SN8P2501C

USER'S MANUAL

Preliminary Version 0.3

SN8P2501C

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 0.1	Feb. 2010	First issue.
VER 0.2	Jun. 2010	Add SN8P2501C EV-KIT schematic.
VER 0.3	Jun. 2010	Modify “4M/4” to “16M/16”.

Table of Content

1 PRODUCT OVERVIEW

1.1 FEATURES

- **Memory configuration**
  ROM size: 1K * 16 bits.
  RAM size: 48 * 8 bits.

- **4 levels stack buffer.**

- **3 interrupt sources**
  2 internal interrupts: T0, TC0
  1 external interrupt: INT0

- **I/O pin configuration**
  Bi-directional: P0, P1, P2, P5.
  Wakeup: P0, P1 level change.
  Pull-up resisters: P0, P1, P2, P5.
  Input only: P1.1
  Programmable open-drain: P1.0
  External interrupt: P0.0 (PEDGE edge trigger)

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

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

- **Fcpu (Instruction cycle)**
  Fcpu = Fosc/1, Fosc/2, Fosc/4, Fosc/8, Fosc/16, Fosc/32, Fosc/64, Fosc/128

- **One 8-bit basic timer. (T0).**
- **One 8-bit timer with external event counter, Buzzer and PWM. (TC0).**

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

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

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

¨ **Package (Chip form support)**
  DIP 14 pin
  SOP 14 pin
  MSOP 10 pin

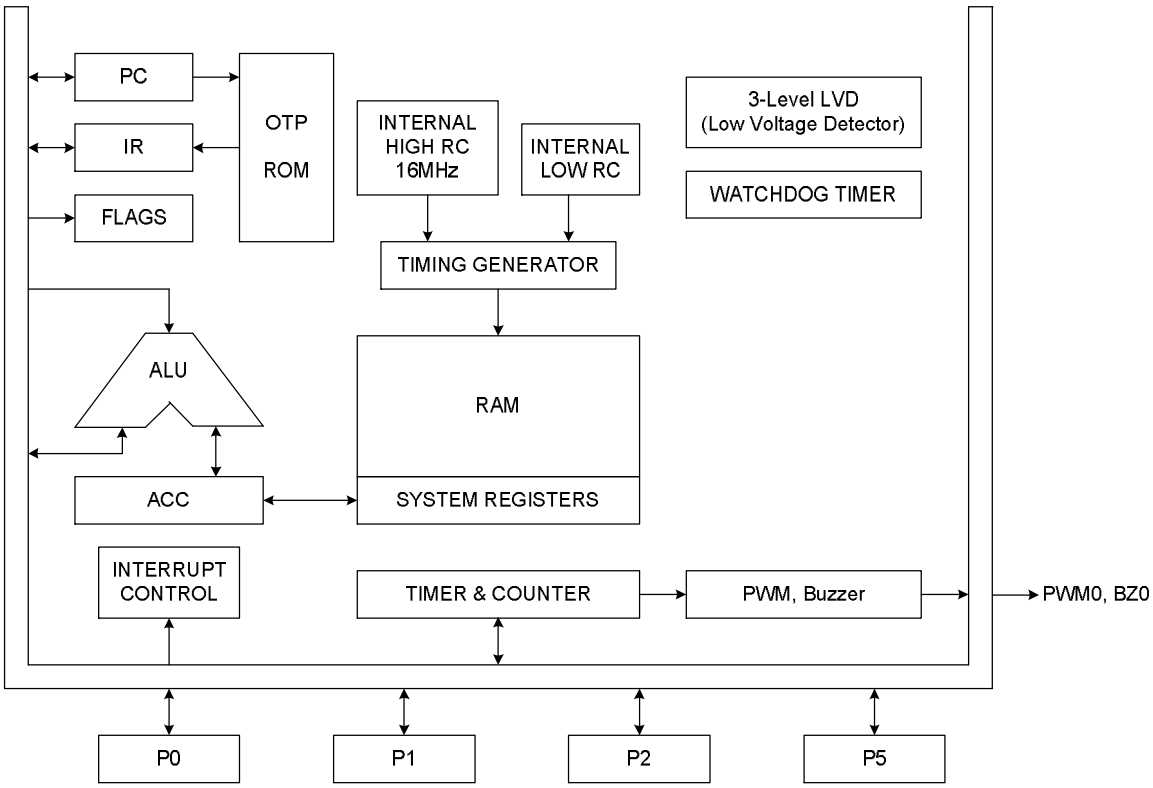
F Features Selection Table

CHIP	ROM	RAM	Stack	Timer		I/O	PWM Buzzer	IHRC	Ext. OSC	0.5s RTC	Wake-up Pin No.	Package
				T0	TC0							
SN8P2501B	1K*16	48	4	V	V	12	1	V	V	V	5	DIP14/ SOP14/ SSOP16
SN8P2501C	1K*16	48	4	V	V	8	1	V	-	-	5	DIP14/ SOP14/ MSOP10

F Migration SN8P2501B to SN8P2501C

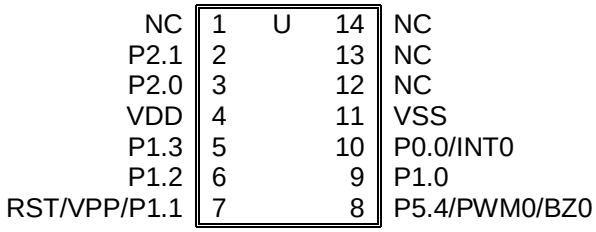
Item	SN8P2501B	SN8P2501C
Port 2 I/O pin	P2.0~P2.5	P2.0, P2.1
External oscillator	4M/12M/32K/RC	Not support external oscillator
IHRC 16MHz	IHRC 16MHz with RTC	IHRC 16MHz without RTC
Fcpu	Fosc/1~Fosc/16	Fosc/1~Fosc/128
T0 timer	T0 timer with RTC	T0 timer without RTC
Noise_Filter code option	Build in Noise_Filter option	Noise_Filter always enables.
Low power mode	No low power option.	Build in low power option for lower power consumption.

1.2 SYSTEM BLOCK DIAGRAM

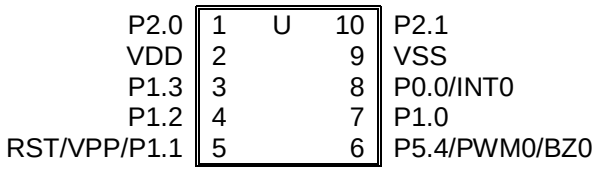


1.3 PIN ASSIGNMENT

SN8P2501CP (DIP 14 pins)
SN8P2501CS (SOP 14 pins)



SN8P2501CA (MSOP 10 pins)

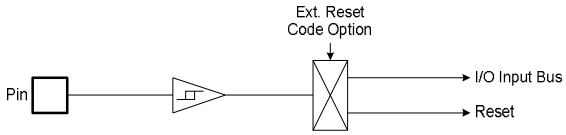


1.4 PIN DESCRIPTIONS

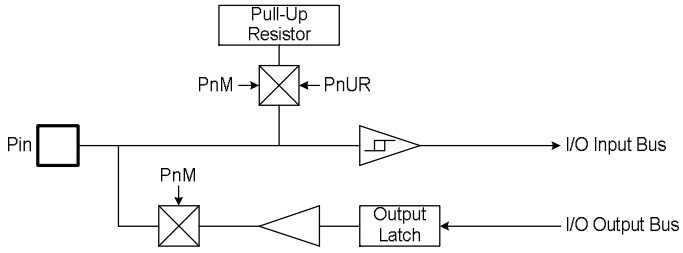
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital and analog circuit.
P1.1/RST/VPP	I, P	RST: System external reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”.
		VPP: OTP 12.3V power input pin in programming mode.
		P1.1: Input only pin with Schmitt trigger structure and no pull-up resistor. Level change wake-up.
P0.0/INT0	I/O	P0.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		INT0: External interrupt 0 input pin.
P1.0	I/O	P1.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. Programmable open-drain structure.
P1.2	I/O	P1.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
P1.3	I/O	P1.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
P5.4/PWM0/BZ0	I/O	P5.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		PWM0: PWM output pin.
		BZ0: Buzzer TC0/2 output pin.
P2.0	I/O	P2.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
P2.1	I/O	P2.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.

1.5 PIN CIRCUIT DIAGRAMS

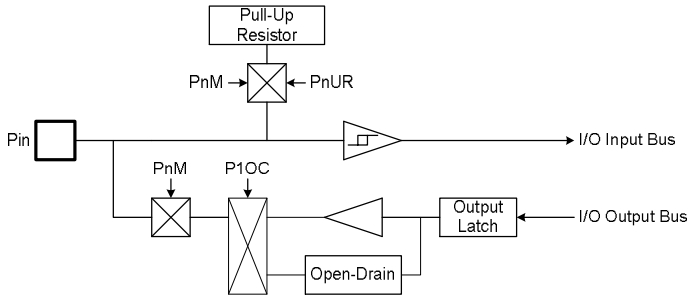
- **P1.1 Reset shared pin structure:**



- **P0, P1, P2, P5 GPIO structure:**



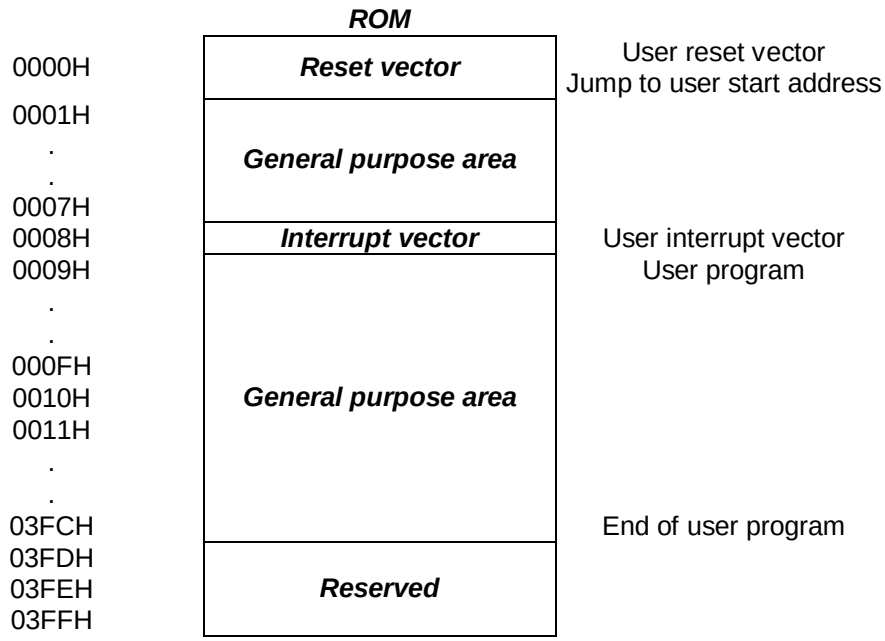
- **Port 1.0 structure:**



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (ROM)

F ***1K words ROM***



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

2.1.1 RESET VECTOR (0000H)

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

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

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

Ø 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.***

Ø 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
```

Ø 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.

Ø 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 à 00), à Y=Y+1
                NOP                     ;
                                        ;
@@:             MOVC                    ; To lookup data, R = 51H, ACC = 05H.
                …                       ;
TABLE1:         DW      0035H           ; To define a word (16 bits) data.
                DW      5105H
                DW      2012H
                …
```

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

Ø Example: INC_YZ macro.

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

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

Ø 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.

Ø 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.***

Ø 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.

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

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

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

Ø 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).

Ø 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

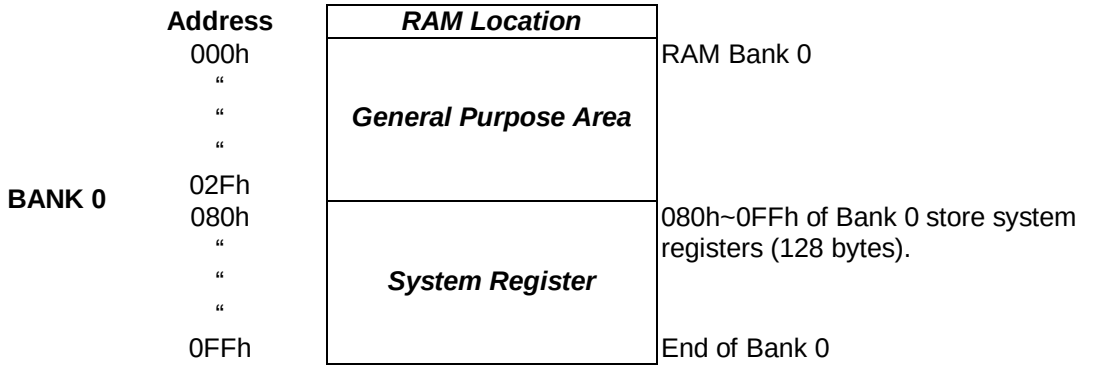
The last ROM address are reserved area. User should avoid these addresses (last address) when calculate the Checksum value.

Ø 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)

F ***48 X 8-bit RAM***



The 48-byte general purpose RAM is separated into Bank 0. Sonix provides “Bank 0” type instructions (e.g. b0mov, b0add, b0bts1, b0bset…) to control Bank 0 RAM directly.

2.2.1 SYSTEM REGISTER

2.2.1.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	-	-	R	Z	Y	-	PFLAG	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	P2	-	-	P5	-	-	T0M	T0C	TC0M	TC0C	-	-	-	STKP
E	P0UR	P1UR	P2UR	-	-	P5UR	-	@YZ	-	P1OC	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 SYSTEM REGISTER DESCRIPTION

R = Working register and ROM look-up data buffer.
PFLAG = Special flag register.
INTRQ = Interrupt request register.
WDTR = Watchdog timer clear register.
PnM = Port n input/output mode register.
PnUR = Port n pull-up resister control register.
PCH, PCL = Program counter.
T0C = T0 counting register.
TC0C = TC0 counting register.
P1OC = P1.0 open-drain control register.
STKP = Stack pointer buffer.

Y, Z = Working, @YZ and ROM addressing register.
PEDGE = P0.0 edge direction register.
INTEN = Interrupt enable register.
Pn = Port n data buffer.
OSCM = Oscillator mode register.
T0M = T0 mode register.
TC0M = TC0 mode register.
TC0R = TC0 auto-reload data buffer.
@YZ = RAM YZ indirect addressing index pointer.
STK0~STK3 = Stack 0 ~ stack 3 buffer.

2.2.1.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD	LVD36	LVD24		C	DC	Z	R/W	PFLAG
0B8H								P00M	R/W	P0M
0BFH				P00G1	P00G0				R/W	PEDGE
0C0H					P13W	P12W	P11W	P10W	W	P1W
0C1H					P13M	P12M		P10M	R/W	P1M
0C2H							P21M	P20M	R/W	P2M
0C5H				P54M					R/W	P5M
0C8H			TC0IRQ	T0IRQ				P00IRQ	R/W	INTRQ
0C9H			TC0IEN	T0IEN				P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH							PC9	PC8	R/W	PCH
0D0H								P00	R/W	P0
0D1H					P13	P12	P11	P10	R/W	P1
0D2H							P21	P20	R/W	P2
0D5H				P54					R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DFH	GIE						STKPB1	STKPB0	R/W	STKP
0E0H								P00R	W	P0UR
0E1H					P13R	P12R		P10R	W	P1UR
0E2H							P21R	P20R	W	P2UR
0E5H				P54R					W	P5UR
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E9H								P10OC	W	P1OC
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H							S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH							S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH							S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH							S0PC9	S0PC8	R/W	STK0H

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

2.2.2 ACCUMULATOR

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

Ø 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.

Ø Example: Protect ACC and working registers.

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

                RETI                ; Exit interrupt service vector
```

2.2.3 PROGRAM FLAG

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

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

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

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

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

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

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

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

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

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

2.2.4 PROGRAM COUNTER

The program counter (PC) is a 10-bit binary counter separated into the high-byte 2 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.
Besides, it can be replaced with specific address by executing CALL or JMP instruction. When JMP or CALL instruction is executed, the destination address will be inserted to bit 0 ~ bit 9.

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

F 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
```

F 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.***

Ø 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
            …
```

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

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

2.2.5 Y, Z REGISTERS

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

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

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

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

Ø 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
```

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

```
        B0MOV   Y, #0           ; Y = 0, bank 0
        B0MOV   Z, #07FH        ; Z = 7FH, the last address of the data memory area
CLR_YZ_BUF:
        CLR     @YZ             ; Clear @YZ to be zero
        DECMS   Z               ; Z – 1, if Z= 0, finish the routine
        JMP     CLR_YZ_BUF      ; Not zero
        CLR     @YZ
END_CLR:                        ; End of clear general purpose data memory area of bank 0
        …
```

2.2.6 R REGISTER

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

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

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

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

2.3 ADDRESSING MODE

2.3.1 IMMEDIATE ADDRESSING MODE

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

Ø Example: Move the immediate data 12H to ACC.

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

Ø 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.

Ø 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.
```

Ø 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).

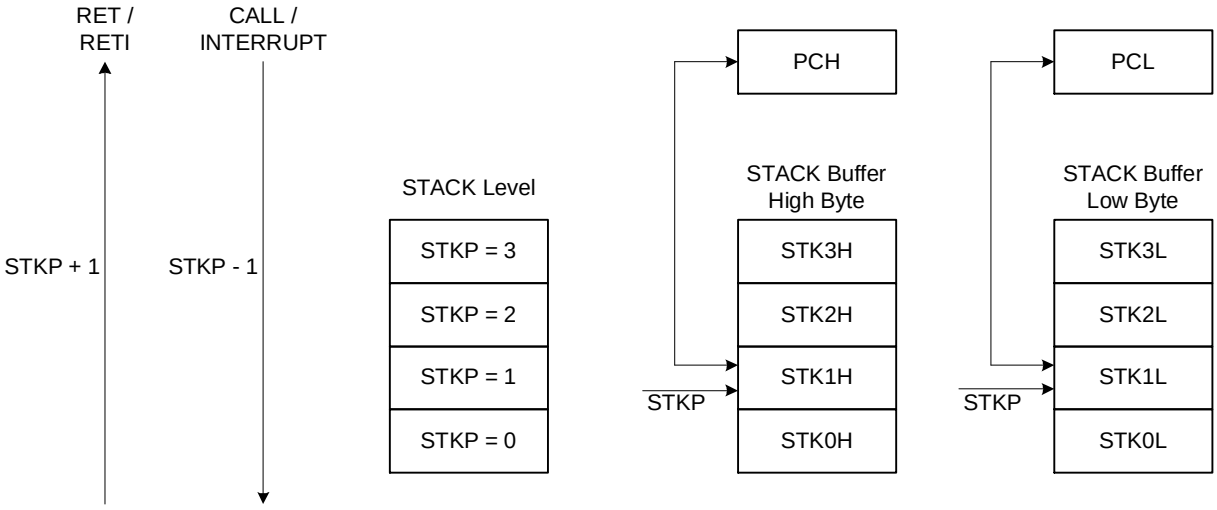
Ø Example: Indirectly addressing mode with @YZ register.

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

2.4 STACK OPERATION

2.4.1 OVERVIEW

The stack buffer has 4-level. These buffers are designed to push and pop up program counter's (PC) data when interrupt service routine and "CALL" instruction are executed. The STKP register is a pointer designed to point active level in order to push or pop up data from stack buffer. The STKnH and STKnL are the stack buffers to store program counter (PC) data.



2.4.2 STACK REGISTERS

The stack pointer (STKP) is a 2-bit register to store the address used to access the stack buffer, 9-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

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

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

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

Bit[2:0] **STKPBn:** Stack pointer (n = 0 ~ 1)

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

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

```
MOV     A, #00000011B
B0MOV   STKP, A
```

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

0F0H~0F8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnL	SnPC7	SnPC6	SnPC5	SnPC4	SnPC3	SnPC2	SnPC1	SnPC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

STKn = STKnH , STKnL (n = 3 ~ 0)

2.4.3 STACK OPERATION EXAMPLE

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

Stack Level	***STKP Register***		***Stack Buffer***		***Description***
	STKPB1	**STKPB0**	**High Byte**	**Low Byte**	
0	1	1	Free	Free	-
1	1	0	STK0H	STK0L	-
2	0	1	STK1H	STK1L	-
3	0	0	STK2H	STK2L	-
4	1	1	STK3H	STK3L	-
> 4	1	0	-	-	**Stack Over, error**

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

Stack Level	***STKP Register***		***Stack Buffer***		***Description***
	STKPB1	**STKPB0**	**High Byte**	**Low Byte**	
4	1	1	STK3H	STK3L	-
3	0	0	STK2H	STK2L	-
2	0	1	STK1H	STK1L	-
1	1	0	STK0H	STK0L	-
0	1	1	Free	Free	-

2.5 CODE OPTION TABLE

The code option is the system hardware configurations including system clock rate, 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 1 oscillator clocks.
	Fhosc/2	Instruction cycle is 2 oscillator clocks.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
	Fhosc/16	Instruction cycle is 16 oscillator clocks.
	Fhosc/32	Instruction cycle is 32 oscillator clocks.
	Fhosc/64	Instruction cycle is 64 oscillator clocks.
	Fhosc/128	Instruction cycle is 128 oscillator clocks.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
Reset_Pin	Reset	Enable External reset pin.
	P11	Enable P1.1 input only without pull-up resister.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
Low_Power	Enable	Enable low power to reduce operating current.
	Disable	Disable low power option.
LVD	LVD_L	LVD will reset chip if VDD is below 2.0V
	LVD_M	LVD will reset chip if VDD is below 2.0V Enable LVD24 bit of PFLAG register for 2.4V low voltage indicator.
	LVD_H	LVD will reset chip if VDD is below 2.4V Enable LVD36 bit of PFLAG register for 3.6V low voltage indicator.
	LVD_MAX	LVD will reset chip if VDD is below 3.6V

2.5.1 Fcpu code option

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

2.5.2 Reset_Pin code option

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

- **Reset:** The reset pin is external reset function. When falling edge trigger occurring, the system will be reset.
- **P11:** Set reset pin to general input only pin (P1.1). The external reset function is disabled and the pin is input pin.

2.5.3 Security code option

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

2.5.4 Low_Power code option

The Low_Power code option can reduce operating current and only support system clock rate less than 4mips.

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

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

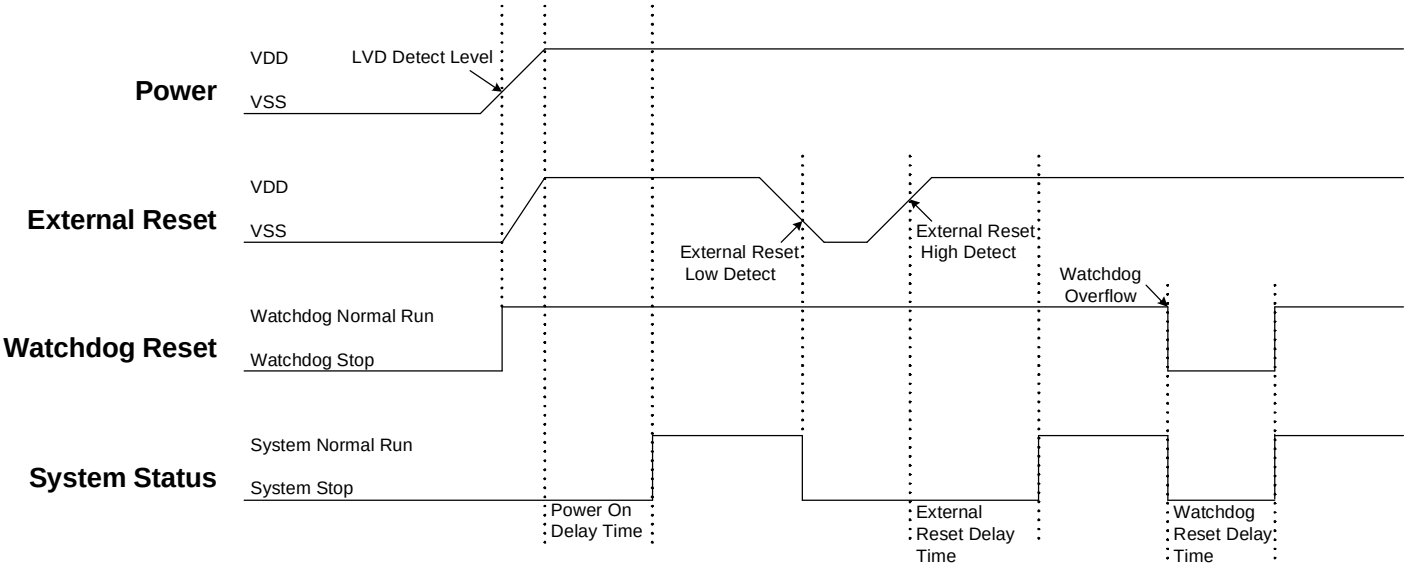
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

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

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

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

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



3.2 POWER ON RESET

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

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

3.3 WATCHDOG RESET

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

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

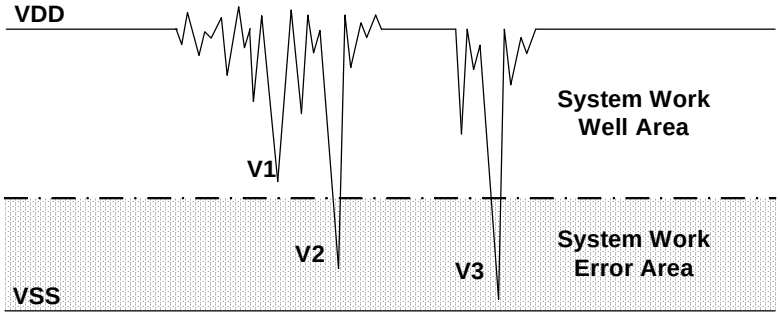
Watchdog timer application note is as following.

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

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

3.4 BROWN OUT RESET

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



Brown Out Reset Diagram

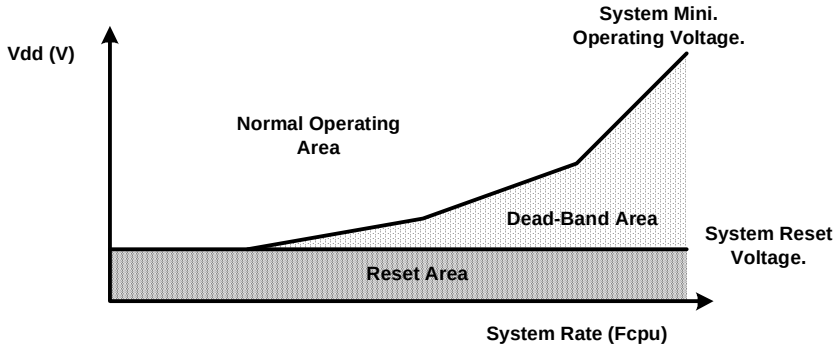
The power dropping might through the voltage range that's the system dead-band. The dead-band means the power range can't offer the system minimum operation power requirement. The above diagram is a typical brown out reset diagram. There is a serious noise under the VDD, and VDD voltage drops very deep.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

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

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

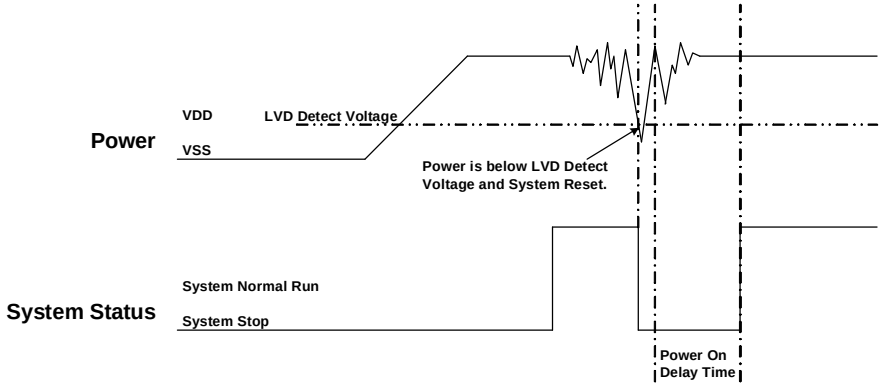
3.5 THE SYSTEM OPERATING VOLTAGE

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



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

3.6 LOW VOLTAGE DETECTOR (LVD)



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

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

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

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

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

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

LVD_L
If VDD < 2.0V, system will be reset.
Disable LVD24 and LVD36 bit of PFLAG register.

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

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

LVD_MAX
If VDD < 3.6V, system will be reset.

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

3.7 BROWN OUT RESET IMPROVEMENT

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

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

- ***Note:***
  1. ***The " Zener diode reset circuit", "Voltage bias reset circuit" and "External reset IC" can completely improve the brown out reset, DC low battery and AC slow power down conditions.***
  2. ***For AC power application and enhance EFT performance, the system clock is 16MHz/16 (1 mips) and use external reset (" Zener diode reset circuit", "Voltage bias reset circuit", "External reset IC"). The structure can improve noise effective and get good EFT characteristic.***

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

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

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

3.8 EXTERNAL RESET

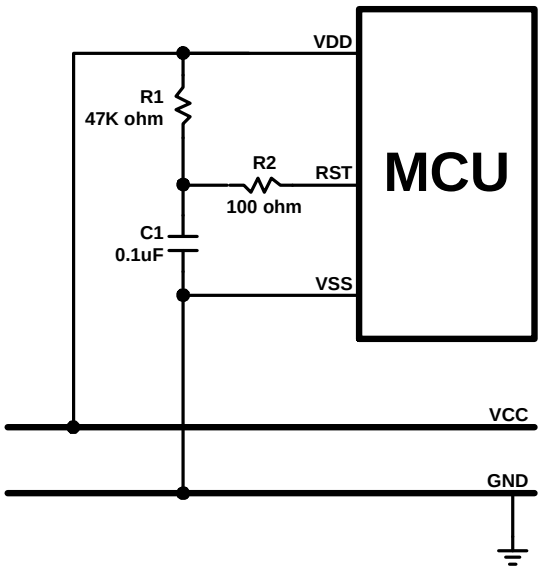
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

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

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

3.9 EXTERNAL RESET CIRCUIT

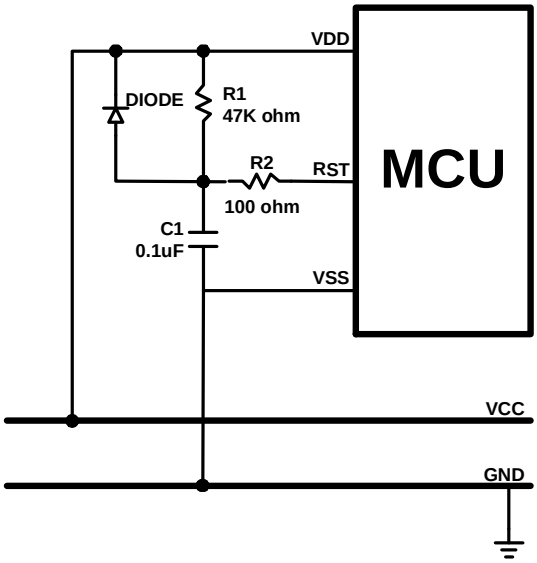
3.9.1 Simply RC Reset Circuit



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

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

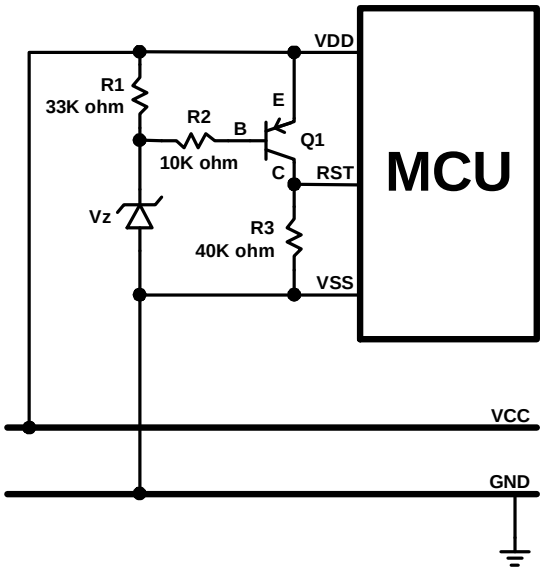
3.9.2 Diode & RC Reset Circuit



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

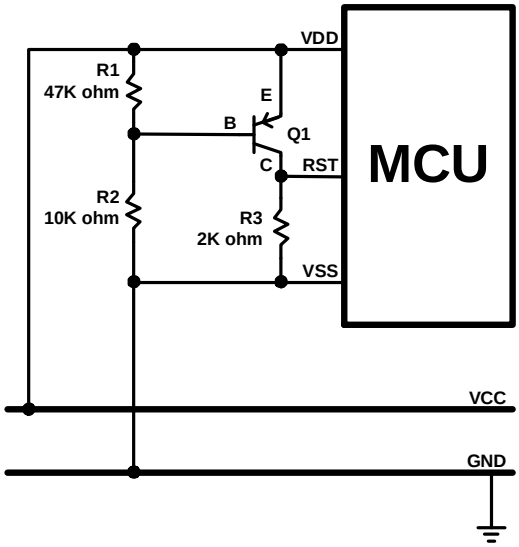
- ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.9.3 Zener Diode Reset Circuit



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

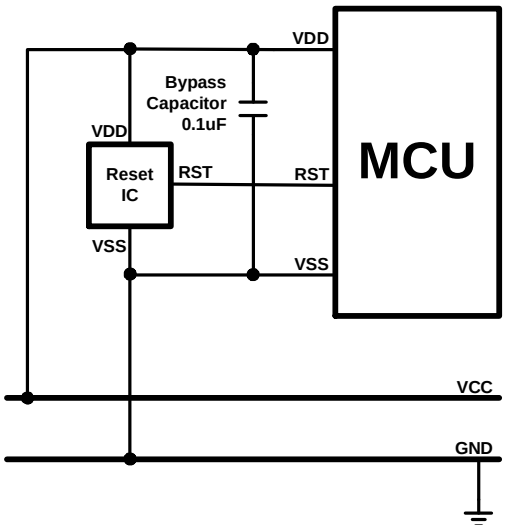
3.9.4 Voltage Bias Reset Circuit



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

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

3.9.5 External Reset IC



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

4 SYSTEM CLOCK

4.1 OVERVIEW

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

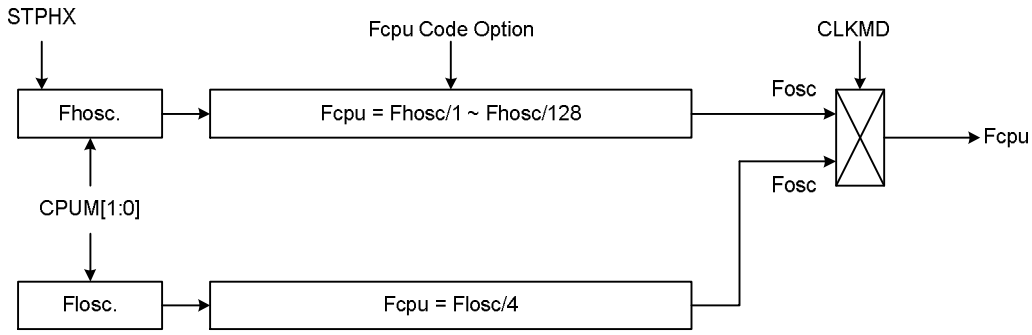
- **High-speed oscillator**

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

- **Low-speed oscillator**

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

- **System clock block diagram**



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

4.2 FCPU (INSTRUCTION CYCLE)

The system clock rate is instruction cycle called "**Fcpu**" which is divided from the system clock source and decides the system operating rate. Fcpu rate is selected by Fcpu code option and the range is **Fhosc/1~Fhosc/128** under system normal mode. If the Fcpu code option is Fhosc/4, the Fcpu frequency is 16MHz/4 = 4MHz. Under system slow mode, the Fcpu is fixed Flosc/4, 16KHz/4=4KHz @3V, 32KHz/4=8KHz @5V.

4.3 SYSTEM HIGH-SPEED CLOCK

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

4.4 SYSTEM LOW-SPEED CLOCK

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

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

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

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

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

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

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

4.5 OSCM REGISTER

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

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	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:** High-speed oscillator control bit.
0 = The high-speed oscillator free run.
1 = The high-speed oscillator stops. Internal low-speed RC oscillator is still running.

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

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

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

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

4.6 SYSTEM CLOCK MEASUREMENT

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

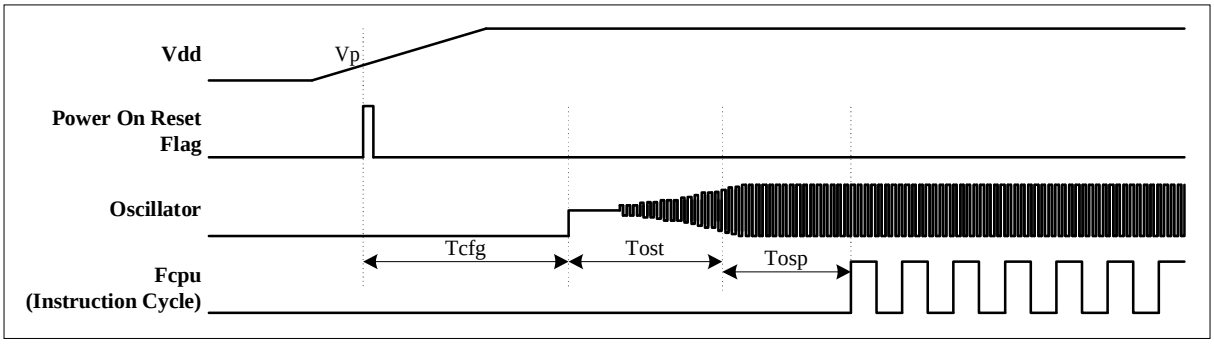
Ø Example: Fcpu instruction cycle of internal oscillator.

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

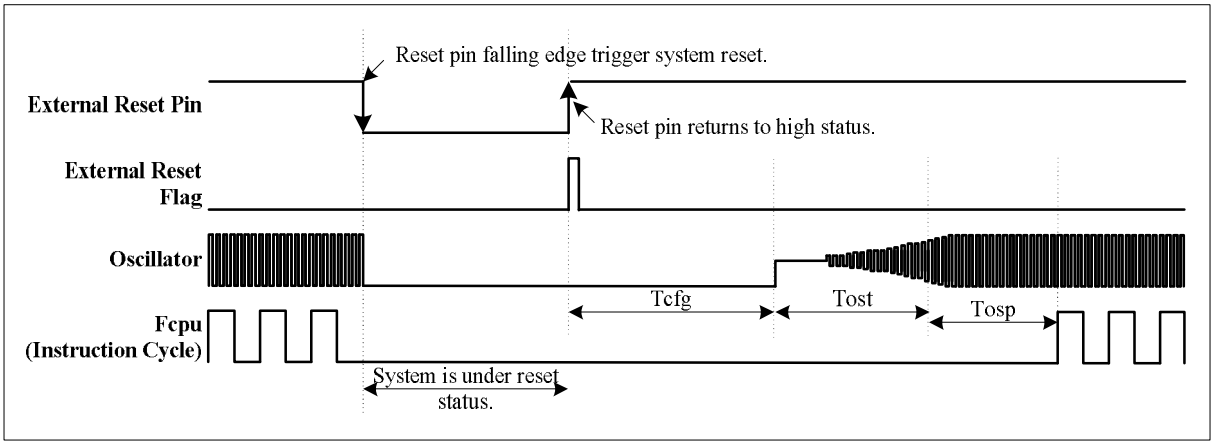
4.7 SYSTEM CLOCK TIMING

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

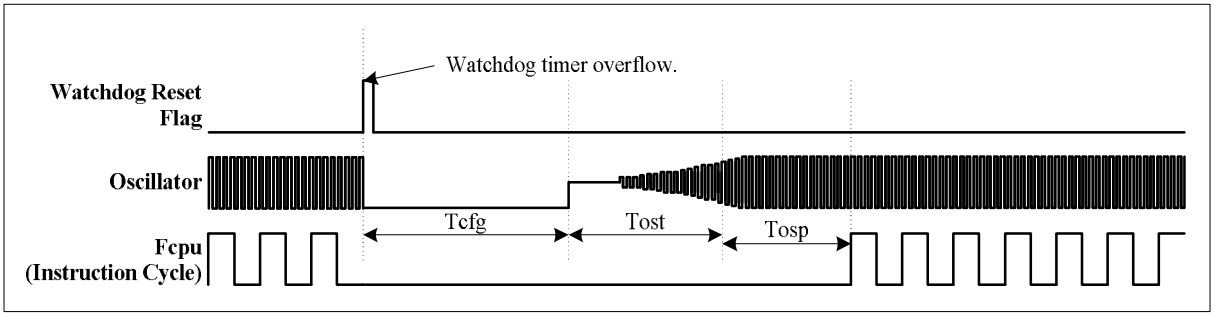
l Power On Reset Timing



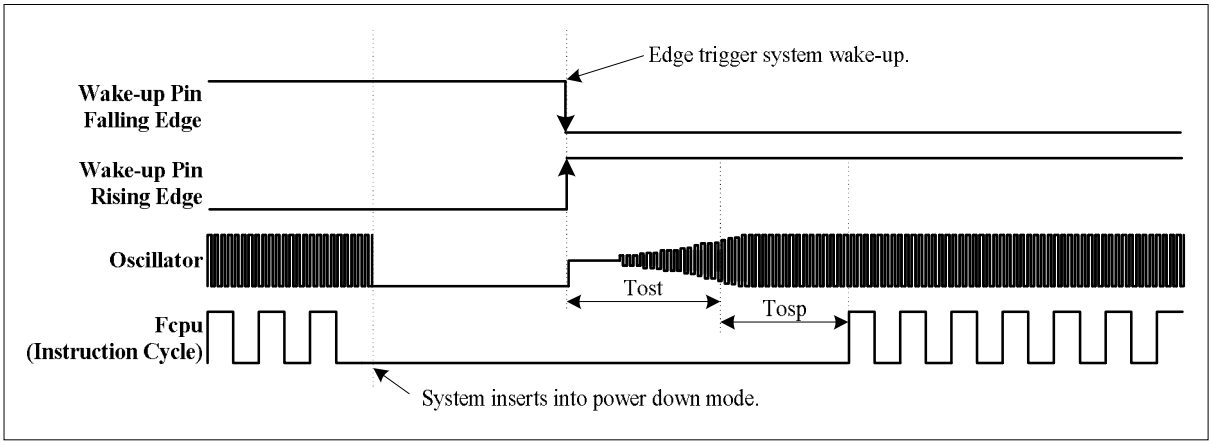
l External Reset Pin Reset Timing



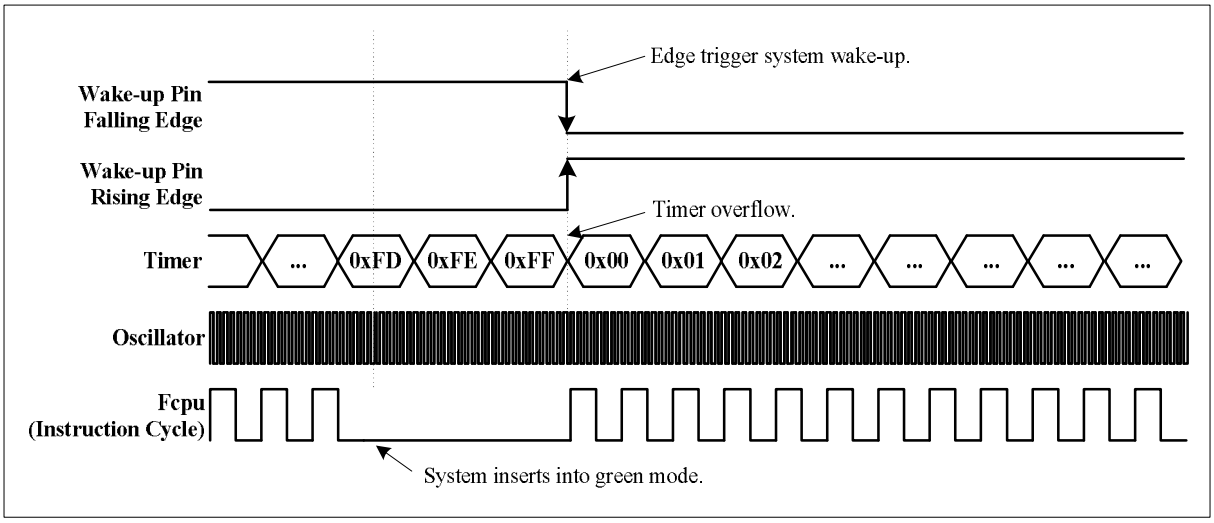
l Watchdog Reset Timing



I Power Down Mode Wake-up Timing

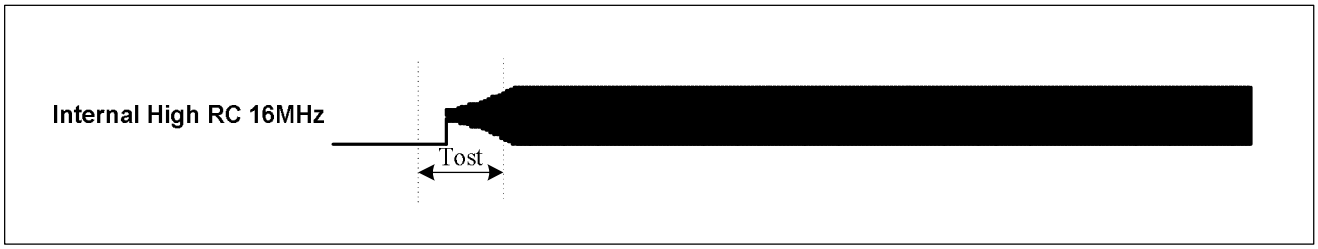


I Green Mode Wake-up Timing



I Oscillator Start-up Time

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



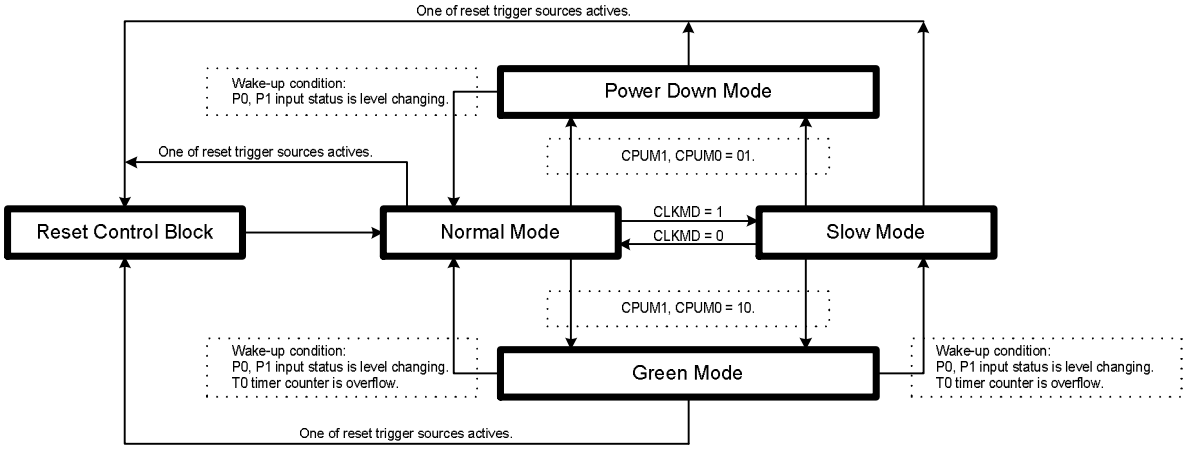
5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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

Operating Mode Control Block



Operating Mode Clock Control Table

Operating Mode	Normal Mode	Slow Mode	Green Mode	Power Down Mode
IHRC	Running	By STPHX	By STPHX	Stop
ILRC	Running	Running	Running	Stop
CPU instruction	Executing	Executing	Stop	Stop
T0 timer	By T0ENB	By T0ENB	By T0ENB	Inactive
TC0 timer	By TC0ENB	By TC0ENB	By TC0ENB (PWM active)	Inactive
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option
Internal interrupt	All active	All active	T0, TC0	All inactive
External interrupt	All active	All active	All active	All inactive
Wakeup source	-	-	P0, P1, T0, Reset	P0, P1, Reset

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

5.2 NORMAL MODE

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

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

5.3 SLOW MODE

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

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

5.4 POWER DOWN MDOE

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

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

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

5.5 GREEN MODE

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

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

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

5.6 OPERATING MODE CONTROL MACRO

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

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

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

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

Ø Example: Switch normal mode to slow mode.

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

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

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

Ø Example: Switch normal/slow mode to green mode.

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

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

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

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

5.7 WAKEUP

5.7.1 OVERVIEW

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

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

5.7.2 WAKEUP TIME

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

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

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

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

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

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

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

5.7.3 P1W WAKEUP CONTROL REGISTER

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

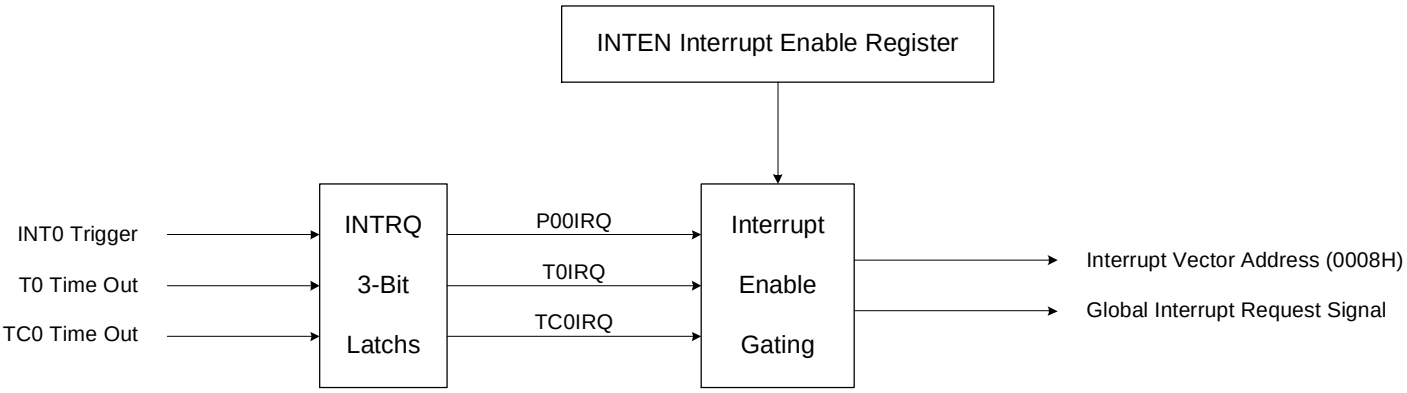
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	-	-	-	-	P13W	P12W	P11W	P10W
Read/Write	-	-	-	-	W	W	W	W
After reset	-	-	-	-	0	0	0	0

Bit[3:0] **P10W~P13W:** Port 1 wakeup function control bits.
0 = Disable P1n wakeup function.
1 = Enable P1n wakeup function.

6 INTERRUPT

6.1 OVERVIEW

This MCU provides three interrupt sources, including two internal interrupt (T0/TC0) and one external interrupt (INT0). The external interrupt can wakeup the chip while the system is switched from power down mode to high-speed 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. All of the interrupt request signals are stored in INTRQ register.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

INTEN is the interrupt request control register including one internal interrupts, one 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	-	-	TC0IEN	T0IEN	-	-	-	P00IEN
Read/Write	-	-	R/W	R/W	-	-	-	R/W
After reset	-	-	0	0	-	-	-	0

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

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

Bit 5 **TC0IEN:** TC0 timer interrupt control bit.
0 = Disable TC0 interrupt function.
1 = Enable TC0 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	-	-	TC0IRQ	T0IRQ	-	-	-	P00IRQ
Read/Write	-	-	R/W	R/W	-	-	-	R/W
After reset	-	-	0	0	-	-	-	0

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

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

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

6.4 GIE GLOBAL INTERRUPT OPERATION

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

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

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

n 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.***

Ø 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 INT0 (P0.0) INTERRUPT OPERATION

When the INT0 trigger occurs, the P00IRQ will be set to "1" no matter the P00IEN is enable or disable. If the P00IEN = 1 and the trigger event P00IRQ is also set to be "1". As the result, the system will execute the interrupt vector (ORG 8). If the P00IEN = 0 and the trigger event P00IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the P00IRQ is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

- ***Note: The interrupt trigger direction of P0.0 is control by PEDGE register.***

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

Bit[4:3] **P00G[1:0]:** P0.0 interrupt trigger edge control bits.
00 = reserved.
01 = rising edge.
10 = falling edge.
11 = rising/falling bi-direction (Level change trigger).

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

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

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

Ø Example: INT0 interrupt service routine.

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

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

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

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

        RETI                    ; Exit interrupt vector
```

6.7 T0 INTERRUPT OPERATION

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

Ø Example: T0 interrupt request setup. Fcpu = 16MHz / 16.

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

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

        B0BSET      FGIE            ; Enable GIE
```

Ø Example: T0 interrupt service routine.

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

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

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

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


        RETI                        ; Exit interrupt vector
```

6.8 TC0 INTERRUPT OPERATION

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

Ø Example: TC0 interrupt request setup.

```
        B0BCLR      FTC0IEN     ; Disable TC0 interrupt service
        B0BCLR      FTC0ENB     ; Disable TC0 timer
        MOV         A, #20H     ;
        B0MOV       TC0M, A     ; Set TC0 clock = Fcpu / 64
        MOV         A, #74H     ; Set TC0C initial value = 74H
        B0MOV       TC0C, A     ; Set TC0 interval = 10 ms

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

        B0BSET      FGIE        ; Enable GIE
```

Ø Example: TC0 interrupt service routine.

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

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

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

        B0BCLR      FTC0IRQ     ; Reset TC0IRQ
        MOV         A, #74H
        B0MOV       TC0C, A     ; Reset TC0C.
        …                       ; TC0 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.9 MULTI-INTERRUPT OPERATION

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

Interrupt Name	*Trigger Event Description*
P00IRQ	P0.0 trigger controlled by PEDGE
T0IRQ	T0C overflow
TC0IRQ	TC0C overflow

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.

Ø 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          ; Jump to INT0 interrupt service routine
INTT0CHK:                                   ; Check T0 interrupt request
                B0BTS1      FT0IEN          ; Check T0IEN
                JMP         INTTC0CHK       ; Jump check to next interrupt
                B0BTS0      FT0IRQ          ; Check T0IRQ
                JMP         INTT0           ; Jump to T0 interrupt service routine
INTTC0CHK:                                  ; Check TC0 interrupt request
                B0BTS1      FTC0IEN         ; Check TC0IEN
                JMP         INT_EXIT        ; Jump to exit of IRQ
                B0BTS0      FTC0IRQ         ; Check TC0IRQ
                JMP         INTTC0          ; Jump to TC0 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 8 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
P1.1	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P5.4	I/O	BZ0/PWM0	DC	TC0ENB=1, TC0OUT=1 or PWM0OUT=1

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

7.2 I/O PORT MODE

The port direction is programmed by PnM register. When the bit of PnM register is "0", the pin is input mode. When the bit of PnM register is "1", the pin is output mode.

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

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

0C2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2M	-	-	-	-	-	-	P21M	P20M
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	-	P54M	-	-	-	-
Read/Write	-	-	-	R/W	-	-	-	-
After reset	-	-	-	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. ***P1.1 input only pin, and the P1M.1 is undefined.***

Ø Example: I/O mode selecting

```
CLR     P0M         ; Set all ports to be input mode.
CLR     P2M
CLR     P5M

MOV     A, #0FFH    ; Set all ports to be output mode.
B0MOV   P0M, A
B0MOV   P2M,A
B0MOV   P5M, A

B0BCLR  P2M.0       ; Set P2.0 to be input mode.

B0BSET  P2M.0       ; Set P2.0 to be output mode.
```

7.3 I/O PULL UP REGISTER

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

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

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

0E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2UR	-	-	-	-	-	-	P21R	P20R
Read/Write	-	-	-	-	-	-	W	W
After reset	-	-	-	-	-	-	0	0

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	-	-	-	P54R	-	-	-	-
Read/Write	-	-	-	W	-	-	-	-
After reset	-	-	-	0	-	-	-	-

- ***Note: P1.1 is input only pin and without pull-up resister. The P1UR.1 is undefined.***

Ø Example: I/O Pull up Register

```
MOV     A, #0FFH    ; Enable Port0, 2, 5 Pull-up register,
B0MOV   P0UR, A     ;
B0MOV   P2UR,A
B0MOV   P5UR, A
```

7.4 I/O PORT DATA REGISTER

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

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

0D2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2	-	-	-	-	-	-	P21	P20
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	-	P54	-	-	-	-
Read/Write	-	-	-	R/W	-	-	-	-
After reset	-	-	-	0	-	-	-	-

- ***Note: The P11 keeps "1" when external reset enable by code option.***

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

```
B0MOV   A, P0       ; Read data from Port 0
B0MOV   A, P2       ; Read data from Port 2
B0MOV   A, P5       ; Read data from Port 5
```

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

```
MOV     A, #0FFH    ; Write data FFH to all Port.
B0MOV   P0, A
B0MOV   P2, A
B0MOV   P5, A
```

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

```
B0BSET  P2.0        ; Set P2.0 and P1.3 to be "1".
B0BSET  P1.3

B0BCLR  P2.0        ; Set P2.0 and P1.3 to be "0".
B0BCLR  P1.3
```

8 TIMERS

8.1 WATCHDOG TIMER

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

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

VDD	*Internal Low RC Freq.*	*Watchdog Overflow Time*
3V	16KHz	512ms
5V	32KHz	256ms

The watchdog timer has three operating options controlled "WatchDog" code option.

- **Disable:** Disable watchdog timer function.
- **Enable:** Enable watchdog timer function. Watchdog timer actives in normal mode and slow mode. In power down mode and green mode, the watchdog timer stops.
- **Always_On:** Enable watchdog timer function. The watchdog timer actives and not stop in power down mode and green mode.

In high noisy environment, the "Always_On" option of watchdog operations is the strongly recommendation to make the system reset under error situations and re-start again.

Watchdog clear is controlled by WDTR register. Moving **0x5A** data into WDTR is to reset watchdog timer.

0CCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Ø 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
```

Ø Example: Clear watchdog timer by "@RST_WDT" macro of Sonix IDE.

```
Main:
        @RST_WDT                ; Clear the watchdog timer.
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     MAIN
```

Watchdog timer application note is as following.

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

Ø Example: An operation of watchdog timer is as following. To clear the watchdog timer counter in the top of the main routine of the program.

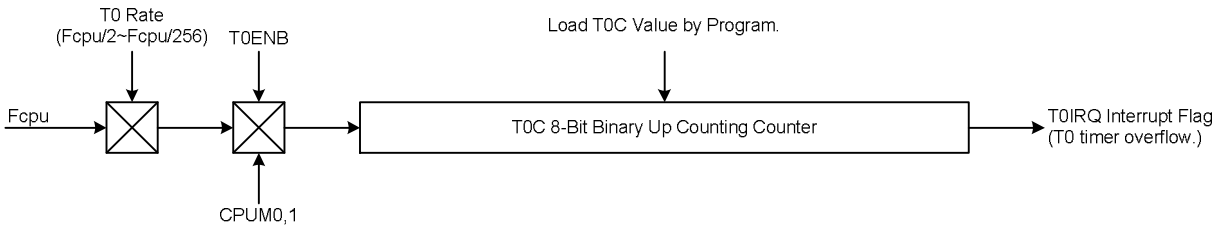
```
Main:
                …                               ; Check I/O.
                …                               ; Check RAM
Err:            JMP $                           ; I/O or RAM error. Program jump here and don't
                                                ; clear watchdog. Wait watchdog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

Correct:                                        ; I/O and RAM are correct. Clear watchdog timer and
                                                ; execute program.
                MOV         A, #5AH             ; Clear the watchdog timer.
                B0MOV       WDTR, A
                …
                CALL        SUB1
                CALL        SUB2
                …
                …
                …
                JMP         MAIN
```

8.2 T0 8-BIT BASIC TIMER

8.2.1 OVERVIEW

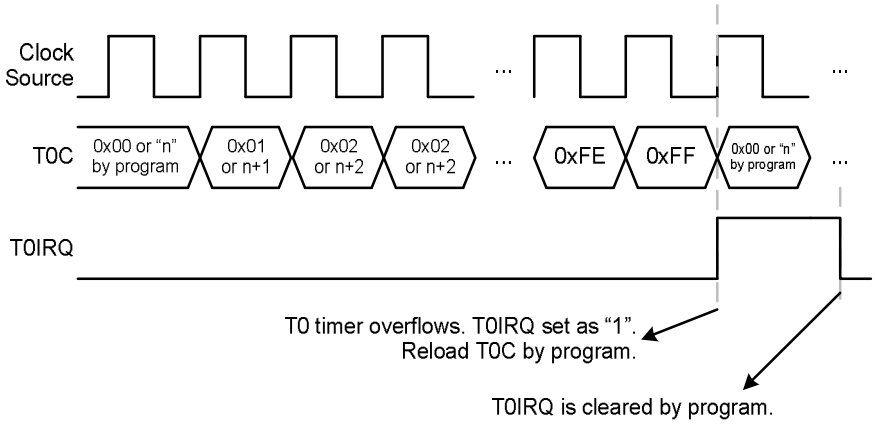
The T0 timer is an 8-bit binary up timer with basic timer function. The basic timer function supports flag indicator (T0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T0M, T0C registers. The T0 builds in green mode wake-up function. When T0 timer overflow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

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



8.2.2 T0 TIMER OPERATION

T0 timer is controlled by T0ENB bit. When T0ENB=0, T0 timer stops. When T0ENB=1, T0 timer starts to count. T0C increases “1” by timer clock source. When T0 overflow event occurs, T0IRQ flag is set as ”1” to indicate overflow and cleared by program. The overflow condition is T0C count from full scale (0xFF) to zero scale (0x00). T0 doesn’t build in double buffer, so load T0C by program when T0 timer overflows to fix the correct interval time. If T0 timer interrupt function is enabled (T0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 8) and executes interrupt service routine after T0 overflow occurrence. Clear T0IRQ by program is necessary in interrupt procedure. T0 timer can works in normal mode, slow mode and green mode. In green mode, T0 keeps counting, set T0IRQ and wakes up system when T0 timer overflows.



T0 clock source is Fcpu (instruction cycle) through T0rate[2:0] pre-scaler to decide Fcpu/2~Fcpu/256. T0 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

T0rate[2:0]	T0 Clock	T0 Interval Time Fhosc=16MHz, Fcpu=Fhosc/2	
		max. (ms)	Unit (us)
000b	Fcpu/256	8.192	32
001b	Fcpu/128	4.096	16
010b	Fcpu/64	2.048	8
011b	Fcpu/32	1.024	4
100b	Fcpu/16	0.576	2.25
101b	Fcpu/8	0.256	1
110b	Fcpu/4	0.128	0.5
111b	Fcpu/2	0.064	0.25

8.2.3 T0M MODE REGISTER

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

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

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

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

8.2.4 T0C COUNTING REGISTER

T0C is T0 8-bit counter. When T0C overflow occurs, the T0IRQ flag is set as “1” and cleared by program. The T0C decides T0 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T0C register, and then enable T0 timer to make sure the fist cycle correct. After one T0 overflow occurs, the T0C register is loaded a correct value by program.

0D9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0C	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of T0C initial value is as following.

T0C initial value = 256 - (T0 interrupt interval time * T0 clock rate)

Ø **Example: To calculation T0C to obtain 10ms T0 interval time. T0 clock source is Fcpu = 16MHz/16 = 1MHz. Select T0RATE=001 (Fcpu/128).**
T0 interval time = 10ms. T0 clock rate = 16MHz/16/128

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

8.2.5 T0 TIMER OPERATION EXPLAME

I T0 TIMER CONFIGURATION:

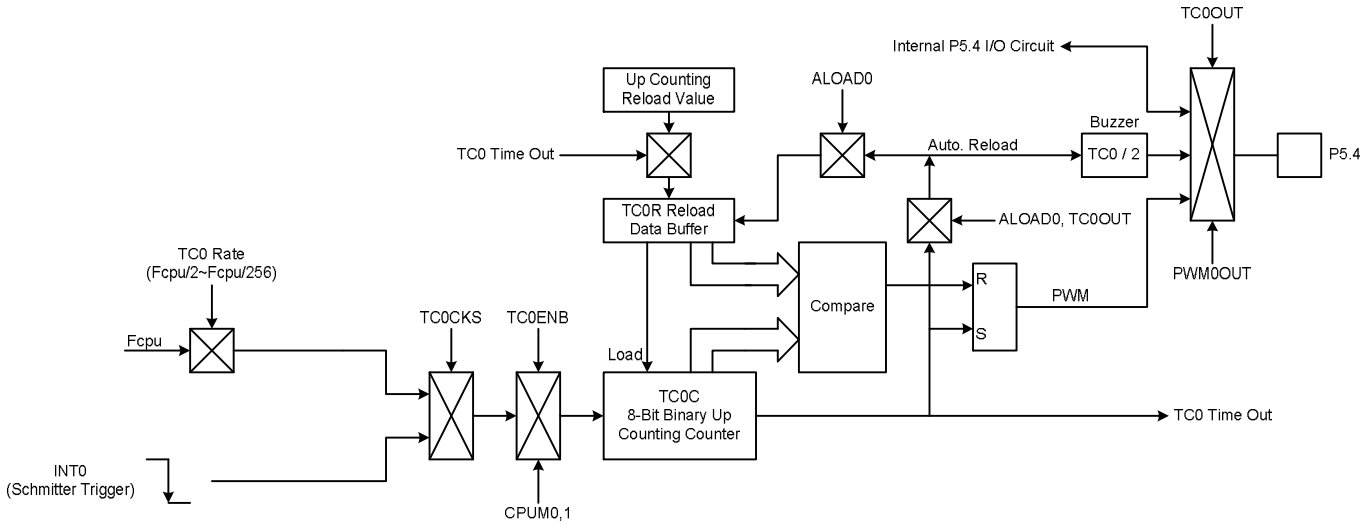
```
; Reset T0 timer.
        CLR     T0M             ; Clear T0M register.

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

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

; Clear T0IRQ
        B0BCLR  FT0IRQ

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

8.3 TC0 8-BIT TIMER/COUNTER

8.3.1 OVERVIEW

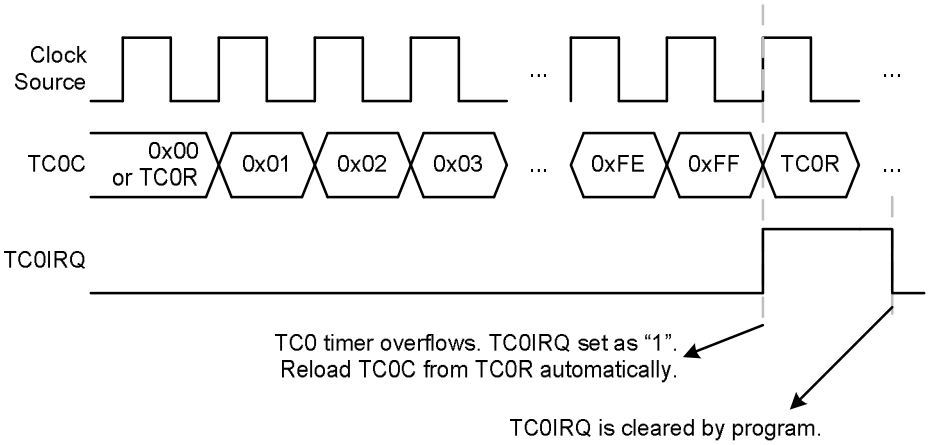
The TC0 timer is an 8-bit binary up timer with basic timer, event counter, buzzer and PWM functions. The basic timer function supports flag indicator (TC0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TC0M, TC0C, TC0R registers. The event counter is changing TC0 clock source from system clock (Fcpu) to external clock like signal (e.g. continuous pulse, R/C type oscillating signal…). TC0 becomes a counter to count external clock number to implement measure application. TC0 also builds in buzzer and PWM functions. The cycle/resolution of buzzer and PWM are controlled by TC0 timer clock rate and TC0R registers, so the buzzer and PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster…The main purposes of the TC0 timer are as following.

- **F** **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **F** **Interrupt function:** TC0 timer function supports interrupt function. When TC0 timer occurs overflow, the TC0IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **F** **Event Counter:** The event counter function counts the external clock counts.
- **F** **PWM output:** The PWM is duty/cycle programmable controlled by T0rate and TC0R registers.
- **F** **Buzzer output:** The Buzzer output signal is 1/2 cycle of TC0 interval time.
- **F** **Green mode function:** All TC0 functions (timer, PWM, Buzzer, event counter, auto-reload) keep running in green mode and no wake-up function.



8.3.2 TC0 TIMER OPERATION

TC0 timer is controlled by TC0ENB bit. When TC0ENB=0, TC0 timer stops. When TC0ENB=1, TC0 timer starts to count. Before enabling TC0 timer, setup TC0 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…TC0C increases "1" by timer clock source. When TC0 overflow event occurs, TC0IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is TC0C count from full scale (0xFF) to zero scale (0x00). In difference function modes, TC0C value relates to operation. If TC0C value changing effects operation, the transition of operations would make timer function error. So TC0 builds in double buffer to avoid these situations happen. The double buffer concept is to flash TC0C during TC0 counting, to set the new value to TC0R (reload buffer), and the new value will be loaded from TC0R to TC0C after TC0 overflow occurrence automatically. In the next cycle, the TC0 timer runs under new conditions, and no any transitions occur. The auto-reload function is controlled by ALOAD0 bit in timer/counter mode, and enabled automatically in PWM mode as TC0 enables. If TC0 timer interrupt function is enabled (TC0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 8) and executes interrupt service routine after TC0 overflow occurrence. Clear TC0IRQ by program is necessary in interrupt procedure. TC0 timer can works in normal mode, slow mode and green mode. But in green mode, TC0 keep counting, set TC0IRQ and outputs PWM, but can't wake-up system.



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

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

8.3.3 TC0M MODE REGISTER

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

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **PWM0OUT:** PWM output control bit.
0 = Disable PWM output function, and P5.4 is GPIO mode.
1 = Enable PWM output function, and P5.4 outputs PWM signal. PWM duty controlled by TC0OUT, ALOAD0 bits.

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

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

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

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

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

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

8.3.4 TC0C COUNTING REGISTER

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

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

The equation of TC0C initial value is as following.

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

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

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

8.3.5 TC0R AUTO-RELOAD REGISTER

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

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

The equation of TC0R initial value is as following.

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

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

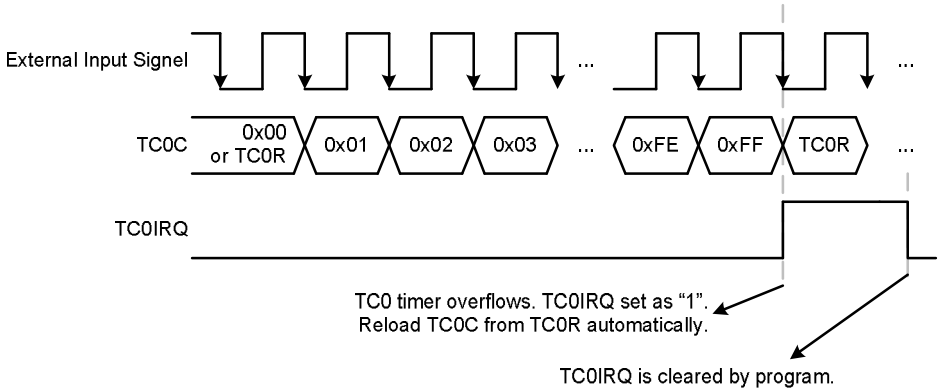
TC0CKS	PWM0	ALOAD0	TC0OUT	N	TC0R valid value	TC0R value binary type
0	0	x	x	256	0x00~0xFF	00000000b~11111111b
	1	0	0	256	0x00~0xFF	00000000b~11111111b
	1	0	1	64	0x00~0x3F	xx000000b~xx111111b
	1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
	1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
1	-	-	-	256	0x00~0xFF	00000000b~11111111b

Ø **Example: To calculation TC0C and TC0R value to obtain 10ms TC0 interval time. TC0 clock source is Fcpu = 16MHz/16 = 1MHz. Select TC0RATE=001 (Fcpu/128).**
TC0 interval time = 10ms. TC0 clock rate = 16MHz/16/128

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

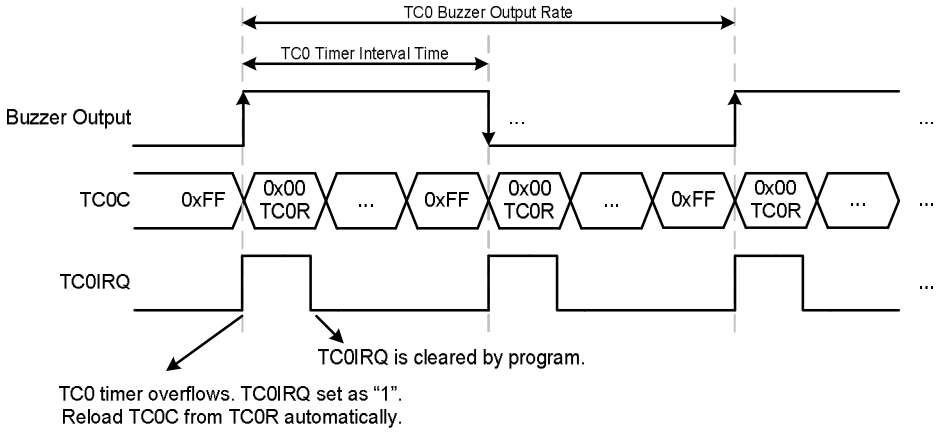
8.3.6 TC0 EVENT COUNTER

TC0 event counter is set the TC0 clock source from external input pin (P0.0). When TC0CKS1=1, TC0 clock source is switch to external input pin (P0.0). TC0 event counter trigger direction is falling edge. When one falling edge occurs, TC0C will up one count. When TC0C counts from 0xFF to 0x00, TC0 triggers overflow event. The external event counter input pin's wake-up function of GPIO mode is disabled when TC0 event counter function enabled to avoid event counter signal trigger system wake-up and not keep in power saving mode. The external event counter input pin's external interrupt function is also disabled when TC0 event counter function enabled, and the P00IRQ bit keeps "0" status. The event counter usually is used to measure external continuous signal rate, e.g. continuous pulse, R/C type oscillating signal…These signal phase don't synchronize with MCU's main clock. Use TC0 event to measure it and calculate the signal rate in program for different applications.

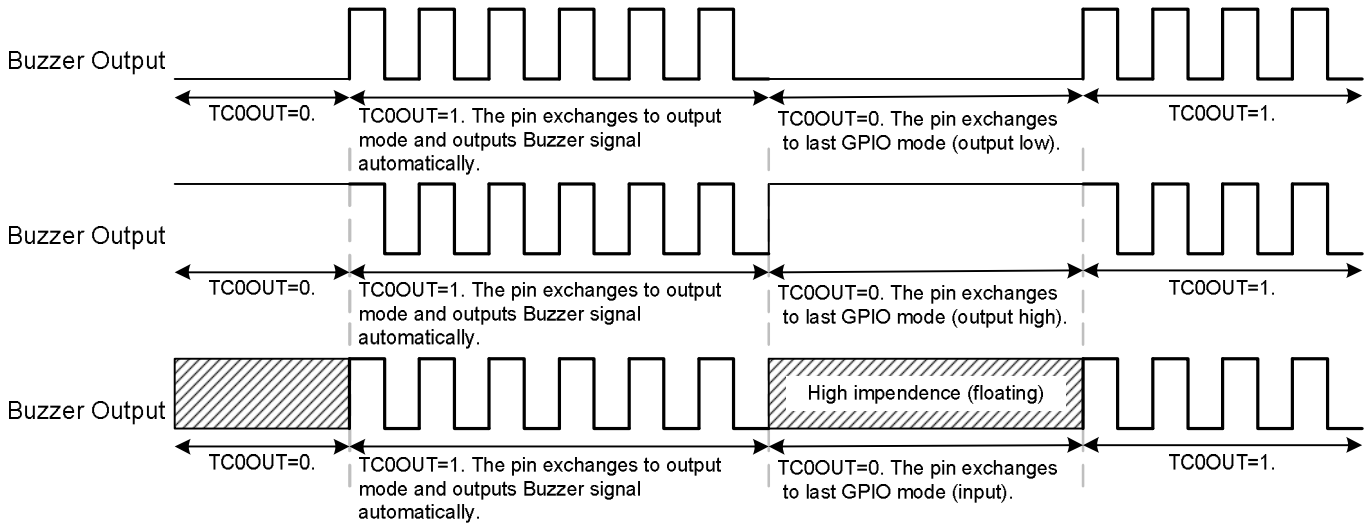


8.3.7 TC0 BUZZER OUTPUT

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



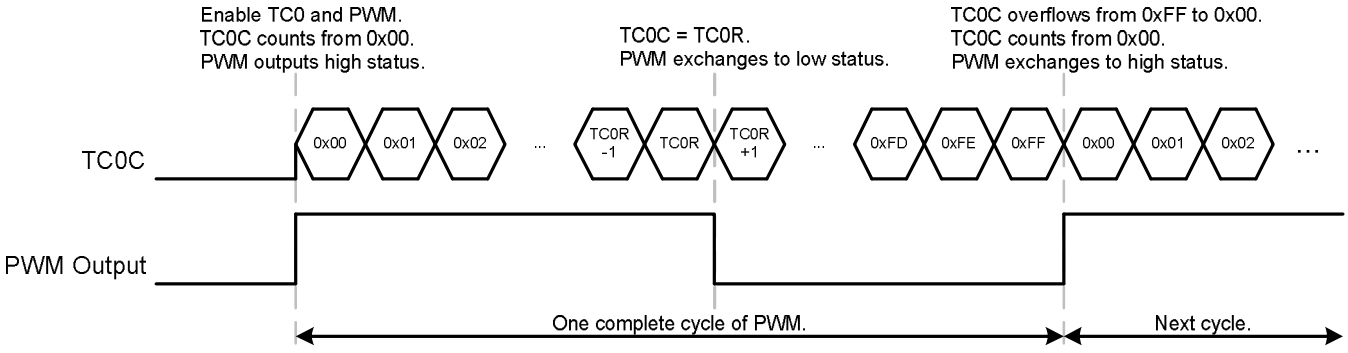
When buzzer outputs, TC0IRQ still actives as TC0 overflows, and TC0 interrupt function actives as TC0IEN = 1. But strongly recommend be careful to use buzzer and TC0 timer together, and make sure both functions work well.
The buzzer output pin is shared with GPIO and switch to output buzzer signal as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



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

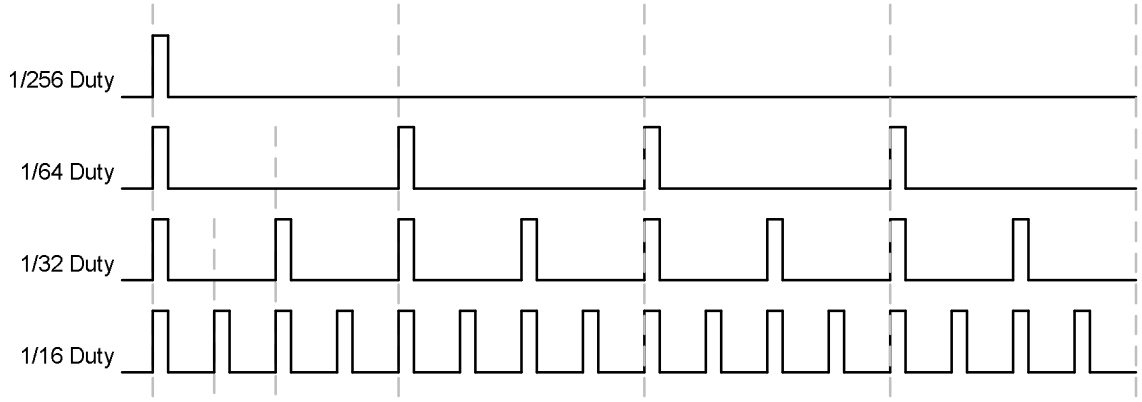
8.3.8 PULSE WIDTH MODULATION (PWM)

The PWM is duty/cycle programmable design to offer various PWM signals. When TC0 timer enables and PWM0OUT bit sets as "1" (enable PWM output), the PWM output pin (P5.4) outputs PWM signal. One cycle of PWM signal is high pulse first, and then low pulse outputs. TC0rate[2:0] bits control the cycle of PWM, ALOAD0 and TC0OUT bits decides the resolution of PWM, and TC0R decides the duty (high pulse width length) of PWM. TC0C initial value is zero when TC0 timer enables and TC0 timer overflows. When TC0C count is equal to TC0R, the PWM high pulse finishes and exchanges to low level. When TC0 overflows (TC0C counts from 0xFF to 0x00), one complete PWM cycle finishes. The PWM exchanges to high level for next cycle. The PWM is auto-reload design to load TC0R when TC0 overflows and the end of PWM's cycle, to keeps PWM continuity. If modify the PWM duty by program as PWM outputting, the new duty occurs at next cycle when TC0R loaded from the reload buffer.

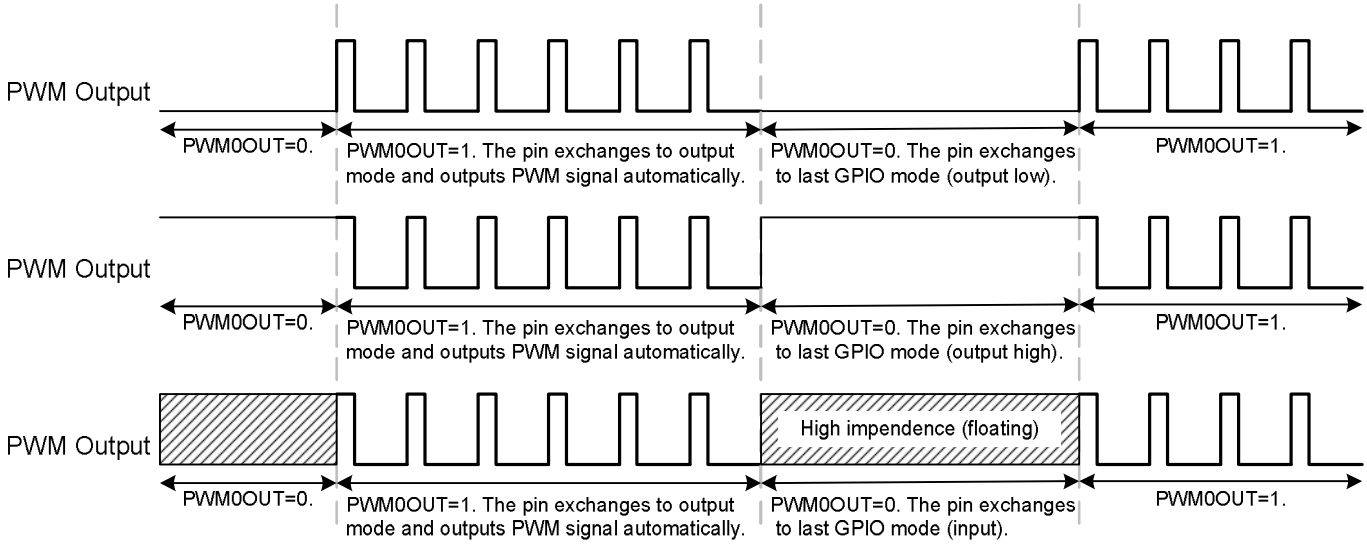


The resolution of PWM includes 1/256, 1/64, 1/32, 1/16 controlled by ALOAD0 and TC0OUT bits to implement high speed PWM signal. ALOAD0, TC0OUT = 00, the PWM resolution is 1/256. ALOAD0, TC0OUT = 01, the PWM resolution is 1/64. ALOAD0, TC0OUT = 10, the PWM resolution is 1/32. ALOAD0, TC0OUT = 11, the PWM resolution is 1/16. If modify the PWM resolution, the TC0R PWM duty control range must be modified to meet resolution. When PWM outputs, TC0IRQ still actives as TC0 overflows, and TC0 interrupt function actives as TC0IEN = 1. But strongly recommend be careful to use PWM and TC0 timer together, and make sure both functions work well.

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



The PWM output pin is shared with GPIO and switch to output PWM signal as PWM0OUT=1 automatically. If PWM0OUT bit is cleared to disable PWM, the output pin exchanges to last GPIO mode automatically. It easily to implement carry signal on/off operation, not to control TC0ENB bit.



8.3.9 TC0 TIMER OPERATION EXPLAME

I TC0 TIMER CONFIGURATION:

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

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

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

; Clear TC0IRQ
        B0BCLR  FTC0IRQ

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

I TC0 EVENT COUNTER CONFIGURATION:

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

; Set TC0 auto-reload function.
        B0BSET  FALOAD0

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

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

; Clear TC0IRQ
        B0BCLR  FTC0IRQ

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

I TC0 BUZZER OUTPUT CONFIGURATION:

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

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

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

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

I TC0 PWM CONFIGURATION:

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

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

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

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

; Clear TC0C as initial value.
        CLR     TC0C

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

9 INSTRUCTION TABLE

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

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

10 ELECTRICAL CHARACTERISTIC

10.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2501CP, SN8P2501CS..... 0°C ~ + 70°C
SN8P2501CPD, SN8P2501CSD..... –40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

10.2 ELECTRICAL CHARACTERISTIC

I DC CHARACTERISTIC

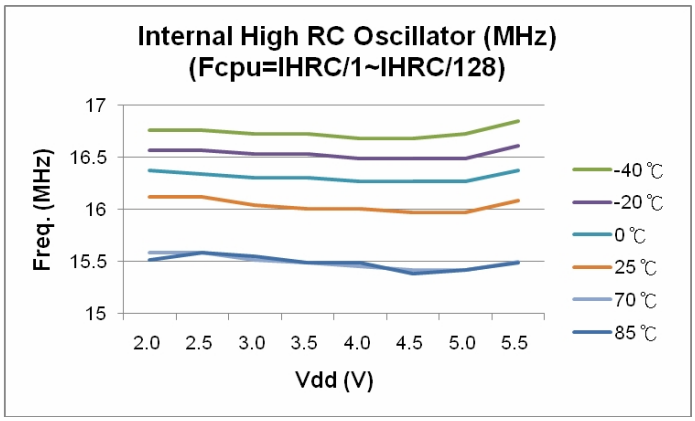
(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 16MHz,Fcpu=1MHz,ambient temperature is 25°C unless otherwise note.)

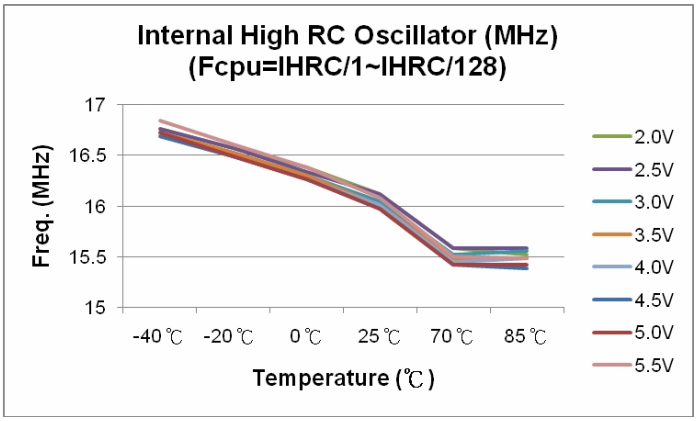
PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, *25°C, Fcpu = 1MHz*		2.2	-	5.5	V
		Normal mode, Vpp = Vdd, *-40°C~85°C*		2.4	-	5.5	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	150	
I/O output source current	IoH	Vop = Vdd – 0.5V		8	15	-	mA
sink current	IoL	Vop = Vss + 0.5V		8	15	-	
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current	Idd1	Run Mode (Low power disable)	Vdd= 3V, Fcpu = 16MHz	-	2.8	-	mA
			Vdd= 5V, Fcpu = 16MHz	-	5.8	-	mA
			Vdd= 3V, Fcpu = 4MHz	-	1.5	-	mA
			Vdd= 5V, Fcpu = 4MHz	-	3	-	mA
			Vdd= 3V, Fcpu = 1MHz	-	1.1	-	mA
			Vdd= 5V, Fcpu = 1MHz	-	2.3	-	mA
			Vdd= 3V, Fcpu = 125KHz	-	0.9	-	mA
			Vdd= 5V, Fcpu = 125KHz	-	2.1	-	mA
	Idd2	Run Mode (Low power enable)	Vdd= 3V, Fcpu = 16MHz	-	2.8	-	mA
			Vdd= 5V, Fcpu = 16MHz	-	5.8	-	mA
			Vdd= 3V, Fcpu = 4MHz	-	1.3	-	mA
			Vdd= 5V, Fcpu = 4MHz	-	2.2	-	mA
			Vdd= 3V, Fcpu = 1MHz	-	0.7	-	mA
			Vdd= 5V, Fcpu = 1MHz	-	1	-	mA
			Vdd= 3V, Fcpu = 125KHz	-	0.5	-	mA
			Vdd= 5V, Fcpu = 125KHz	-	0.6	-	mA
	Idd3	Slow Mode (Internal low RC, Stop high clock)	Vdd= 3V, ILRC=16KHz	-	2.5	-	uA
			Vdd= 5V, ILRC=32KHz	-	7.8	-	uA
	Idd4	Sleep Mode	Vdd= 5V/3V	-	1	2	uA
	Idd5	Green Mode (No loading, Watchdog Disable)	Vdd= 3V, IHRC=16MHz	-	0.45	-	mA
			Vdd= 5V, IHRC=16MHz	-	0.5	-	mA
			Vdd= 3V, ILRC=16KHz	-	1.5	-	uA
			Vdd= 5V, ILRC=32KHz	-	4.5	-	uA
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	*25°C*, Vdd=2.2V~ 5.5V Fcpu=Fhosc/1~Fhosc/128	15.68	16	16.32	MHz
			-40°C~85°C,Vdd=2.4V~ 5.5V Fcpu=Fhosc/1~Fhosc/128	15.2	16	16.8	MHz
LVD Voltage	Vdet0	Low voltage reset level. *-40°C~85°C*		1.6	2.0	2.3	V
	Vdet1	Low voltage reset/indicator level. *-40°C~85°C*		1.8	2.4	3	V
	Vdet2	Low voltage reset/indicator level. *-40°C~85°C*		2.5	3.6	4.5	V

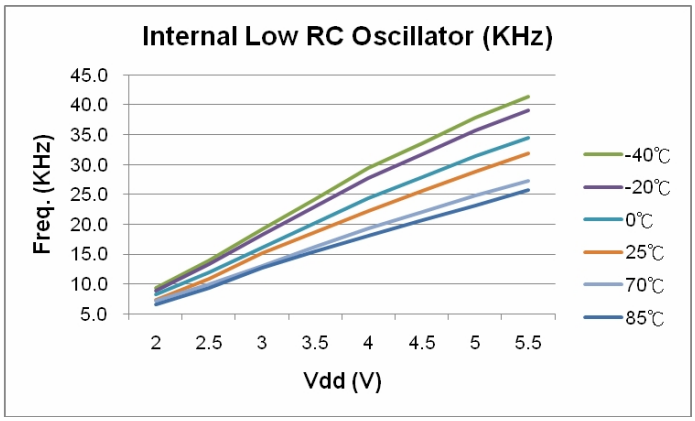
" * " These parameters are for design reference, not tested.

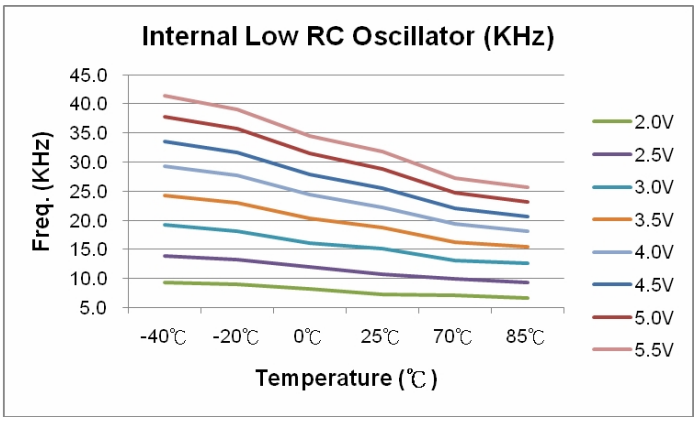
10.3 CHARACTERISTIC GRAPHS

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









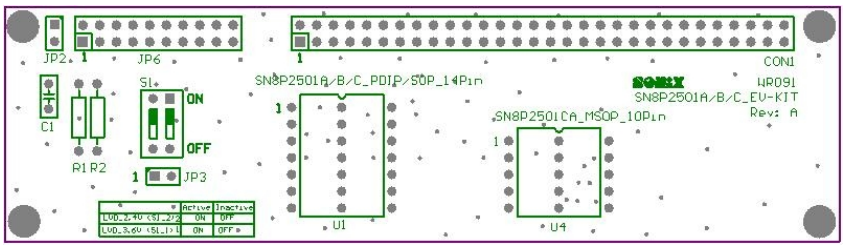
11 DEVELOPMENT TOOL

SONIX provides ICE (in circuit emulation), IDE (Integrated Development Environment) and EV-kit for SN8P2501C development. ICE and EV-kit are external hardware devices, and IDE is a friendly user interface for firmware development and emulation. These development tools' version is as following.

- **ICE: SN8ICE2K Plus 2. (Please install 16MHz crystal in ICE to implement IHRC emulation.).**
- **EV-kit: SN8P2501A/B/C_EV-kit Rev. A.**
- **IDE: SONiX IDE M2IDE_V125.**
- **Writer: MPIII writer.**

11.1 SN8P2501C EV-KIT

SN8P2501C EV-kit PCB Outline:



- CON1: Connect to SN8ICE 2K Plus CON1 (includes GPIO, EV-KIT control signal, and the others).
- JP6: Connect to SN8ICE 2K Plus JP3 (EV-KIT communication bus with ICE, control signal, and the others).
- S1: LVD24V / LVD36V control switch. To emulate LVD2.4V flag / reset function and LVD3.6V / flag function.

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

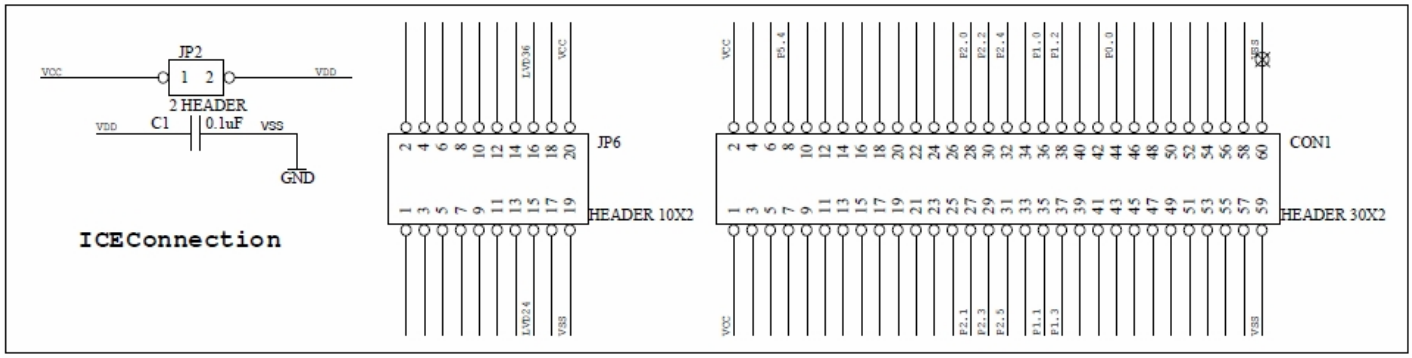
- JP2: Chip and ICE power connector.
- U1: SN8P2501A/B/C chip 14-pin packages connector for connecting to user's target board.

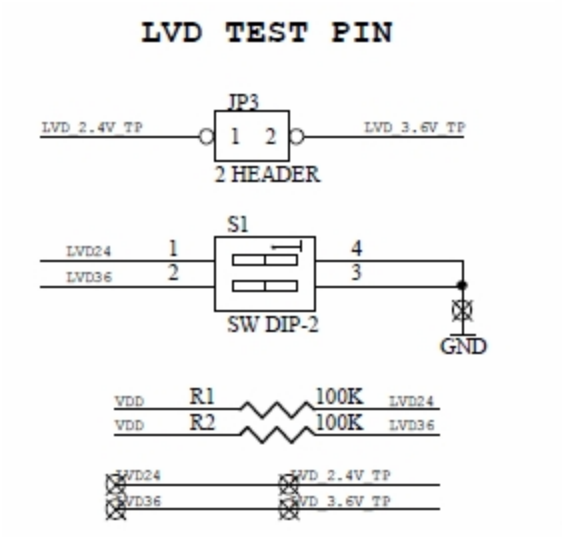
P2.2	1	U	14	P2.3
P2.1	2		13	P2.4
P2.0	3		12	P2.5
VDD	4		11	VSS
P1.3/XIN	5		10	P0.0/INT0
P1.2/XOUT	6		9	P1.0
RST/VPP/P1.1	7		8	P5.4/PWM0/BZ0

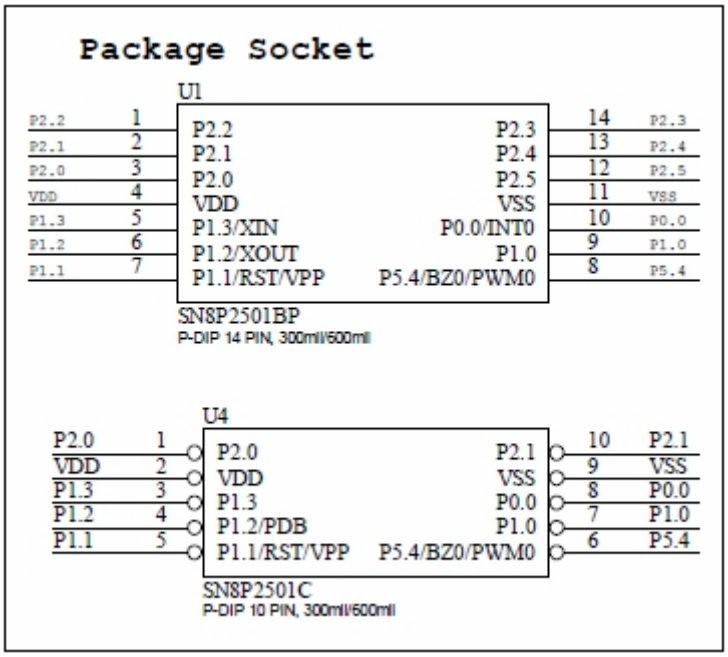
- U2: SN8P2501C chip MSOP 10-pin package connector for connecting to user's target board.

P2.0	1	U	10	P2.1
VDD	2		9	VSS
P1.3/XIN	3		8	P0.0/INT0
P1.2/XOUT	4		7	P1.0
RST/VPP/P1.1	5		6	P5.4/PWM0/BZ0

SN8P2318 EV-kit schematic:





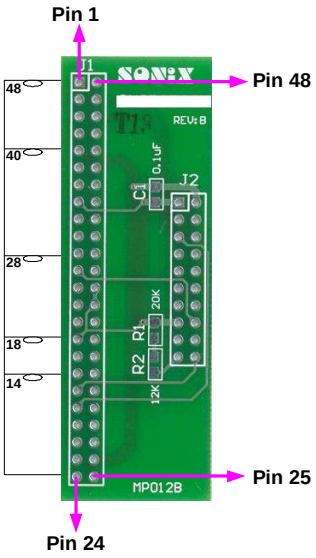


11.2 ICE AND EV-KIT APPLICATION NOTIC

- SN8ICE2K Plus 2 power switch must be turned off before you connect the SN8P2501C EV-KIT to SN8ICE2K Plus 2.
- Connect EV-KIT's JP6/CON1 to ICE's JP3/CON1.
- Turn on SN8ICE2K Plus 2 power switch to start emulation.
- **It is necessary to connect 16MHz crystal in ICE for IHRC_16M mode emulation. SN8ICE2K Plus 2 doesn't support over 8-mips instruction cycle, but real chip does.**

12 OTP PROGRAMMING PIN

12.1 WRITER TRANSITION BOARD SOCKET PIN ASSIGNMENT



JP3 (Mapping to 48-pin text tool)

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

Writer JP1/JP2

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

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

12.2 PROGRAMMING PIN MAPPING:

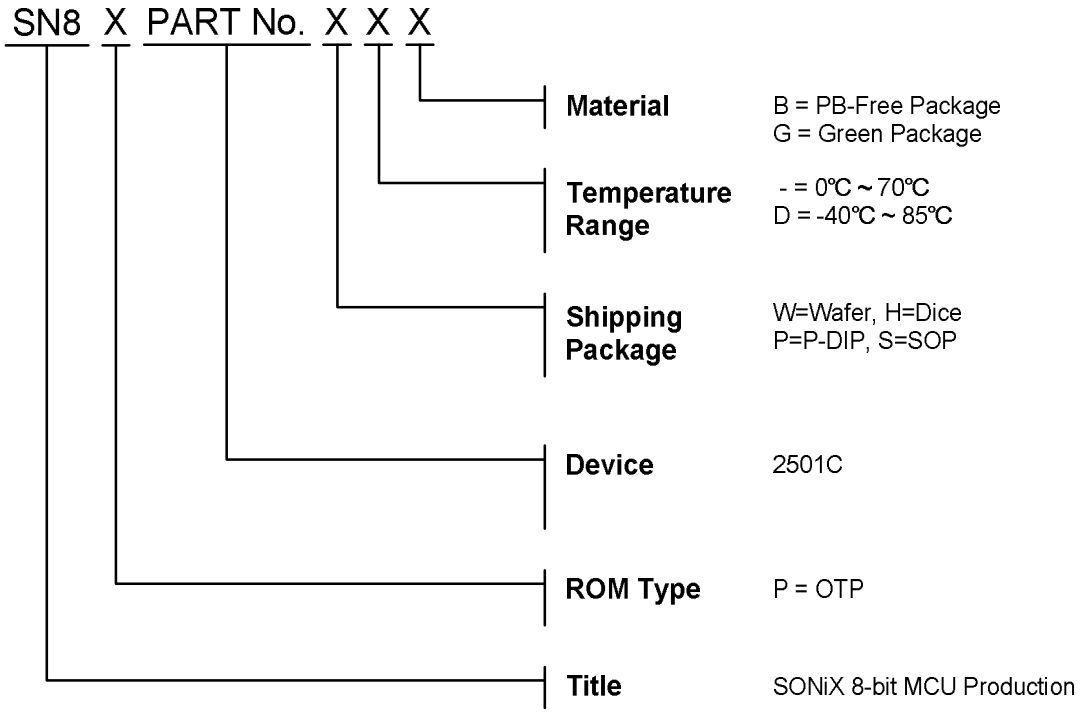
Programming Pin Information of SN8P2501C Series							
Chip Name		SN8P2501CP/S(DIP/SOP)			SN8P2501CA/S(MSOP)		
Writer Connector		IC and JP3 48-pin text tool Pin Assignment					
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number
1	VDD	4	VDD	21	2	VDD	21
2	GND	11	VSS	28	9	VSS	28
3	CLK	10	P0.0	27	8	P0.0	27
4	CE	-	-	-	-	-	-
5	PGM	9	P1.0	26	7	P1.0	26
6	OE	8	P5.4	25	6	P5.4	25
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	7	RST	24	5	RST	24
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	6	P1.2	23	4	P1.2	23

13 Marking Definition

13.1 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.

13.2 MARKING INDETIFICATION SYSTEM



13.3 MARKING EXAMPLE

I Wafer, Dice:

Name	ROM Type	Device	Package	Temperature	Material
S8P2501CW	OTP	2501C	Wafer	0℃~70℃	-
SN8P2501CH	OTP	2501C	Dice	0℃~70℃	-

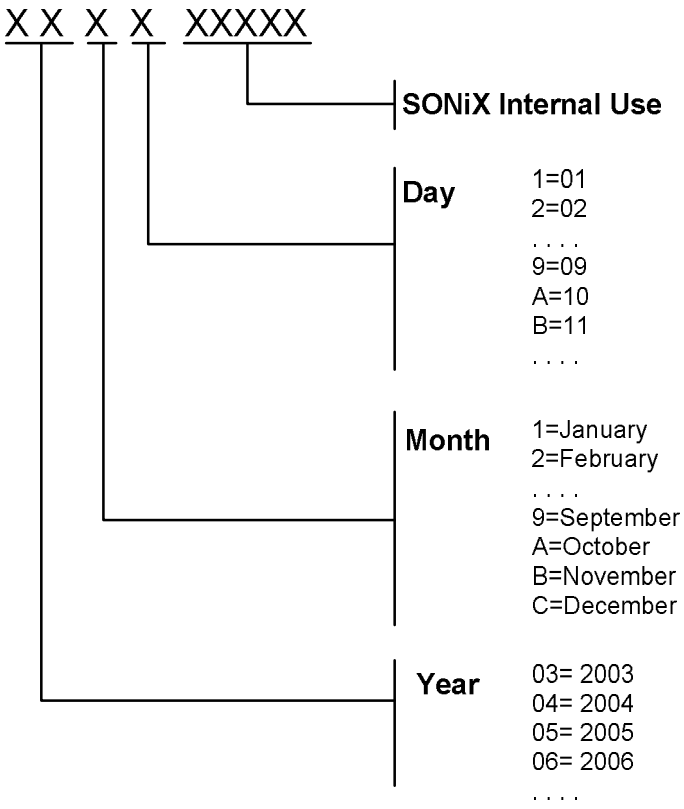
I Green Package:

Name	ROM Type	Device	Package	Temperature	Material
SN8P2501CPG	OTP	2501C	DIP	0℃~70℃	Green Package
SN8P2501CSG	OTP	2501C	SOP	0℃~70℃	Green Package
SN8P2501CAG	OTP	2501C	MSOP	0℃~70℃	Green Package
SN8P2501CPDG	OTP	2501C	DIP	-40℃~85℃	Green Package
SN8P2501CSDG	OTP	2501C	SOP	-40℃~85℃	Green Package
SN8P2501CADG	OTP	2501C	MSOP	-40℃~85℃	Green Package

I PB-Free Package:

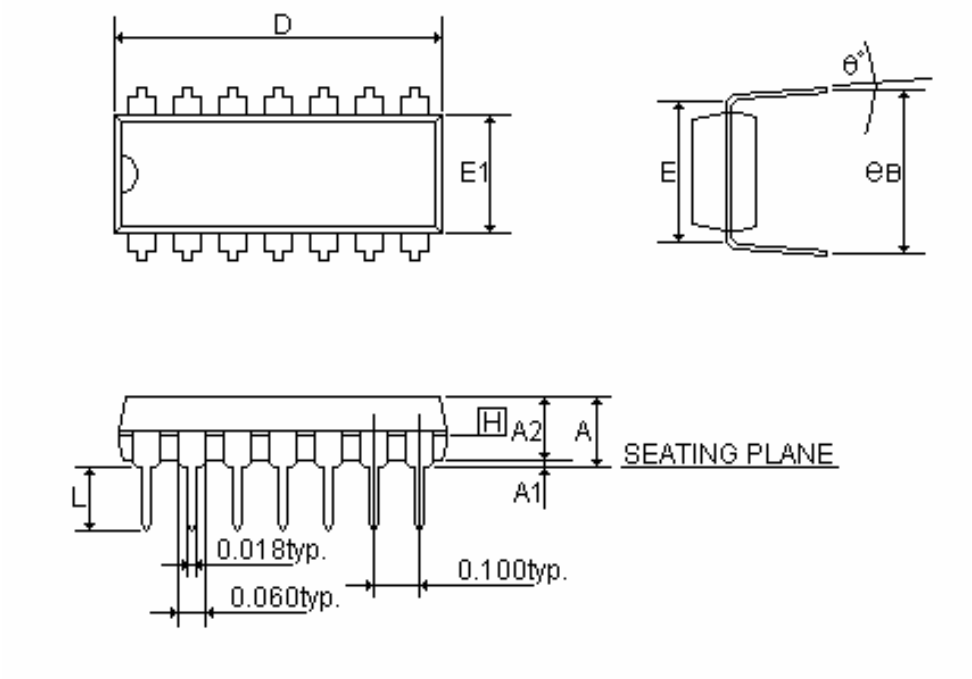
Name	ROM Type	Device	Package	Temperature	Material
SN8P2501CPB	OTP	2501C	DIP	0℃~70℃	PB-Free Package
SN8P2501CSB	OTP	2501C	SOP	0℃~70℃	PB-Free Package
SN8P2501CAB	OTP	2501C	MSOP	0℃~70℃	PB-Free Package
SN8P2501CPDB	OTP	2501C	DIP	-40℃~85℃	PB-Free Package
SN8P2501CSDB	OTP	2501C	SOP	-40℃~85℃	PB-Free Package
SN8P2501CADB	OTP	2501C	MSOP	-40℃~85℃	PB-Free Package

13.4 DATECODE SYSTEM



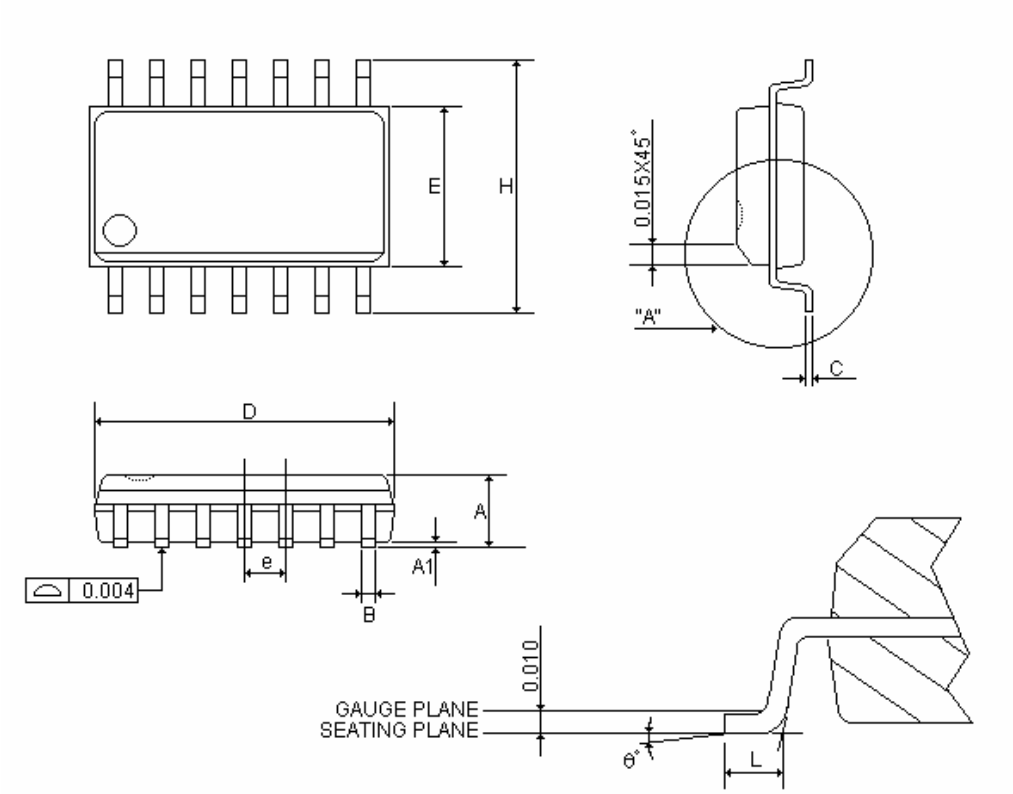
14 PACKAGE INFORMATION

14.1 P-DIP 14 PIN



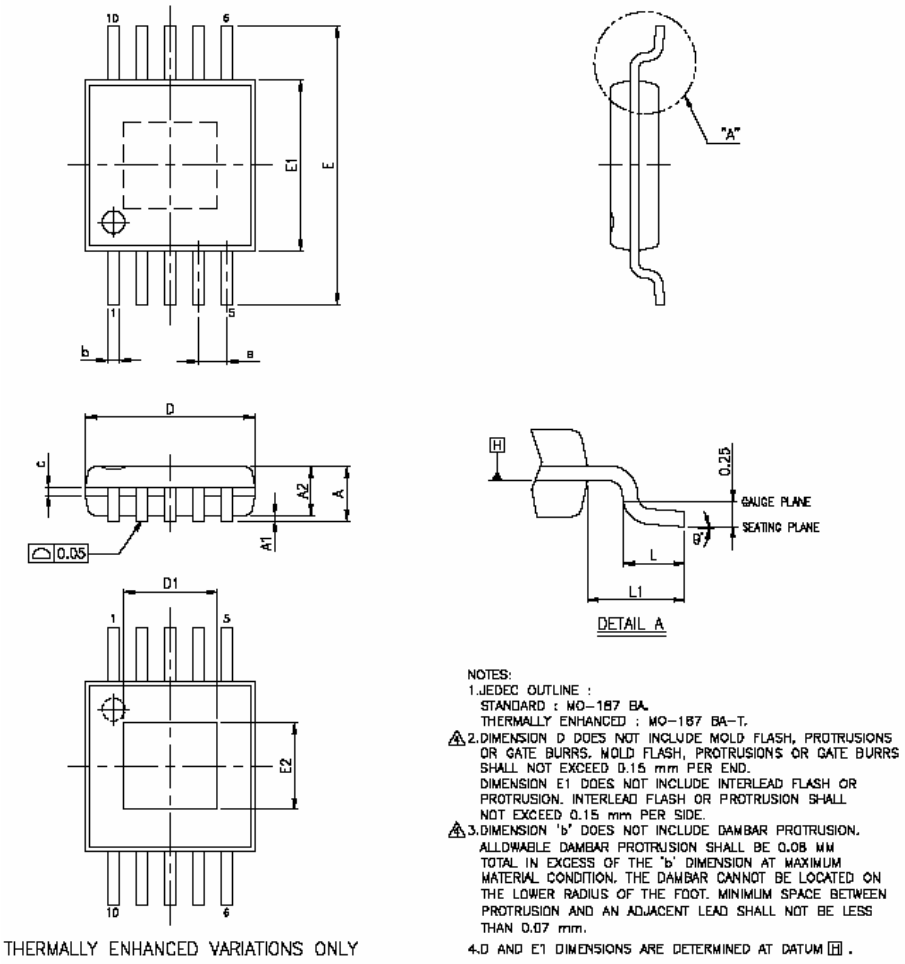
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.735	0.075	0.775	18.669	1.905	19.685
E		0.300			7.62	
E1	0.245	0.250	0.255	6.223	6.35	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
e B	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

14.2 SOP 14 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.058	0.064	0.068	1.4732	1.6256	1.7272
A1	0.004	-	0.010	0.1016	-	0.254
B	0.013	0.016	0.020	0.3302	0.4064	0.508
C	0.0075	0.008	0.0098	0.1905	0.2032	0.2490
D	0.336	0.341	0.344	8.5344	8.6614	8.7376
E	0.150	0.154	0.157	3.81	3.9116	3.9878
e	-	0.050	-	-	1.27	-
H	0.228	0.236	0.244	5.7912	5.9944	6.1976
L	0.015	0.025	0.050	0.381	0.635	1.27
θ°	0°	-	8°	0°	-	8°

14.3 MSOP 10 PIN



NOTES:
1. JEDEC OUTLINE :
   STANDARD : MO-187 BA.
   THERMALLY ENHANCED : MO-187 BA-T.
2. DIMENSION D DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS. MOLD FLASH, PROTRUSIONS OR GATE BURRS SHALL NOT EXCEED 0.15 mm PER END. DIMENSION E1 DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION. INTERLEAD FLASH OR PROTRUSION SHALL NOT EXCEED 0.15 mm PER SIDE.
3. DIMENSION 'b' DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 MM TOTAL IN EXCESS OF THE 'b' DIMENSION AT MAXIMUM MATERIAL CONDITION. THE DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OF THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND AN ADJACENT LEAD SHALL NOT BE LESS THAN 0.07 mm.
4. D AND E1 DIMENSIONS ARE DETERMINED AT DATUM H.

SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.043	-	-	1.10
A1	0.000	-	0.006	0.00	-	0.15
A2	0.030	0.033	0.037	0.75	0.85	0.95
b	0.007	-	0.011	0.17	-	0.27
c	0.003	-		0.08	-	0.23
D	0.118 BSC			3.00 BSC		
E	0.193 BSC			4.90 BSC		
E1	0.118 BSC			3.00 BSC		
e	0.197 BSC			0.50 BSC		
L	0.016	0.024	0.031	0.40	0.60	0.80
L1	0.374 REF			0.95 REF		
θ	0	-	8	0	-	8

PAD SIZE	E2 (mm)		D1 (mm)	
	MIN	MAX	MIN	MAX
75x70E	1.52	1.91	1.42	1.78

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

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

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

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