SN8P26L38

USER'S MANUAL

Version 1.3

SN8P26L38

SONiX 8-Bit Micro-Controller

AMENDMENT HISTORY

Version	Date	Description
VER 0.1	Dec. 2007	First Issue.
VER 0.2	Jan. 2008	Modify system register table.
VER 0.3	Feb. 2008	Modify internal low RC frequency to 10KHz.
VER 0.4	Otc. 2008	Add SN8P26L38F LQFP package type.
VER 1.0	Aug. 2009	Modify comparator pin assignment description.
VER 1.1	Jun. 2010	1. Modify UART baud rate description. 2. Modify IR register name. 3. Modify SYSTEM REGISTER TABLE.
VER. 1.2	Feb. 2013	Modify LQFP package type marking definition.
VER. 1.3	Apr. 2013	Add RAM limited: The 1E6H, 1E7H of RAM address doesn't support directly addressing mode to access RAM but support indirectly addressing mode @HL/@YZ.

Table of Content

1 PRODUCT OVERVIEW

1.1 FEATURES

- **Memory configuration**
  OTP ROM size: 8K * 16 bits.
  RAM size: 880 * 8 bits.

- **8 levels stack buffer**

- **I/O pin configuration**
  Bi-directional: P0, P1, P2, P3, P4, P5
  Programmable open-drain: P1.0, P1.1, P5.0~P5.2, P3.2, P3.3.
  Wakeup:P0, P1 level change trigger.
  P1 wake-up function controlled by P1W.
  Pull-up resisters: P0, P1, P2, P3, P4, P5
  External interrupt input: P0.0, P0.1
  External Interrupt trigger edge:
  P0.0 controlled by PEDGE register

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

- **2-ch analog comparators with internal selectable reference voltage 0.9V/1.0V/1.1V/1.2V and external reference input.**

- **8 interrupt sources**
  6 internal interrupts: T0, TC1, CM0, CM1, SIO, UART
  2 external interrupts: INT0 INT1

- **Powerful instructions**
  One clock per instruction cycle (1T)
  All ROM area JMP instruction.
  All ROM area CALL address instruction.
  All ROM area lookup table function (MOVC)

- **Two 8-bit Timer/Counter**
  T0: Basic timer.
  TC1: Auto-reload timer/counter.

- **One RTC timer (T0).**
- **One channels PWM output.**
- **One channels buzzer output.**
- **One channel IR output (duty/cycle programmable PWM, TC0).**
- **On chip watchdog timer and clock source is internal low clock RC type (about 10KHz @3V).**

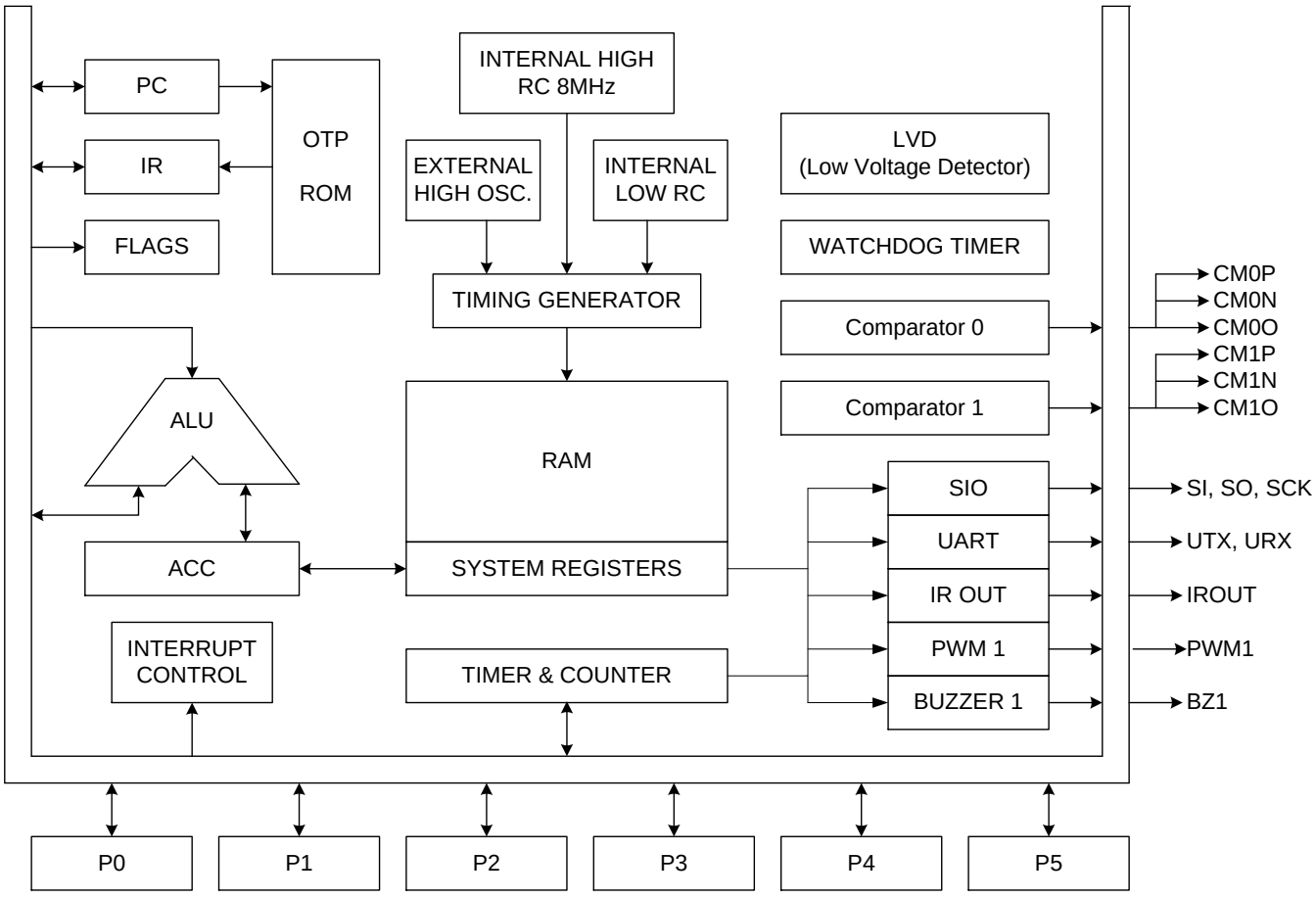
- **One channel SIO interface.**
- **One channel UART interface.**

- **Four system clocks**
  External high clock: RC type up to 8 MHz
  External high clock: Crystal type up to 8 MHz
  Internal high clock: RC type 8MHz.
  Internal low clock: RC type 10KHz(3V).

- **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)**
  P-DIP 48 pins
  SSOP 48 pins
  LQFP 48 pins

1.2 SYSTEM BLOCK DIAGRAM

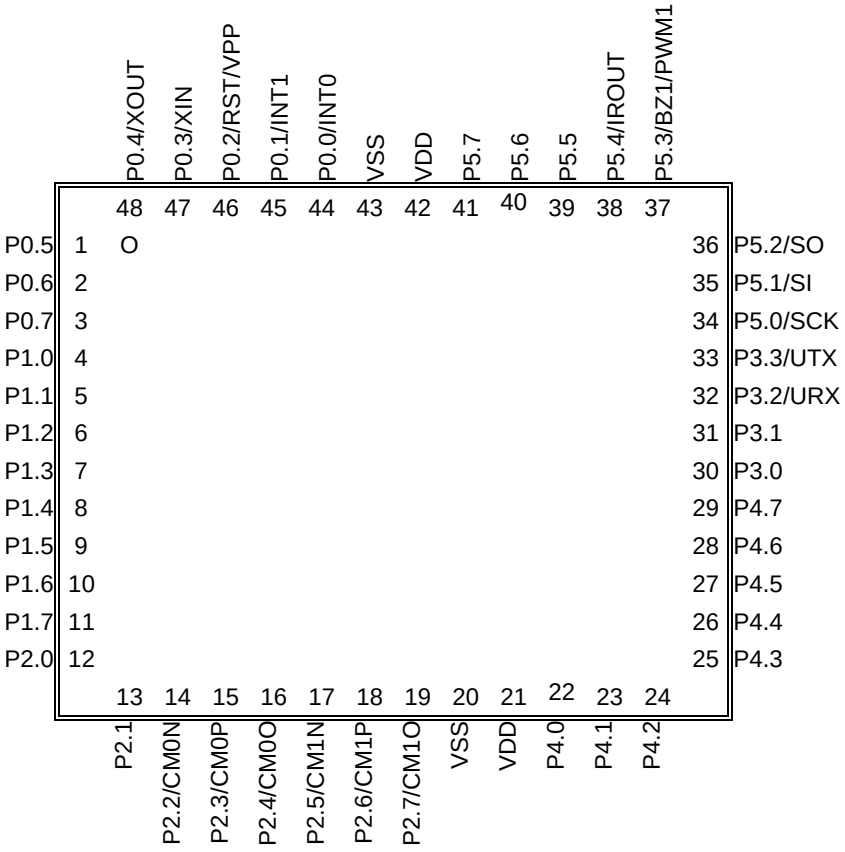


1.3 PIN ASSIGNMENT

SN8P26L38P (P-DIP 48 pins)
SN8P26L38X (SSOP 48 pins)

VSS	1	U	48	VDD
P0.0/INT0	2		47	P5.7
P0.1/INT1	3		46	P5.6
P0.2/RST/VPP	4		45	P5.5
P0.3/XIN	5		44	P5.4/IROUT
P0.4/XOUT	6		43	P5.3/PWM1/BZ1
P0.5	7		42	P5.2/SO
P0.6	8		41	P5.1/SI
P0.7	9		40	P5.0/SCK
P1.0	10		39	P3.3/UTX
P1.1	11		38	P3.2/URX
P1.2	12		37	P3.1
P1.3	13		36	P3.0
P1.4	14		35	P4.7
P1.5	15		34	P4.6
P1.6	16		33	P4.5
P1.7	17		32	P4.4
P2.0	18		31	P4.3
P2.1	19		30	P4.2
P2.2/CM0N	20		29	P4.1
P2.3/CM0P	21		28	P4.0
P2.4/CM0O	22		27	VDD
P2.5/CM1N	23		26	VSS
P2.6/CM1P	24		25	P2.7/CM1O

SN8P26L38F (LQFP 48 pins)



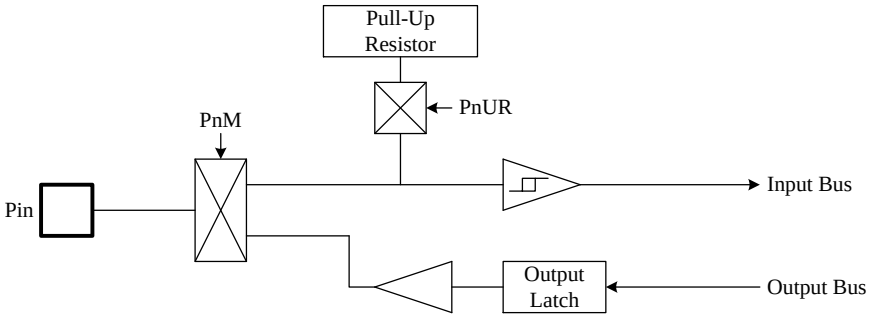
1.4 PIN DESCRIPTIONS

PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital and analog circuit.
P0.2/RST/ VPP	I, P	RST: System external reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”. Build-in wake-up function.
		VPP: OTP power input pin in programming mode.
		P0.2: Input only pin with Schmitt trigger structure and no pull-up resistor.
XIN/P0.3	I/O	XIN: Oscillator input pin while external oscillator enable (crystal and RC).
		P0.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Build-in wake-up function.
XOUT/P0.4	I/O	XOUT: Oscillator output pin while external crystal enable.
		P0.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Build-in wake-up function.
P0.0/INT0	I/O	P0.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Build-in wake-up function.
		INT0: External interrupt 0 input pin.
P0.1/INT1	I/O	P0.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Build-in wake-up function.
		INT1: External interrupt 0 input pin.
		TC1 event counter input pin.
P0[7:5]	I/O	P0[7:5]: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Build-in wake-up function.
P1[1:0]	I/O	P1[1:0]: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Build-in wake-up function. Open-drain structure controlled by P1OC register.
P1[7:0]	I/O	P1[7:2]: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Build-in wake-up function.
P2[1:0]	I/O	P2[1:0]: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
P2.2/CM0N	I/O	P2.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM0N: The negative input pin of comparator.
P2.3/CM0P	I/O	P2.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM0P: The positive input pin of comparator.
		BTO: Band-gap trimming mode output pin.
P2.4/CM0O	I/O	P2.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM0O: The output pin of comparator.
P2.5/CM1N	I/O	P2.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM1N: The negative input pin of comparator.
P2.6/CM1P	I/O	P2.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM1P: The positive input pin of comparator.
P2.7/CM1O	I/O	P2.7: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		CM1O: The output pin of comparator.
P3[1:0]	I/O	P3[1:0]: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Open-drain structure controlled by P1OC register.
P3.2/URX	I/O	P3.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Open-drain structure controlled by P1OC register.
		URX: UART data receive pin.
P3.3/UTX	I/O	P3.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Open-drain structure controlled by P1OC register.
		UTX: UART data transmit pin.
P4[7:0]	I/O	P4[7:0]: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
P5.0/SCK	I/O	P5.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Open-drain structure controlled by P1OC register.
		SCK: SIO clock pin.
P5.1/SI	I/O	P5.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Open-drain structure controlled by P1OC register.
		SI: SIO data input pin.
P5.2/SO	I/O	P5.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.

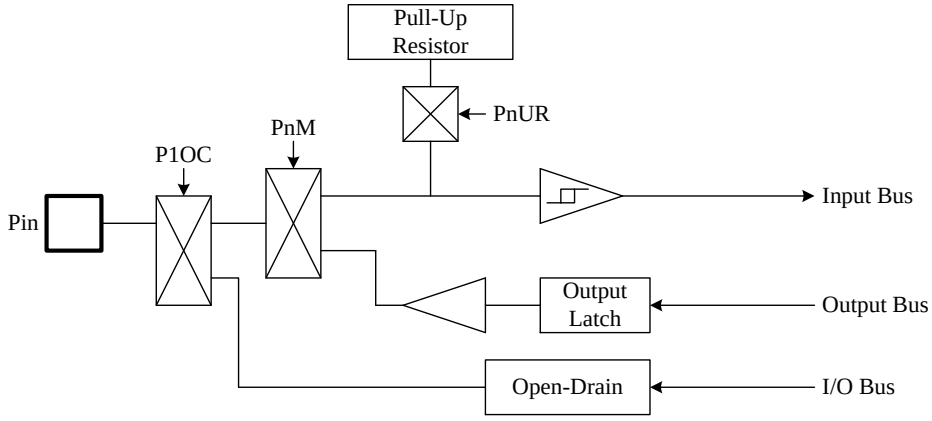
		Open-drain structure controlled by P1OC register.
		SO: SIO data output pin.
P5.3/BZ1/PWM1	I/O	P5.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		BZ1: Programmable buzzer output pin from TC1/2 signal.
		PWM1: Programmable PWM output pin from TC1.
P5.4/IROUT	I/O	P5.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		IROUT: IR signal output pin.
P5[7:5]	I/O	P5[7:5]: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.

1.5 PIN CIRCUIT DIAGRAMS

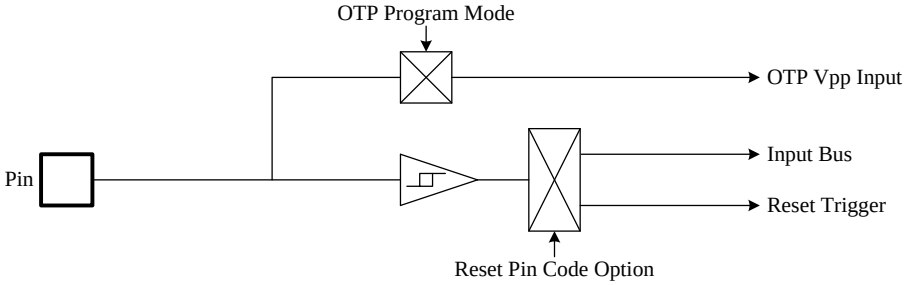
- **General purpose I/O pin:**



- **General purpose I/O pin with open-drain structure:**

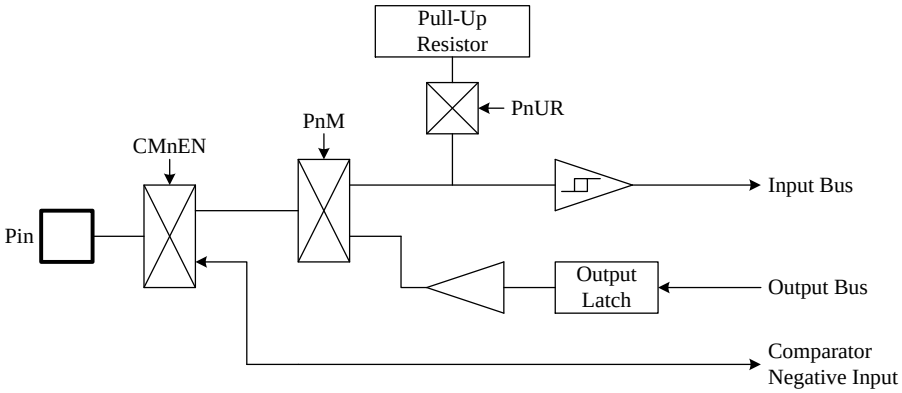


- **Input only pin shared with reset pin:**

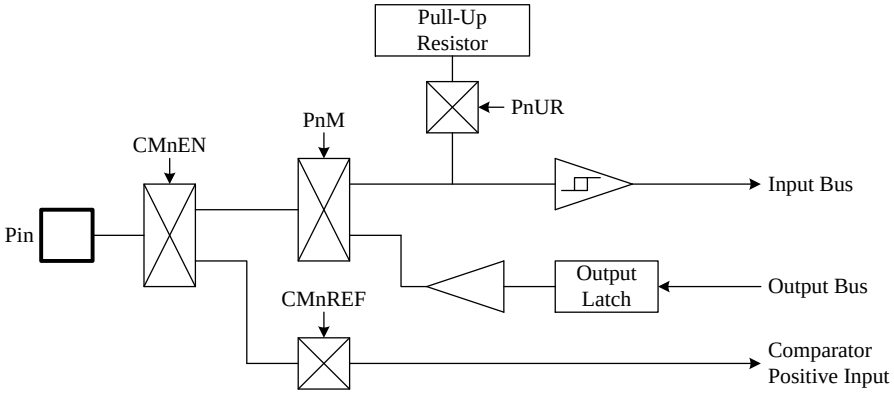


- **General purpose I/O pin shared with Comparator:**

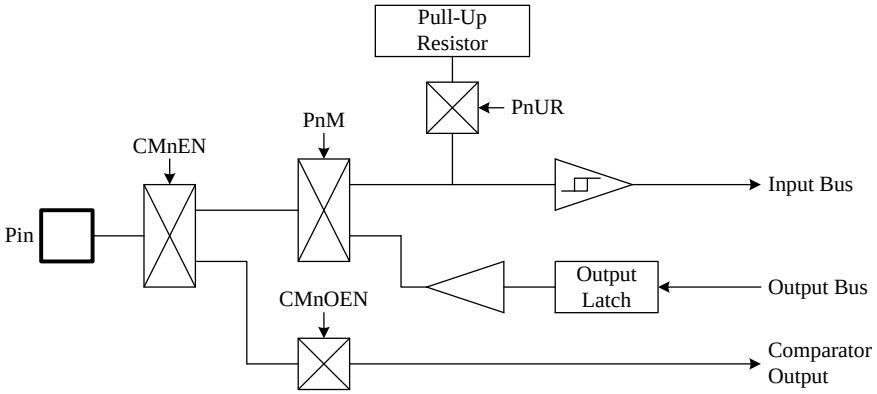
Comparator Negative Pin:



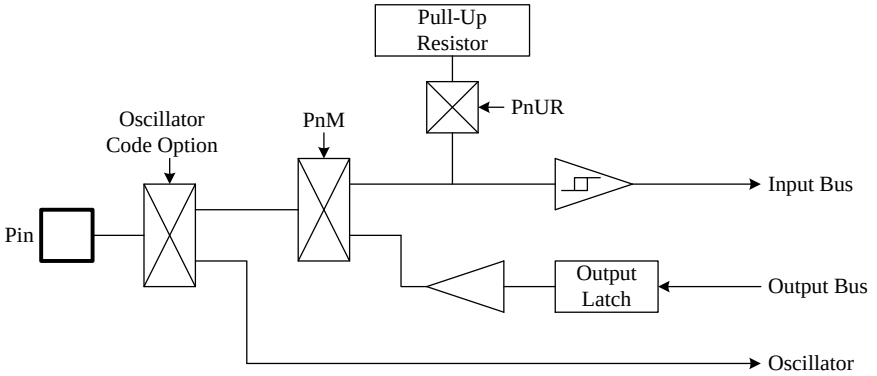
Comparator Positive Pin:



Comparator Output Pin:



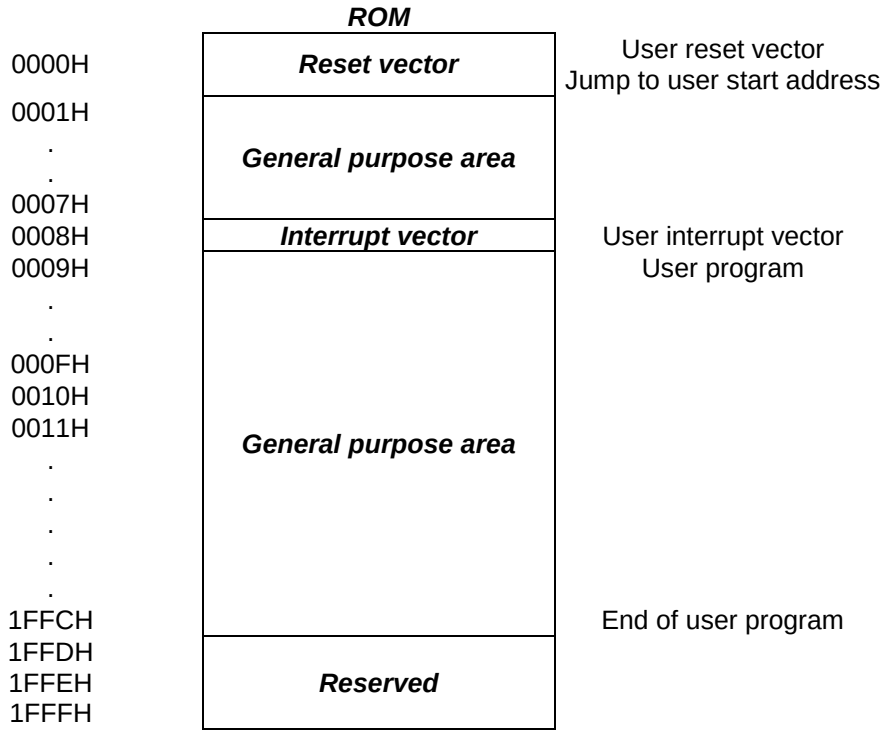
- **General purpose I/O pin shared with external oscillator:**



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (ROM)

☞ ***8K 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 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
        ADD     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

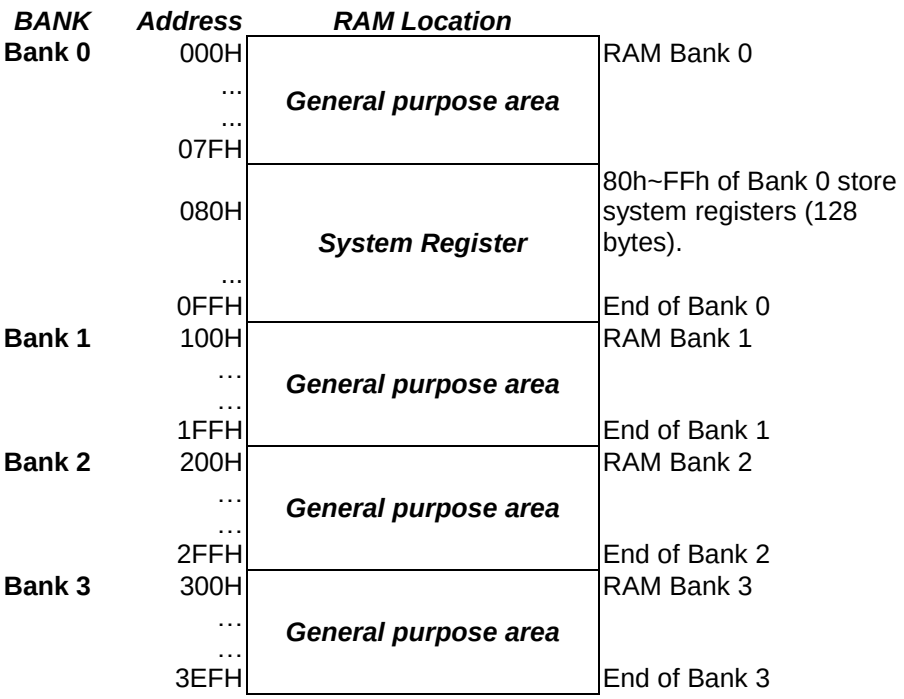
The last ROM address are reserved area. User should avoid these addresses (last address) when calculate the Checksum value.

➢ **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)

☞ ***880 X 8-bit RAM***



The 880-byte general purpose RAM is separated into Bank 0~Bank 3. Accessing the two banks' RAM is controlled by "RBANK" register. When RBANK = 0, the program controls Bank 0 RAM directly. When RBANK = 1, the program controls Bank 1 RAM directly. Under one bank condition and need to access the other bank RAM, setup the RBANK register is necessary. Sonix provides "Bank 0" type instructions (e.g. b0mov, b0add, b0bts1, b0bset…) to control Bank 0 RAM in non-zero RAM bank condition directly.

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

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

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

✸ ***The 1E6H, 1E7H of RAM address doesn't support directly addressing mode to access RAM but support indirectly addressing mode @HL/@YZ.***

2.2.1 SYSTEM REGISTER

2.2.1.1SYSTEM 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	-	-	-	-	-	-	-	-	-	-	-	-	CMP0M	CMP1M	-	-
A	T1M	T1CL	T1CH	-	URTX	URRX	URBRC	URTXD1	URTXD2	URRXD1	URRXD2	-	-	-	-	-
B	-	-	-	-	SIOM	SIOR	SIOB	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M	P3M	P4M	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	IRR	PCL	PCH
D	P0	P1	P2	P3	P4	P5	-	-	T0M	T0C	IRM	IRC	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	P2UR	P3UR	P4UR	P5UR	@HL	@YZ	IRD	P1OC	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2SYSTEM REGISTER DESCRIPTION

PFLAG = ROM page and special flag register.
H, L = Working, @HL and ROM addressing register.
P1W = Port 1 wakeup register.
CMPnM = Comparator control register.
PEDGE = P0.0 edge direction register.
PnM = Port n input/output mode register.
P1OC = Open-drain control register.
INTRQ = Interrupt request register.
OSCM = Oscillator mode register.
T0M = T0 mode register.
TC1M = TC1 mode control register.
TC1R = TC1 auto-reload buffer.
T1CH,L = T1 16-bit counter register.
IRC = IR cycle control register.
IRD = IR duty control register.
URTX = UART transmit control register.
URTXD1,2 = UART transmit data buffers.
URBRC = UART baud rate control register.
SIOR = SIO clock rate control register.
STKP = Stack pointer buffer.

R = Working register and ROM look-up data buffer.
Y, Z = Working, @YZ and ROM addressing register.
RBANK = Ram bank selection register.
@HL = RAM HL indirect addressing index pointer.
@YZ = RAM YZ indirect addressing index pointer.
Pn = Port n data buffer.
PnUR = Port n pull-up resister control register.
INTEN = Interrupt enable register.
PCH, PCL = Program counter.
T0C = T0 counting register.
TC1C = TC1 counter register.
T1M = T1 mode register.
IRM = IR output control register.
IRR = IR auto-reload register.
WDTR = Watchdog timer clear register.
URRX = UART receive control register.
URRXD1,2 = UART receive data buffers.
SIOM = SIO mode control register.
SIOB = SIO data buffer.
STK0~STK7 = Stack 0 ~ stack 7 buffer.

2.2.1.3BIT 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	LVD28	LVD24	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	RBANKS2	RBANKS1	RBANKS0	R/W	RBANK
09CH	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CMS1	CMS0	R/W	CMP0M
09DH	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	-	-	R/W	CMP1M
0A0H	T1ENB	T1rate2	T1rate1	T1rate0					R/W	T1M
0A1H	T1CL7	T1CL6	T1CL5	T1CL4	T1CL3	T1CL2	T1CL1	T1CL0	R/W	T1CL
0A2H	T1CH7	T1CH6	T1CH5	T1CH4	T1CH3	T1CH2	T1CH1	T1CH0	R/W	T1CH
0A4H				UTXEN	UTXPEN	UTXPS	UTXM		R/W	URTX
0A5H	URXEN	URXS1	URXS0	URXPEN	URXPS	URXPC	URXM		R/W	URRX
0A6H	UDIV4	UDIV3	UDIV2	UDIV1	UDIV0	UPCS2	UPCS1	UPCS0	R/W	URBRC
0A7H	UTXD17	UTXD16	UTXD15	UTXD14	UTXD13	UTXD12	UTXD11	UTXD10	R/W	URTXD1
0A8H	UTXD27	UTXD26	UTXD25	UTXD24	UTXD23	UTXD22	UTXD21	UTXD20	R/W	URTXD2
0A9H	URXD17	URXD16	URXD15	URXD14	URXD13	URXD12	URXD11	URXD10	R/W	URRXD1
0AAH	URXD27	URXD26	URXD25	URXD24	URXD23	URXD22	URXD21	URXD20	R/W	URRXD2
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	P07M	P06M	P05M	P04M	P03M	-	P01M	P00M	R/W	P0M
0BFH	-	-	-	P00G1	P00G0	-	-	-	R/W	PEDGE
0C0H	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W wakeup register
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M I/O direction
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M I/O direction
0C3H	-	-	-	-	P33M	P32M	P31M	P30M	R/W	P3M I/O direction
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M I/O direction
0C5H	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M I/O direction
0C8H	SIOIRQ	TC1IRQ	T1IRQ	T0IRQ	RXIRQ	TXIRQ	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	SIOIEN	TC1IEN	T1IEN	T0IEN	RXIEN	TXIEN	P01IEN	P00IEN	R/W	INTEN
0CAH	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0	W	IRR
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0 data buffer
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1 data buffer
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2 data buffer
0D3H	-	-	-	-	P33	P32	P31	P30	R/W	P3 data buffer
0D4H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4 data buffer
0D5H	P57	P56	P55	P54	P53	P52	P51	P50	R/W	P5 data buffer
0D8H	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	-	-	-	-	-	-	IREN	CREN	R/W	IRM
0DBH	IRC7	IRC6	IRC5	IRC4	IRC3	IRC2	IRC1	IRC0	R/W	IRC
0DCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	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 stack pointer
0E0H	P07R	P06R	P05R	P04R	P03R	-	P01R	P00R	W	P0 pull-up register
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1 pull-up register
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2 pull-up register
0E3H	-	-	-	-	P33R	P32R	P31R	P30R	W	P3 pull-up register
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4 pull-up register
0E5H	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R	W	P5 pull-up register
0E6H	@HL7	@ HL 6	@ HL5	@ HL4	@ HL3	@ HL2	@ HL1	@ HL0	R/W	@HL index pointer
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ index pointer
0E8H	IRD7	IRD6	IRD5	IRD4	IRD3	IRD2	IRD1	IRD0	W	IRD
0E9H	P52OC	P51OC	P50OC	P33OC	P32OC	-	P11OC	P10OC	W	P1OC
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	1	1	1	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	1	1	1	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L

0F5H	1	1	1	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	1	1	1	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	1	1	1	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	1	1	1	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	1	1	1	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	1	1	1	S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ ***Note:***

1. ***To avoid system error, please be 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.***
5. ***For detail description, please refer to the "System Register Quick Reference Table".***

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, LVD28 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	LVD28	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 **LVD30:** LVD 2.8V operating flag and only support LVD code option is LVD_H.
0 = Inactive (VDD > 2.8V).
1 = Active (VDD ≦ 2.8V).

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

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

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

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

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

2.2.4 PROGRAM COUNTER

The program counter (PC) is a 13-bit binary counter separated into the high-byte 5 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

Besides, it can be replaced with specific address by executing CALL or JMP instruction. When JMP or CALL instruction is executed, the destination address will be inserted to bit 0 ~ bit 12.

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

☞ **ONE ADDRESS SKIPPING**

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

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

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

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

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

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

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

INCS instruction:

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

INCMS instruction:

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

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

DECS instruction:

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

DECMS instruction:

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

☞ **MULTI-ADDRESS JUMPING**

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

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

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

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

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

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

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

2.2.5 H, L REGISTERS

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

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

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

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	X	X	X	X	X	X	X	X

➢ **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 REGISTERS

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

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

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

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

2.3 ADDRESSING MODE

2.3.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.3.2 DIRECTLY ADDRESSING MODE

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

Example: Move 0x12 RAM location data into ACC.

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

Example: Move ACC data into 0x12 RAM location.

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

2.3.3 INDIRECTLY ADDRESSING MODE

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

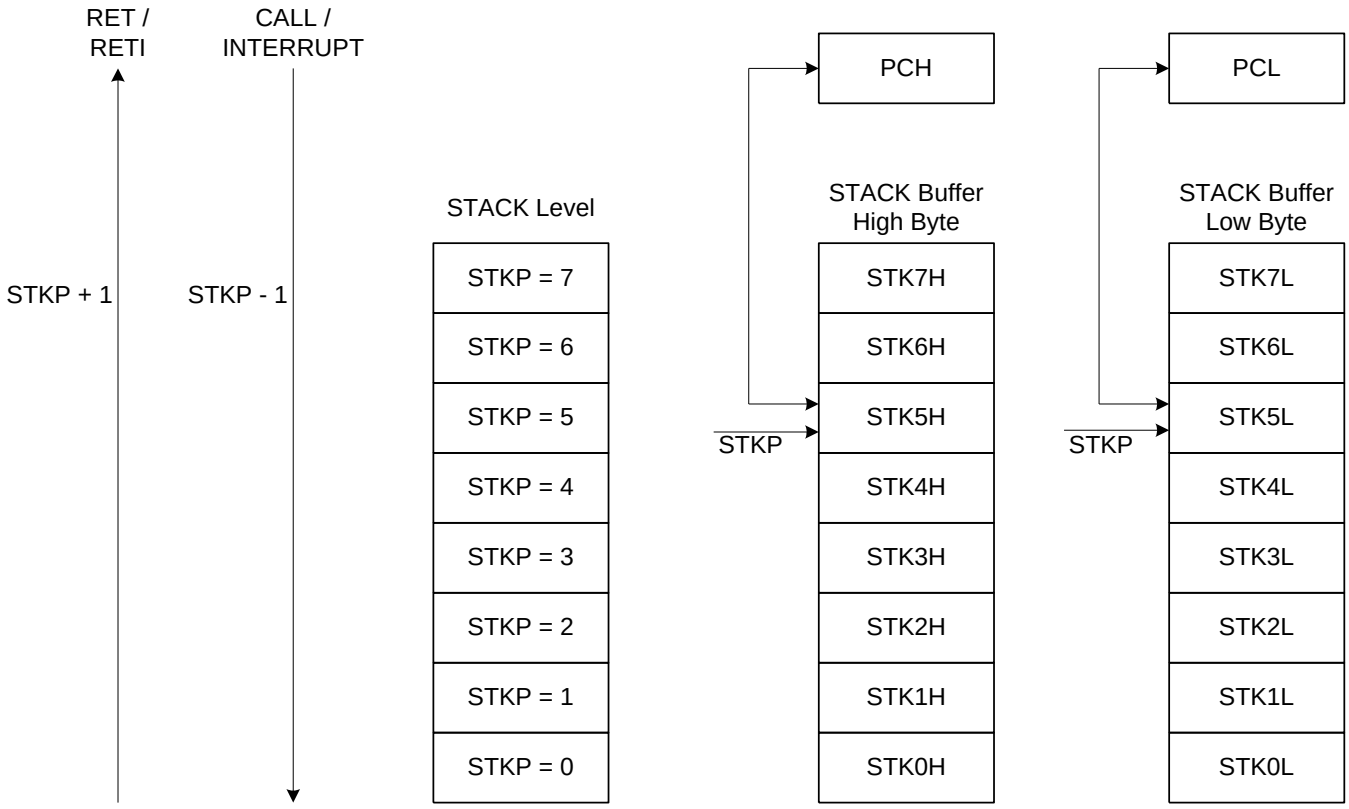
- **Example: Indirectly addressing mode with @YZ register.**

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

2.4 STACK OPERATION

2.4.1 OVERVIEW

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



2.4.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 13-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

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

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

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

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

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

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

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

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

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

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

2.4.3 STACK OPERATION EXAMPLE

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

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

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

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

2.5 CODE OPTION TABLE

The code option is the system hardware configurations including oscillator type, 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		
		Fcpu	Fhosc/1	Instruction cycle is oscillator clo
			Fhosc/2	Instruction cycle is 2 oscillator c
			Fhosc/4	Instruction cycle is 4 oscillator c
			Fhosc/8	Instruction cycle is 8 oscillator c
High_Clk	IHRC_8M	High speed internal 8MHz RC. XIN/XOUT become to P0.3/P0.4 bi-direction I/O pins.		
	IHRC_RTC	High speed internal 8MHz RC with 0.5sec RTC. XIN/XOUT become to P0.3/P0.4 bit-direction I/O pins.		
	RC	Low cost RC for external high clock oscillator and XOUT becomes to P0.4 bit-direction I/O pin.		
	32K X'tal	Low frequency, power saving crystal (e.g. 32.768KHz) for external high clock oscillator.		
	8M X'tal	High speed crystal /resonator (e.g. 8MHz) for external high clock oscillator.		
	4M X'tal	Standard crystal /resonator (e.g. 4M) for external high clock oscillator.		
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.		
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.		
	Disable	Disable Watchdog function.		
Reset_Pin	Reset	Enable External reset pin.		
	P02	Enable P0.2 input only without pull-up resister.		
LVD	LVD_L	LVD will reset chip if VDD is below 1.8V		
	LVD_M	LVD will reset chip if VDD is below 1.8V 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 LVD28 bit of PFLAG register for 2.8V low voltage indicator.		
Security	Enable	Enable ROM code Security function.		
	Disable	Disable ROM code Security function.		

2.5.1 RESET_PIN CODE OPTION

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

- **Reset:** The reset pin external reset function. When falling edge trigger occurring, the system will be reset.
- **P02:** Set reset pin to general purpose input only pin (P0.2). The external reset function is disable and the pin is input pin.

2.5.2 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)

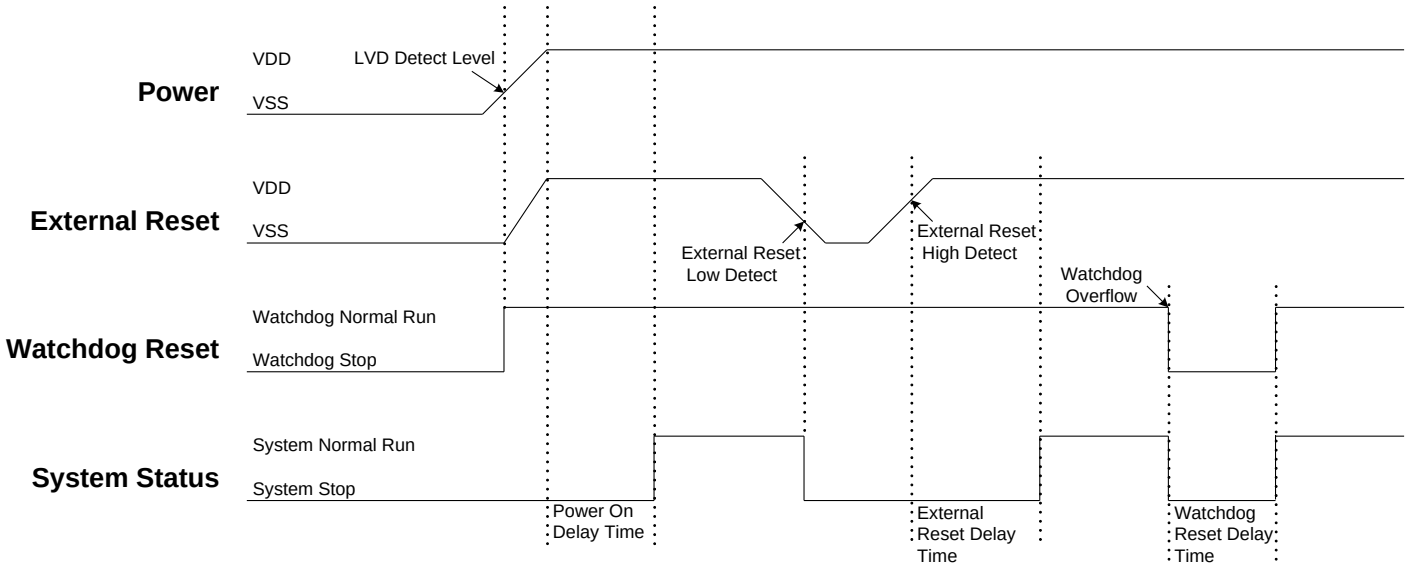
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status 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	LVD28	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

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

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

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



3.2 POWER ON RESET

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

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

3.3 WATCHDOG RESET

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

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

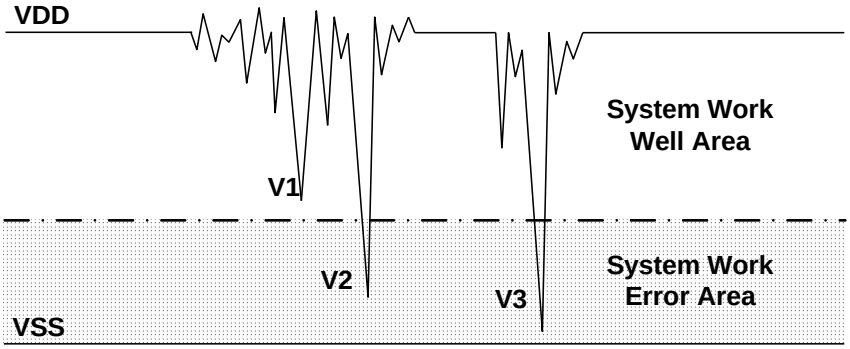
Watchdog timer application note is as following.

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

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

3.4 BROWN OUT RESET

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



Brown Out Reset Diagram

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

DC application:

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

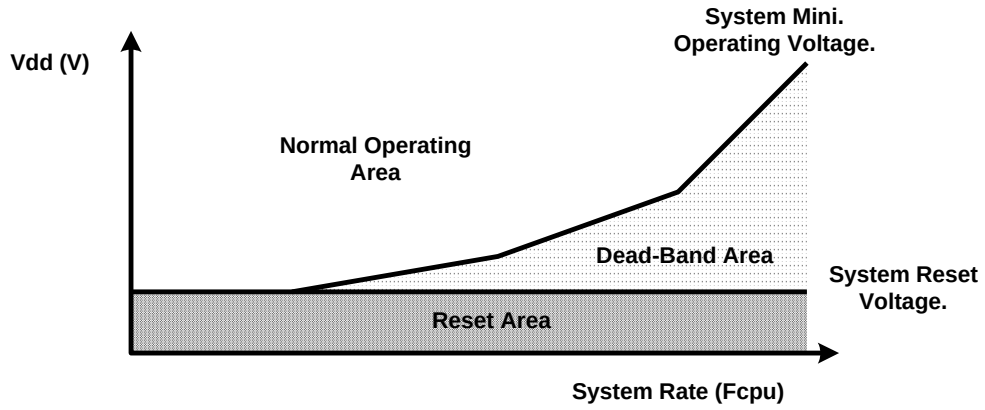
AC application:

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

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

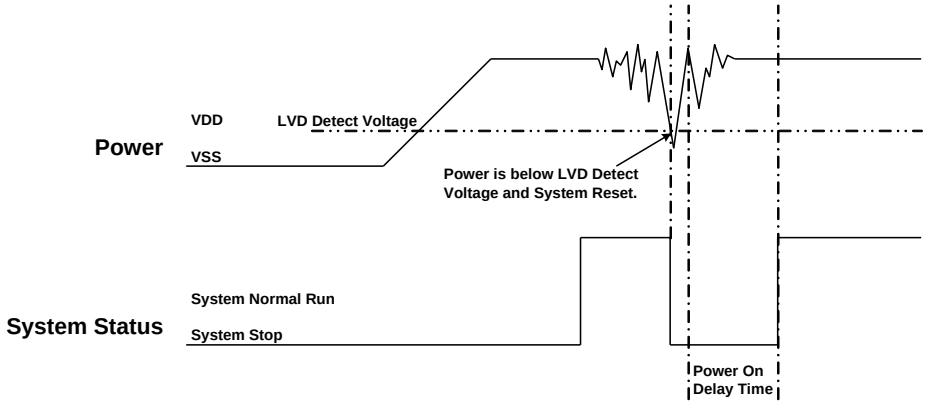
3.4.1 THE SYSTEM OPERATING VOLTAGE

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



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

3.4.2 LOW VOLTAGE DETECTOR (LVD)



The LVD (low voltage detector) is built-in Sonix 8-bit MCU to be brown out reset protection. When the VDD drops and is below LVD detect voltage, the LVD would be triggered, and the system is reset. The LVD detect level is different by each MCU. The LVD voltage level is a point of voltage and not easy to cover all dead-band range. Using LVD to improve brown out reset is depend on application requirement and environment. If the power variation is very deep, violent and trigger the LVD, the LVD can be the protection. If the power variation can touch the LVD detect level and make system work error, the LVD can't be the protection and need to other reset methods. More detail LVD information is in the electrical characteristic section.

The LVD is three levels design (1.8V/2.4V/2.8V) and controlled by LVD code option. The 1.8V LVD is always enable for power on reset and Brown Out reset. The 2.4V LVD includes LVD reset function and flag function to indicate VDD status function. The 2.8V includes flag function to indicate VDD status. LVD flag function can be an **easy low battery detector**. LVD24, LVD28 flags indicate VDD voltage level. For low battery detect application, only checking LVD24, LVD28 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	LVD28	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 **LVD28:** LVD 2.8V operating flag and only support LVD code option is LVD_H.
0 = Inactive (VDD > 2.8V).
1 = Active (VDD <= 2.8V).

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**
1.8V Reset	Available	Available	Available
2.4V Flag	-	Available	-
2.4V Reset	-	-	Available
2.8V Flag	-	-	Available

LVD_L
If VDD < 1.8V, system will be reset.
Disable LVD24 and LVD28 bit of PFLAG register

LVD_M
If VDD < 1.8V, system will be reset.
Enable LVD24 bit of PFLAG register. If VDD > 2.4V, LVD24 is “0”. If VDD ≦ 2.4V, LVD24 flag is “1”
Disable LVD28 bit of PFLAG register

LVD2_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 LVD28 bit of PFLAG register. If VDD > 2.8V, LVD28 is “0”. If VDD ≦ 2.8V, LVD28 flag is “1”

✸ ***Note:***
1. ***After any LVD reset, LVD24, LVD28 flags are cleared.***
2. ***The voltage level of LVD 2.4V or 2.8V is for design reference only. Don’t use the LVD indicator as precision VDD measurement.***

3.4.3 BROWN OUT RESET IMPROVEMENT

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

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

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

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

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

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

3.5 EXTERNAL RESET

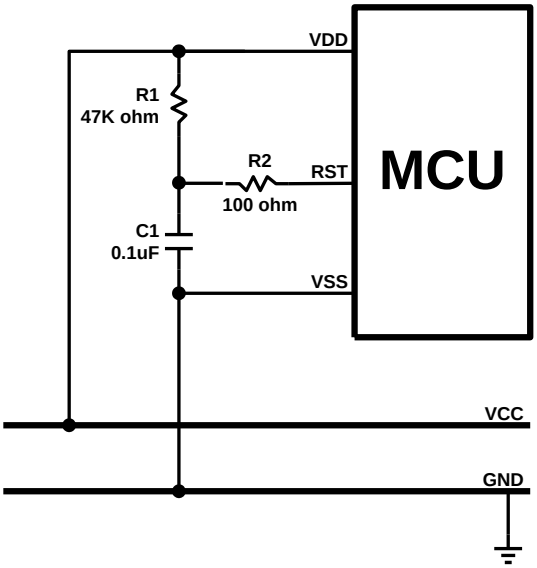
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

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

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

3.6 EXTERNAL RESET CIRCUIT

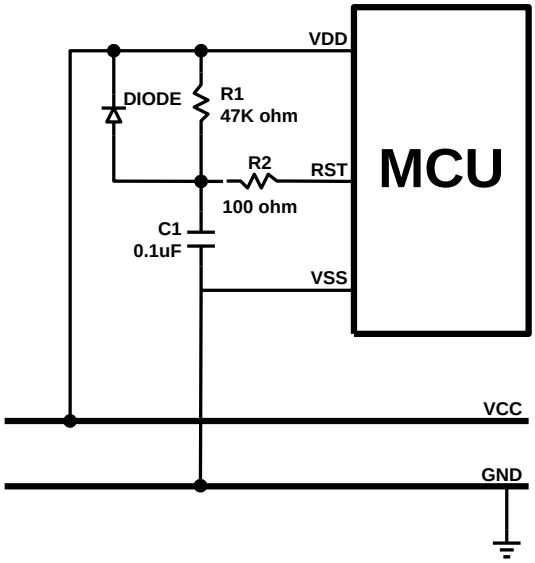
3.6.1 Simply RC Reset Circuit



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

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

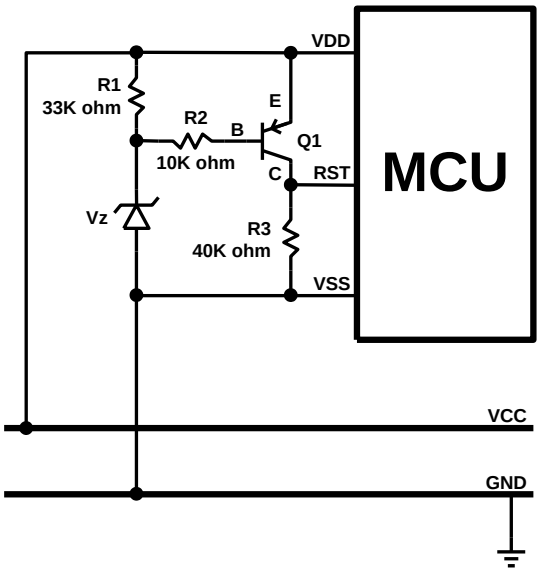
3.6.2 Diode & RC Reset Circuit



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

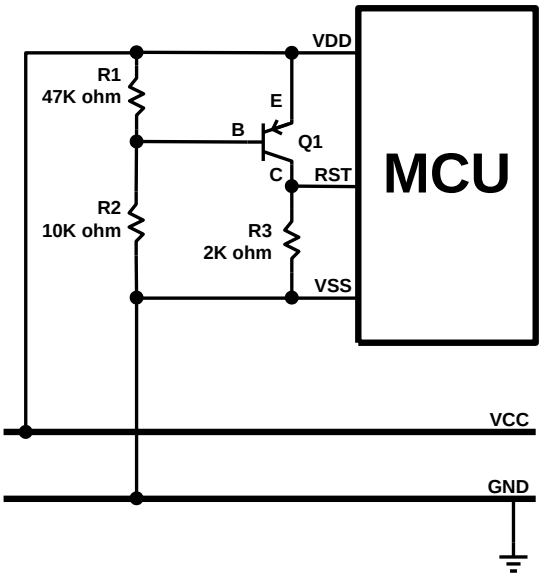
✷ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.6.3 Zener Diode Reset Circuit



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

3.6.4 Voltage Bias Reset Circuit

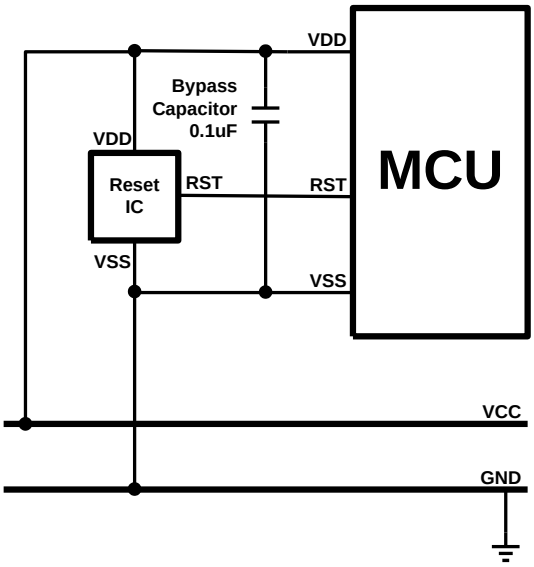


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to “0.7V x (R1 + R2) / R1”, the C terminal of the PNP transistor outputs high voltage and MCU operates normally. When VDD is below “0.7V x (R1 + R2) / R1”, the C terminal of the PNP transistor outputs low voltage and MCU is in reset mode.

Decide the reset detect voltage by R1, R2 resistances. Select the right R1, R2 value to conform the application. In the circuit diagram condition, the MCU’s reset pin level varies with VDD voltage variation, and the differential voltage is 0.7V. If the VDD drops and the voltage lower than reset pin detect level, the system would be reset. If want to make the reset active earlier, set the R2 > R1 and the cap between VDD and C terminal voltage is larger than 0.7V. The external reset circuit is with a stable current through R1 and R2. For power consumption issue application, e.g. DC power system, the current must be considered to whole system power consumption.

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

3.6.5 External Reset IC



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

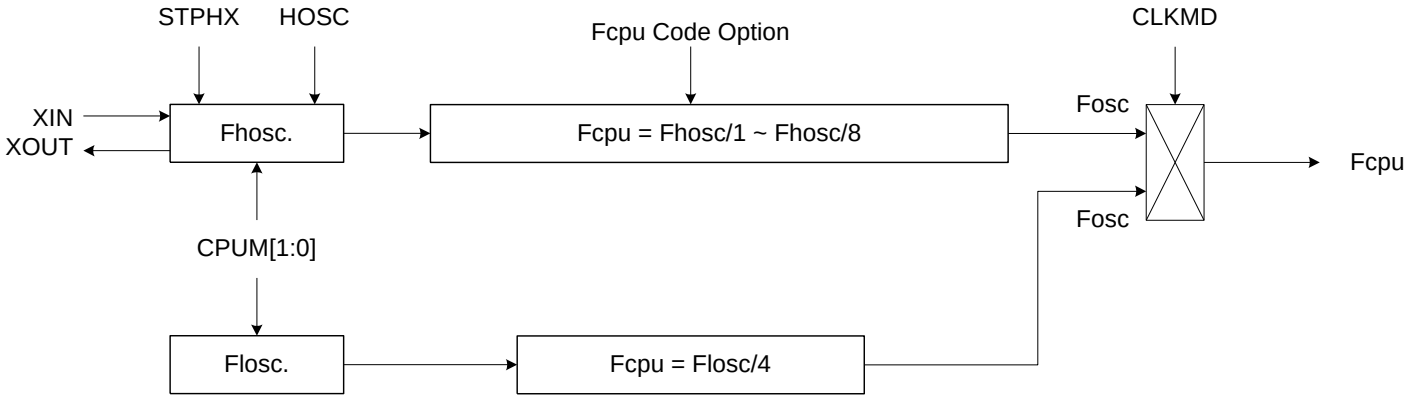
4 SYSTEM CLOCK

4.1 OVERVIEW

The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit or on-chip 8MHz high-speed RC oscillator circuit (IHRC 8MHz). The low-speed clock is generated from on-chip low-speed RC oscillator circuit (ILRC 10KHz @3V).
Both the high-speed clock and the low-speed clock can be system clock (Fosc). The system clock in slow mode is divided by 4 to be the instruction cycle (Fcpu).

- ☞ **Normal Mode (High Clock):** **Fcpu = Fhosc / N**, N = 1 ~ 8, Select N by Fcpu code option.
- ☞ **Slow Mode (Low Clock):** **Fcpu = Flosc/4.**

4.2 CLOCK BLOCK DIAGRAM



- HOSC: High_Clk code option.
- Fhosc: External high-speed clock / Internal high-speed RC clock.
- Flosc: Internal low-speed RC clock (about 10KHz@3V).
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.3 Fcpu (INSTRUCTION CYCLE)

Fcpu is instruction cycle which is divided from the system high clock source and decides the system operating rate. Fcpu rate is selected by code option from Fhosc/1~Fhosc/8 If the system high clock source is from external 16MHz crystal, and the Fcpu code option is Fhosc/4, the Fcpu frequency is 16MHz/4 = 4MHz. The code option doesn't support slow mode because the Fcpu of slow mode is fixed Flosc/4 condition.

In high noisy environment, below "Fhosc/4" of Fcpu code option is the strongly recommendation to reduce high frequency noise effect.

4.4 OSCM REGISTER

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

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

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

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

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

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

High_CLK = IHRC_8M: "STPHX=1" disables internal high speed RC type oscillator.
High_CLK = IHRC_RTC: "STPHX=1" disables internal high speed RC type oscillator and external 32768Hz crystal.
High_CLK = RC, 4M, 8M, 32K: "STPHX=1" disables external oscillator.

4.5 SYSTEM HIGH CLOCK

The system high clock is from internal 8MHz oscillator RC type or external oscillator. The high clock type is controlled by "High_Clk" code option.

- **IHRC_8M:** The system clock source is from internal high speed 8MHz RC type oscillator. In the mode, XIN and XOUT pins are bi-direction GPIO mode, and not to connect any external oscillator.
- **IHRC_RTC:** The system clock source is from internal high speed 8MHz RC type oscillator, and includes external low speed 32768Hz crystal for RTC function. The XIN and XOUT pins are defined to drive external 32768Hz crystal and not GPIO mode.
- **RC:** The system clock source is from external low cost RC type oscillator. The RC oscillator circuit only connects to XIN pin, and the XOUT pin is bi-direction GPIO mode.
- **32K X'tal:** The system clock source is from external low speed 32768Hz crystal. The option only supports 32768Hz crystal.
- **8M X'tal:** The system clock source is from external high speed crystal/resonator. The oscillator bandwidth is 8MHz~16MHz.
- **4M X'tal:** The system clock source is from external high speed crystal/resonator. The oscillator bandwidth is 1MHz~10MHz.

4.6 INTERNAL HIGH RC

The chip is built-in RC type internal high clock (8MHz) controlled by “IHRC_8M” or “IHRC_RTC” code options. In “IHRC_8M” mode, the system clock is from internal 8MHz RC type oscillator and XIN / XOUT pins are general-purpose I/O pins. In “IHRC_RTC” mode, the system clock is from internal 8MHz RC type oscillator and XIN / XOUT pins are connected with external 32768 crystal for real time clock (RTC).

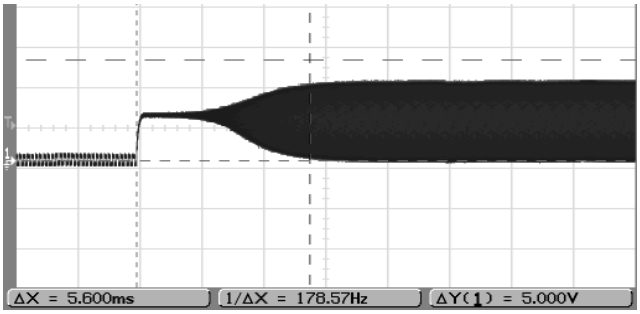
- **IHRC:** High clock is internal 8MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.
- **IHRC_RTC:** High clock is internal 8MHz oscillator RC type. XIN/XOUT pins are connected with external 32768Hz crystal/ceramic oscillator for RTC clock source.

The RTC period is 0.5 sec and RTC timer is T0. Please consult “T0 Timer” chapter to apply RTC function.

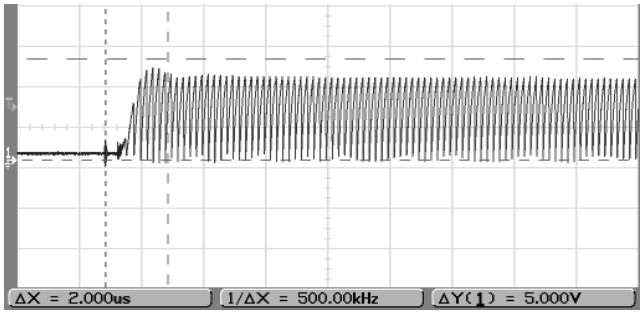
4.7 EXTERNAL HIGH CLOCK

External high clock includes three modules (Crystal/Ceramic, RC and external clock signal). The high clock oscillator module is controlled by High_Clk code option. The start up time of crystal/ceramic and RC type oscillator is different. RC type oscillator’s start-up time is very short, but the crystal’s is longer. The oscillator start-up time decides reset time length.

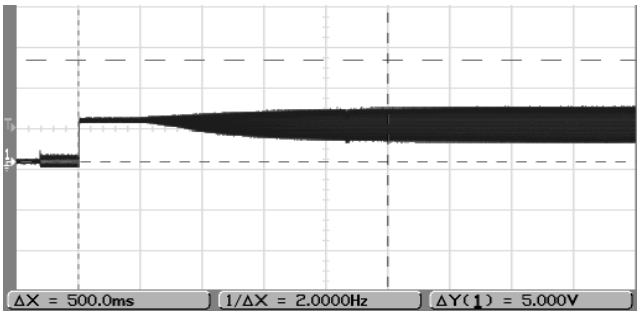
4MHz Crystal



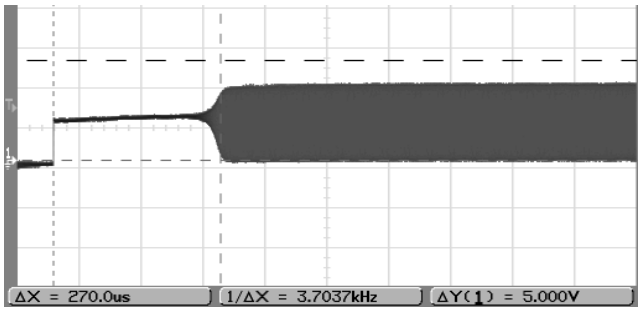
RC



32768Hz Crystal

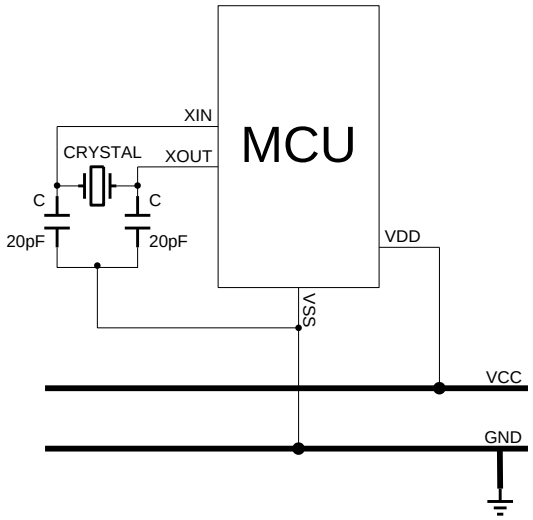


4MHz Ceramic



4.7.1 CRYSTAL/CERAMIC

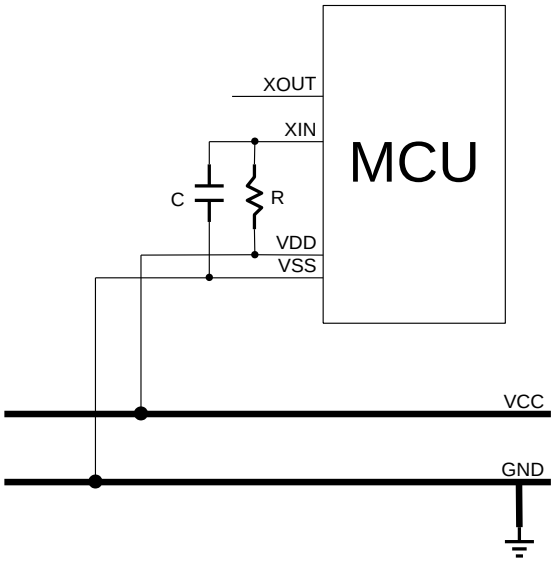
Crystal/Ceramic devices are driven by XIN, XOUT pins. For high/normal/low frequency, the driving currents are different. High_Clk code option supports different frequencies. 8M option is for high speed (ex. 8MHz). 4M option is for normal speed (ex. 4MHz). 32K option is for low speed (ex. 32768Hz).



✸ ***Note: Connect the Crystal/Ceramic and C as near as possible to the XIN/XOUT/VSS pins of micro-controller.***

4.7.2 RC

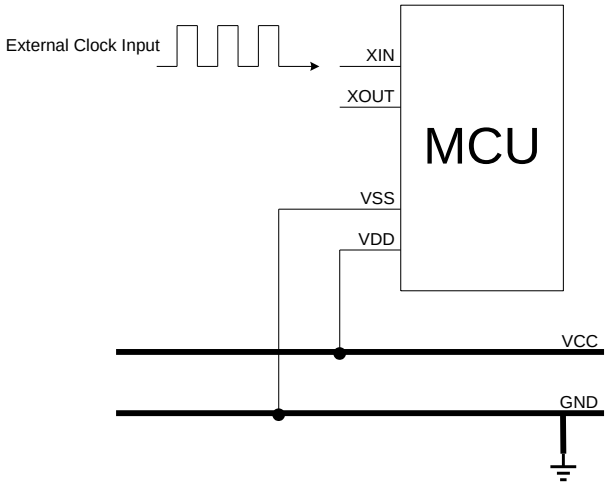
Selecting RC oscillator is by RC option of High_Clk code option. RC type oscillator's frequency is up to 10MHz. Using "R" value is to change frequency. 50P~100P is good value for "C". XOUT pin is general purpose I/O pin.



✸ ***Note: Connect the R and C as near as possible to the VDD pin of micro-controller.***

4.7.3 EXTERNAL CLOCK SIGNAL

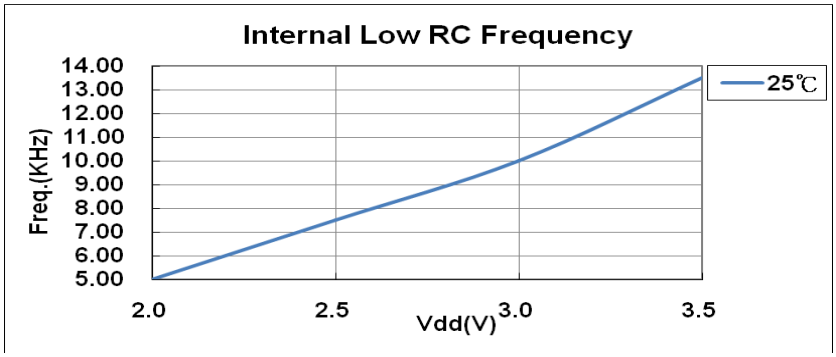
Selecting external clock signal input to be system clock is by RC option of High_Clk code option. The external clock signal is input from XIN pin. XOUT pin is general purpose I/O pin.



✸ ***Note: The GND of external oscillator circuit must be as near as possible to VSS pin of micro-controller.***

4.8 SYSTEM LOW 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 10KHz at 3V. 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.

☞ ***Flosc = Internal low RC oscillator (about 10KHz @3V).***
☞ ***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 455K mode and watchdog disable. If system is in 455K mode and watchdog disable, only 455K 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 bits of OSCM register.***

4.8.1 SYSTEM CLOCK MEASUREMENT

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

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

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

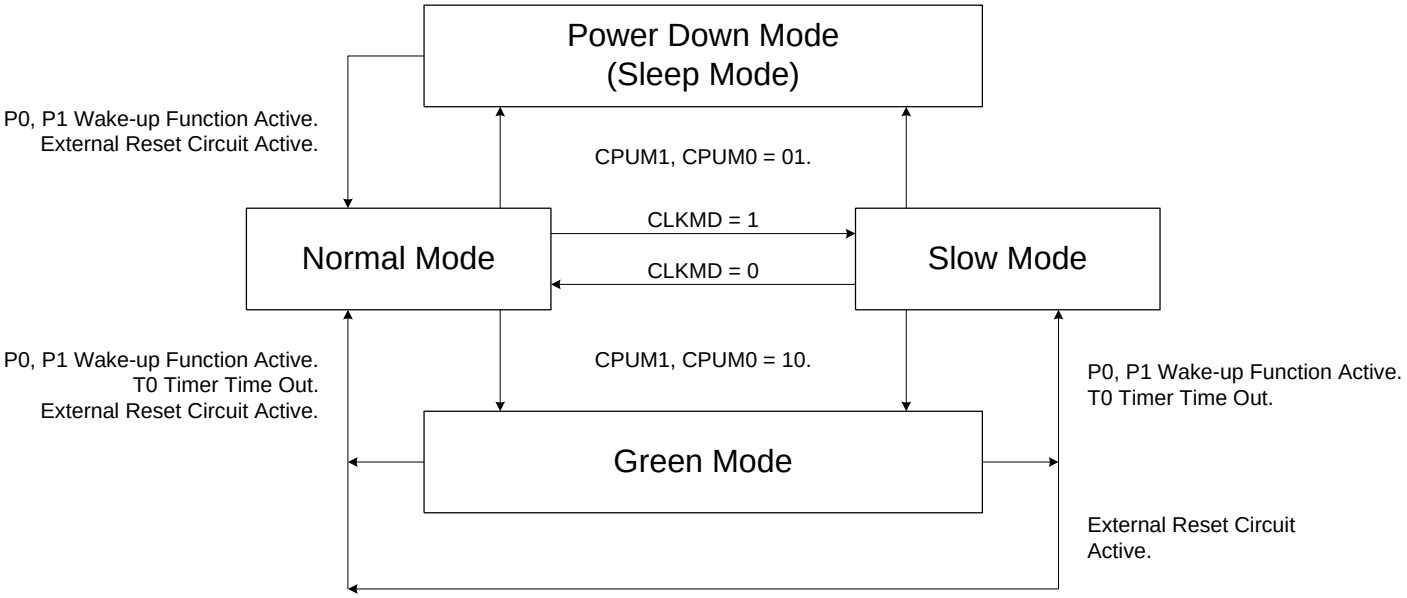
✸ ***Note: Do not measure the RC frequency directly from XIN; the probe impendence will affect the RC frequency.***

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

The chip is featured with low power consumption by switching around four different modes as following.

- High-speed mode
- Low-speed mode
- Power-down mode (Sleep mode)
- Green mode



System Mode Switching Diagram

Operating mode description

MODE	NORMAL	SLOW	GREEN	POWER DOWN (SLEEP)	REMARK
EHOSC	Running	By STPHX	By STPHX	Stop	
IHRC	Running	By STPHX	By STPHX	Stop	
ILRC	Running	Running	Running	Stop	
EHOSC with **RTC**	Running	By STPHX	Running	Stop	
IHRC with **RTC**	Running	By STPHX	Stop	Stop	
ILRC with **RTC**	Running	Running	Stop	Stop	
CPU instruction	Executing	Executing	Stop	Stop	
T0 timer	*Active	*Active	*Active	Inactive	* Active if T0ENB=1
TC1 timer	*Active	*Active	Inactive	Inactive	* Active if TC1ENB=1
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	Refer to code option description
Internal interrupt	All active	All active	T0	All inactive	
External interrupt	All active	All active	All active	All inactive	
Wakeup source	-	-	P0, P1, T0 Reset	P0, P1, Reset	

- **EHOSC:** External high clock
- **IHRC:** Internal high clock (8M RC oscillator)
- **ILRC:** Internal low clock (10K RC oscillator at 3V)

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 (10Hz @3V). The slow mode is controlled by CLKMD bit of OSCM register. When CLKMD=0, the system is in normal mode. When CLKMD=1, the system inserts into slow mode. The high speed oscillator won't be disabled automatically after switching to slow mode, and must be disabled by SPTHX bit to reduce power consumption. In slow mode, the system rate is fixed Flosc/4 (Flosc is internal low speed RC type oscillator frequency).

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

5.4 POWER DOWN MDOE

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

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

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

5.5 GREEN MODE

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

- The program stops executing, and full functions are disabled.
- Only the timer with wake-up function actives.
- The oscillator to be the system clock source keeps running, and the other oscillators operation is depend on system operation mode configuration.
- If inserting green mode from normal mode, the system insets to normal mode after wake-up.
- If inserting green mode from slow mode, the system insets to slow mode after wake-up.
- The green mode wake-up source are P0, P1 level change trigger and unique time overflow.

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

5.6 OPERATING MODE CONTROL MACRO

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
        CLR         T0C         ; Clear T0 counter.
        B0BSET      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 2048 external 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/Fosc * 2048 (sec) + high clock start-up time

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

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

The wakeup time = 1/Fosc * 2048 = 0.512 ms (Fosc = 4MHz)
The total wakeup time = 0.512 ms + oscillator start-up time

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

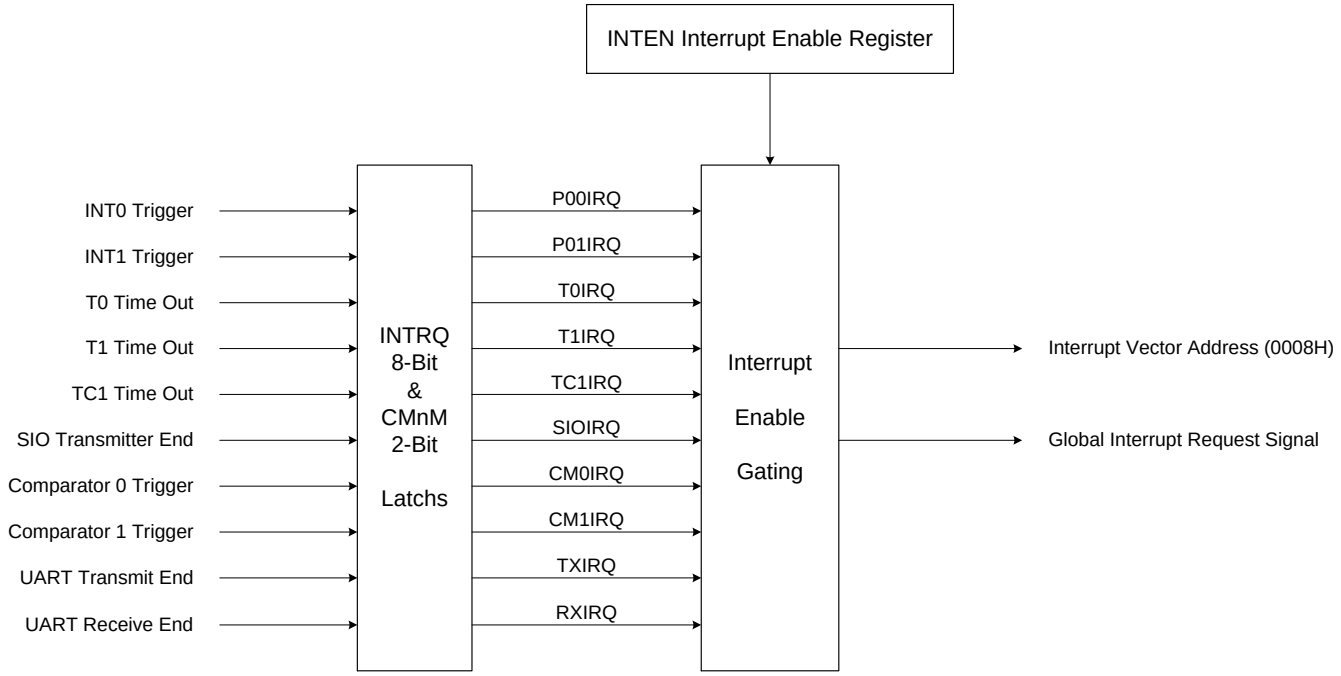
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 10 interrupt sources, including 8 internal interrupt (T0/T1/TC1/CM0/CM1/SIO/URRX/URTX) and 2 external interrupt (INT0/INT1). The external interrupt can wakeup the chip while the system is switched from power down mode to high-speed normal mode, and interrupt request is latched until return to normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to “0” for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to “1” to accept the next interrupts’ request. Most of the interrupt request signals are stored in INTRQ register, but comparator interrupt request flags are stored in CMnM registers.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

INTEN is the interrupt request control register including 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	SIOIEN	TC1IEN	T1IEN	T0IEN	RXIEN	TXIEN	P01IEN	P00IEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **P00IEN:** External P0.0 interrupt (INT0) control bit.
0 = Disable INT0 interrupt function.
1 = Enable INT0 interrupt function.

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

Bit 2 **TXIEN:** UART transmit interrupt control bit.
0 = Disable UART transmit interrupt function.
1 = Enable UART transmit interrupt function.

Bit 3 **RXIEN:** UART receive interrupt control bit.
0 = Disable UART receive interrupt function.
1 = Enable UART receive interrupt function.

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

Bit 5 **T1IEN:** T1 timer interrupt control bit.
0 = Disable T1 interrupt function.
1 = Enable T1 interrupt function.

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

Bit 7 **SIOIEN:** SIO interrupt control bit.
0 = Disable SIO interrupt function.
1 = Enable SIO interrupt function.

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

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

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	SIOIRQ	TC1IRQ	T1IRQ	T0IRQ	RXIRQ	TXIRQ	P01IRQ	P00IRQ
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **P00IRQ:** External P0.0 interrupt (INT0) request flag.
0 = None INT0 interrupt request.
1 = INT0 interrupt request.

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

Bit 2 **TXIRQ:** UART transmit interrupt request flag.
0 = None UART transmit interrupt request.
1 = UART transmit interrupt request.

Bit 3 **RXIRQ:** UART receive interrupt request flag.
0 = None UART receive interrupt request.
1 = UART receive interrupt request.

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

Bit 5 **T1IRQ:** T1 timer interrupt request flag.
0 = None T1 interrupt request.
1 = T1 interrupt request.

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

Bit 7 **SIOIRQ:** SIO interrupt request flag.
0 = None SIO interrupt request.
1 = SIO interrupt request.

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

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

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)

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

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

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

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

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

```
        MOV     A, #98H
        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 INT1 (P0.1) INTERRUPT OPERATION

When the INT1 trigger occurs, the P01IRQ will be set to “1” no matter the P01IEN is enable or disable. If the P01IEN = 1 and the trigger event P01IRQ is also set to be “1”. As the result, the system will execute the interrupt vector (ORG 8). If the P01IEN = 0 and the trigger event P01IRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the P01IRQ is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

✸ ***Note: The interrupt trigger direction of P0.1 is falling edge.***

➢ **Example: INT1 interrupt request setup.**

```
        B0BSET      FP01IEN     ; Enable INT1 interrupt service
        B0BCLR      FP01IRQ     ; Clear INT1 interrupt request flag
        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: INT1 interrupt service routine.**

```
        ORG         8           ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:
        …                       ; Push routine to save ACC and PFLAG to buffers.
        B0BTS1      FP01IRQ     ; Check P01IRQ
        JMP         EXIT_INT    ; P01IRQ = 0, exit interrupt vector
        B0BCLR      FP01IRQ     ; Reset P01IRQ
        …                       ; INT1 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.
        RETI                    ; Exit interrupt vector
```

6.8 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 as no RTC function.**

```
        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
```

✸ ***Note:***
1. ***In RTC mode, clear T0IRQ must be after 1/2 RTC clock source (32768Hz), or the RTC interval time is error. The delay is about 16us and use T0 interrupt service routine executing time to be the 16us delay time.***
2. ***In RTC mode, don't reset T0C in interrupt service routine.***

➢ **Example: T0 interrupt service routine with RTC function.**

```
                ORG         8               ; Interrupt vector
                JMP         INT_SERVICE
INT_SERVICE:
                …                           ; Push routine to save ACC and PFLAG to buffers.
> 16us
                B0BTS1      FT0IRQ          ; Check T0IRQ
                JMP         EXIT_INT        ; T0IRQ = 0, exit interrupt vector
                …                           ; T0 interrupt service routine
                …                           ; The time must be longer than 16us.
                B0BCLR      FT0IRQ          ; Reset T0IRQ
                …
EXIT_INT:
                …                           ; Pop routine to load ACC and PFLAG from buffers.

                RETI                        ; Exit interrupt vector
```

6.9 T1 INTERRUPT OPERATION

When the T1C (T1CH, T1CL) counter occurs overflow, the T1IRQ will be set to “1” however the T1IEN is enable or disable. If the T1IEN = 1, the trigger event will make the T1IRQ to be “1” and the system enter interrupt vector. If the T1IEN = 0, the trigger event will make the T1IRQ to be “1” but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

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

```
        B0BCLR      FT1IEN          ; Disable T1 interrupt service
        B0BCLR      FT1ENB          ; Disable T1 timer
        MOV         A, #20H         ;
        B0MOV       T1M, A          ; Set T1 clock = Fcpu / 64 and falling edge trigger.
        CLR         T1C

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

        B0BSET      FGIE            ; Enable GIE
```

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

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

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

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

        B0BCLR      FT1IRQ          ; Reset T1IRQ
        B0MOV       A, T1C
        B0MOV       T1CBUF, A       ; Save pulse width.
        CLR         T1C
        …                           ; T1 interrupt service routine
        …
EXIT_INT:
        POP                         ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

6.10 TC1 INTERRUPT OPERATION

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

➢ **Example: TC1 interrupt request setup.**

```
        B0BCLR      FTC1IEN     ; Disable TC1 interrupt service
        B0BCLR      FTC1ENB     ; Disable TC1 timer
        MOV         A, #20H     ;
        B0MOV       TC1M, A     ; Set TC1 clock = Fcpu / 64
        MOV         A, #74H     ; Set TC1C initial value = 74H
        B0MOV       TC1C, A     ; Set TC1 interval = 10 ms

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

        B0BSET      FGIE        ; Enable GIE
```

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

```
        ORG         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
```

6.11 COMPARATOR INTERRUPT OPERATION (CMP0, CMP1)

Sonix provides 2 sets 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.

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMP0M	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CMS1	CMS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

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

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

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

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

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

➢ **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
```

6.12 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
```

6.13 UART INTERRUPT OPERATION

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

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

```
        B0BSET      FRXIEN      ; Enable UART receive interrupt service
        B0BCLR      FRXIRQ      ; Clear UART receive interrupt request flag

        B0BSET      FTXIEN      ; Enable UART transmit interrupt service
        B0BCLR      FTXIRQ      ; Clear UART transmit interrupt request flag
        B0BSET      FGIE        ; Enable GIE
```

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

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

        B0BTS1      FRXIRQ      ; Check RXIRQ
        JMP         EXIT_INT    ; RXIRQ = 0, exit interrupt vector

        B0BCLR      FRXIRQ      ; Reset RXIRQ
        …                       ; UART receive interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.14 MULTI-INTERRUPT OPERATION

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

Interrupt Name	*Trigger Event Description*
P00IRQ	P0.0 trigger controlled by PEDGE
P01IRQ	P0.1 falling edge trigger.
T0IRQ	T0C overflow
T1IRQ	T1C overflow
TC1IRQ	TC1C overflow
SIOIRQ	SIO transmitter successfully.
CM0IRQ	Comparator 0 output level transition.
CM1IRQ	Comparator 1 output level transition.
RXIRQ/TXIRQ	UART transmitter successfully.

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

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

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

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

INTP00CHK:                      ; Check INT0 interrupt request
        B0BTS1  FP00IEN         ; Check P00IEN
        JMP     INTT0CHK        ; Jump check to next interrupt
        B0BTS0  FP00IRQ         ; Check P00IRQ
        JMP     INTP00
INTT0CHK:                       ; Check T0 interrupt request
        B0BTS1  FT0IEN          ; Check T0IEN
        JMP     INTTC1CHK       ; Jump check to next interrupt
        B0BTS0  FT0IRQ          ; Check T0IRQ
        JMP     INTT0           ; Jump to T0 interrupt service routine
INTTC1CHK:                      ; Check TC1 interrupt request
        B0BTS1  FTC1IEN         ; Check TC1IEN
        JMP     INTSIOHK        ; Jump check to next interrupt
        B0BTS0  FTC1IRQ         ; Check TC1IRQ
        JMP     INTTC1          ; Jump to TC1 interrupt service routine
INTSIOHK:                       ; Check SIO interrupt request
        B0BTS1  FSIOIEN         ; Check SIOIEN
        JMP     …               ; Jump check to next interrupt
        B0BTS0  FSIOIRQ         ; Check SIOIRQ
        JMP     INTSIO          ; Jump to SIO interrupt service routine
        …
        …
INT_EXIT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

7 I/O PORT

7.1 OVERVIEW

The micro-controller builds in 44 pin I/O. Most of the I/O pins are mixed with analog pins and special function pins. The I/O shared pin list is as following.

I/O Pin		Shared Pin		Shared Pin Control Condition
Name	Type	Name	Type	
P0.0	I/O	INT0	DC	P00IEN=1
P0.1	IO	INT1	DC	P01IEN=1
P0.2	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P0.3	I/O	XIN	AC	High_CLK code option = IHRC_RTC, RC, 32K, 4M, 8M
P0.4	I/O	XOUT	AC	High_CLK code option = IHRC_RTC, 32K, 4M, 8M
P2.2	I/O	CM0N	AC	CM0EN=1
P2.3	I/O	CM0P	AC	CM0EN=1, CM0REF=0
P2.4	I/O	CM0O	AC	CM0EN=1, CM0OEN=1
P2.5	I/O	CM1N	AC	CM1EN=1
P2.6	I/O	CM1P	AC	CM1EN=1, CM1REF=0
P2.7	I/O	CM1O	AC	CM1EN=1, CM1OEN=1
P3.2	I/O	URX	DC	URXEN=1.
P3.3	I/O	UTX	DC	UTXEN=1.
P5.0	I/O	SCK	DC	SENB=1.
P5.1	I/O	SI	DC	SENB=1.
P5.2	I/O	SO	DC	SENB=1.
P5.3	I/O	BZ1/PWM1	DC	TC1ENB=1, TC1OUT=1 or PWM1OUT=1
P5.4	I/O	IROUT	DC	IREN=1

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

7.2 I/O PORT MODE

The port direction is programmed by PnM register. 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	P07M	P06M	P05M	P04M	P03M	-	P01M	P00M
Read/Write	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
After reset	0	0	0	0	0	-	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	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

0C2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2M	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M
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

0C3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3M	-	-	-	-	P33M	P32M	P31M	P30M
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

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

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

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

✸ ***Note:***
1. ***Users can program them by bit control instructions (B0BSET, B0BCLR).***
2. ***P0.2 is input only pin, and the P0M.2 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.2       ; Set P1.2 to be input mode.

B0BSET  P1M.2       ; Set P1.2 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	P07R	P06R	P05R	P04R	P03R	-	P01R	P00R
Read/Write	W	W	W	W	W	-	W	W
After reset	0	0	0	0	0	-	0	0

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

0E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2UR	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0E3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3UR	-	-	-	-	P33R	P32R	P31R	P30R
Read/Write	-	-	-	-	W	W	W	W
After reset	-	-	-	-	0	0	0	0

0E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

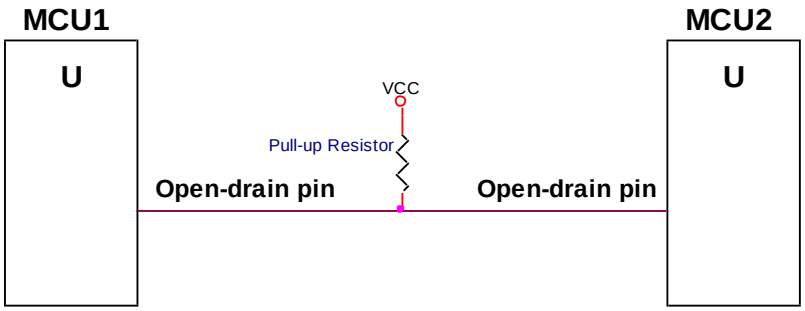
✸ ***Note: P0.2 is input only pin and without pull-up resister. The P0UR.2 keeps "1".***

➢ **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 OPEN-DRAIN REGISTER

P1.0/P1.1/P3.2/P3.3/P5.0/P5.1/P5.2 built in open-drain function. P1.0/P1.1/P3.2/P3.3/P5.0/P5.1/P5.2 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	P33OC	P32OC	-	P11OC	P10OC
Read/Write	W	W	W	W	W	-	W	W
After reset	0	0	0	0	0	-	0	0

Bit 0 **P10OC:** P1.0 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 1 **P11OC:** P1.1 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 3 **P32OC:** P3.2 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 4 **P33OC:** P3.3 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 5 **P50OC:** P5.0 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 6 **P51OC:** P5.1 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 7 **P52OC:** P5.2 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

➢ **Example: Enable P1.0 to open-drain mode and output high.**

```
B0BSET      P1.0        ; Set P1.0 buffer high.

B0BSET      P10M        ; Enable P1.0 output mode.
MOV         A, #01H     ; Enable P1.0 open-drain function.
B0MOV       P1OC, A
```

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

➢ **Example: Disable P1.0 to open-drain mode and output low.**

```
MOV         A, #0       ; Disable P1.0 open-drain function.
B0MOV       P1OC, A
```

✸ ***Note: After disable P1 open-drain function, P1 mode returns to last I/O mode.***

7.5 I/O PORT DATA REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	P07	P06	P05	P04	P03	P02	P01	P00
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	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

0D2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2	P27	P26	P25	P24	P23	P22	P21	P20
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0D3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3	-	-	-	-	P33	P32	P31	P30
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

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

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

✸ ***Note: The P02 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  P1.3    ; Set P1.3 and P5.4 to be "1".
B0BSET  P5.4

B0BCLR  P1.3    ; Set P1.3 and P5.4 to be "0".
B0BCLR  P5.4
```

8 TIMERS

8.1 WATCHDOG TIMER

The watchdog timer (WDT) is a binary up counter designed for monitoring program execution. If the program goes into the unknown status by noise interference, WDT overflow signal raises and resets MCU. Watchdog clock controlled by code option and the clock source is internal low-speed oscillator (10KHz @3V).

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

VDD	*Internal Low RC Freq.*	*Watchdog Overflow Time*
3V	10KHz	819.2ms

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.**

```
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.**

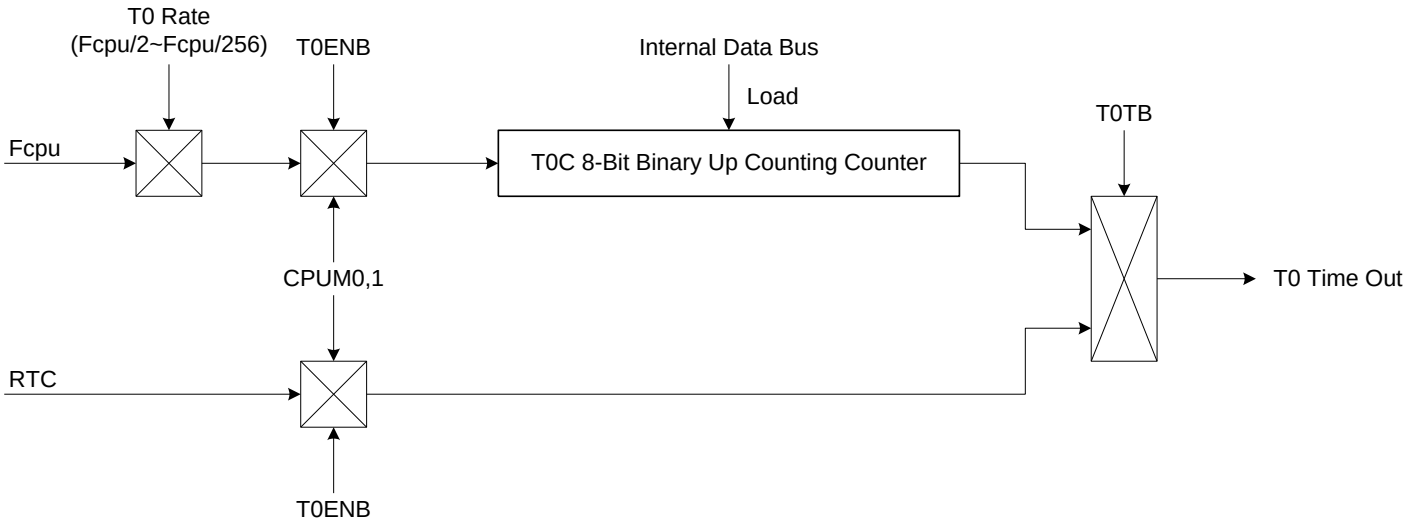
```
Main:
        …                           ; Check I/O.
        …                           ; Check RAM
Err:    JMP $                       ; I/O or RAM error. Program jump here and don't
                                    ; clear watchdog. Wait watchdog timer overflow to reset IC.

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 TIMER 0 (T0)

8.2.1 OVERVIEW

The T0 is an 8-bit binary up timer and event counter. If T0 timer occurs an overflow (from FFH to 00H), it will continue counting and issue a time-out signal to trigger T0 interrupt to request interrupt service.
The main purposes of the T0 timer is as following.

- **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **RTC timer:** Generates interrupts at real time intervals based on the selected clock source. **RTC function is only available in High_Clk code option = "IHRC_RTC".**
- **Green mode wakeup function:** T0 can be green mode wake-up time as T0ENB = 1. System will be wake-up by T0 time out.



✸ ***Note:1. In RTC mode, clear T0IRQ must be after 1/2 RTC clock source (32768Hz), or the RTC interval time is error. The delay is about 16us and use T0 interrupt service routine executing time to be the 16us delay time.***
2. In RTC mode, the T0 interval time is fixed at 0.5 sec and T0C is 256 counts.

8.2.2 T0M MODE REGISTER

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

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

Bit [6:4] **T0RATE[2:0]:** T0 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
…
110 = fcpu/4.
111 = fcpu/2.

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

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

8.2.3 T0C COUNTING REGISTER

T0C is an 8-bit counter register for T0 interval time control.

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 * input clock)

- **Example: To set 10ms interval time for T0 interrupt. High clock is external 4MHz. Fcpu=Fosc/4. Select T0RATE=010 (Fcpu/64).**

*T0C initial value = 256 - (T0 interrupt interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 64)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 64)*
= 100
= 64H

The basic timer table interval time of T0.

T0RATE	T0CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fcpu/256	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.125 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

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

8.2.4 T0 TIMER OPERATION SEQUENCE

T0 timer operation sequence of setup T0 timer is as following.

- **Stop T0 timer counting, disable T0 interrupt function and clear T0 interrupt request flag.**

```
B0BCLR      FT0ENB          ; T0 timer.
B0BCLR      FT0IEN          ; T0 interrupt function is disabled.
B0BCLR      FT0IRQ          ; T0 interrupt request flag is cleared.
```

- **Set T0 timer rate.**

```
MOV         A, #0xxx0000b   ;The T0 rate control bits exist in bit4~bit6 of T0M. The
                            ; value is from x000xxxxb~x111xxxxb.
B0MOV       T0M,A           ; T0 timer is disabled.
```

- **Set T0 clock source from Fcpu or RTC.**

```
B0BCLR      FT0TB           ; Select T0 Fcpu clock source.
```

or

```
B0BSET      FT0TB           ; Select T0 RTC clock source.
```

- **Set T0 interrupt interval time.**

```
MOV         A,#7FH
B0MOV       T0C,A           ; Set T0C value.
```

- **Set T0 timer function mode.**

```
B0BSET      FT0IEN          ; Enable T0 interrupt function.
```

- **Enable T0 timer.**

```
B0BSET      FT0ENB          ; Enable T0 timer.
```

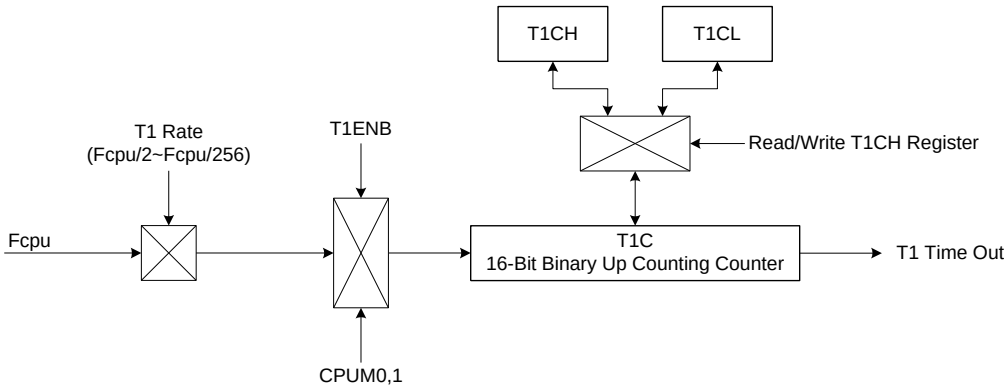
8.3 TIMER 1 (T1)

8.3.1 OVERVIEW

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

The main purposes of the T1 timer is as following.

☞ **16-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.



8.3.2 T1M MODE REGISTER

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1RATE2	T1RATE1	T1RATE0	-	-	-	-
Read/Write	R/W	R/W	R/W	R/W	-	-	-	-
After reset	0	0	0	0	-	-	-	-

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

Bit [6:4] **T1RATE[2:0]:** T1 timer internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
…
110 = fcpu/4.
111 = fcpu/2.

8.3.3 T1CH, T1CL COUNTING REGISTER

T1C is an 16-bit counter register for T1 interval time control. T1CH is high byte of T1C. T1CL is low byte of T1C.

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CL	T1CL7	T1CL6	T1CL5	T1CL4	T1CL3	T1CL2	T1CL1	T1CL0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CH	T1CH7	T1CH6	T1CH5	T1CH4	T1CH3	T1CH2	T1CH1	T1CH0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of T1C [T1CH, T1CL] initial value is as following.

T1C initial value = 65536 - (T1 interrupt interval time * input clock)

- **Example: To set 10ms interval time for T1 interrupt. High clock is external 4MHz. Fcpu=Fosc/4. Select T1RATE=010 (Fcpu/64).**

*T1C initial value = 65536 - (T1 interrupt interval time * input clock)*
*= 65536 - (10ms * 4MHz / 4 / 64)*
*= 65536 - (10^{-2} * 4 * 10^{6} / 4 / 64)*
= 65380
= FF64H ;T1CH=0xFF, T1CL=0x64

The basic timer table interval time of T1.

T1RATE	T1CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fcpu/256	16.777 s	256 us	2048 s	31250 us
001	Fcpu/128	8.388 s	128 us	1024 s	15625 us
010	Fcpu/64	4.194 s	64 us	512 s	7812.5 us
011	Fcpu/32	2.097 s	32 us	256 s	3906.25 us
100	Fcpu/16	1.048 s	16 us	128 s	1953.125 us
101	Fcpu/8	524.288 ms	8 us	64 s	976.563 us
110	Fcpu/4	262.144 ms	4 us	31 s	488.281 us
111	Fcpu/2	131.072 ms	2 us	16 s	244.141 us

The T1 16-bit counter buffer is T1CH and T1CL combination. System provides a routine to process the 16-bit data buffer under 8-bit situation to make high/low bytes of 16-bit data processed at the same time. T1CH register is the key to control the T1 16-bit counter buffer processed through T1CH, T1CL buffers. Export T1C 16-bit buffer data to T1CH, T1CL registers is by reading T1CH register. Import T1C 16-bit buffer data from T1CH, T1CL registers is by writing T1CH register after setting T1CL register data.

➢ **Example: Reading T1C 16-bit buffer data is controlled by reading T1CH register. Read T1CH register data and low byte data of T1C 16-bit buffer exporting to T1CL register at the same time.**

```
B0MOV       A, T1CH         ; Read T1CH first and T1C low byte data exported to T1CL.
…
B0MOV       A, T1CL         ; Read T1CL data from buffer.
…
```

✸ ***Note: Read T1CH first when reading T1C 16-bit buffer.***

➢ **Example: Writing and setting T1C 16-bit buffer data is controlled by writing data into T1CH register. When writing T1CH register data, T1CH, T1CL data are imported into T1C 16-bit buffer at the same time. Setting T1CL register data first is necessary, or the T1C low byte data would be error.**

```
…
B0MOV       T1CL, A         ; Write T1CL data into T1CL buffer first.
…
B0MOV       T1CH, A         ; Write T1CH data and T1CH, T1CL are imported to T1C
                            ; 16-bit buffer.
```

✸ ***Note: Write T1CL first when writing T1C 16-bit buffer.***

➢ **Example: Write T1CL is by write T1CH. Only executing "CLR T1CL" and no do any T1CH writing operation can't clear T1CL buffer. Clear T1CL must be using "MOV" instruction as following.**

```
MOV         A, #0
B0MOV       T1CL, A         ; Write "0" into T1CL to clear T1CL.
…
B0MOV       T1CH, A         ; Write T1CH data and T1CH, T1CL are imported to T1C
                            ; 16-bit buffer.
```

✸ ***Note: Don't clear T1CL by "CLR" instruction.***

8.3.4 T1 TIMER OPERATION SEQUENCE

T1 timer operation sequence of setup T1 timer is as following.

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

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

☞ **Set T1 timer rate.**

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

☞ **Set T1 interrupt interval time.**

```
MOV         A,#7FH
B0MOV       T1CL,A          ; Set T1CL value.
MOV         A,#7FH
B0MOV       T1CH,A          ; Set T1CH value.
```

☞ **Set T1 timer function mode.**

```
B0BSET      FT1IEN          ; Enable T1 interrupt function.
```

☞ **Enable T1 timer.**

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

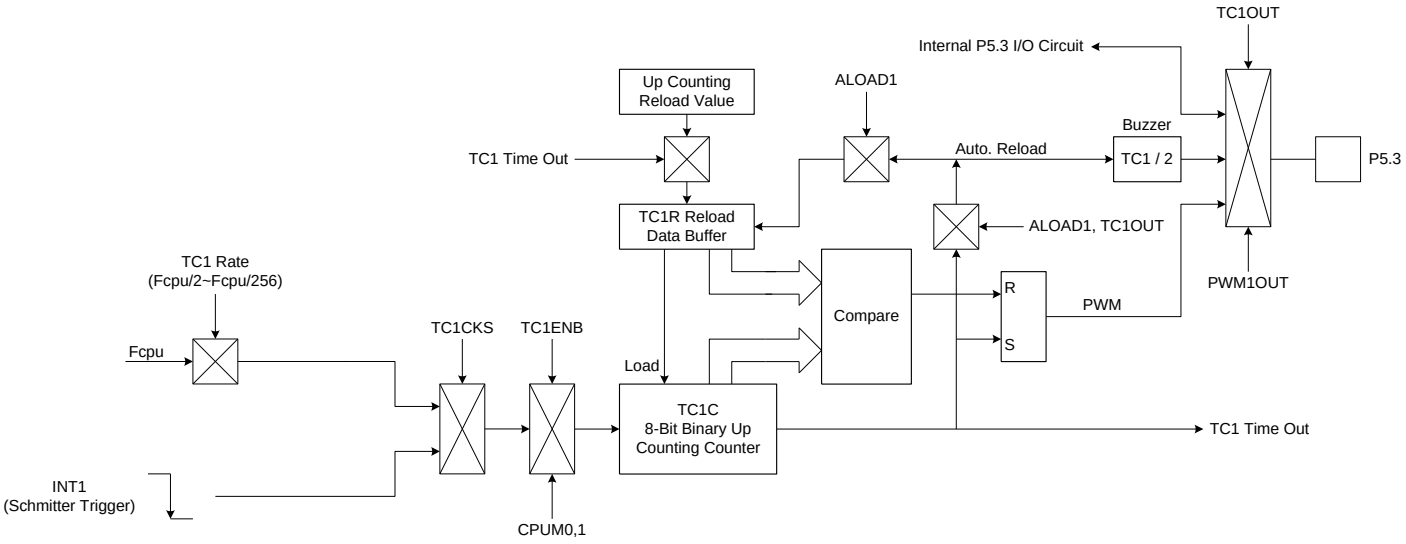
8.4 TIMER/COUNTER 0 (TC1)

8.4.1 OVERVIEW

The TC1 is an 8-bit binary up counting timer with double buffers. TC1 has two clock sources including internal clock and external clock for counting a precision time. The internal clock source is from Fcpu. The external clock is INT1 from P0.1 pin (Falling edge trigger). Using TC1M register selects TC1C's clock source from internal or external. If TC1 timer occurs an overflow, it will continue counting and issue a time-out signal to trigger TC1 interrupt to request interrupt service. TC1 overflow time is 0xFF to 0X00 normally. Under PWM mode, TC1 overflow is decided by PWM cycle controlled by ALOAD1 and TC1OUT bits.

The main purposes of the TC1 timer is as following.

- **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **External event counter:** Counts system "events" based on falling edge detection of external clock signals at the INT1 input pin.
- **Buzzer output**
- **PWM output**



8.4.2 TC1M MODE REGISTER

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1M	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **PWM1OUT:** PWM output control bit.
0 = Disable PWM output.
1 = Enable PWM output. PWM duty controlled by TC1OUT, ALOAD1 bits.

Bit 1 **TC1OUT:** TC1 time out toggle signal output control bit. **Only valid when PWM1OUT = 0.**
0 = Disable, P5.3 is I/O function.
1 = Enable, P5.3 is output TC1OUT signal.

Bit 2 **ALOAD1:** Auto-reload control bit. **Only valid when PWM1OUT = 0.**
0 = Disable TC1 auto-reload function.
1 = Enable TC1 auto-reload function.

Bit 3 **TC1CKS:** TC1 clock source select bit.
0 = Internal clock (Fcpu or Fosc).
1 = External clock from P0.1/INT1 pin.

Bit [6:4] **TC1RATE[2:0]:** TC1 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
…
110 = fcpu/4.
111 = fcpu/2.

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

✸ ***Note: When TC1CKS=1, TC1 became an external event counter and TC1RATE is useless. No more P0.1 interrupt request will be raised. (P0.1IRQ will be always 0).***

8.4.3 TC1C COUNTING REGISTER

TC1C is an 8-bit counter register for TC1 interval time control.

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 = N - (TC1 interrupt interval time * input clock)

N is TC1 overflow boundary number. TC1 timer overflow time has six types (TC1 timer, TC1 event counter, TC1 Fcpu clock source, TC1 Fosc clock source, PWM mode and no PWM mode). These parameters decide TC1 overflow time and valid value as follow table.

TC1CKS	PWM1	ALOAD1	TC1OUT	N	TC1C valid value	TC1C value binary type	Remark
0	0	x	x	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
	1	0	0	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
	1	0	1	64	0x00~0x3F	xx000000b~xx111111b	Overflow per 64 count
	1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b	Overflow per 32 count
	1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b	Overflow per 16 count
1	-	-	-	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count

➢ **Example: To set 10ms interval time for TC1 interrupt. TC1 clock source is Fcpu (TC1KS=0) and no PWM output (PWM1=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC1RATE=010 (Fcpu/64).**

*TC1C initial value = N - (TC1 interrupt interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 64)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 64)*
= 100
= 64H

The basic timer table interval time of TC1.

TC1RATE	TC1CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fcpu/256	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.125 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

8.4.4 TC1R AUTO-LOAD REGISTER

TC1 timer is with auto-load function controlled by ALOAD1 bit of TC1M. When TC1C overflow occurring, TC1R value will load to TC1C by system. It is easy to generate an accurate time, and users don't reset TC1C during interrupt service routine.

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 TC1 interval time error and glitch in PWM and Buzzer output.

✸ ***Note: Under PWM mode, auto-load is enabled automatically. The ALOAD1 bit is selecting overflow boundary.***

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 = N - (TC1 interrupt interval time * input clock)

N is TC1 overflow boundary number. TC1 timer overflow time has six types (TC1 timer, TC1 event counter, TC1 Fcpu clock source, TC1 Fosc clock source, PWM mode and no PWM mode). These parameters decide TC1 overflow time and valid value as follow table.

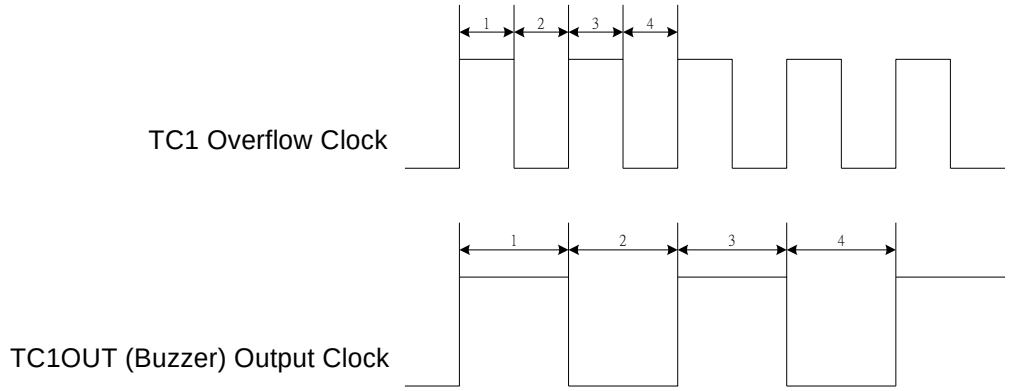
TC1CKS	PWM1	ALOAD1	TC1OUT	N	TC1R valid value	TC1R value binary type
0	0	x	x	256	0x00~0xFF	00000000b~11111111b
	1	0	0	256	0x00~0xFF	00000000b~11111111b
	1	0	1	64	0x00~0x3F	xx000000b~xx111111b
	1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
	1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
1	-	-	-	256	0x00~0xFF	00000000b~11111111b

➢ **Example: To set 10ms interval time for TC1 interrupt. TC1 clock source is Fcpu (TC1KS=0) and no PWM output (PWM1=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC1RATE=010 (Fcpu/64).**

*TC1R initial value = N - (TC1 interrupt interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 64)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 64)*
= 100
= 64H

8.4.5 TC1 CLOCK FREQUENCY OUTPUT (BUZZER)

Buzzer output (TC1OUT) is from TC1 timer/counter frequency output function. By setting the TC1 clock frequency, the clock signal is output to P5.3 and the P5.3 general purpose I/O function is auto-disable. The TC1OUT frequency is divided by 2 from TC1 interval time. TC1OUT frequency is 1/2 TC1 frequency. The TC1 clock has many combinations and easily to make difference frequency. The TC1OUT frequency waveform is as following.



➢ **Example: Setup TC1OUT output from TC1 to TC1OUT (P5.3). The external high-speed clock is 4MHz. Fcpu = Fosc/4 = 1MIPS. The TC1OUT frequency is 1KHz. Because the TC1OUT signal is divided by 2, set the TC1 clock to 2KHz. The TC1 clock source is from external oscillator clock. TC1 rate is Fcpu/4. The TC1RATE2~TC1RATE1 = 110. TC1C = TC1R = 131.**

```
MOV       A,#01100000B
B0MOV     TC1M,A          ; Set the TC1 rate to Fcpu/4

MOV       A,#131          ; Set the auto-reload reference value
B0MOV     TC1C,A
B0MOV     TC1R,A

B0BSET    FTC1OUT         ; Enable TC1 output to P5.3 and disable P5.3 I/O function
B0BSET    FALOAD1         ; Enable TC1 auto-reload function
B0BSET    FTC1ENB         ; Enable TC1 timer
```

✷ ***Note: Buzzer output is enable, and "PWM1OUT" must be "0".***

8.4.6 TC1 TIMER OPERATION SEQUENCE

TC1 timer operation includes timer interrupt, event counter, TC1OUT and PWM. The sequence of setup TC1 timer is as following.

- **Stop TC1 timer counting, disable TC1 interrupt function and clear TC1 interrupt request flag.**

```
B0BCLR    FTC1ENB     ; TC1 timer, TC1OUT and PWM stop.
B0BCLR    FTC1IEN     ; TC1 interrupt function is disabled.
B0BCLR    FTC1IRQ     ; TC1 interrupt request flag is cleared.
```

- **Set TC1 timer rate. (Besides event counter mode.)**

```
MOV       A, #0xxx0000b   ;The TC1 rate control bits exist in bit4~bit6 of TC1M. The
                          ; value is from x000xxxxb~x111xxxxb.
B0MOV     TC1M,A          ; TC1 interrupt function is disabled.
```

- **Set TC1 timer clock source.**

```
; Select TC1 internal / external clock source.
B0BCLR    FTC1CKS     ; Select TC1 internal clock source.
or
B0BSET    FTC1CKS     ; Select TC1 external clock source.
```

- **Set TC1 timer auto-load mode.**

```
B0BCLR    FALOAD1     ; Enable TC1 auto reload function.
or
B0BSET    FALOAD1     ; Disable TC1 auto reload function.
```

- **Set TC1 interrupt interval time, TC1OUT (Buzzer) frequency or PWM duty cycle.**

```
; Set TC1 interrupt interval time, TC1OUT (Buzzer) frequency or PWM duty.
MOV       A,#7FH      ; TC1C and TC1R value is decided by TC1 mode.
B0MOV     TC1C,A      ; Set TC1C value.
B0MOV     TC1R,A      ; Set TC1R value under auto reload mode or PWM mode.

; In PWM mode, set PWM cycle.
B0BCLR    FALOAD1     ; ALOAD1, TC1OUT = 00, PWM cycle boundary is
B0BCLR    FTC1OUT     ; 0~255.
or
B0BCLR    FALOAD1     ; ALOAD1, TC1OUT = 01, PWM cycle boundary is
B0BSET    FTC1OUT     ; 0~63.
or
B0BSET    FALOAD1     ; ALOAD1, TC1OUT = 10, PWM cycle boundary is
B0BCLR    FTC1OUT     ; 0~31.
or
B0BSET    FALOAD1     ; ALOAD1, TC1OUT = 11, PWM cycle boundary is
B0BSET    FTC1OUT     ; 0~15.
```

- **Set TC1 timer function mode.**

```
B0BSET    FTC1IEN     ; Enable TC1 interrupt function.
or
B0BSET    FTC1OUT     ; Enable TC1OUT (Buzzer) function.
or
B0BSET    FPWM1OUT    ; Enable PWM function.
```

- **Enable TC1 timer.**

```
B0BSET    FTC1ENB     ; Enable TC1 timer.
```

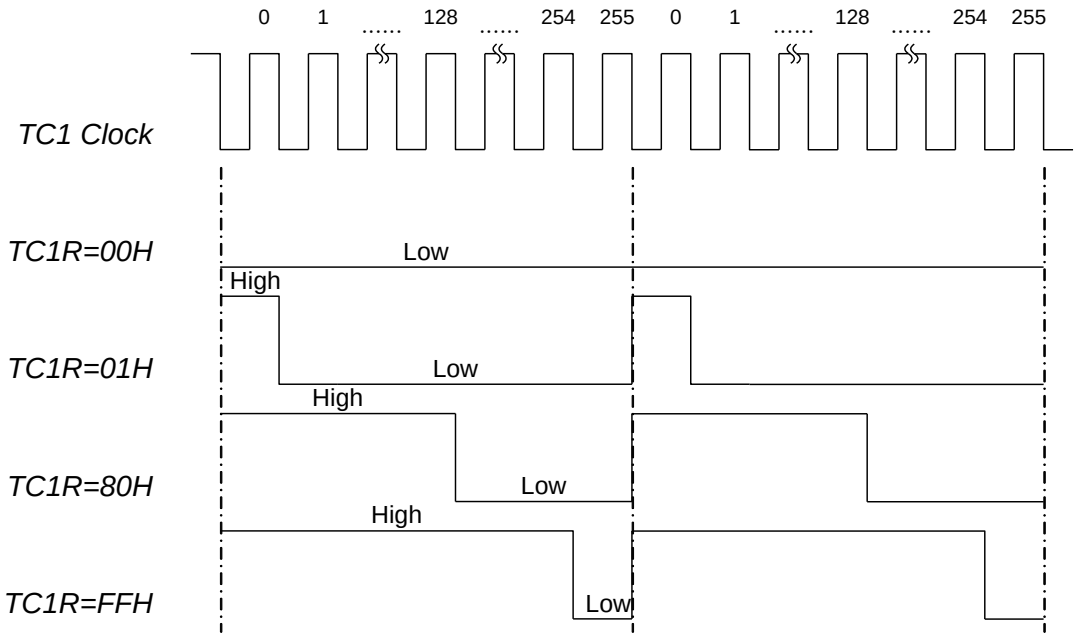
8.5 PWM1 MODE

8.5.1 OVERVIEW

PWM function is generated by TC1 timer counter and output the PWM signal to PWM1OUT pin (P5.3). The 8-bit counter counts modulus 256, 64, 32, 16 controlled by ALOAD1, TC1OUT bits. The value of the 8-bit counter (TC1C) is compared to the contents of the reference register (TC1R). When the reference register value (TC1R) is equal to the counter value (TC1C), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWM1 output is TC1R/256, 64, 32, 16.

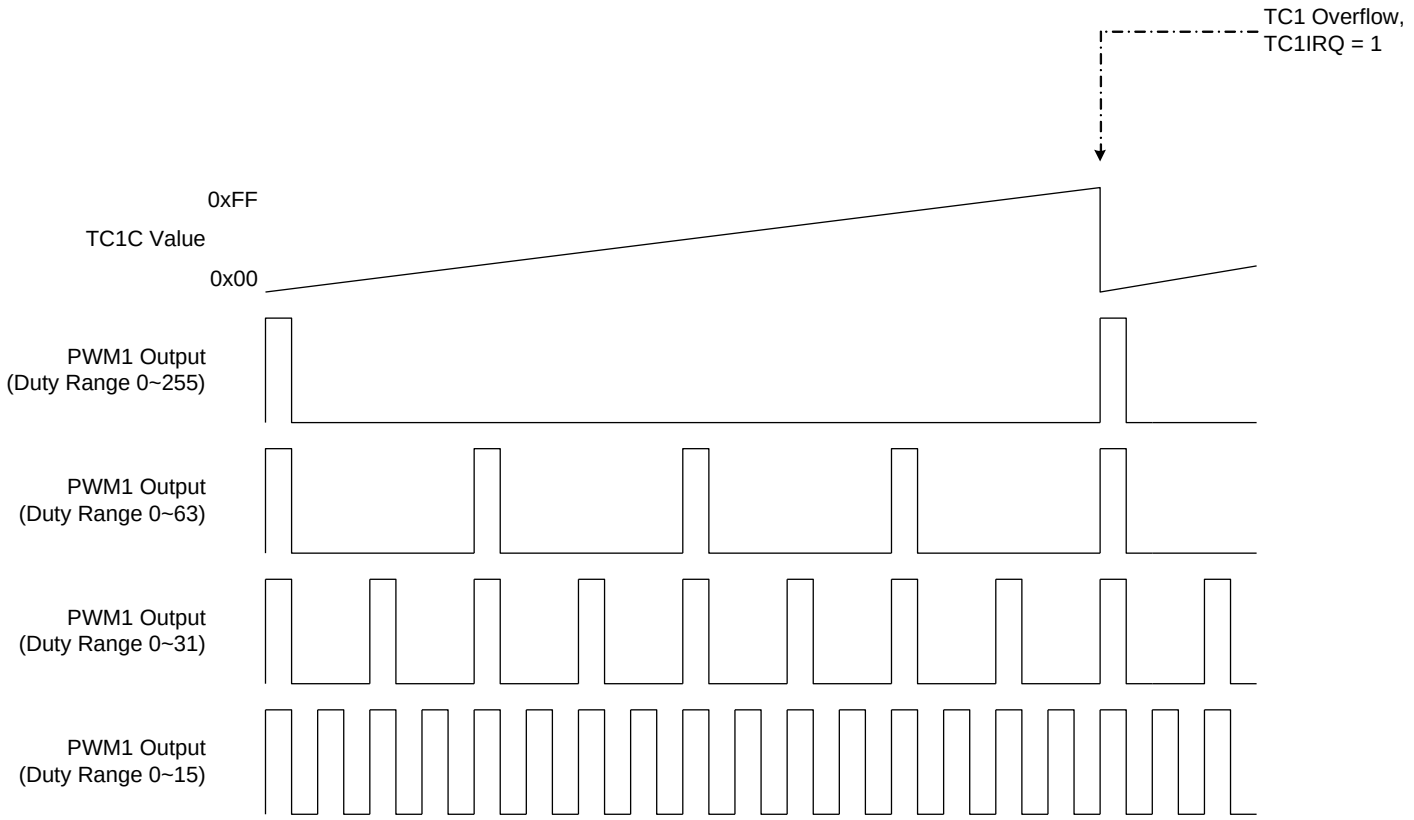
ALOAD1	TC1OUT	PWM duty range	TC1C valid value	TC1R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0	0	0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count
0	1	0/64~63/64	0x00~0x3F	0x00~0x3F	31.25K	Overflow per 64 count
1	0	0/32~31/32	0x00~0x1F	0x00~0x1F	62.5K	Overflow per 32 count
1	1	0/16~15/16	0x00~0x0F	0x00~0x0F	125K	Overflow per 16 count

The Output duty of PWM is with different TC1R. Duty range is from 0/256~255/256.



8.5.2 TC1IRQ AND PWM DUTY

In PWM mode, the frequency of TC1IRQ is depended on PWM duty range. From following diagram, the TC1IRQ frequency is related with PWM duty.



8.5.3 PWM PROGRAM EXAMPLE

➢ **Example: Setup PWM1 output from TC1 to PWM1OUT (P5.3). The external high-speed oscillator clock is 4MHz. Fcpu = Fosc/4. The duty of PWM is 30/256. The PWM frequency is about 1KHz. The PWM clock source is from external oscillator clock. TC1 rate is Fcpu/4. The TC1RATE2~TC1RATE1 = 110. TC1C = TC1R = 30.**

```
MOV     A,#01100000B
B0MOV   TC1M,A          ; Set the TC1 rate to Fcpu/4

MOV     A,#30           ; Set the PWM duty to 30/256
B0MOV   TC1C,A
B0MOV   TC1R,A

B0BCLR  FTC1OUT         ; Set duty range as 0/256~255/256.
B0BCLR  FALOAD1
B0BSET  FPWM1OUT        ; Enable PWM1 output to P5.3 and disable P5.3 I/O function
B0BSET  FTC1ENB         ; Enable TC1 timer
```

✸ ***Note: The TC1R is write-only register. Don't process them using INCMS, DECMS instructions.***

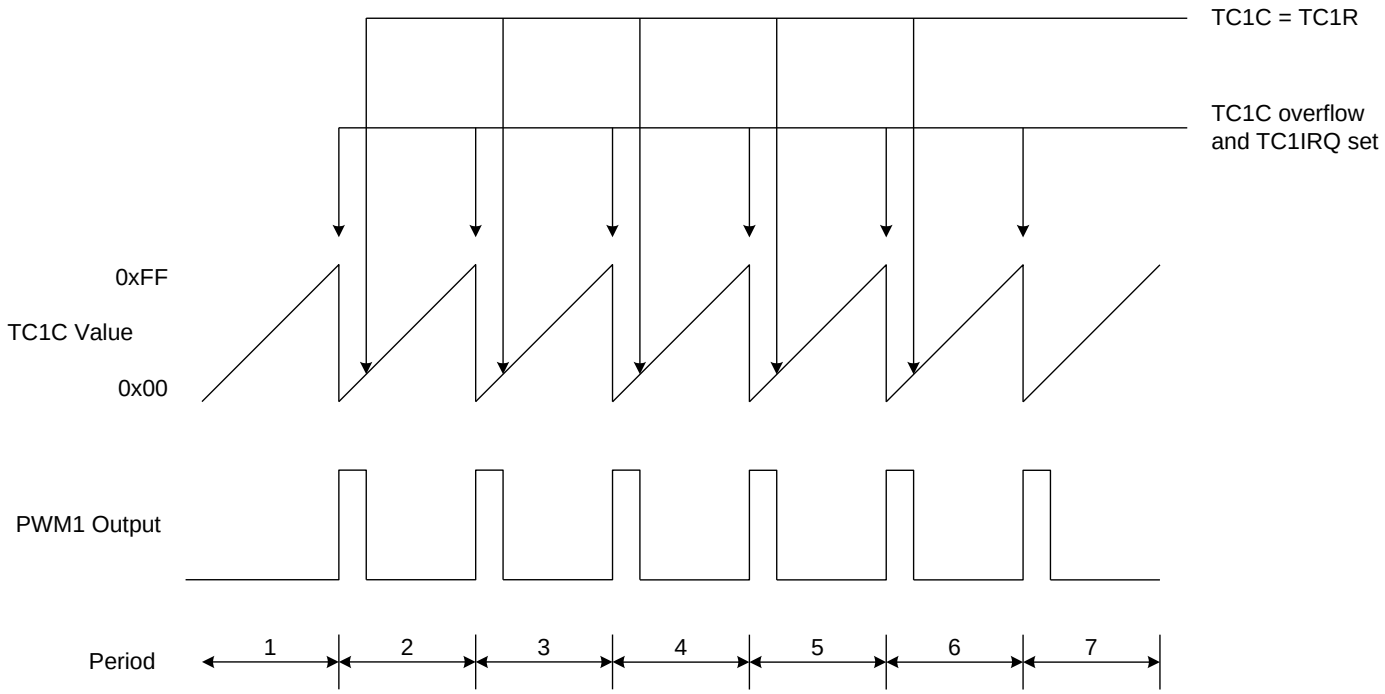
➢ **Example: Modify TC1R registers' value.**

```
MOV     A, #30H         ; Input a number using B0MOV instruction.
B0MOV   TC1R, A

INCMS   BUF0            ; Get the new TC1R value from the BUF0 buffer defined by
NOP                     ; programming.
B0MOV   A, BUF0
B0MOV   TC1R, A
```

✸ ***Note: The PWM can work with interrupt request.***

8.5.4 PWM1 DUTY CHANGING NOTICE

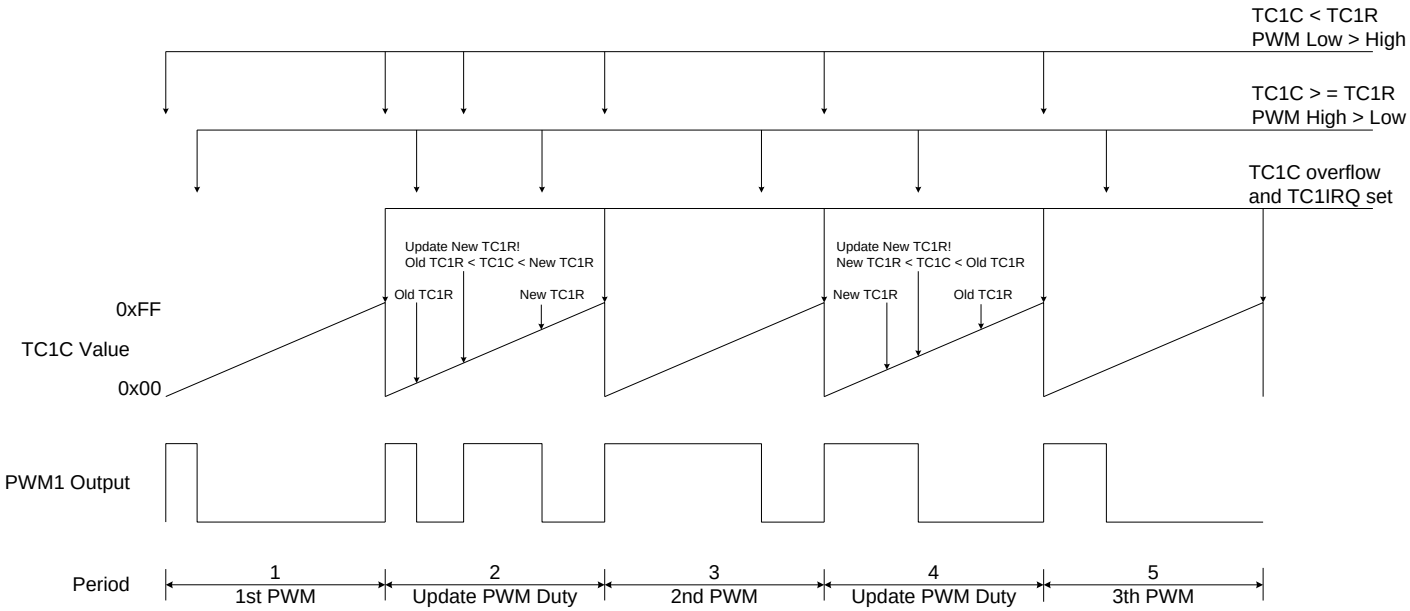
In PWM mode, the system will compare TC1C and TC1R all the time. When TC1C<TC1R, the PWM will output logic "High", when TC1C≧TC1R, the PWM will output logic "Low". If TC1C is changed in certain period, the PWM duty will change immediately. If TC1R is fixed all the time, the PWM waveform is also the same.



Above diagram is shown the waveform with fixed TC1R. In every TC1C overflow PWM output "High, when TC1C≧ TC1R PWM output "Low".

✷ ***Note: Setting PWM duty in program processing must be at the new cycle start.***

If TC1R is changing in the program processing, the PWM waveform will became as following diagram.



In period 2 and period 4, new Duty (TC1R) is set, but the PWM output waveform of period 2 and period 4 are wrong. In period 2, the new TC1R value is greater than old TC1R value. If setting new TC1R is after PWM output "low", system is getting TC1C < TC1R result and making PWM output "high". There are two high level periods in the cycle, and the waveform is unexpected. Until next cycle, PWM outputs correct duty. In period 4, the new TC1R value is smaller than the old TC1R value. If setting new TC1R is before PWM output "low", system is getting TC1C≧TC1R result and making PWM output "low". In the cycle, the high duty is shorter than last cycle and longer than correct cycle. It is an unexpected PWM output.

Though the wrong waveforms only exist in one cycle, it is still a problem for precise PWM application and might make outside loading operations error. The solution is to load new TC1R after TC1 timer overflow. Using TC1IRQ status to determine TC1 timer is overflow or not. When TC1IRQ becomes "1", to set the new TC1R value into TC1R buffer, and the unexpected PWM output is resolved.

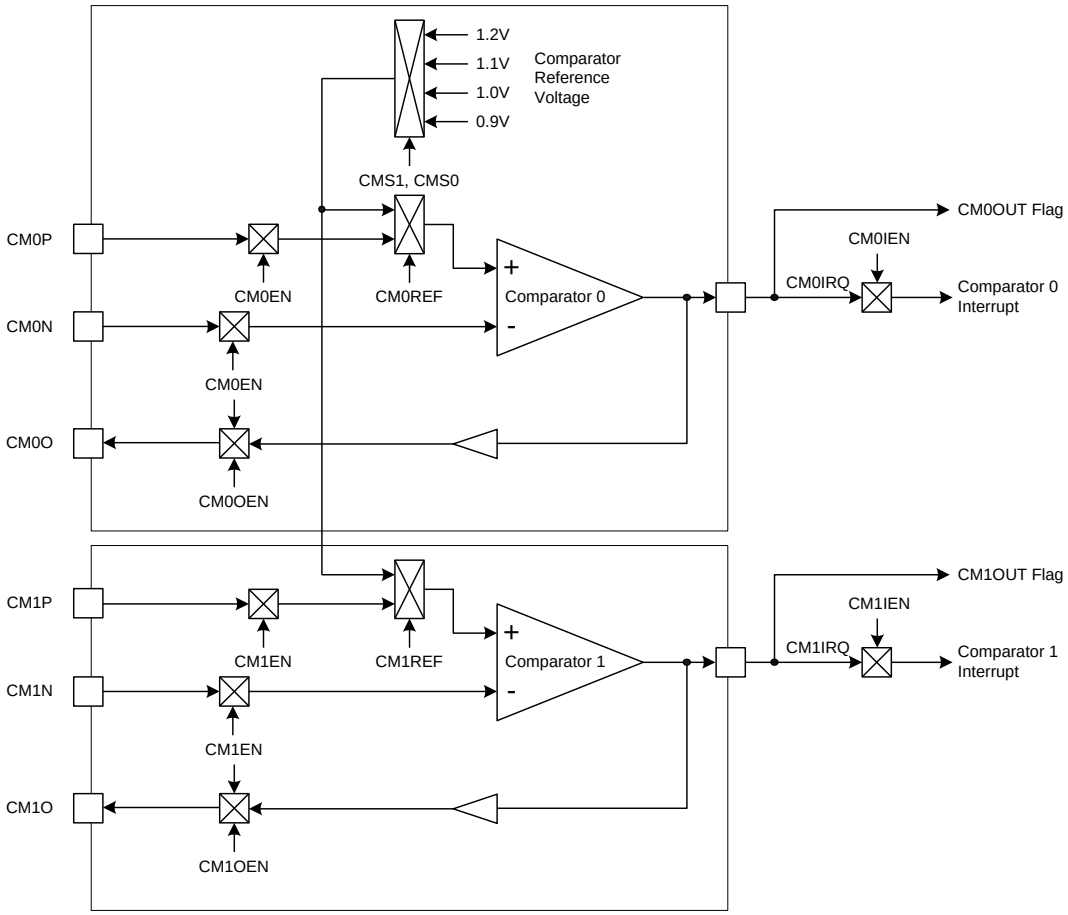
➢ **Example: Using TC1 interrupt function to set new TC1R value for changing PWM duty.**

```
MAIN:
        …
        B0MOV       TC1RBUF, A      ; Load new PWM duty setting value into TC1RBUF.
        …
        …

INT_SER:
        …                           ; Push routine to save ACC and PFLAG to buffers.
        …
        B0BTS1      FTC1IRQ
        JMP         INT_SER90
        B0MOV       A, TC1RBUF      ; When TC1 Interrupt occurs, update TC1R.
        B0MOV       TC1R, A
        …
        …
INT_SER90:
        …                           ; Pop routine to load ACC and PFLAG from buffers.
        RETI
```

9 ANALOG COMPARATOR

9.1 OVERVIEW

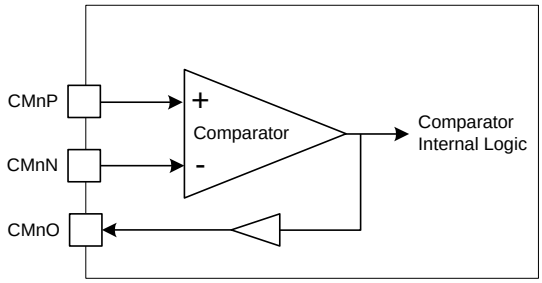
The analog comparator function includes two channel analog comparators and internal reference voltage. The main purpose of the comparator is to compare DC power voltage for low power indicator. The analog comparator negative input pin, positive input pin and output pin are shared with GPIO and controlled by registers. The 2-channel analog comparator structure is as following.



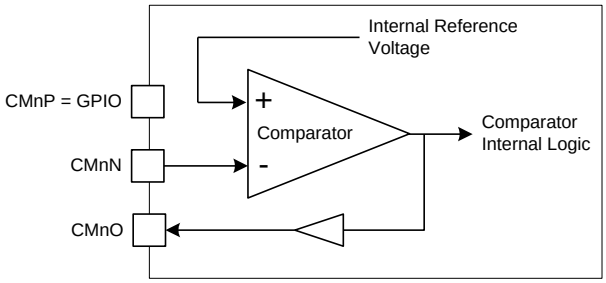
Comparator 0 Pin assignment:
CM0P: Comparator 0 positive input pin shared with P2.3. CM0P enables when CM0EN=1 and CM0REF=0.
CM0N: Comparator 0 negative input pin shared with P2.2. CM0N enables when CM0EN=1.
CM0O: Comparator 0 output pin shared with P2.4. CM0O enables when CM0EN=1 and CM0OEN = 1.

Comparator 1 Pin assignment:
CM1P: Comparator 1 positive input pin shared with P2.6. CM1P enables when CM1EN=1 and CM1REF=0.
CM1N: Comparator 1 negative input pin shared with P2.5. CM1N enables when CM1EN=1.
CM1O: Comparator 1 output pin shared with P2.7. CM1O enables when CM1EN=1 and CM1OEN = 1.

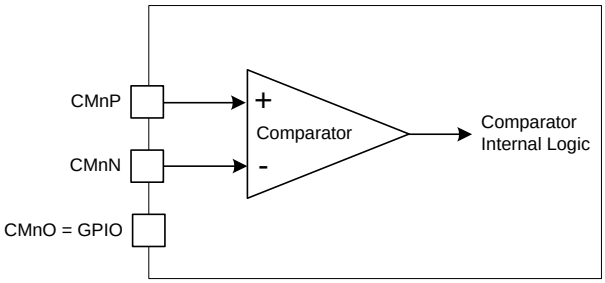
The comparator pins are GPIO mode except above conditions.



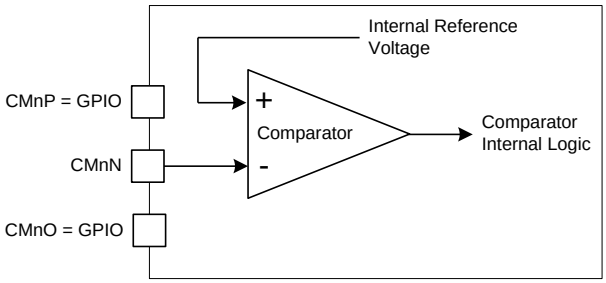
CMnEN = 1, CMnOEN = 1, CMnREF = 0



CMnEN = 1, CMnOEN = 1, CMnREF = 1



CMnEN = 1, CMnOEN = 0, CMnREF = 0



CMnEN = 1, CMnOEN = 0, CMnREF = 1

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

9.2 CMP0M REGISTER

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMP0M	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CMS1	CMS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **CM0EN:** Comparator 0 control bit.
0 = Disable. All comparator pins are GPIO.
1 = Enable. CM0N pin is analog input pin.

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

Bit 5 **CM0IRQ:** Comparator 0 interrupt request flag. CM0IRQ is latched to "1" as occurring comparator interrupt request, and it is cleared by program.
0 = No comparator interrupt request.
1 = Comparator interrupt request occurs when CM0P voltage or comparator 0 reference voltage is larger than CM0N voltage.

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

Bit 3 **CM0REF:** Comparator 0 internal reference voltage control bit.
0 = Disable. CM0P pin is analog input pin.
1 = Enable. CM0P pin is GPIO.

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

Bit[1:0] **CMS[1:0]:** Comparator internal reference voltage select bit.
00 = 0.9V, 01 = 1.0V, 10 = 1.1V, 11 = 1.2V

9.3 CMP1M REGISTER

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

Bit 7 **CM1EN:** Comparator 1 control bit.
0 = Disable. All comparator pins are GPIO.
1 = Enable. CM1N pin is analog input pin.

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

Bit 5 **CM1IRQ:** Comparator 1 interrupt request flag. CM1IRQ is latched to "1" as occurring comparator interrupt request, and it is cleared by program.
0 = No comparator interrupt request.
1 = Comparator interrupt request occurs when CM1P voltage or comparator 1 reference voltage is larger than CM1N voltage.

Bit 4 **CM1OEN:** Comparator 1 output pin control bit.
0 = Disable. CM1O pin is GPIO.
1 = Enable. CM1O pin is comparator output pin.

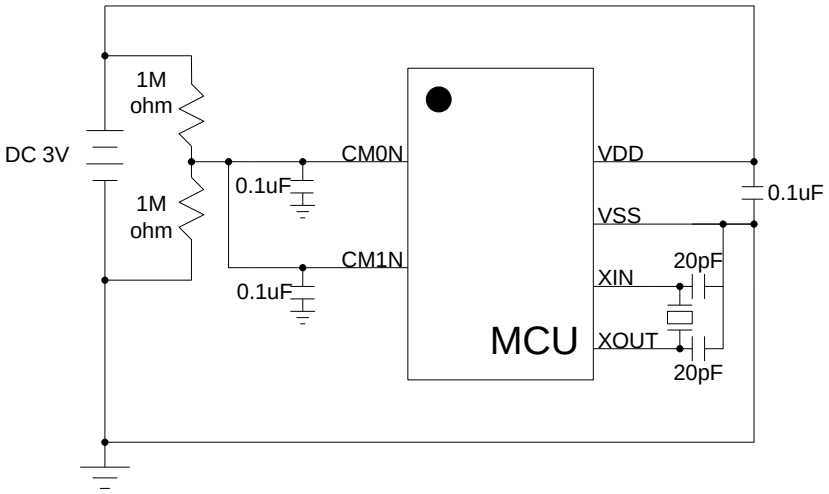
Bit 3 **CM1REF:** Comparator 1 internal reference voltage control bit.
0 = Disable. CM1P pin is analog input pin.
1 = Enable. CM1P pin is GPIO.

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

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

9.4 ANALOG COMPARATOR APPLICATION

This is a using the analog comparator to do two levels low battery detector. There are two low battery levels which are 2.2V and 2.0V. When the battery level is less than 2.2V, the system does low power process. When the battery level is less than 2.0V, the system does no power process. The battery detect level is 1/2 bias voltage of battery power source. The comparator positive voltage (reference voltage) is comparator internal reference voltage. The application circuit is as following.



The application circuit use internal reference and the comparator output process by internal flag, so the circuit only uses CMnN pin to input battery 1/2 bias voltage to compare with internal reference voltage. Use comparator 0 to check battery 2.2V and comparator 1 to check battery 2.0V.

➢ **Example: Use 2-ch comparators to detect battery status. The battery voltage less than 2.2V is low battery status. The battery voltage less than 2.0V is no battery status. This case is polling CMnOUT flag to check the battery voltage status and do difference processes. Users also can use the comparator interrupt function to obtain immediately process.**

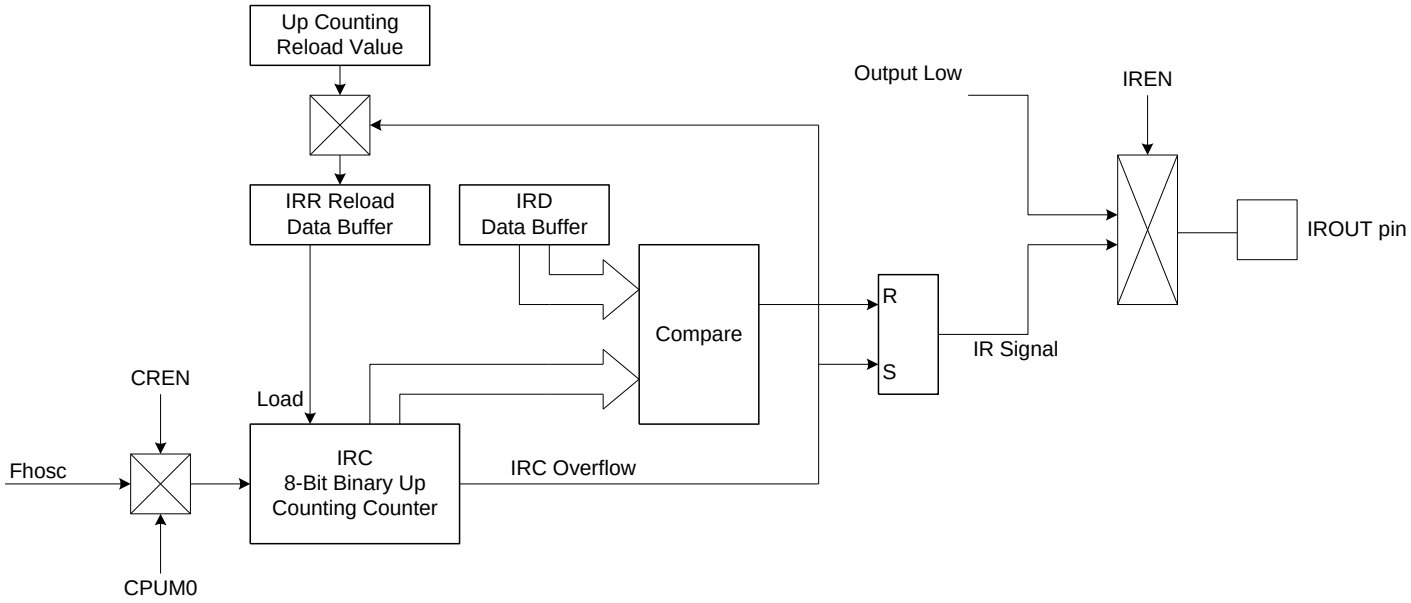
```
; The comparator initialize.
        MOV     A, #00001010b   ; Enable internal reference 1.1V.
        B0MOV   CMP0M, A        ; Disable comparator output pin.
        B0MOV   CMP1M, A        ; Disable comparator interrupt.

        B0BSET  FCM0EN          ; Enable comparator 0.
        B0BSET  FCM1EN          ; Enable comparator 1.

; Check 2.2V battery voltage.
CMP0_CHK:
        B0BSET  FCMS1           ; Set internal reference voltage = 1.1V.
        B0BCLR  FCMS0
        NOP                     ; Delay 2 instructions cycle to be the internal band-gap
        NOP                     ; setup time.

        B0BTS1  FCM0OUT         ; Check comparator 0 status through CM0OUT flag.
        JMP     Main            ; Not low battery, return to main loop.
        JMP     CMP1_CHK        ; Is low battery status, go to check comparator 1.

; Check 2.0V battery voltage.
CMP1_CHK:
        B0BCLR  FCMS1           ; Set internal reference voltage = 1.0V.
        B0BSET  FCMS0
        NOP                     ; Delay 2 instructions cycle to be the internal band-gap
        NOP                     ; setup time.

        B0BTS1  FCM1OUT         ; Check comparator 1 status through CM1OUT flag.
```

```
        JMP         LowBat      ; Is low battery status, go to low battery routine.
        JMP         NoBat       ; Is no battery status, go to no battery routine.

; Low battery process.
LowBat:
        …
        …
        JMP         Main        ; Return to main loop.

; No battery process.
NoBat:
        …
        …
        JMP         Main        ; Return to main loop.
```

10 IR OUTPUT

10.1 OVERVIEW

IR signal is generated by IR timer. The IR output pin is IROUT pin. When IREN bit of IRM is set as logic “1”, IROUT pin exchanges from GPIO to IR output mode. If CREN = 0 or system is in power down mode, IROUT pin is tied to low status. The IR timer is an 8-bit binary up counting timer for IR signal generator. The IR signal is duty/cycle changeable type controlled by IRR and IRD. IRR decides IR’s cycle and IRD decides IR’s duty. IR counter clock source is only from Fhosc (system high clock source), eg. IHRC_8M, 4MHz or 455KHz crystal. If Fhosc is 4MHz, the IR counter clock rate is 4MHz. IR timer only generate IR output and no interrupt function. When enable IR output function (CREN=1), IR output status is low level. IRC initial value is IRR and starts to count. When IRC=IRD, IR output status change to low level and finishes high duty operation. When IRC overflow occurs (IRC changes from 0xFF to 0x00), IR output low duty operation stops. System loads IRR into IRC automatically and next cycle starts.



10.2 IR CONTROL REGISTER

10.2.1 IRM MODE REGISTER

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRM	-	-	-	-	-	-	IREN	CREN
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

Bit 1 **IREN:** IROUT pin control bit.
0 = Disable. IROUT pin is P5.4 GPIO mode.
1 = Enable. IROUT pin is output low status.

Bit 0 **CREN:** IR carry signal output control bit.
0 = Disable. IROUT pin is output low status.
1 = Enable. IROUT pin outputs IR carry signal.

✷ ***Note: IR carry output condition is IREN=1 and CREN=1. If CREN=1 and IREN=0, the IROUT pin is P5.4 GPIO mode.***

10.2.2 IRC COUNTING REGISTER

IRC is an 8-bit counter register for IR interval time control.

0DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRC	IRC7	IRC6	IRC5	IRC4	IRC3	IRC2	IRC1	IRC0
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: Set IRC=IRR before IR output enable to make sure the first cycle correct.***

10.2.3 IRR AUTO-LOAD REGISTER

IRR decides IR signal frequency. IR timer is with auto-load function. When IRC overflow occurs, IRR value will load to IRC. It is easy to generate an accurate time for IR signal cycle, and users don't reset IRC during interrupt service routine.

IR is double buffer design. If new IRR value is set by program, the new value is stored in 1st buffer. Until IR overflow occurs, the new value moves to real IRR buffer.

0CDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRR	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of IRRinitial value is as following.

IRR initial value = 256 - (IR interrupt interval time * input clock)

✷ ***Note: The input clock is 4MHz of external 4MHz oscillator.***

Example: Set IR cycle frequency is 38KHz. Input clock is 4MHz.

*IRR initial value = 256 - (IR interrupt interval time * input clock)*

IR interval time = 1/38KHz = 26.3us
Input clock = external oscillator 4MHz.

*IRR = 256 - (26.3us * 4MHz)*
= 150.8
≈ 151
= 97h

10.2.4 IRD IR DUTY CONTROL REGISTER

The IR signal is duty changeable by IRD. IRD decides the IR output signal low pulse width length. When IRC=IRD, the IR signal changes from high pulse to low pulse. The low pulse stops when IRC overflow. The high pulse width is IRD-IRR, and the low pulse width is 256-IRD. It is easy to modulate IR duty/cycle by IRR and IRD registers.

0E8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRD	IRD7	IRD6	IRD5	IRD4	IRD3	IRD2	IRD1	IRD0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of IRD initial value is as following.

IRD initial value = IRR + (256-IRR) / (1/IR duty)

Example: Set IRD for 38KHz IR and duty is 1/3. Input clock is 4MHz.

IRD initial value = IRR + (256-IRR) / (1/IR duty)

IRR of 38KHz = 151

IRD = 151 + (256-151)/(1/ (1/3))
= 186
= BAh

Common IR signal table. System clock is 4MHz.

IR Freq. (KHz)	IRC IRR		IRD						Freq. Error Rate
			1/2 duty		1/3 duty		1/4duty		
	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	
32	131	83	193.50	C1	172.67	AC	162.25	A2	0.00%
36	145	91	200.50	C8	182.00	B6	172.75	AC	0.10%
38	151	97	203.50	CB	186.00	BA	177.25	B1	0.25%
39.2	154	9A	205.00	CD	188.00	BC	179.50	B3	0.04%
40	156	9C	206.00	CE	189.33	BD	181.00	B5	0.00%
56	185	B9	220.50	DC	208.67	D0	202.75	CA	0.60%

10.2.5 IR OUTPUT OPERATION SEQUENCE

☞ **Set IRC and IRR for IR cycle.**

```
MOV     A, #IRCYCVAL    ;IRC, IRR value for IR cycle.
MOV     IRC, A
MOV     IRR, A
```

☞ **Set IRD for IR duty.**

```
MOV     A, #IRDUTYVAL   ;IRD value for IR duty.
MOV     IRD, A
```

☞ **Enable IR output.**

```
BSET    FIREN           ; Set IROUT pin to IR carry output function.
BSET    FCREN           ; Set IR carry signal output.
```

11 SERIAL INPUT/OUTPUT TRANSCEIVER (SIO)

11.1 OVERVIEW

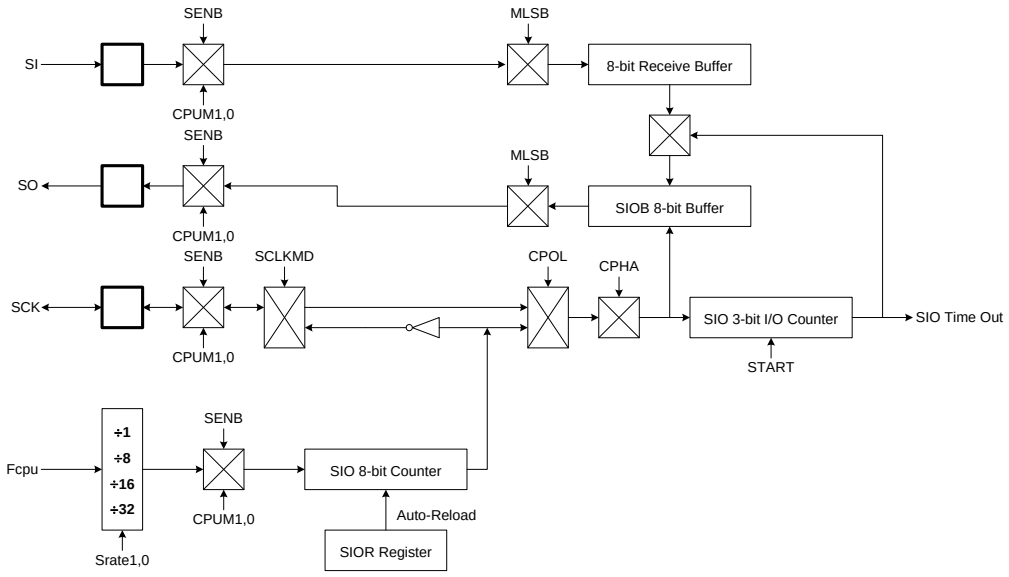
The SIO (serial input/output) transceiver is a serial communicate interface for data exchanging from one MCU to one MCU or other hardware peripherals. It is a simple 8-bit interface without a major definition of protocol, packet or control bits. The SIO transceiver includes three pins, clock (SCK), data 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.**

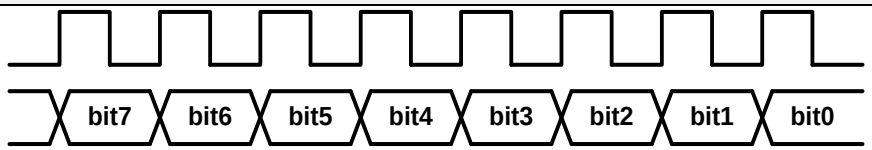
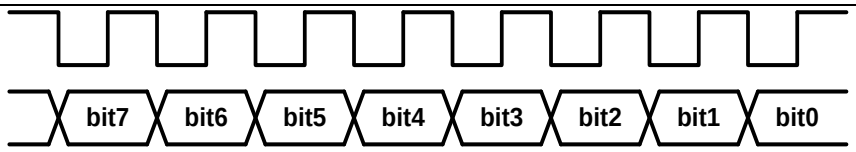
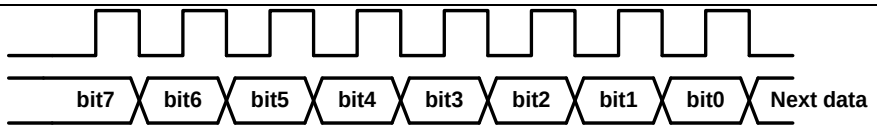
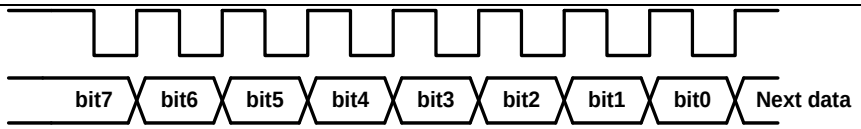
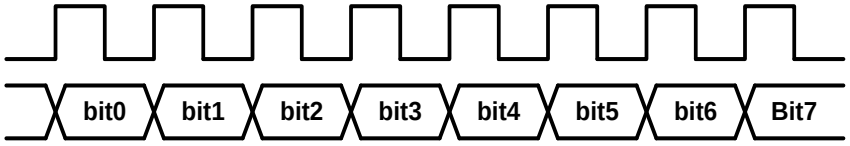
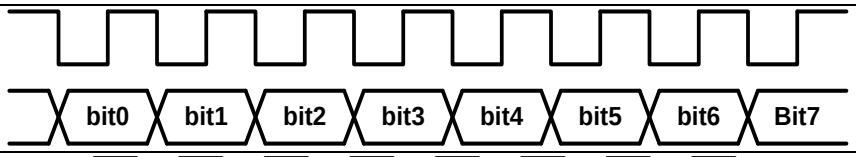
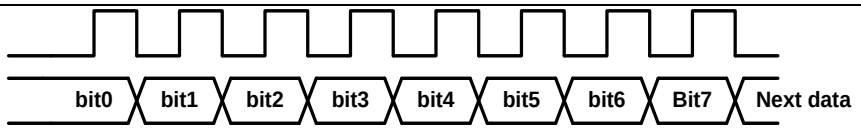
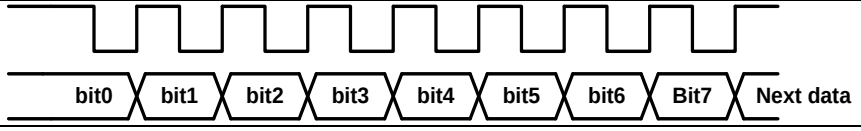
11.2 SIO OPERATION

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.***

11.3 SIOM MODE REGISTER

SIOM initial value = 0000 0000

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	

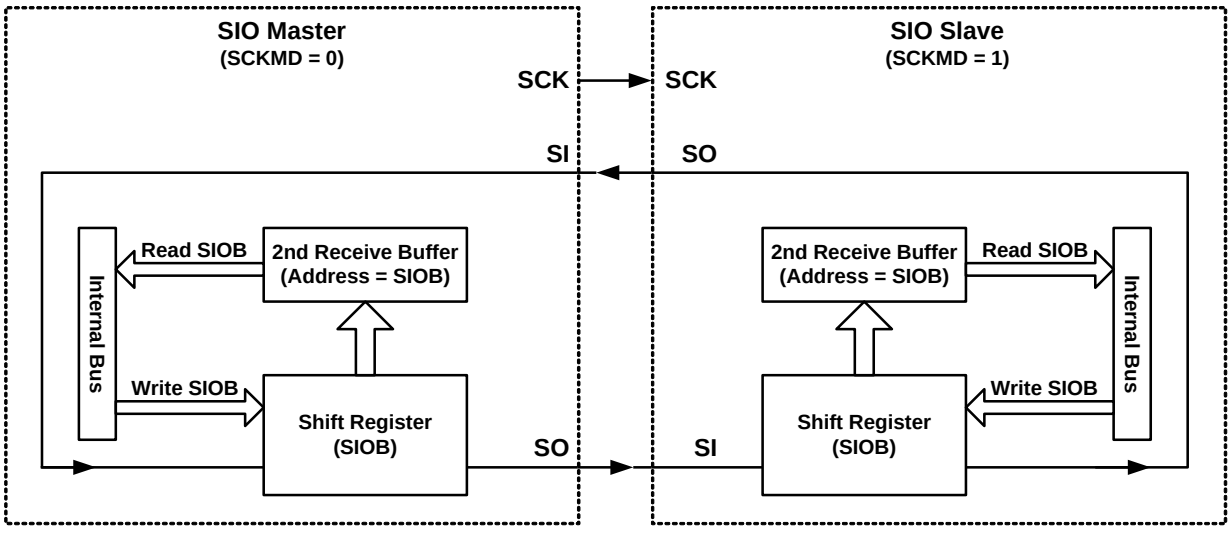
✷ ***Note:***
1. ***If SCKMD=1 for external clock, the SIO is in SLAVE mode. If SCKMD=0 for internal clock, the SIO is in MASTER mode.***
2. ***Don't set SENB and START bits in the same time. That makes the SIO function error.***
3. ***SIO pin can be push-pull structure and open-drain structure controlled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

11.4 SIOB DATA BUFFER

SIOB initial value = 0000 0000

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

11.5 SIOR REGISTER DESCRIPTION

SIOR initial value = 0000 0000

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)

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

- **Example: Setup the SIO clock to be 5KHz. Fosc = 3.58MHz. SIO's rate = Fcpu = Fosc/4.**

SIOR = 256 – (1/(5KHz) * 3.58MHz/4)
= 256 – (0.0002*895000)
= 256 – 179
= 77

12 Universal Asynchronous Receiver/Transmitter (UART)

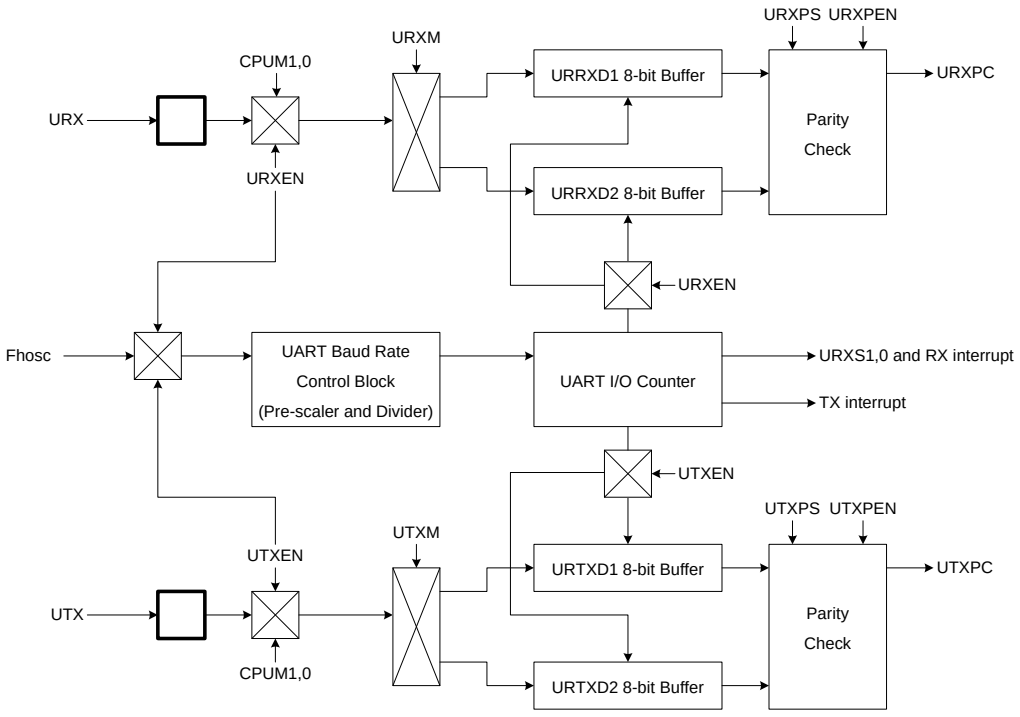
12.1 OVERVIEW

The UART interface is an universal asynchronous receiver/transmitter method. The serial interface is applied to low speed data transfer and communicate with low speed peripheral devices. The UART transceiver of Sonix 8-bit MCU allows RS232 standard and supports one and two bytes data length. The transfer format has start bit, 8/16-bit data, parity bit and stop bit. Programmable baud rate supports different speed peripheral devices. UART I/O pins support push-pull and open-drain structures controlled by register. The UART features include the following:

- **Full-duplex, 2-wire asynchronous data transfer.**
- **Programmable baud rate.**
- **8-bit and 16-bit data length.**
- **Odd and even parity bit.**
- **End-of-Transfer interrupt.**

12.2 UART OPERATION

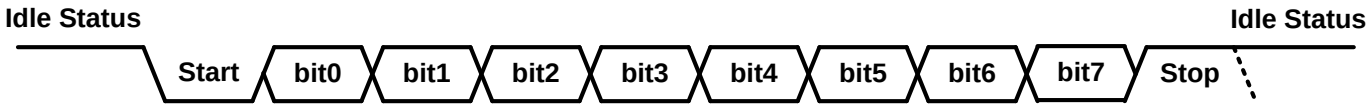
The UART RX and TX pins are shared with GPIO. When UART enables (RXDEN=1, TXDEN=1), the UART shared pins transfers to UART purpose and disable GPIO function automatically. When UART disables, the UART pins returns to GPIO last status. The UART data buffer length supports 1-byte and 2-byte. After UART RX operation finished, the RXIRQ sets as “1”. After UART TX operation finished, the TXIRQ sets as “1”. The UART IRQ bits are cleared by program. If the RXIEN or TXIEN set to enable, the RXIRQ and TXIRQ triggers the interrupt request and program counter jumps to interrupt vector to execute interrupt service routine.



UART Interface Circuit Diagram

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

UART Transfer Format with Parity Bit



UART Transfer Format without Parity Bit

Bus Idle Status

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

Start Bit

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

8-bit Data

The data format is 8-bit length, and MSB transfers first following start bit. The one bit data duration is the unit of UART baud rate controlled by register.

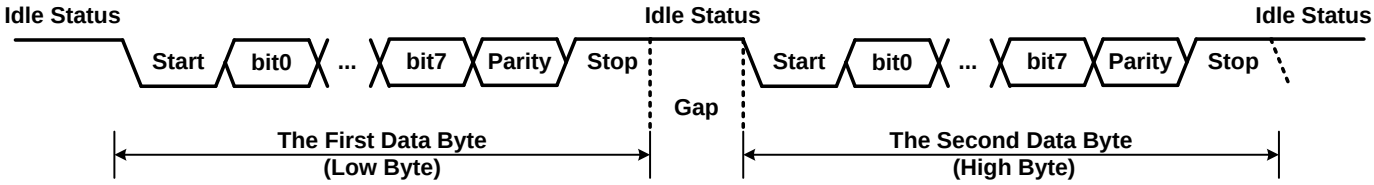
Parity Bit

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

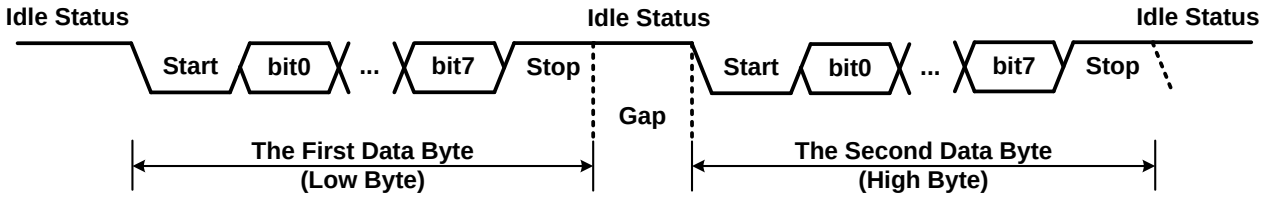
Stop Bit

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

The UART communication supports 2-byte data length. The function is for continuous data streams and immediate data request. The 2-byte data format is a continuously byte data form. The gap between the 2-byte data is unit baud rate. The first byte data stores in URRXD1 (receiver) and URTXD1 (transmitter). The second byte data stores in URRXD2 (receiver) and URTXD2 (transmitter).The 2-byte data format is as following.



2-Byte Transfer Format with Parity Bit



2-Byte Transfer Format without Parity Bit

The UART supports interrupt function. RXIEN/TXIEN are UART transfer interrupt function control bit. RXIEN=0, disable UART receiver interrupt function. TXIEN=0, disable UART transmitter interrupt function. RXIEN=1, enable UART receiver interrupt function. TXIEN=1, enable UART transmitter interrupt function. When UART interrupt function enable, the program counter points to interrupt vector (ORG 8) to do UART interrupt service routine after UART operating. TXIRQ/RXIQ are UART interrupt request flags, and also to be the UART operating status indicator when RXIEN=0 or TXIEN=0, but cleared by program. When UART operation finished, the RXIRQ/TXIRQ would be set to “1”.

✸ ***Note: The first step of UART operation is to setup the UART pins’ mode. Enable URXEN/UTXEN to control UART pins’ mode.***

12.3 UART RECEIVER CONTROL REGISTER

URRX initial value = xxx0000x

0A5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRX	URXEN	URXS1	URXS0	URXPEN	URXPS	URXPC	URXM	-
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 **URXEN:** UART RX control bit.
0 = Disable UART RX. URX pin keeps and returns to GPIO function.
1 = Enable UART RX. URX pin receives UART data.

Bit[6:5] **URXS1, URXS0:** UART RX status indicator.
00 = No data received.
01 = Data received, but parity checking error occurrence.
10, 11 = Data received successfully.

Bit 4 **URXPEN:** UART RX parity bit check function control bit.
0 = Disable UART RX parity bit check function. The data stream doesn't include parity bit.
1 = Enable UART RX parity bit check function. The data stream includes parity bit.

Bit 3 **URXPS:** UART RX parity bit format control bit.
0 = UART RX parity bit format is even parity.
1= UART RX parity bit format is odd parity.

Bit 2 **URXPC:** UART RX parity bit checking status bit.
0 = UART RX parity bit checking is error.
1 = UART RX parity bit checking is correct.

Bit 1 **URXM:** UART RX data buffer length control bit.
0 = 1-byte.
1 = 2-byte.

12.4 UART TRANSMITTER CONTROL REGISTER

URTX initial value = xxx0000x

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTX	-	-	-	UTXEN	UTXPEN	UTXPS	UTXM	-
Read/Write	-	-	-	R/W	R/W	R/W	R/W	-
After reset	-	-	-	0	0	0	0	-

Bit 4 **UTXEN:** UART TX control bit.
0 = Disable UART TX. UTX pin keeps and returns to GPIO function.
1 = Enable UART TX. UTX pin transmits UART data.

Bit 3 **UTXPEN:** UART TX parity bit check function control bit.
0 = Disable UART TX parity bit check function. The data stream doesn't include parity bit.
1 = Enable UART TX parity bit check function. The data stream includes parity bit.

Bit 2 **UTXPS:** UART TX parity bit format control bit.
0 = UART TX parity bit format is even parity.
1= UART TX parity bit format is odd parity.

Bit 1 **UTXM:** UART TX data buffer length control bit.
0 = 1-byte.
1 = 2-byte.

12.5 UART BAUD RATE CONTROL REGISTER

URBRC initial value = 11010101

0A6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URBRC	UDIV4	UDIV3	UDIV2	UDIV1	UDIV0	UPCS2	UPCS1	UPCS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	1	1	0	1	0	1	0	1

Bit[7:3] **UDIV[4:0]:** UART baud rate divider.

Bit[2:0] **UPCS[2:0]:** UART baud rate pre-scalar.
000=Fhosc/2, 001=Fhosc/4, 010=Fhosc/8, 011=Fhosc/16, 100=Fhosc/32, 101=Fhosc/64, 110=Fhosc/128, 111=Fhosc/256

The UART baud rate clock source is Fhosc and divided by pre-scalar and divider. The equation is as following.

Pre-scaler UPCS[2:0]=000b, 001b:

$$UART\ Baud\ Rate = Fhosc/2^{PreScaler}/(Divider+1)/16$$

Pre-scaler UPCS[2:0]=010b~111b:

$$UART\ Baud\ Rate = Fhosc/2^{PreScaler}/(Divider)/16$$

Baud Rate	Fhosc = 16MHz			Fhosc = 8MHz			Fhosc = 4MHz		
	UPCS[2:0]	UDIV[4:0]	Inaccuracy	UPCS[2:0]	UDIV[4:0]	Inaccuracy	UPCS[2:0]	UDIV[4:0]	Inaccuracy
1200	101	11010	0.16%	100	11010	0.16%	011	11010	0.16%
2400	100	11010	0.16%	011	11010	0.16%	010	11010	0.16%
4800	011	11010	0.16%	010	11010	0.16%	001	11001	0.16%
9600	010	11010	0.16%	010	01101	0.16%	000	11001	0.16%
14400	010	10001	2.12%	001	10000	2.12%	000	10000	2.12%
19200	010	01101	0.16%	000	11001	0.16%	000	01100	0.16%
38400	001	01100	0.16%	000	01100	0.16%	000	00110	-6.99%
51200	000	10010	2.80%	000	01001	-2.34%	000	00100	-2.34%
57600	000	10000	2.12%	000	01000	-3.55%	000	00011	8.51%
102400	000	01001	-2.34%	000	00100	-2.34%	000	00010	-18.62%
115200	000	01000	-3.55%	000	00011	8.51%	000	00001	8.51%

12.6 UART DATA BUFFER

URTXD1 initial value = 0000 0000

0A7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTXD1	UTXD17	UTXD16	UTXD15	UTXD14	UTXD13	UTXD12	UTXD11	UTXD10
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] **URTXD1:** UART transmitted data buffer byte 1.

URTXD2 initial value = 0000 0000

0A8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTXD2	UTXD27	UTXD26	UTXD25	UTXD24	UTXD23	UTXD22	UTXD21	UTXD20
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] **URTXD2:** UART transmitted data buffer byte 2.

URRXD1 initial value = 0000 0000

0A9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRXD1	URXD17	URXD16	URXD15	URXD14	URXD13	URXD12	URXD11	URXD10
Read/Write	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

Bit[7:0] **URRXD1:** UART received data buffer byte 1.

URRXD2 initial value = 0000 0000

0AAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRXD2	URXD27	URXD26	URXD25	URXD24	URXD23	URXD22	URXD21	URXD20
Read/Write	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

Bit[7:0] **URRXD2:** UART received data buffer byte 2.

UART Data Mode	URTXD2	URTXD1	URRXD2	URRXD1
1-byte	0x00	1-byte data	0x00	1-byte data
2-byte	High-byte data	Low-byte data	High-byte data	Low-byte data

13 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 bits) into buffers.	-	-	-	1
C	POP		To pop ACC and PFLAG (except NT0, NPD bits) into 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”.

14 ELECTRICAL CHARACTERISTIC

14.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P26L38P, SN8P26L38X, SN8P26L38F ..... 0°C ~ + 70°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

14.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 3.0V, fosc = 4MHz, Fcpu=1MHZ, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, 25℃, Fcpu = 2mips.		1.8	3.0	3.6	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, *25℃*		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O output source current	IoH	Vop = Vdd – 0.5V		8	10	-	mA
sink current	IoL	Vop = Vss + 0.5V		8	12	-	
INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current	Idd1	Run Mode (No loading, Fcpu = Fosc/4)	Vdd= 3V, 4Mhz	-	1	2	mA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd=3V, ILRC 10Khz	-	5	10	uA
	Idd3	Sleep Mode	Vdd= 3V, *25℃*		1	2	uA
	Idd4	Green Mode (No loading, Fcpu = Fosc/4, Watchdog Disable)	Vdd= 3V, 4Mhz	-	0.25	0.5	mA
			Vdd=3V, ILRC 10Khz	-	3	6	uA
Internal High Oscillator Freq.	Fihrc	Internal High RC (IHRC)	*25℃*, Vdd= 3V, Fcpu = 1MHz	7.84	8	8.16	Mhz
Band-gap Output Voltage	Vbd1	CMS0, CMS1 = 00		-	0.9	-	V
	Vbd2	CMS0, CMS1 = 01		-	1.0	-	
	Vbd3	CMS0, CMS1 = 10		-	1.1	-	
	Vbd4	CMS0, CMS1 = 11		-	1.2	-	
Comparator Current	Icm1	Fcpu = 1MHz, Vdd=3V. Disable internal reference.		-	50	-	uA
	Icm2	Fcpu = 1MHz, Vdd=3V. Enable internal reference.		-	150	-	
Comparator Input Offset Voltage	Vcmoff	Fcpu = 1MHz, Vdd=3V,		-	±5	-	mV
LVD Voltage	Vdet0	Low voltage reset level.		-	1.7	-	V
	Vdet1	Low voltage reset level. Fcpu = 1 MHz. Low voltage indicator level. Fcpu = 1 MHz.		-	2.4	-	V
	Vdet2	Low voltage indicator level. Fcpu = 1 MHz		-	2.8	-	V

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

15 DEVELOPMENT TOOL

SONIX provides ICE (in circuit emulation), IDE (Integrated Development Environment) and EV-kit for SN8P26L38 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.

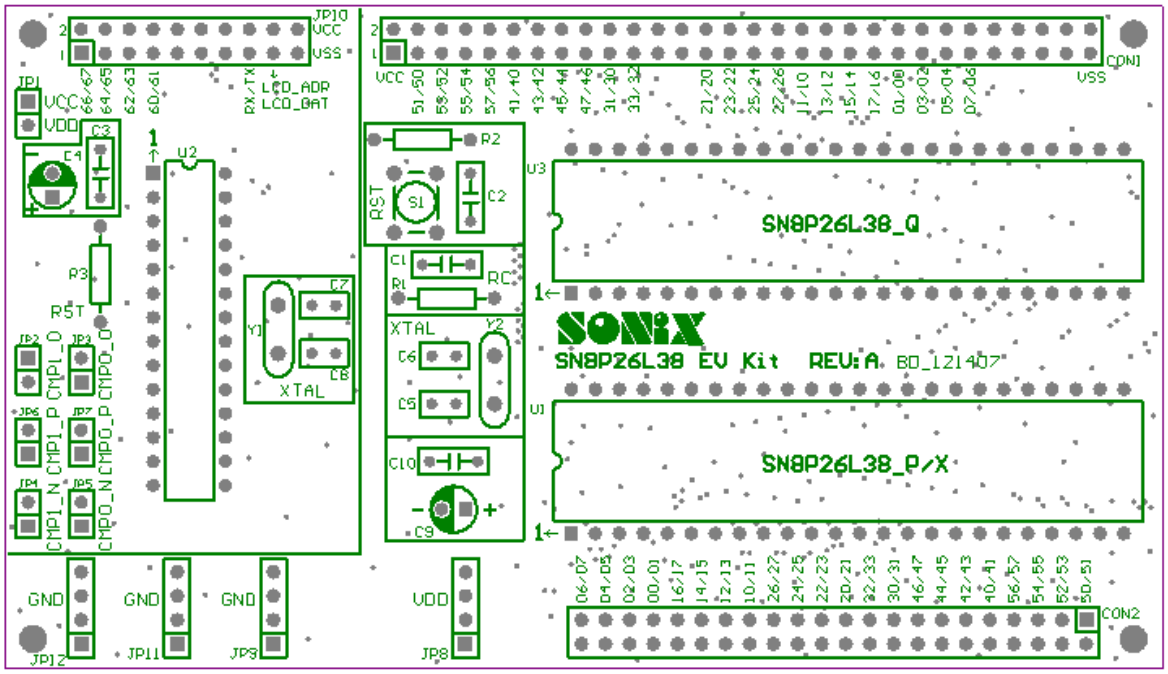
- **ICE: SN8ICE2K**
- **EV-kit: SN8P26L38 EV-kit Rev. B.**
- **IDE: SONiX IDE M2IDE_V115.**
- **Writer: MPIII WRITE-LV.**

15.1 SN8P26L38 EV-kit

SN8P26L38 EV-kit includes ICE interface, GPIO interface and EV-chip module.

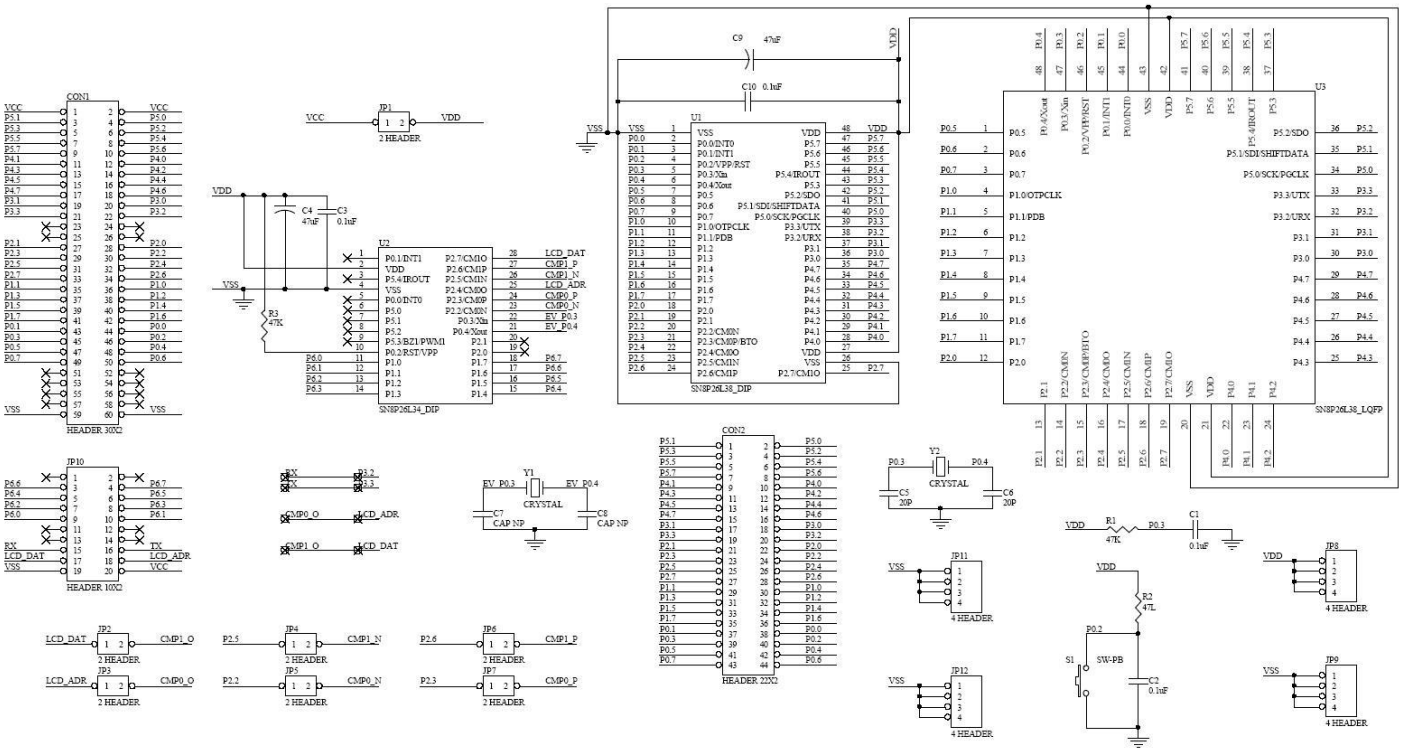
- **EV-chip module: .Emulate comparator function.**

The schematic of SN8P26L38 EV-kit is as following.



- CON1, JP10: ICE interface connected to SN8ICE2K .
- JP1: EV-Kit power connector between VCC and VDD. VCC is the power source from SN8ICE2K. VDD is the power of KV-kit.
- CON2: GPIO connector for test.
- U2: SN8P26L34 EV-chip for comparator emulation.
- U1: SN8P26L38 DIP and SSOP type connector for connecting to user's target board.
- U3: SN8P26L38 LQFP type connector for connecting with LQFP 48 pin socket.
- CM0_P: Comparator 0 positive input pin.
- CM0_N: Comparator 0 negative input pin.
- CM0_O: Comparator 0 output pin.
- CM1_P: Comparator 1 positive input pin.
- CM1_N: Comparator 1 negative input pin.
- CM1_O: Comparator 1 output pin.

- SN8P26L38 EV-kit Circuit



15.2 ICE and EV-KIT APPLICATION NOTIC

SN8P26L38 EV-kit includes comparator emulation module. There is a SN8P26L38P chip programmed emulating code to emulate comparator function. The SN8P26L38 comparator pins are shared with P2 GPIO pins. In ICE environment, the comparator pins isn't connected with GPIO pin.

- The Comparator emulation is from the SN8P26L34 EV-chip of SN8P26L38 EV-kit. For comparator emulation, input and output comparator signals from these pins.
- The P2 comparator shared pin GPIO emulation is from P2 pins of SN8P26L38 EV-kit.
- The SN8P26L38 EV-kit power level must be external 3V. Don't using ICE internal_5V power. Disconnect internal_5V pin of SN8ICE2K ICE and supply 3V power from external power source.

16 OTP PROGRAMMING PIN

16.1 The pin assignment of Easy Writer transition board socket:

Easy Writer JP1/JP2

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

JP1 for MP transition board

Easy Writer JP3 (Mapping to 48-pin text tool)

DIP1	1	48	DIP48
DIP2	2	47	DIP47
DIP3	3	46	DIP46
DIP4	4	45	DIP45
DIP5	5	44	DIP44
DIP6	6	43	DIP43
DIP7	7	42	DIP42
DIP8	8	41	DIP41
DIP9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP38
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

JP3 for MP transition board

16.2 Programming Pin Mapping:

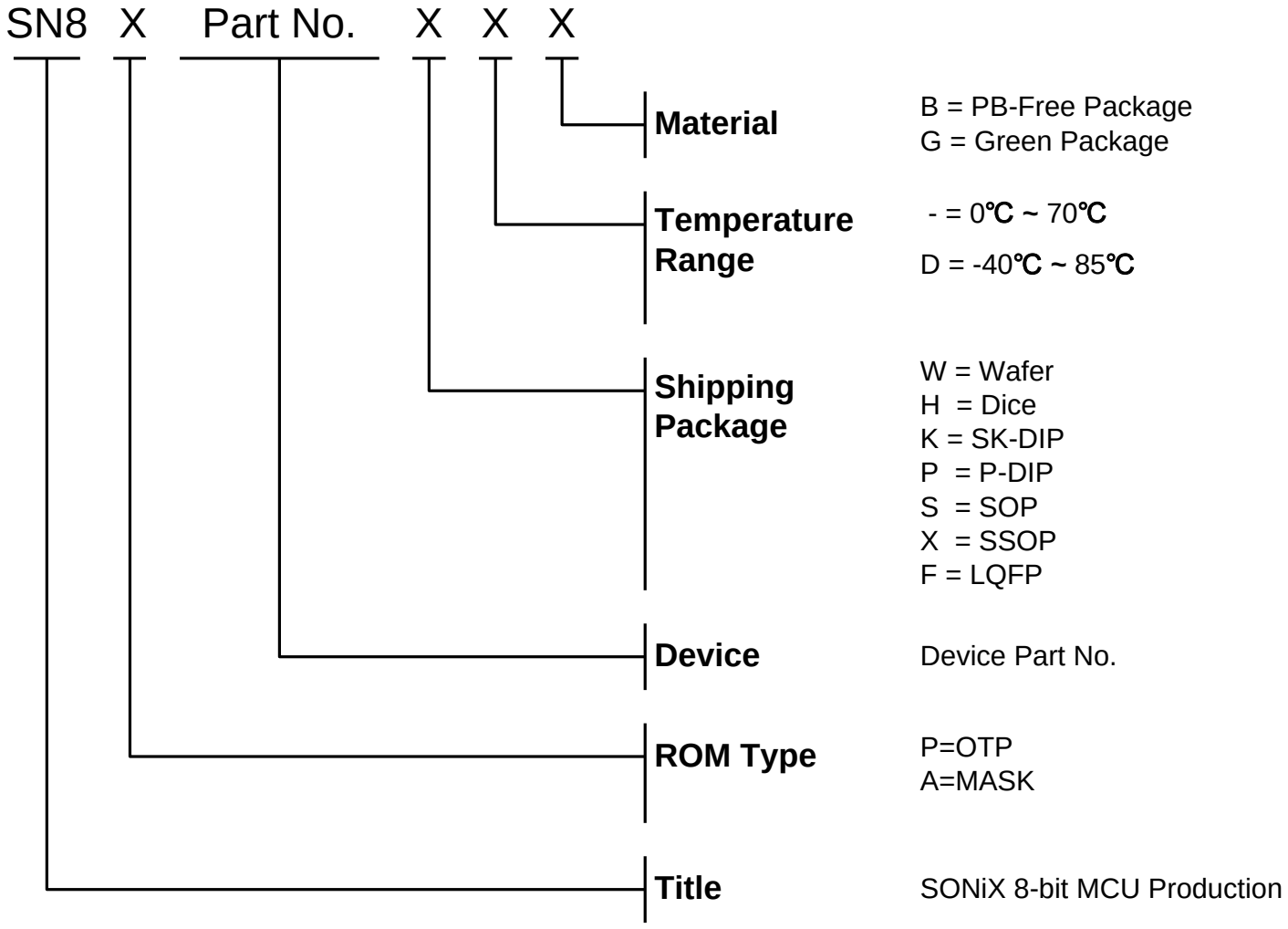
Programming Information of SN8P26L38									
Chip Name		SN8P26L38P/X		SN8P26L38F					
EZ Writer Connector		OTP IC / JP3 Pin Assignment							
Number	Name	Number	Pin	Number	Pin	Number	Pin	Number	Pin
1	VDD	27,48	VDD	21,42	VDD				
2	GND	1,26	VSS	20,43	VSS				
3	CLK	40	P5.0	34	P5.0				
4	CE		-		-				
5	PGM	10	P1.0	4	P1.0				
6	OE	41	P5.1	35	P5.1				
7	D1		-		-				
8	D0		-		-				
9	D3		-		-				
10	D2		-		-				
11	D5		-		-				
12	D4		-		-				
13	D7		-		-				
14	D6		-		-				
15	VDD		-		-				
16	VPP	4	RST	46	RST				
17	HLS		-		-				
18	RST		-		-				
19	-		-		-				
20	ALSB/PDB	11,45	P1.1, P5.5	5,39	P1.1, P5.5				

17 Marking Definition

17.1 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.

17.2 MARKING INDETIFICATION SYSTEM

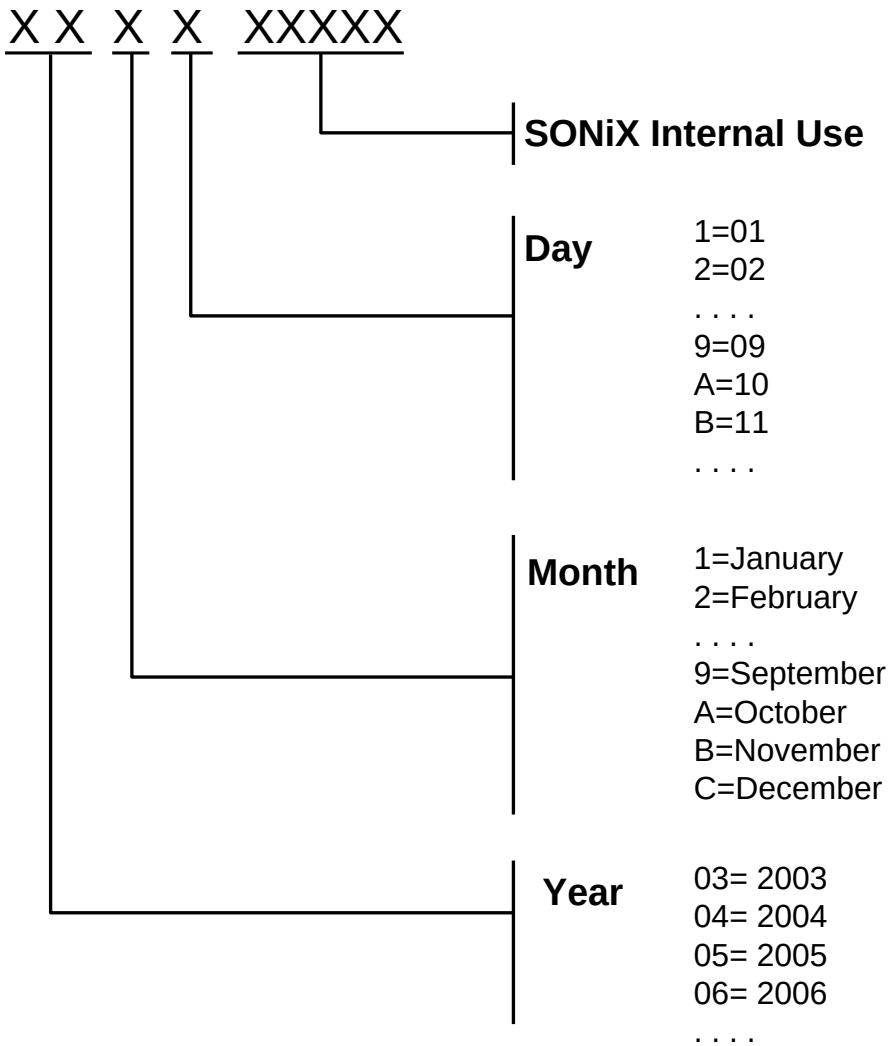


✸ ***Note: SN8P26L38 doesn't support -40ºC~85ºC temperature range and MASK ROM type.***

17.3 MARKING EXAMPLE

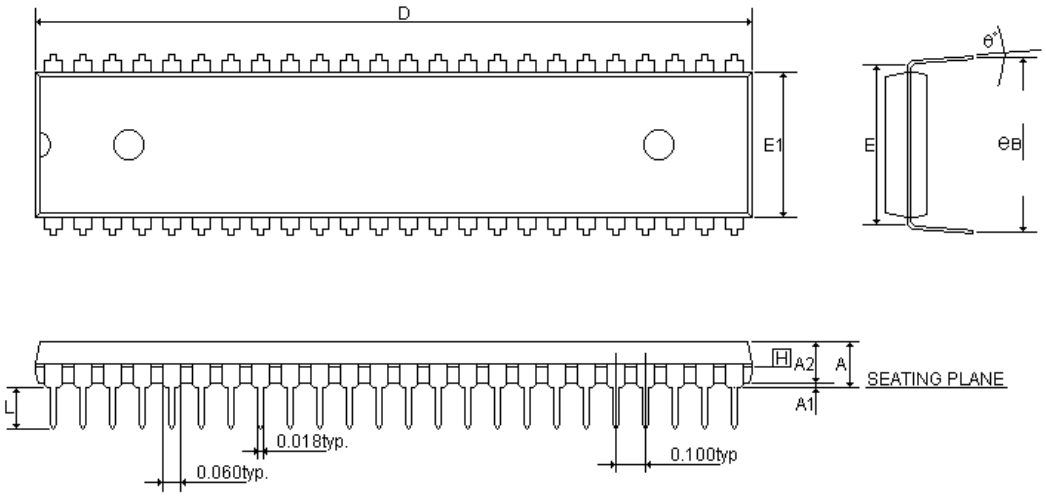
Name	ROM Type	Device	Package	Temperature	Material
SN8P26L38PB	OTP	26L38	PDIP	0℃~70℃	PB-Free Package
SN8P26L38XB	OTP	26L38	SSOP	0℃~70℃	PB-Free Package
SN8P26L38FB	OTP	26L38	LQFP	0℃~70℃	PB-Free Package
SN8P26L38PG	OTP	26L38	PDIP	0℃~70℃	Green Package
SN8P26L38XG	OTP	26L38	SSOP	0℃~70℃	Green Package
SN8P26L38FG	OTP	26L38	LQFP	0℃~70℃	Green Package
S8P26L38W	OTP	26L38	Wafer	0℃~70℃	-
SN8P26L38H	OTP	26L38	Dice	0℃~70℃	-

17.4 DATECODE SYSTEM



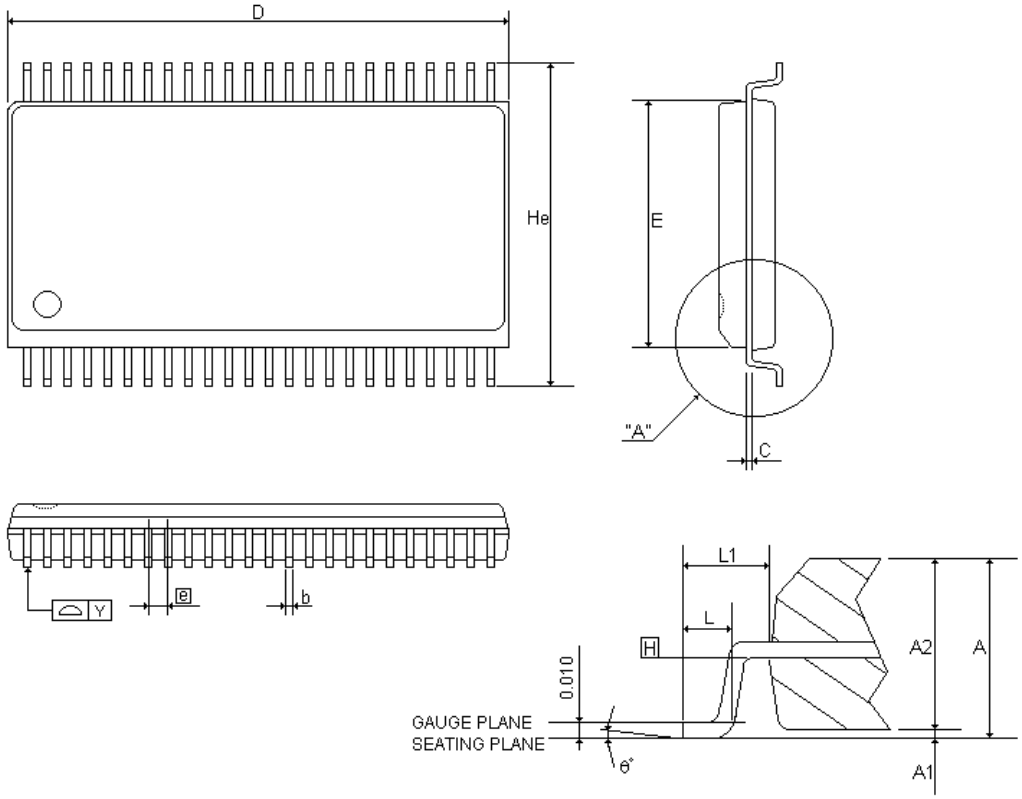
18 PACKAGE INFORMATION

18.1 P-DIP 48 PIN



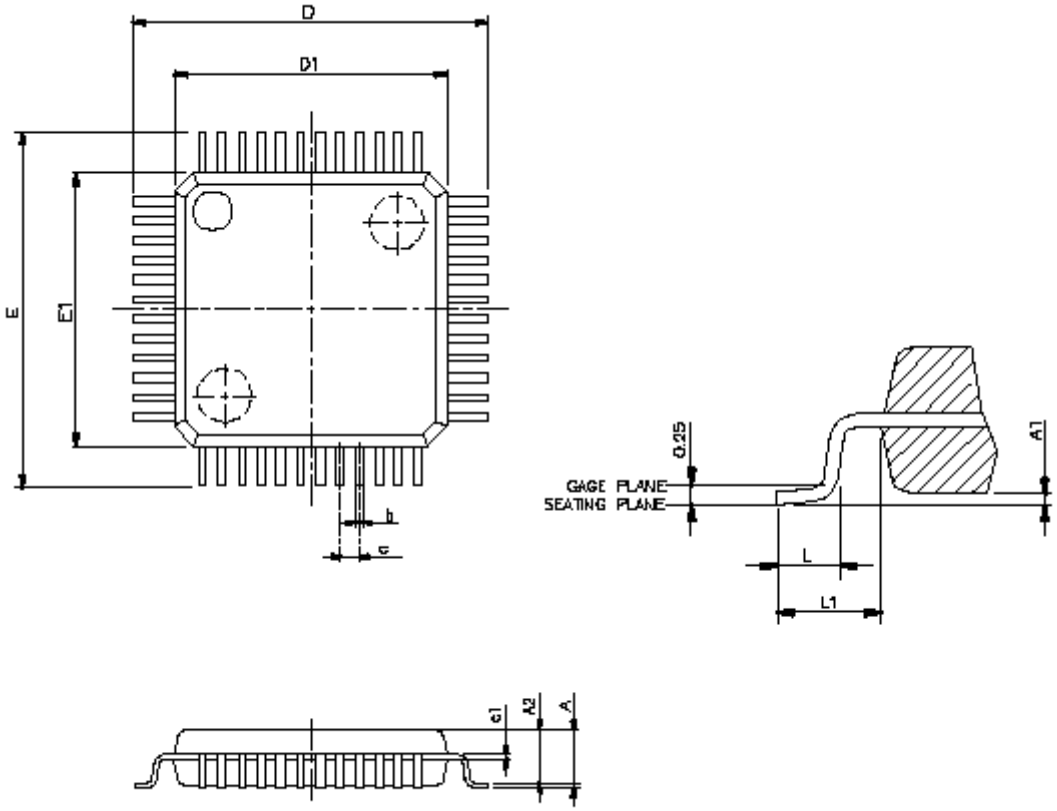
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.220	-	-	5.588
A1	0.015	-	-	0.381	-	-
A2	0.150	0.155	0.160	3.810	3.937	4.064
D	2.400	2.450	2.550	60.960	62.230	64.770
E		0.600			15.240	
E1	0.540	0.545	0.550	13.716	13.843	13.970
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.630	0.650	0.067	16.002	16.510	1.702
θ°	0°	7°	15°	0°	7°	15°

18.2 SSOP 48 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.095	0.102	0.110	2.413	2.591	2.794
A1	0.008	0.012	0.016	0.203	0.305	0.406
A2	0.089	0.094	0.099	2.261	2.388	2.515
b	0.008	0.010	0.030	0.203	0.254	0.762
C	-	0.008	-	-	0.203	-
D	0.620	0.625	0.630	15.748	15.875	16.002
E	0.291	0.295	0.299	7.391	7.493	7.595
[e]	-	0.025	-	-	0.635	-
He	0.396	0.406	0.416	10.058	10.312	10.566
L	0.020	0.030	0.040	0.508	0.762	1.016
L1	-	0.056	-	-	1.422	-
Y	-	-	0.003	-	-	0.076
θ°	0°	-	8°	0°	-	8°

18.3 LQFP 48 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.063	-	-	1.6
A1	0.002	-	0.006	0.05	-	0.15
A2	0.053	-	0.057	1.35	-	1.45
c1	0.004	-	0.006	0.09	-	0.16
D	0.354 BSC			9.00 BSC		
D1	0.276 BSC			7.00 BSC		
E	0.354 BSC			9.00 BSC		
E1	0.276 BSC			7.00 BSC		
e	0.020 BSC			0.5 BSC		
B	0.007	-	0.011	0.17	-	0.27
L	0.018	-	0.030	0.45	-	0.75
L1	0.039 REF			1 REF		

Main Office:
Address: 10F-1, NO. 36, Taiyuan Stree., Chupei City, Hsinchu, Taiwan R.O.C.
Tel: 886-3-5600 888
Fax: 886-3-5600 889
Taipei Office:
Address: 15F-2, NO. 171, Song Ted Road, Taipei, Taiwan R.O.C.
Tel: 886-2-2759 1980
Fax: 886-2-2759 8180
Hong Kong Office:
Unit 1519, Chevalier Commercial Center, No.8 Wang Hoi Road, Kowloon Bay, Kowloon. Hong Kong.
Tel: 852-2723-8086
Fax: 852-2723-9179
Technical Support by Email:
Sn8fae@sonix.com.tw