SN8P2524

USER'S MANUAL

Version 1.0

SN8P2524

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 1.0	Nov. 2011	First issue.

Table of Content

1 PRODUCT OVERVIEW

1.1 FEATURES

- **Memory configuration**
  ROM size: 2K * 16 bits.
  RAM size: 256 * 8 bits.

- **8 levels stack buffer.**

- **6 interrupt sources**
  5 internal interrupts: T0, TC1, CM0, SIO, MSP
  1 external interrupt: INT0

- **I/O pin configuration**
  Bi-directional: P0, P1, P5.
  Wakeup: P0, P1 level change.
  Pull-up resisters: P0, P1, P5.
  Programmable open-drain: P5.0~P5.2
  Comparator input pin: CM0N0~CM0N15.
  Comparator output pin: CM0O.

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

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

- **One 8-bit basic timer. (T0).**
- **One 8-bit timer with duty/cycle programmable PWM. (TC1).**
- **8-ch LED PWM driver.**

- **16-channel comparator.**
- **SIO serial input/output interface.**
- **MSP slave mode interface.**
- **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)

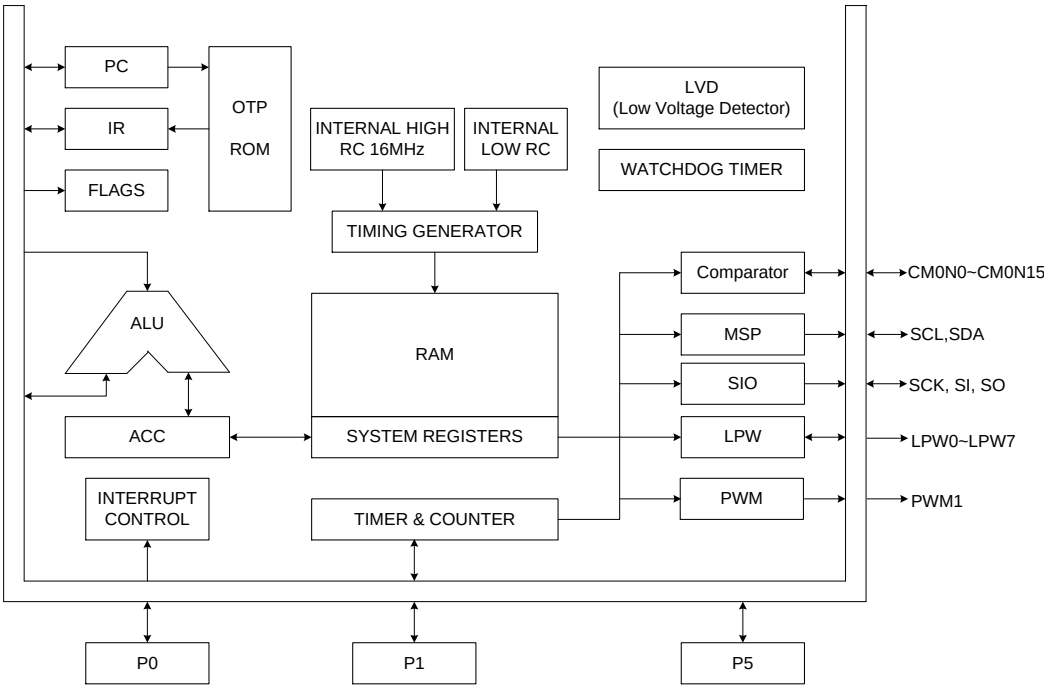
- **Four operating modes**
  Normal mode: Both high and low clock active
  Slow mode: Low clock only
  Sleep mode: Both high and low clock stop
  Green mode: Periodical wakeup by timer

- **Package (Chip form support)**
  SKDIP 24 pin
  SOP 24 pin
  SSOP 28 pin

☞ **Features Selection Table**

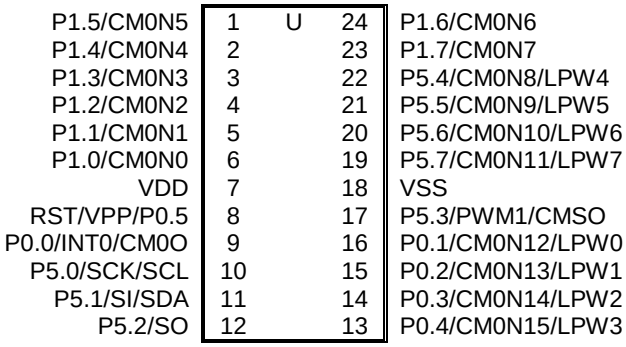
CHIP	ROM	RAM	Stack	Timer				SIO	MSP	I/O	Comp-arator	PWM	Wake-up Pin No.	Package
				T0	TC0	TC1	T1							
SN8P2522	2K*16	128	8	V	V	V	V	V	-	16	8-ch	2	9	DIP18/SOP18/ SSOP20
SN8P2523	2K*16	256	8	V	-	V	-	V	V	22	12-ch	8+1	14	SKDIP24/SOP24 /SSOP28
SN8P25231	2K*16	256	8	V	-	V	-	V	V	16	8-ch	2+1	12	DIP18/SOP18/ SSOP20
SN8P2524	2K*16	256	8	V	-	V	-	V	V	22	16-ch	8+1	14	SKDIP24/SOP24 /SSOP28

1.2 SYSTEM BLOCK DIAGRAM



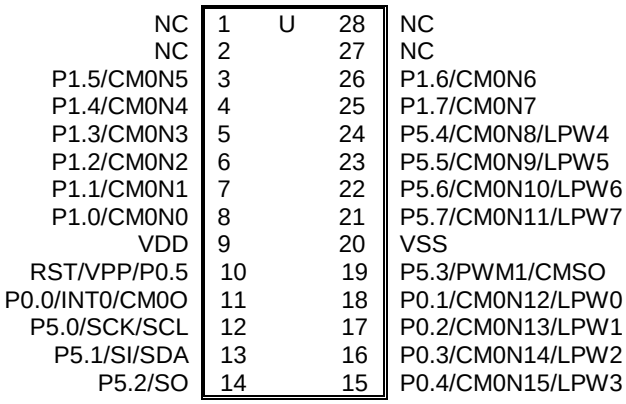
1.3 PIN ASSIGNMENT

SN8P2524K (SKDIP 24 pins)
SN8P2524S (SOP 24 pins)



SN8P2524K
SN8P2524S

SN8P2524X (SSOP 28 pins)



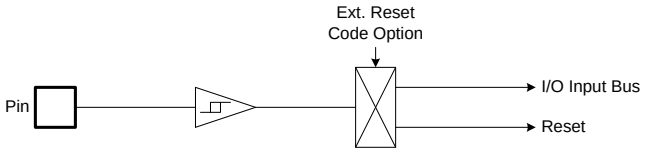
SN8P2524X

1.4 PIN DESCRIPTIONS

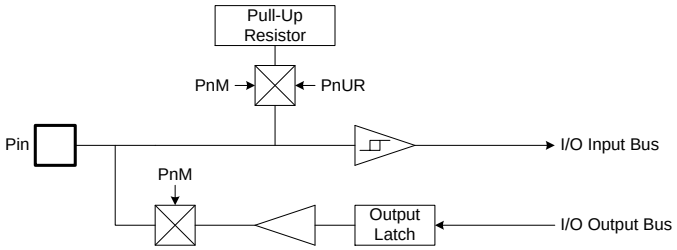
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins.
P0.5/RST/VPP	I, P	RST: System reset input pin. Schmitt trigger structure, low active, normal stay to “high”.
		VPP: OTP 12.3V power input pin in programming mode.
		P0.5: Input only pin. Schmitt trigger structure. Built-in wakeup function. The external reset function must be disabled.
P0.0/INT0/CM0O	I/O	Port 0.0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters and wakeup function.
		INT0 trigger pin (Schmitt trigger).
		CM0O: The output pin of comparator.
P0[4:1]/ LPW[3:0]	I/O	P0.1-0.4 bi-direction pin. No Schmitt trigger structure. Without built-in pull-up resisters.
		LPW[3:0]: LED[3:0] PWM output pin.
P1[7:0]/CM0N[7:0]	I/O	Port 1 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters and wakeup function.
		CM0N[7:0]: Channel 0~7 of comparator negative input pin.
P5.0/SCK/SCL	I/O	Port 5.0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Programmable open-drain.
		SCK: SIO clock pin.
		SCL: MSP clock pin
P5.1/SI/SDA	I/O	Port 5.1 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Programmable open-drain.
		SI: SIO data input pin.
		SDA: MSP data pin
P5.2/SO	I/O	Port 5.2 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Programmable open-drain.
		SO: SIO data output pin.
P5.3/PWM1/CMSO	I/O	Port 5.3 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		PWM1: PWM output pin.
		CMSO: Compensation output pin.
P5[7:4]/CM0N[11:8]/ LPW[7:4]	I/O	Port 5.4-5.7 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM0N[11:8]: Channel 8~11 of comparator negative input pin.
		LPW[7:4]: LED[7:4] PWM output pin.

1.5 PIN CIRCUIT DIAGRAMS

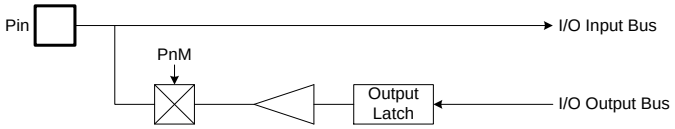
- **Port 0.5 structure:**



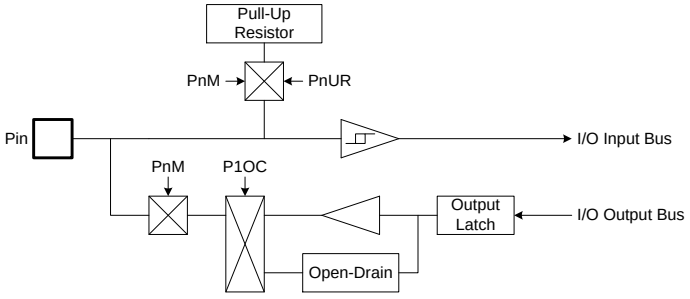
- **Port 0.0, Port 5.3 structure:**



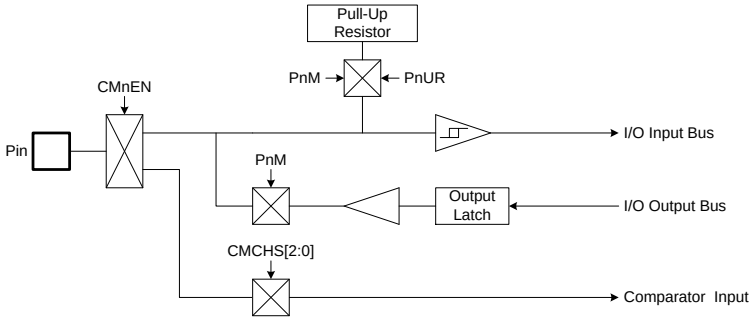
- **Port 0.1~0.4 structure:**



- **Port 5.0~ 5.2 structures:**



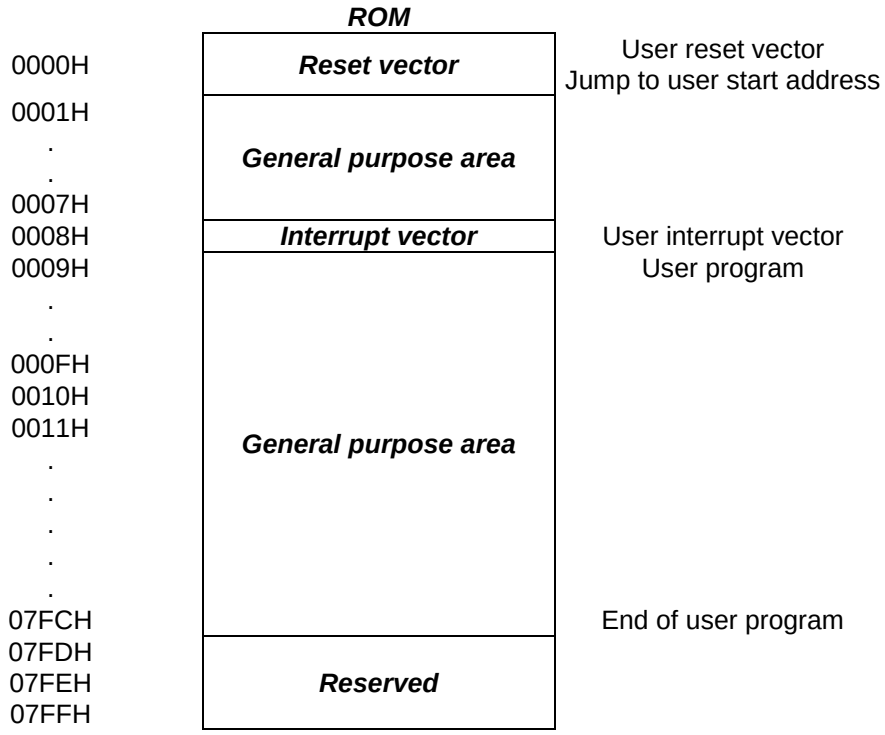
- **Port 1.0~ 1.7/Port 5.4~ 5.7 structures:**



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (ROM)

☞ ***2K words ROM***



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

2.1.1 RESET VECTOR (0000H)

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

- ☞ **Power On Reset (NT0=1, NPD=0).**
- ☞ **Watchdog Reset (NT0=0, NPD=0).**
- ☞ **External Reset (NT0=1, NPD=1).**

After power on reset, external reset or watchdog timer overflow reset, then the chip will restart the program from address 0000h and all system registers will be set as default values. It is easy to know reset status from NT0, NPD flags of PFLAG register. The following example shows the way to define the reset vector in the program memory.

➢ **Example: Defining Reset Vector**

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

                ORG     10H
START:                              ; 0010H, The head of user program.
                …                   ; User program
                …


                ENDP                ; End of program
```

2.1.2 INTERRUPT VECTOR (0008H)

A 1-word vector address area is used to execute interrupt request. If any interrupt service executes, the program counter (PC) value is stored in stack buffer and jump to 0008h of program memory to execute the vectored interrupt. Users have to define the interrupt vector. The following example shows the way to define the interrupt vector in the program memory.

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

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

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

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

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

                ENDP                ; End of program
```

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

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

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

                ENDP                ; End of program.
```

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

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

2.1.3 LOOK-UP TABLE DESCRIPTION

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

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

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

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

➢ **Example: INC_YZ macro.**

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

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

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

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

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

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

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

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

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

        B0BTS1  FC              ; Check the carry flag.
        JMP     GETDATA         ; FC = 0
        INCMS   Y               ; FC = 1. Y+1.
        NOP
GETDATA:                        ;
        MOVC                    ; To lookup data. If BUF = 0, data is 0x0035
                                ; If BUF = 1, data is 0x5105
                                ; If BUF = 2, data is 0x2012
        …
TABLE1: DW      0035H           ; To define a word (16 bits) data.
        DW      5105H
        DW      2012H
        …
```

2.1.4 JUMP TABLE DESCRIPTION

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

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

➢ **Example: Jump table.**

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

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

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

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

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

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

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

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

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

➢ **Example: "@JMP_A" operation.**

; Before compiling program.

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

; After compiling program.

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

2.1.5 CHECKSUM CALCULATION

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

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

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

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

2.2 DATA MEMORY (RAM)

☞ ***128 X 8-bit RAM***

	Address	RAM location	
	000h	***General purpose area***	RAM Bank 0
	“		
	“		
	“		
	“		
	“		
BANK 0	07Fh		End of Bank 0
	080h	***System register***	080h~0FFh of Bank 0 store system registers (128 bytes).
	“		
	“		
	“		
	“		
	0FFh	***End of bank 0 area***	
	100h	***General purpose area***	RAM Bank 1
	“		
BANK 1	“		
	17Fh		End of Bank 1

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	MSPSTAT	MSPM1	MSPBUF	MSPADR	-	-	-	-	-	-	-	-	CM0M	CM0M1	-	-
A	-	-	-	-	-	-	-	-	TC0R0	TC0R1	TC0R2	TC0R3	TC0R4	TC0R5	TC0R6	TC0R7
B	-	-	-	-	SIOM	SIOR	SIOB	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	-	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	LPWS	PCL	PCH
D	P0	P1	-	-	-	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	-	-	-	P5UR	@HL	@YZ	-	P1OC	TC1D	-	-	-	-	-
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.
R = Working register and ROM look-up data buffer.
CM0M = Comparator configuration register.
SIOM = SIO configuration register.
SIOB = SIO data buffer.
PEDGE = P0.0 edge direction register.
INTEN = Interrupt enable register.
WDTR = Watchdog timer clear register.
Pn = Port n data buffer.
T0M = T0 mode register.
TC0M = TC0 mode register.
TC0Rn = TC0 pwm data buffer.
TC1M = TC1 mode register.
TC1R = TC1 auto-reload data buffer.
@HL = RAM HL indirect addressing index pointer.
STK0~STK7 = Stack 0 ~ stack 7 buffer.
MSPSTAT MSP status register
MSPBUF MSP buffer data

Y, Z = Working, @YZ and ROM addressing register.
PFLAG = ROM page and special flag register.
CM0M1 = Comparator configuration register.
SIOR = SIO clock register.
PnM = Port n input/output mode register.
INTRQ = Interrupt request register.
OSCM = Oscillator mode register.
PCH, PCL = Program counter.
PnUR = Port n pull-up resister control register.
T0C = T0 counting register.
TC0C = TC0 counting register.
LPWS = PWM channel selection.
TC1C = TC1 counting register.
TC1D = TC1 duty control register.
@YZ = RAM YZ indirect addressing index pointer.
STKP = Stack pointer buffer.
MSPM1 MSP mode register1
MSPADR MSP address register

2.2.1.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD	LVD36	LVD24		C	DC	Z	R/W	PFLAG
087H								RBNKS0	R/W	RBANK
090H		CKE	D_A	P	S	RED_WRT		BF	R/W	MSPSTAT
091H	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK	GCEN		R/W	MSPM1
092H	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0	R/W	MSPBUF
093H	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0	R/W	MSPADR
09CH	CM0EN	CM0OUT	CM0S1	CM0S0	CMCH3	CMCH2	CMCH1	CMCH0	R/W	CM0M
09DH					CMDB1	CMDB0	CM0OEN	CM0G	R/W	CM0M1
0A8H	TC0R07	TC0R06	TC0R05	TC0R04	TC0R03	TC0R02	TC0R01	TC0R00	W	TC0R0
0A9H	TC0R17	TC0R16	TC0R15	TC0R14	TC0R13	TC0R12	TC0R11	TC0R10	W	TC0R1
0AAH	TC0R27	TC0R26	TC0R25	TC0R24	TC0R23	TC0R22	TC0R21	TC0R20	W	TC0R2
0ABH	TC0R37	TC0R36	TC0R35	TC0R34	TC0R33	TC0R32	TC0R31	TC0R30	W	TC0R3
0ACH	TC0R47	TC0R46	TC0R45	TC0R44	TC0R43	TC0R42	TC0R41	TC0R40	W	TC0R4
0ADH	TC0R57	TC0R56	TC0R55	TC0R54	TC0R53	TC0R52	TC0R51	TC0R50	W	TC0R5
0AEH	TC0R67	TC0R66	TC0R65	TC0R64	TC0R63	TC0R62	TC0R61	TC0R60	W	TC0R6
0AFH	TC0R77	TC0R76	TC0R75	TC0R74	TC0R73	TC0R72	TC0R71	TC0R70	W	TC0R7
0B4H	SENB	START	SRATE1	SRATE0	MLSB	SCLKMD	CPOL	CPHA	R/W	SIOM
0B5H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	W	SIOR
0B6H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB
0B8H				P04M	P03M	P02M	P01M	P00M	R/W	P0M
0BFH				P00G1	P00G0				R/W	PEDGE
0C0H	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C5H	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H	CM0IRQ	TC1IRQ		T0IRQ	SIOIRQ		MSPIRQ	P00IRQ	R/W	INTRQ
0C9H	CM0IEN	TC1IEN		T0IEN	SIOIEN		MSPIEN	P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	LPWS7	LPWS6	LPWS5	LPWS4	LPWS3	LPWS2	LPWS1	LPWS0	R/W	LPWS
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH						PC10	PC9	PC8	R/W	PCH
0D0H			P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D5H	P57	P56	P55	P54	P53	P52	P51	P50	R/W	P5
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				PWMDR	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0		TC1CKS0		PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H								P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E5H	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E9H				P52OC	P51OC	P50OC			W	P1OC
0EAH	TC1D7	TC1D6	TC1D5	TC1D4	TC1D3	TC1D2	TC1D1	TC1D0	R/W	TC1D
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H						S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H						S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H						S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H						S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H						S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH						S2PC10	S2PC9	S2PC8	R/W	STK2H

0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH						S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH						S0PC10	S0PC9	S0PC8	R/W	STK0H

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

2.2.2 ACCUMULATOR

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

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

```
; Read ACC data and store in BUF data memory.
                MOV         BUF, A
; Write a immediate data into ACC.
                MOV         A, #0FH
; Write ACC data from BUF data memory.
                MOV         A, BUF
; or
                B0MOV       A, BUF
```

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

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

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

                RETI                ; Exit interrupt service vector
```

2.2.3 PROGRAM FLAG

The PFLAG register contains the arithmetic status of ALU operation, system reset status and LVD detecting status. NT0, NPD bits indicate system reset status including power on reset, LVD reset, reset by external pin active and watchdog reset. C, DC, Z bits indicate the result status of ALU operation. LVD24, LVD36 bits indicate LVD detecting power voltage status.

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

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

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

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

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

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

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

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

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

2.2.4 PROGRAM COUNTER

The program counter (PC) is a 11-bit binary counter separated into the high-byte 3 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 10.

	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	-	-	-	-	-	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	-	-	-	-	-	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.
```

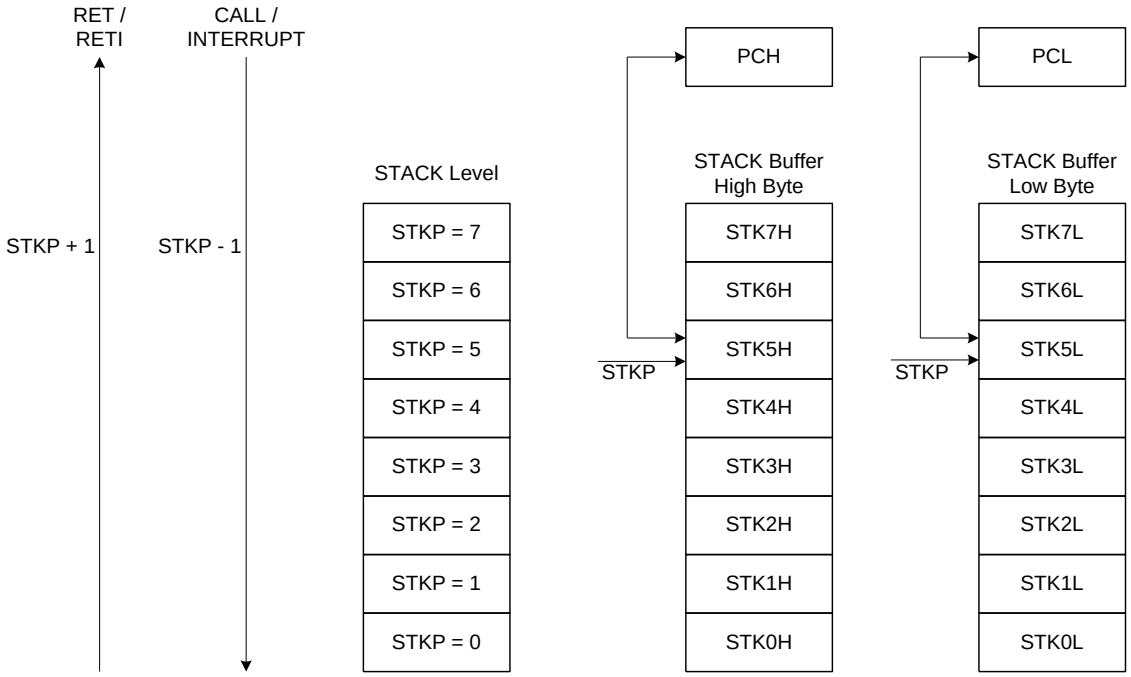
- **Example: Indirectly addressing mode with @YZ register**

```
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, 11-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	-	-	-	-	-	SnPC10	SnPC9	SnPC8
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	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 noise filter option, watchdog timer operation, LVD option, reset pin option and OTP ROM security control. The code option items are as following table:

Code Option	Content	Function Description
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.
Fcpu	Fhosc/1	Instruction cycle is 1 oscillator clocks.
	Fhosc/2	Instruction cycle is 2 oscillator clocks.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
	Fhosc/16	Instruction cycle is 16 oscillator clocks.
Reset_Pin	Reset	Enable External reset pin.
	P05	Enable P0.5 input only without pull-up resister.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
LVD	LVD_L	LVD will reset chip if VDD is below 2.0V
	LVD_M	LVD will reset chip if VDD is below 2.0V Enable LVD24 bit of PFLAG register for 2.4V low voltage indicator.
	LVD_H	LVD will reset chip if VDD is below 2.4V Enable LVD36 bit of PFLAG register for 3.6V low voltage indicator.
	LVD_MAX	LVD will reset chip if VDD is below 3.6V

2.5.1 Fcpu code option

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

2.5.2 Reset_Pin code option

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

- **Reset:** The reset pin is external reset function. When falling edge trigger occurring, the system will be reset.
- **P05:** Set reset pin to general input only pin (P0.5). The external reset function is disabled and the pin is input pin.

2.5.3 Security code option

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

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

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

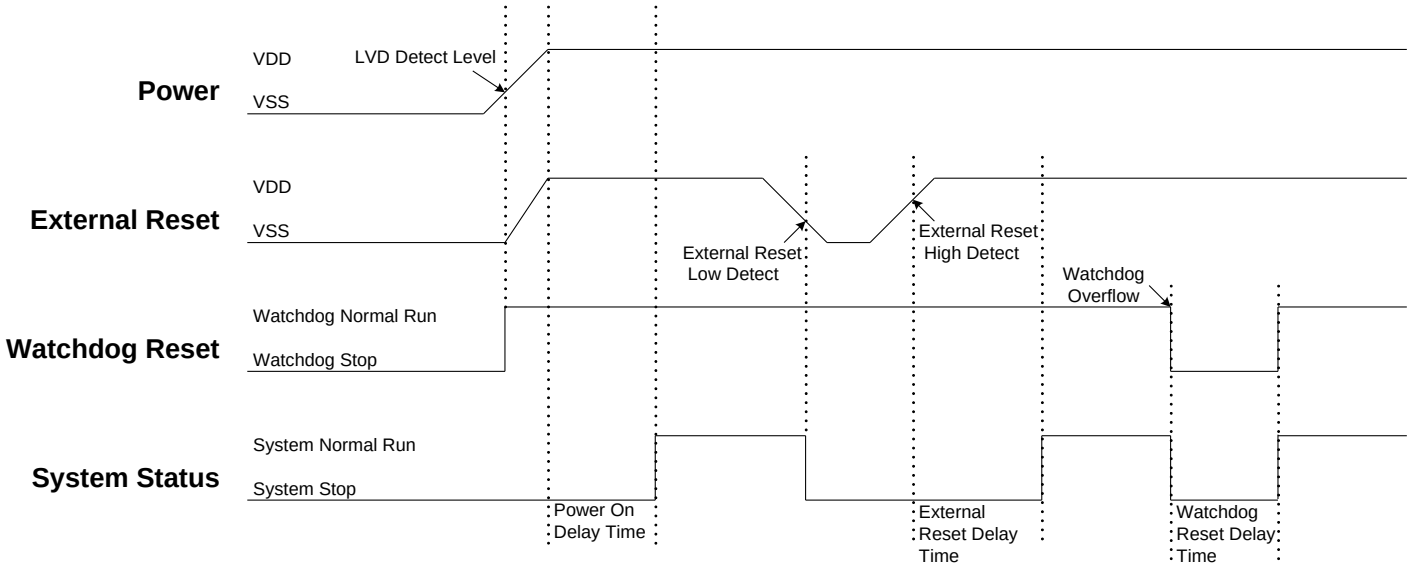
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

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

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

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

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



3.2 POWER ON RESET

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

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

3.3 WATCHDOG RESET

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

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

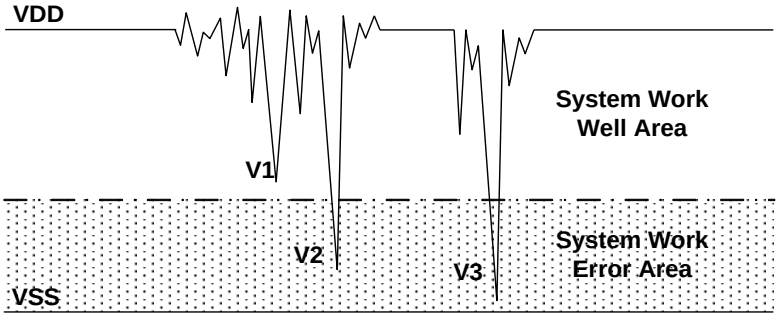
Watchdog timer application note is as following.

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

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

3.4 BROWN OUT RESET

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



Brown Out Reset Diagram

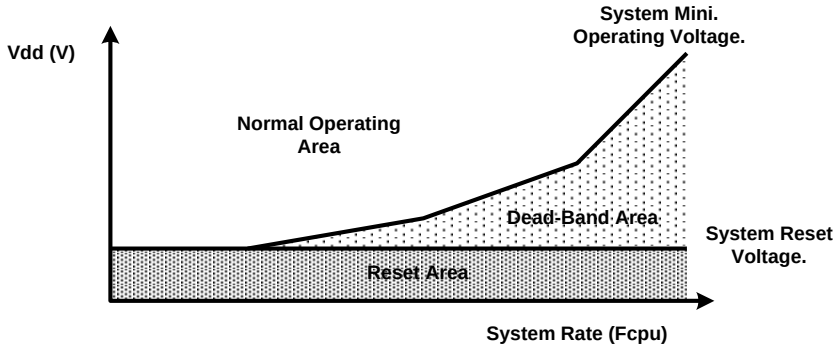
The power dropping might through the voltage range that's the system dead-band. The dead-band means the power range can't offer the system minimum operation power requirement. The above diagram is a typical brown out reset diagram. There is a serious noise under the VDD, and VDD voltage drops very deep.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

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

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

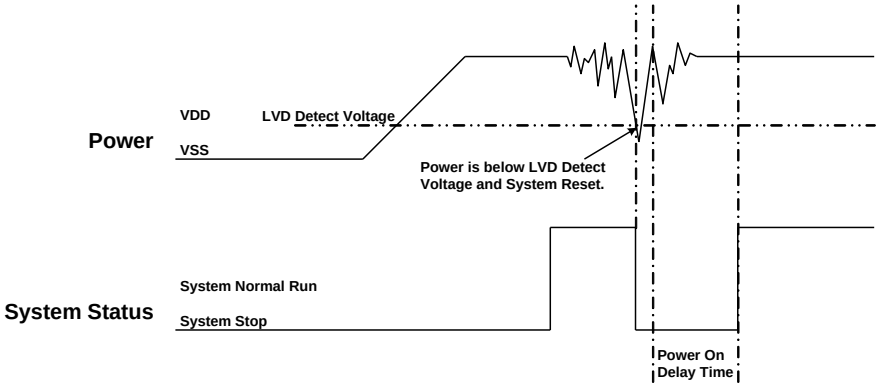
3.5 THE SYSTEM OPERATING VOLTAGE

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



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

3.6 LOW VOLTAGE DETECTOR (LVD)



The LVD (low voltage detector) is built-in Sonix 8-bit MCU to be brown out reset protection. When the VDD drops and is below LVD detect voltage, the LVD would be triggered, and the system is reset. The LVD detect level is different by each MCU. The LVD voltage level is a point of voltage and not easy to cover all dead-band range. Using LVD to improve brown out reset is depend on application requirement and environment. If the power variation is very deep, violent and trigger the LVD, the LVD can be the protection. If the power variation can touch the LVD detect level and make system work error, the LVD can't be the protection and need to other reset methods. More detail LVD information is in the electrical characteristic section.
The LVD is three levels design (2.0V/2.4V/3.6V) and controlled by LVD code option. The 2.0V LVD is always enable for power on reset and Brown Out reset. The 2.4V LVD includes LVD reset function and flag function to indicate VDD status function. The 3.6V includes flag function to indicate VDD status. LVD flag function can be an **easy low battery detector**. LVD24, LVD36 flags indicate VDD voltage level. For low battery detect application, only checking LVD24, LVD36 status to be battery status. This is a cheap and easy solution.

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

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

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

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

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

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

3.7 BROWN OUT RESET IMPROVEMENT

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

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

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

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

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

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

3.8 EXTERNAL RESET

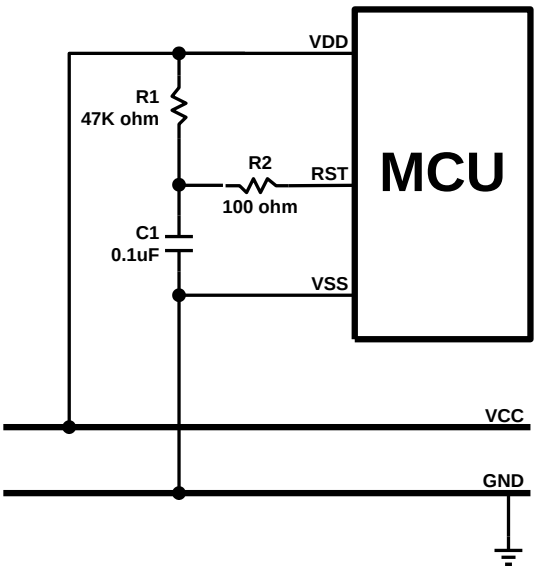
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

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

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

3.9 EXTERNAL RESET CIRCUIT

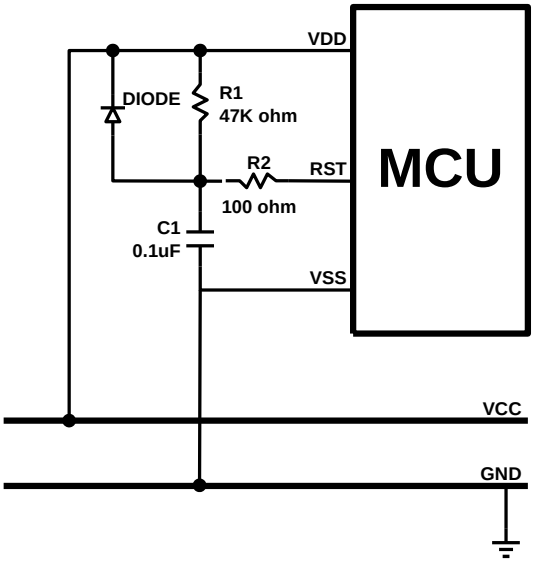
3.9.1 Simply RC Reset Circuit



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

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

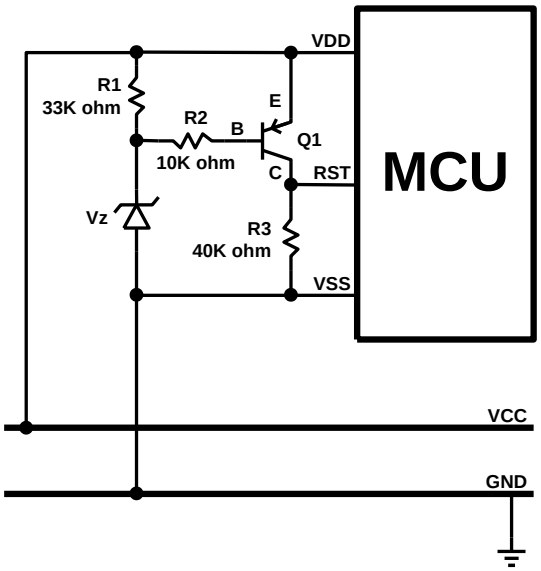
3.9.2 Diode & RC Reset Circuit



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

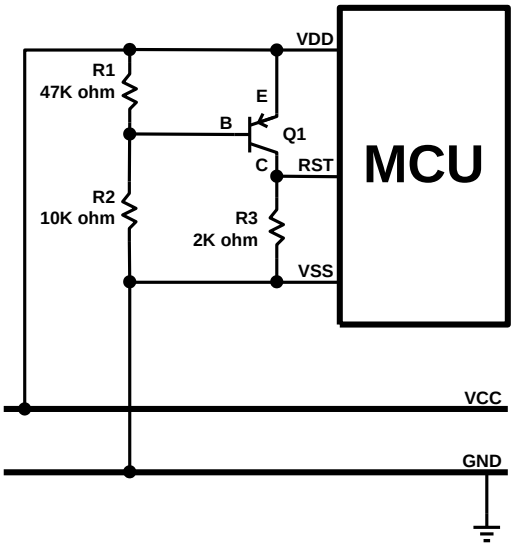
✷ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.9.3 Zener Diode Reset Circuit



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

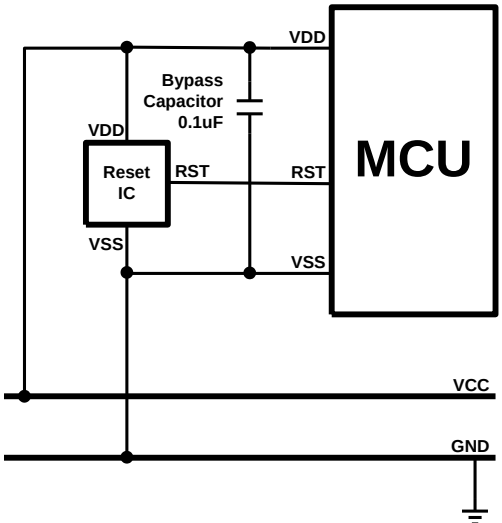
3.9.4 Voltage Bias Reset Circuit



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

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

3.9.5 External Reset IC



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

4 SYSTEM CLOCK

4.1 OVERVIEW

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

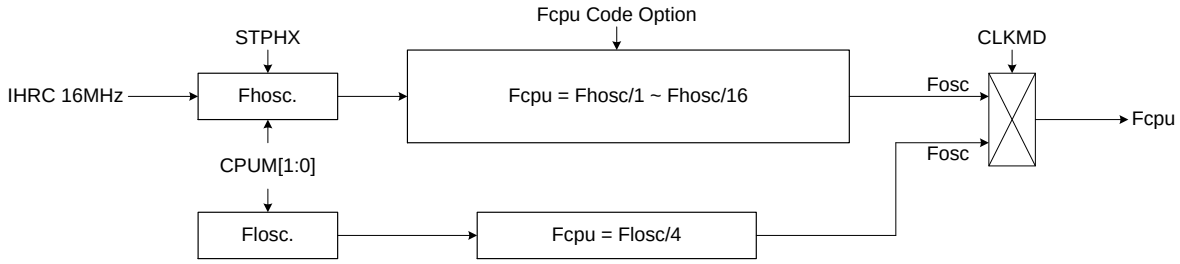
- **High-speed oscillator**

Internal high-speed oscillator is 16MHz RC type called “**IHRC**”.

- **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, 32KHz@5V).
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.2 FCPU (INSTRUCTION CYCLE)

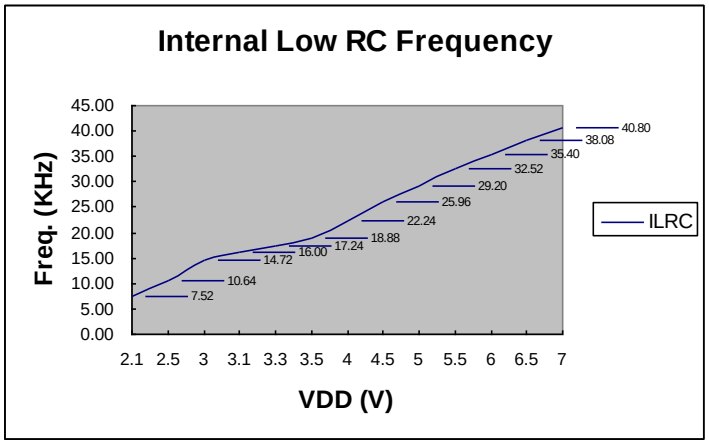
The system clock rate is instruction cycle called “**Fcpu**” which is divided from the system clock source and decides the system operating rate. Fcpu rate is selected by Fcpu code option and the range is **Fhosc/1~Fhosc/16** under system normal mode. If Fcpu code option is Fhosc/4, the Fcpu frequency 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 high-speed clock is internal high-speed RC type oscillator and is the system clock source. The internal high-speed oscillator is 16MHz RC type. The accuracy is ± 2% under commercial condition.

4.4 SYSTEM LOW-SPEED CLOCK

The system low clock source is the internal low-speed oscillator built in the micro-controller. The low-speed oscillator uses RC type oscillator circuit. The frequency is affected by the voltage and temperature of the system. In common condition, the frequency of the RC oscillator is about 16KHz at 3V and 32KHz at 5V. The relation between the RC frequency and voltage is as the following figure.



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

- ***Flosc = Internal low RC oscillator (about 16KHz @3V, 32KHz @5V).***
- ***Slow mode Fcpu = Flosc / 4***

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

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

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

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

4.5 OSCM REGISTER

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

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

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

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

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

"STPHX" bit controls internal high speed RC type oscillator operation. When "STPHX=0", the internal high speed RC type oscillator active. When "STPHX=1", the internal high speed RC type oscillator are disabled. T

4.6 SYSTEM CLOCK MEASUREMENT

Under design period, the users can measure system clock speed by software instruction cycle (Fcpu).

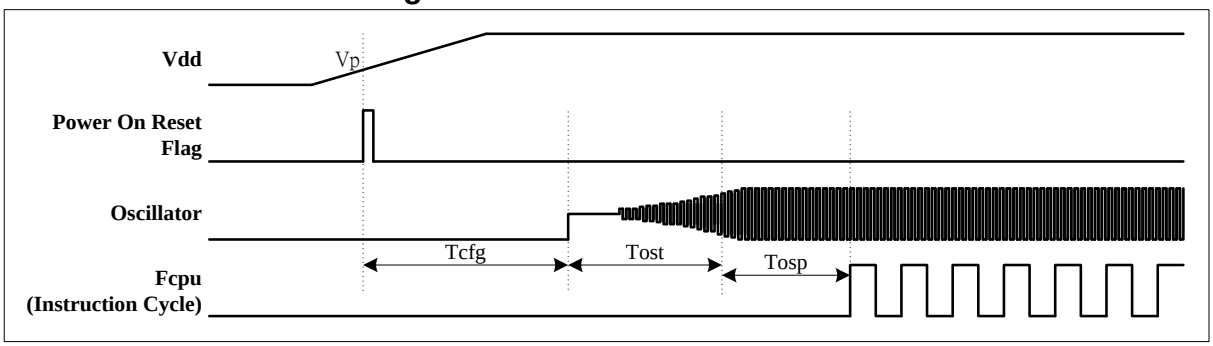
➢ **Example: Fcpu instruction cycle of external oscillator.**

```
        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
```

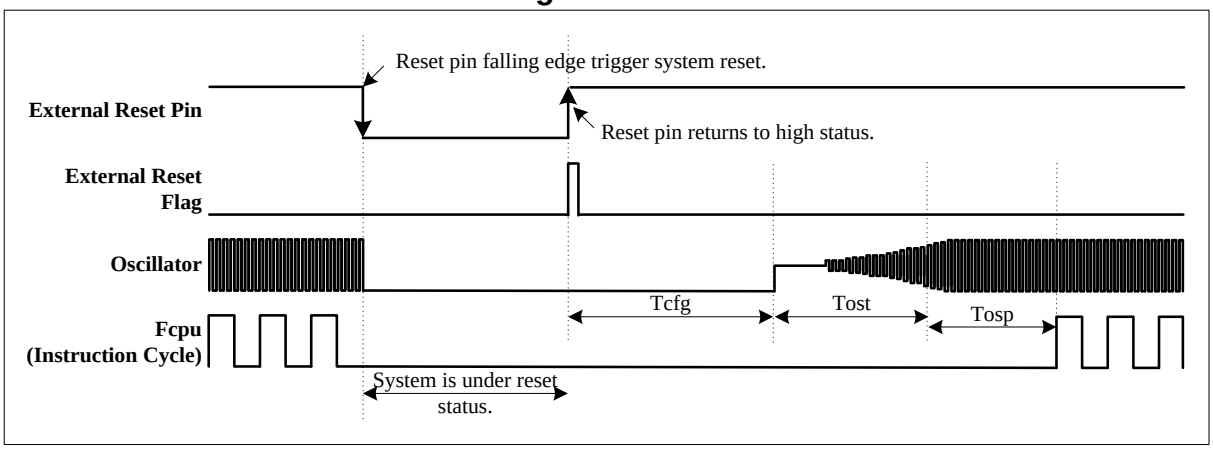
4.7 SYSTEM CLOCK TIMING

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. 32*F_{hosc}……Internal high-speed RC type oscillator.	2us @ F_{hosc} = 16MHz

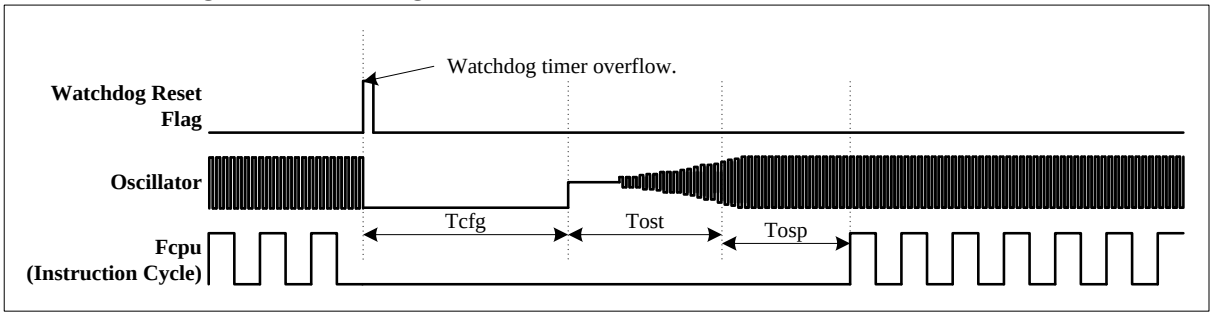
➢ Power On Reset Timing



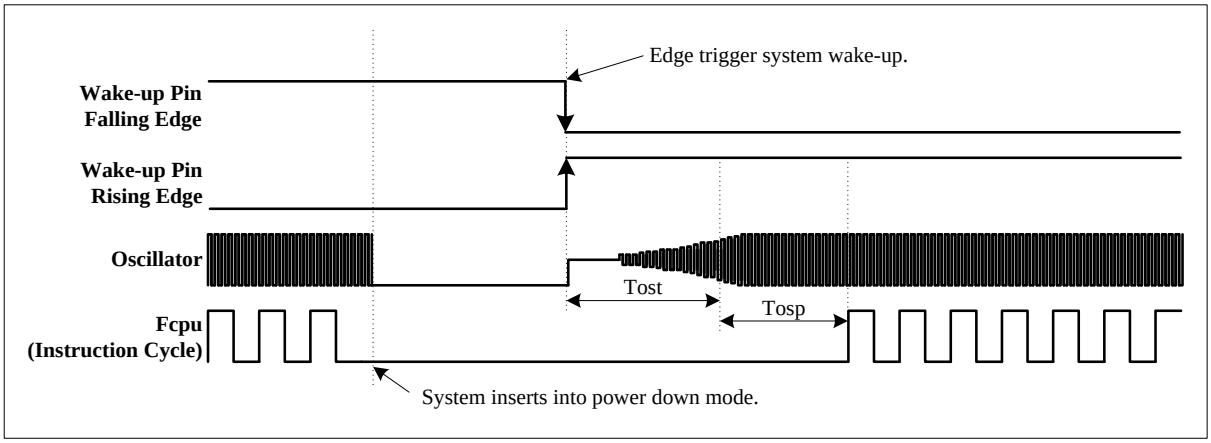
➢ External Reset Pin Reset Timing



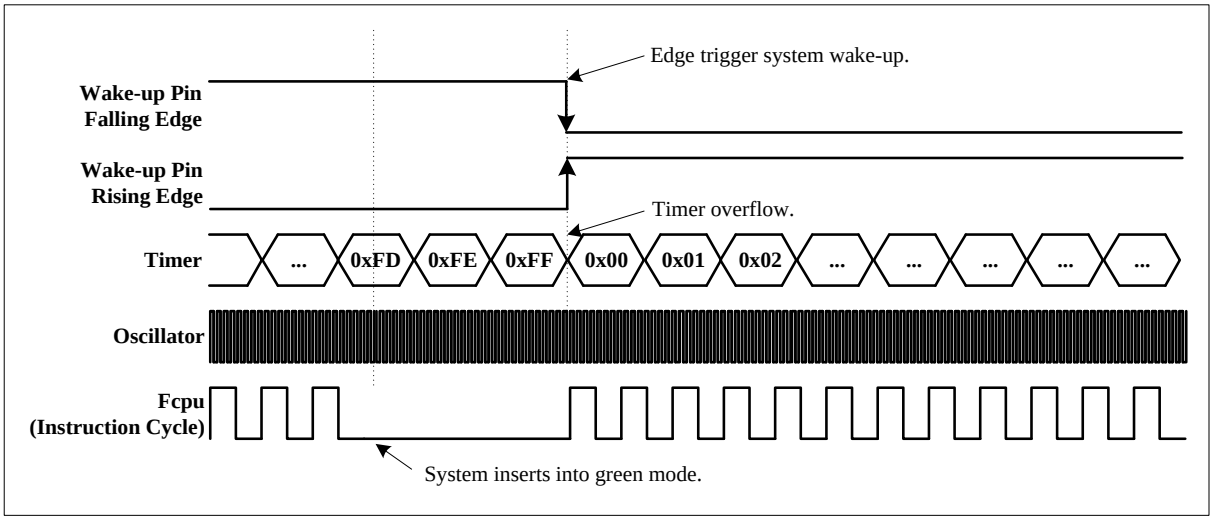
➢ Watchdog Reset Timing



➢ Power Down Mode Wake-up Timing

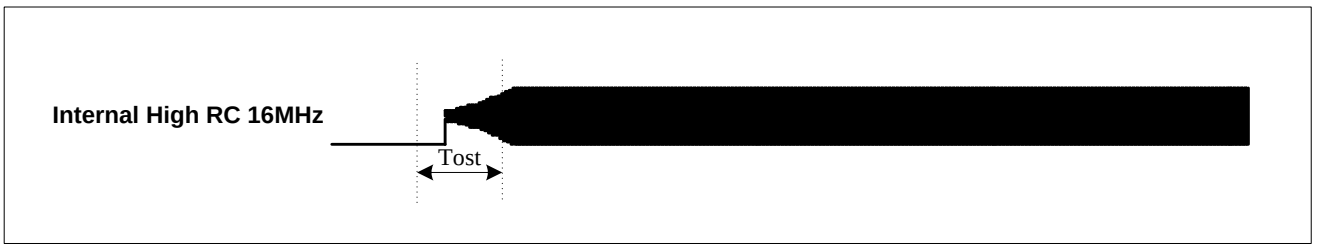


➢ Green Mode Wake-up Timing



➢ 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.



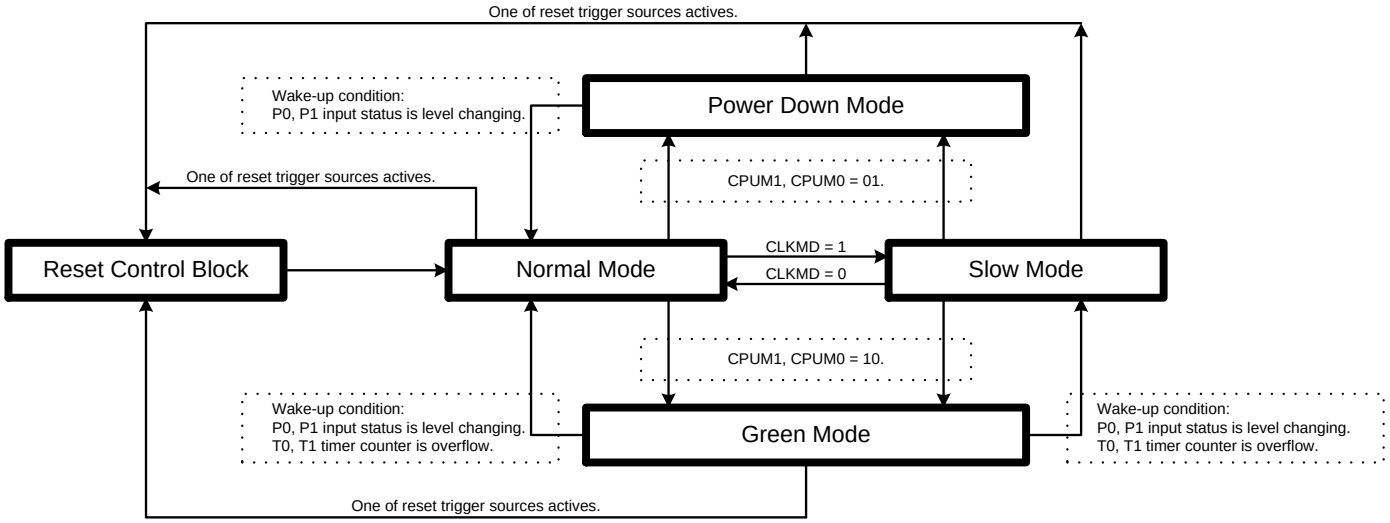
5 SYSTEM OPERATION MODE

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	By STPHX	Stop
ILRC	Running	Running	Running	Stop
CPU instruction	Executing	Executing	Stop	Stop
T0 timer	By T0ENB	By T0ENB	By T0ENB	Inactive
TC0 timer	By TC0ENB	By TC0ENB	By TC0ENB (PWM active)	Inactive
TC1 timer	By TC1ENB	By TC1ENB	By TC1ENB (PWM active)	Inactive
T1 timer	By T1ENB	By T1ENB	By T1ENB	Inactive
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option
Internal interrupt	All active	All active	T0	All inactive
External interrupt	All active	All active	All active	All inactive
Wakeup source	-	-	P0, P1, T0, CMP0, Reset	P0, P1, Reset

- **IHRC:** Internal high-speed oscillator RC type.
- **ILRC:** Internal low-speed oscillator RC type.

5.2 NORMAL MODE

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

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

5.3 SLOW MODE

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

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

5.4 POWER DOWN MODE

The power down mode is the system ideal status. No program execution and oscillator operation. Whole chip is under low power consumption status under 1uA. The power down mode is waked up by P0, P1 hardware level change trigger. P1 wake-up function is controlled by P1W register. Any operating modes into power down mode, the system is waked up to normal mode. Inserting power down mode is controlled by CPUM0 bit of OSCM register. When CPUM0=1, the system inserts into power down mode. After system wake-up from power down mode, the CPUM0 bit is disabled (zero status) automatically.

- The program stops executing, and full functions are disabled.
- All oscillators including external high speed oscillator, internal high speed oscillator and internal low speed oscillator stop.
- The power consumption is under 1uA.
- The system inserts into normal mode after wake-up from power down mode.
- The power down mode wake-up source is P0 and P1 level change trigger.

✸ ***Note: If the system is in normal mode, to set STPHX=1 to disable the high clock oscillator. The system is under no system clock condition. This condition makes the system stay as power down mode, and can be wake-up by P0, P1 level change trigger.***

5.5 GREEN MODE

The green mode is another system ideal status not like power down mode. In power down mode, all functions and hardware devices are disabled. But in green mode, the system clock source keeps running, so the power consumption of green mode is larger than power down mode. In green mode, the program isn't executed, but the timer with wake-up function actives as enabled, and the timer clock source is the non-stop system clock. The green mode has 2 wake-up sources. One is the P0, P1 level change trigger wake-up. The other one is internal timer with wake-up function occurring overflow. That's mean users can setup one fix period to timer, and the system is waked up until the time out. Inserting green mode is controlled by CPUM1 bit of OSCM register. When CPUM1=1, the system inserts into green mode. After system wake-up from green mode, the CPUM1 bit is disabled (zero status) automatically.

- The program stops executing, and full functions are disabled.
- Only the timer with wake-up function actives.
- The oscillator to be the system clock source keeps running, and the other oscillators operation is depend on system operation mode configuration.
- If inserting green mode from normal mode, the system insets to normal mode after wake-up.
- If inserting green mode from slow mode, the system insets to slow mode after wake-up.
- The green mode wake-up sources are P0, P1 level change trigger and unique time overflow.
- PWM and buzzer output functions active in green mode, but the timer can't wake-up the system as overflow.

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

5.6 OPERATING MODE CONTROL MACRO

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

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

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

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

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

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

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

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

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

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

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

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

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

5.7 WAKEUP

5.7.1 OVERVIEW

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

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

5.7.2 WAKEUP TIME

When the system is in power down mode (sleep mode), the high clock oscillator stops. When waked up from power down mode, MCU waits for 32 internal high-speed oscillator clocks as the wakeup time to stable the oscillator circuit. After the wakeup time, the system goes into the normal mode.

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

The value of the wakeup time is as the following.

The Wakeup time = 1/Fhosc * 32 (sec) + high clock start-up time

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

The wakeup time = 1/Fhosc * 32 = 2 us (Fhosc = 16MHz)

5.7.3 P1W WAKEUP CONTROL REGISTER

Under power down mode (sleep mode) and green mode, the I/O ports with wakeup function are able to wake the system up to normal mode. The wake-up trigger edge is level changing. When wake-up pin occurs rising edge or falling edge, the system is waked up by the trigger edge. The Port 0 and Port 1 have wakeup function. Port 0 wakeup function always enables, but the Port 1 is controlled by the P1W register.

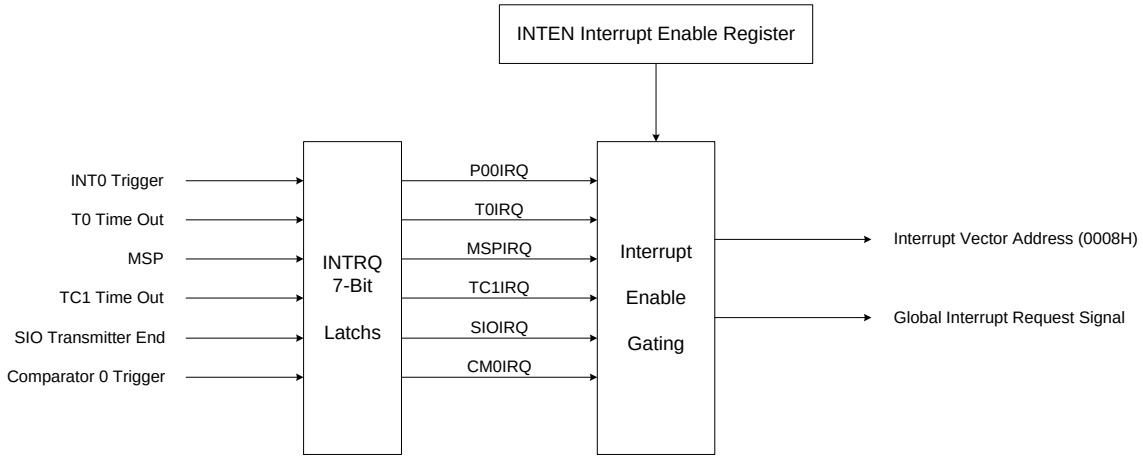
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 6 interrupt sources, including 5 internal interrupt (T0/MSP/TC1/CM0/SIO) and 1 external interrupt (INT0). The external interrupt can wakeup the chip while the system is switched from power down mode to high-speed normal mode, and interrupt request is latched until return to normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to “0” for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to “1” to accept the next interrupts’ request. The interrupt request signals are stored in INTRQ register.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	CM0IEN	TC1IEN	-	T0IEN	SIOIEN	-	MSPIEN	P00IEN
Read/Write	R/W	R/W	-	R/W	R/W	-	R/W	R/W
After reset	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 **MSPIEN:** MSP interrupt control bit.
0 = Disable MSP interrupt function.
1 = Enable MSP interrupt function.

Bit 3 **SIOIEN:** SIO interrupt control bit.
0 = Disable SIO interrupt function.
1 = Enable SIO interrupt function.

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

Bit 6 **TC1IEN:** TC1 timer interrupt control bit.
0 = Disable TC1 interrupt function.
1 = Enable TC1 interrupt function.

Bit 7 **CM0IEN:** Comparator 0 interrupt control bit.
0 = Disable comparator 0 interrupt function.
1 = Enable comparator 0 interrupt function.

6.3 INTRQ INTERRUPT REQUEST REGISTER

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

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	CM0IRQ	TC1IRQ	-	T0IRQ	SIOIRQ	-	MSPIRQ	P00IRQ
Read/Write	R/W	R/W	-	R/W	R/W	-	R/W	R/W
After reset	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 **MSPIRQ:** MSP interrupt request flag.
0 = None MSP interrupt request.
1 = MSP interrupt request.

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

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

Bit 6 **TC1IRQ:** TC1 timer interrupt request flag.
0 = None TC1 interrupt request.
1 = TC1 interrupt request.

Bit 7 **CM0IRQ:** Comparator 0 interrupt request flag.
0 = None comparator 0 interrupt request.
1 = Comparator 0 interrupt request.

6.4 GIE GLOBAL INTERRUPT OPERATION

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

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

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

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

```
B0BSET      FGIE        ; Enable GIE
```

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

6.5 PUSH, POP ROUTINE

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

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

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

```
                ORG     0
                JMP     START

                ORG     8
                JMP     INT_SERVICE

                ORG     10H
START:
                …

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

                RETI                    ; Exit interrupt service vector
                …
                ENDP
```

6.6 EXTERNAL INTERRUPT OPERATION (INT0)

INT0 is external interrupt trigger source and builds in edge trigger configuration function. When the external edge trigger occurs, the external interrupt request flag will be set to “1” no matter the external interrupt control bit enabled or disable. When external interrupt control bit is enabled and external interrupt edge trigger is occurring, the program counter will jump to the interrupt vector (ORG 8) and execute interrupt service routine.

The external interrupt builds in wake-up latch function. That means when the system is triggered wake-up from power down mode, the wake-up source is external interrupt source (P0.0), and the trigger edge direction matches interrupt edge configuration, the trigger edge will be latched, and the system executes interrupt service routine fist after wake-up.

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

Bit[4:3] **P00G[1:0]:** INT0 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

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

```
        MOV     A, #18H
        B0MOV   PEDGE, A        ; Set INT0 interrupt trigger as bi-direction edge.

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

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

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

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

        B0BCLR  FP00IRQ         ; Reset P00IRQ
        …                       ; INT0 interrupt service routine
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.
        RETI                    ; Exit interrupt vector
```

6.7 T0 INTERRUPT OPERATION

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

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

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

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

        B0BSET      FGIE        ; Enable GIE
```

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

```
        ORG         8           ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

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

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

        B0BCLR      FT0IRQ      ; Reset T0IRQ
        MOV         A, #74H
        B0MOV       T0C, A      ; Reset T0C.
        …                       ; T0 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.


        RETI                    ; Exit interrupt vector
```

6.8 TC1 INTERRUPT OPERATION

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

➢ **Example: TC1 interrupt request setup. Fcpu = 16Mhz / 16.**

```
        B0BCLR      FTC1IEN         ; Disable TC1 interrupt service
        B0BCLR      FTC1ENB         ; Disable TC1 timer
        MOV         A, #20H         ;
        B0MOV       TC1M, A         ; Set TC1 clock = Fcpu / 64
        MOV         A, #64H         ; Set TC1C initial value = 64H
        B0MOV       TC1C, A         ; Set TC1 interval = 10 ms

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

        B0BSET      FGIE            ; Enable GIE
```

➢ **Example: TC1 interrupt service routine.**

```
        ORG         8               ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

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

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

        B0BCLR      FTC1IRQ         ; Reset TC1IRQ
        MOV         A, #74H
        B0MOV       TC1C, A         ; Reset TC1C.
        …                           ; TC1 interrupt service routine
        …
EXIT_INT:
        …                           ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

6.9 SIO INTERRUPT OPERATION

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

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

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

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

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

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

        B0BCLR      FSIOIRQ     ; Reset SIOIRQ
        …                       ; SIO interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.10 COMPARATOR INTERRUPT OPERATION (CMP0)

Sonix provides one comparator with interrupt function in the micro-controller. The comparator interrupt trigger edge direction is the rising edge of comparator output . 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.

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

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

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

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

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

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

7 I/O PORT

7.1 OVERVIEW

The micro-controller builds in 22 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
		CM0O	AC	CM0EN=1, CM0OEN=1
P0.1	I/O	LPW0	DC	TC0ENB=1, LPWS0=1
		CM0N12	AC	CM0EN=1, CMCH[3:0] = 1100
P0.2	I/O	LPW1	DC	TC0ENB=1, LPWS1=1
		CM0N13	AC	CM0EN=1, CMCH[3:0] = 1101
P0.3	I/O	LPW2	DC	TC0ENB=1, LPWS2=1
		CM0N14	AC	CM0EN=1, CMCH[3:0] = 1110
P0.4	I/O	LPW3	DC	TC0ENB=1, LPWS3=1
		CM0N15	AC	CM0EN=1, CMCH[3:0] = 1111
P0.5	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P1.0	I/O	CM0N0	AC	CM0EN=1, CMCH[3:0] = 0000
P1.1	I/O	CM0N1	AC	CM0EN=1, CMCH[3:0] = 0001
P1.2	I/O	CM0N2	AC	CM0EN=1, CMCH[3:0] = 0010
P1.3	I/O	CM0N3	AC	CM0EN=1, CMCH[3:0] = 0011
P1.4	I/O	CM0N4	AC	CM0EN=1, CMCH[3:0] = 0100
P1.5	I/O	CM0N5	AC	CM0EN=1, CMCH[3:0] = 0101
P1.6	I/O	CM0N6	AC	CM0EN=1, CMCH[3:0] = 0110
P1.7	I/O	CM0N7	AC	CM0EN=1, CMCH[3:0] = 0111
P5.0	I/O	SCK	DC	SENB=1
		SCL	DC	MSPENB=1
P5.1	I/O	SI	DC	SENB=1
		SDA	DC	MSPENB=1
P5.2	I/O	SO	DC	SENB=1
P5.3	I/O	PWM1	DC	TC1ENB=1, PWM1OUT=1
P5.4	I/O	LPW4	DC	TC0ENB=1, LPWS4=1
		CM0N8	AC	CM0EN=1, CMCH[3:0] = 1000
P5.5	I/O	LPW5	DC	TC0ENB=1, LPWS5=1
		CM0N9	AC	CM0EN=1, CMCH[3:0] = 1001
P5.6	I/O	LPW6	DC	TC0ENB=1, LPWS6=1
		CM0N10	AC	CM0EN=1, CMCH[3:0] = 1010
P5.7	I/O	LPW7	DC	TC0ENB=1, LPWS7=1
		CM0N11	AC	CM0EN=1, CMCH[3:0] = 1011

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

7.2 I/O PORT MODE

The port direction is programmed by PnM register. All I/O ports can select input or output direction.

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	-	-	-	P04M	P03M	P02M	P01M	P00M
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

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

✸ ***Note:***
1. ***Users can program them by bit control instructions (B0BSET, B0BCLR).***
2. ***P0.5 input only pin, and the P0M.5 keeps "1".***

➢ **Example: I/O mode selecting**

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

MOV     A, #0FFH    ; Set all ports to be output mode.
B0MOV   P0M, A
B0MOV   P1M,A
B0MOV   P5M, A

B0BCLR  P1M.0       ; Set P1.0 to be input mode.

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

7.3 I/O PULL UP REGISTER

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

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

✸ ***Note: P0.5 is input only pin and without pull-up resister. The P0UR.5 keeps "1".***

➢ **Example: I/O Pull up Register**

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

7.4 I/O PORT DATA REGISTER

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

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	P17	P16	P15	P14	P13	P12	P11	P10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	P57	P56	P55	P54	P53	P52	P51	P50
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

✸ ***Note: The P05 keeps "1" when external reset enable by code option.***

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

```
B0MOV   A, P0       ; Read data from Port 0
B0MOV   A, P1       ; Read data from Port 1
B0MOV   A, P5       ; Read data from Port 5
```

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

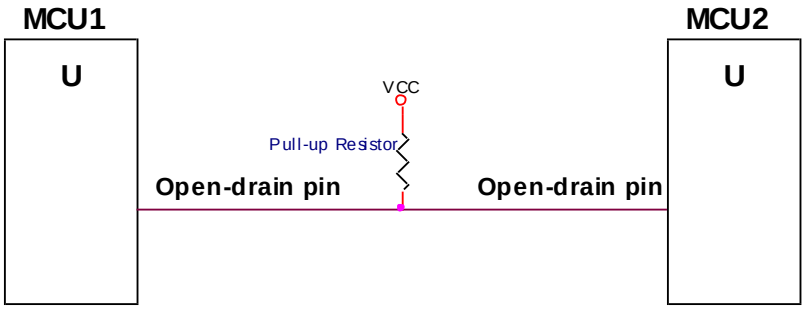
```
MOV     A, #0FFH    ; Write data FFH to all Port.
B0MOV   P0, A
B0MOV   P1, A
B0MOV   P5, A
```

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

```
B0BSET  P1.0        ; Set P1.0 and P5.3 to be "1".
B0BSET  P5.3

B0BCLR  P1.0        ; Set P1.0 and P5.3 to be "0".
B0BCLR  P5.3
```

7.5 I/O OPEN-DRAIN REGISTER

P5.0~P5.4 built in open-drain function. These pins must be set as output mode when enable open-drain function. Open-drain external circuit is as following.



The pull-up resistor is necessary. Open-drain output high is driven by pull-up resistor. Output low is sunken by MCU's pin.

0E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1OC	-	-	-	P52OC	P51OC	P50OC	-	-
Read/Write	-	-	-	W	W	W	-	-
After reset	-	-	-	0	0	0	-	-

Bit 2 **P50OC:** P5.0 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 3 **P51OC:** P5.1 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 4 **P52OC:** P5.2 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

➢ **Example: Enable P5.0 to open-drain mode and output high.**

```
B0BSET    P5.0          ; Set P5.0 buffer high.

B0BSET    P50M          ; Enable P5.0 output mode.
MOV       A, #04H       ; Enable P5.0 open-drain function.
B0MOV     P1OC, A
```

✸ ***Note: P1OC is write only register. Setting P10OC must be used "MOV" instructions.***

➢ **Example: Disable open-drain mode.**

```
MOV       A, #0         ; Disable open-drain function.
B0MOV     P1OC, A
```

✸ ***Note: After disable open-drain function, I/O mode returns to last I/O mode.***

8 TIMERS

8.1 WATCHDOGTIMER

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.**

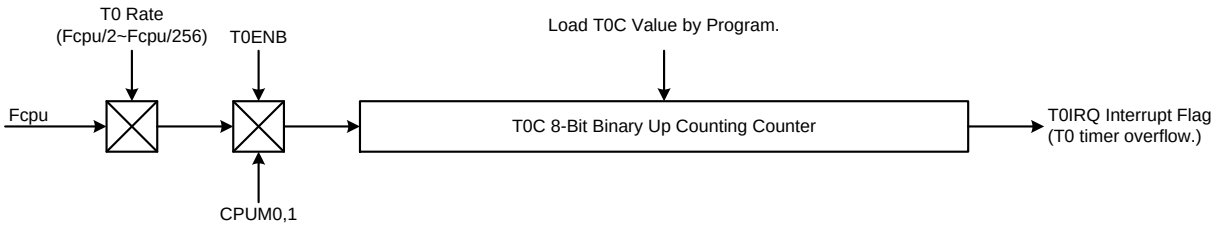
```
Main:
                …                               ; Check I/O.
                …                               ; Check RAM
Err:            JMP $                           ; I/O or RAM error. Program jump here and don't
                                                ; clear watchdog. Wait watchdog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

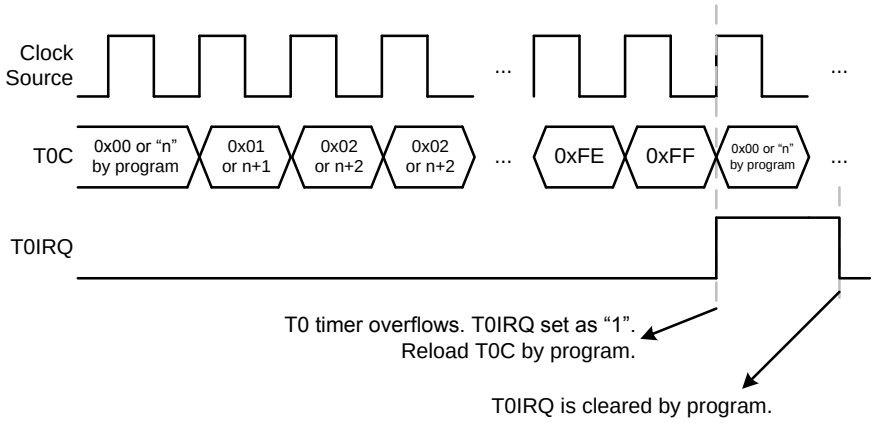
The T0 timer is an 8-bit binary up timer with basic timer function. The basic timer function supports flag indicator (T0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T0M, T0C registers. The T0 builds in green mode wake-up function. When T0 timer overflow 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:**

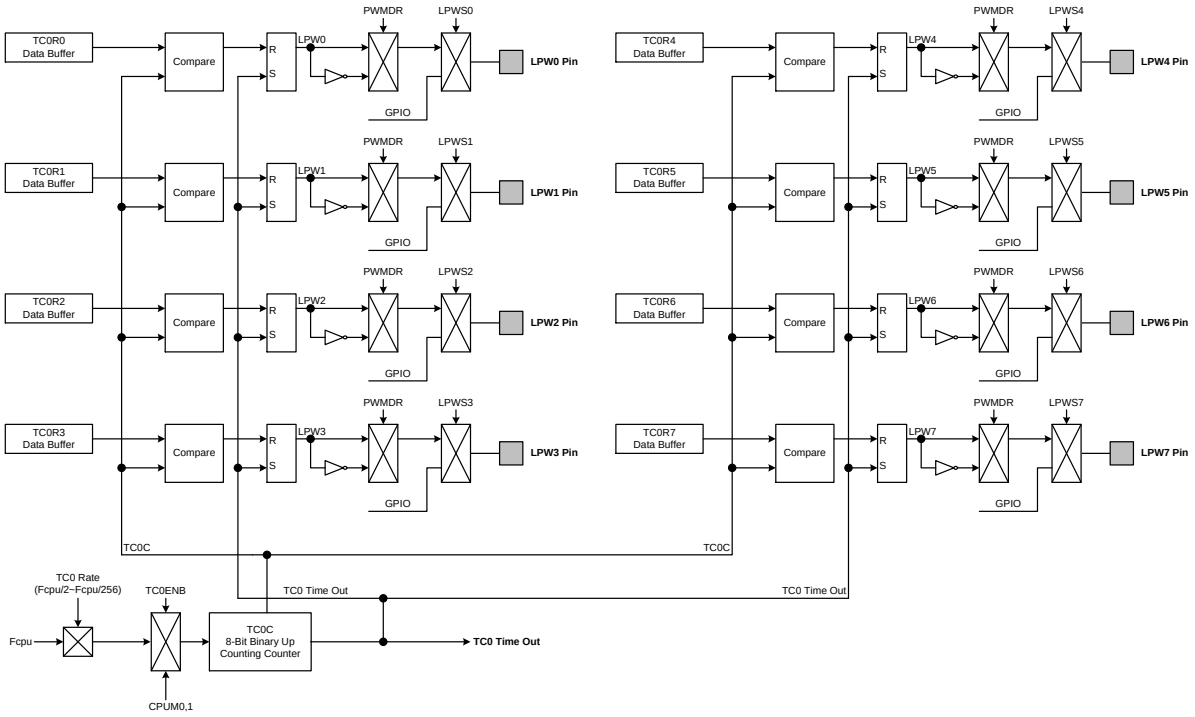
```
; Reset T0 timer.
        CLR     T0M             ; Clear T0M register.

; Set T0 clock source and T0 rate.
        MOV     A, #0nnn00000b
        B0MOV   T0M, A

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

; Clear T0IRQ
        B0BCLR  FT0IRQ

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

8.3 8-CH LED PWM Driver (TC0)

8.3.1 OVERVIEW

The TC0 timer is an 8-bit binary up timer, and the main purpose is to generator 8-ch PWM outputs. TC0 doesn't support normal timer function and interrupt function. TC0C 256-step period decides PWM's cycle through TC0rate[2:0] setting TC0 clock rate. TC0R0~TC0R7 decide each channel of 8 PWMs' duty. When TC0C=0, PWM output pulse status. When TC0C=TC0R, PWM exchanges to idle status to generator PWM's duty. The PWM phase is controlled by PWMDR bit. When PWMDR=0, PWM status is idle low and high duty. When PWMDR=1, PWM status is idle high and low duty. The main purpose of the TC0 timer is as following. TC0Rn is not auto-reload designed. If modify PWM duty as PWM outputting through TC0Rn registers, the PWM output waveform will change immediately.

- ☞ **PWM output:** The PWM is duty/cycle programmable controlled by TC0rate, TC0Rn and LPWS register.
- ☞ **Green mode function:** TC0's PWM function keeps running in green mode.



8.3.2 TC0M CONTROL REGISTER

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	-	-	-	PWMDR
Read/Write	R/W	R/W	R/W	R/W	-	-	-	R/W
After reset	0	0	0	0	-	-	-	0

Bit 0 **PWMDR:** PWM output phase control bit.
0 = High pulse and low idle status.
1 = Low pulse and high idle status.

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

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

0A8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R0	TC0R07	TC0R06	TC0R05	TC0R04	TC0R03	TC0R02	TC0R01	TC0R00
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0A9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R1	TC0R17	TC0R16	TC0R15	TC0R14	TC0R13	TC0R12	TC0R11	TC0R10
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0AAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R2	TC0R27	TC0R26	TC0R25	TC0R24	TC0R23	TC0R22	TC0R21	TC0R20
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0ABH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R3	TC0R37	TC0R36	TC0R35	TC0R34	TC0R33	TC0R32	TC0R31	TC0R30
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0ACH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R4	TC0R47	TC0R46	TC0R45	TC0R44	TC0R43	TC0R42	TC0R41	TC0R40
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0ADH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R5	TC0R57	TC0R56	TC0R55	TC0R54	TC0R53	TC0R52	TC0R51	TC0R50
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R6	TC0R67	TC0R66	TC0R65	TC0R64	TC0R63	TC0R62	TC0R61	TC0R60
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R7	TC0R77	TC0R76	TC0R75	TC0R74	TC0R73	TC0R72	TC0R71	TC0R70
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

- **Example: To calculation TC0R value to obtain 10ms TC0 pulse width. TC0 clock source is Fcpu = 16MHz/16 = 1MHz. Select TC0RATE=001 (Fcpu/128).**
  TC0 pulse width = 10ms. TC0 clock rate = 16MHz/16/128

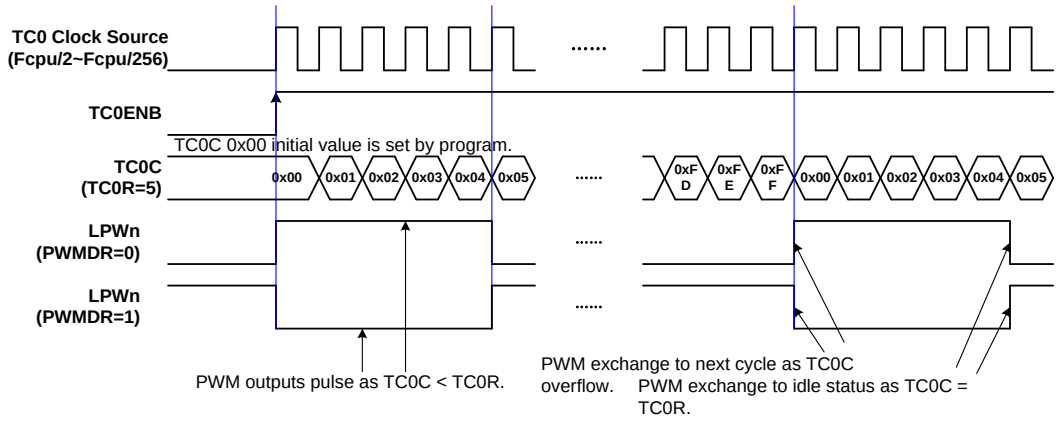
  *TC0R initial value = 256 - (TC0 interval time * input clock)*
  *= 256 - (10ms * 16MHz / 16 / 128)*
  *= 256 - (10^{-2} * 16 * 10^{6} / 16 / 128)*
  = B2H

0CDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LPWS	LPWS7	LPWS6	LPWS5	LPWS4	LPWS3	LPWS2	LPWS1	LPWS0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **LPWS [7:0]:** LED PWM channel selection.
LPWS0: LED PWM channel 1. 0=Disable, LED pin is GPIO mode. 1=Enable, LED pin output PWM signal.
LPWS1: LED PWM channel 2. 0=Disable, LED pin is GPIO mode. 1=Enable, LED pin output PWM signal.
LPWS2: LED PWM channel 3. 0=Disable, LED pin is GPIO mode. 1=Enable, LED pin output PWM signal.
LPWS3: LED PWM channel 4. 0=Disable, LED pin is GPIO mode. 1=Enable, LED pin output PWM signal.
LPWS4: LED PWM channel 5. 0=Disable, LED pin is GPIO mode. 1=Enable, LED pin output PWM signal.
LPWS5: LED PWM channel 6. 0=Disable, LED pin is GPIO mode. 1=Enable, LED pin output PWM signal.
LPWS6: LED PWM channel 7. 0=Disable, LED pin is GPIO mode. 1=Enable, LED pin output PWM signal.
LPWS7: LED PWM channel 8. 0=Disable, LED pin is GPIO mode. 1=Enable, LED pin output PWM signal.

8.3.3 TC0 PWM OPERATION

TC0 PWM cycle is TC0C 0x00~0xFF period controlled by TC0Rate[2:0] bits. It is not to set TC0C initial value, and only to clear TC0C before setting TC0ENB. TC0 PWM has 8 channels selected by LPWS register. If the bit of LPWS register is set, the channel is switched to LED PWM output mode and isolates GPIO function. If the bit is cleared, the channel returns to GPIO mode and GPIO last status. It is easy to set PWM idle status as PWM non-output mode.



8.3.4 PWM OPERATION EXAMPLE

- **PWM CONFIGURATION:**

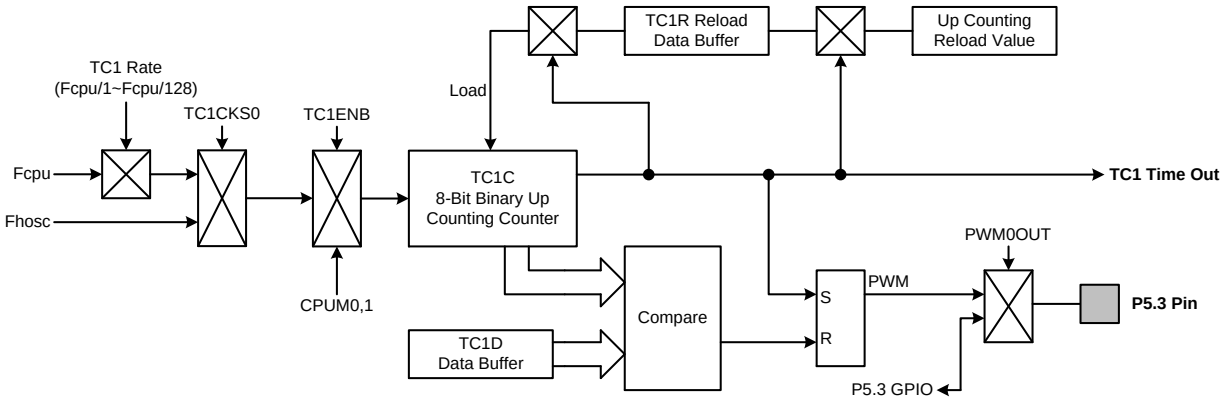
```
; Reset TC0 timer.
        CLR     TC0M            ; Clear TC0M register.

; Set TC0 clock rate.
        MOV     A, #0nnn0000b   ; Set TC0RATE[2:0].
        B0MOV   TC0M, A

        B0BCLR  FPWMDR          ; High pulse and low idle status.
; or
        B0BSET  FPWMDR          ; Low pulse and high idle status.

; Set TC0Rn register for PWM duty.
        CLR     TC0C
        MOV     A, #value
        B0MOV   TC0Rn, A        ; Set PWM duty, n=0~7.
; Enable PWM and TC0 timer.
        B0BSET  FLPWSn          ; Enable PWM channel n, n=0~7.
        B0BSET  FTC0ENB         ; Enable TC0 timer.
```

8.4 TC1 8-BIT TIMER

8.4.1 OVERVIEW

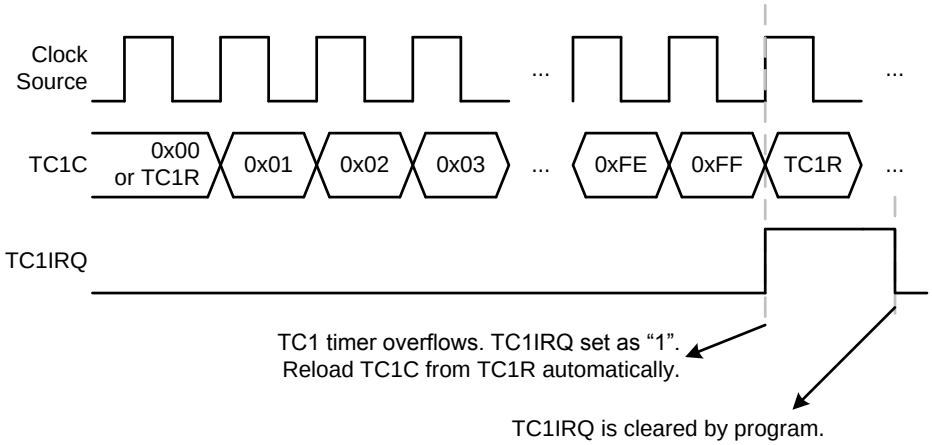
The TC1 timer is an 8-bit binary up timer with basic timer and PWM functions. The basic timer function supports flag indicator (TC1IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TC1M, TC1C, TC1R registers. TC1 builds in duty/cycle programmable PWM. The PWM cycle and resolution are controlled by TC1 timer clock rate, TC1R and TC1D registers, so the PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster…The main purposes of the TC1 timer are as following.

- **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** TC1 timer function supports interrupt function. When TC1 timer occurs overflow, the TC1IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by TC1R and TC1D registers.
- **Green mode function:** All TC1 functions (timer, PWM, auto-reload) keep running in green mode and no wake-up function.



8.4.2 TC1 TIMER OPERATION

TC1 timer is controlled by TC1ENB bit. When TC1ENB=0, TC1 timer stops. When TC1ENB=1, TC1 timer starts to count. Before enabling TC1 timer, setup TC1 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…TC1C increases "1" by timer clock source. When TC1 overflow event occurs, TC1IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is TC1C count from full scale (0xFF) to zero scale (0x00). In difference function modes, TC1C value relates to operation. If TC1C value changing effects operation, the transition of operations would make timer function error. So TC1 builds in double buffer to avoid these situations happen. The double buffer concept is to flash TC1C during TC1 counting, to set the new value to TC1R (reload buffer), and the new value will be loaded from TC1R to TC1C after TC1 overflow occurrence automatically. In the next cycle, the TC1 timer runs under new conditions, and no any transitions occur. The auto-reload function is no any control interface and always actives as TC1 enables. If TC1 timer interrupt function is enabled (TC1IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 8) and executes interrupt service routine after TC1 overflow occurrence. Clear TC1IRQ by program is necessary in interrupt procedure. TC1 timer can works in normal mode, slow mode and green mode. But in green mode, TC1 keep counting, set TC1IRQ and outputs PWM, but can't wake-up system.



TC1 provides different clock sources to implement different applications and configurations. TC1 clock source includes Fcpu (instruction cycle) and Fhosc (high speed oscillator) controlled by TC1CKS0 bits. TC1CKS0 bit selects the clock source is from Fcpu or Fhosc. If TC1CKS0=0, TC1 clock source is Fcpu through TC1rate[2:0] pre-scaler to decide Fcpu/1~Fcpu/128. If TC1CKS0=1, TC1 clock source is Fhosc without any divider. TC1rate[2:0] pre-scaler is unless when TC1CKS0=1 condition. TC1 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

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

8.4.3 TC1M MODE REGISTER

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

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1M	TC1ENB	TC1rate2	TC1rate1	TC1rate0	-	TC1CKS0	-	PWM1OUT
Read/Write	R/W	R/W	R/W	R/W	-	R/W	-	R/W
After reset	0	0	0	0	-	0	-	0

Bit 0 **PWM1OUT:** PWM output control bit.
0 = Disable PWM output function, and P5.3 is GPIO mode.
1 = Enable PWM output function, and P5.3 outputs PWM signal.

Bit 2 **TC1CKS0:** TC1 clock source select bit.
0 = Fcpu.
1 = Fhosc. **TC1rate[2:0] bits are useless.**

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

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

8.4.4 TC1C COUNTING REGISTER

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

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

The equation of TC1C initial value is as following.

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

8.4.5 TC1R AUTO-RELOAD REGISTER

TC1 timer builds in auto-reload function, and TC1R register stores reload data. When TC1C overflow occurs, TC1C register is loaded data from TC1R register automatically. Under TC1 timer counting status, to modify TC1 interval time is to modify TC1R register, not TC1C register. New TC1C data of TC1 interval time will be updated after TC1 timer overflow occurrence, TC1R loads new value to TC1C register. But at the first time to setup T0M, TC1C and TC1R must be set the same value before enabling TC1 timer. TC1 is double buffer design. If new TC1R value is set by program, the new value is stored in 1st buffer. Until TC1 overflow occurs, the new value moves to real TC1R buffer. This way can avoid any transitional condition to effect the correctness of TC1 interval time and PWM output signal.

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

The equation of TC1R initial value is as following.

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

- **Example: To calculation TC1C and TC1R value to obtain 10ms TC1 interval time. TC1 clock source is Fcpu = 16MHz/16 = 1MHz. Select TC1RATE=000 (Fcpu/128).**
  TC1 interval time = 10ms. TC1 clock rate = 16MHz/16/128

  *TC1C/TC1R initial value = 256 - (TC1 interval time * input clock)*
  *= 256 - (10ms * 16MHz / 16 / 128)*
  *= 256 - (10^{-2} * 16 * 10^{6} / 16 / 128)*
  = B2H

8.4.6 TC1D PWM DUTY REGISTER

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

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

The equation of TC1D initial value is as following.

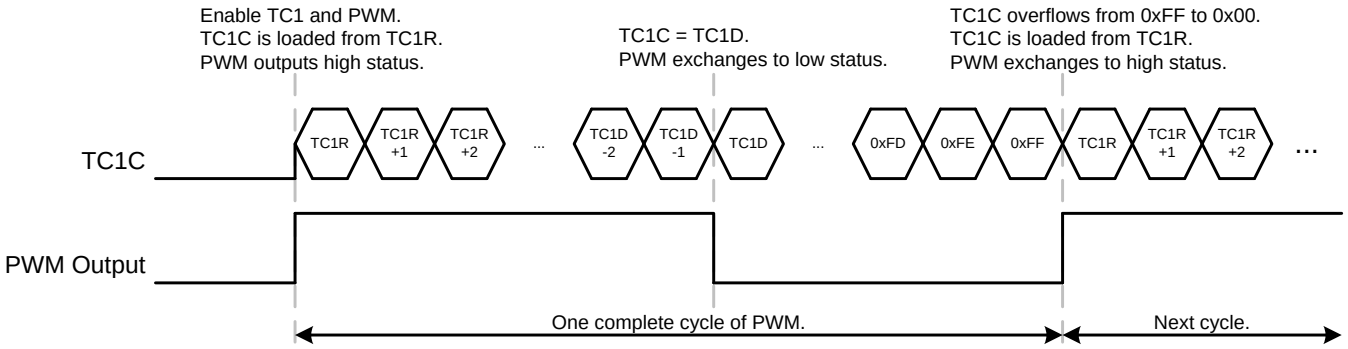
TC1D initial value = TC1R + (PWM high pulse width period / TC1 clock rate)

- **Example: To calculate TC1D value to obtain 1/3 duty PWM signal. The TC1 clock source is Fcpu = 16MHz/16 = 1MHz. Select TC1RATE=000 (Fcpu/128).**
  TC1R = B2H. TC1 interval time = 10ms. So the PWM cycle is 100Hz. In 1/3 duty condition, the high pulse width is about 3.33ms.

  TC1D initial value = B2H + (PWM high pulse width period / TC1 clock rate)
  *= B2H + (3.33ms * 16MHz / 16 / 128)*
  = B2H + 1AH
  = CCH

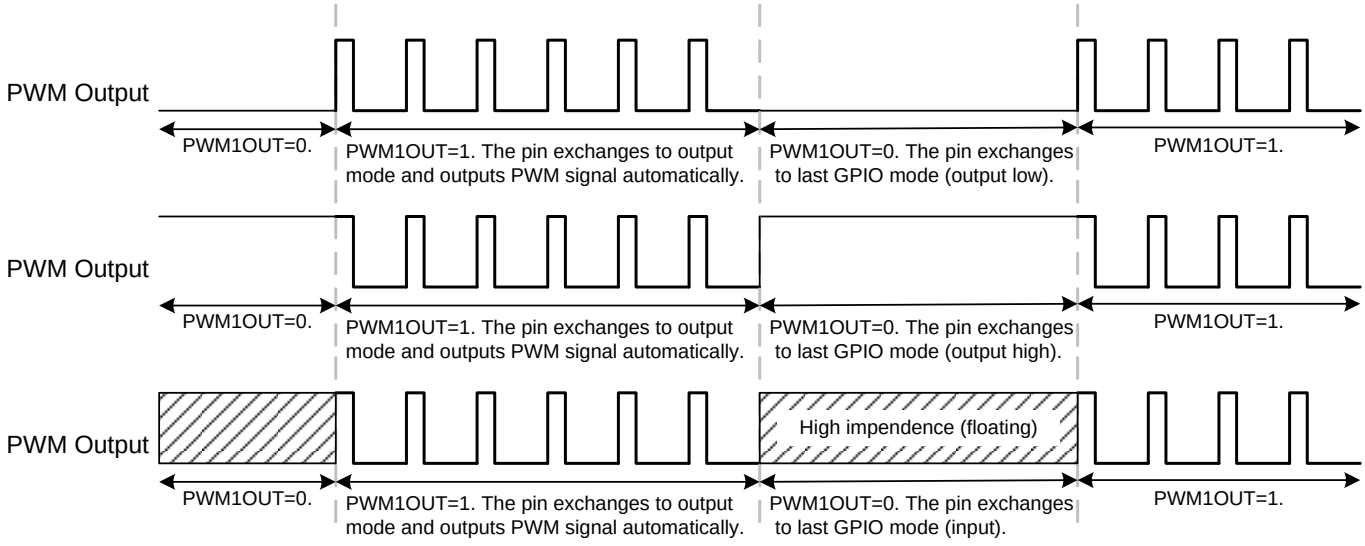
8.4.7 PULSE WIDTH MODULATION (PWM)

The PWM is duty/cycle programmable design to offer various PWM signals. When TC1 timer enables and PWM1OUT bit sets as “1” (enable PWM output), the PWM output pin (P5.3) outputs PWM signal. One cycle of PWM signal is high pulse first, and then low pulse outputs. TC1R register controls the cycle of PWM, and TC1D decides the duty (high pulse width length) of PWM. TC1C initial value is TC1R reloaded when TC1 timer enables and TC1 timer overflows. When TC1C count is equal to TC1D, the PWM high pulse finishes and exchanges to low level. When TC1 overflows (TC1C counts from 0xFF to 0x00), one complete PWM cycle finishes. The PWM exchanges to high level for next cycle. The PWM is auto-reload design to load TC1C from TC1R automatically when TC1 overflows and the end of PWM’s cycle, to keeps PWM continuity. If modify the PWM cycle by program as PWM outputting, the new cycle occurs at next cycle when TC1C loaded from TC1R.



The resolution of PWM is decided by TC1R. TC1R range is from 0x00~0xFF. If TC1R = 0x00, PWM’s resolution is 1/256. If TC1R = 0x80, PWM’s resolution is 1/128. TC1D controls the high pulse width of PWM for PWM’s duty. When TC1C = TC1D, PWM output exchanges to low status. TC1D must be greater than TC1R, or the PWM signal keeps low status. When PWM outputs, TC1IRQ still actives as TC1 overflows, and TC1 interrupt function actives as TC1IEN = 1. But strongly recommend be careful to use PWM and TC1 timer together, and make sure both functions work well.

The PWM output pin is shared with GPIO and switch to output PWM signal as PWM1OUT=1 automatically. If PWM1OUT bit is cleared to disable PWM, the output pin exchanges to last GPIO mode automatically. It easily to implement carry signal on/off operation, not to control TC1ENB bit.



8.4.8 TC1 TIMER OPERATION EXPLAME

- **TC1 TIMER CONFIGURATION:**

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

; Set TC1 clock source and TC1 rate.
        MOV     A, #0nnn0n00b
        B0MOV   TC1M, A

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

; Clear TC1IRQ
        B0BCLR  FTC1IRQ

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

- **TC1 PWM CONFIGURATION:**

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

; Set TC1 clock source and TC1 rate.
        MOV     A, #0nnn0n00b
        B0MOV   TC1M, A

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

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

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

9 SERIAL INPUT/OUTPUT TRANSCEIVER (SIO)

9.1 OVERVIEW

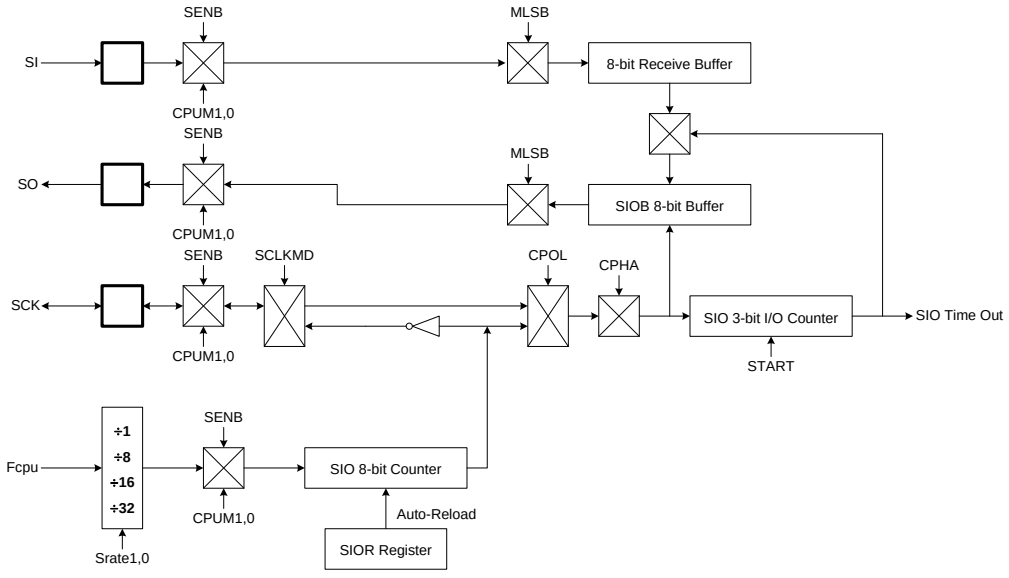
The SIO (serial input/output) transceiver is a serial communicate interface for data exchanging from one MCU to one MCU or other hardware peripherals. It is a simple 8-bit interface without a major definition of protocol, packet or control bits. The SIO transceiver includes three pins, clock (SCK), data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

The SIO features include the following:

- **Full-duplex, 3-wire synchronous data transfer.**
- **Master (SCK is clock output) or Slave (SCK is clock input) operation.**
- **MSB/LSB first data transfer.**
- **The start phase of data sampling location selection is 1st-phase or 2nd-phase controlled register.**
- **SCK, SI, SO are programmable open-drain output pin for multiple salve devices application.**
- **Two programmable bit rates (Only in master mode).**
- **End-of-Transfer interrupt.**

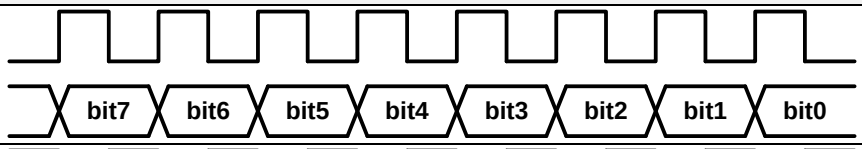
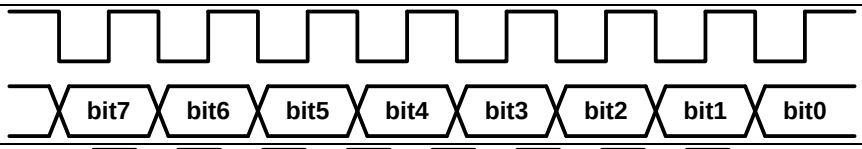
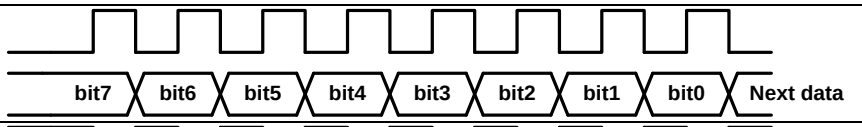
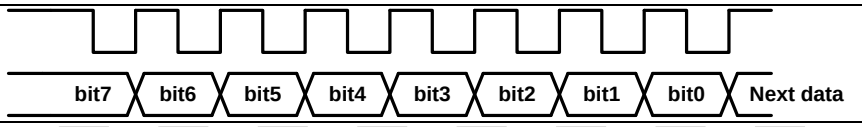
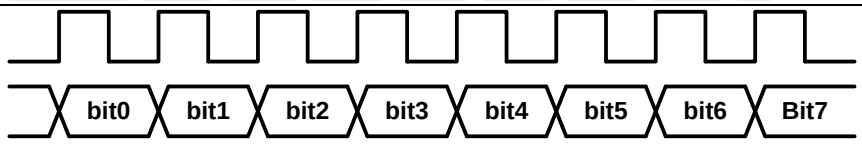
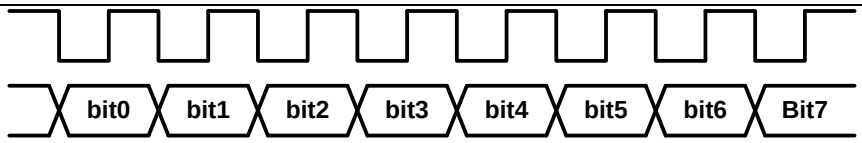
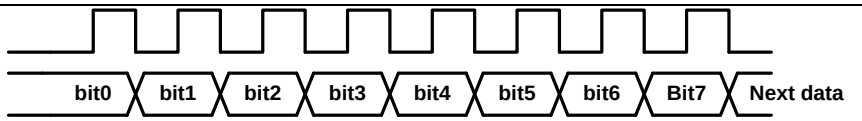
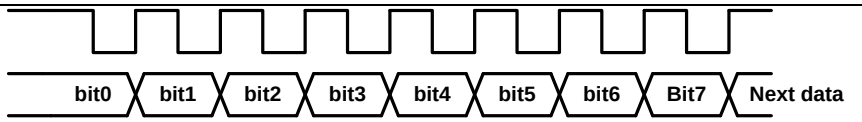
9.2 SIO OPERATION

The SIOM register can control SIO operating function, such as: transmit/receive, clock rate, data transfer direction, SIO clock idle status and clock control phase and starting this circuit. This SIO circuit will transmit or receive 8-bit data automatically by setting SENB and START bits in SIOM register. The SIO data buffer is double buffer design. When the SIO operating, the SIOB register stores transfer data and one internal buffer stores receive data. When SIO operation is successfully, the internal buffer reloads into SIOB register automatically. The SIO 8-bit counter and SIOR register are designed to generate SIO's clock source with auto-reload function. The 3-bit I/O counter can monitor the operation of SIO and announce an interrupt request after transmitting/ receiving 8-bit data. After transferring 8-bit data, this circuit will be disabled automatically and re-transfer data by programming SIOM register. CPOL bit is designed to control SIO clock idle status. CPHA bit is designed to control the clock edge direction of data receive. CPOL and CPHA bits decide the SIO format. The SIO data transfer direction is controlled by MLSB bit to decide MSB first or LSB first.



SIO Interface Circuit Diagram

The SIO supports 8-mode format controlled by MLSB, CPOL and CPHA bits. The edge direction is “Data Transfer Edge”. When setting rising edge, that means to receive and transmit one bit data at SCK rising edge, and data transition is at SCK falling edge. When setting falling edge, that means to receive and transmit one bit data at SCK falling edge, and data transition is at SCK rising edge.
“CPHA” is the clock phase bit controls the phase of the clock on which data is sampled. When CPHA=1, the SCK first edge is for data transition, and receive and transmit data is at SCK 2nd edge. When CPHA=0, the 1st bit is fixed already, and the SCK first edge is to receive and transmit data. The SIO data transfer timing as following figure:

MLSB	CPOL	CPHA	Diagrams	Description
0	0	1		SCK idle status = Low. The transfer first bit = MSB. SCK data transfer edge = Falling edge.
0	1	1		SCK idle status = High. The transfer first bit = MSB. SCK data transfer edge = Rising edge.
0	0	0		SCK idle status = Low. The transfer first bit = MSB. SCK data transfer edge = Rising edge.
0	1	0		SCK idle status = High. The transfer first bit = MSB. SCK data transfer edge = Falling edge.
1	0	1		SCK idle status = Low. The transfer first bit = LSB. SCK data transfer edge = Falling edge.
1	1	1		SCK idle status = High. The transfer first bit = LSB. SCK data transfer edge = Rising edge.
1	0	0		SCK idle status = Low. The transfer first bit = LSB. SCK data transfer edge = Rising edge.
1	1	0		SCK idle status = High. The transfer first bit = LSB. SCK data transfer edge = Falling edge.

SIO Data Transfer Timing

The SIO supports interrupt function. SIOIEN is SIO interrupt function control bit. SIOIEN=0, disable SIO interrupt function. SIOIEN=1, enable SIO interrupt function. When SIO interrupt function enable, the program counter points to interrupt vector (ORG 8) to do SIO interrupt service routine after SIO operating. SIOIRQ is SIO interrupt request flag, and also to be the SIO operating status indicator when SIOIEN = 0, but cleared by program. When SIO operation finished, the SIOIRQ would be set to "1", and the operation is the inverse status of SIO "START" control bit.
The SIOIRQ and SIO START bit indicating the end status of SIO operation is after one 8-bit data transferring. The duration from SIO transfer end to SIOIRQ/START active is about **"1/2*SIO clock"**, means the SIO end indicator doesn't active immediately.

✷ ***Note: The first step of SIO operation is to setup the SIO pins' mode. Enable SENB, select CPOL and CPHA bits. These bits control SIO pins' mode.***

9.3 SIOM MODE REGISTER

0B4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOM	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **SENB:** SIO function control bit.
0 = Disable SIO function. P5.0~P5.2 are GPIO.
1 = Enable SIO function. P5.0~P5.2 are SIO pins. **SIO pin structure can be push-pull structure and open-drain structure controlled by P1OC register.**

Bit 6 **START:** SIO progress control bit.
0 = End of transfer.
1 = SIO transmitting.

Bit [5:4] **SRATE1,0:** SIO's transfer rate select bit. **These 2-bits are workless when SCKMD=1.**
00 = fcpu.
01 = fcpu/32
10 = fcpu/16
11 = fcpu/8.

Bit 3 **MLSB:** MSB/LSB transfer first.
0 = MSB transmit first.
1 = LSB transmit first.

Bit 2 **SCKMD:** SIO's clock mode select bit.
0 = Internal. (Master mode)
1 = External. (Slave mode)

Bit 1 **CPOL:** SCK idle status control bit.
0 = SCK idle status is low status.
1 = SCK idle status is high status.

Bit 0 **CPHA:** The Clock Phase bit controls the phase of the clock on which data is sampled.
0 = Data receive at the first clock phase.
1 = Data receive at the second clock phase.

Because SIO function is shared with Port5 for P5.0 as SCK, P5.1 as SI and P5.2 as SO. The following table shows the Port5[2:0] I/O mode behavior and setting when SIO function enable and disable.

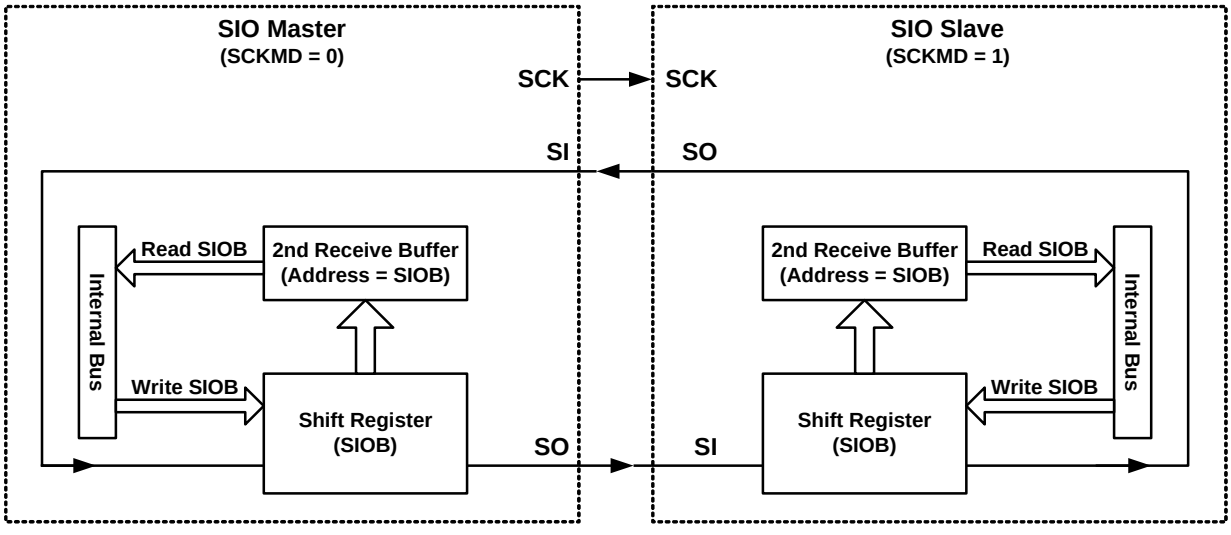
SENB=1 (SIO Function Enable)		
P5.0/SCK	(SCKMD=1) SIO source = External clock	P5.0 will change to Input mode automatically, no matter what P5M setting.
	(SCKMD=0) SIO source = Internal clock	P5.0 will change to Output mode automatically, no matter what P5M setting.
P5.1/SI	P5.1 must be set as Input mode in P5M ,or the SIO function will be abnormal	
P5.2/SO	SIO = Transmitter/Receiver	P5.2 will change to Output mode automatically, no matter what P5M setting.
SENB=0 (SIO Function Disable)		
P5.0/P5.1/P5.2	Port5[2:0] I/O mode are fully controlled by P5M when SIO function Disable	

✷ ***Note:***
1. ***If SCKMD=1 for external clock, the SIO is in SLAVE mode. If SCKMD=0 for internal clock, the SIO is in MASTER mode.***
2. ***Don't set SENB and START bits in the same time. That makes the SIO function error.***
3. ***SIO pin can be push-pull structure and open-drain structure controlled by P1OC register.***

9.4 SIOB DATA BUFFER

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOB	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

SIOB is the SIO data buffer register. It stores serial I/O transmit and receive data. The system is single-buffered in the transmit direction and double-buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SIOB Data Register before the entire shift cycle is completed. When receiving data, however, a received byte must be read from the SIOB Data Register before the next byte has been completely shifted in. Otherwise, the first byte is lost. Following figure shows a typical SIO transfer between two micro-controllers. Master MCU sends SCK for initial the data transfer. Both master and slave MCU must work in the same clock edge direction, and then both controllers would send and receive data at the same time.



SIO Data Transfer Diagram

9.5 SIOR REGISTER DESCRIPTION

0B5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOR	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The SIOR is designed for the SIO counter to reload the counted value when end of counting. It is like a post-scaler of SIO clock source and let SIO has more flexible to setting SCK range. Users can set the SIOR value to setup SIO transfer time. To setup SIOR value equation to desire transfer time is as following.

SCK frequency = (SIO rate / (256 - SIOR))/2

SIOR = 256 - (1 / (2 * SCK frequency) * SIO rate)

- **Example: Setup the SIO clock to be 5KHz. Fhosc = 16MHz. SIO's rate = Fcpu = Fhosc/16.**

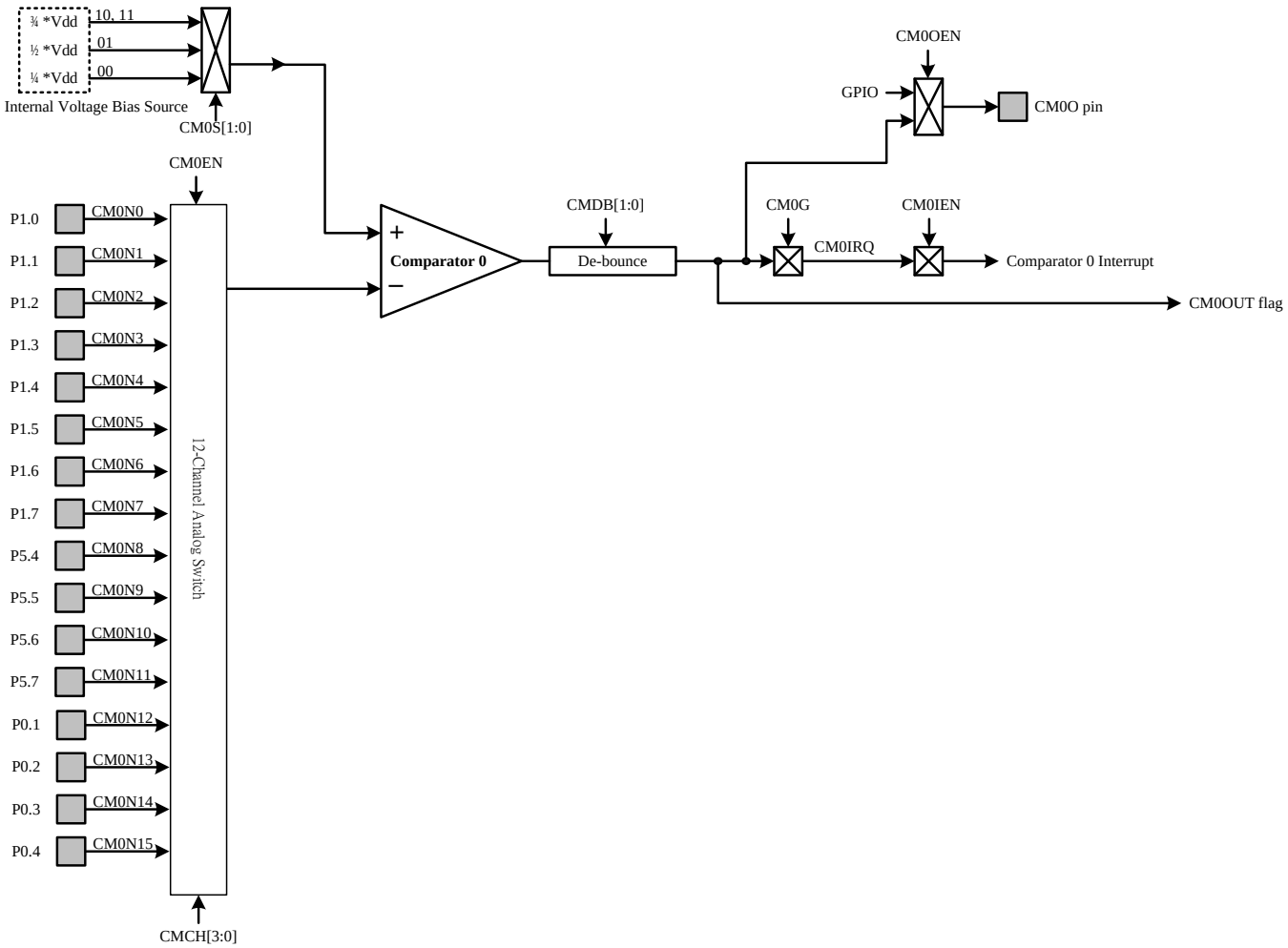
SIOR = 256 – (1/(2*5KHz) * 16MHz/16)
= 256 – (0.0001*1000000)
= 256 – 100
= 156

10 16-CHANNEL ANALOG COMPARAOTR

10.1 OVERVIEW

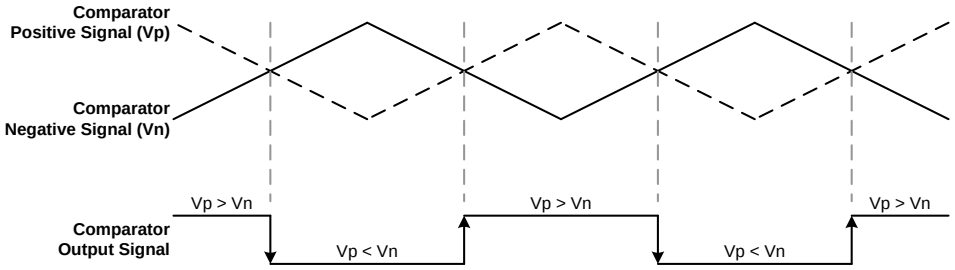
The analog comparator compares negative input voltage, and then output the result to comparator output terminal. The comparator has multi-input selection for different applications. The comparator negative input terminal is up to 16-channel controlled by CMCH[3:0]. The comparator positive input terminal has three selections controlled by CM0S[1:0] bits. The comparator output terminal connects to external pin CM0O and connects to internal path. There is a programmable direction function to decide comparator trigger edge for indicator function. The comparator has flag indicator, interrupt function and green mode weak-up function for different application.

- 16-channel negative input selection.
- Comparator output function.
- Programmable internal reference voltage connected to comparator's positive terminal.
- Programmable trigger direction.
- Interrupt function.
- Green mode wake-up function.



10.2 COMPARATOR OPERATION

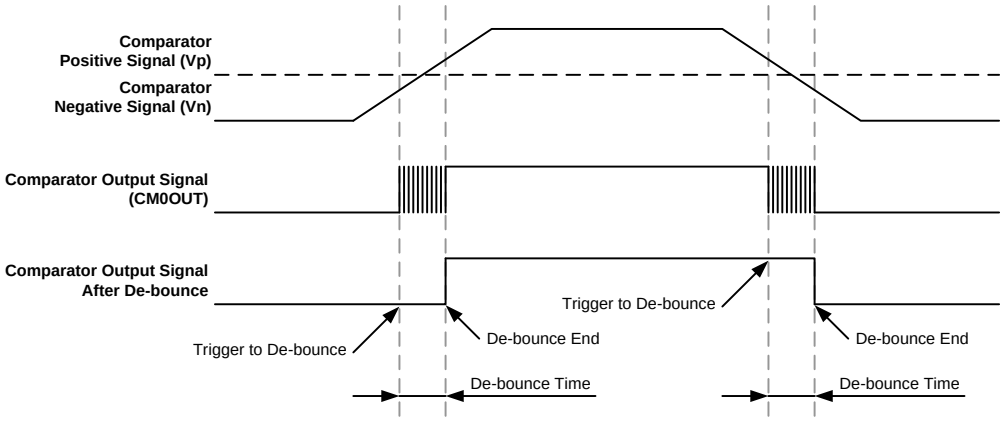
The comparator operation is to compare the voltage between comparator positive input and negative input terminals. When the positive input voltage is greater than the negative input voltage, the comparator output is high status. When the positive input voltage is smaller than the negative input voltage, the comparator output is low status.



The comparator builds in interrupt function. The interrupt function trigger edge is selected by CM0G. The trigger edge supports rising edge (CM0G=0), falling edge (CM0G=1). If the trigger edge condition is found, the CM0IRQ is set as “1”. If the comparator interrupt function enables, the system will execute interrupt routine. The CM0IRQ must be cleared by program.

The comparator builds in green mode wake-up function. The comparator green mode wake-up trigger edge is bi-direction. The comparator’s wake-up function only supports green mode, not power down mode. If the trigger edge condition (comparator output status exchanging) is found, the system will be wake-up from green mode. If the trigger edge direction is interrupt trigger condition, the CM0IRQ is set as “1”. Of course the interrupt routine is executed if the interrupt function enabled. When the wake-up trigger edge direction is equal to interrupt trigger condition, the system will execute interrupt operation after green mode wake-up immediately.

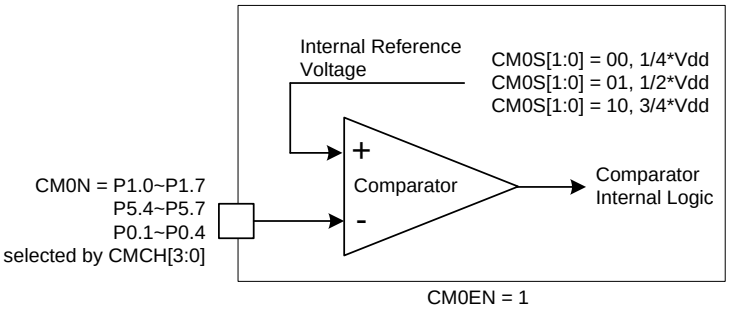
The critical condition is comparator positive voltage equal to comparator negative voltage, and the voltage range is decided comparator offset parameter of input common mode. In the voltage range, the comparator output signal is unstable and keeps oscillating until the differential voltage exits the range. In the condition, the comparator flag (CM0IRQ) latches the first exchanging and issue the status, but the status is a transient, not a stable condition. So the comparator builds in a filter to de-bounce the transient condition. The comparator output signal is through a de-bounce circuit to filter comparator transient status. The de-bounce time is controlled by CMDB[1:0] bits that means the comparator minimum response time is 1*Fcpu, 2*Fcpu, 3*Fcpu or no de-bounce. The de-bounce time depends on the signal slew rate and selected by program.



The comparator positive input terminal includes internal reference voltage source. The internal reference voltage source supports three levels which are 1/4*Vdd (CM0S[1:0] = 00), 1/2*Vdd (CM0S[1:0] = 01) and 3/4*Vdd (CM0S[1:0] = 10,11).

The comparator negative input terminal supports maximum 16-channel controlled by CMCH [3:0]. 0000=P1.0. 0001=P1.1. 0010=P1.2. 0011=P1.3. 0100=P1.4. 0101=P1.5. 0110=P1.6. 0111=P1.7. 1000=P5.4. 1001=P5.5. 1010=P5.6. 1011=P5.7. 1100=P0.1. 1101=P0.2. 1110=P0.3. 1111=P0.4. These channels selected is only when the comparator enables (CM0EN=1), the 12-channel analog switch works, or not workable. If one pin is selected to be comparator negative input pin, the pin is switched to input mode and connected to comparator negative input terminal. When the system selects to other pin or comparator disables, the original channel will returns to last GPIO mode automatically.

The comparator output status can output to CM0O pin controlled by CM0OEN bit. When CM0OEN=0, the comparator output pin is GPIO mode. If CM0OEN=1, CM0O pin outputs comparator output status and isolates GPIO mode. The comparator output terminal connects to internal path. The CM0OUT flag is the CM0OUT shows the comparator result immediately, but the CM0IRQ only indicates the event of the comparator result. The comparator output terminal through de-bounce circuit generates the comparator trigger edge controlled by CM0G. The even condition is controlled by register and includes rising edge (CM0OUT changes from low to high), falling edge (CM0OUT changes from high to low) controlled by CM0G bit. The CM0IRQ = 1 condition makes the comparator interrupt service executed when CM0IEN (comparator interrupt control bit) set.



10.3 COMPARATOR CONTROL REGISTER

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M	CM0EN	CM0OUT	CM0S1	CM0S0	CMCH3	CMCH2	CMCH1	CMCH0
Read/Write	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	1	0	0	0	0	0	0

Bit 7 **CM0EN:** Comparator 0 control bit.
0 = Disable. P1[7:0], P5[7:4] and P0[4:1]are GPIO mode.
1 = Enable. Comparator negative input pins are controlled by CMCH[3:0] bits.

Bit 6 **CM0OUT:** Comparator 0 output flag bit. **The comparator output status is "1" as comparator disabled.**
0 = Comparator internal reference voltage is less than CM0N voltage.
1 = Comparator internal reference voltage is larger than CM0N voltage.

Bit [5:4] **CM0S[1:0]:** Comparator 0 positive input voltage control bit.
00 = Internal 1/4*Vdd and enable internal reference voltage generator.
01 = Internal 1/2*Vdd and enable internal reference voltage generator.
10, 11 = Internal 3/4*Vdd and enable internal reference voltage generator.

Bit [3:0] **CMCH[3:0]:** Comparator 0 negative input pin control bit.
0000 = Comparator negative input pin is P1.0.
0001 = Comparator negative input pin is P1.1.
0010 = Comparator negative input pin is P1.2.
0011 = Comparator negative input pin is P1.3.
0100 = Comparator negative input pin is P1.4.
0101 = Comparator negative input pin is P1.5.
0110 = Comparator negative input pin is P1.6.
0111 = Comparator negative input pin is P1.7.
1000 = Comparator negative input pin is P5.4.
1001 = Comparator negative input pin is P5.5.
1010 = Comparator negative input pin is P5.6.
1011 = Comparator negative input pin is P5.7.
1100 = Comparator negative input pin is P0.1.
1101 = Comparator negative input pin is P0.2.
1110 = Comparator negative input pin is P0.3.
1111 = Comparator negative input pin is P0.4.

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M1	-	-	-	-	CMDB1	CMDB0	CM0OEN	CM0G
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After Reset	-	-	-	-	0	0	0	0

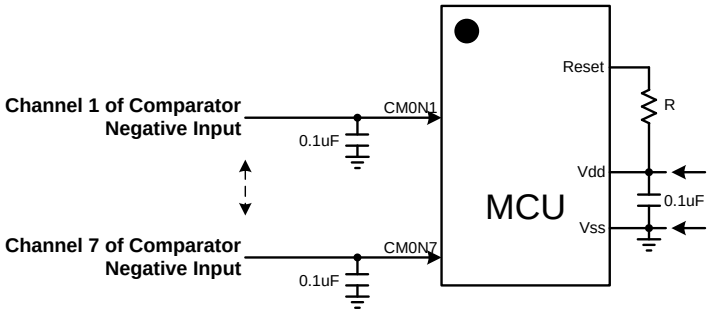
Bit [3:2] **CMDB[1:0]:** Comparator output denounce time select bit.
00 = 1*Fcpu.
01 = 2*Fcpu.
10 = 3*Fcpu.
11 = No de-bounce.

Bit 1 **CM0OEN:** Comparator output pin control bit.
0 = Disable. CM0O pin is GPIO mode.
1 = Enable comparator output pin. P0.0 pin exchanges to comparator output pin (CM0O pin), and GPIO function is isolated.

Bit 0 **CM0G:** Comparator interrupt trigger direction control bit.
0 = Rising edge trigger. comparator internal reference voltage > CM0N.
1 = Falling edge trigger. comparator internal reference voltage < CM0N.

10.4 COMPARATOR APPLICATION NOTICE

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



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

```
; The comparator 0 initialize.
        MOV     A, #00010000b   ; CM0S[1:0]=01b, enable comparator internal 1/2*Vdd
        B0MOV   CM0M, A         ; reference voltage to be comparator positive source.
                                ; CMCH[3:0]=0000b, select comparator negative input pin
                                ; is CM0N0.
        MOV     A, #00000000b   ; CM0G=0b, set comparator interrupt request as
        B0MOV   CM0M1, A        ; rising edge.
                                ; CMDB[1:0]=00b, no de-bounce.
        B0BSET  FCM0IEN         ; CM0IEN=1, enable comparator 0 interrupt function.
        B0BCLR  FCM0IRQ         ; CM0IRQ=0, clear comparator 0 interrupt request flag.

        B0BSET  FCM0EN          ; Enable comparator 0.

Main:                           ; Main loop.
        …
        JMP     MAIN
        …

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

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

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

11 MAIN SERIAL PORT (MSP)

11.1 OVERVIEW

The MSP (Main Serial Port) is a serial communication interface for data exchanging from one MCU to one MCU or other hardware peripherals. These peripheral devices may be serial EEPROM, A/D converters, Display device, etc. The MSP module only can operate in slave mode:

- **Slave mode (with general address call).**

The MSP features include the following:

- **2-wire synchronous data transfer/receiver.**
- **Slave (SCL is clock input) operation.**
- **SCL, SDA are programmable open-drain output pin for multiplex salve devices application.**
- **Support 400K clock rate @ Fcpu=4MIPs.**
- **End-of-Transfer/Receiver interrupt.**

11.2 MSP STATUS REGISTER

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPSTAT	-	CKE	D_A	P	S	RED_WRT	-	BF
Read/Write	-	R/W	R	R	R	R	-	R
After reset	-	0	0	0	0	0	-	0

Bit 6 **CKE:** Slave Clock Edge Control bit
In Slave Mode: Receive Address or Data byte
0= Latch Data on SCL Rising Edge. **(Default)**
1= Latch Data on SCL Falling Edge.

✸ ***Note:***
1. ***In Slave Transmit mode, Address Received depended on CKE setting. Data Transfer on SCL Falling Edge.***
2. ***In Slave Receiver mode, Address and Data Received depended on CKE setting.***

Bit 5 **D_A_:** Data/Address_ bit
0=Indicates the last byte received or transmitted was address.
1= Indicates the last byte received or transmitted was data.

Bit 4 **P:** Stop bit
0 = Stop bit was not detected.
1 = Indicates that a stop bit has been detected last.

✸ ***Note: It will be cleared when STOP bit was detected.***

Bit 3 **S:** Start bit.
0 = Start bit was not detected.
1 = Indicates that a start bit has been detected last

✸ ***Note: It will be cleared when Start bit was detected.***

Bit 2 **RED_WRT:** Read/Write bit information.
This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next start bit, stop bit, or not ACK bit.
In slave mode:
0 = Write.
1 = Read.

Bit 0 **BF:** Buffer Full Status bit
Receive
1 = Receive complete, MSPBUF is full.
0 = Receive not complete, MSPBUF is empty.
Transmit
1 = Data Transmit in progress (does not include the ACK and stop bits), MSPBUF is full.
0 = Data Transmit complete (does not include the ACK and stop bits), MSPBUF is empty.

11.3 MSP MODE REGISTER1

091H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPM1	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK	GCEN	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
After reset	0	0	0	0	0	0	0	-

Bit 7 **WCOL:** Write Collision Detect bit
Slave Mode:
0 = No collision
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)

Bit 6 **MSPPOV:** Receive Overflow Indicator bit
0 = No overflow.
1 = A byte is received while the SSPBUF register is still holding the previous byte. SSPOV is a “don’t care” in transmit mode. SSPOV must be cleared in software in either mode. (must be cleared in software)

Bit 5 **MSPENB:** MSP Communication Enable.
0 = Disables serial port and configures these pins as I/O port pins
1 = Enables serial port and configures SCL, SDA as the source of the serial port pins

✸ ***Note: MSP status register will be clear after MSP Disable. So, user should setting MSP register again before MSP Enable.***

```
Ex:  B0BCLR   FMSPENB
     CALL     MSP_init_setting
     B0BSET   FMSPENB
```

Bit 4 **CKP:** SCL Clock Priority Control bit
In MSP Slave mode
0 = Hold SCL keeping Low. (Ensure data setup time and Slave device ready.)
1 = Release SCL Clock
(Slave Transistor mode CKP function always enables, Slave Receiver CPK function control by SLRXCKP)

Bit 3 **SLRXCKP:** Slave Receiver mode SCL Clock Priority Control bit
In MSP Slave Receiver mode.
0 = Disable CKP function.
1 = Enable CKP function.
In MSP Slave and Slave Transistor mode Unused.

Bit 2 **MSPWK:** MSP Wake-up indication bit
0 = MCU NOT wake-up by MSP.
1 = MCU wake-up by MSP

✸ ***Note: Clear MSPWK before entering Power down mode for indication the wake-up source from MSP or not***

Bit 1 **GCEN**: General Call Enable bit (In Slave mode only)
0 = General call address disabled
1 = Enable interrupt when a general call address (0000h) is received.

11.4 MSP MSPBUF REGISTER

MSPBUF initial value = 0000 0000

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPBUF	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0
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

11.5 MSP MSPADR REGISTER

MSPADR initial value = 0000 0000

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPADR	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0
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

11.6 SLAVE MODE OPERATION

When an address is matched or data transfer after and address match is received, the hardware automatically will generate the acknowledge (ACK_) signal, and load MSPBUF (MSP buffer register) with the received data from MSPSR.

There are some conditions that will cause MSP function will not reply ACK_ signal:

- Data Buffer already full: BF=1 (MSPSTAT bit 0), when another transfer was received.
- Data Overflow: MSPOV=1 (MSPM1 bit 6), when another transfer was received

When BF=1, means MSPBUF data is still not read by MCU, so MSPSR will not load data into MSPBUF, but MSPIRQ and MSPOV bit will still set to 1. BF bit will be clear automatically when reading MSPBUF register. MSPOV bit must be clear through by Software.

11.6.1 ADDRESSING

When MSP Slave function has been enabled, it will wait a START signal occur. Following the START signal, 8-bit address will shift into the MSPSR register. The data of MSPSR[7:1] is compare with MSPADDR register on the falling edge of eight SCL pulse, If the address are the same, the BF and SSPOV bit are both clear, the following event occur:

1. MSPSR register is loaded into MSPBUF on the falling edge of eight SCL pulse.
2. Buffer full bit (BF) is set to 1, on the falling edge of eight SCL pulse.
3. An ACK_ signal is generated.
4. MSP interrupt request MSPIRQ is set on the falling edge of ninth SCL pulse.

Status when Data is Received		**MSPSP→ MSPBUF**	**Reply an ACK_ signal**	**Set MSPIRQ**
BF	**MSPOV**			
0	0	Yes	Yes	Yes
*0	*1	Yes	No	Yes
1	0	No	No	Yes
1	1	No	No	Yes

Data Received Action Table

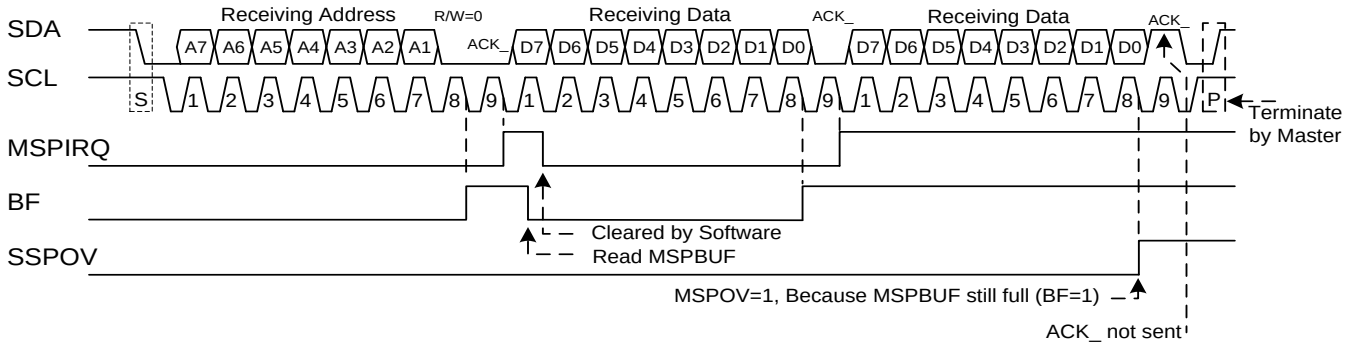
✸ ***Note: BF=0, MSPOV=1 shows the software is not set properly to clear Overflow register.***

11.6.2 SLAVE RECEIVING

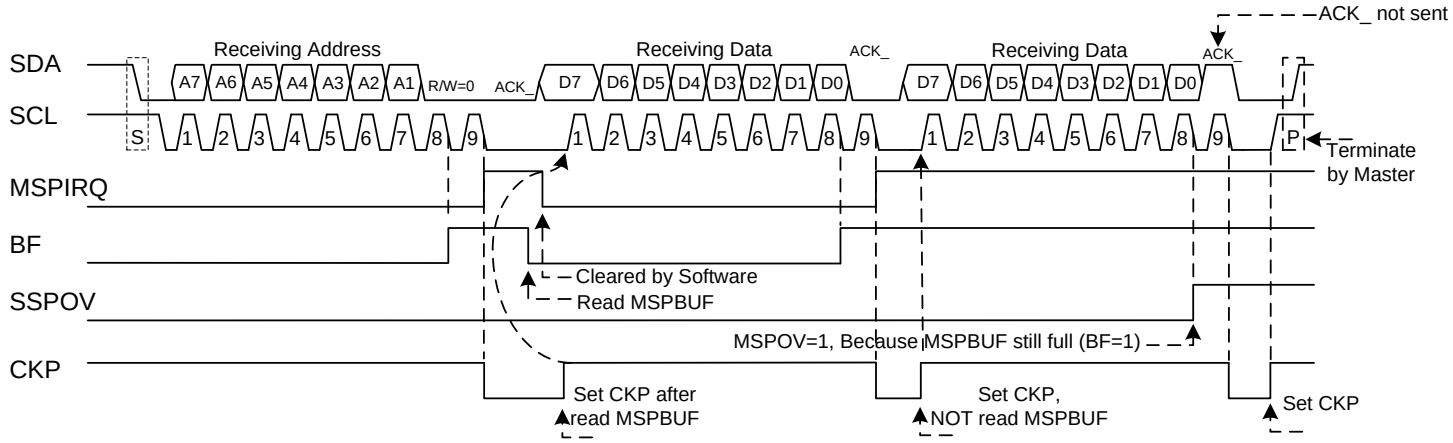
When the R/W bit of address byte =0 and address is matched, the R/W bit of MSPSTAT is cleared. The address will be load into MSPBUF. After reply an ACK_ signal, MSP will receive data every 8 clock. The CKP function enable or disable (Default) is controlled by SLRXCKP bit and data latch edge -Rising edge (Default) or Falling edge is controlled by CPE bit.
When overflow occur, no acknowledge signal replied which either BF=1 or MSPOV=1.
MSP interrupt is generated in every data transfer. The MSPIRQ bit must be clear by software.

Following is the Slave Receiving Diagram

SLRXCKP=0

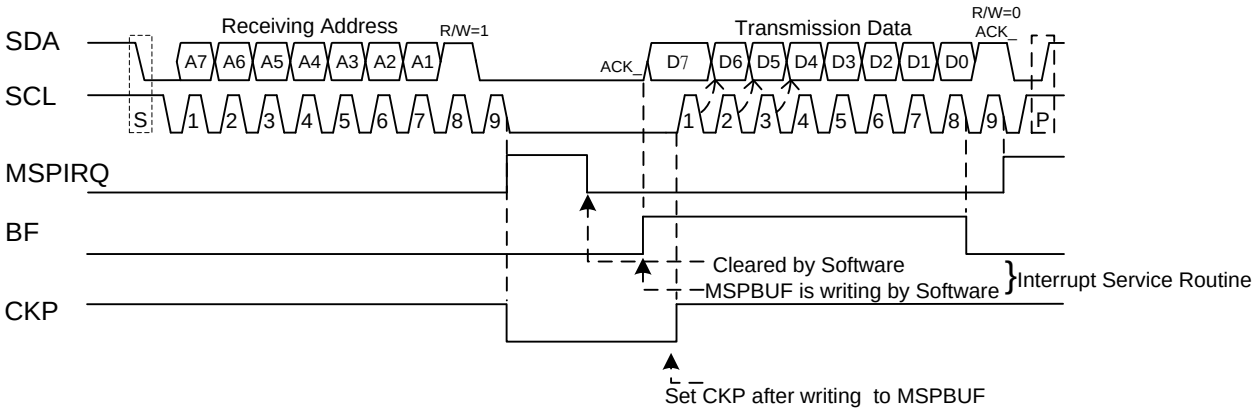


SLRXCKP=1



11.6.3 SLAVE TRANSMISSION

After address match, the following R/W bit is set, MSPSTAT bit 2 R/W will be set. The received address will be load to MSPBUF and ACK_ will be sent at ninth clock then SCL will be hold low. Transmission data will be load into MSPBUF which also load to MSPSR register. The Master should monitor SCL pin signal. The slave device may hold on the master by keep CKP low. When set. After load MSPBUF, set CKP bit, MSPBUF data will shift out on the falling edge on SCL signal. This will ensure the SDA signal is valid on the SCL high duty.
An MSP interrupt is generated on every byte transmission. The MSPIRQ will be set on the ninth clock of SCL. Clear MSPIRQ by software. MSPSTAT register can monitor the status of data transmission.
In Slave transmission mode, an ACK_ signal from master-receiver is latched on rising edge of ninth clock of SCL. If ACK_ = high, transmission is complete. Slave device will reset logic and waiting another START signal. If ACK_= low, slave must load MSPBUF which also MSPSR, and set CKP=1 to start data transmission again.



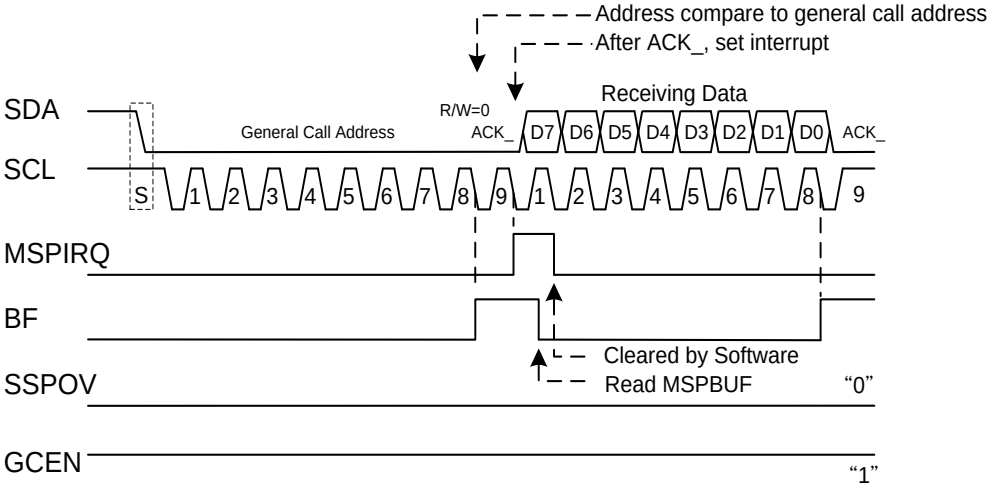
MSP Slave Transmission Timing Diagram

11.6.4 GENERAL CALL ADDRESS

In MSP bus, the first 7-byte is the Slave address. Only the address match MSPADDR the Slave will response an ACK_. The exception is the general call address which can address all slave devices. When this address occur, all devices should response an acknowledge.

The general call address is a special address which is reserved as all “0” of 7-bytes address. The general call address function is control by GCEN bit. Set this bit will enable general call address and clear it will disable. When GECN=1, following a START signal, 8-bit will shift into MSPSR and the address is compared with MSPADD and also the general call address which fixed by hardware.

If the genera call address matches, the MSPSR data is transferred into MSPBUF, the BF flag bit is set, and in the falling edge of the ninth clock (ACK_) MSPIRQ flag set for interrupt request. In the interrupt service routine, reading MSPBUF can check if the address is the general call address or device specific.

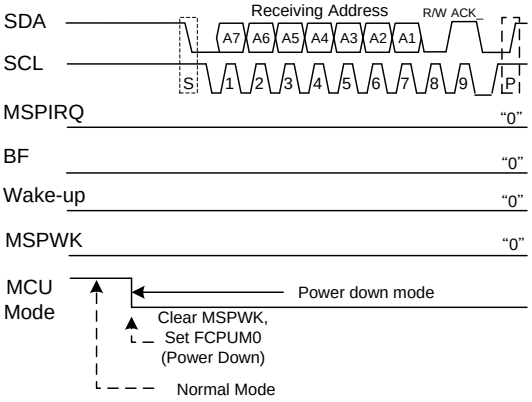


General Call Address Timing Diagram

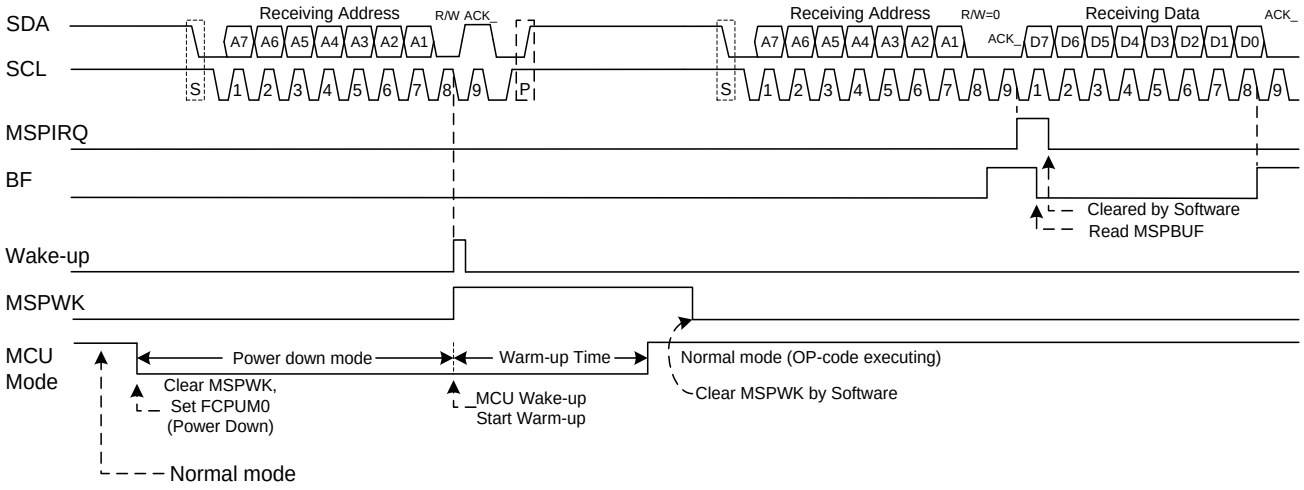
11.6.5 SLAVE WAKE UP

When MCU enter Power down mode, if MSBENB bit is still set, MCU can wake-up by matched device address. The address of MSP bus following START bit, 8-byte address will shift into MSPSR, if address matched, an NOT Acknowledge will response on the ninth clock of SCL and MCU will be wake-up, MSPWKset and start wake-up procedure but MSPIRQ will not set and MSPSR data will not load to MSPUBF. After MCU finish wake-up procedure, MSP will be in idle status and waiting master's START signal. Control register BF, MSPIRQ, MSPOV and MSPBUF will be the same status/data before power down.

If address not matches, a NOT acknowledge is still sent on the ninth clock of SCL, but MCU will be NOT wake-up and still keep in power down mode.



MSP Wake-up Timing Diagram: Address NOT Matched



MSP Wake-up Timing Diagram: Address Matched

After into power down mode, we need to disable MSP and then enable MSP to reset MSP function and re-write the I2C slave address.

➢ **Example:**

```
B0BSET      FCPUM0
B0BCLR      FMSPENB
NOP
B0BSET      FMSPENB
MOV         A, #0xnn
B0MOV       MSPADR, A       ; Re-write the I2C slave address again.
```

✸ ***Note:***

1. ***MSP function only can work on Normal mode, when wake-up from power down mode, MCU must operate in Normal mode before Master sent START signal.***
2. ***In MSP wake-up, if the address not matches, MCU will keep in power down mode.***
3. ***Clear MSPWK before enter power down mode by Software for wake-up indication.***

12 INSTRUCTION TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
	MOV	A,M	A ← M	-	-	√	1
M	MOV	M,A	M ← A	-	-	-	1
O	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
V	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
E	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I, "M" only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z…)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
A	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1+N
R	ADD	A,M	A ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
I	ADD	M,A	M ← A + M, if occur carry, then C=1, else C=0	√	√	√	1+N
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
H	ADD	A,I	A ← A + I, if occur carry, then C=1, else C=0	√	√	√	1
M	SBC	A,M	A ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
E	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
T	SUB	A,M	A ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
I	SUB	M,A	M ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1+N
C	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
	AND	A,M	A ← A and M	-	-	√	1
L	AND	M,A	M ← A and M	-	-	√	1+N
O	AND	A,I	A ← A and I	-	-	√	1
G	OR	A,M	A ← A or M	-	-	√	1
I	OR	M,A	M ← A or M	-	-	√	1+N
C	OR	A,I	A ← A or I	-	-	√	1
	XOR	A,M	A ← A xor M	-	-	√	1
	XOR	M,A	M ← A xor M	-	-	√	1+N
	XOR	A,I	A ← A xor I	-	-	√	1
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
R	RRC	M	A ← RRC M	√	-	-	1
O	RRCM	M	M ← RRC M	√	-	-	1+N
C	RLC	M	A ← RLC M	√	-	-	1
E	RLCM	M	M ← RLC M	√	-	-	1+N
S	CLR	M	M ← 0	-	-	-	1
S	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
B	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
R	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1+ S
A	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
N	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1+ S
C	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1+N+S
H	BTS0	M.b	If M.b = 0, then skip next instruction	-	-	-	1 + S
	BTS1	M.b	If M.b = 1, then skip next instruction	-	-	-	1 + S
	B0BTS0	M.b	If M(bank 0).b = 0, then skip next instruction	-	-	-	1 + S
	B0BTS1	M.b	If M(bank 0).b = 1, then skip next instruction	-	-	-	1 + S
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
M	RET		PC ← Stack	-	-	-	2
I	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
S	PUSH		To push ACC and PFLAG (except NT0, NPD bit) into buffers.	-	-	-	1
C	POP		To pop ACC and PFLAG (except NT0, NPD bit) from buffers.	√	√	√	1
	NOP		No operation	-	-	-	1

***Note:** 1. "M" is system register or RAM. If "M" is system registers then "N" = 0, otherwise "N" = 1.*
2. If branch condition is true then "S = 1", otherwise "S = 0".

13 ELECTRICAL CHARACTERISTIC

13.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2524K, SN8P2524S, SN8P2524X..... 0°C ~ + 70°C
SN8P2524KD, SN8P2524SD, SN8P2524XD..... –40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

13.2 ELECTRICAL CHARACTERISTIC

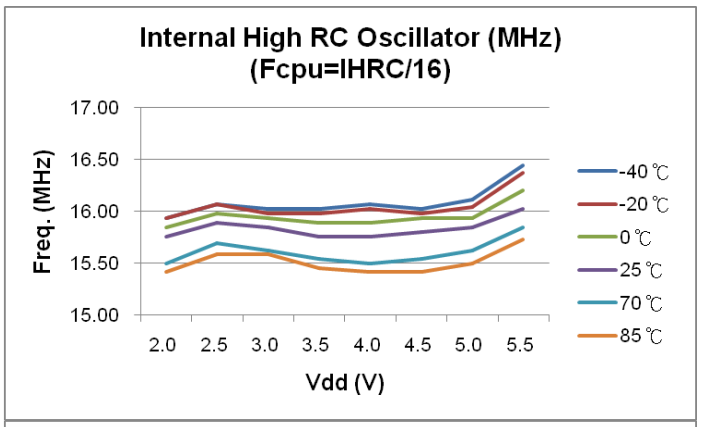
(All of voltages refer to Vss, Vdd = 5.0V, fosc = 16MHz,fcpu=1MHZ,ambient temperature is 25℃ unless otherwise note.)

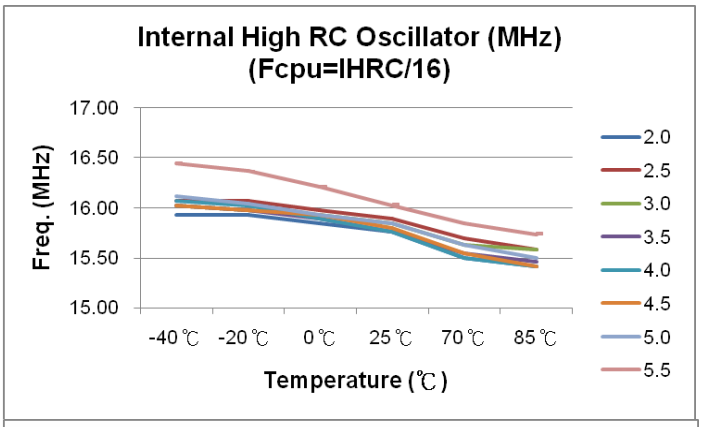
PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, *25℃, Fcpu = 1MHz.*		2.2	-	5.5	V
		Normal mode, Vpp = Vdd, *-40℃~85℃*		2.4	-	5.5	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	150	
I/O output source current	IoH	Vop = Vdd – 0.5V		8	-	-	mA
sink current	IoL1	Vop = Vss + 0.5V		8	-	-	mA
	IoL2	Vop = Vss + 1.5V, P5.3, P5.4 only.		150	200	250	mA
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current (Disable Comparator)	Idd1	Run Mode (No loading, IHRC)	Vdd= 3V, Fcpu = 16MHz/2	-	5	-	mA
			Vdd= 5V, Fcpu = 16MHz/2	-	7	-	mA
			Vdd= 3V, Fcpu = 16MHz/4	-	2	-	mA
			Vdd= 5V, Fcpu = 16MHz/4	-	4	-	mA
			Vdd= 3V, Fcpu = 16MHz/16	-	1.5	-	mA
			Vdd= 5V, Fcpu = 16MHz/16	-	3	-	mA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd= 3V, ILRC=16KHz	-	3.5	-	uA
			Vdd= 5V, ILRC=32KHz	-	10	-	uA
	Idd3	Sleep Mode	Vdd= 5V/3V	-	1	2	uA
	Idd4	Green Mode (No loading, Watchdog Disable)	Vdd= 3V, IHRC=16MHz	-	0.35	-	mA
			Vdd= 5V, IHRC=16MHz	-	0.55	-	mA
			Vdd= 3V, ILRC=16KHz	-	2	-	uA
			Vdd= 5V, ILRC=32KHz	-	5.5	-	uA
Internal High Oscillator Freq.	Fihrc	Internal High RC (IHRC)	*25℃*, Vdd=2.2V~ 5.5V Fcpu=Fhosc/2~Fhosc/16	15.68	16	16.32	MHz
			-40℃~85℃,Vdd=2.4V~ 5.5V Fcpu=Fhosc/2~Fhosc/16	15.2	16	16.8	MHz
*LVD Voltage	Vdet0	Low voltage reset level. *25℃*		1.9	2.0	2.1	V
		Low voltage reset level. *-40℃~85℃*		1.8	2.0	2.3	V
	Vdet1	Low voltage reset/indicator level. *25℃*		2.3	2.4	2.5	V
		Low voltage reset/indicator level. *-40℃~85℃*		2.2	2.4	2.7	V
	Vdet2	Low voltage reset/indicator level. *25℃*		3.5	3.6	3.7	V
		Low voltage reset/indicator level. *-40℃~85℃*		3.3	3.6	3.9	V
Comparator Quiescent Current	Icmq	Iout=0		-	-	1	uA
Comparator Operating Current	Icmop	Internal reference disables. Vdd = 3V		-	30	-	uA
		Internal reference disables. Vdd = 5V		-	40	-	uA
		Internal reference enables. Vdd = 3V		-	50	-	uA
		Internal reference enables. Vdd = 5V		-	65	-	uA
Comparator Input Offset Voltage	*Vcmof	Vcm=Vss		-10	-	+10	mV
Common Mode Input Voltage Range	Vcm	Vdd=5.0V		Vss-0.3	-	Vdd+0.3	V

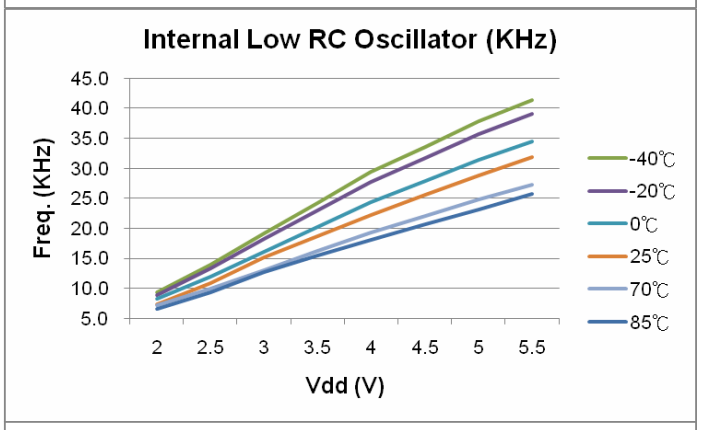
***These parameters are for design reference, not tested.**

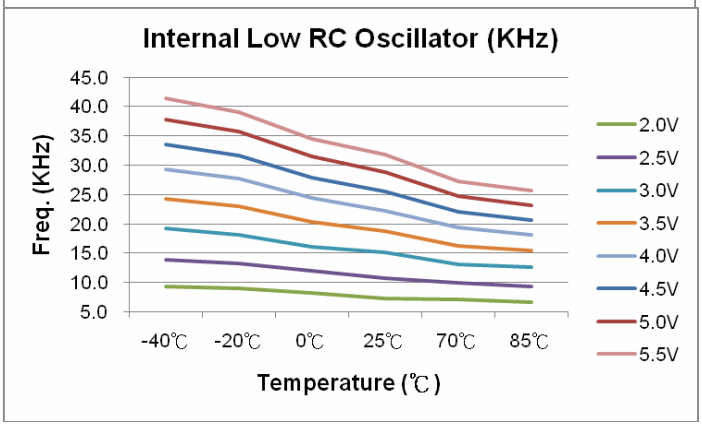
13.3 CHARACTERISTIC GRAPHS

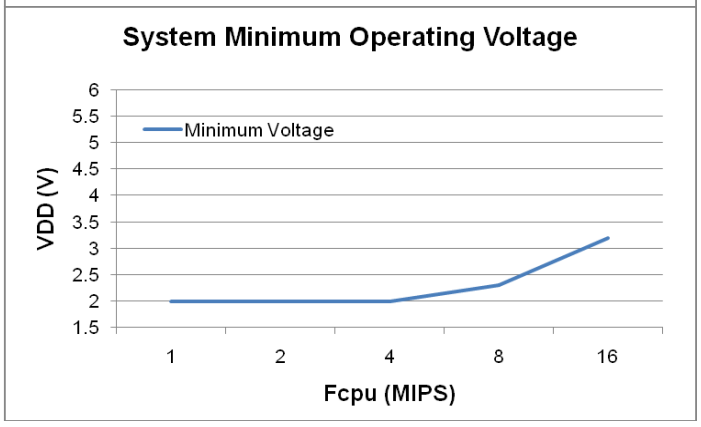
The Graphs in this section are for design guidance, not tested or guaranteed. In some graphs, the data presented are outside specified operating range. This is for information only and devices are guaranteed to operate properly only within the specified range (-40℃~+85℃ curves are for design reference).











14 DEVELOPMENT TOOL

SONIX provides ICE (in circuit emulation), IDE (Integrated Development Environment) and EV- KIT for SN8P2524 development. ICE and EV-KIT are external hardware devices, and IDE is a friendly user interface for firmware development and emulation. These development tools' version is as following.

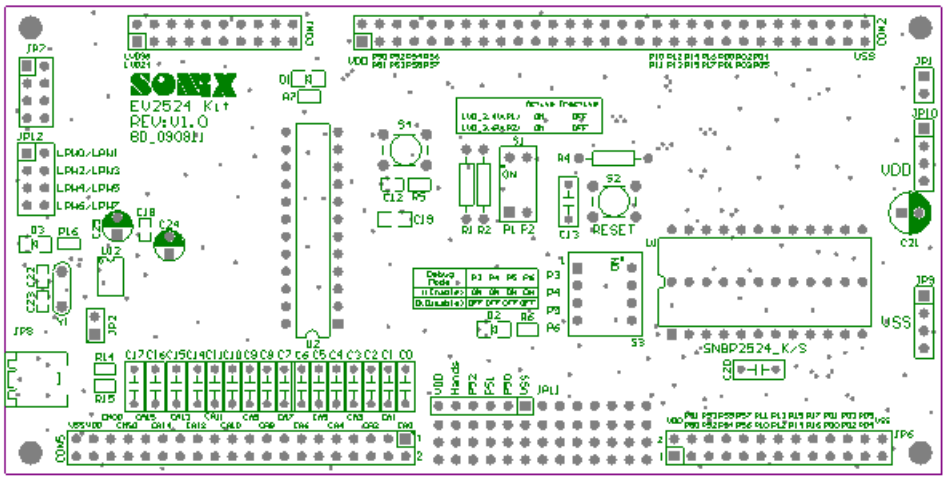
- **ICE: SN8ICE2K Plus II. (Please install 16MHz crystal in ICE to implement IHRC emulation.).**
- **ICE emulation speed maximum: 8 MIPS @ 5V (e.g. 16Mhz crystal, Fcpu = Fosc/2).**
- **EV-KIT: EV2524 Kit REV: V1.0.**
- **IDE: SONiX IDE M2IDE_V132 or greater.**
- **Writer: MPIII writer.**
- **Writer transition board: SN8P2523**

14.1 SN8P2524 EV-KIT

SN8P2524 EV- KIT includes ICE interface, GPIO interface and EV-chip, PWM module.

EV-chip module: Emulate 16 channel comparator function.
PWM module: Emulate 8 channel pwm function.

SN8P2524 EV- KIT PCB Outline:



- CON1, CON2: ICE interface connected to SN8ICE2K Plus 2.
- JP6: GPIO connector.
- U2: SN8P2524 EV-chip for comparator emulation.
- U1: SN8P2524 SKDIP/SOP socket for connecting to user's target board.
- S1: LVD24 and LVD36 emulating switch.
- C0~C11, C14~C17: 16 channel comparator capacitors.
- CON5 (CA0~CA15): 16 channel comparator connectors.
- JP12: 8 channel PWM connectors.

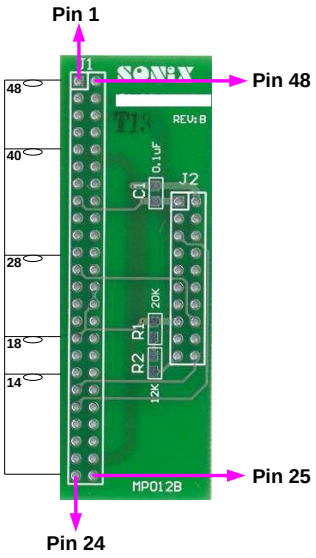
14.2 ICE and EV-KIT APPLICATION NOTIC

1. SN8P2524 EV- KIT includes comparator emulation module and 8-pwm module.
2. The SN8P2524 EV-chip programmed emulating code to emulate comparator function.
3. For comparator emulation, input comparator signals are from CA0~CA15 pins.
4. For 8-PWM emulation, output pins are from connector JP12 (LPW0~LPW7).
5. For the GPIO function emulation is from connector JP6.
6. SN8P2524 EV- KIT power switch table as follows:

	JP1	JP2	JP10
ICE Power	**SHORT**	OPEN	OPEN
External Power	OPEN	OPEN	**SHORT**
USB Power	OPEN	**SHORT**	OPEN

15 OTP PROGRAMMING PIN

15.1 WRITER TRANSITION BOARD SOCKET PIN ASSIGNMENT



JP3 (Mapping to 48-pin text tool)

DIP 1	1	48	DIP48
DIP 2	2	47	DIP47
DIP 3	3	46	DIP46
DIP 4	4	45	DIP45
DIP 5	5	44	DIP44
DIP 6	6	43	DIP43
DIP 7	7	42	DIP42
DIP 8	8	41	DIP41
DIP 9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP37
DIP13	13	36	DIP36
DIP14	14	35	DIP35
DIP15	15	34	DIP34
DIP16	16	33	DIP33
DIP17	17	32	DIP32
DIP18	18	31	DIP31
DIP19	19	30	DIP30
DIP20	20	29	DIP29
DIP21	21	28	DIP28
DIP22	22	27	DIP27
DIP23	23	26	DIP26
DIP24	24	25	DIP25

Writer JP1/JP2

VDD	1	2	VSS
CLK/PGCLK	3	4	CE
PGM/OTPCLK	5	6	OE/ShiftDat
D1	7	8	D0
D3	9	10	D2
D5	11	12	D4
D7	13	14	D6
VDD	15	16	VPP
HLS	17	18	RST
-	19	20	ALSB/PDB

JP1 for Writer transition board
JP2 for dice and >48 pin package

15.2 PROGRAMMING PIN MAPPING:

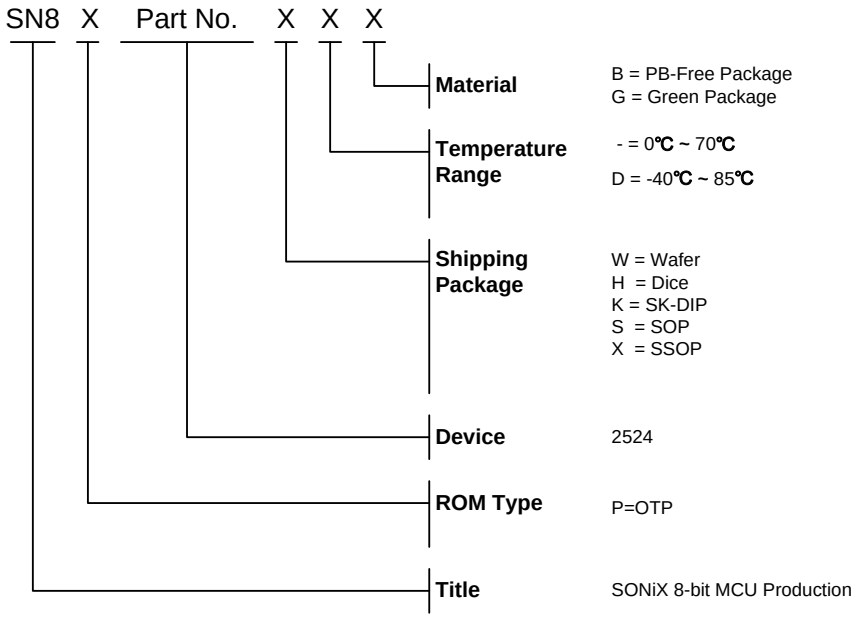
Programming Pin Information of SN8P2524 Series							
Chip Name		SN8P2524K/S(SKDIP/SOP)			SN8P2524X(SSOP)		
Writer Connector		IC and JP3 48-pin text tool Pin Assignment					
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number
1	VDD	7	VDD	19	9	VDD	19
2	GND	18	VSS	30	20	VSS	30
3	CLK	4	P1.2	16	6	P1.2	16
4	CE	-	-	-	-	-	-
5	PGM	6	P1.0	18	8	P1.0	18
6	OE	3	P1.3	15	5	P1.3	15
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	8	RST	20	10	RST	20
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	5	P1.1	17	7	P1.1	17

16 Marking Definition

16.1 INTRODUCTION

There are many different types in Sonix 8-bit MCU production line. This note listed the production definition of all 8-bit MCU for order or obtain information. This definition is only for Blank OTP MCU.

16.2 MARKING INDETIFICATION SYSTEM



16.3 MARKING EXAMPLE

- **Wafer, Dice:**

Name	ROM Type	Device	Package	Temperature	Material
S8P2524W	OTP	2524	Wafer	0℃~70℃	-
SN8P2524H	OTP	2524	Dice	0℃~70℃	-

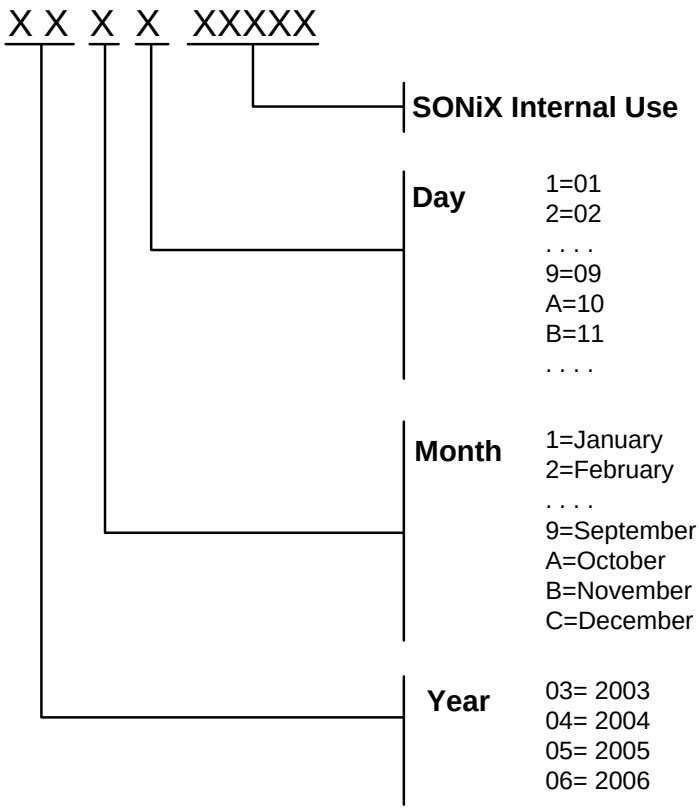
- **Green Package:**

Name	ROM Type	Device	Package	Temperature	Material
SN8P2524KG	OTP	2524	SK-DIP	0℃~70℃	Green Package
SN8P2524SG	OTP	2524	SOP	0℃~70℃	Green Package
SN8P2524XG	OTP	2524	SSOP	0℃~70℃	Green Package
SN8P2524KDG	OTP	2524	SK-DIP	-40℃~85℃	Green Package
SN8P2524SDG	OTP	2524	SOP	-40℃~85℃	Green Package
SN8P2524XDG	OTP	2524	SSOP	-40℃~85℃	Green Package

- **PB-Free Package:**

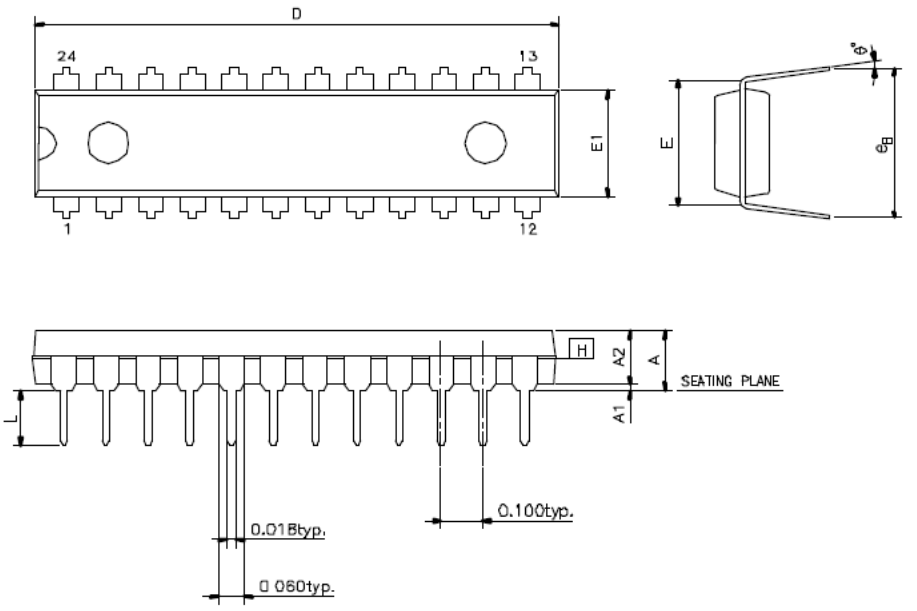
Name	ROM Type	Device	Package	Temperature	Material
SN8P2524KB	OTP	2524	SK-DIP	0℃~70℃	PB-Free Package
SN8P2524SB	OTP	2524	SOP	0℃~70℃	PB-Free Package
SN8P2524XB	OTP	2524	SSOP	0℃~70℃	PB-Free Package
SN8P2524KDB	OTP	2524	SK-DIP	-40℃~85℃	PB-Free Package
SN8P2524SDB	OTP	2524	SOP	-40℃~85℃	PB-Free Package
SN8P2524XDB	OTP	2524	SSOP	-40℃~85℃	PB-Free Package

16.4 DATECODE SYSTEM



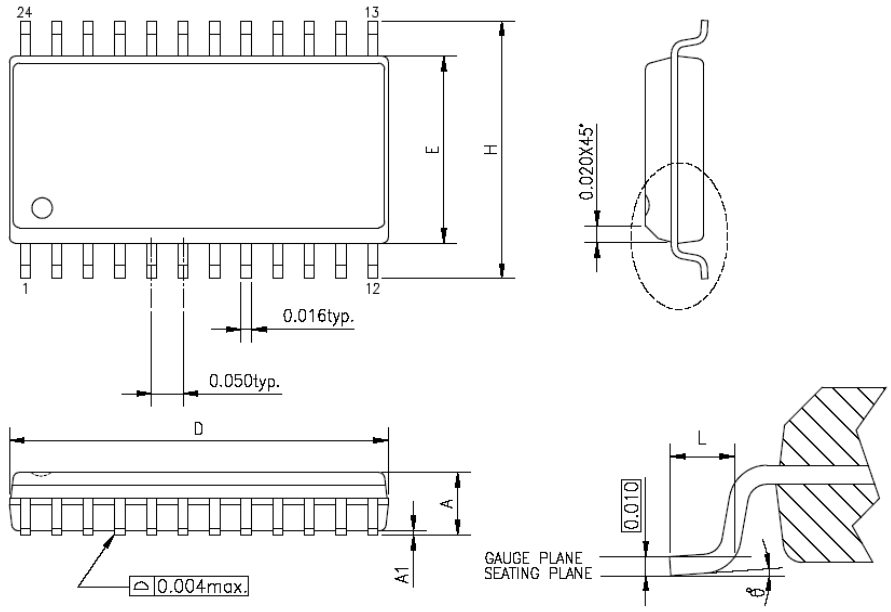
17 PACKAGE INFORMATION

17.1 SK-DIP 24 PIN



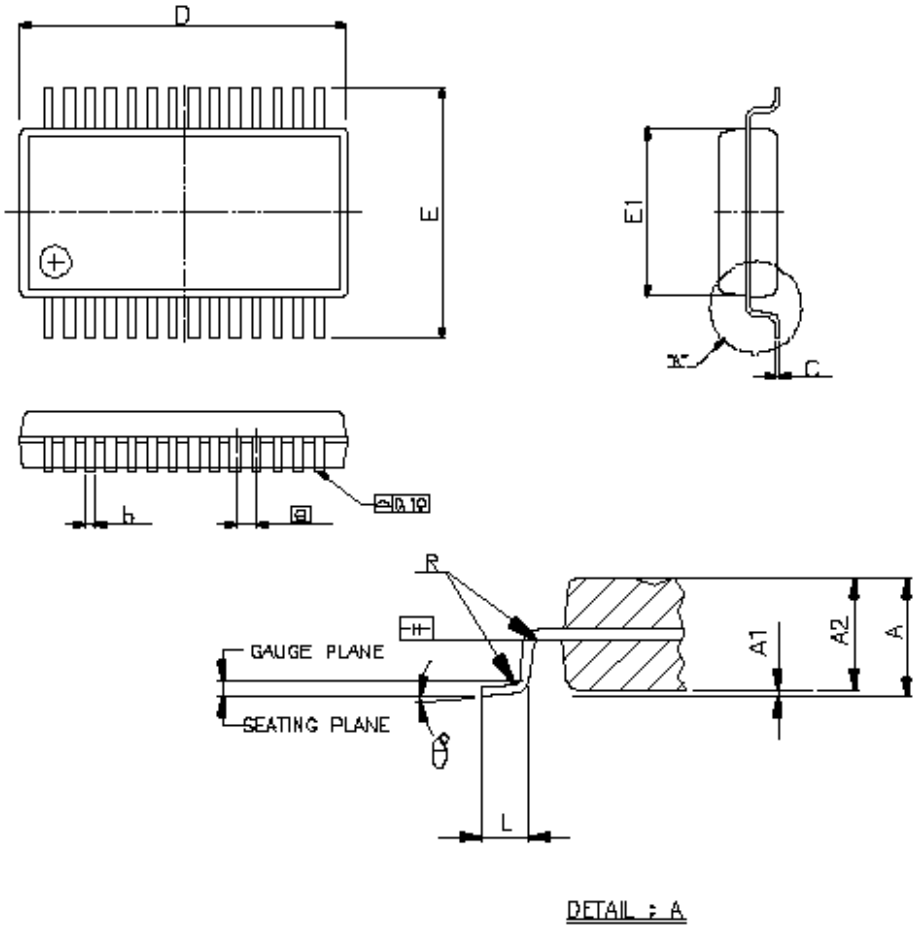
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-		0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.735	0.755	0.775	18.669	19.177	19.685
E	0.30 BSC			7.620 BSC		
E1	0.253	0.258	0.263	6.426	6.553	6.680
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509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

17.2 SOP 24 PIN



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

17.3 SSOP 28 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.08	-	-	2.13
A1	0.00	-	0.01	0.05	-	0.25
A2	0.06	0.07	0.07	1.63	1.75	1.88
b	0.01	-	0.01	0.22	-	0.38
C	0.00	-	0.01	0.09	-	0.20
D	0.39	0.40	0.41	9.90	10.20	10.50
E	0.29	0.31	0.32	7.40	7.80	8.20
E1	0.20	0.21	0.22	5.00	5.30	5.60
[e]	0.0259BSC			0.65BSC		
L	0.02	0.04	0.04	0.63	0.90	1.03
R	0.00	-	-	0.09	-	-
θ°	0°	4°	8°	0°	4°	8°

Main Office:
Address: 10F-1, NO. 36, Taiyuan Stree, Chupei City, Hsinchu, Taiwan R.O.C.
Tel: 886-3-560 0888
Fax: 886-3-560 0889
Taipei Office:
Address: 15F-2, NO. 171, Song Ted Road, Taipei, Taiwan R.O.C.
Tel: 886-2-2759 1980
Fax: 886-2-2759 8180
Hong Kong Office:
Unit 1519, 15/F Chevalier Commerical Centre, No.8 Wang Hoi Road, Kowloon, Kln., Hong Kong.
Tel: 852-2723-8086
Fax: 852-2723-9179
Technical Support by Email:
Sn8fae@sonix.com.tw