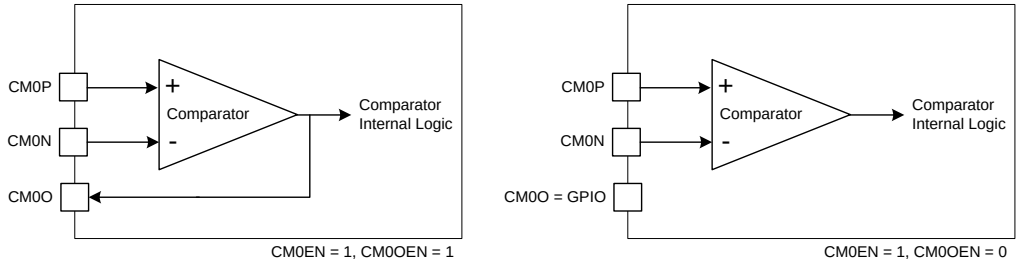
- **Normal comparator function:** General comparator mode compares the two tensions of positive input terminal and negative input terminal.
- **Interrupt function:** Comparator 0 supports interrupt function. When comparator 0 output edge direction is equal to edge selection, the CM0IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **TC0 pulse generator trigger source:** Comparator 0 can be TC0 pulse generator trigger source controlled by CM0SF bit. When TC0PO = 1 and CM0SF = 1, comparator 0 output status triggers TC0 pulse generator to outputs pulse signal.
- **Green mode function:** Comparator 0 still actives in green mode, but no wake-up function. CM0IRQ can be latched as trigger event occurrence until system wakes up and returns to normal mode. After system wakes up, the comparator 0 interrupt service routine is executed by program.



10.2 NORMAL COMPARATOR MODE

10.2.1 COMPARATOR ENABLE

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 enabled connected to comparator positive terminal. CM0OEN controls comparator output connected to GPIO or not. When CM0OEN=1, CM0O pin is comparator 0 output status. When CM0OEN=0, comparator output status can be read through CM0OUT flag.



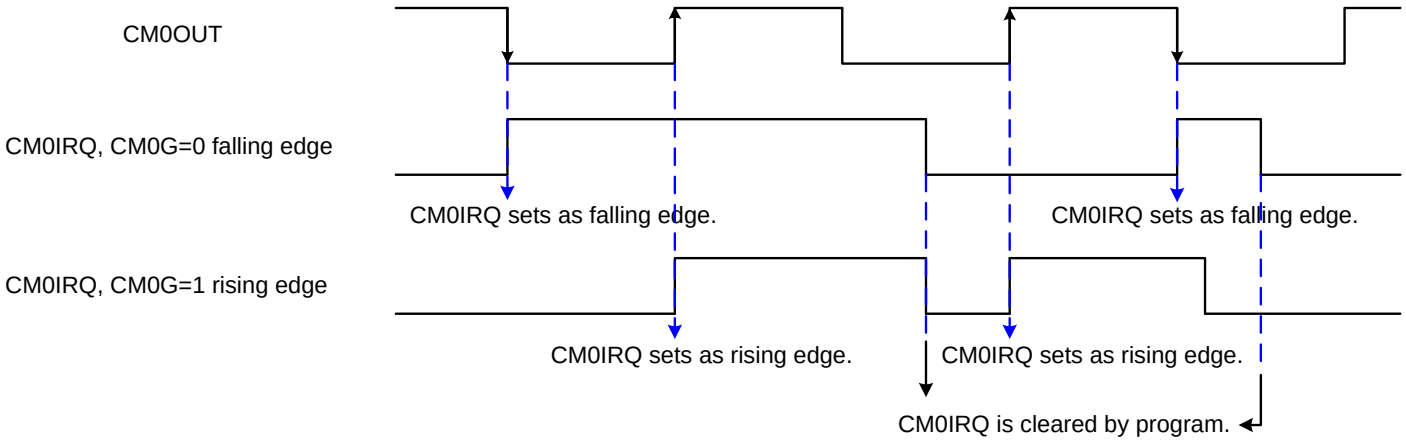
- ***Note: The comparator enable condition is fixed CM0EN=1, or the comparator pins are GPIO mode and comparator is disabled.***

10.2.2 CM0OUT, CM0G AND CM0IRQ 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 bit. When CM0G = 0, the comparator 0 interrupt trigger direction is falling edge. When CM0G = 1, the comparator 0 interrupt trigger direction is rising 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 bit including rising edge and falling edge. If CM0G = 0, comparator output trigger edge is falling edge. If CM0G = 1, comparator output trigger edge is rising edge. The edge detection is from comparator output signal through de-bounce and delay processor. When comparator output edge event occurs and equal CM0G condition, CM0IRQ flag is issued. If CM0IEN = 1, program counter points to interrupt vector to execute interrupt service routine.

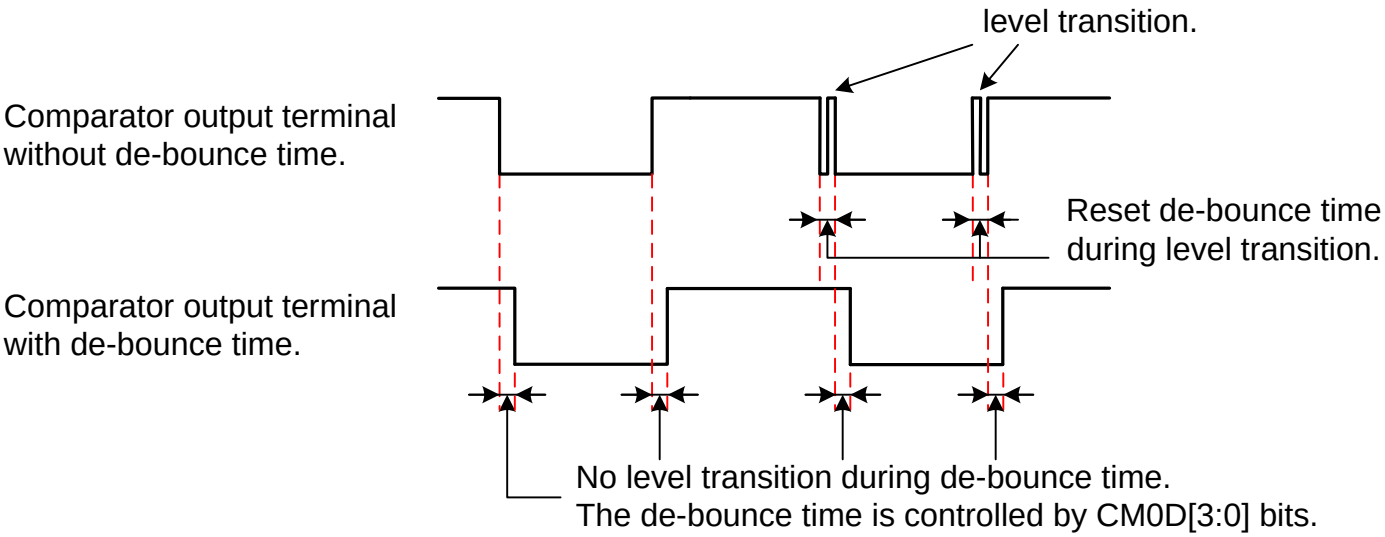


10.2.3 DE-BOUNCE TIME CONTROL

Comparator 0 compares positive terminal's voltage and negative terminal's voltage, and then output result to output pin or delay circuit. When V+ > V-, comparator outputs high status. When V+ < V-, comparator outputs low status. Comparator output terminal builds in de-bounce time control block to achieve output hysteresis to filter output transition condition. The de-bounce time option has 16-step including no de-bounce, 1/Fcpu, 2/Fcpu, 3/Fcpu, 4/Fcpu, 5/Fcpu, 6/Fcpu, 7/Fcpu, 8/Fcpu, 9/Fcpu, 10/Fhosc, 11/Fcpu, 12/Fcpu, 13/Fcpu, 14/Fcpu, 15/Fcpu controlled by CM0D[3:0] bits.

CM0D[3:0]	**0000b**	**0001b**	**0010b**	**0011b**	**0100b**	**0101b**	**0110b**	**0111b**
	No de-bounce	1/Fcpu	2/ Fcpu	3/ Fcpu	4/ Fcpu	5/ Fcpu	6/ Fcpu	7/ Fcpu
De-bounce time (us) Fhosc=16MHz	0	0.0625	0.125	0.1875	0.25	0.3125	0.375	0.4375
De-bounce time (us) Fhosc=4MHz	0	0.25	0.5	0.75	1	1.25	1.5	1.75
CM0D[3:0]	**1000b**	**1001b**	**1010b**	**1011b**	**1100b**	**1101b**	**1110b**	**1111b**
	8/ Fcpu	9/ Fcpu	10/ Fcpu	11/ Fcpu	12/ Fcpu	13/ Fcpu	14/ Fcpu	15/ Fcpu
De-bounce time (us) Fhosc=16MHz	0.5	0.5625	0.625	0.6875	0.75	0.8125	0.875	0.9375
De-bounce time (us) Fhosc=4MHz	2	2.25	2.5	2.75	3	3.25	3.5	3.75

Comparator output terminal:



10.2.4 DELAY TIME CONTROL

Comparator 0 builds in output delay function. The output delay function clock source is controlled by CM0CLK bit.
The comparator 0 builds in delay control block. When CM0CLK = 1, the delay option has 16-step including no delay, 1/Fcpu, 2/Fcpu ~ 14/Fcpu, 15/Fcpu controlled by CM0DY[3:0] bits. When CM0CLK = 0, the delay option has 16-step including no delay, 1/Fhosc, 2/Fhosc ~ 14/Fhosc, 15/Fhosc controlled by CM0DY[3:0] bits

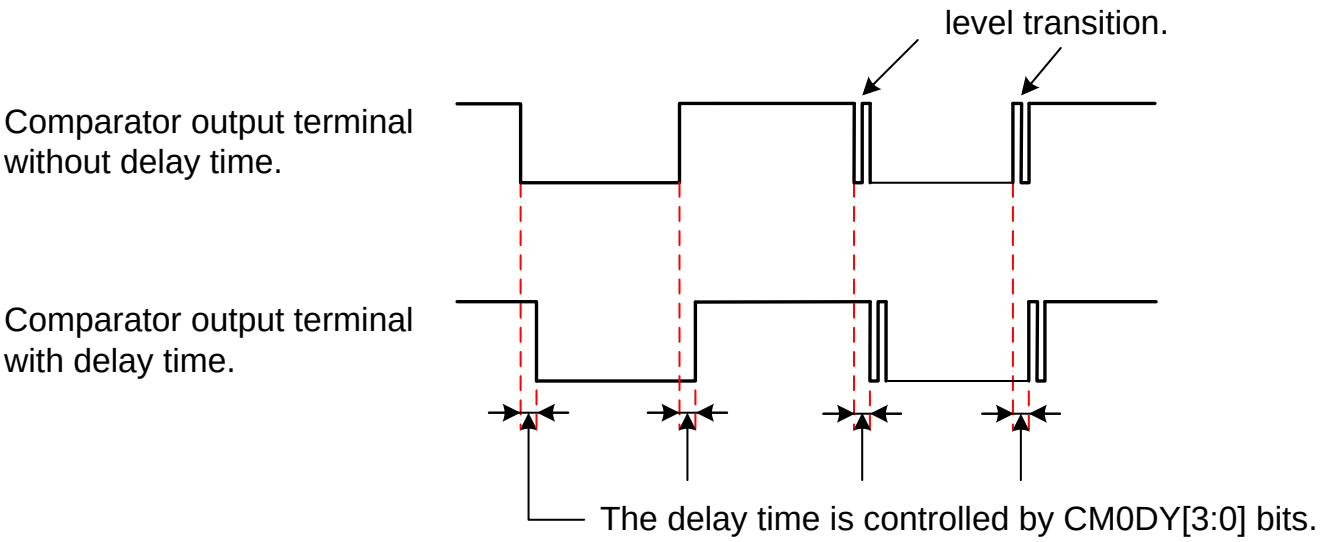
CM0CLK = 0:

CM0DY[3:0]	**0000b**	**0001b**	**0010b**	**0011b**	**0100b**	**0101b**	**0110b**	**0111b**
	No de-bounce	1/Fhosc	2/ Fhosc	3/ Fhosc	4/ Fhosc	5/ Fhosc	6/ Fhosc	7/ Fhosc
Delay time (us) Fcpu=Fhosc/4 =16MHz/4=4MHz	0	0.0625	0.125	0.1875	0.25	0.3125	0.375	0.4375
Delay time (us) Fcpu=Fhosc/16 =16MHz/16=1MHz	0	0.0625	0.125	0.1875	0.25	0.3125	0.375	0.4375
CM0DY[3:0]	**1000b**	**1001b**	**1010b**	**1011b**	**1100b**	**1101b**	**1110b**	**1111b**
	8/ Fhosc	9/ Fhosc	10/ Fhosc	11/ Fhosc	12/ Fhosc	13/ Fhosc	14/ Fhosc	15/ Fhosc
Delay time (us) Fcpu=Fhosc/4 =16MHz/4=4MHz	0.5	0.5625	0.625	0.6875	0.75	0.8125	0.875	0.9375
Delay time (us) Fcpu=Fhosc/16 =16MHz/16=1MHz	0.5	0.5625	0.625	0.6875	0.75	0.8125	0.875	0.9375

CM0CLK = 1:

CM0DY[3:0]	**0000b**	**0001b**	**0010b**	**0011b**	**0100b**	**0101b**	**0110b**	**0111b**
	No de-bounce	1/Fcpu	2/Fcpu	3/Fcpu	4/Fcpu	5/Fcpu	6/Fcpu	7/Fcpu
Delay time (us) Fcpu=Fhosc/4 =16MHz/4=4MHz	0	0.25	0.5	0.75	1	1.25	1.5	1.75
Delay time (us) Fcpu=Fhosc/16 =16MHz/16=1MHz	0	1	2	3	4	5	6	7
CM0DY[3:0]	**1000b**	**1001b**	**1010b**	**1011b**	**1100b**	**1101b**	**1110b**	**1111b**
	8/Fcpu	9/Fcpu	10/Fcpu	11/Fcpu	12/Fcpu	13/Fcpu	14/Fcpu	15/Fcpu
Delay time (us) Fcpu=Fhosc/4 =16MHz/4=4MHz	2	2.25	2.5	2.75	3	3.25	3.5	3.75
Delay time (us) Fcpu=Fhosc/16 =16MHz/16=1MHz	8	9	10	11	12	13	14	15

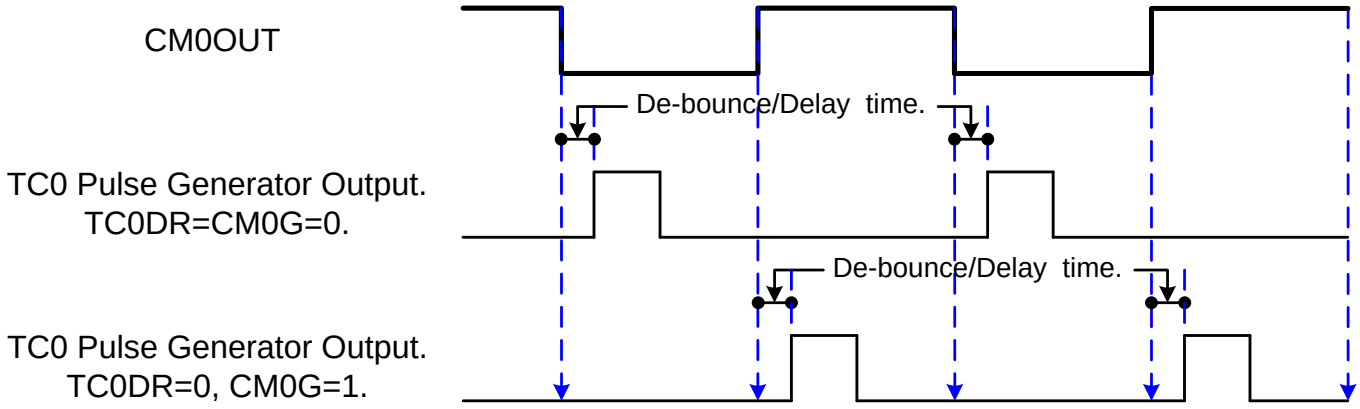
Comparator output terminal (No de-bounce):



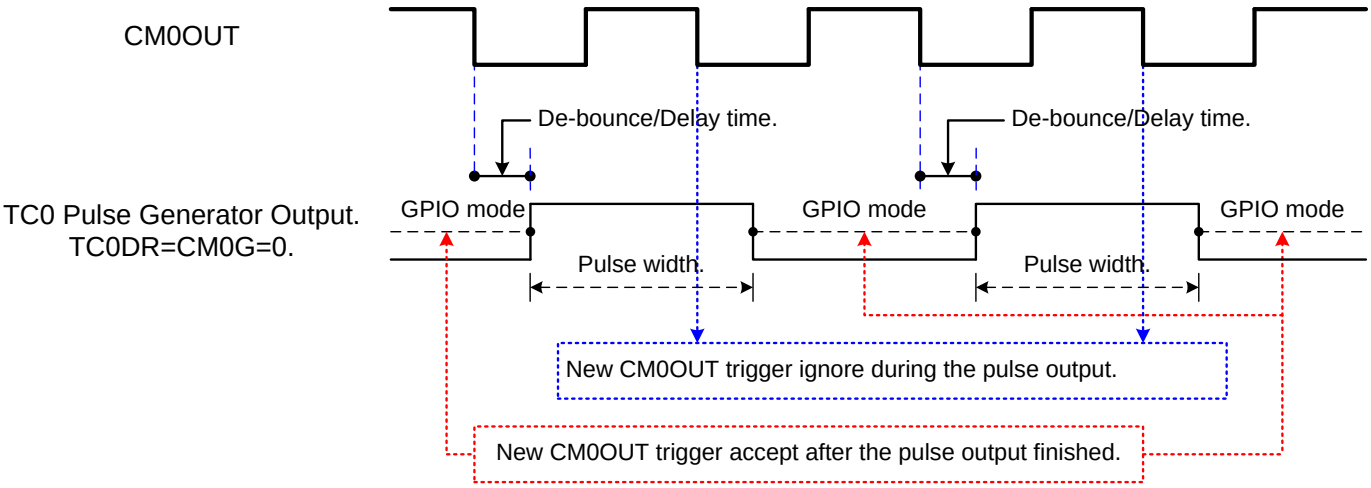
10.3 COMPARATOR 0 SPECIAL FUCNITON

Besides normal comparator function, comparator 0 builds in a special mode to trigger TC0 pulse generator through comparator output edge and controlled by CM0SF bit. When CM0SF=1, comparator 0 special mode is enabled. If comparator 0 output trigger condition occurs, TC0 pulse generator is triggered to output a pulse signal, and comparator interrupt function actives. More detail operation is referred to TC0 pulse generator contents.

- **TC0 pulse generator outputs signal (CM0G = 0/1).**



- **New CM0OUT trigger accept after the pulse output finished but ignore during the pulse output.**



10.4 COMPARATOR MODE REGISTER

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M	CM0EN	CM0OEN	CM0OUT	CM0SF	CM0G	CM0CLK	-	-
Read/Write	R/W	R/W	R	R/W	R/W	R/W	-	-
After Reset	0	0	0	0	0	0	-	-

Bit 2 **CM0CLK:** Comparator output delay time select bit.
0 = Comparator output delay time clock source is Fhosc.
1 = Comparator output delay time clock source is Fcpu.

Bit 3 **CM0G:** Comparator output trigger direction control bit.
0 = Falling edge trigger. Comparator output status is from high to low as CM0P < CM0N.
1 = Rising edge trigger. Comparator output status is from low to high as CM0P > CM0N.

Bit 4 **CM0SF:** Comparator 0 special mode control bit.
0 = Disable. Comparator 0 is normal comparator function.
1 = Enable. Comparator 0 output edge triggers TC0 pulse generator.

Bit 5 **CM0OUT:** Comparator 0 output flag bit (CM0EN = 0, CM0OUTdefault value is “0”).
0 = CM0P voltage is less than CM0N voltage.
1 = CM0P voltage is larger than CM0N voltage.

Bit 6 **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 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.

099H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMDB0	CM0DY3	CM0DY2	CM0DY1	CM0DY0	CM0D3	CM0D2	CM0D1	CM0D0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:4] **CM0DY[3:0]:** Comparator 0 delay time control bit.
CM0CLK =0
0000=No delay, 0001=1/Fhosc, 0010=2/Fhosc, 0011=3/Fhosc, 0100=4/Fhosc, 0101=5/Fhosc, 0110=6/Fhosc, 0111=7/Fhosc, 1000=8/Fhosc, 1001=9/Fhosc, 1010=10/Fhosc, 1011=11/Fhosc, 1100=12/Fhosc, 1101=13/Fhosc, 1110=14/Fhosc, 1111=15/Fhosc
CM0CLK = 1
0000=No delay, 0001=1/Fcpu, 0010=2/Fcpu, 0011=3/Fcpu, 0100=4/Fcpu, 0101=5/Fcpu, 0110=6/Fcpu, 0111=7/Fcpu, 1000=8/Fcpu, 1001=9/Fcpu, 1010=10/Fcpu, 1011=11/Fcpu, 1100=12/Fcpu,1101=13/Fcpu, 1110=14/Fcpu, 1111=15/Fcpu

Bit [3:0] **CM0D[3:0]:** Comparator 0 de-bounce time control bit.
0000=No de-bounce, 0001=1/Fcpu, 0010=2/Fcpu, 0011=3/Fcpu, 0100=4/Fcpu, 0101=5/Fcpu, 0110=6/Fcpu, 0111=7/Fcpu, 1000=8/Fcpu, 1001=9/Fcpu, 1010=10/Fcpu, 1011=11/Fcpu, 1100=12/Fcpu, 1101=13/Fcpu, 1110=14/Fcpu, 1111=15/Fcpu

10.5 COMPARATOR 0 PIN CONFIGURATION

The P0.2~P0.3 is shared with comparator input function and Schmitt trigger structure. Connect an analog signal to 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 0 will encounter above current leakage situation. CMCON is Port0 Configuration register. Write “1” into CMCON.n will configure related port 0 pin as pure analog input pin to avoid current leakage.

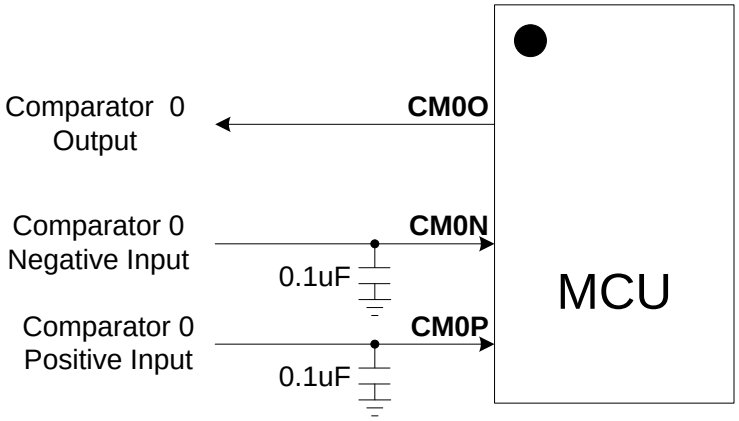
0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMCON	-	-	-	-	CM2NCON	CM1NCON	CM0NCON	CM0PCON
Read/Write	-	-	-	-	W	W	W	W
After reset	-	-	-	-	0	0	0	0

Bit1 **CM0NCON:** P0.3 configuration control bits.
0 = P0.3 can be an analog input (Comparator 0 negative input pin) or digital I/O pins.
1 = P0.3 is pure analog input, can’t be a digital I/O pin.

Bit0 **CM0PCON:** P0.2 configuration control bits.
0 = P0.2 can be an analog input (Comparator 0 positive input pin) or digital I/O pins.
1 = P0.2 is pure analog input, can’t be a digital I/O pin.

10.6 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.



10.7 COMPARATOR 0 OPERATION EXPLAME

➢ **COMPARATOR 0 CONFIGURATION:**

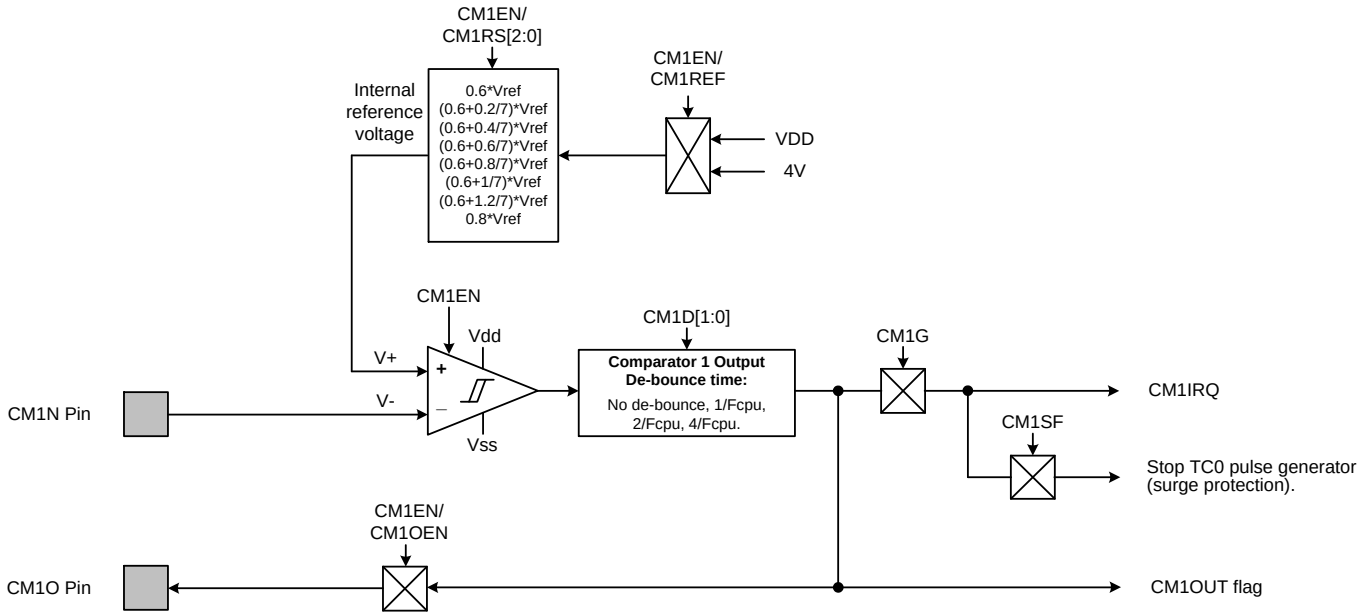
```
; Reset Comparator 0.
        CLR     CM0M            ; Clear CM0M register.

; Set Comparator 0 function mode.
        B0BCLR  FCM0SF          ; Normal comparator mode.
; or
        B0BSET  FCM0SF          ; Special function mode.

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

; Set Comparator 0 interrupt trigger edge.
        B0BCLR  FCM0G           ; Falling edge.
; or
        B0BSET  FCM0G           ; Rising edge.

; Set Comparator 0 output de-bounce time.
        B0MOV   A, CMDB0        ; Set CM0D[3:0] for comparator output de-bounce.
        AND     A, #1111 0000b
        OR      A, #0000 nnnnb
        B0MOV   CMDB0, A

; Set Comparator 0 output delay time.
        B0MOV   A, CMDB0        ; Set CM0DY[3:0] for comparator output de-bounce.
        AND     A, #00001111b
        OR      A, #nnnn0000b
        B0MOV   CMDB0, A

; Set Comparator 0 pin configuration.
        B0MOV   A, CMCON        ; Set CMCON[1:0] for isolate GPIO path.
        OR      A, #00000011b
        B0MOV   CMCON, A

; Clear CM0IRQ
        B0BCLR  FCM0IRQ

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

11 ANALOG COMPARAOTR 1

11.1 OVERVIEW

The micro-controller builds in one comparator with stopping TC0 pulse generator function (surge protection). The comparator has normal comparator mode and stopping TC0 pulse output trigger source. The comparator is not Rail-to-Rail structure. That means the input voltage is not real from Vdd~Vss (Reference to “Electrical characteristics” chapter). When the positive input voltage is greater than the negative input voltage, the comparator output is high. When the positive input voltage is smaller than the negative input voltage, the comparator output is low. The comparator builds in internal reference voltage connected to comparator positive terminal. The main purposes of comparator 1 are as following.

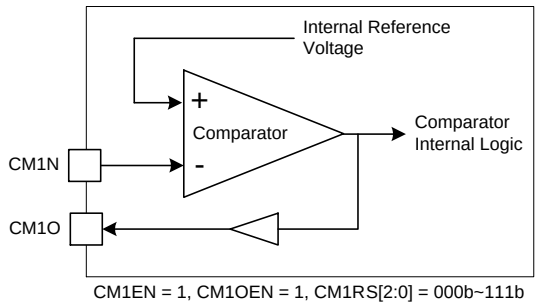
- **Normal comparator function:** General comparator mode compares the two tensions of positive input terminal and negative input terminal.
- **Interrupt function:** Comparator 1 supports interrupt function. When comparator 1 output edge direction is equal to edge selection, the CM1IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **TC0 pulse generator trigger stopping source:** Comparator 1 can be TC0 pulse generator stopping trigger source controlled by CM1SF bit. When TC0PO = 1 and CM1SF = 1, comparator 1 output status triggers TC0 pulse generator to stop outputting pulse signal (Surge protection).
- **Green mode function:** Comparator 1 still actives in green mode, but no wake-up function. CM1IRQ can be latched as trigger event occurrence until system wakes up. After system wakes up, the comparator 1 interrupt service routine is executed by program.



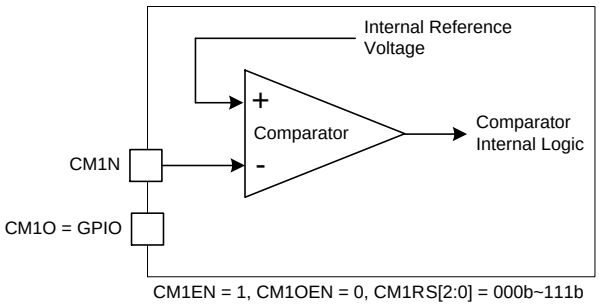
11.2 NORMAL COMPARATOR MODE

11.2.1 COMPARATOR ENABLE

Comparator pins are shared with GPIO controlled by CM1EN bit. When CM1EN=1, CM1N pin is enabled connected to comparator negative terminal. Comparator positive terminal is connected to internal reference voltage. The internal reference voltage is controlled by CM1RS [2:0] bits. When CM1RS [2:0] = 000b~111b, comparator positive terminal is connected to the internal reference voltage. The internal reference voltage includes 8-level which are 0.6*Vref, (0.6+0.2/7)*Vref, (0.6+0.4/7)*Vref, (0.6+0.6/7)*Vref, (0.6+0.8/7)*Vref, (0.6+1/7)*Vref, (0.6+1.2/7)*Vref, 0.8*Vref. The internal reference voltage power source is Vdd or 4V controlled by CM1REF bit CM1OEN controls comparator output to CM1O pin. When CM1EN = 1 and CM1OEN = 1, CM1O pin is comparator 1 output status. When CM1EN = 1 and CM1OEN = 0, comparator output status can be read only through CM1OUT flag.



CM1EN = 1, CM1OEN = 1, CM1RS[2:0] = 000b~111b



CM1EN = 1, CM1OEN = 0, CM1RS[2:0] = 000b~111b

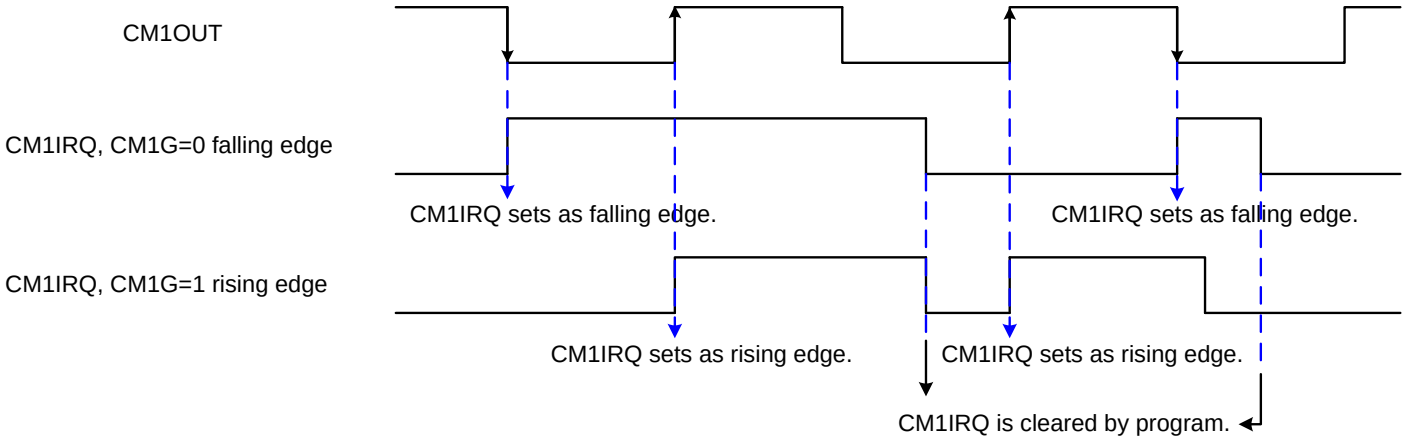
- ***Note: The comparator enable condition is fixed CM1EN=1, or the comparator pins are GPIO mode and comparator is disabled.***

11.2.2 CM1OUT, CM1G AND CM1IRQ STATUS

The CM1OUT and CM1IRQ bits indicate the comparator result. The CM1OUT shows the comparator result immediately, but the CM1IRQ only indicates the event of the comparator result. The event condition is controlled by register and includes rising edge (CM1OUT changes from low to high) and falling edge (CM1OUT changes from high to low) controlled by CM1G bit. When CM1G = 0, the comparator 1 interrupt trigger direction is falling edge. When CM1G = 1, the comparator 1 interrupt trigger direction is rising edge.

- ***Note: CM1OUT is comparator raw output without latch. It varies depend on the comparator process result. But the CM1IRQ is latch comparator output result. It must be cleared by program.***

Comparator supports interrupt function. The interrupt trigger condition can be selected through CM1G bit including rising edge and falling edge. If CM1G = 0, comparator output trigger edge is falling edge. If CM1G = 1, comparator output trigger edge is rising edge. The edge detection is from comparator output signal through de-bounce processor. When comparator output edge event occurs and equal CM1G condition, CM1IRQ flag is issued. If CM1IEN = 1, program counter points to interrupt vector to execute interrupt service routine.

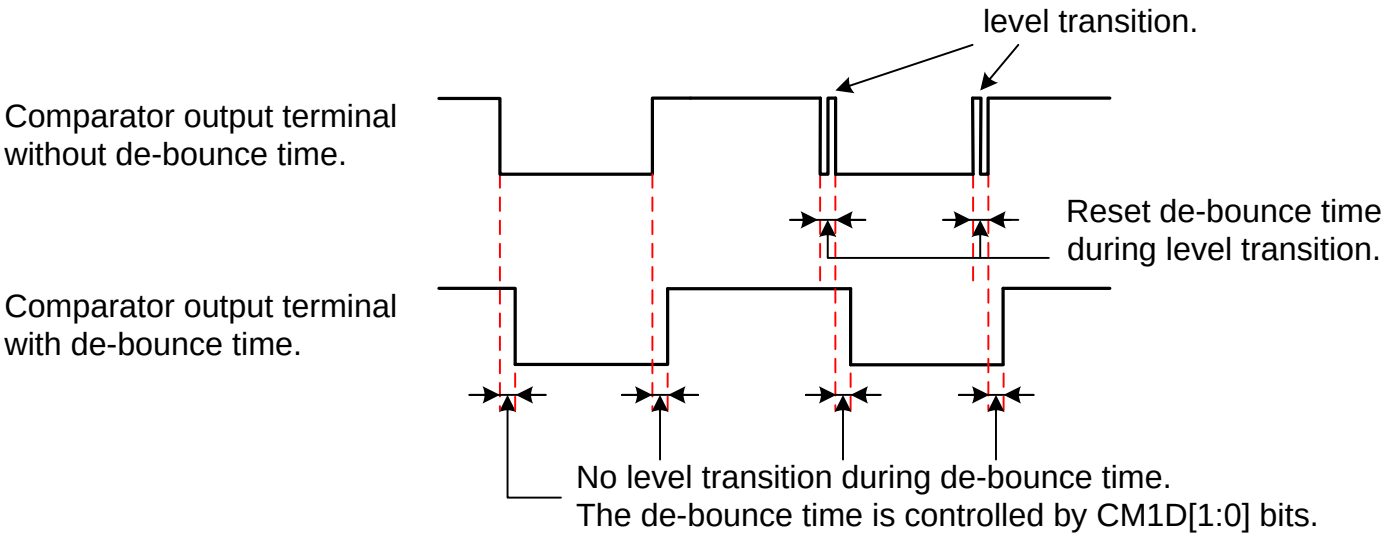


11.2.3 DE-BOUNCE TIME CONTROL

Comparator 1 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. Comparator output terminal builds in de-bounce time control block to achieve output hysteresis to filter output transition condition. The de-bounce time option has 4-step including no de-bounce, 1/Fcpu, 2/Fcpu, 4/Fcpu controlled by CM1D [1:0] bits.

CM1D[1:0]	**00b**	**01b**	**01b**	**11b**
	No de-bounce	1/Fcpu	2/Fcpu	4/Fcpu
De-bounce time (us) Fcpu=Fhosc/4 =16MHz/4=4MHz	0	0.25	0.5	1
De-bounce time (us) Fcpu=Fhosc/16 =16MHz/16=1MHz	0	1	2	4

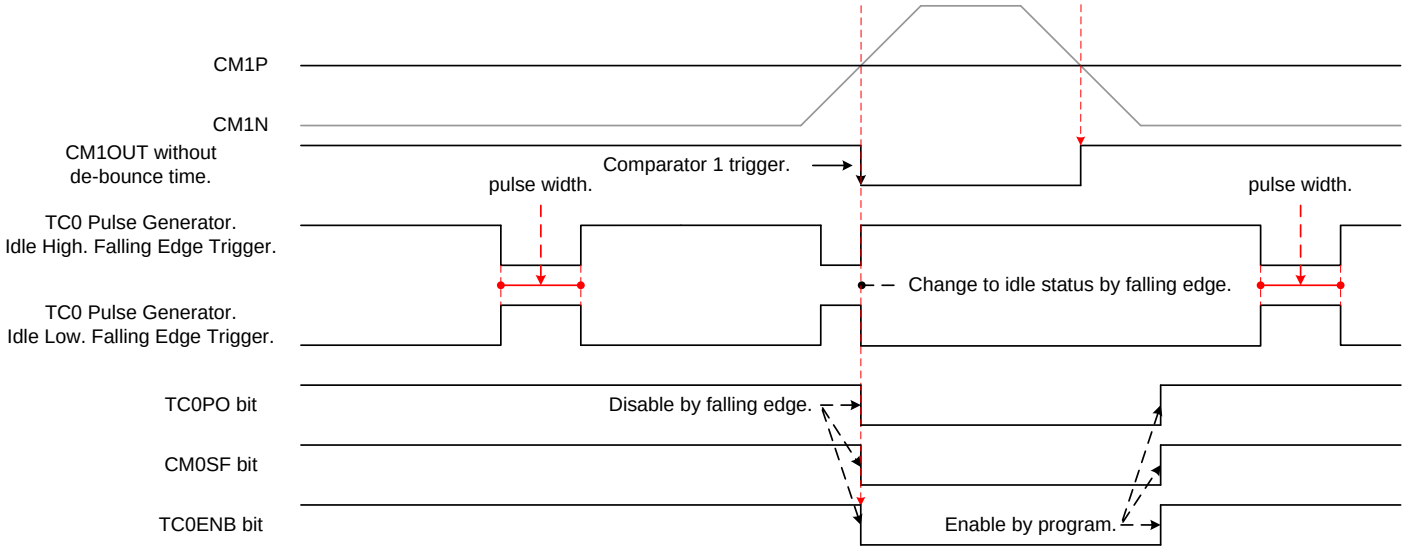
Comparator output terminal:



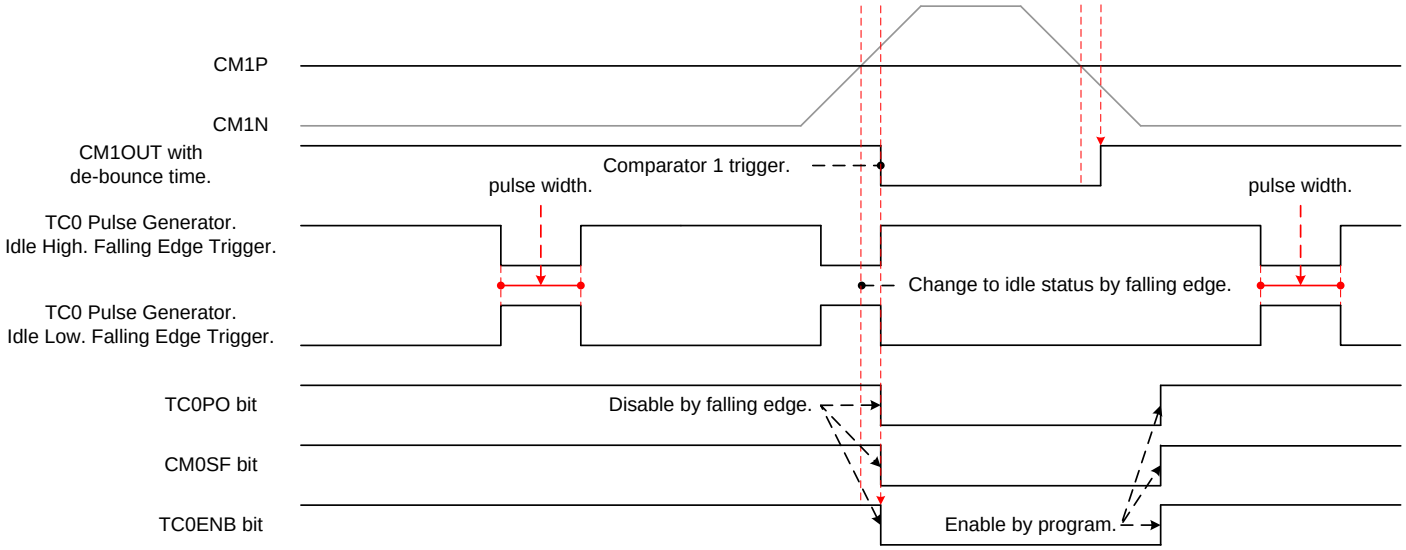
11.3 COMPARATOR 1 SPECIAL FUCNITON

Besides normal comparator function, comparator 1 builds in a special mode for IH-cooker application (stop TC0 pulse generator output signal, surge protection). The special mode is to trigger TC0 pulse generator stopping output (surge protection) through comparator output edge and controlled by CM1SF bit. When CM1SF=1, comparator 1 special mode is enabled. If comparator 1 output trigger condition occurs, TC0 pulse generator function is disabled to turn off external device (like IGBT driving pulse signal). In this condition, TC0PO, TC0ENB and CM0SF bits are cleared to disable pulse output function automatically. Pulse output pin exchanges to GPIO mode and last status. CM1IRQ is issued to indicate surge event. It is necessary to enable pulse generator by program.

- **Stop TC0 pulse output @falling edge trigger, without de-bounce time.**



- **Stop TC0 pulse output @falling edge trigger, with de-bounce time.**



- ***Note: If TC0 pulse output is stopped by comparator 1 special mode trigger, the CM0SF, TC0PO and TC0ENB bits are cleared automatically. It is necessary to set CM0SF, TC0PO and TC0ENB bits by program to recover TC0 pulse generator function.***

11.4 COMPARATOR 1 MODE REGISTER

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM1M	CM1EN	CM1OEN	CM1OUT	CM1SF	CM1G	CM1RS2	CM1RS1	CM1RS0
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 7 **CM1EN:** Comparator 1 control bit.
0 = Disable. Comparator pins are GPIO mode.
1 = Enable. CM1N pin is comparator negative input pin, and GPIO function is isolated. CM1O is controlled by CM1OEN bit. Comparator 1 positive terminal is connected to internal reference voltage. The internal reference voltage is selected by CM1RS [2:0] bits.

Bit 6 **CM1OEN:** Comparator 1 output pin control bit.
0 = Disable. CM1O pin is GPIO mode.
1 = Enable. CM1O pin is comparator output and isolate GPIO function.

Bit 5 **CM1OUT:** Comparator 1 output flag bit.
0 = Comparator 1 positive voltage is less than CM1N voltage (CM1EN = 0, CM1OUTdefault value is “0”).
1 = Comparator 1 positive voltage is larger than CM1N voltage.

Bit 4 **CM1SF:** Comparator 1 special mode control bit.
0 = Disable. Comparator 1 is normal comparator function.
1 = Enable. Comparator 1 output edge triggers TC0 pulse generator stopping output (surge protection).

Bit 3 **CM1G:** Comparator output trigger direction control bit.
0 = Falling edge trigger. Comparator output status is from high to low as V+ < V-
1 = Rising edge trigger. Comparator output status is from low to high as V+ > V-.

Bit [2:0] **CM1RS [2:0]:** Comparator positive terminal’s internal reference voltage select bit.
000 = Internal 0.6*Vref.
011 = Internal (0.6+0.2/7)*Vref.
010 = Internal (0.6+0.4/7)*Vref.
011 = Internal (0.6+0.6/7)*Vref.
100 = Internal (0.6+0.8/7)*Vref.
101 = Internal (0.6+1/7)*Vref.
110 = Internal (0.6+1.2/7)*Vref.
111 = Internal 0.8*Vref.

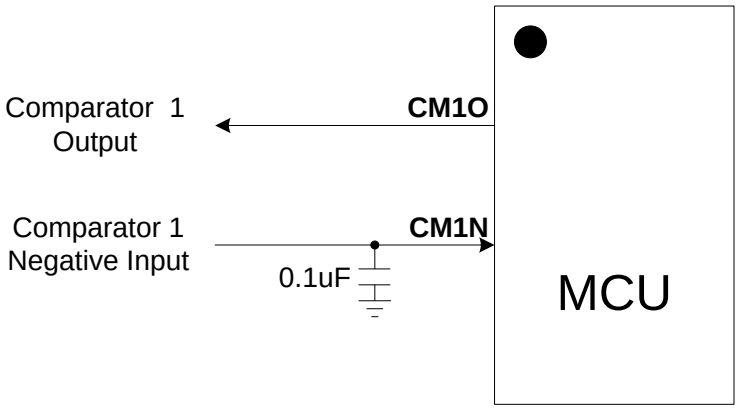
09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMDB1	-	-	-	-	-	CM1REF	CM1D1	CM1D0
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After Reset	-	-	-	-	-	0	0	0

Bit 2 **CM1REF:** Comparator 1 positive internal reference voltage power source select bit.
0 = Internal reference voltage power source is VDD.
1 = Internal reference voltage power source is 4V.

Bit [1:0] **CM1D [1:0]:** Comparator 1 de-bounce time control bit.
00=No de-bounce, 01=1/Fcpu, 10=2/Fcpu, 11=4/Fcpu.

11.5 COMPARATOR 1 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.



11.6 COMPARATOR 1 OPERATION EXPLAME

➢ **COMPARATOR 1 CONFIGURATION:**

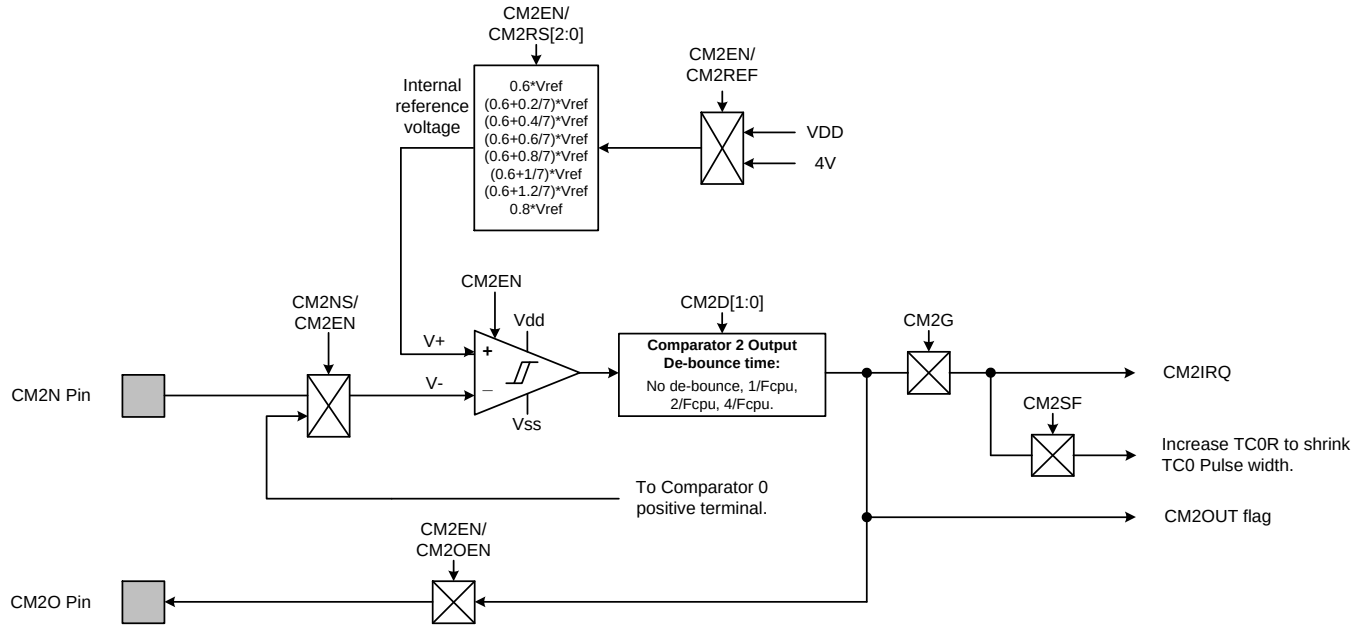
```
; Reset Comparator 1.
        CLR     CM1M            ; Clear CM1M register.

; Set Comparator 1 positive terminal.
        MOV     A, #00000nnnb   ; Set CM1RS[2:0] for comparator positive terminal.
        B0MOV   CM1M, A

; Set Comparator 1 function mode.
        B0BCLR  FCM1SF          ; Normal comparator mode.
; or
        B0BSET  FCM1SF          ; Special function mode.

; Set Comparator 1 output pin.
        B0BCLR  FCM1OEN         ; Disable comparator 1 output pin.
; or
        B0BSET  FCM1OEN         ; Enable comparator 1 output pin.

; Set Comparator 1 interrupt trigger edge.
        B0BCLR  FCM1G           ; Falling edge.
; or
        B0BSET  FCM1G           ; Rising edge.

; Set Comparator 1 positive terminal's internal reference voltage source.
        B0BCLR  FCM1REF         ; Internal Vdd.
or
        B0BSET  FCM1REF         ; Internal 4V.

; If FCM1REF = 1, Comparator 1 positive terminal's internal 4V reference voltage power must be enabled.
        B0BSET  FINTR4V         ; Enable internal 4V reference voltage power.

; Set Comparator 1 output de-bounce.
        B0MOV   A, CMDB1        ; Set CM1D[1:0] for comparator output de-bounce.
        AND     A, #00000100b
        OR      A, #000000nnb
        B0MOV   CMDB1, A

; Set Comparator 1 pin configuration.
        B0MOV   A, CMCON        ; Set CMCON.2 for isolate GPIO path.
        OR      A, #00000100b
        B0MOV   CMCON, A

; Clear CM1IRQ
        B0BCLR  FCM1IRQ

; Enable Comparator 1 and interrupt function.
        B0BSET  FCM1IEN         ; Enable Comparator 1 interrupt function.
        B0BSET  FCM1EN          ; Enable Comparator 1.
```

12 ANALOG COMPARAOTR 2

12.1 OVERVIEW

The micro-controller builds in one comparator with shrinking TC0 pulse width function. The comparator has normal comparator mode and shrinking TC0 pulse width trigger source. The comparator is not Rail-to-Rail structure. That means the input/output voltage is not real from Vdd~Vss (Reference to “Electrical characteristics” chapter). When the positive input voltage is greater than the negative input voltage, the comparator output is high. When the positive input voltage is smaller than the negative input voltage, the comparator output is low. The comparator builds in internal reference voltage connected to comparator positive terminal. The main purposes of comparator 2 are as following.

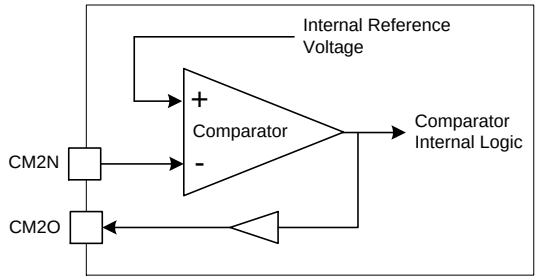
- **Normal comparator function:** General comparator mode compares the two tensions of positive input terminal and negative input terminal.
- **Interrupt function:** Comparator 2 supports interrupt function. When comparator 2 output edge direction equals to edge selection, the CM2IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **TC0 pulse width shrinking source:** Comparator 2 can be TC0 pulse width shrinking trigger source controlled by CM2SF bit. When TC0PO = 1 and CM2SF = 1, comparator 2 output status triggers to shrink TC0 pulse width through increasing TC0R register. TC0 pulse width shrinking unit time is controlled by PWS[1:0].
- **Green mode function:** Comparator 2 still actives in green mode, but no wake-up function. CM2IRQ can be latched as trigger event occurrence until system wakes up. After system wakes up, the comparator 2 interrupt service routine is executed by program.



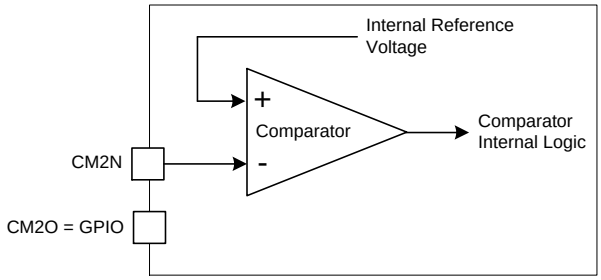
12.2 NORMAL COMPARATOR MODE

12.2.1 COMPARATOR ENABLE

Comparator pins are shared with GPIO controlled by CM2EN bit. When CM2EN=1 and CM2NS=0, CM2N pin is enabled connected to comparator negative terminal When CM2EN=1and CM2NS=1, CM0P pin is connected to comparator 2 negative terminal with internal automatically, CM2N pin is GPIO function. Comparator positive terminal is connected to internal reference voltage. The internal reference voltage is controlled by CM2RS [2:0] bits. When CM2RS [2:0]=000b~111b, comparator positive terminal is connected to the internal reference voltage. The internal reference voltage includes 8-level which are 0.6*Vref, (0.6+0.2/7)*Vref, (0.6+0.4/7)*Vref, (0.6+0.6/7)*Vref, (0.6+0.8/7)*Vref, (0.6+1/7)*Vref, (0.6+1.2/7)*Vref, 0.8*Vref. The internal reference voltage power source is Vdd or 4V controlled by CM2REF bit. CM2OEN controls comparator output to CM2O pin. When CM2EN = 1 and CM2OEN = 1, CM2O pin is comparator 2 output status. When CM2EN = 1 and CM2OEN = 0, comparator output status can be read only through CM2OUT flag.



CM2EN = 1, CM2OEN = 1, CM2RS[2:0] = 000b~111b



CM2EN = 1, CM2OEN = 0, CM2RS[2:0] = 000b~111b

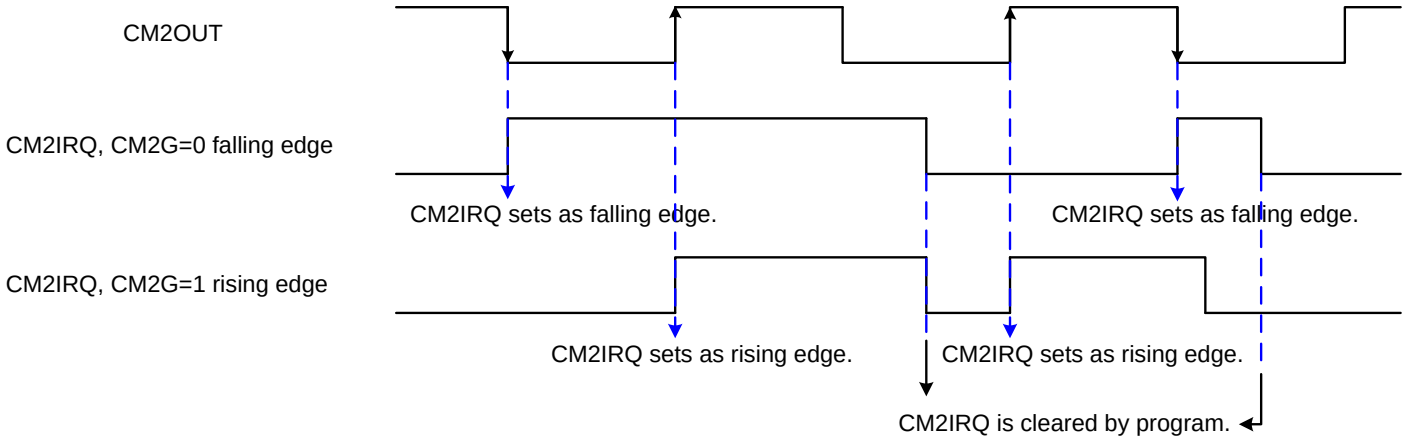
- ***Note: The comparator enable condition is fixed CM2EN=1, or the comparator pins are GPIO mode and comparator is disabled.***

12.2.2 CM2OUT, CM1G AND CM1IRQ STATUS

The CM2OUT and CM2IRQ bits indicate the comparator result. The CM2OUT shows the comparator result immediately, but the CM2IRQ only indicates the event of the comparator result. The event condition is controlled by register and includes rising edge (CM2OUT changes from low to high) and falling edge (CM2OUT changes from high to low) controlled by CM2G bit. When CM2G = 0, the comparator 2 interrupt trigger direction is falling edge. When CM2G = 1, the comparator 2 interrupt trigger direction is rising edge.

- ***Note: CM2OUT is comparator raw output without latch. It varies depend on the comparator process result. But the CM2IRQ is latch comparator output result. It must be cleared by program.***

Comparator supports interrupt function. The interrupt trigger condition can be selected through CM2G bit including rising edge and falling edge. If CM2G = 0, comparator output trigger edge is falling edge. If CM2G = 1, comparator output trigger edge is rising edge. The edge detection is from comparator output signal through delay processor. When comparator output edge event occurs and equal CM2G condition, CM2IRQ flag is issued. If CM2IEN = 1, program counter points to interrupt vector to execute interrupt service routine.

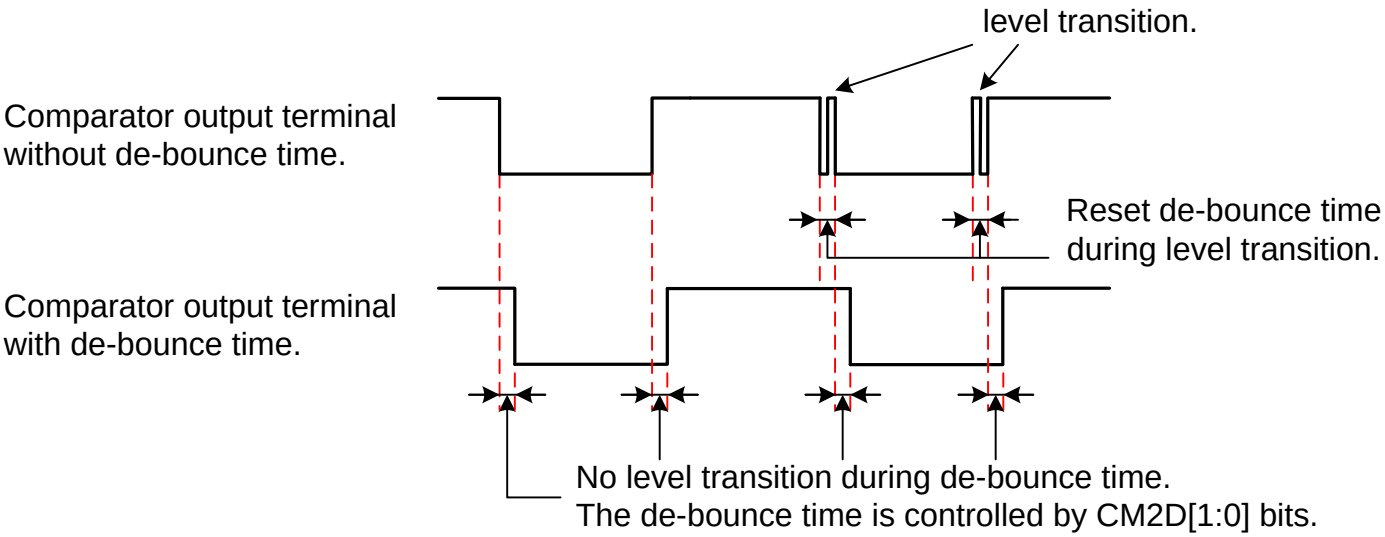


12.2.3 DE-BOUNCE TIME CONTROL

Comparator 2 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. Comparator output terminal builds in de-bounce time control block to achieve output hysteresis to filter output transition condition. The de-bounce time option has 4-step including no de-bounce, 1/Fcpu, 2/Fcpu, 4/Fcpu controlled by CM2D [1:0] bits.

CM2D[1:0]	**00b**	**01b**	**01b**	**11b**
	No de-bounce	1/Fcpu	2/Fcpu	4/Fcpu
De-bounce time (us) Fcpu=Fhosc/4 =16MHz/4=4MHz	0	0.25	0.5	1
De-bounce time (us) Fcpu=Fhosc/16 =16MHz/16=1MHz	0	1	2	4

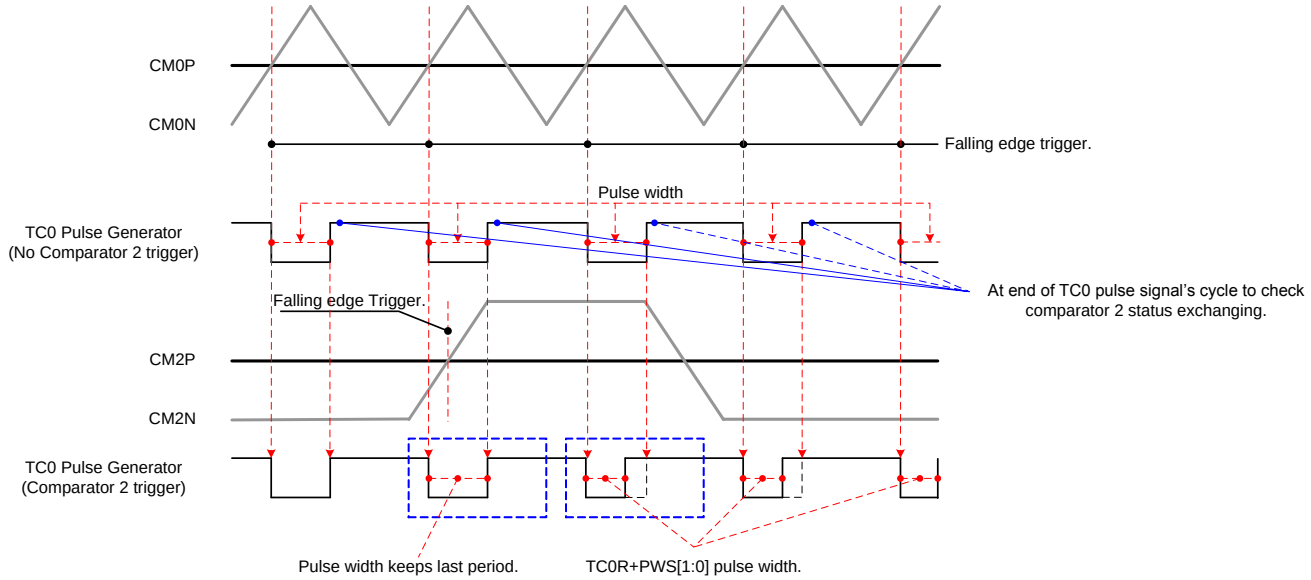
Comparator output terminal:



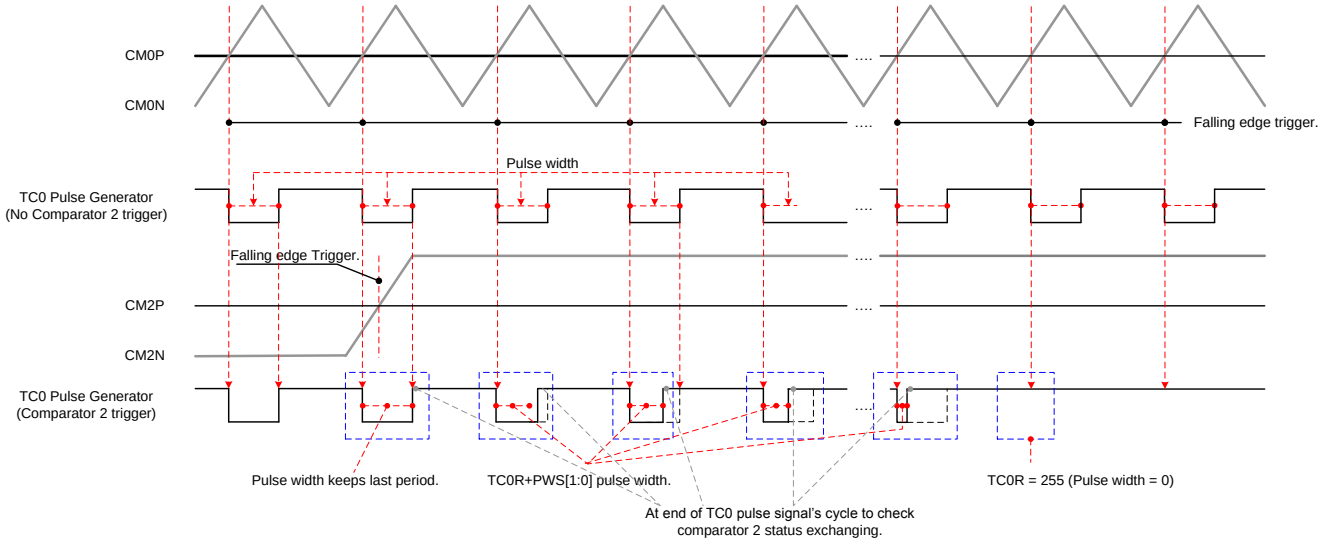
12.3 COMPARATOR 2 SPECIAL FUCNITON

Besides normal comparator function, comparator 2 builds in a special mode to shrink TC0 pulse width through increasing TC0R register. The special mode is to trigger increasing TC0R register and TC0 pulse width will be shrunk through comparator output edge and controlled by CM2SF bit. When CM2SF = 1, comparator 2 special mode is enabled. If comparator 2 output trigger condition occurs, TC0R register increases unit time to shrink TC0 pulse width. The unit time is controlled by PWS[1:0].

Step 1: When comparator 2 first trigger occurs, TC0R + PWS[1:0] once to shrink TC0 pulse width. TC0 pulse width reduces unit time automatically. The hardware checks comparator 2 high tension status at the end of TC0 pulse signal's cycle. If the comparator output status exchanges, to expand TC0 pulse width through increasing TC0R register by program.



Step 2: If comparator output status doesn't exchange, TC0R + PWS[1:0] at the end of one TC0 pulse signal's cycle and outputs the new pulse until comparator 2 output status exchanges until TC0R = 255.



✸ ***Note: If TC0R is increased to 0xFF, TC0R will keep 0xFF and not increase again, even the comparator output status never occurs exchanging.***

12.4 COMPARATOR 2 MODE REGISTER

09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM2M	CM2EN	CM2OEN	CM2OUT	CM2SF	CM2G	CM2RS2	CM2RS1	CM2RS0
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 7 **CM2EN:** Comparator 2 control bit.
0 = Disable. Comparator pins are GPIO mode.
1 = Enable. CM2N pin is comparator negative input pin, and GPIO function is isolated. CM2O is controlled by CM2OEN bit. Comparator 2 positive terminal is connected to internal reference voltage. The internal reference voltage is selected by CM2RS [2:0] bits.

Bit 6 **CM2OEN:** Comparator 2 output pin control bit.
0 = Disable. CM2O pin is GPIO mode.
1 = Enable. CM2O pin is comparator output and isolate GPIO function.

Bit 5 **CM2OUT:** Comparator 2 output flag bit.
0 = Comparator 2 positive voltage is less than CM2N voltage (CM2EN = 0, CM2OUTdefault value is “0”).
1 = Comparator 2 positive voltage is larger than CM2N voltage.

Bit 4 **CM2SF:** Comparator 2 special mode control bit.
0 = Disable. Comparator 2 is normal comparator function.
1 = Enable. Comparator 2 output edge triggers Shrinking TC0 pulse width.

Bit 3 **CM2G:** Comparator output trigger direction control bit.
0 = Falling edge trigger. Comparator output status is from high to low as V+ < V-.
1 = Rising edge trigger. Comparator output status is from low to high as V+ > V-.

Bit [2:0] **CM2RS [2:0]:** Comparator positive terminal’s internal reference voltage select bit.
000 = Internal 0.6*Vref.
011 = Internal (0.6+0.2/7)*Vref.
010 = Internal (0.6+0.4/7)*Vref.
011 = Internal (0.6+0.6/7)*Vref.
100 = Internal (0.6+0.8/7)*Vref.
101 = Internal (0.6+1/7)*Vref.
110 = Internal (0.6+1.2/7)*Vref.
111 = Internal 0.8*Vref.

09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMDB2	-	-	PWS1	PWS0	CM2REF	CM2NS	CM2D1	CM2D0
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	0	0	0	0	0	0

Bit [5:4] **PWS [1:0]:** Comparator 2 output de-bounce unit time select bits.
00b = 1 unit time, 01b = 2 unit times, 10b = 4 unit times, 11b = 8 unit times.

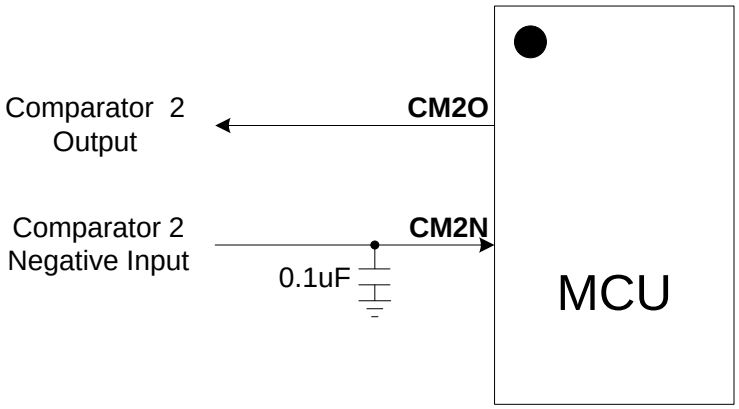
Bit 3 **CM2REF:** Comparator 2 positive internal reference voltage power source select bit.
0 = Internal reference voltage source is VDD.
1 = Internal reference voltage source is 4V.

Bit 2 **CM2NS:** Comparator 2 negative input select bit.
0 = CM2N pin is comparator negative input pin, and GPIO function is isolated.
1 = CM2N pin is GPIO function and comparator 0 positive (V+) terminal is connected to comparator 2 negative terminal automatically (internal connect).

Bit [1:0] **CM2D[1:0]:** Comparator 2 de-bounce time control bit.
00=No de-bounce, 01=1/Fcpu, 10=2/Fcpu, 11=4/Fcpu.

12.5 COMPARATOR 2 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.



12.6 COMPARATOR 2 OPERATION EXPLAME

➢ **COMPARATOR 2 CONFIGURATION:**

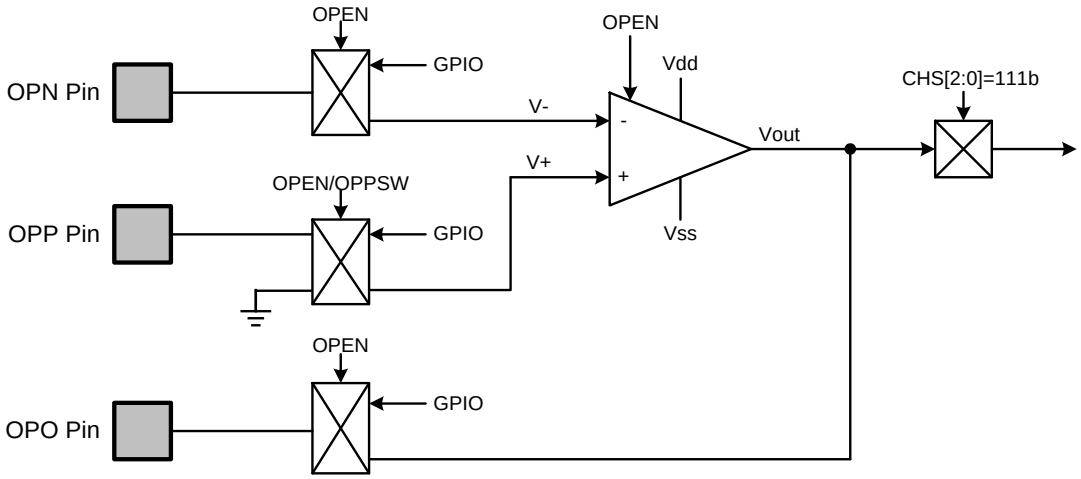
```
; Reset Comparator 2.
        CLR     CM2M            ; Clear CM2M register.

; Set Comparator 2 positive terminal.
        MOV     A, #00000nnnb   ; Set CM2RS[2:0] for comparator positive terminal.
        B0MOV   CM2M, A

; Set Comparator 2 function mode.
        B0BCLR  FCM2SF          ; Normal comparator mode.
; or
        B0BSET  FCM2SF          ; Special function mode.

; Set Comparator 2 output pin.
        B0BCLR  FCM2OEN         ; Disable comparator 2 output pin.
; or
        B0BSET  FCM2OEN         ; Enable comparator 2 output pin.

; Set Comparator 2 interrupt trigger edge.
        B0BCLR  FCM2G           ; Falling edge.
; or
        B0BSET  FCM2G           ; Rising edge.

; Set Comparator 2 positive terminal's internal reference voltage source.
        B0BCLR  FCM2REF         ; Internal Vdd.
or
        B0BSET  FCM2REF         ; Internal 4V.

; If FCM2REF = 1, Comparator 2 positive terminal's internal 4V reference voltage power must be enabled.
        B0BSET  FINTR4V         ; Enable internal 4V reference voltage power.

; Set Comparator 2 negative input source.
        B0MOV   A, CMDB2        ; Set CM2NS for comparator negative input source.
```

```
        AND         A, #00111011b
        OR          A, #00000100b
        B0MOV       CMDB2, A

; Set Comparator 2 output de-bounce.
        B0MOV       A, CMDB2        ; Set CM2D[1:0] for comparator output de-bounce.
        AND         A, #00111100b
        OR          A, #000000nnb
        B0MOV       CMDB2, A

; Set Comparator 2 output de-bounce unit time.
        B0MOV       A, CMDB2        ; Set PWS[1:0] for comparator output de-bounce unit time.
        AND         A, #00001111b
        OR          A, #00nn00nnb
        B0MOV       CMDB2, A

; Set Comparator 2 pin configuration.
        B0MOV       A, CMCON        ; Set CMCON.3 for isolate GPIO path.
        OR          A, #00001000b
        B0MOV       CMCON, A

; Clear CM2IRQ
        B0BCLR      FCM2IRQ

; Enable Comparator 2 and interrupt function.
        B0BSET      FCM2IEN         ; Enable Comparator 2 interrupt function.
        B0BSET      FCM2EN          ; Enable Comparator 2.
```

13 OPERATIONAL AMPLIFIER

13.1 OVERVIEW

The micro-controller builds in one operational amplifier (OP-Amp). The OP-Amp power range is Vss~Vdd. OP-Amp input signal and output voltage are within the voltage range. The OP-Amp output pin is programmable to connect with ADC input channel for voltage measurement. The OP-Amp positive pin is programmable to connect internal ground for inverting amplifier application. The OP-Amp pins are shared with GPIO controlled by OPEN/OPPSW bits.



OP- Amp pins selection table is as following:

OPEN bit	OPP Positive Pin (V+)	OPN Negative Pin (V-)	OPO Output Pin (Vout)
OPEN=0 (OP-amp disables)	OPP pin is P4.2 GPIO mode.	OPN pin is P4.1 GPIO mode.	OPO pin is P4.0 GPIO mode.
OPEN=1 (OP-amp enables)	OPPSW=0: OP-amp's V+ terminal is connected to OPP pin and isolate GPIO path. OPPSW= 1 : OP-amp's V+ terminal is connected to internal ground, and OPP pin is P4.2 GPIO path.	OP-amp's V+ terminal is connected to OPN pin isolated GPIO path.	CHS[2:0]≠ 111b: Vout is connected to OPO pin and isolate GPIO path. CHS[2:0]= 111b: Vout is connected to OPO pin and ADC input channel AIN7, isolate GPIO path.

✸ ***Note: If OP-AMP disables, these pins exchange to GPIO mode and last status.***

13.2 OP-AMP REGISTER

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

Bit 1 **OPPSW:** OP-Amp positive input source select bit.
0 = OPP pin (P4.2) is OP-Amp positive input, GPIO mode and ADC AIN2 input channel are isolated.
1 = OPP pin (P4.2) is GPIO mode or ADC AIN2 input channel. OP-Amp positive terminal connect to internal ground.

Bit 0 **OPEN:** OP-Amp control bit.
0 = Disable. P4.0, P4.1, P4.2 are GPIO mode.
1 = Enable. P4.0, P4.1, P4.2 are OP-Amp pins selected by OPPSW bit and CHS[2:0] bits.

13.3 OP-AMP OPERATION EXPLAME

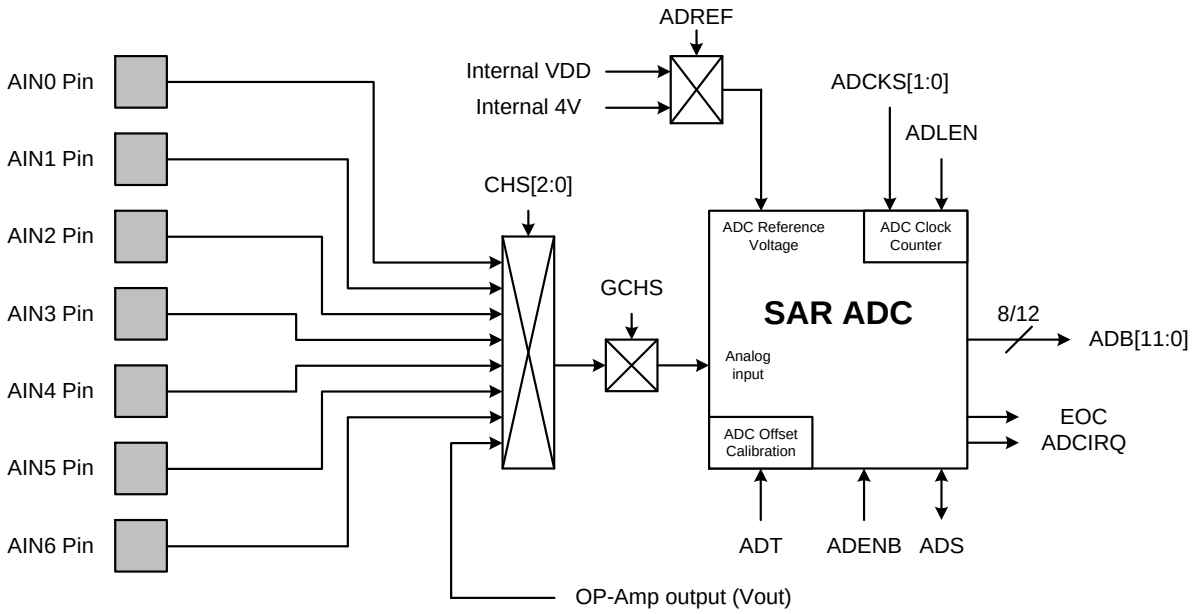
➢ **OP-AMP CONFIGURATION:**

```
; Reset OP-AMP.
        CLR     OPM             ; Clear OPM register.

; Set OPP pin mode.
        B0BCLR  FOPPSW          ; OPP pin (P4.2) is OP-Amp positive input terminal
; or
        B0BSET  FOPPSW          ; OPP pin (P4.2) is GPIO mode or ADC AIN2 input channel.
                                ; OP-Amp positive terminal connect to internal ground.

; Set OP-AMP pin configuration (OPPSW = 0).
        B0MOV   A, P4CON        ; Set P4CON for isolate GPIO path.
        OR      A, #00000111b
        B0MOV   P4CON, A

        B0MOV   A, P4M          ; Set P4.0~P4.2 are input mode.
        AND     A, #11111000b
        B0MOV   P4M, A

; Set OP-AMP pin configuration (OPPSW = 1).
        B0MOV   A, P4CON        ; Set P4CON for isolate GPIO path.
        AND     A, #11111100b
        OR      A, #00000011b
        B0MOV   P4CON, A

        B0MOV   A, P4M          ; Set P4.0~P4.1 are input mode.
        AND     A, #11111100b
        B0MOV   P4M, A

; Enable OP-AMP function.
        B0BSET  FOPEN           ; Enable OP-AMP.
```

14 7+1 CHANNEL ANALOG TO DIGITAL CONVERTER (ADC)

14.1 OVERVIEW

The analog to digital converter (ADC) is SAR structure with 7+1-input sources (7-channels: P4.0~P4.6, 1-channel: OP-Amp output terminal) 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~AIN6, OP-Amp output terminal) to measure 8 different analog signal sources controlled by CHS [2:0] and GCHS bits. The ADC resolution can be selected 8-bit and 12-bit resolutions through ADLEN bit. The ADC converting rate can be selected by ADCKS [1:0] bits to decide ADC converting time. The ADC reference high voltage includes two sources controlled by ADREF bit. One is internal reference voltage Vdd (ADREF = 0), and the other one is internal 4V reference voltage (ADREF = 1). The ADC builds in P4CON register to set pure analog input pin. It is necessary to set P4 as input mode without pull-up resistor by program. After setup ADENB and ADS bits, the ADC starts to convert analog signal to digital data. When the conversion is complete, the ADC circuit will set EOC and ADCIRQ bits to “1” and the digital data outputs in ADB and ADR registers. If the ADCIEN = 1, the ADC interrupt request occurs and executes interrupt service routine when ADCIRQ = 1 after ADC converting. If ADC interrupt function is enabled (ADCIEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 8H) and executes interrupt service routine after finishing ADC converting. Clear ADCIRQ by program is necessary in interrupt procedure.



14.2 ADC OPERATION DESCRIPTION AND NOTIC

14.2.1 ADC SIGNAL FORMAT

ADC sampling voltage range is limited by high/low reference voltage. The ADC low reference voltage is Vss and not changeable. The ADC high reference voltage includes internal Vdd and internal 4V reference voltage source controlled by ADREF bit. If ADREF = 0, ADC reference voltage is from internal Vdd (MCU power voltage). If ADREF = 1, ADC reference voltage is from internal 4V voltage source. 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 internal high reference voltage range.

- **ADC Internal Low Reference Voltage = 0V.**
- **ADC Internal High Reference Voltage = Vdd (ADREF = 0) or 4V (ADREF = 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**

14.2.2 ADC CONVERTING TIME

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

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

ADLEN	ADCKS1, ADCKS0	ADC Clock Rate	Fcpu=4MHz		Fcpu=1MHz	
			ADC Converting time	ADC Converting Rate	ADC Converting time	ADC Converting Rate
1 (12-bit)	00	Fcpu/16	1/(4MHz/16/4)*16 = 256 us	3.906KHz	1/(1MHz/16/4)*16 = 1024 us	0.9767KHz
	01	Fcpu/8	1/(4MHz/8/4)*16 = 128 us	7.813KHz	1/(1MHz/8/4)*16 = 512 us	1.9531KHz
	10	Fcpu	1/(4MHz/1/4)*16 = 16 us	62.5KHz	1/(1MHz/1/4)*16 = 64 us	15.625KHz
	11	Fcpu/2	1/(4MHz/2/4)*16 = 32 us	31.25KHz	1/(1MHz/2/4)*16 = 128 us	7.8125KHz

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

ADLEN	ADCKS1, ADCKS0	ADC Clock Rate	Fcpu=4MHz		Fcpu=1MHz	
			ADC Converting time	ADC Converting Rate	ADC Converting time	ADC Converting Rate
0 (8-bit)	00	Fcpu/16	1/(4MHz/16/4)*12 = 192 us	5.208KHz	1/(1MHz/16/4)*12 = 768 us	1.3021KHz
	01	Fcpu/8	1/(4MHz/8/4)*12 = 96 us	10.416KHz	1/(1MHz/8/4)*12 = 384 us	2.6042KHz
	10	Fcpu	1/(4MHz/1/4)*12 = 12 us	83.333KHz	1/(1MHz/1/4)*12 = 48 us	20.8333KHz
	11	Fcpu/2	1/(4MHz/2/4)*12 = 24 us	41.667KHz	1/(1MHz/2/4)*12 = 96 us	10.4167KHz

14.2.3 ADC PIN CONFIGURATION

ADC input channels are shared with Port4. ADC channel selection is through CHS[2:0] bits. 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 port 4 can be configured as ADC input in the same time. The pins of Port4 configured as ADC input channel must be set input mode, disable internal pull-up and enable P4CON first by program. After selecting ADC input channel through 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.**
- **P4CON bits of ADC input channel must be set.**

ADC input pins are shared with digital I/O pins. Connect an analog signal to COMS digital input pin, especially, the analog signal level is about 1/2 VDD will cause extra current leakage. In the power down mode, the above leakage current will be a big problem. Unfortunately, if users connect more than one analog input signal to port 4 will encounter above current leakage situation. P4CON is Port4 configuration register. Write "1" into P4CON [6:0] will configure related port 4 pin as pure analog input pin to avoid current leakage.

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

Bit[6:0] **P4CON[6:0]:** P4.n configuration control bits.
0 = P4.n can be an analog input (ADC input) or digital I/O pins.
1 = P4.n is pure analog input, can't be a digital I/O pin.

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

14.3 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	ADREF	CHS2	CHS1	CHS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	1	0	0	0	0	0

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

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

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

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

Bit 3 **ADREF:** ADC high reference voltage source control bit.
0 = Internal reference voltage is VDD.
1 = Internal reference voltage is 4V.

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 = OP-AMP output terminal.

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	INTR4V	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
After reset	0	0	0	0	-	-	-	-

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

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

14.4 ADC DATA BUFFER REGISTERS

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

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

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
Read/Write	R	R	R	R	R	R	R	R
After reset	-	-	-	-	-	-	-	-

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

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	INTR4V	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
After reset	0	0	0	0	-	-	-	-

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

The AIN input voltage v.s. ADB output data

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

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

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

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

14.5 ADC OPERATION EXAMLPE

➢ **ADC CONFIGURATION:**

```
; Reset ADC.
        CLR     ADM             ; Clear TC0M register.

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

; Set ADC high reference voltage source.
        B0BCLR  FADREF          ; Internal Vdd.
or
        B0BSET  FADREF          ; Internal 4V.

; If FADREF = 1, ADC internal 4V reference voltage power must be enabled..
        B0BSET  FINTR4V         ; Enable internal 4V reference voltage power.

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

; Enable ADC.
        B0BSET  FADCENB

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

; Select ADC input channel.
        MOV     A, #value       ; Set 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:***

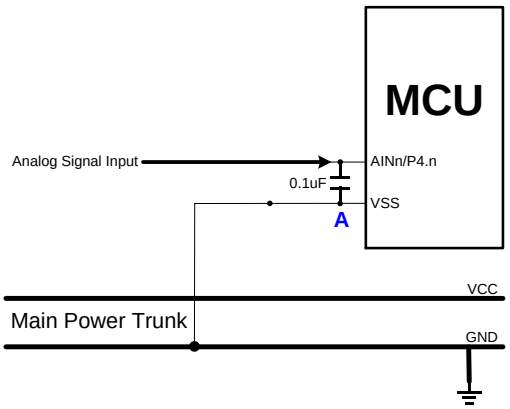
1. ***When ADENB is enabled, the system must be delay 100us to be the ADC warm-up time by program, and then set ADS to do ADC converting. The 100us delay time is necessary after ADENB setting (not ADS setting), or the ADC converting result would be error. Normally, the ADENB is set one time when the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

14.6 ADC APPLICATION CIRCUIT

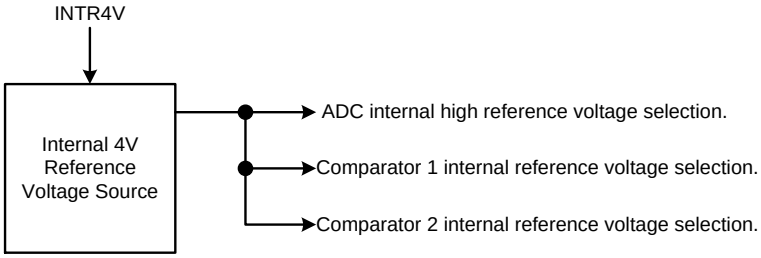


The analog signal is inputted to ADC input pin "AINn/P4.n". The ADC input signal must be through a 0.1uF capacitor "A". The 0.1uF capacitor is set between ADC input pin and VSS pin, and must be on the side of the ADC input pin as possible. Don't connect the capacitor's ground pin to ground plain directly,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

15 INTERNAL 4V REFERENCE VOLTAGE

15.1 OVERVIEW

The micro-controller builds in one internal 4V reference voltage source which can be ADC high reference voltage source and comparator 1/2 positive reference voltage source. The internal 4V reference voltage is a high precision voltage source after trimming. The trimmed accuracy is 4V (Toleaance = +2% ~ -2%). It is regulated design, so the power voltage must be greater than 4.5V, or the output voltage is equal to Vdd. The internal 4V reference voltage is controlled by INTR4V control bit. When INTR4V sets as “1”, the internal 4V reference voltage is enabled and outputs to the three functions’ reference voltage selection control block. ADC’s selection bit is ADREF bit. Comparator 1 selection bit is CM1REF bit. Comparator 2 selection bit is CM2REF bit. If INTR4V isn’t enabled, but the tree functions’ select it, the functions’ reference voltage would be error.



15.2 INTERNAL 4V REFERENCE VOLTAGE CONTROL REGISTER

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	INTR4V	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

Bit 7 INTR4V**:** Internal 4V reference voltage control bit.
0 = Disable internal 4V reference voltage.
1 = Enable internal 4V reference voltage.

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMDB1	-	-	-	-	-	CM1REF	CM1D1	CM1D0
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After Reset	-	-	-	-	-	0	0	0

Bit 2 **CM1REF:** Comparator 1 positive internal reference voltage power source select bit.
0 = Reference voltage source is internal VDD.
1 = Reference voltage source is internal 4V reference voltage.

09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMDB2	-	-	PWS1	PWS0	CM2REF	CM2NS	CM2D1	CM2D0
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	0	0	0	0	0	0

Bit 3 **CM2REF:** Comparator 2 positive internal reference voltage power source select bit.
0 = Reference voltage source is internal VDD.
1 = Reference voltage source is internal 4V reference voltage.

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	ADREF	CHS2	CHS1	CHS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit 4 ADREF**:** ADC internal reference voltage select bit.
0 = High reference voltage source is internal VDD.
1 = High reference voltage source is internal 4V reference voltage.

15.3 INTERNAL 4V REFERENCE VOLTAGE OPERATION EXPLAME

➢ **INTERNAL 4V REFERENCE VOLTAGE CONFIGURATION:**

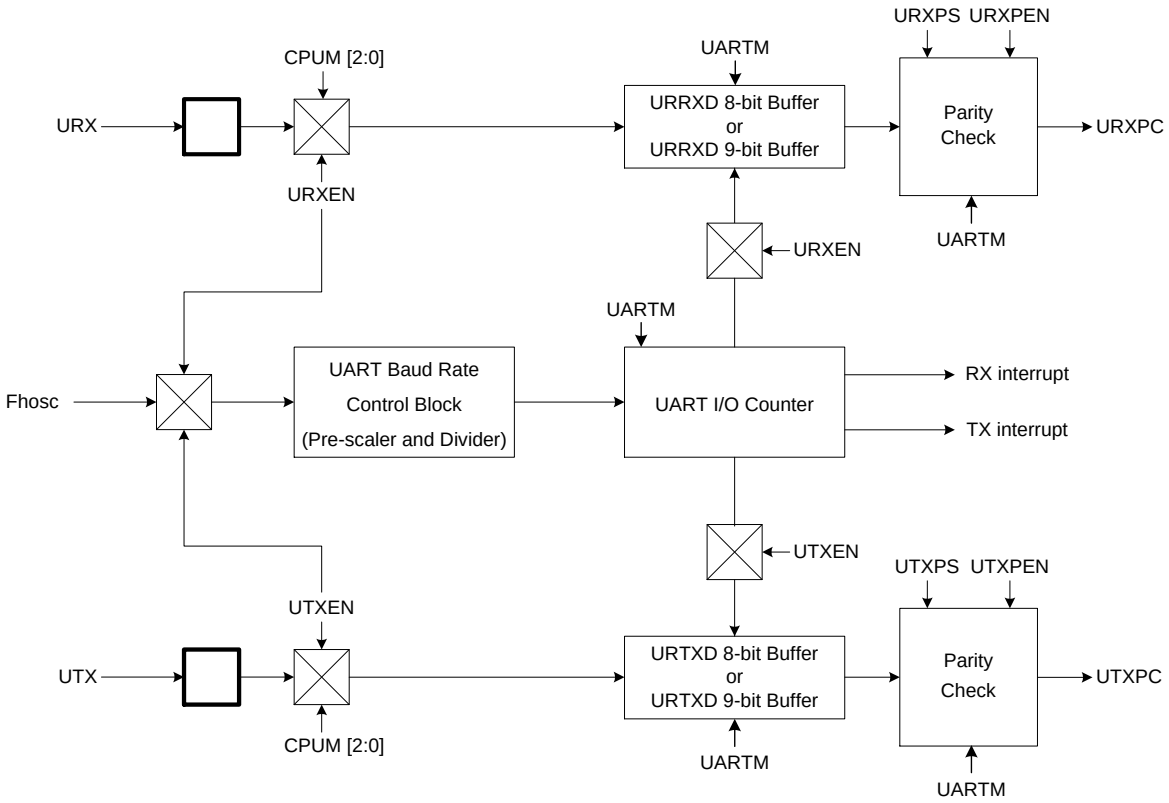
```
; Enable internal 4V reference voltage.
        B0BSET      FINTR4V         ; Enable internal 4V reference voltage..

; Set Comparator 1 positive terminal internal reference voltage = internal 4V reference voltage
        B0BSET      FCM1REF         ; Comparator 1 reference voltage = 4V.

; Set Comparator 2 positive terminal internal reference voltage = internal 4V reference voltage
        B0BSET      FCM2REF         ; Comparator 2 reference voltage = 4V.

; Set ADC high reference voltage = internal 4V reference voltage
        B0BSET      FADREF          ; AD internal reference voltage = 4V.

; Disable internal 4V reference voltage.
        B0BCLR      FINTR4V         ; Disable internal 4V reference voltage..
```

16 UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER (UART)

16.1 OVERVIEW

The UART interface is a universal asynchronous receiver/transmitter method. The serial interface is applied to low speed data transfer and communicate with low speed peripheral devices. The UART transceiver of Sonix 8-bit MCU allows RS232 standard supporting one byte data length and 9-bit data length. The transfer format has start-bit, 8-bit or 9-bit data, parity bit and stop bit. Programmable baud rate supports different speed peripheral devices. UART I/O pins support push-pull and open-drain structures. The UART features include the following:

- **Full-duplex, 2-wire asynchronous data transfer.**
- **Programmable baud rate.**
- **8-bit /9-bit data length.**
- **Odd and even parity bit.**
- **End-of-Transfer interrupt.**
- **Support wide range baud rate.**



UART Interface Structure Diagram

16.2 UART OPERATION

The UART RX and TX pins are shared with GPIO. When UART enables (URXEN=1, UTXEN=1), the UART shared pins transfers to UART purpose and disable GPIO function automatically. When UART disables, the UART pins returns to GPIO last status. The UART data buffer length supports 8-byte (UARTM = 0) and 9-bit (UARTM = 1).

The UART supports interrupt function. URXIEN/UTXIEN is UART transfer interrupt function control bit. URXIEN=0, disable UART receiver interrupt function. UTXIEN=0, disable UART transmitter interrupt function. URXIEN=1, enable UART receiver interrupt function. UTXIEN=1, enable UART transmitter interrupt function. When UART interrupt function enable, the program counter points to interrupt vector (ORG 0008H) to do UART interrupt service routine after UART operating. URXIRQ/UTXIRQ is UART interrupt request flag, and also to be the UART operating status indicator when URXIEN=0 or UTXIEN=0, but cleared by program. When UART operation finished, the URXIRQ/UTXIRQ would be set to “1”.

The UART also builds in “Busy Bit” to indicate UART bus status. URXBZ bit is UART RX operation indicator. UTXBZ bit is UART TX operation indicator. If bus is transmitting, the busy bit is “1” status. If bus is finishing operation or in idle status, the busy bit is “0” status.

UART TX operation is controlled by loading UTXD data buffer. After UART TX configuration, load transmitted data into UTXD 8-bit buffer, and then UART starts to transmit the pocket following UART TX configuration.

UART RX operation is controlled by receiving the start bit from master terminal. After UART RX configuration, URX pin detects the falling edge of start bit, and then UART starts to receive the pocket from master terminal.

UART provides URXPC bit and UFMER bit to check received pocket. URXPC bit is received parity bit checker. If received parity is error, URXPC sets as “1”. If URXPC bit is zero after receiving pocket, the parity is correct. UFMER bit is received stream frame checker. The stream frame error definition includes “Start bit error”, “Stop bit error”, “Stream length error”, “UART baud rate error”... Each of frame error conditions makes UFMER bit sets as “1” after receiving pocket.

✷ ***Note: When UARTM = 0, moved data to UTXD register by program. The Uart TX starts transition. When UARTM = 1, set as UTXD8 bit first by program, then moved data to UTXD register by program. The Uart TX starts transition.***

16.3 UART BAUD RATE

UART clock is 2-stage structure including a pre-scaler and an 8-bit buffer. UART clock source is generated from system oscillator called Fhosc. Fhosc passes through UART pre-scaler to get UART main clock called Fuart. UART pre-scaler has 8 selections (Fhosc/1, Fhosc/2, Fhosc/4, Fhosc/8, Fhosc/16, Fhosc/32, Fhosc/64, Fhosc/128) and 3-bit control bits (URS [2:0]). UART main clock (Fuart) purposes are the front-end clock and through UART 8-bit buffer (URCR) to obtain UART processing clock and decide UART baud rate.

UART Pre-scaler Selection, URS[2:0]	UART Main Clock Rate	Fuart (Fhosc=16MHz)
000b	Fhosc/1	16MHz
001b	Fhosc/2	8MHz
010b	Fhosc/4	4MHz
011b	Fhosc/8	2MHz
100b	Fhosc/16	1MHz
101b	Fhosc/32	0.5MHz
110b	Fhosc/64	0.25MHz
111b	Fhosc/128	0.125MHz

0E6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URCR	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0
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 UART baud rate clock source is Fhosc and divided by pre-scalar. The equation is as following.

UART Baud Rate = 1/2 *(Fuart * 1/(256 - URCR))…bps

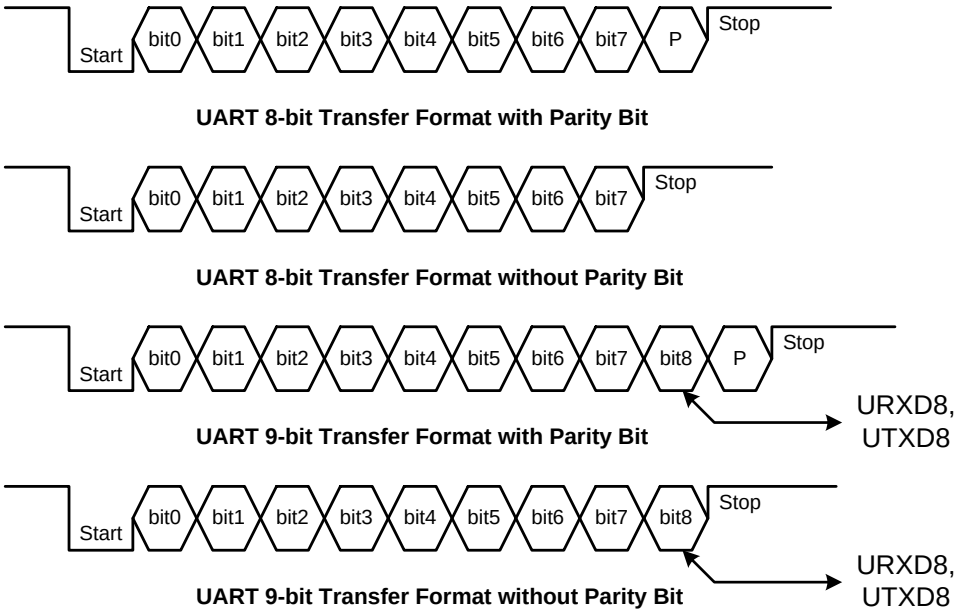
Fhosc = 16MHz

Baud Rate	UART Pre-scaler	URS[2:0]	URCR (Hex)	Accuracy (%)
1200	Fhosc/32	101b	30	-0.16%
2400	Fhosc/32	101b	98	-0.16%
4800	Fhosc/32	101b	CC	-0.16%
9600	Fhosc/32	101b	E6	-0.16%
19200	Fhosc/32	101b	F3	-0.16%
38400	Fhosc/1	000b	30	-0.16%
51200	Fhosc/1	000b	64	-0.16%
57600	Fhosc/1	000b	75	0.08%
102400	Fhosc/1	000b	B2	-0.16%
115200	Fhosc/1	000b	BB	-0.64%
128000	Fhosc/1	000b	C1	0.80%
250000	Fhosc/1	000b	E0	0.00%

✷ ***Note: We strongly recommend not to set URCR = 0xFF, or UART operation would be error.***

16.4 UART TRANSFER FORMAT

The UART transfer format includes “Bus idle status”, “Start bit”, “8-bit or 9-bit Data”, “Parity bit” and “Stop bit” as following.



Bus Idle Status: The bus idle status is the bus non-operating status. The UART receiver bus idle status of MCU is floating status and tied high by the transmitter device terminal. The UART transmitter bus idle status of MCU is high status. The UART bus will be set when URXEN and UTXEN are enabled.

Start Bit: UART is a asynchronous type of communication and need a attention bit to offer receiver the transfer starting. The start bit is a simple format which is high to low edge change and the duration is one bit period. The start bit is easily recognized by the receiver.

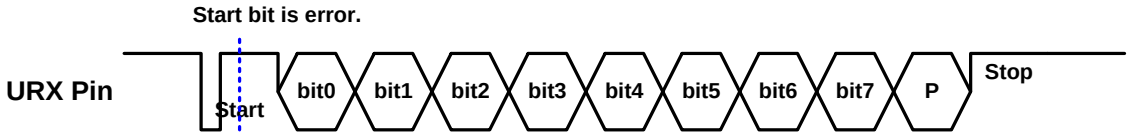
8-bit or 9-bit Data: When UARTM set as 0, the data format is 8-bit length, and LSB transfers first following start bit. The one bit data duration is the unit of UART baud rate controlled by register. When UARTM set as 1, the data format is 9-bit length, and LSB transfers first following start bit. The one bit data duration is the unit of UART baud rate controlled by register.

Parity Bit: The parity bit purpose is to detect data error condition. It is an extra bit following the data stream. The parity bit includes odd and even check methods controlled by URXPS/UTXPS bits. After receiving data and parity bit, the parity check executes automatically. The URXPC bit indicates the parity check result. The parity bit function is controlled by URXPEN/UTXPEN bits. If the parity bit function is disabled, the UART transfer contents remove the parity bit and the stop bit follows the data stream directly.

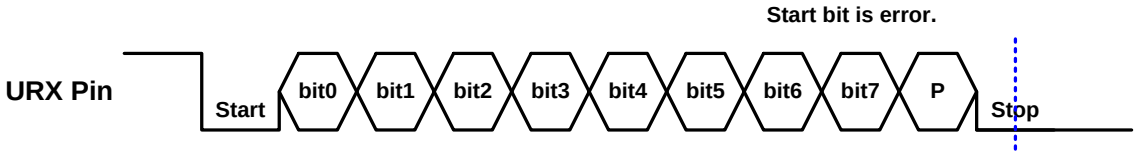
Stop Bit: The stop bit is like start bit using a simple format to indicate the end of UART transfer. The stop bit format is low to high edge change and the duration is one bit period.

16.5 ABNORMAL POCKET

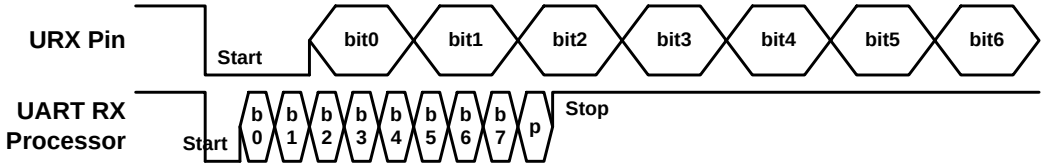
The abnormal pocket occurs in UART RX mode. Break pocket is one abnormal pocket of the UART architecture. The abnormal pocket includes Stream period error, start bit error, stop bit error…When UART receives abnormal pocket, the UFMER bit will be set “1”, and UART issues URXIRQ. The system finds the abnormal pocket through firmware. UART changes to initial status until detecting next start bit.



UART check the start bit is error and issue UFMER flag, but the UART still finishes receiving the pocket.



UART check the stop bit is error and issue UFMER flag, but the UART still finishes receiving the pocket.



If the host’s UART baud rate isn’t match to receiver terminal, the received pocket is error. But it is not easy to differentiate the pocket is correct or not, because the received error pocket maybe match UART rule, but the data is error. Use checking UFMER bit and URXPC bit status to decide the stream. If the two conditions seem like correct, but the pocket is abnormal, UART will accept the pocket as correct one.

✸ ***Note: When RX received data and the package data is fail (included start / stop bits status), check bits rising (UFMER = 1 / URXPC = 1) and keep status. So UFMER/URXPC bits is clear by program before RX received next new data.***

16.6 UART RECEIVER CONTROL REGISTER

0EAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRX	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **URXEN**: UART RX control bit.
0 = Disable UART RX. URX pin is GPIO mode or returns to GPIO status.
1 = Enable UART RX. URX pin exchanges from GPIO mode to UART RX mode.

Bit 6 **URXPEN**: UART RX parity bit control bit.
0 = Disable UART RX parity bit function. The data stream doesn't include parity bit.
1 = Enable UART RX parity bit function. The data stream includes parity bit.

Bit 5 **UTXPS**: UART RX parity bit format control bit.
0 = UART RX parity bit format is even parity.
1 = UART RX parity bit format is odd parity.

Bit 4 **URXPC**: UART RX parity bit checking flag.
0 = Parity bit is correct or no parity function.
1 = Parity bit is error.

Bit 3 **UFMER**: UART RX stream frame error flag bit.
0 = Collect UART frame.
1 = UART frame is error including start/stop bit, stream length.

Bit [2:0] **URS [2:0]**: UART per-scalar select bit.
000 = Fhosc/1, 001 = Fhosc/2, 010 = Fhosc/4, 011 = Fhosc/8, 100 = Fhosc/16, 101 = Fhosc/32, 110 = Fhosc/64, 111 = Fhosc/128.

16.7 UART TRANSMITTER CONTROL REGISTER

0E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTX	UTXEN	UTXPEN	UTXPS	UARTM	URXBZ	UTXBZ	UTXD8	URXD8
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R
After reset	0	0	0	0	0	0	0	0

Bit 7 **UTXEN**: UART TX control bit.
0 = Disable UART TX. UTX pin is GPIO mode or returns to GPIO status.
1 = Enable UART TX. UTX pin exchanges from GPIO mode to UART TX mode and idle high status.

Bit 6 **UTXPEN**: UART TX parity bit control bit.
0 = Disable UART TX parity bit function. The data stream doesn't include parity bit.
1 = Enable UART TX parity bit function. The data stream includes parity bit.

Bit 5 **UTXPS**: UART TX parity bit format control bit.
0 = UART TX parity bit format is even parity.
1 = UART TX parity bit format is odd parity.

Bit 4 **UARTM**: UART data buffer length control bit.
0 = UTX/URX buffer length are 8-bit.
1 = UTX/URX buffer length are 9-bit.

Bit 3 **URXBZ**: UART RX operating status flag.
0 = UART RX is idle or the end of processing.
1 = UART RX is busy and processing.

Bit 2 **UTXBZ**: UART TX operating status flag.
0 = UART TX is idle or the end of processing.
1 = UART TX is busy and processing.

Bit 1 **TXD8:** UART transmitted data buffer's bit 8
UARTM = 1. UTXD8 is transmitted data buffer's bit 8
UARTM = 0. Initial valus is "0".

Bit 0 **RXD8:** UART received data buffer's bit 8
UARTM = 1. URXD8 is received data buffer's bit 8
UARTM = 0. Keep "0".

✸ ***Note: URXBZ and UTXBZ bits are UART operating indicators. After setting UART RX/TX operations, set (2*Fcpu/Fuart)*NOP instruction is necessary, and then check UART status through URXBZ and UTXBZ bits.***

16.8 UART DATA BUFFER

0EC	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UTXD	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **UTXD:** UART transmitted data buffer.

0ED	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URXD	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0
Read/Write	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **URXD:** UART received data buffer.

16.9 UART OPERATION EXAMLPE

➢ **UART TX Configuration:**

```
; Select parity bit function.
        B0BCLR      FUTXPEN         ; Disable UART TX parity bit function.
;or
        B0BSET      FUTXPEN         ; Enable UART TX parity bit function.

; Select parity bit format.
        B0BCLR      FUTXPS          ; UART TX parity bit format is even parity.
;or
        B0BSET      FUTXPS          ; UART TX parity bit format is odd parity.

; Set UART baud rate.
        MOV         A, #value1      ; Set UART pre-scaler URS [2:0].
        B0MOV       URRX, A
        MOV         A, #value2      ; Set UART baud rate 8-bit buffer.
        B0MOV       URCR, A

; Select data buffer length.
        B0BCLR      FUARTM          ; UART data buffer is 8-bit length.
; or
        B0BSET      FUARTM          ; UART data buffer is 9-bit length.

; Enable UART TX pin.
        B0BSET      FUTXEN          ; Enable UART TX function and UART TX pin.

; Enable UART TX interrupt function.
        B0BCLR      FUTXIRQ         ; Clear UART TX interrupt flag.
        B0BSET      FUTXIEN         ; Enable UART TX interrupt function.

; Load TX data buffer and execute TX transmitter (UARTM = 0).
        MOV         A, #value3      ; Load 8-bit data to UTXD data buffer.
        B0MOV       UTXD, A
                                    ; After loading UTXD, UART TX starts to transmit.
        NOP                         ; One instruction delay for UTXBZ flag.

; Load TX data buffer and execute TX transmitter (UARTM = 1).
        B0BCLR      FUTXD8          ; Clear UTXD8 bit is “0”.
; or
        B0BSET      FUTXD8          ; Set UTXD8 bit is “1”.

        MOV         A, #value4      ; Load 8-bit data to UTXD data buffer.
        B0MOV       UTXD, A
                                    ; After loading UTXD, UART TX starts to transmit.
        NOP                         ; One instruction delay for UTXBZ flag.

; Check TX operation.
        B0BTS0      FUTXBZ          ; Check UTXBZ bit.
        JMP         CHKTX           ; UTXBZ=1, TX is operating.
        JMP         ENDTX           ; UTXBZ=0, the end of TX.
```

✸ ***Note: UART TX operation is started through loading UTXD data buffer.***

➢ **UART RX Configuration:**

```
; Select parity bit function.
        B0BCLR      FURXPEN     ; Disable UART RX parity bit function.
;or
        B0BSET      FURXPEN     ; Enable UART RX parity bit function.

; Select parity bit format.
        B0BCLR      FURXPS      ; UART RX parity bit format is even parity.
;or
        B0BSET      FURXPS      ; UART RX parity bit format is odd parity.

; Set UART baud rate.
        MOV         A, #value1  ; Set UART pre-scaler URS[2:0].
        B0MOV       URRX, A
        MOV         A, #value2  ; Set UART baud rate 8-bit buffer.
        B0MOV       URCR, A

; Select data buffer length.
        B0BCLR      FUARTM      ; UART data buffer is 8-bit length.
; or
        B0BSET      FUARTM      ; UART data buffer is 9-bit length.

; Enable UART RX pin.
        B0BSET      FURXEN      ; Enable UART RX function and UART RX pin.

; Enable UART RX interrupt function.
        B0BCLR      FURXIRQ     ; Clear UART RX interrupt flag.
        B0BSET      FURXIEN     ; Enable UART RX interrupt function.
        NOP                     ; One instruction delay for URXBZ flag.


; Check RX operation.
        B0BTS0      FURXBZ      ; Check URXBZ bit.
        JMP         CHKRX       ; URXBZ=1, RX is operating.
        JMP         ENDRX       ; URXBZ=0, the end of RX.
```

✸ ***Note: UART RX operation is started as start bit transmitted from master terminal.***

17 INSTRUCTION TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I, “M” only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z…)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
ARITHMETIC	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1+N
	ADD	A,M	A ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
	ADD	M,A	M ← A + M, if occur carry, then C=1, else C=0	√	√	√	1+N
	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
	ADD	A,I	A ← A + I, if occur carry, then C=1, else C=0	√	√	√	1
	SBC	A,M	A ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
	SUB	A,M	A ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
	SUB	M,A	M ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1+N
	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
	DAA		To adjust ACC’s data format from HEX to DEC.	√	-	-	1
	MUL	A,M	R, A ← A * M, The LB of product stored in Acc and HB stored in R register. ZF affected by Acc.	-	-	√	2
LOGIC	AND	A,M	A ← A and M	-	-	√	1
	AND	M,A	M ← A and M	-	-	√	1+N
	AND	A,I	A ← A and I	-	-	√	1
	OR	A,M	A ← A or M	-	-	√	1
	OR	M,A	M ← A or M	-	-	√	1+N
	OR	A,I	A ← A or I	-	-	√	1
	XOR	A,M	A ← A xor M	-	-	√	1
	XOR	M,A	M ← A xor M	-	-	√	1+N
	XOR	A,I	A ← A xor I	-	-	√	1
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1+N
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1+N
	CLR	M	M ← 0	-	-	-	1
	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
BRANCH	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1+ S
	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1+ S
	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1+N+S
	BTS0	M.b	If M.b = 0, then skip next instruction	-	-	-	1 + S
	BTS1	M.b	If M.b = 1, then skip next instruction	-	-	-	1 + S
	B0BTS0	M.b	If M(bank 0).b = 0, then skip next instruction	-	-	-	1 + S
	B0BTS1	M.b	If M(bank 0).b = 1, then skip next instruction	-	-	-	1 + S
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
MISC	RET		PC ← Stack	-	-	-	2
	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
	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”.

18 ELECTRICAL CHARACTERISTIC

18.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2741AP, SN8P2741AS..... 0°C ~ + 70°C
SN8P2741APD, SN8P2741ASD..... –40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

18.2 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. Fcpu = 1MHz		2.2	-	5.5	V
		Normal mode. Fcpu = 4MHz		2.3	-	5.5	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.8Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	150	
I/O output source current	IoH	Vop = Vdd – 0.5V		8	15	-	mA
sink current	IoL	Vop = Vss + 0.5V		8	15	-	
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current (Disable ADC, OP-amp, Comparator)	Idd1	Run Mode (No loading)	Vdd= 3V, Fcpu = 4MHz	-	2	-	mA
			Vdd= 5V, Fcpu = 4MHz	-	4	-	
			Vdd= 3V, Fcpu = 1MHz	-	1.5	-	
			Vdd= 5V, Fcpu = 1MHz	-	3	-	
	Idd2	Slow Mode	Vdd= 3V, ILRC=16KHz	-	3.5	-	uA
			Vdd= 5V, ILRC=32KHz	-	10	-	
	Idd3	Sleep Mode	Vdd= 5V/3V	-	1	2	uA
	Idd4	Green Mode (No loading, Watchdog Disable)	Vdd= 3V, IHRC=16MHz	-	0.35	-	mA
			Vdd= 5V, IHRC=16MHz	-	0.55	-	
			Vdd= 3V, ILRC=16KHz	-	3	-	uA
			Vdd= 5V, ILRC=32KHz	-	5.5	-	
Internal High Oscillator Freq.	Fihrc	Internal High RC (IHRC)	25°C, Vdd = 2.2V~ 5.5V, Fcpu = Fhosc/4 ~ Fhosc/16	15.68	16	16.32	MHz
			-40°C~85°C, Vdd = 2.3V~ 5.5V, Fcpu = Fhosc/4 ~ Fhosc/16	15.2	16	16.8	MHz
LVD Voltage	Vdet0	Low voltage reset level (Trimmed, sleep mode, 25°C)		2.0	2.1	2.2	V
		Low voltage reset level (Non-trimmed, sleep mode, default value, 25°C)		1.9	2.1	2.3	
		Low voltage reset level (Trimmed, sleep mode, 0°C ~70°C)		1.9	2.1	2.3	
		Low voltage reset level (Non-trimmed, sleep mode, default value, 0°C~70°C)		1.8	2.1	2.4	

" *" 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
Operating voltage	Vop		2.2	-	5.5	V
*Supply current	Iadc	Reference voltage = Vdd, Vdd = 5V.	-	-	250	uA
	Vpor	Reference voltage = 4V, Vdd = 5V.	-	-	350	
*ADC reference voltage	Vhref	Internal Vdd	-	Vdd	-	V
		Internal reference voltage 4V	-	4	-	
	Vlref	Low reference voltage.	-	0	-	
AIN0 ~ AIN7 input voltage	Vani	Vdd = 5.0V	0	-	Avrefh	V
*ADC enable time	Tast	Ready to start convert after set ADENB = "1"	100	-	-	us
ADC Clock Frequency	F_{ADCLK}	Fosc = 16MHz.	-	-	4M	Hz
ADC Converting Time	F_{ADCT}	Vdd=2.2V~5.5V (12-bit)	64	-	-	1/FAD CLK
		Vdd=2.2V~5.5V (8-bit)	48	-	-	
ADC Converting Rate (FADS=1 Frequency)	F_{ADCR}	Vdd=2.2V~5.5V (12-bit)	-	-	62.5	K/sec
		Vdd=2.2V~5.5V (8-bit)	-	-	83.3	K/sec
Differential Nonlinearity	DNL	Vdd=AVREFH=3V, FADCR = 62.5K	-1	-	+1	LSB
Integral Nonlinearity	INL	Vdd=AVREFH=3V, FADCR = 62.5K	-2	-	+2	LSB
No Missing Code	NMC	AVREFH=Vdd=3V, FADCR = 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.

INTERNAL 4V REFERENCE VOLTAGE 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	Vop		2.2	-	5.5	V
*Supply current	Iref	Vdd=5V	-	100	-	uA
Internal 4V Reference Voltage	Vref	Trimmed. VDD=4.5~5.5V	-2		+2	%
		Non-trimmed (default Value). VDD=4.5~5.5V	-5		+5	
		*VDD=2.2~4.5V	-	VDD	-	V

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

OP AMP 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	Vop		2.2	-	5.5	V
*Supply current	Iop	Vdd = 5V, Unit-gain-buffer (V+ tie to Vss).	-	200	-	uA
		Vdd = 3V, Unit-gain-buffer (V+ tie to Vss).	-	180	-	
*Common Mode Input Voltage Range	Vcmr	Vdd = 5.0V	Vss+2	-	Vdd	mV
*Input Offset Voltage	Vos	Unit-gain-buffer (V+ tie to 1/2*Vdd).	-	+3	-	mV
*Power Supply Rejection Ratio	PSRR	Vcm = Vss	50	-	70	dB
*Common Mode Rejection Ratio	CMRR	Vcm=-0.3V~5.0V. Vdd=5V.	50		-	dB
*Open-Loop Gain (Large Signal)	Aol	Vout = 0.2V~Vdd-0.2V. Vcm=Vss.	90			dB
Output Voltage Swing	Vos	Vopp = 2.5V.	Vss+15		Vdd-15	mV
Output Short Current	Isc	Unit Gain Buffer, Vi=Vdd~Vss, Vo=Vss~Vdd, Vdd=5V.	-	25	-	mA
*Output Slew Rate	Tosr	Unit-gain buffer, Vo=rising Vss~Vdd. Vdd=5V.	-	5	-	us
		Unit-gain buffer, Vo=falling Vdd~Vss. Vdd=5V.	-	5	-	

" *" 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
Operating voltage	Vcm		2.2	-	5.5	V
*Supply current	Icm	CM0P=Vdd, CM0N=Vss, VDD = 5V	-	150	-	uA
*Input Offset Voltage	Vos	Vcm=Vss	-5	-	+5	mV
Response Time	Trs1	Vp=Vo=rising Vss~Vdd. Vn=2.5V. Vdd=5V.	-	200	-	ns
	Trs2	Vp=Vo=falling Vdd~Vss. Vn=2.5V. Vdd=5V.	-	75	-	
Output Slew Rate Time	Tsr	Vo=rising Vss~Vdd or falling Vdd~Vss. Vdd=5V.	-	6	-	ns
Common Mode Input Voltage Range	Vcmr	Vdd=5.0V	Vss		Vdd-1.5	V

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

- **COMPARATOR 1/2 CHARACTERISTIC**

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

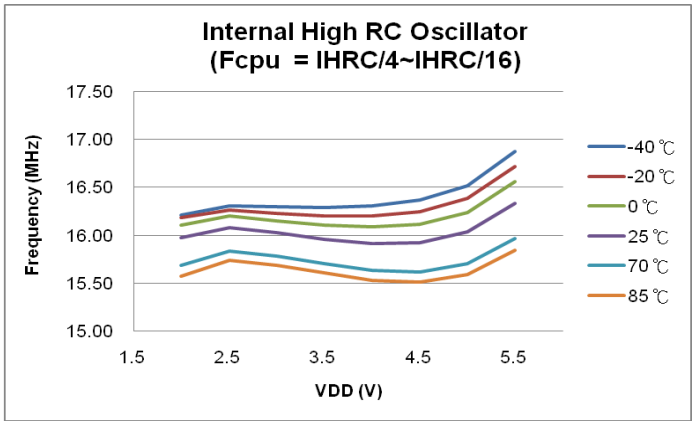
PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Operating voltage	Vcm		2.2	-	5.5	V
*Supply current	Icm1	Vdd=5V. Internal 4V reference voltage disabled.	-	400	-	uA
	Icm2	Vdd=5V. Internal 4V reference voltage enabled.	-	500	-	
*Input Offset Voltage	Vos	Vcm=Vss	-5	-	+5	mV
Response Time	Trs1	Vp=Vo=rising Vss~Vdd. Vp=2.5V. Vdd=5V.	-	50	-	ns
	Trs2	Vp=Vo=falling Vdd~Vss. Vp=2.5V. Vdd=5V.	-	150	-	
Output Slew Rate Time	Tsr	Vo=rising Vss~Vdd or falling Vdd~Vss. Vdd=5V.	-	-	10	ns
Common Mode Input Voltage Range	Vcmr	Vdd=5.0V	Vss+0.5		Vdd-0.5	V

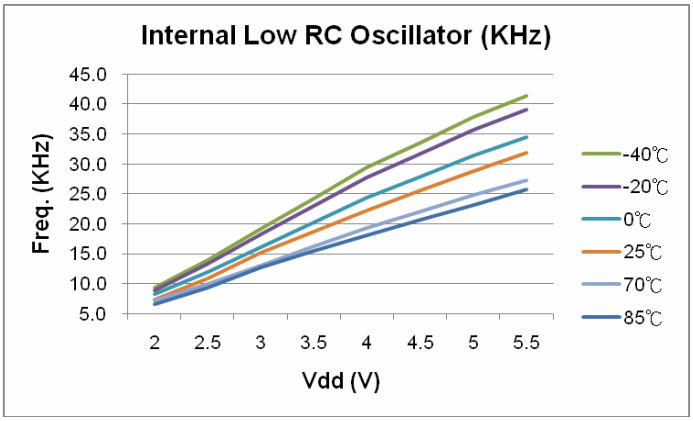
" *" These parameters are for design reference, not tested.

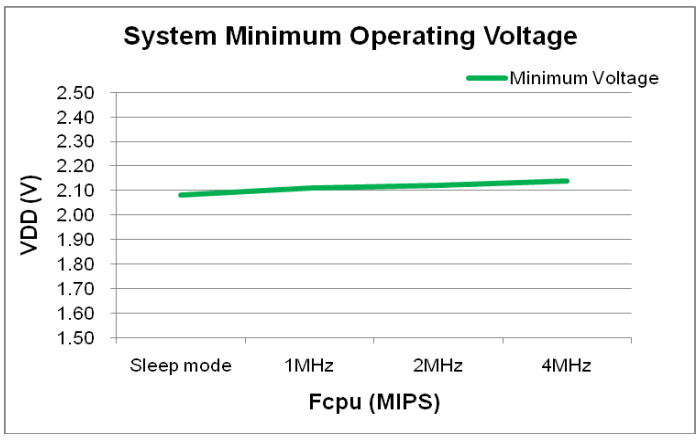
18.3 CHARACTERISTIC GRAPHS

The Graphs in this section are for design guidance, not tested or guaranteed. In some graphs, the data presented are outside specified operating range. This is for information only and devices are guaranteed to operate properly only within the specified range (-40℃~+85℃ curves are for design reference).

.







19 DEVELOPMENT TOOL

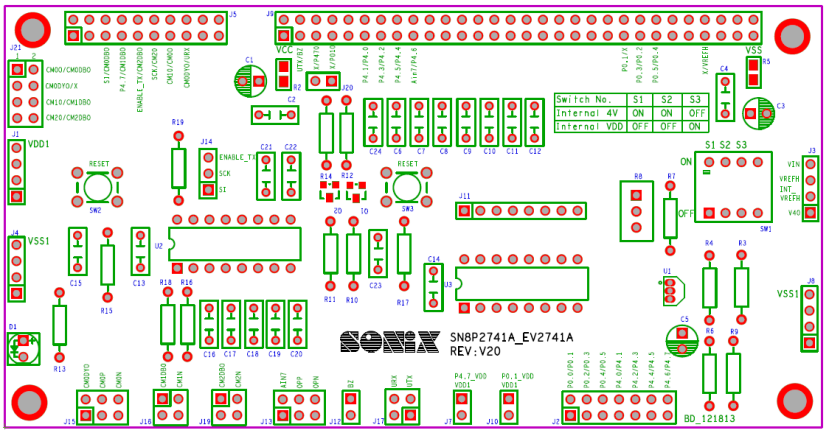
SONIX provides ICE (in circuit emulation), IDE (Integrated Development Environment) and EV-kit for SN8P2741A development. ICE and EV-kit are external hardware devices, and IDE is a friendly user interface for firmware development and emulation. These development tools' version is as following.

- **ICE: SN8ICE2K Plus II. (Please install 16MHz crystal in ICE to implement IHRC emulation.)**
- **ICE emulation speed maximum: 8 MIPS @ 5V (e.g. 16Mhz crystal, Fcpu = Fosc/).**
- **EV-kit: EV2741A KIT REV: V1.0.**
- **IDE: SONiX IDE M2IDE_V139 and later version.**
- **Writer: MPIII writer.**
- **Writer transition board: SN8P2741A P/S**

19.1 SN8P2741A EV-KIT

SONIX provides SN8P2741A series MCU which includes PWM, ADC, Comparator 0~2, OP-AMP and OP analog functions. These functions aren't built in SN8ICE2K Plus 2. To emulate the functions must be through SN8P2741A real chip. The real chip provides an EV-KIT to achieve PWM and the analog functions emulations. For SN8P2741A ICE emulation, the EV-Kit includes OP-AMP/Comparator/PWM/ADC/Internal 4V or VDD Reference Voltage and switch circuits.

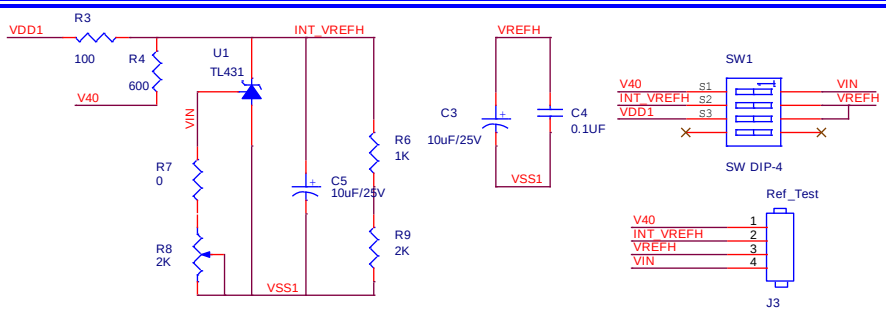
EV2741A KIT PCB Outline:



- **D1**: EV-KIT power indicator.
- **SW1:** ADC reference voltage selection. The reference voltage is connected to VREFH pin of CON1. The max. reference voltage is VDD. If VDD < INT_VREFH_4.0V, the ADC reference voltage is VDD.

Switch No.	S1	S2	S3	S4
Internal 4V	ON	ON	OFF	OFF
Internal VDD	OFF	OFF	ON	OFF

- **R8:** 2K ohm VR to adjust ADC internal reference voltage. Internal reference voltage have to be correct. Set SW1 to Internal 4V mode, input power VDD = 5V, measure internal reference voltage from J3. Adjust R8 to make J3 voltage = 4.0V.

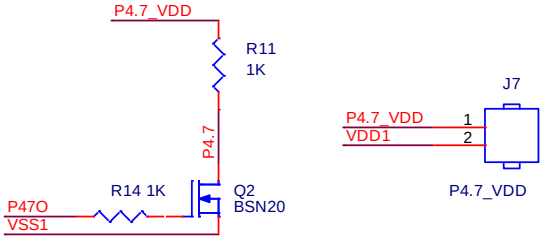


- **SW2**: SN8P2741A EV-chip reset key. If EV-KIT active fail, press SW2 to reset EV-KIT Real Chip (U2).
- **U2/U3**: SN8P2741A form connector for connecting to user's target board.

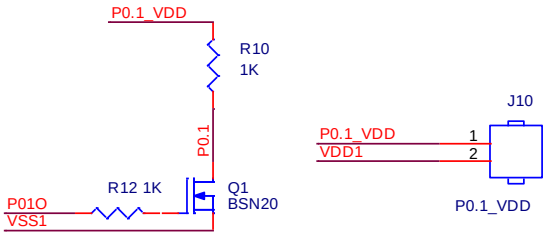
VSS	1	U	16	VDD
P0.0/INT0/BZ	2		15	P4.7/URX/RST/VPP
P0.1/PWM0	3		14	P4.6/AIN6/UTX
P0.2/CM0P	4		13	P4.5/AIN5/CM0O
P0.3/CM0N	5		12	P4.4/AIN4/CM1O
P0.4/CM1N	6		11	P4.3/AIN3/CM2O
P0.5/CM2N	7		10	P4.2/AIN2/OPP
P4.0/AIN0/OPO	8		9	P4.1/AIN1/OPN

- **J2**:
  - P0.0~P0.5 are GPIO function only. But do not support Comparator 0~2 analog functions.
  - P4.0~P4.2 are GPIO and ADC input functions. But do not support OP-AMP analog function.
  - P4.3~P4.5 are GPIO and ADC input functions. But do not support Comparator 0~2 output function.
  - P4.6 is GPIO and ADC input functions. But do not support Uart UTX output function.
  - P4.7 is GPIO function. But do not support Uart URX and RST functions.

- **J5**: Connect to SN8ICE2K Plus 2 JP3 (EV-KIT communication bus with ICE, control signal, and the others).
- **J7**: P4.7 open-drain structure power pin.



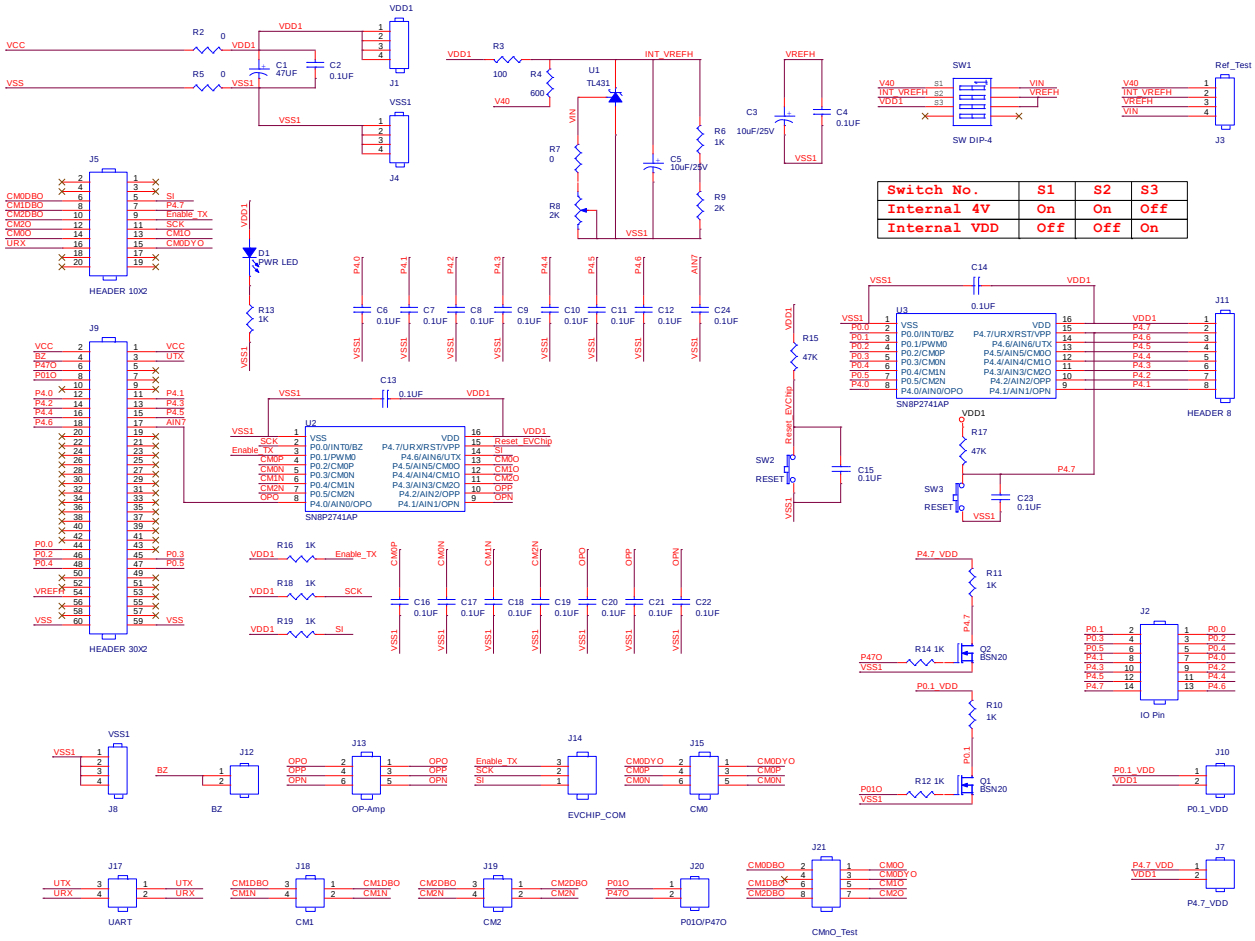
- **J9**: Connect to SN8ICE2K Plus 2 CON1 (includes GPIO, EV-KIT control signal, and the others).
- **J10**: P0.1 open-drain structure power pin.



- **J12**: Buzzer logic function only. But do not support P0.0/INT0 GPIO function.
- **J13**: OP-AMP analog function only. But do not support P4.0/P4.1/P4.2 GPIO function.
- **J15**: Comparator 0 analog positive/negative/output pin. But do not support P0.2/P0.3/P4.5 GPIO function.
- **J17**: Uart UTX/URX logic function only. But do not support P4.6/P4.7 GPIO function.
- **J18**: Comparator 1 analog positive/negative/output pin. But do not support P0.4/P4.4 GPIO function.
- **J19**: Comparator 2 analog positive/negative/output pin. But do not support P0.5/P4.3 GPIO function.
- **C6~C12:** Connect 0.1uF capacitors to AIN0~AIN6 input which are ADC's (channel 0~6) bypass capacitors.

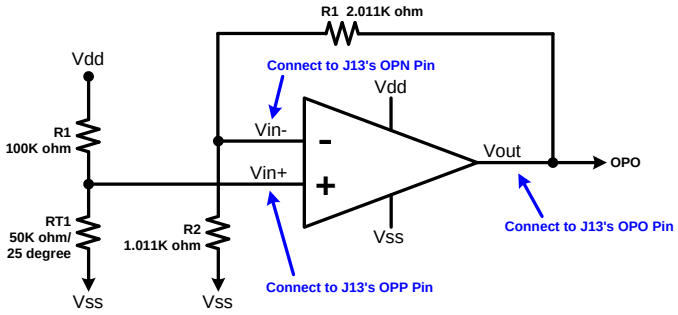
- **C7:** Connect 0.1uF capacitors to AIN7 input which is ADC's (channel 7) bypass capacitors (OP-AMP output terminal).
- **C16:** Connect 0.1uF capacitors to SN8P2741A EV-chip's CM0P input pin which is Comparator 0's bypass capacitors.
- **C17:** Connect 0.1uF capacitors to SN8P2741A EV-chip's CM0N input pin which is Comparator 0's bypass capacitors.
- **C18:** Connect 0.1uF capacitors to SN8P2741A EV-chip's CM1N input pin which is Comparator 1's bypass capacitors.
- **C19:** Connect 0.1uF capacitors to SN8P2741A EV-chip's CM2N input pin which is Comparator 2's bypass capacitors.
- **C20:** Connect 0.1uF capacitors to SN8P2741A EV-chip's OPO input pin which is OP-AMP's bypass capacitors.
- **C21:** Connect 0.1uF capacitors to SN8P2741A EV-chip's OPP input pin which is OP-AMP's bypass capacitors.
- **C22:** Connect 0.1uF capacitors to SN8P2741A EV-chip's OPN input pin which is OP-AMP's bypass capacitors.

EV2741A KIT schematic:



19.2 ICE AND EV-KIT APPLICATION NOTIC

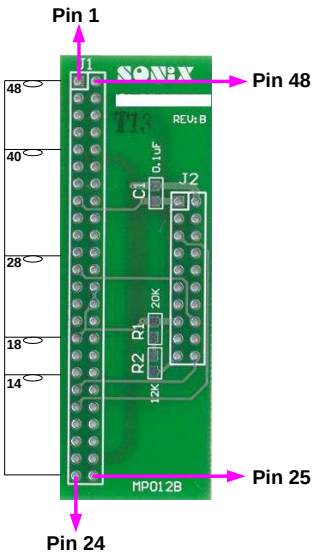
1. SN8ICE2K Plus 2 power switch must be turned off before you connect the EV2741A KIT to SN8ICE2K Plus 2.
2. Connect EV-KIT's J5/J9 to ICE's JP3/CON1.
3. SN8ICE2K Plus 2's AVREFH/VDD jumper pin must be removed.
4. Turn on SN8ICE2K Plus power switch after user had finished step 1~3.
5. User observes EV-KIT's power LED (D1) is light after turn on SN8ICE2K Plus power switch. If LED (D1) is not 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.**
7. When ADC function enable. The ADM's bit 3 (FADREF) is set as High. S1/S2 must be turn-on. S3 must be turn-off.
8. When ADC function enable. The ADM's bit 3 (FADREF) is set as Low. S1/S2 must be turn-off. S3 must be turn-on.
9. **If i**nternal reference voltage is not correct, adjust R8 resistor and measure internal reference voltage from J3.
10. The OP-AMP of SN8P2741A application circuit is as the below figure (OP-AMP connect).



11. When SN8P2741A's OP-AMP function enable as the above figure. OPO (J13) is OP-AMP's output. OPP (JP23) is OP-AMP's non-inverse. OPN (J13) is OP-AMP's inverse.
12. When OPEN =1 and OPPNS = 1, EV-Chip support the OP-AMP function.
13. When CMP0 function enable. The CM0P/CM0N (JP15) will be external analog signal input pin. The CM0O (J15) will be CMP0's output result.
14. When CMP1 function enable. The CM1N (JP18) will be external analog signal input pin. The CM1O (J18) will be CMP1's output result.
15. When CMP2 function enable. The CM1N (JP19) will be external analog signal input pin. The CM2O (J19) will be CMP2's output result.
16. When user uses Comparator 0~2's CM0P/CM0N/CM0O/CM1N/CM1O/CM2N/CM2O analog function, user must be connecting to J15/J18/J19.
17. J2's support P02/P03/P45/P04/P44/P05/P43 GPIO/ADC function and doesn't include Comparator 0/1/2 function.
18. **When user uses TC0 special function (Pulse generator function and TC0 clock source is Fcpu.) in ICE emulation.**
    If user sets IDE breakpoint, the PWM plus generator output status will be unknown (Fcpu stop).
19. **When user uses TC0 special function (Pulse generator function and TC0 clock source is Fhosc.) in ICE emulation.**
    If user sets IDE breakpoint, the PWM plus generator output will be finished (Fhosc still work). And the PWM pulse generator output will be back to idle status.
20. When user uses comparator 0 output de-bounce time control (CM0D3~CM0D0, clock source is Fcpu) in ICE emulation. If user sets IDE breakpoint, the CM0O output will be effected (Fcpu stop)
21. When user uses comparator 0 output delay time control (CM0DY3~CM0DY0, clock source is Fcpu) in ICE emulation. If user sets IDE breakpoint, the CM0O output will be effected (Fcpu stop).
22. When user uses comparator 0 output delay time control (CM0DY3~CM0DY0, clock source is Fhosc) in ICE emulation. If user sets IDE breakpoint, the CM0O output will be not effected (Fhosc still work).
23. When user uses CMP1 and CMP2's de-bounce time control (CM1D1~CM1D0, CM2D1~CM2D0, clock source is Fcpu) in ICE emulation. If user sets IDE breakpoint, the CM0O output will be effected (Fcpu stop)
24. **EV-Chip (U2) support to emulate Comparator 1/2 positive internal reference voltage (VDD/4V). EV-Kit support external circuit (R8/SW1/U1/…) to emulate ADC internal reference voltage (VDD/4V). So Comparator 0/1 and ADC internal reference voltage power level (VDD/4V) are different. User have to take care (Real chip SN8P2741A doesn't have this problem).**

20 OTP PROGRAMMING PIN

20.1 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

20.2 PROGRAMMING PIN MAPPING:

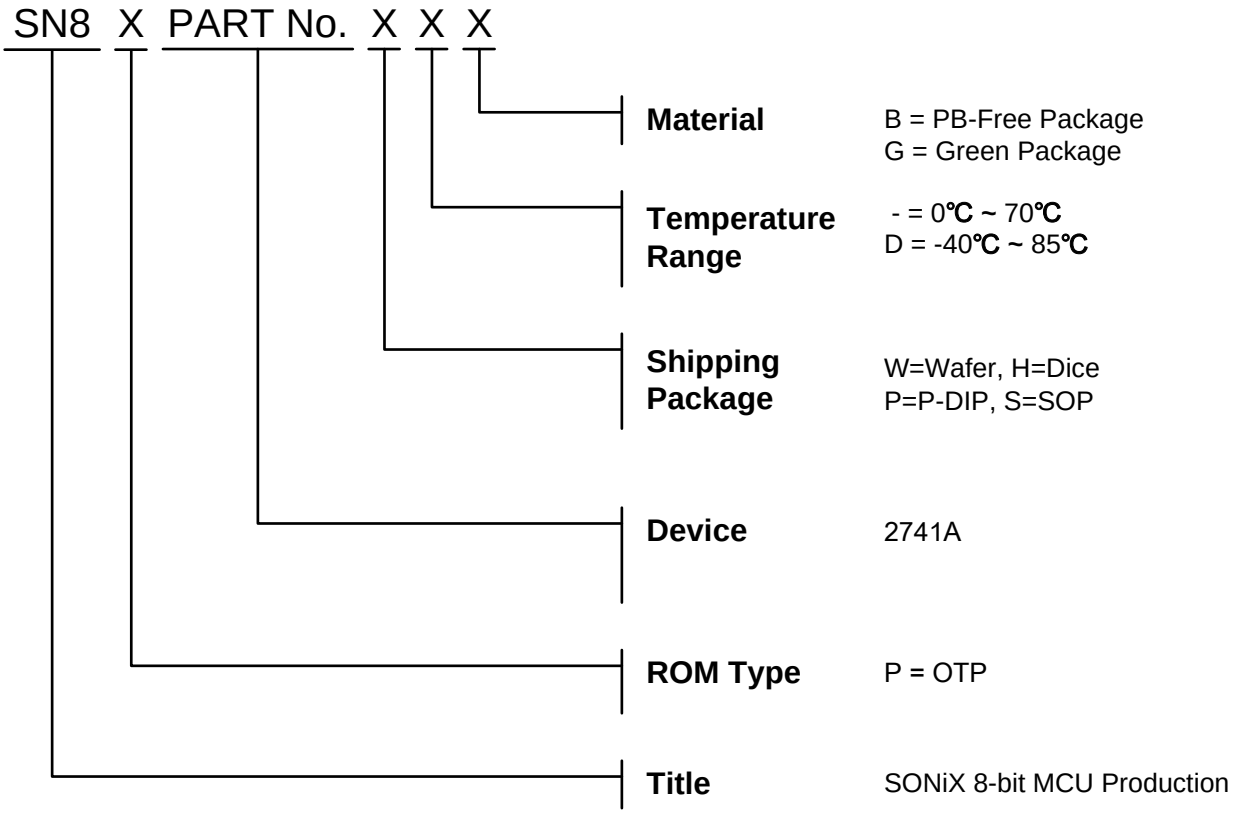
Programming Pin Information of SN8P2741A Series							
Chip Name		SN8P2741AP(PDIP)/S(SOP)					
Writer Connector		IC and JP3 48-pin text tool Pin Assignment					
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number
1	VDD	16	VDD	32			
2	GND	1	VSS	17			
3	CLK	12	P4.4	28			
4	CE	-	-	-			
5	PGM	13	P4.5	29			
6	OE	11	P4.3	27			
7	D1	-	-	-			
8	D0	-	-	-			
9	D3	-	-	-			
10	D2	-	-	-			
11	D5	-	-	-			
12	D4	-	-	-			
13	D7	-	-	-			
14	D6	-	-	-			
15	VDD	-	-	-			
16	VPP	15	RST	31			
17	HLS	-	-	-			
18	RST	-	-	-			
19	-	-	-	-			
20	ALSB/PDB	10	P4.2	26			

21 MARKING DEFINITION

21.1 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.

21.2 MARKING INDETIFICATION SYSTEM



21.3 MARKING EXAMPLE

- **Wafer, Dice:**

Name	ROM Type	Device	Package	Temperature	Material
S8P2741AW	OTP	2741A	Wafer	0℃~70℃	-
SN8P2741AH	OTP	2741A	Dice	0℃~70℃	-

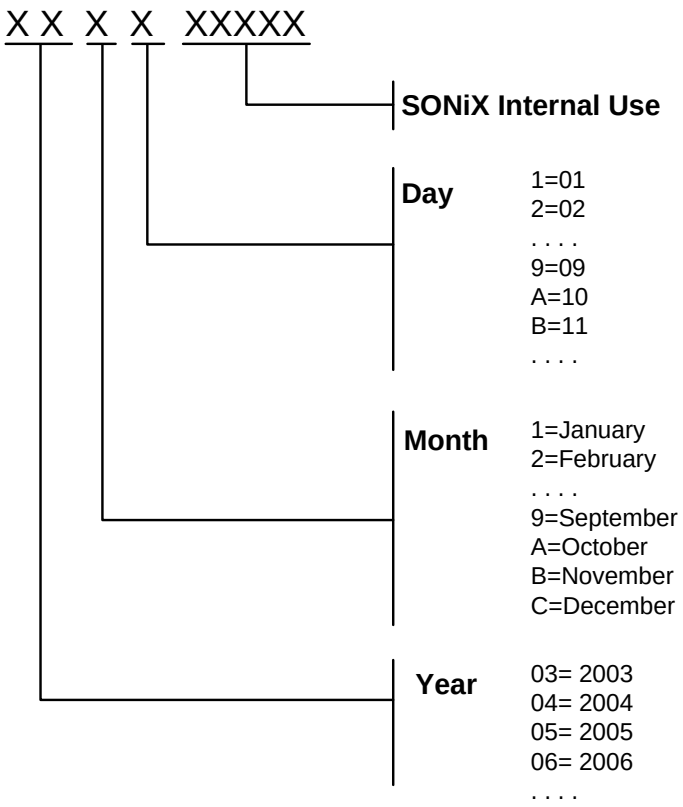
- **Green Package:**

Name	ROM Type	Device	Package	Temperature	Material
SN8P2741APG	OTP	2741A	P-DIP	0℃~70℃	Green Package
SN8P2741ASG	OTP	2741A	SOP	0℃~70℃	Green Package
SN8P2741APDG	OTP	2741A	P-DIP	-40℃~85℃	Green Package
SN8P2741ASDG	OTP	2741A	SOP	-40℃~85℃	Green Package

- **PB-Free Package:**

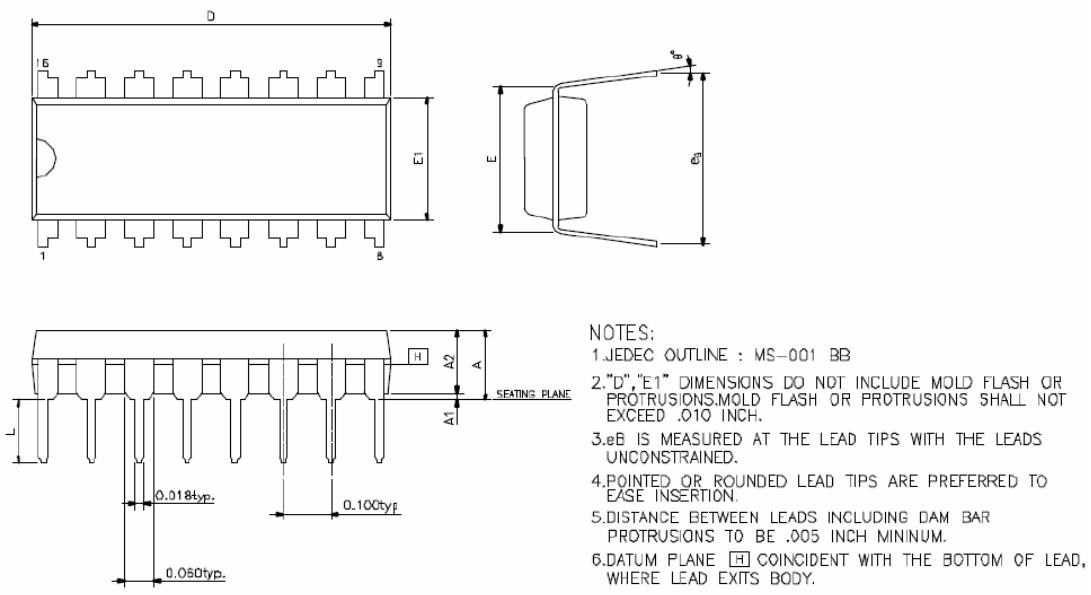
Name	ROM Type	Device	Package	Temperature	Material
SN8P2741APB	OTP	2741A	P-DIP	0℃~70℃	PB-Free Package
SN8P2741ASB	OTP	2741A	SOP	0℃~70℃	PB-Free Package
SN8P2741APDB	OTP	2741A	P-DIP	-40℃~85℃	PB-Free Package
SN8P2741ADB	OTP	2741A	SOP	-40℃~85℃	PB-Free Package

21.4 DATECODE SYSTEM



22 PACKAGE INFORMATION

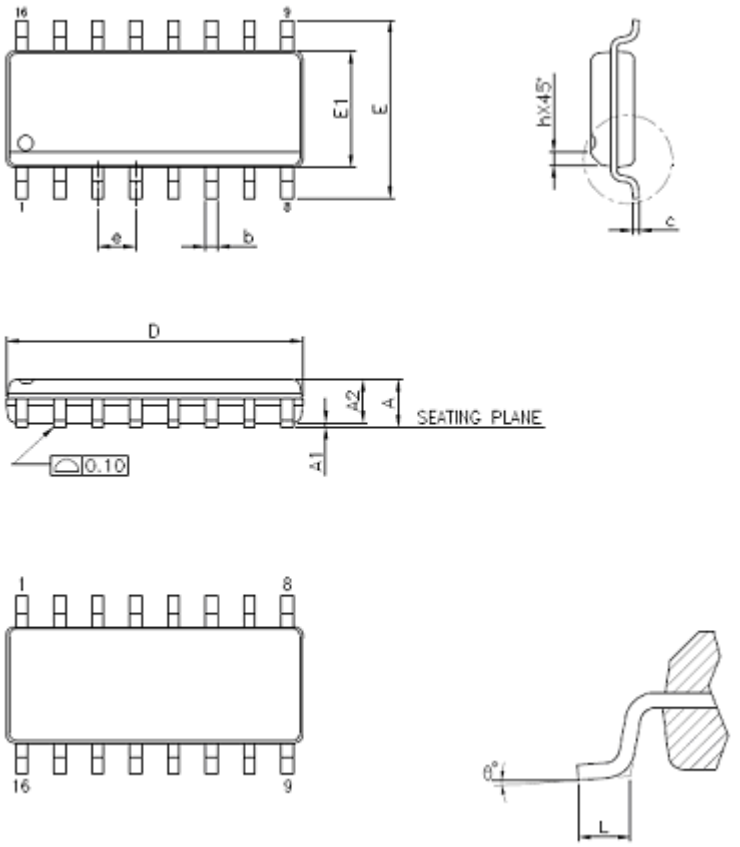
22.1 P-DIP 16 PIN



NOTES:
1. JEDEC OUTLINE : MS-001 BB
2. "D","E1" DIMENSIONS DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS.MOLD FLASH OR PROTRUSIONS SHALL NOT EXCEED .010 INCH.
3. eB IS MEASURED AT THE LEAD TIPS WITH THE LEADS UNCONSTRAINED.
4. POINTED OR ROUNDED LEAD TIPS ARE PREFERRED TO EASE INSERTION.
5. DISTANCE BETWEEN LEADS INCLUDING DAM BAR PROTRUSIONS TO BE .005 INCH MININUM.
6. DATUM PLANE [H] COINCIDENT WITH THE BOTTOM OF LEAD, WHERE LEAD EXITS BODY.

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°

22.2 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°

Corporate Headquarters:
10F-1, No.36, Taiyuan Street, Chupei City, Hsinchu, Taiwan
TEL :(886)3-5600-888 FAX :(886)3-5600-889
Taipei Sales Office:
15F-2, No.171 Song Ted Road, Taipei, Taiwan
TEL:(886)2-2759-1980 FAX:(886)2-2759-8180
mkt@sonix.com.tw | sales@sonix.com.tw
Hong Kong Sales Office:
Unit 2603, 26/F CCT Telecom Building, No. 11 Wo Shing Street, Fo Tan, New Territories, Hong Kong
TEL:(852)2723-8086 FAX:(852)2723-9179
hk@sonix.com.tw
Shenzhen Contact Office:
High Tech Industrial Park, Shenzhen, China
TEL:(86)755-2671-9666 FAX:(86)755-2671-9786
mkt@sonix.com.tw | sales@sonix.com.tw
Sonix Japan Office:
Kobayashi bldg. 2F, 4-8-27,Kudanminami, Chiyodaku,Tokyo, 102-0074, Japan
TEL:(81)3-6272-6070 FAX:(81)3-6272-6165
jpsales@sonix.com.tw
FAE Support via Email:
8-bit Microcontroller Products: sa1fae@sonix.com.tw
All Products: fae@sonix.com.tw