

SN8P2722

USER'S MANUAL

Version 1.7

SONiX 8-Bit Micro-Controller

SONiX reserves the right to make change without further notice to any products herein to improve reliability, function or design. SONiX does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. SONiX products are not designed, intended, or authorized for use as components in systems intended, for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SONiX product could create a situation where personal injury or death may occur. Should Buyer purchase or use SONiX products for any such unintended or unauthorized application. Buyer shall indemnify and hold SONiX and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that SONiX was negligent regarding the design or manufacture of the part.

AMENDENT HISTORY

| Version | Date | Description |
|---------|-----------|---|
| VER 0.1 | Apr. 2007 | First issue. |
| VER 1.0 | Jun. 2007 | Add development tool section. |
| VER 1.1 | Oct. 2007 | Modify ADC circuit section. |
| VER 1.2 | Jun. 2010 | Modify PWM section. |
| VER 1.3 | Oct. 2011 | 1. Modify "WATCHDOG TIMER" chapter example program fail. B0BSET FWDRST >> MOV A, #5AH B0MOV WDTR,A 2. Adjust Chapter sequence. |
| VER 1.4 | Oct. 2011 | Modify "TC0 CLOCK FREQUENCY OUTPUT (BUZZER)" chapter description : TC0 Setup TC0OUT output from TC0 to TC0OUT (P5.4). The external high-speed clock is 4MHz.. The TC0OUT frequency is 0.5KHz. Because the TC0OUT signal is divided by 2, set the TC0 clock to 1KHz. The TC0 clock source is from external oscillator clock. TC0 rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 131. >> TC0 Setup TC0OUT output from TC0 to TC0OUT (P5.4). The external high-speed clock is 4MHz.. Fcpu = Fosc/4 = 1MIPS. The TC0OUT frequency is 1KHz. Because the TC0OUT signal is divided by 2, set the TC0 clock to 2KHz. The TC0 clock source is from external oscillator clock. TC0 rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 131. |
| VER 1.5 | Oct. 2011 | Modify "ELECTRICAL CHARACTERICS" chapter operating temperature from 0~70°C to -10~70°C and others. |
| VER 1.6 | Jul. 2012 | Modify "ELECTRICAL CHARACTERICS" chapter operating temperature from -10~70°C to -20~85°C and others. |
| VER 1.7 | Jul. 2014 | Add TSSOP20 package type. |

Table of Content

| | |
|---|-----------|
| AMENDMENT HISTORY | 2 |
| 1 PRODUCT OVERVIEW | 6 |
| 1.1 FEATURES | 6 |
| 1.2 SYSTEM BLOCK DIAGRAM | 7 |
| 1.3 PIN ASSIGNMENT | 8 |
| 1.4 PIN DESCRIPTIONS | 8 |
| 1.5 PIN CIRCUIT DIAGRAMS | 9 |
| 2 CENTRAL PROCESSOR UNIT (CPU) | 10 |
| 2.1 MEMORY MAP | 10 |
| 2.1.1 PROGRAM MEMORY (ROM) | 10 |
| 2.1.2 RESET VECTOR (0000H) | 11 |
| 2.1.3 INTERRUPT VECTOR (0008H) | 12 |
| 2.1.4 LOOK-UP TABLE DESCRIPTION | 14 |
| 2.1.5 JUMP TABLE DESCRIPTION | 16 |
| 2.1.6 CHECKSUM CALCULATION | 18 |
| 2.1.7 CODE OPTION TABLE | 19 |
| 2.1.8 DATA MEMORY (RAM) | 20 |
| 2.1.9 SYSTEM REGISTER | 20 |
| 2.1.9.1 SYSTEM REGISTER TABLE | 20 |
| 2.1.9.2 SYSTEM REGISTER DESCRIPTION | 20 |
| 2.1.9.3 BIT DEFINITION of SYSTEM REGISTER | 21 |
| 2.1.10 ACCUMULATOR | 22 |
| 2.1.11 PROGRAM FLAG | 23 |
| 2.1.12 PROGRAM COUNTER | 24 |
| 2.1.13 Y, Z REGISTERS | 27 |
| 2.1.14 R REGISTERS | 28 |
| 2.2 ADDRESSING MODE | 29 |
| 2.2.1 IMMEDIATE ADDRESSING MODE | 29 |
| 2.2.2 DIRECTLY ADDRESSING MODE | 29 |
| 2.2.3 INDIRECTLY ADDRESSING MODE | 29 |
| 2.3 STACK OPERATION | 30 |
| 2.3.1 OVERVIEW | 30 |
| 2.3.2 STACK REGISTERS | 31 |
| 2.3.3 STACK OPERATION EXAMPLE | 32 |
| 3 RESET | 33 |
| 3.1 OVERVIEW | 33 |
| 3.2 POWER ON RESET | 34 |
| 3.3 WATCHDOG RESET | 34 |
| 3.4 BROWN OUT RESET | 35 |
| 3.4.1 BROWN OUT DESCRIPTION | 35 |
| 3.4.2 THE SYSTEM OPERATING VOLTAGE | 36 |
| 3.4.3 LOW VOLTAGE DETECTOR (LVD) | 36 |
| 3.4.4 BROWN OUT RESET IMPROVEMENT | 38 |
| 3.5 EXTERNAL RESET | 39 |
| 3.6 EXTERNAL RESET CIRCUIT | 39 |
| 3.6.1 Simply RC Reset Circuit | 39 |
| 3.6.2 Diode & RC Reset Circuit | 40 |
| 3.6.3 Zener Diode Reset Circuit | 40 |

| | | |
|----------|--|-----------|
| 3.6.4 | Voltage Bias Reset Circuit | 41 |
| 3.6.5 | External Reset IC..... | 42 |
| 4 | SYSTEM CLOCK..... | 43 |
| 4.1 | OVERVIEW | 43 |
| 4.2 | CLOCK BLOCK DIAGRAM | 43 |
| 4.3 | OSCM REGISTER | 44 |
| 4.4 | SYSTEM HIGH CLOCK | 44 |
| 4.4.1 | INTERNAL HIGH RC | 44 |
| 4.4.2 | EXTERNAL HIGH CLOCK..... | 45 |
| 4.4.2.1 | CRYSTAL/CERAMIC..... | 45 |
| 4.4.2.2 | RC..... | 46 |
| 4.4.2.3 | EXTERNAL CLOCK SIGNAL..... | 46 |
| 4.5 | SYSTEM LOW CLOCK | 47 |
| 4.5.1 | SYSTEM CLOCK MEASUREMENT | 48 |
| 5 | SYSTEM OPERATION MODE | 49 |
| 5.1 | OVERVIEW..... | 49 |
| 5.2 | SYSTEM MODE SWITCHING EXAMPLE..... | 50 |
| 5.3 | WAKEUP..... | 52 |
| 5.3.1 | OVERVIEW..... | 52 |
| 5.3.2 | WAKEUP TIME | 52 |
| 6 | INTERRUPT..... | 53 |
| 6.1 | OVERVIEW..... | 53 |
| 6.2 | INTEN INTERRUPT ENABLE REGISTER..... | 54 |
| 6.3 | INTRQ INTERRUPT REQUEST REGISTER | 54 |
| 6.4 | GIE GLOBAL INTERRUPT OPERATION | 55 |
| 6.5 | PUSH, POP ROUTINE..... | 55 |
| 6.6 | EXTERNAL INTERRUPT OPERATION (INT0)..... | 56 |
| 6.7 | T0 INTERRUPT OPERATION..... | 57 |
| 6.8 | TC0 INTERRUPT OPERATION..... | 58 |
| 6.9 | ADC INTERRUPT OPERATION | 59 |
| 6.10 | MULTI-INTERRUPT OPERATION..... | 60 |
| 7 | I/O PORT | 61 |
| 7.1 | I/O PORT MODE | 61 |
| 7.2 | I/O PULL UP REGISTER | 62 |
| 7.3 | I/O PORT DATA REGISTER | 63 |
| 7.4 | PORT 4 ADC SHARE PIN..... | 64 |
| 8 | TIMERS | 66 |
| 8.1 | WATCHDOG TIMER..... | 66 |
| 8.2 | TIMER 0 (T0)..... | 68 |
| 8.2.1 | OVERVIEW | 68 |
| 8.2.2 | T0M MODE REGISTER | 68 |
| 8.2.3 | T0C COUNTING REGISTER | 69 |
| 8.2.4 | T0 TIMER OPERATION SEQUENCE..... | 70 |
| 8.3 | TIMER/COUNTER 0 (TC0) | 71 |
| 8.3.1 | OVERVIEW | 71 |
| 8.3.2 | TC0M MODE REGISTER..... | 72 |
| 8.3.3 | TC0C COUNTING REGISTER | 73 |
| 8.3.4 | TC0R AUTO-LOAD REGISTER..... | 74 |
| 8.3.5 | TC0 CLOCK FREQUENCY OUTPUT (BUZZER)..... | 75 |
| 8.3.6 | TC0 TIMER OPERATION SEQUENCE | 76 |
| 8.4 | PWM0 MODE..... | 77 |
| 8.4.1 | OVERVIEW | 77 |

| | | |
|-----------|--|-----------|
| 8.4.2 | TC0IRQ and PWM Duty | 78 |
| 8.4.3 | PWM Duty with TC0R Changing | 79 |
| 8.4.4 | PWM PROGRAM EXAMPLE | 80 |
| 9 | 5 CHANNEL ANALOG TO DIGITAL CONVERTER..... | 81 |
| 9.1 | OVERVIEW | 81 |
| 9.2 | ADM REGISTER | 82 |
| 9.3 | ADR REGISTERS | 82 |
| 9.4 | ADB REGISTERS | 83 |
| 9.5 | P4CON REGISTERS | 84 |
| 9.6 | ADC CONVERTING TIME | 84 |
| 9.7 | ADC ROUTINE EXAMPLE | 85 |
| 9.8 | ADC CIRCUIT | 85 |
| 10 | 2K/4K BUZZER GENERATOR..... | 86 |
| 10.1 | OVERVIEW | 86 |
| 10.2 | BZM REGISTER..... | 87 |
| 11 | INSTRUCTION TABLE | 88 |
| 12 | ELECTRICAL CHARACTERISTIC | 89 |
| 12.1 | ABSOLUTE MAXIMUM RATING | 89 |
| 12.2 | ELECTRICAL CHARACTERISTIC | 89 |
| 13 | SN8P2722 DEVELOPMENT TOOL..... | 91 |
| 13.1 | SN8P2722 EV-KIT | 91 |
| 13.2 | ICE AND EV-KIT APPLICATION NOTIC | 92 |
| 14 | OTP PROGRAMMING PIN..... | 93 |
| 14.1 | EASY WRITER TRANSITION BOARD SOCKET PIN ASSIGNMENT | 93 |
| 14.2 | PROGRAMMING PIN MAPPING:..... | 94 |
| 15 | MARKING DEFINITION..... | 95 |
| 15.1 | INTRODUCTION | 95 |
| 15.2 | MARKING INDETIFICATION SYSTEM..... | 95 |
| 15.3 | MARKING EXAMPLE | 96 |
| 15.4 | DATECODE SYSTEM | 96 |
| 16 | PACKAGE INFORMATION | 97 |
| 16.1 | P-DIP 20 PIN | 97 |
| 16.2 | SOP 20 PIN..... | 98 |
| 16.3 | SSOP 20 PIN..... | 99 |
| 16.4 | TSSOP 20 PIN | 100 |

1 PRODUCT OVERVIEW

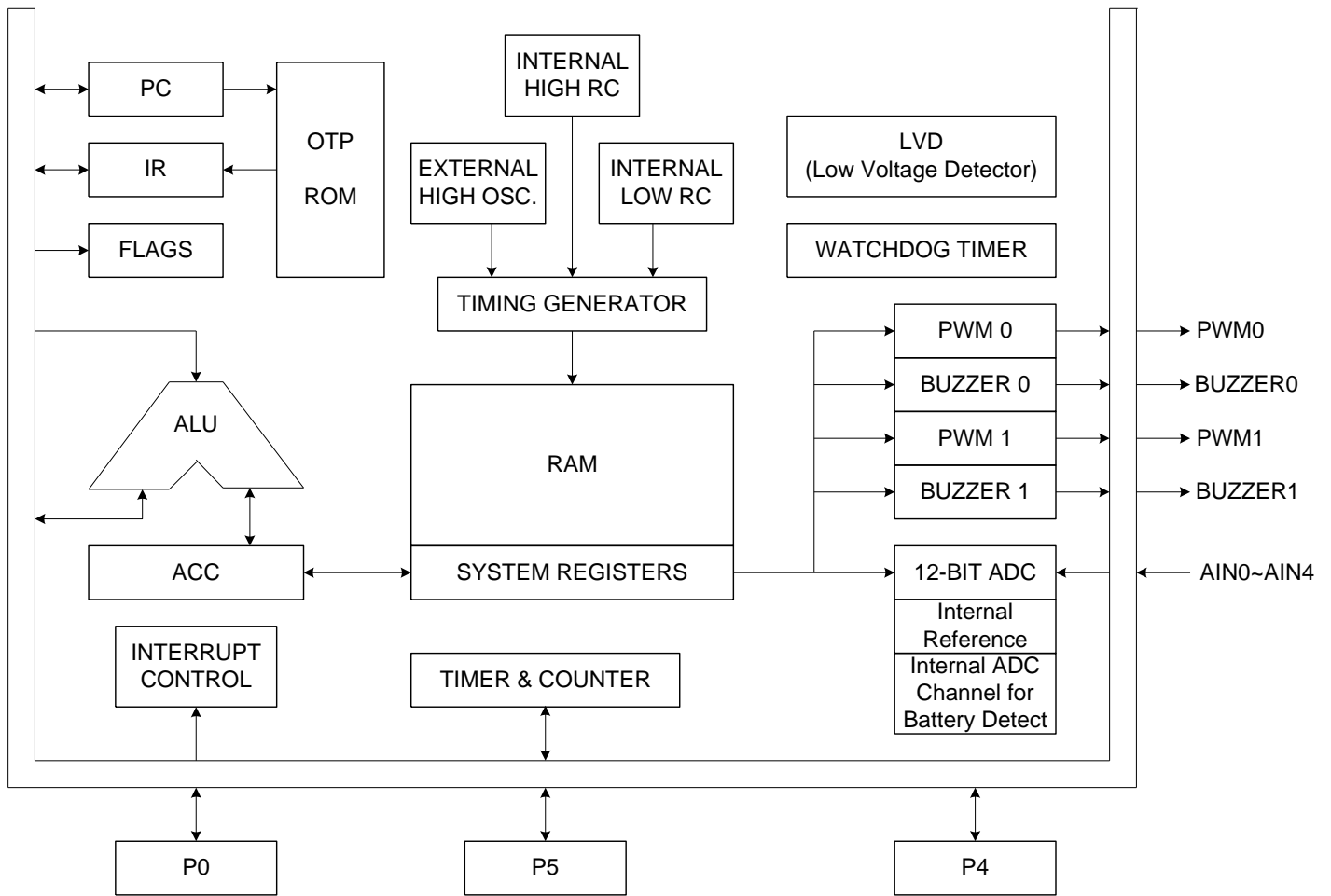
1.1 FEATURES

- ◆ **Memory configuration**
ROM size: 2K * 16 bits.
RAM size: 128 * 8 bits.
- ◆ **8 levels stack buffer.**
- ◆ **Four interrupt sources**
Three internal interrupts: T0, TC0, ADC.
One external interrupts: INT0
- ◆ **I/O pin configuration**
Bi-directional: P0, P4, P5
Wakeup: P0 level change
Pull-up resistors: P0, P4, P5
External interrupt: P0.0
ADC input pin: AIN0~AIN4
- ◆ **Fcpu (Instruction cycle)**
Fcpu = Fosc/4, Fosc/8, Fosc/16.
- ◆ **Powerful instructions**
Instruction's length is one word.
Most of instructions are one cycle only.
All ROM area JMP instruction.
All ROM area lookup table function (MOVC)
- ◆ **Two 8-bit timer. (T0, TC0).**
T0: Basic timer.
TC0: Auto-reload timer/Counter/PWM/TC0OUT output.
- ◆ **One channel 8-bit PWM output.**
- ◆ **One channel 2KHz/4KHz buzzer output.**
- ◆ **On chip watchdog timer and clock source is internal low clock RC type (16KHz @3V, 32KHz @5V).**
- ◆ **5 channel 12-bit ADC.**
- ◆ **Four system clocks**
External high clock: RC type up to 4 MHz
External high clock: Crystal type up to 4 MHz
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)**
P-DIP 20 pin
SOP 20 pin
SSOP 20 pin
TSSOP 20 pin

Features Selection Table

| CHIP | ROM | RAM | Stack | Timer | | | I/O | ADC | 2K/4K Buzzer | PWM TCnOUT | Wake-up Pin No. | Package |
|-----------|-------|-----|-------|-------|-----|-----|-----|--------|-----------------|---------------|--------------------|---------------------------------|
| | | | | T0 | TC0 | TC1 | | | | | | |
| SN8P2711A | 1K*16 | 64 | 4 | - | V | V | 12 | 5+1 ch | - | 2 | 5 | PDIP14/ SOP14/SSOP16 |
| SN8P2722 | 2K*16 | 128 | 8 | V | V | - | 18 | 5-ch | V | 1 | 8 | PDIP20/SOP20/ SSOP20/TSSOP20 |

1.2 SYSTEM BLOCK DIAGRAM



1.3 PIN ASSIGNMENT

SN8P2722P (PDIP 20 pins)
 SN8P2722S (SOP 20 pins)
 SN8P2722X (SSOP 20 pins)
 SN8P2722T (TSSOP 20 pins)

| | | | | |
|--------------|----|---|----|-----------------|
| VSS | 1 | U | 20 | VDD |
| XIN/P0.1 | 2 | | 19 | P0.0/INT0 |
| XOUT/P0.2 | 3 | | 18 | P4.4/AIN4 |
| RST/VPP/P0.3 | 4 | | 17 | P4.3/AIN3 |
| P0.4/BZ | 5 | | 16 | P4.2/AIN2 |
| P0.5 | 6 | | 15 | P4.1/AIN1 |
| P0.6 | 7 | | 14 | P4.0/AIN0 |
| P0.7 | 8 | | 13 | P5.4/TC0OUT/PWM |
| P5.0 | 9 | | 12 | P5.3 |
| P5.1 | 10 | | 11 | P5.2 |

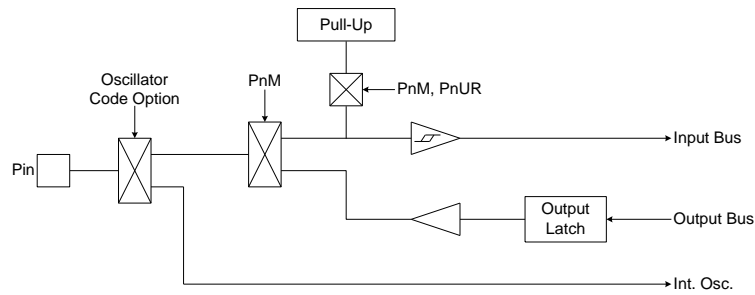
SN8P2722P
SN8P2722S
SN8P2722X
SN8P2722T

1.4 PIN DESCRIPTIONS

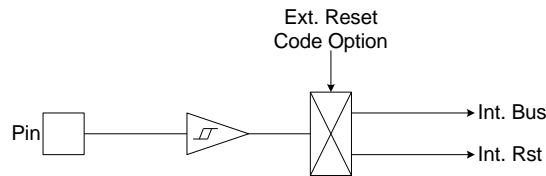
| PIN NAME | TYPE | DESCRIPTION |
|------------------|------|---|
| VDD, VSS | P | Power supply input pins. |
| P0.3/RST/VPP | I, P | RST: System reset input pin. Schmitt trigger structure, low active, normal stay to "high". VPP: OTP 12.3V power input pin in programming mode. P0.3: Input only pin (Schmitt trigger) if disable external reset function, without build-in pull-up resistor and built-in wakeup function. |
| XIN/P0.1 | I/O | XIN: Oscillator input pin while external oscillator enable (crystal and RC). P0.1: Port 0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistors and wakeup function. |
| XOUT/P0.2 | I/O | XOUT: Oscillator output pin while external crystal enable. P0.2: Port 0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistors and wakeup function. |
| P0.0/INT0 | I/O | Port 0.0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistors and wakeup function. INT0 trigger pin (Schmitt trigger). TC0 event counter input pin. |
| P0.4/BZ | I/O | Port 0.4 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistors and wakeup function. BZ: 2KHz/4KHz buzzer output pin. |
| P0[7:5] | I/O | Port 0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistors. |
| P4[4:0]/AIN[4:0] | I/O | Port 4 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistors. AIN[4:0]: ADC analogy signal input. |
| P5[3:0] | I/O | Port 5 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistors. |
| P5.4/PWM/TC0OUT | I/O | Port 5.4 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistors. PWM: PWM output pin. TC0OUT: TC0 ÷ 2 signal output pin. |

1.5 PIN CIRCUIT DIAGRAMS

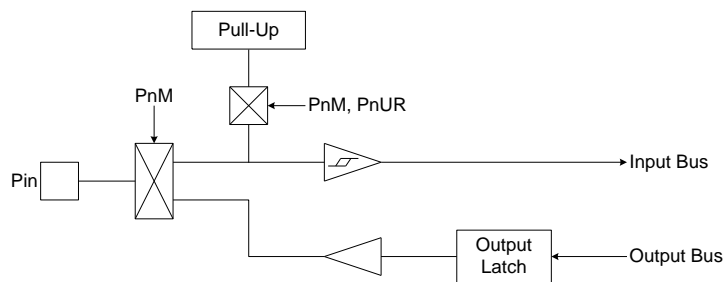
Port 0.1, P0.2 structure:



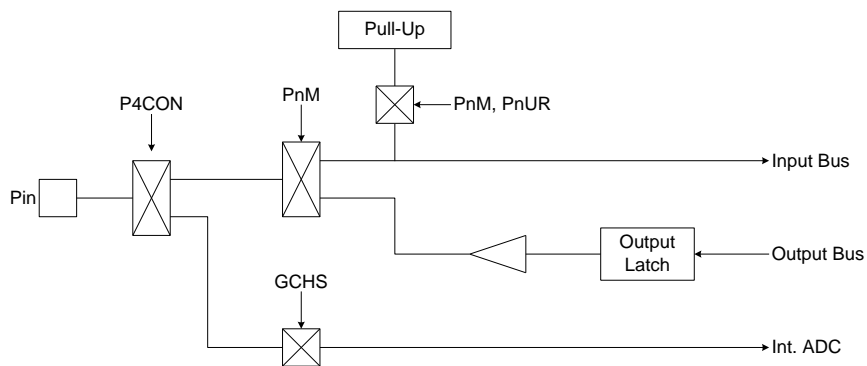
Port 0.3 structure:



Port 0, Port 5 structure:



Port 4 structure:

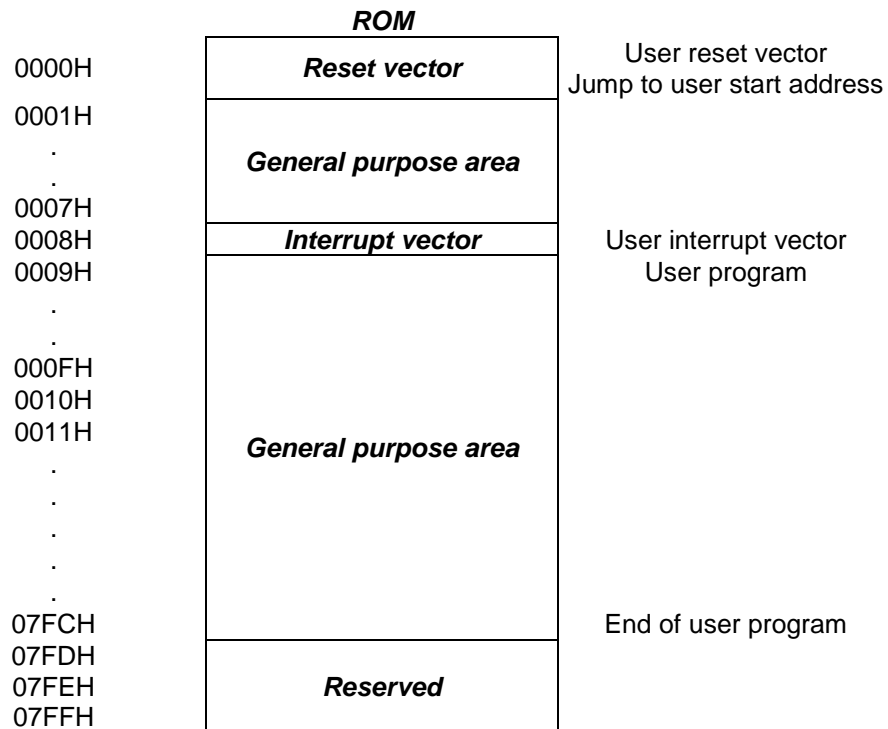


2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ 2K words ROM



2.1.2 RESET VECTOR (0000H)

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

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

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

➤ Example: Defining Reset Vector

```

                ORG      0                ; 0000H
                JMP      START           ; Jump to user program address.
                ...
START:          ORG      10H             ; 0010H, The head of user program.
                ...                     ; User program
                ...
                ENDP                    ; End of program
```

2.1.3 INTERRUPT VECTOR (0008H)

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

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

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

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

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

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

    ENDP                ; End of program
```

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

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

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

MY_IRQ:
    ...              ; The head of interrupt service routine.
    PUSH   ACC        ; Save ACC and PFLAG register to buffers.
    ...
    ...
    POP    ACC        ; Load ACC and PFLAG register from buffers.
    RETI   0          ; 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.

2.1.4 LOOK-UP TABLE DESCRIPTION

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

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

```

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

                                ; Increment the index address for next address.
                                ; Z+1
                                ; Z is not overflow.
INCMS    Z                ; Z overflow (FFH → 00), → Y=Y+1
JMP      @F
INCMS    Y
NOP
                                ;
@@:      MOVC              ; To lookup data, R = 51H, ACC = 05H.
...
TABLE1:  DW      0035H    ; To define a word (16 bits) data.
          DW      5105H
          DW      2012H
          ...

```

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

➤ **Example: INC_YZ macro.**

```

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

INCMS    Y                ; Y+1
NOP      ; Not overflow

@@:
ENDM

```

➤ **Example: Modify above example by “INC_YZ” macro.**

```

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

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

```

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

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

```

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

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

        B0BTS1   FC            ; Check the carry flag.
        JMP      GETDATA      ; FC = 0
        INCMS   Y              ; FC = 1. Y+1.
        NOP

GETDATA: ;
        MOVC     ; To lookup data. If BUF = 0, data is 0x0035
        ; If BUF = 1, data is 0x5105
        ; If BUF = 2, data is 0x2012
        ...

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

```

2.1.5 JUMP TABLE DESCRIPTION

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

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

➤ **Example: Jump table.**

```

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

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

```

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

➤ **Example: If “jump table” crosses over ROM boundary will cause errors.**

```

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

```

* **Note:** “VAL” is the number of the jump table listing number.

➤ Example: “@JMP_A” application in SONIX macro file called “MACRO3.H”.

```

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

```

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

➤ Example: “@JMP_A” operation.

; Before compiling program.

```

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

```

; After compiling program.

```

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

```

2.1.6 CHECKSUM CALCULATION

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

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

```

MOV      A,#END_USER_CODE$L
B0MOV   END_ADDR1, A      ; Save low end address to end_addr1
MOV      A,#END_USER_CODE$M
B0MOV   END_ADDR2, A      ; Save middle end address to end_addr2
CLR     Y                  ; Set Y to 00H
CLR     Z                  ; Set Z to 00H

@@:
MOV     FC
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

```

2.1.7 CODE OPTION TABLE

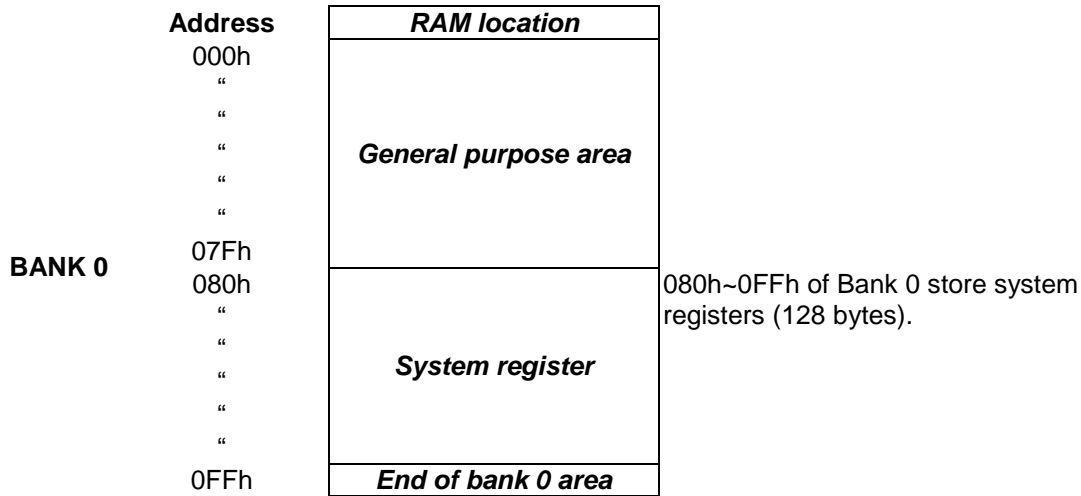
| Code Option | Content | Function Description |
|--------------|-----------|--|
| High_Clk | IHRC_16M | High speed internal 16MHz RC. XIN/XOUT become to P0.2/P0.3 bi-direction I/O pins. |
| | RC | Low cost RC for external high clock oscillator and XOUT becomes to P0.2 bit-direction I/O pin. |
| | 12M X'tal | High speed crystal /resonator (e.g. 12MHz) for external high clock oscillator. |
| | 4M X'tal | Standard crystal /resonator (e.g. 4M) for external high clock oscillator. |
| Watch_Dog | Always_On | Watchdog timer is always on enable even in power down and green mode. |
| | Enable | Enable watchdog timer. Watchdog timer stops in power down mode and green mode. |
| | Disable | Disable Watchdog function. |
| Fcpu | Fhosc/4 | Instruction cycle is 4 oscillator clocks. |
| | Fhosc/8 | Instruction cycle is 8 oscillator clocks. |
| | Fhosc/16 | Instruction cycle is 16 oscillator clocks. |
| Reset_Pin | Reset | Enable External reset pin. |
| | P03 | Enable P0.3 input only without pull-up resistor. |
| Security | Enable | Enable ROM code Security function. |
| | Disable | Disable ROM code Security function. |
| Noise_Filter | Enable | Enable Noise Filter. |
| | Disable | Disable Noise Filter. |
| 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. |

* **Note:**

1. In high noisy environment, enable "Noise Filter" and set Watch_Dog as "Always_On" is strongly recommended.
2. Fcpu code option is only available for High Clock. Fcpu of slow mode is Fosc/4 (the Fosc is internal low clock).

2.1.8 DATA MEMORY (RAM)

128 X 8-bit RAM



2.1.9 SYSTEM REGISTER

2.1.9.1 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 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | P4CON |
| B | - | ADM | ADB | ADR | - | - | - | - | P0M | - | - | - | - | - | - | PEDGE |
| C | - | - | - | - | P4M | P5M | - | - | INTRQ | INTEN | OSCM | - | WDTR | TC0R | PCL | PCH |
| D | P0 | - | - | - | P4 | P5 | - | - | T0M | T0C | TC0M | TC0C | BZM | - | - | STKP |
| E | P0UR | - | - | - | P4UR | P5UR | - | @YZ | - | - | - | - | - | - | - | - |
| F | STK7L | STK7H | STK6L | STK6H | STK5L | STK5H | STK4L | STK4H | STK3L | STK3H | STK2L | STK2H | STK1L | STK1H | STK0L | STK0H |

2.1.9.2 SYSTEM REGISTER DESCRIPTION

R = Working register and ROM look-up data buffer.
PFLAG = ROM page and special flag register.
ADB = ADC data buffer.
PnM = Port n input/output mode register.
INTRQ = Interrupt request register.
OSCM = Oscillator mode register.
WDTR = Watchdog timer clear register.
Pn = Port n data buffer.
T0M = T0 mode register.
TC0M = TC0 mode register.
TC0R = TC0 auto-reload data buffer.
STK0~STK7 = Stack 0 ~ stack 7 buffer.

Y, Z = Working, @YZ and ROM addressing register.
P4CON = P4 configuration register.
ADM = ADC's mode register.
ADR = ADC resolution selection register.
PEDGE = P0.0 edge direction register.
INTEN = Interrupt enable register.
PCH, PCL = Program counter.
PnUR = Port n pull-up resistor control register.
T0C = T0 counting register.
TC0C = TC0 counting register.
BZM = Buzzer control register.
@YZ = RAM YZ indirect addressing index pointer.
STKP = Stack pointer buffer.

2.1.9.3 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 |
| 0AFH | | | | P4CON4 | P4CON3 | P4CON2 | P4CON1 | P4CON0 | R/W | P4CON |
| 0B1H | ADENB | ADS | EOC | GCHS | | CHS2 | CHS1 | CHS0 | R/W | ADM |
| 0B2H | ADB11 | ADB10 | ADB9 | ADB8 | ADB7 | ADB6 | ADB5 | ADB4 | R | ADB |
| 0B3H | | ADCKS1 | | ADCKS0 | ADB3 | ADB2 | ADB1 | ADB0 | R/W | ADR |
| 0B8H | P07M | P06M | P05M | P04M | | P02M | P01M | P00M | R/W | P0M |
| 0BFH | | | | P00G1 | P00G0 | | | | R/W | PEDGE |
| 0C4H | | | | P44M | P43M | P42M | P41M | P40M | R/W | P4M |
| 0C5H | | | | P54M | P53M | P52M | P51M | P50M | R/W | P5M |
| 0C8H | ADCIRQ | | TC0IRQ | T0IRQ | | | | P00IRQ | R/W | INTRQ |
| 0C9H | ADCIEN | | 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 | | | | | | PC10 | PC9 | PC8 | R/W | PCH |
| 0D0H | P07 | P06 | P05 | P04 | P03 | P02 | P01 | P00 | R/W | P0 |
| 0D4H | | | | P44 | P43 | P42 | P41 | P40 | R/W | P4 |
| 0D5H | | | | P54 | P53 | P52 | P51 | P50 | R/W | P5 |
| 0D8H | T0ENB | T0rate2 | T0rate1 | T0rate0 | | | | | R/W | T0M |
| 0DAH | TC0ENB | TC0rate2 | TC0rate1 | TC0rate0 | TC0CKS | ALOAD0 | TC0OUT | PWM0OUT | R/W | TC0M |
| 0DBH | TC0C7 | TC0C6 | TC0C5 | TC0C4 | TC0C3 | TC0C2 | TC0C1 | TC0C0 | R/W | TC0C |
| 0DCH | BZEN | BZrate1 | BZrate0 | | | | | | R/W | BZM |
| 0DFH | GIE | | | | | STKPB2 | STKPB1 | STKPB0 | R/W | STKP |
| 0E0H | P07R | P06R | P05R | P04R | | P02R | P01R | P00R | W | P0UR |
| 0E4H | | | | P44R | P43R | P42R | P41R | P40R | W | P4UR |
| 0E5H | | | | P54R | P53R | P52R | P51R | P50R | W | P5UR |
| 0E7H | @YZ7 | @YZ6 | @YZ5 | @YZ4 | @YZ3 | @YZ2 | @YZ1 | @YZ0 | R/W | @YZ |
| 0F0H | S7PC7 | S7PC6 | S7PC5 | S7PC4 | S7PC3 | S7PC2 | S7PC1 | S7PC0 | R/W | STK7L |
| 0F1H | | | | | | S7PC10 | S7PC9 | S7PC8 | R/W | STK7H |
| 0F2H | S6PC7 | S6PC6 | S6PC5 | S6PC4 | S6PC3 | S6PC2 | S6PC1 | S6PC0 | R/W | STK6L |
| 0F3H | | | | | | S6PC10 | S6PC9 | S6PC8 | R/W | STK6H |
| 0F4H | S5PC7 | S5PC6 | S5PC5 | S5PC4 | S5PC3 | S5PC2 | S5PC1 | S5PC0 | R/W | STK5L |
| 0F5H | | | | | | S5PC10 | S5PC9 | S5PC8 | R/W | STK5H |
| 0F6H | S4PC7 | S4PC6 | S4PC5 | S4PC4 | S4PC3 | S4PC2 | S4PC1 | S4PC0 | R/W | STK4L |
| 0F7H | | | | | | S4PC10 | S4PC9 | S4PC8 | R/W | STK4H |
| 0F8H | S3PC7 | S3PC6 | S3PC5 | S3PC4 | S3PC3 | S3PC2 | S3PC1 | S3PC0 | R/W | STK3L |
| 0F9H | | | | | | S3PC10 | S3PC9 | S3PC8 | R/W | STK3H |
| 0FAH | S2PC7 | S2PC6 | S2PC5 | S2PC4 | S2PC3 | S2PC2 | S2PC1 | S2PC0 | R/W | STK2L |
| 0FBH | | | | | | S2PC10 | S2PC9 | S2PC8 | R/W | STK2H |
| 0FCH | S1PC7 | S1PC6 | S1PC5 | S1PC4 | S1PC3 | S1PC2 | S1PC1 | S1PC0 | R/W | STK1L |
| 0FDH | | | | | | S1PC10 | S1PC9 | S1PC8 | R/W | STK1H |
| 0FEH | S0PC7 | S0PC6 | S0PC5 | S0PC4 | S0PC3 | S0PC2 | S0PC1 | S0PC0 | R/W | STK0L |
| 0FFH | | | | | | S0PC10 | S0PC9 | S0PC8 | R/W | STK0H |

*** Note:**

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

2.1.10 ACCUMULATOR

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

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

; Read ACC data and store in BUF data memory.

```
MOV     BUF, A
```

; Write a immediate data into ACC.

```
MOV     A, #0FH
```

; Write ACC data from BUF data memory.

```
MOV     A, BUF
```

; or

```
B0MOV   A, BUF
```

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

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

INT_SERVICE:

```
PUSH                                ; Save ACC and PFLAG to buffers.
```

```
...
```

```
POP                                  ; Load ACC and PFLAG from buffers.
```

```
RETI                                 ; Exit interrupt service vector
```

2.1.11 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.

2.1.12 PROGRAM COUNTER

The program counter (PC) is a 11-bit binary counter separated into the high-byte 3 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

Besides, it can be replaced with specific address by executing CALL or JMP instruction. When JMP or CALL instruction is executed, the destination address will be inserted to bit 0 ~ bit 10.

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

ONE ADDRESS SKIPPING

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

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

```

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

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

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

```

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


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

INCS instruction:

INCS BUF0
JMP C0STEP ; Jump to C0STEP if ACC is not zero.

...

...

C0STEP: NOP

INCMS instruction:

INCMS BUF0
JMP C0STEP ; Jump to C0STEP if BUF0 is not zero.

...

...

C0STEP: NOP

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

DECS instruction:

DECS BUF0
JMP C0STEP ; Jump to C0STEP if ACC is not zero.

...

...

C0STEP: NOP

DECMS instruction:

DECMS BUF0
JMP C0STEP ; Jump to C0STEP if BUF0 is not zero.

...

...

C0STEP: NOP

☞ MULTI-ADDRESS JUMPING

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

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

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

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

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

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

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

2.1.13 Y, Z REGISTERS

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

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

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

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

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

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

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

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

CLR_YZ_BUF:

```
CLR      @YZ            ; Clear @YZ to be zero
```

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

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

...

2.1.14 R REGISTERS

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

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

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

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

2.2 ADDRESSING MODE

2.2.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.2.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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

2.2.3 INDIRECTLY ADDRESSING MODE

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

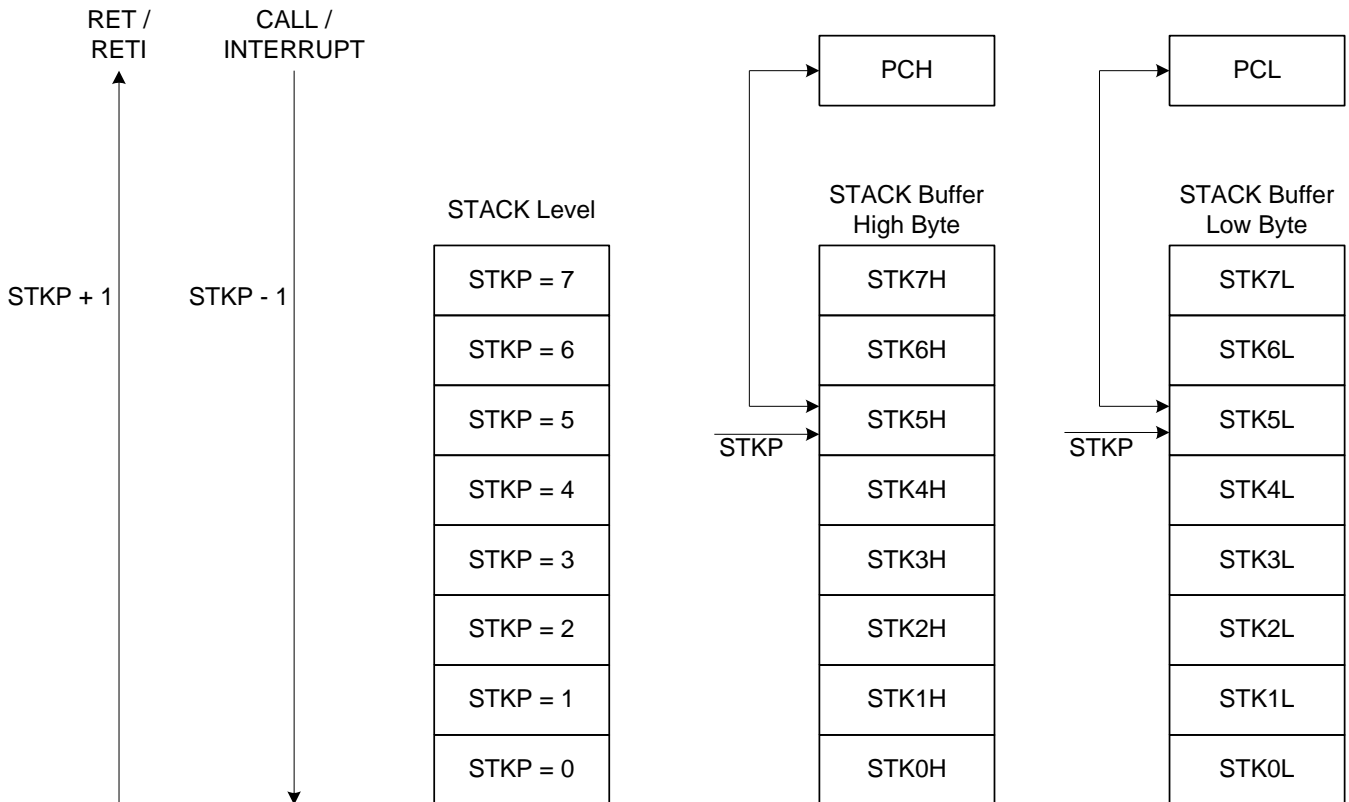
- **Example: Indirectly addressing mode with @YZ register.**

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

2.3 STACK OPERATION

2.3.1 OVERVIEW

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



2.3.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 11-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

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

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

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

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

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

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

```
MOV      A, #01111111B
B0MOV   STKP, A
```

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

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

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

2.3.3 STACK OPERATION EXAMPLE

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

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

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

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

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)

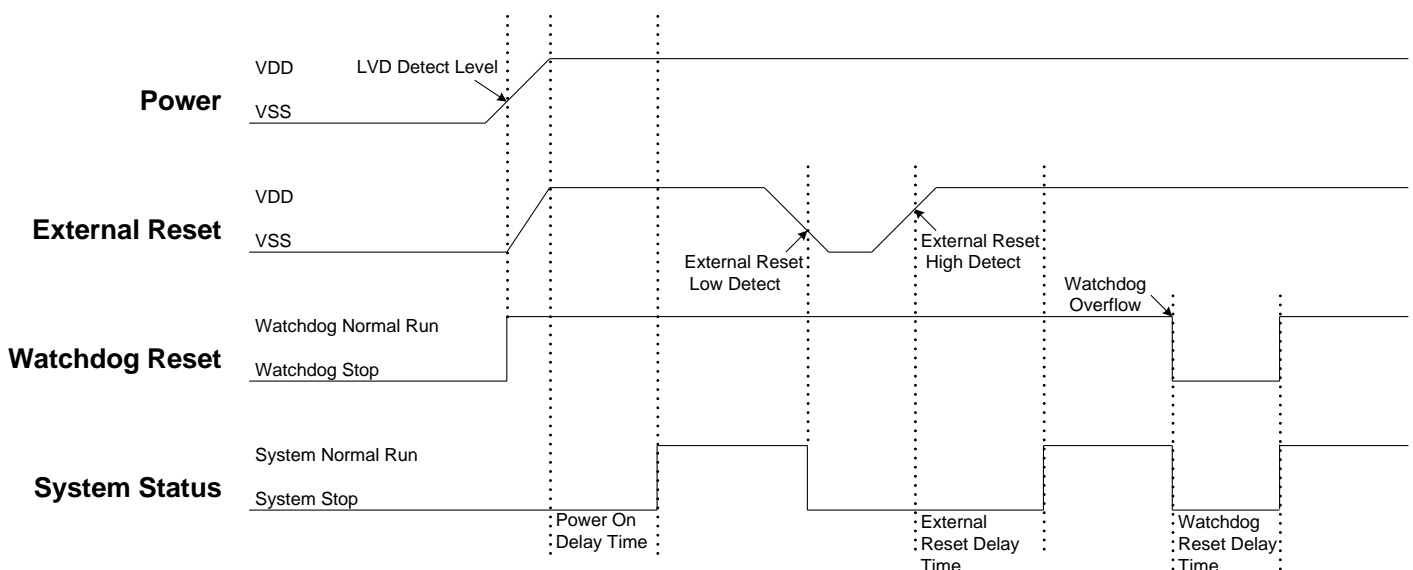
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

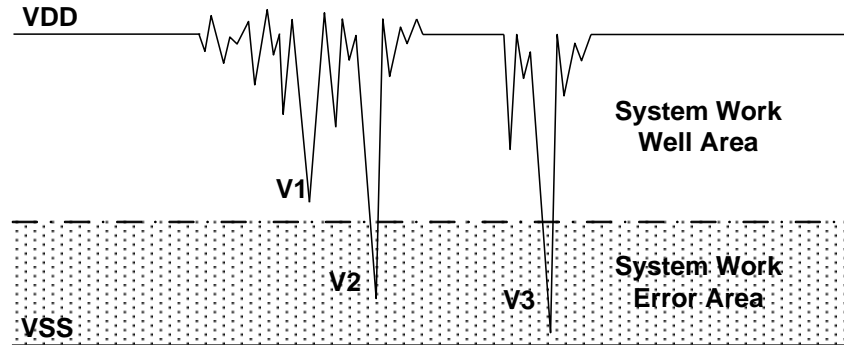
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

3.4.1 BROWN OUT DESCRIPTION

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



Brown Out Reset Diagram

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

DC application:

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

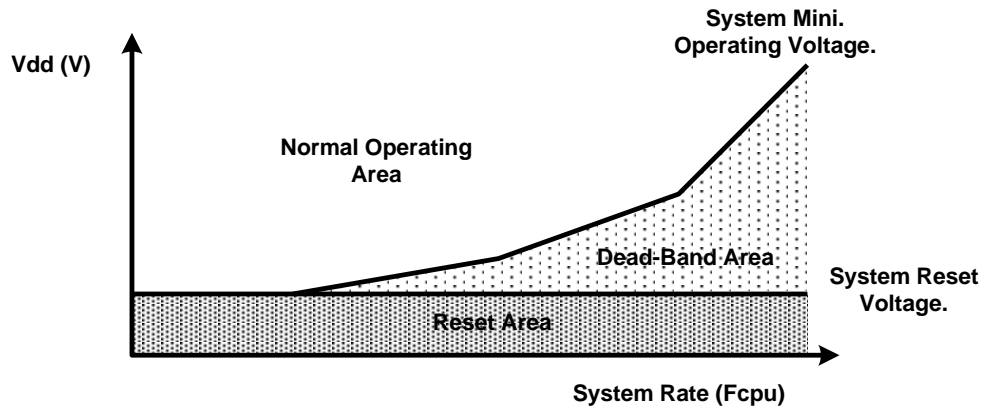
AC application:

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

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

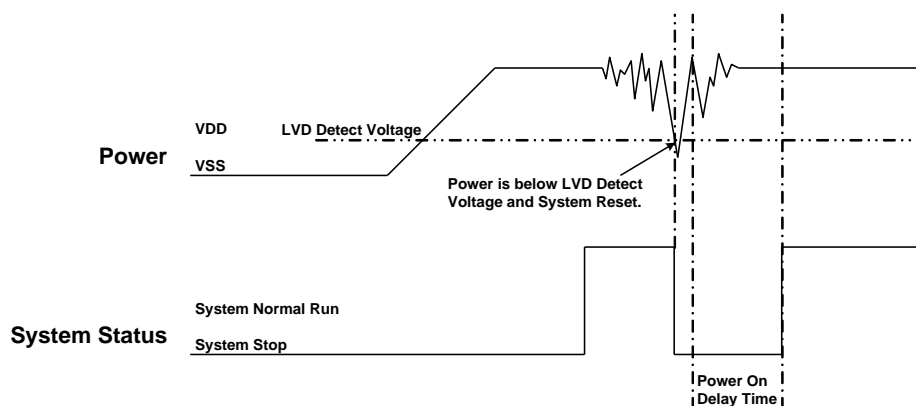
3.4.2 THE SYSTEM OPERATING VOLTAGE

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



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

3.4.3 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 |
| 2.0V Reset | Available | Available | Available |
| 2.4V Flag | - | Available | - |
| 2.4V Reset | - | - | Available |
| 3.6V Flag | - | - | 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.

LVD2_H

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

*** 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.**

3.4.4 BROWN OUT RESET IMPROVEMENT

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

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

* **Note:**

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

Watchdog reset:

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

If the system reset by watchdog and the power is still in dead-band, the system reset sequence won't be successful and the system stays in reset status until the power return to normal range. Watchdog timer application note is as following.

Reduce the system executing rate:

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

External reset circuit:

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

3.5 EXTERNAL RESET

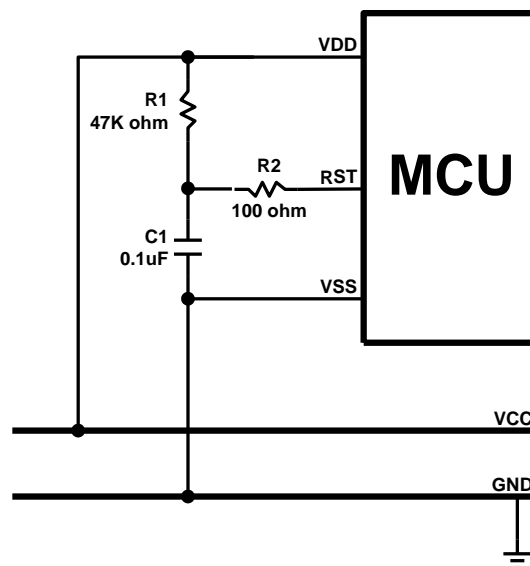
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

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

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

3.6 EXTERNAL RESET CIRCUIT

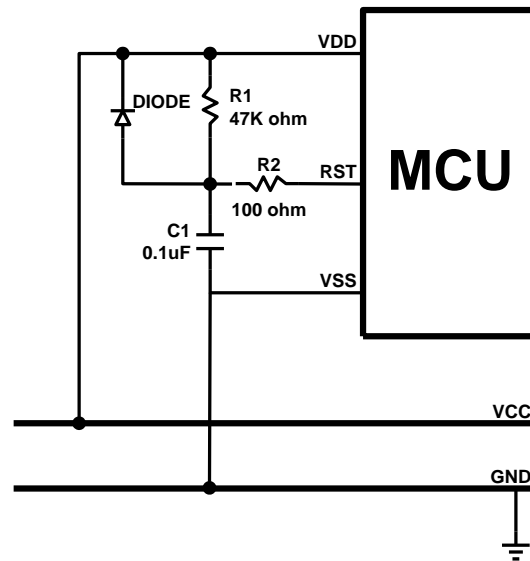
3.6.1 Simply RC Reset Circuit



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

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

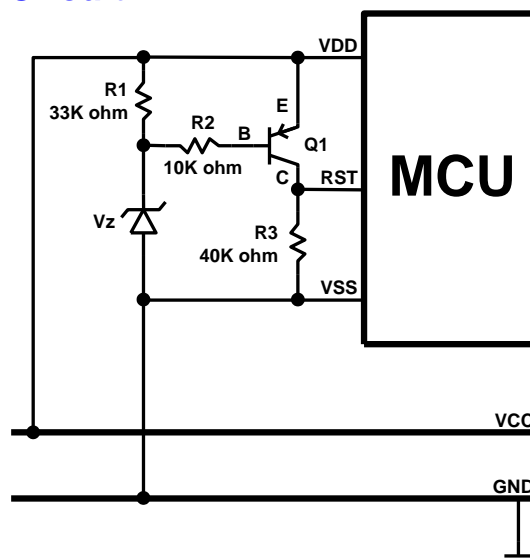
3.6.2 Diode & RC Reset Circuit



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

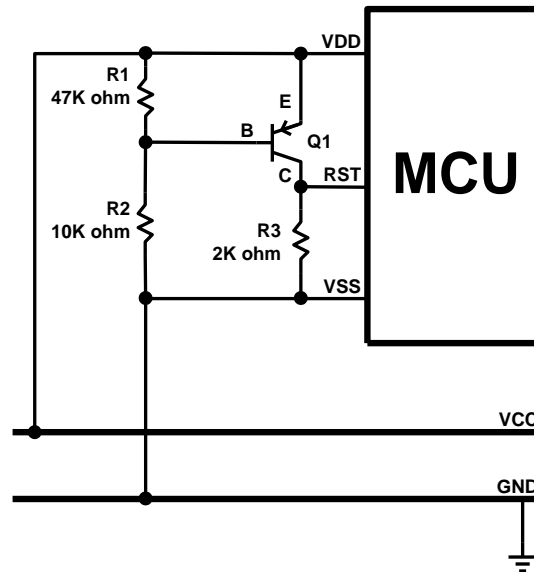
* **Note:** The R2 100 ohm resistor of “Simply reset circuit” and “Diode & RC reset circuit” is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).

3.6.3 Zener Diode Reset Circuit



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

3.6.4 Voltage Bias Reset Circuit

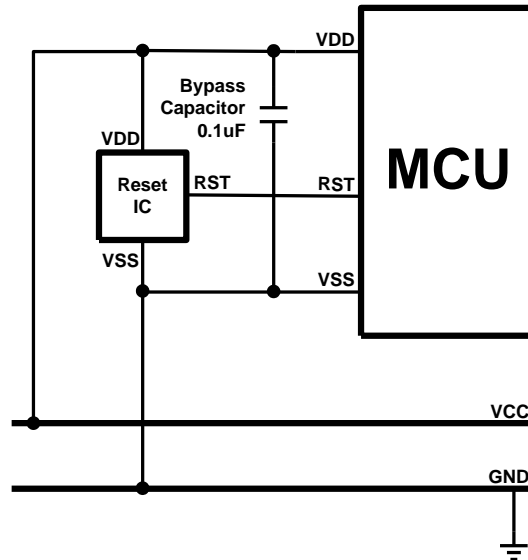


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to $0.7V \times (R1 + R2)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

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

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

3.6.5 External Reset IC



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

4 SYSTEM CLOCK

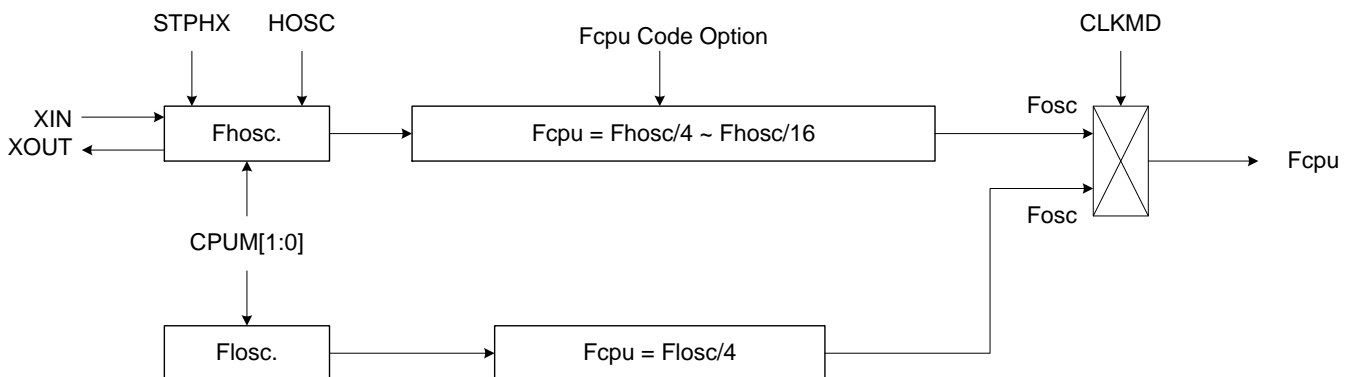
4.1 OVERVIEW

The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit or on-chip 16MHz high-speed RC oscillator circuit (IHRC 16MHz). The low-speed clock is generated from on-chip low-speed RC oscillator circuit (ILRC 16KHz @3V, 32KHz @5V). Both the high-speed clock and the low-speed clock can be system clock (Fosc). The system clock in slow mode is divided by 4 to be the instruction cycle (Fcpu).

- ☞ **Normal Mode (High Clock):** $F_{cpu} = F_{osc} / N$, $N = 4 \sim 16$, Select N by Fcpu code option.
- ☞ **Slow Mode (Low Clock):** $F_{cpu} = F_{osc}/4$.

SONiX provides a “Noise Filter” controlled by code option. In high noisy situation, the noise filter can isolate noise outside and protect system works well.

4.2 CLOCK BLOCK DIAGRAM



- HOSC: High_Clk code option.
- Fhosc: External high-speed clock / Internal high-speed RC clock.
- Fosc: Internal low-speed RC clock (about 16KHz@3V, 32KHz@5V).
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.3 OSCM REGISTER

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

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

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

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

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

➤ **Example: Stop high-speed oscillator**

```
B0BSET     FSTPHX                 ; To stop external high-speed oscillator only.
```

➤ **Example: When entering the power down mode (sleep mode), both high-speed oscillator and internal low-speed oscillator will be stopped.**

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

4.4 SYSTEM HIGH CLOCK

The system high clock is from internal 16MHz oscillator RC type or external oscillator. The high clock type is controlled by "High_Clk" code option.

| High_Clk Code Option | Description |
|----------------------|--|
| IHRC_16M | The high clock is internal 16MHz oscillator RC type. XIN and XOUT pins are general purpose I/O pins. |
| RC | The high clock is external RC type oscillator. XOUT pin is general purpose I/O pin. |
| 32K | The high clock is external 32768Hz low speed oscillator. |
| 12M | The high clock is external high speed oscillator. The typical frequency is 12MHz. |
| 4M | The high clock is external oscillator. The typical frequency is 4MHz. |

4.4.1 INTERNAL HIGH RC

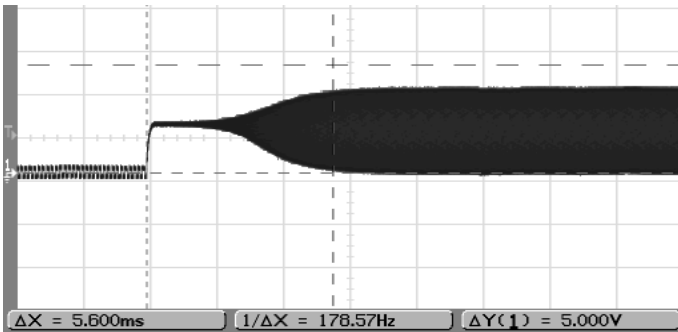
The chip is built-in RC type internal high clock (16MHz) controlled by "IHRC_16M" code option. In "IHRC_16M" mode, the system clock is from internal 16MHz RC type oscillator and XIN / XOUT pins are general-purpose I/O pins.

- **IHRC**: High clock is internal 16MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.

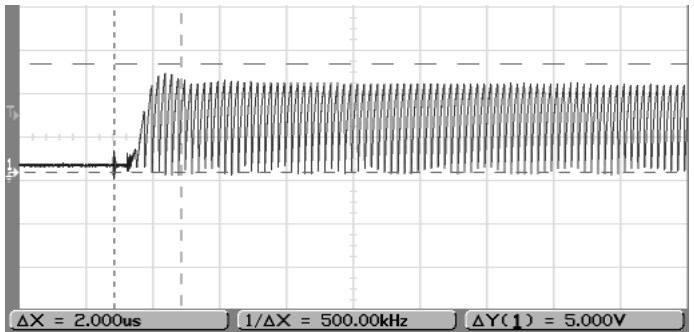
4.4.2 EXTERNAL HIGH CLOCK

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

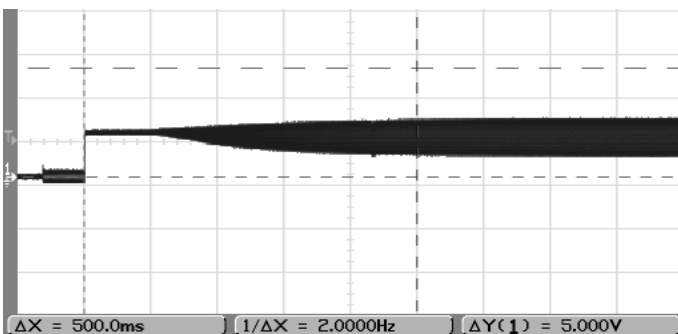
4MHz Crystal



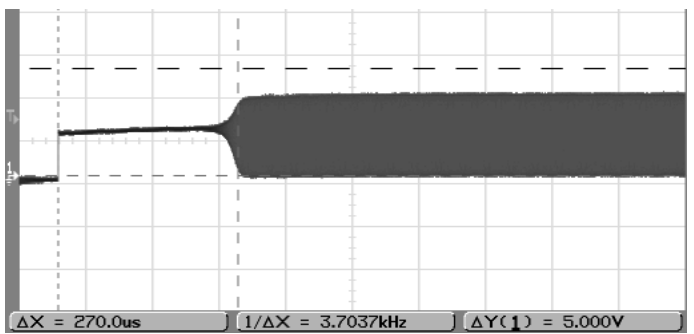
RC



32768Hz Crystal

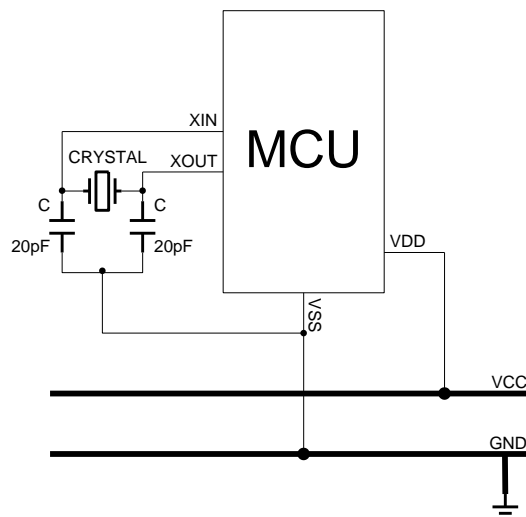


4MHz Ceramic



4.4.2.1 CRYSTAL/CERAMIC

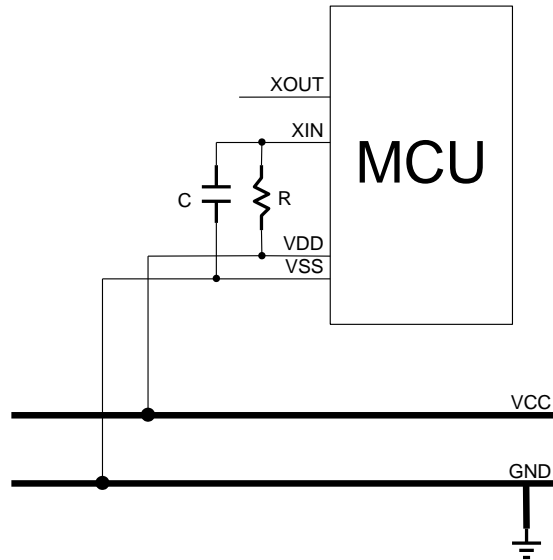
Crystal/Ceramic devices are driven by XIN, XOUT pins. For high/normal/low frequency, the driving currents are different. High_Clk code option supports different frequencies. 12M option is for high speed (ex. 12MHz). 4M option is for normal speed (ex. 4MHz). 32K option is for low speed (ex. 32768Hz).



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

4.4.2.2 RC

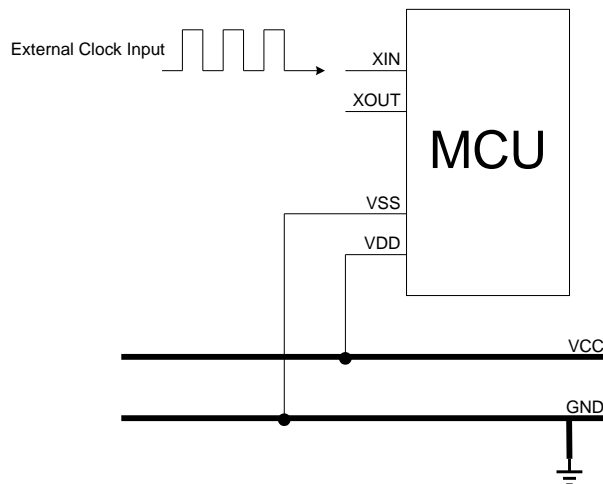
Selecting RC oscillator is by RC option of High_Clk code option. RC type oscillator's frequency is up to 10MHz. Using "R" value is to change frequency. 50P~100P is good value for "C". XOUT pin is general purpose I/O pin.



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

4.4.2.3 EXTERNAL CLOCK SIGNAL

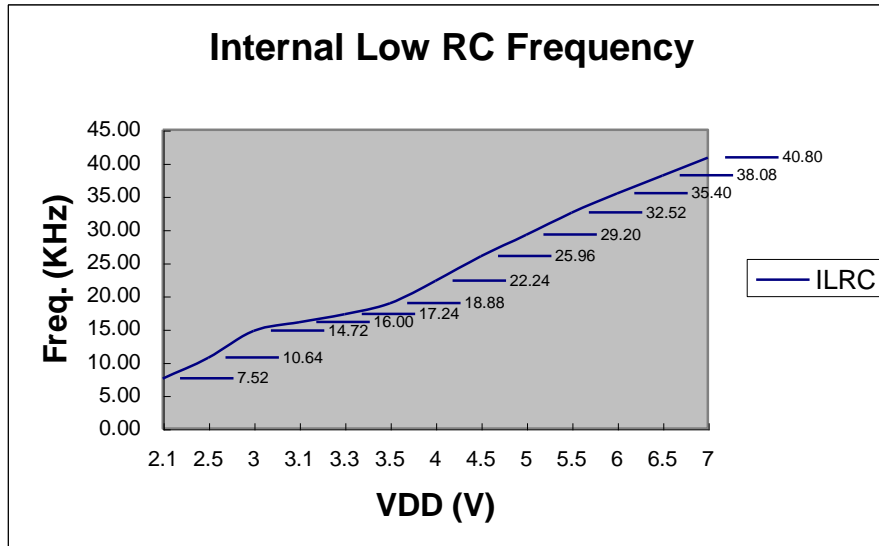
Selecting external clock signal input to be system clock is by RC option of High_Clk code option. The external clock signal is input from XIN pin. XOUT pin is general purpose I/O pin.



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

4.5 SYSTEM LOW CLOCK

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



The internal low RC supports watchdog clock source and system slow mode controlled by CLKMD.

- ☞ **F_{osc} = Internal low RC oscillator (about 16KHz @3V, 32KHz @5V).**
- ☞ **Slow mode $F_{cpu} = F_{osc} / 4$**

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

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

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

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

4.5.1 SYSTEM CLOCK MEASUREMENT

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

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

```
B0BSET    P0M.0        ; Set P0.0 to be output mode for outputting Fcpu toggle signal.
```

@ @:

```
B0BSET    P0.0        ; Output Fcpu toggle signal in low-speed clock mode.
B0BCLR    P0.0        ; Measure the Fcpu frequency by oscilloscope.
JMP       @B
```

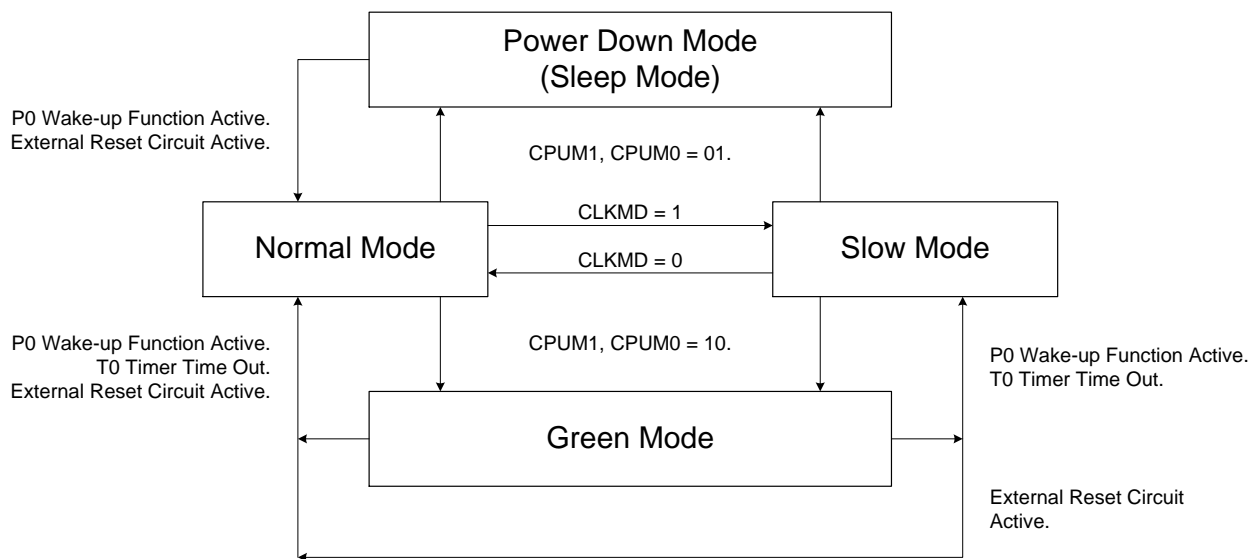
* **Note: Do not measure the RC frequency directly from XIN; the probe impedance will affect the RC frequency.**

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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



System Mode Switching Diagram

Operating mode description

| MODE | NORMAL | SLOW | GREEN | POWER DOWN (SLEEP) | REMARK |
|--------------------|--------------------------|--------------------------|--------------------------|--------------------------|----------------------------------|
| EHOSC | Running | By STPHX | By STPHX | Stop | |
| IHRC | Running | By STPHX | By STPHX | Stop | |
| ILRC | Running | Running | Running | Stop | |
| CPU instruction | Executing | Executing | Stop | Stop | |
| T0 timer | *Active | *Active | *Active | Inactive | * Active if T0ENB=1 |
| TC0 timer | *Active | *Active | Inactive | Inactive | * Active if TC0ENB=1 |
| Watchdog timer | By Watch_Dog Code option | By Watch_Dog Code option | By Watch_Dog Code option | By Watch_Dog Code option | Refer to code option description |
| Internal interrupt | All active | All active | T0 | All inactive | |
| External interrupt | All active | All active | All active | All inactive | |
| Wakeup source | - | - | P0, T0 Reset | P0, Reset | |

- **EHOSC:** External high clock
- **IHRC:** Internal high clock (16M RC oscillator)
- **ILRC:** Internal low clock (16K RC oscillator at 3V, 32K at 5V)

5.2 SYSTEM MODE SWITCHING EXAMPLE

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

```
BOBSET      FCPUM0      ; Set CPUM0 = 1.
```

* **Note: During the sleep, only the wakeup pin and reset can wakeup the system back to the normal mode.**

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

```
BOBSET      FCLKMD      ;To set CLKMD = 1, Change the system into slow mode
BOBSET      FSTPHX      ;To stop external high-speed oscillator for power saving.
```

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

```
BOBCLR      FCLKMD      ;To set CLKMD = 0
```

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

If external high clock stop and program want to switch back normal mode. It is necessary to delay at least 10mS for external clock stable.

```
BOBCLR      FSTPHX      ; Turn on the external high-speed oscillator.
MOV         A, #27      ; If VDD = 5V, internal RC=32KHz (typical) will delay
BOMOV      Z, A
@@:         DECMS      ; 0.125ms X 81 = 10.125ms for external clock stable
           JMP         @B
           ;
BOBCLR      FCLKMD      ; Change the system back to the normal mode
```

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

```
BOBSET      FCPUM1      ; Set CPUM1 = 1.
```

* **Note: If T0 timer wakeup function is disabled in the green mode, only the wakeup pin and reset pin can wakeup the system backs to the previous operation mode.**

➤ **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

| | | |
|--------|--------|--------------------|
| B0BCLR | FCPUM0 | ;To set CPUMx = 10 |
| B0BSET | FCPUM1 | |

* **Note: During the green mode with T0 wake-up function, the wakeup pin and T0 wakeup the system back to the last mode. T0 wake-up period is controlled by program.**

5.3 WAKEUP

5.3.1 OVERVIEW

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

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

5.3.2 WAKEUP TIME

When the system is in power down mode (sleep mode), the high clock oscillator stops. When waked up from power down mode, MCU waits for 2048 external high-speed oscillator clocks as the wakeup time to stable the oscillator circuit. After the wakeup time, the system goes into the normal mode.

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

The value of the wakeup time is as the following.

$$\text{The Wakeup time} = 1/F_{osc} * 2048 \text{ (sec)} + \text{high clock start-up time}$$

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

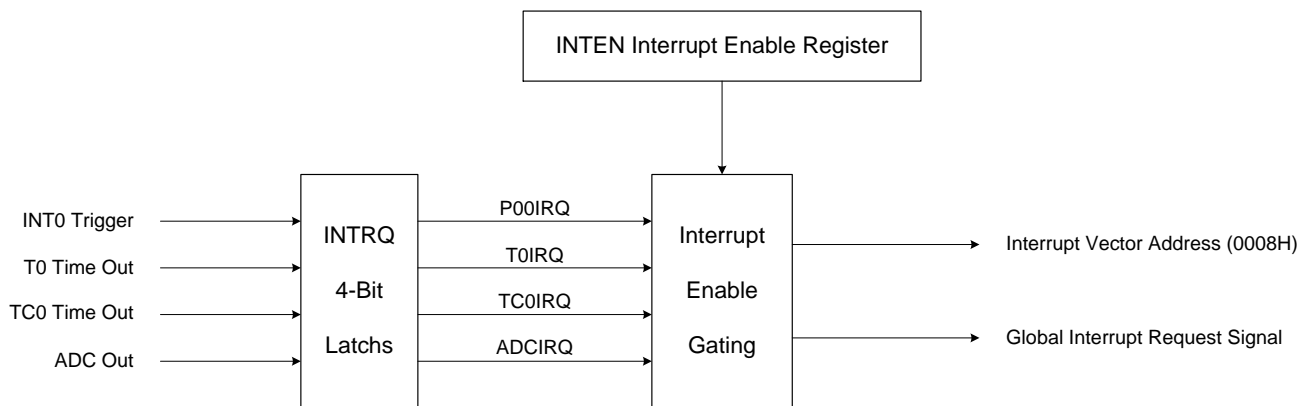
- **Example: In power down mode (sleep mode), the system is waked up. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.**

$$\begin{aligned} \text{The wakeup time} &= 1/F_{osc} * 2048 = 0.512 \text{ ms} \quad (F_{osc} = 4\text{MHz}) \\ \text{The total wakeup time} &= 0.512 \text{ ms} + \text{oscillator start-up time} \end{aligned}$$

6 INTERRUPT

6.1 OVERVIEW

This MCU provides eight interrupt sources, including three internal interrupt (T0/TC0/ADC) 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, and interrupt request is latched until return to normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to “0” for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to “1” to accept the next interrupts’ request. 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 three internal interrupts, two external interrupts enable control bits. One of the register to be set "1" is to enable the interrupt request function. Once of the interrupt occur, the stack is incremented and program jump to ORG 8 to execute interrupt service routines. The program exits the interrupt service routine when the returning interrupt service routine instruction (RETI) is executed.

| 0C9H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------------|--------|-------|--------|-------|-------|-------|-------|--------|
| INTEN | ADCIEN | - | TC0IEN | TOIEN | - | - | - | P00IEN |
| Read/Write | R/W | - | R/W | R/W | - | - | - | R/W |
| After reset | 0 | - | 0 | 0 | - | - | - | 0 |

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

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

Bit 5 **TC0IEN:** TC0 timer interrupt control bit.
0 = Disable TC0 interrupt function.
1 = Enable TC0 interrupt function.

Bit 7 **ADCIEN:** ADC interrupt control bit.
0 = Disable ADC interrupt function.
1 = Enable ADC interrupt function.

6.3 INTRQ INTERRUPT REQUEST REGISTER

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

| 0C8H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------------|--------|-------|--------|-------|-------|-------|-------|--------|
| INTRQ | ADCIRQ | - | TC0IRQ | TOIRQ | - | - | - | P00IRQ |
| Read/Write | R/W | - | R/W | R/W | - | - | - | R/W |
| After reset | 0 | - | 0 | 0 | - | - | - | 0 |

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

Bit 4 **TOIRQ:** 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.

Bit 7 **ADCIRQ:** ADC interrupt request flag.
0 = None ADC interrupt request.
1 = ADC interrupt request.

6.4 GIE GLOBAL INTERRUPT OPERATION

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

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

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

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

```
BOBSET           FGIE                   ; Enable GIE
```

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

6.5 PUSH, POP ROUTINE

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

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

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

```

ORG           0
JMP           START

ORG           8
JMP           INT_SERVICE

ORG           10H
START:
...

INT_SERVICE:
  PUSH                               ; Save ACC and PFLAG to buffers.
  ...
  ...
  POP                                ; Load ACC and PFLAG from buffers.
  RETI                               ; Exit interrupt service vector
  ...
  ENDP

```

6.6 EXTERNAL INTERRUPT OPERATION (INT0)

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

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

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

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

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

```

MOV      A, #98H
B0MOV    PEDGE, A      ; Set INT0 interrupt trigger as bi-direction edge.

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

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

```

ORG      8              ; Interrupt vector
JMP      INT_SERVICE

INT_SERVICE:
...          ; Push routine to save ACC and PFLAG to buffers.

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

B0BCLR   FP00IRQ      ; Reset P00IRQ
...          ; INT0 interrupt service routine

EXIT_INT:
...          ; Pop routine to load ACC and PFLAG from buffers.
RETI      ; Exit interrupt vector
  
```


6.7 T0 INTERRUPT OPERATION

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

➤ Example: T0 interrupt request setup.

```

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

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

B0BSET    FGIE      ; Enable GIE

```

➤ Example: T0 interrupt service routine.

```

ORG       8          ; Interrupt vector
JMP      INT_SERVICE

INT_SERVICE:

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

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

B0BCLR   FT0IRQ    ; Reset T0IRQ
MOV     A, #74H    ; Reset T0C.
B0MOV   T0C, A     ; 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 TC0IRQ is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

➤ 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

```

➤ 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    ; Reset TC0C.
B0MOV   TC0C, A    ; TC0 interrupt service routine
...
...

EXIT_INT:

...          ; Pop routine to load ACC and PFLAG from buffers.

RETI      ; Exit interrupt vector

```

6.9 ADC INTERRUPT OPERATION

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

➤ Example: ADC interrupt request setup.

```

B0BCLR      FADCIEN      ; Disable ADC interrupt service

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

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

B0BSET      FADS         ; Start ADC transformation

```

➤ Example: ADC interrupt service routine.

```

ORG         8             ; Interrupt vector
JMP        INT_SERVICE

INT_SERVICE:

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

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

B0BCLR    FADCIRQ      ; Reset ADCIRQ
...          ; ADC interrupt service routine
...

EXIT_INT:

...             ; Pop routine to load ACC and PFLAG from buffers.

RETI        ; Exit interrupt vector

```

6.10 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.

| <i>Interrupt Name</i> | <i>Trigger Event Description</i> |
|-----------------------|----------------------------------|
| P00IRQ | P0.0 trigger controlled by PEDGE |
| T0IRQ | T0C overflow |
| TC0IRQ | TC0C overflow |
| ADCIRQ | ADC converting end. |

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

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

```

ORG          8          ; Interrupt vector
JMP          INT_SERVICE

INT_SERVICE:
    ...                ; Push routine to save ACC and PFLAG to buffers.

INTP00CHK:                ; Check INT0 interrupt request
    B0BTS1    FP00IEN    ; Check P00IEN
    JMP       INTT0CHK    ; Jump check to next interrupt
    B0BTS0    FP00IRQ    ; Check P00IRQ
    JMP       INTP00

INTT0CHK:                ; Check T0 interrupt request
    B0BTS1    FT0IEN    ; Check T0IEN
    JMP       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       INTADCHK    ; Jump check to next interrupt
    B0BTS0    FTC0IRQ    ; Check TC0IRQ
    JMP       INTTC0      ; Jump to TC0 interrupt service routine

INTADCHK:                ; Check ADC interrupt request
    B0BTS1    FADCIEN    ; Check ADCIEN
    JMP       INT_EXIT    ; Jump to exit of IRQ
    B0BTS0    FADCIRQ    ; Check ADCIRQ
    JMP       INTADC      ; Jump to ADC interrupt service routine

INT_EXIT:
    ...                ; Pop routine to load ACC and PFLAG from buffers.

    RETI                ; Exit interrupt vector

```

7 I/O PORT

7.1 I/O PORT MODE

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

| 0B8H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| P0M | P07M | P06M | P05M | P04M | - | P02M | P01M | P00M |
| Read/Write | R/W | R/W | R/W | R/W | - | R/W | R/W | R/W |
| After reset | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 |

| 0C4H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| P4M | - | - | - | P44M | P43M | P42M | P42M | P40M |
| Read/Write | - | - | - | R/W | R/W | R/W | R/W | R/W |
| After reset | - | - | - | 0 | 0 | 0 | 0 | 0 |

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

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

- * **Note:**
1. Users can program them by bit control instructions (**B0BSET**, **B0BCLR**).
 2. **P0.3** input only pin, and the **P0M.3** keeps "1".

➤ Example: I/O mode selecting

```

CLR      P0M      ; Set all ports to be input mode.
CLR      P4M
CLR      P5M

MOV      A, #0FFH ; Set all ports to be output mode.
B0MOV   P0M, A
B0MOV   P4M, A
B0MOV   P5M, A

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

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

7.2 I/O PULL UP REGISTER

| 0E0H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| P0UR | P07R | P06R | P05R | P04R | - | P02R | P01R | P00R |
| Read/Write | W | W | W | W | - | W | W | W |
| After reset | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 |

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

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

* **Note:** P0.3 is input only pin and without pull-up resistor. The P0UR.3 keeps "1".

➤ **Example: I/O Pull up Register**

```

MOV          A, #0FFH          ; Enable Port0, 4, 5 Pull-up register,
B0MOV       P0UR, A           ;
B0MOV       P4UR, A
B0MOV       P5UR, A
    
```

7.3 I/O PORT DATA REGISTER

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

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

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

*** Note: The P03 keeps "1" when external reset enable by code option.**

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

```
B0MOV    A, P0           ; Read data from Port 0
B0MOV    A, P4           ; Read data from Port 4
B0MOV    A, P5           ; Read data from Port 5
```

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

```
MOV      A, #0FFH       ; Write data FFH to all Port.
B0MOV    P0, A
B0MOV    P4, A
B0MOV    P5, A
```

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

```
BOBSET   P4.0           ; Set P4.0 and P5.3 to be "1".
BOBSET   P5.3

BOBCLR   P4.0           ; Set P4.0 and P5.3 to be "0".
BOBCLR   P5.3
```

7.4 PORT 4 ADC SHARE PIN

The Port 4 is shared with ADC input function and no Schmitt trigger structure. Only one pin of port 4 can be configured as ADC input in the same time by ADM register. The other pins of port 4 are digital I/O pins. Connect an analog signal to COMS digital input pin, especially the analog signal level is about 1/2 VDD will cause extra current leakage. In the power down mode, the above leakage current will be a big problem. Unfortunately, if users connect more than one analog input signal to port 4 will encounter above current leakage situation. P4CON is Port4 Configuration register. Write "1" into P4CON.n will configure related port 4 pin as pure analog input pin to avoid current leakage.

| 0AFH | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------------|-------|-------|-------|--------|--------|--------|--------|--------|
| P4CON | - | - | - | P4CON4 | P4CON3 | P4CON2 | P4CON1 | P4CON0 |
| Read/Write | - | - | - | R/W | R/W | R/W | R/W | R/W |
| After reset | - | - | - | 0 | 0 | 0 | 0 | 0 |

Bit[4:0] **P4CON[4:0]**: P4.n configuration control bits.
 0 = P4.n can be an analog input (ADC input) or digital I/O pins.
 1 = P4.n is pure analog input, can't be a digital I/O pin.

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

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

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

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

Bit[2:0] **CHS[2:0]**: ADC input channels select bit.
 000 = AIN0, 001 = AIN1, 010 = AIN2, 011 = AIN3, 100 = AIN4, 101 = AIN5.

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

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

; Check GCHS and CHS[2:0] status.

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

; Clear P4CON.

| | | |
|--------|---------|---------------------------------|
| BOBCLR | P4CON.1 | ; Enable P4.1 digital function. |
|--------|---------|---------------------------------|

; Enable P4.1 input mode.

| | | |
|--------|-------|---------------------------|
| BOBCLR | P4M.1 | ; Set P4.1 as input mode. |
|--------|-------|---------------------------|

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

; Check GCHS and CHS[2:0] status.

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

; Clear P4CON.

| | | |
|--------|---------|---------------------------------|
| BOBCLR | P4CON.1 | ; Enable P4.1 digital function. |
|--------|---------|---------------------------------|

; Set P4.1 output buffer to avoid glitch.

| | | |
|--------|------|---------------------------|
| BOBSET | P4.1 | ; Set P4.1 buffer as "1". |
|--------|------|---------------------------|

; or

| | | |
|--------|------|---------------------------|
| BOBCLR | P4.1 | ; Set P4.1 buffer as "0". |
|--------|------|---------------------------|

; Enable P4.1 output mode.

| | | |
|--------|-------|---------------------------|
| BOBSET | P4M.1 | ; Set P4.1 as input mode. |
|--------|-------|---------------------------|

8 TIMERS

8.1 WATCHDOG TIMER

The watchdog timer (WDT) is a binary up counter designed for monitoring program execution. If the program goes into the unknown status by noise interference, WDT overflow signal raises and resets MCU. Watchdog clock controlled by code option and the clock source is internal low-speed oscillator (16KHz @3V, 32KHz @5V).

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

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

* **Note: If watchdog is "Always_On" mode, it keeps running event under power down mode or green mode.**

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

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

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

Main:

```

MOV      A,#5AH          ; Clear the watchdog timer.
B0MOV    WDTR,A
...
CALL     SUB1
CALL     SUB2
...
...
...
JMP     MAIN

```

Watchdog timer application note is as following.

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

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

Main:

...

; Check I/O.

...

; Check RAM

Err: JMP \$

; I/O or RAM error. Program jump here and don't

; clear watchdog. Wait watchdog timer overflow to reset IC.

Correct:

; I/O and RAM are correct. Clear watchdog timer and

; execute program.

MOV A,#5AH
B0MOV WDTR,A

; Only one clearing watchdog timer of whole program.

...

CALL SUB1

CALL SUB2

...

...

...

JMP MAIN

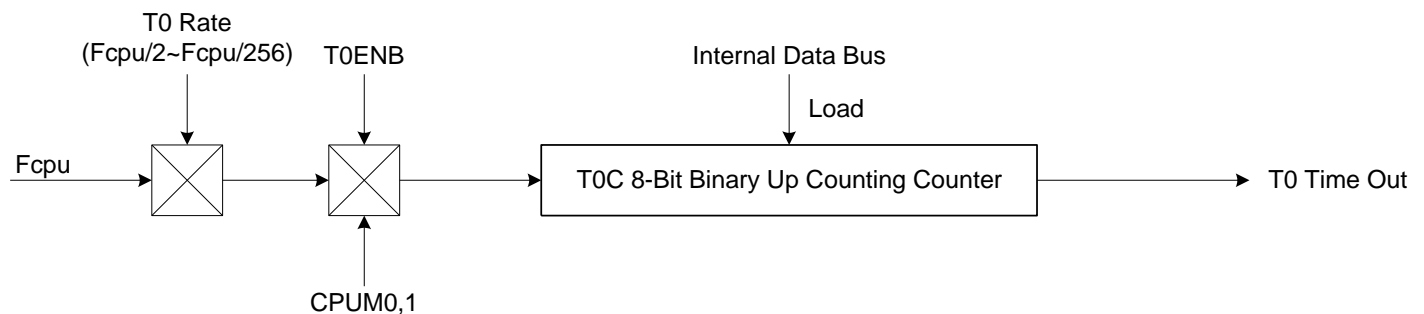
8.2 TIMER 0 (T0)

8.2.1 OVERVIEW

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

The main purposes of the T0 timer is as following.

- ☞ **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- ☞ **Green mode wakeup function:** T0 can be green mode wake-up time as $TOENB = 1$. System will be wake-up by T0 time out.



8.2.2 T0M MODE REGISTER

| 0D8H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|---------|---------|---------|---------|---------|-------|-------|
| T0M | TOENB | T0rate2 | T0rate1 | T0rate0 | TC0CKS2 | TC0CKS1 | - | - |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | - | - |
| After reset | 0 | 0 | 0 | 0 | 0 | 0 | - | - |

Bit [3:2] **TC0CKS[2:1]**: TC0 clock source select bits. Refer to TC0 chapter.

Bit [6:4] **T0RATE[2:0]**: T0 internal clock select bits.

000 = fcpu/256.

001 = fcpu/128.

...

110 = fcpu/4.

111 = fcpu/2.

Bit 7 **TOENB**: T0 counter control bit.

0 = Disable T0 timer.

1 = Enable T0 timer.

8.2.3 T0C COUNTING REGISTER

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

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

The equation of T0C initial value is as following.

$$\text{T0C initial value} = 256 - (\text{T0 interrupt interval time} * \text{input clock})$$

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

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

The basic timer table interval time of T0.

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

8.2.4 T0 TIMER OPERATION SEQUENCE

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

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

```

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

```

☞ **Set T0 timer rate.**

```

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

```

☞ **Set T0 interrupt interval time.**

```

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

```

☞ **Set T0 timer function mode.**

```

B0BSET    FT0IEN    ; Enable T0 interrupt function.

```

☞ **Enable T0 timer.**

```

B0BSET    FT0ENB    ; Enable T0 timer.

```

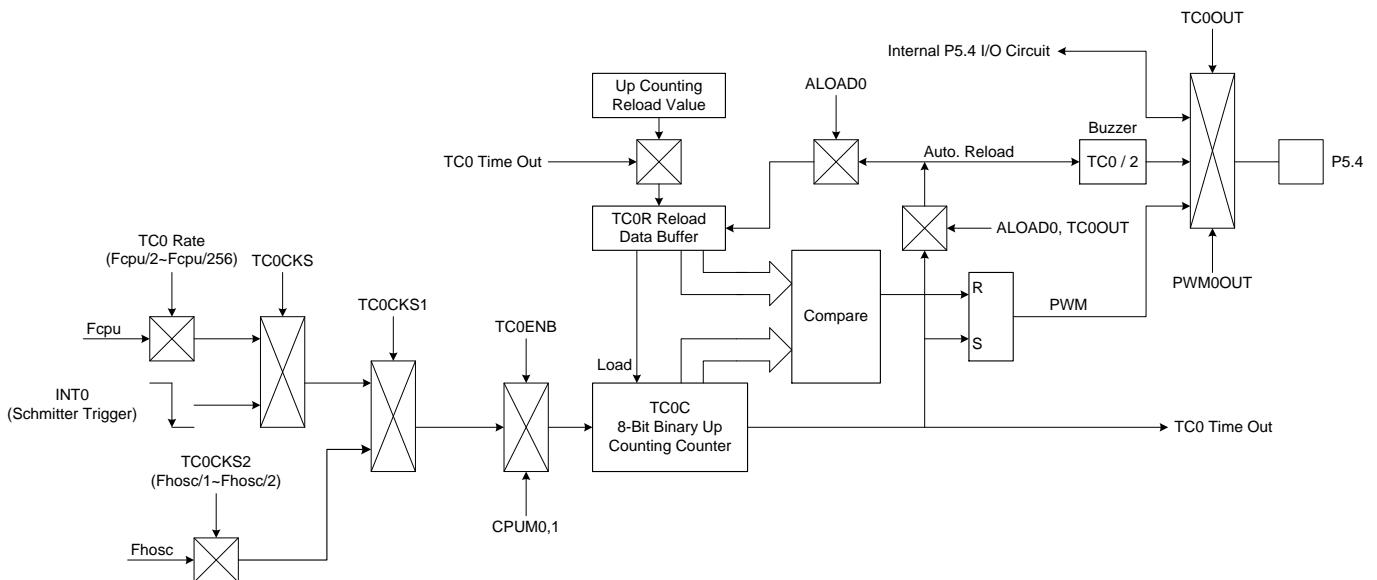
8.3 TIMER/COUNTER 0 (TC0)

8.3.1 OVERVIEW

The TC0 is an 8-bit binary up counting timer with double buffers. TC0 has two clock sources including internal clock and external clock for counting a precision time. The internal clock source is from Fcpu and Fhosc. The external clock is INT0 from P0.0 pin (Falling edge trigger). Using TC0M register selects TC0C's clock source from internal or external. If TC0 timer occurs an overflow, it will continue counting and issue a time-out signal to trigger TC0 interrupt to request interrupt service. TC0 overflow time is 0xFF to 0X00 normally. Under PWM mode, TC0 overflow is decided by PWM cycle controlled by ALOAD0 and TC0OUT bits.

The main purposes of the TC0 timer is as following.

- ☞ **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- ☞ **External event counter:** Counts system "events" based on falling edge detection of external clock signals at the INT0 input pin.
- ☞ **Buzzer output**
- ☞ **PWM output**



8.3.2 TC0M MODE REGISTER

| 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.
1 = Enable PWM output. 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 or Fhosc).
1 = External clock from P0.0/INT0 pin.
- Bit [6:4] **TC0RATE[2:0]:** TC0 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
...
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).**

| 0D8H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|---------|---------|---------|---------|---------|-------|-------|
| T0M | T0ENB | T0rate2 | T0rate1 | T0rate0 | TC0CKS2 | TC0CKS1 | - | - |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | - | - |
| After reset | 0 | 0 | 0 | 0 | 0 | 0 | - | - |

- Bit 2 **TC0CKS1:** TC0 clock source control bit.
0 = TC0 clock source is controlled by TC0CKS bit.
1 = TC0 clock source is controlled by TC0CKS2 bit.
- Bit 1 **TC0CKS2:** TC0 Fhosc clock source control bit.
0 = Fhosc/2
1 = Fhosc/1

8.3.3 TC0C COUNTING REGISTER

TC0C is an 8-bit counter register for TC0 interval time control.

| 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.

$$TC0C \text{ initial value} = N - (TC0 \text{ interrupt interval time} * \text{input clock})$$

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 |

- **Example: To set 10ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=0) and no PWM output (PWM0=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC0RATE=010 (Fcpu/64).**

$$\begin{aligned}
 TC0C \text{ initial value} &= N - (TC0 \text{ interrupt interval time} * \text{input clock}) \\
 &= 256 - (10\text{ms} * 4\text{MHz} / 4 / 64) \\
 &= 256 - (10^{-2} * 4 * 10^6 / 4 / 64) \\
 &= 100 \\
 &= 64H
 \end{aligned}$$

The basic timer table interval time of TC0.

| TC0RATE | TC0CLOCK | High speed mode (Fcpu = 4MHz / 4) | | Low speed mode (Fcpu = 32768Hz / 4) | |
|---------|----------|-----------------------------------|--------------------|-------------------------------------|--------------------|
| | | Max overflow interval | One step = max/256 | Max overflow interval | One step = max/256 |
| 000 | Fcpu/256 | 65.536 ms | 256 us | 8000 ms | 31250 us |
| 001 | Fcpu/128 | 32.768 ms | 128 us | 4000 ms | 15625 us |
| 010 | Fcpu/64 | 16.384 ms | 64 us | 2000 ms | 7812.5 us |
| 011 | Fcpu/32 | 8.192 ms | 32 us | 1000 ms | 3906.25 us |
| 100 | Fcpu/16 | 4.096 ms | 16 us | 500 ms | 1953.125 us |
| 101 | Fcpu/8 | 2.048 ms | 8 us | 250 ms | 976.563 us |
| 110 | Fcpu/4 | 1.024 ms | 4 us | 125 ms | 488.281 us |
| 111 | Fcpu/2 | 0.512 ms | 2 us | 62.5 ms | 244.141 us |

8.3.4 TC0R AUTO-LOAD REGISTER

TC0 timer is with auto-load function controlled by ALOAD0 bit of TC0M. When TC0C overflow occurring, TC0R value will load to TC0C by system. It is easy to generate an accurate time, and users don't reset TC0C during interrupt service routine.

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 TC0 interval time error and glitch in PWM and Buzzer output.

* **Note: Under PWM mode, auto-load is enabled automatically. The ALOAD0 bit is selecting overflow boundary.**

| 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.

$$TC0R \text{ initial value} = N - (TC0 \text{ interrupt interval time} * \text{input clock})$$

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.

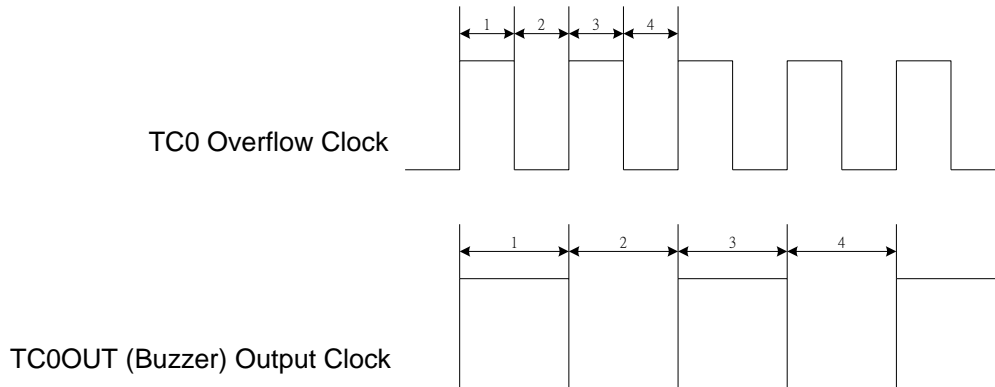
| TC0CKS | PWM0 | ALOAD0 | TC0OUT | N | TC0R valid value | TC0R value binary type |
|--------|------|--------|--------|-----|------------------|------------------------|
| 0 | 0 | x | x | 256 | 0x00~0xFF | 00000000b~11111111b |
| | 1 | 0 | 0 | 256 | 0x00~0xFF | 00000000b~11111111b |
| | 1 | 0 | 1 | 64 | 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 |

➤ **Example: To set 10ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=0) and no PWM output (PWM0=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC0RATE=010 (Fcpu/64).**

$$\begin{aligned}
 TC0R \text{ initial value} &= N - (TC0 \text{ interrupt interval time} * \text{input clock}) \\
 &= 256 - (10\text{ms} * 4\text{MHz} / 4 / 64) \\
 &= 256 - (10^{-2} * 4 * 10^6 / 4 / 64) \\
 &= 100 \\
 &= 64H
 \end{aligned}$$

8.3.5 TC0 CLOCK FREQUENCY OUTPUT (BUZZER)

Buzzer output (TC0OUT) is from TC0 timer/counter frequency output function. By setting the TC0 clock frequency, the clock signal is output to P5.4 and the P5.4 general purpose I/O function is auto-disable. The TC0OUT frequency is divided by 2 from TC0 interval time. TC0OUT frequency is 1/2 TC0 frequency. The TC0 clock has many combinations and easily to make difference frequency. The TC0OUT frequency waveform is as following.



- **Example: Setup TC0OUT output from TC0 to TC0OUT (P5.4). The external high-speed clock is 4MHz. $F_{cpu} = F_{osc}/4 = 1\text{MIPS}$. The TC0OUT frequency is 1KHz. Because the TC0OUT signal is divided by 2, set the TC0 clock to 2KHz. The TC0 clock source is from external oscillator clock. TC0 rate is $F_{cpu}/4$. The $TC0RATE2\text{--}TC0RATE1 = 110$. $TC0C = TC0R = 131$.**

```

MOV      A,#01100000B
B0MOV    TC0M,A          ; Set the TC0 rate to Fcpu/4

MOV      A,#131
B0MOV    TC0C,A          ; Set the auto-reload reference value
B0MOV    TC0R,A

B0BSET   FTC0OUT         ; Enable TC0 output to P5.4 and disable P5.4 I/O function
B0BSET   FALOAD1         ; Enable TC0 auto-reload function
B0BSET   FTC0ENB        ; Enable TC0 timer

```

* **Note: Buzzer output is enable, and "PWM0OUT" must be "0".**

8.3.6 TC0 TIMER OPERATION SEQUENCE

TC0 timer operation includes timer interrupt, event counter, TC0OUT and PWM. The sequence of setup TC0 timer is as following.

☞ **Stop TC0 timer counting, disable TC0 interrupt function and clear TC0 interrupt request flag.**

```
B0BCLR   FTC0ENB   ; TC0 timer, TC0OUT and PWM stop.
B0BCLR   FTC0IEN   ; TC0 interrupt function is disabled.
B0BCLR   FTC0IRQ   ; TC0 interrupt request flag is cleared.
```

☞ **Set TC0 timer rate. (Besides event counter mode.)**

```
MOV      A, #0xxx0000b ;The TC0 rate control bits exist in bit4~bit6 of TC0M. The
                          ; value is from x000xxxxb~x111xxxxb.
B0MOV    TC0M,A        ; TC0 interrupt function is disabled.
```

☞ **Set TC0 timer clock source.**

; Select TC0 internal / external clock source.

```
B0BCLR   FTC0CKS   ; Select TC0 internal clock source.
```

or

```
B0BSET   FTC0CKS   ; Select TC0 external clock source.
```

☞ **Set TC0 timer auto-load mode.**

```
B0BCLR   FALOAD0   ; Enable TC0 auto reload function.
```

or

```
B0BSET   FALOAD0   ; Disable TC0 auto reload function.
```

☞ **Set TC0 interrupt interval time, TC0OUT (Buzzer) frequency or PWM duty cycle.**

; Set TC0 interrupt interval time, TC0OUT (Buzzer) frequency or PWM duty.

```
MOV      A,#7FH      ; TC0C and TC0R value is decided by TC0 mode.
B0MOV    TC0C,A      ; Set TC0C value.
B0MOV    TC0R,A      ; Set TC0R value under auto reload mode or PWM mode.
```

; In PWM mode, set PWM cycle.

```
B0BCLR   FALOAD0   ; ALOAD0, TC0OUT = 00, PWM cycle boundary is
B0BCLR   FTC0OUT   ; 0~255.
```

or

```
B0BCLR   FALOAD0   ; ALOAD0, TC0OUT = 01, PWM cycle boundary is
B0BSET   FTC0OUT   ; 0~63.
```

or

```
B0BSET   FALOAD0   ; ALOAD0, TC0OUT = 10, PWM cycle boundary is
B0BCLR   FTC0OUT   ; 0~31.
```

or

```
B0BSET   FALOAD0   ; ALOAD0, TC0OUT = 11, PWM cycle boundary is
B0BSET   FTC0OUT   ; 0~15.
```

☞ **Set TC0 timer function mode.**

```
B0BSET   FTC0IEN   ; Enable TC0 interrupt function.
```

or

```
B0BSET   FTC0OUT   ; Enable TC0OUT (Buzzer) function.
```

or

```
B0BSET   FPWM0OUT  ; Enable PWM function.
```

☞ **Enable TC0 timer.**

```
B0BSET   FTC0ENB   ; Enable TC0 timer.
```

8.4 PWM0 MODE

8.4.1 OVERVIEW

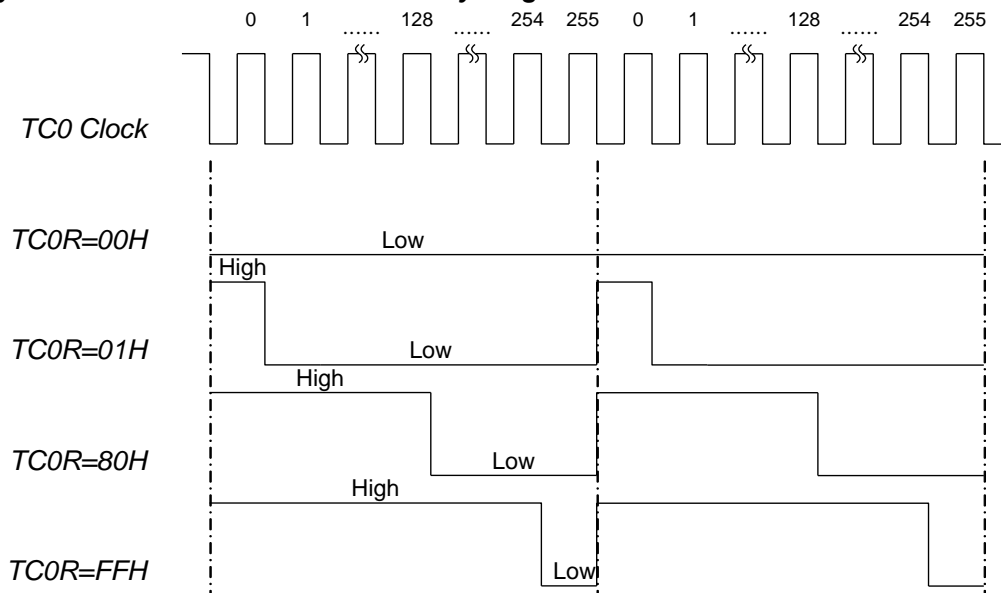
PWM function is generated by TC0 timer counter and output the PWM signal to PWM0OUT pin (P5.4). The 8-bit counter counts modulus 256, 64, 32, 16 controlled by ALOAD0, TC0OUT bits. The value of the 8-bit counter (TC0C) is compared to the contents of the reference register (TC0R). When the reference register value (TC0R) is equal to the counter value (TC0C), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWM0 output is TC0R/256, 64, 32, 16.

PWM output can be held at low level by continuously loading the reference register with 00H. Under PWM operating, to change the PWM's duty cycle is to modify the TC0R.

* **Note:** TC0 is double buffer design. Modifying TC0R to change PWM duty by program, there is no glitch and error duty signal in PWM output waveform. Users can change TC0R any time, and the new reload value is loaded to TC0R buffer at TC0 overflow.

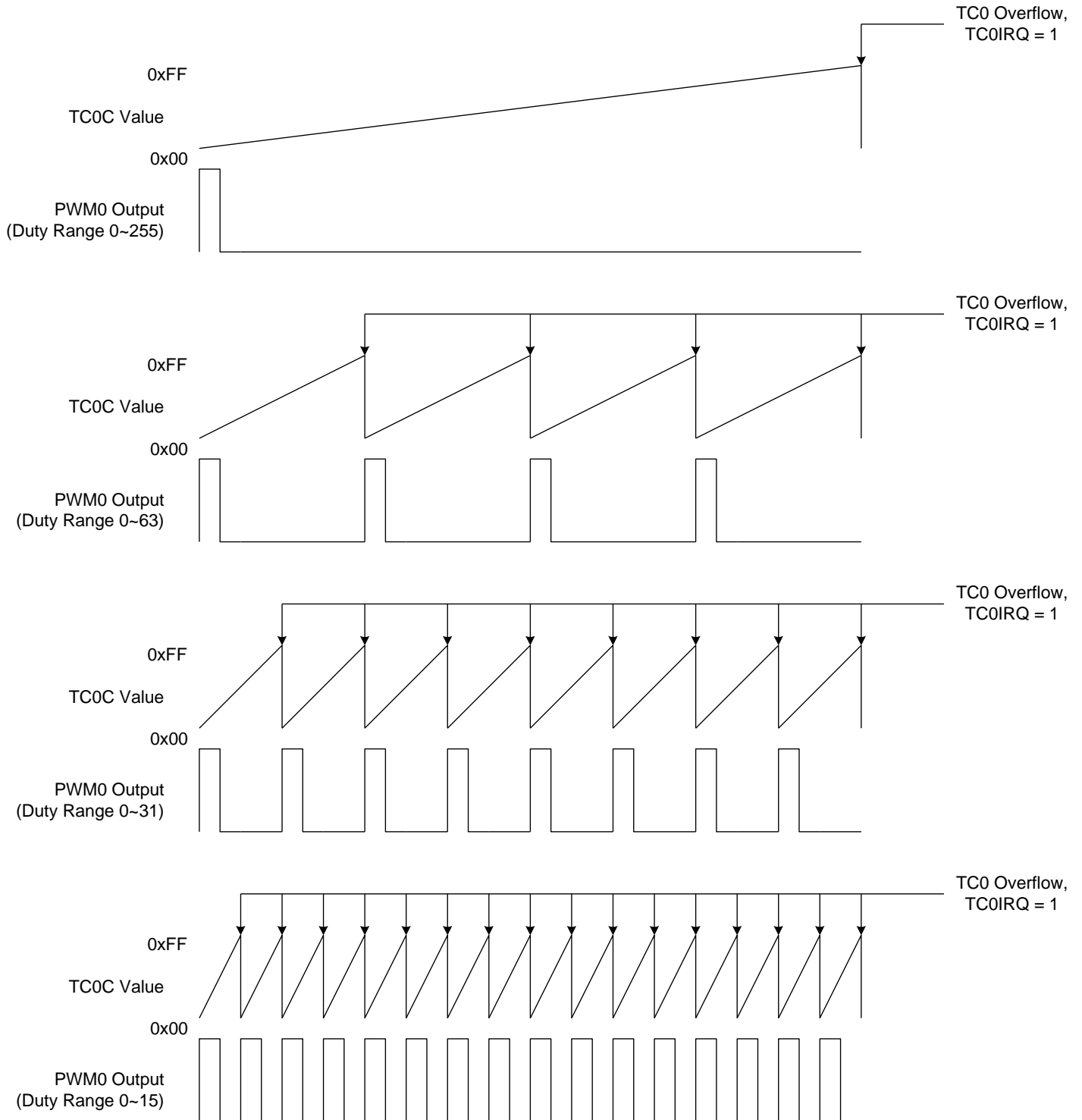
| ALOAD0 | TC0OUT | PWM duty range | TC0C valid value | TC0R valid bits value | MAX. PWM Frequency (F _{cpu} = 4MHz) | Remark |
|--------|--------|----------------|------------------|-----------------------|--|------------------------|
| 0 | 0 | 0/256~255/256 | 0x00~0xFF | 0x00~0xFF | 7.8125K | Overflow per 256 count |
| 0 | 1 | 0/64~63/64 | 0x00~0x3F | 0x00~0x3F | 31.25K | Overflow per 64 count |
| 1 | 0 | 0/32~31/32 | 0x00~0x1F | 0x00~0x1F | 62.5K | Overflow per 32 count |
| 1 | 1 | 0/16~15/16 | 0x00~0x0F | 0x00~0x0F | 125K | Overflow per 16 count |

The Output duty of PWM is with different TC0R. Duty range is from 0/256~255/256.



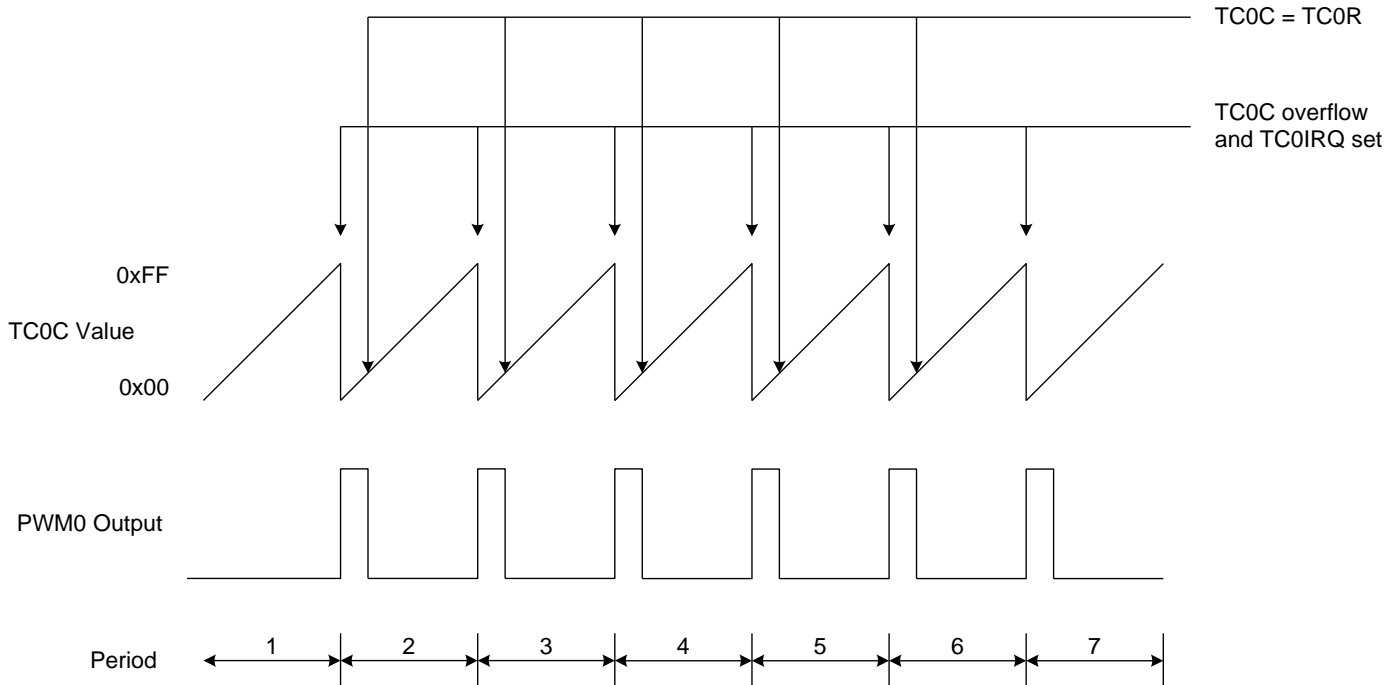
8.4.2 TC0IRQ and PWM Duty

In PWM mode, the frequency of TC0IRQ is depended on PWM duty range. From following diagram, the TC0IRQ frequency is related with PWM duty.

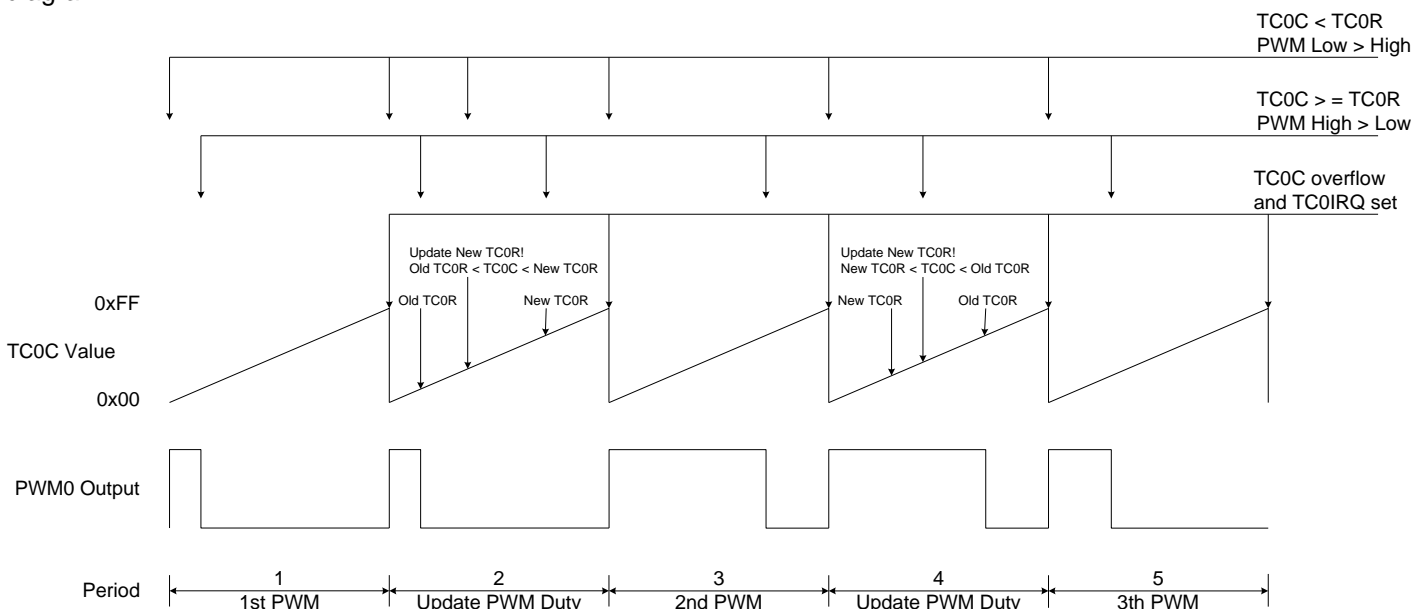


8.4.3 PWM Duty with TC0R Changing

In PWM mode, the system will compare TC0C and TC0R all the time. When $TC0C < TC0R$, the PWM will output logic "High", when $TC0C \geq TC0R$, the PWM will output logic "Low". If TC0C is changed in certain period, the PWM duty will change in next PWM period. If TC0R is fixed all the time, the PWM waveform is also the same.



Above diagram is shown the waveform with fixed TC0R. In every TC0C overflow PWM output "High, when $TC0C \geq TC0R$ PWM output "Low". If TC0R is changing in the program processing, the PWM waveform will become as following diagram.



In period 2 and period 4, new Duty (TC0R) is set. TC0 is double buffer design. The PWM still keeps the same duty in period 2 and period 4, and the new duty is changed in next period. By the way, system can avoid the PWM not changing or H/L changing twice in the same cycle and will prevent the unexpected or error operation.

8.4.4 PWM PROGRAM EXAMPLE

- **Example: Setup PWM0 output from TC0 to PWM0OUT (P5.4).** The external high-speed oscillator clock is 4MHz. $F_{cpu} = F_{osc}/4$. The duty of PWM is 30/256. The PWM frequency is about 1KHz. The PWM clock source is from external oscillator clock. TC0 rate is $F_{cpu}/4$. The $TC0RATE2-TC0RATE1 = 110$. $TC0C = TC0R = 30$.

```

MOV          A,#01100000B
B0MOV       TC0M,A           ; Set the TC0 rate to Fcpu/4

MOV          A,#30
B0MOV       TC0C,A           ; Set the PWM duty to 30/256
B0MOV       TC0R,A

B0BCLR      FTC0OUT          ; Set duty range as 0/256~255/256.
B0BCLR      FALOAD0
B0BSET      FPWM0OUT         ; Enable PWM0 output to P5.4 and disable P5.4 I/O function
B0BSET      FTC0ENB          ; Enable TC0 timer

```

* **Note: The TC0R is write-only register. Don't process them using INCMS, DECMS instructions.**

- **Example: Modify TC0R registers' value.**

```

MOV          A, #30H          ; Input a number using B0MOV instruction.
B0MOV       TC0R, A

INCMS       BUF0              ; Get the new TC0R value from the BUF0 buffer defined by
NOP                                     ; programming.
B0MOV       A, BUF0
B0MOV       TC0R, A

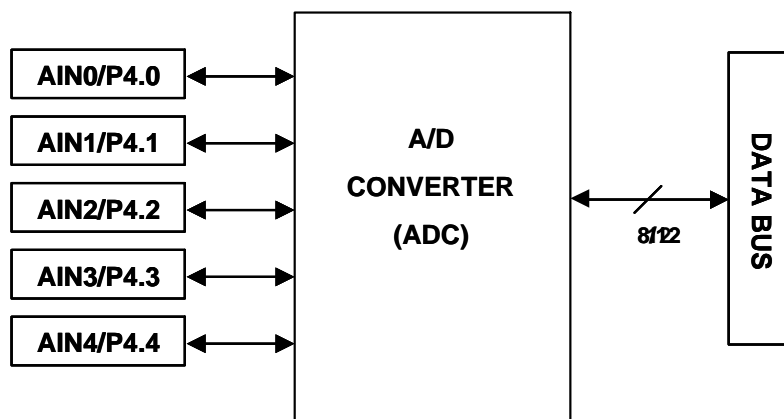
```

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

9 5 CHANNEL ANALOG TO DIGITAL CONVERTER

9.1 OVERVIEW

This analog to digital converter has 8-input sources with up to 4096-step resolution to transfer analog signal into 12-bits digital data. The sequence of ADC operation is to select input source (AIN0 ~ AIN4) at first, then set GCHS and ADS bit to "1" to start conversion. When the conversion is complete, the ADC circuit will set EOC bit to "1" and final value output in ADB register.



- * **Note:** The ADC 12-bit resolution conversion time is 16 steps.
- * **Note:** The analog input level must be between the VDD and VSS.
- * **Note:** ADC programming notice:
 1. Set ADC input pin I/O direction as input mode
 2. Disable pull-up resistor of ADC input pin
 3. Disable ADC (set ADENB = "0") before enter power down (sleep) mode to save power consumption.
 4. Set related bit of P4CON register to avoid extra power consumption in power down mode.
 5. Delay 100uS after enable ADC (set ADENB = "1") to wait ADC circuit ready for conversion.

9.2 ADM REGISTER

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

Bit 7 **ADENB**: ADC control bit.
0 = Disable.
1 = Enable.

Bit 6 **ADS**: ADC start bit.
0 = Stop.
1 = Starting.

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

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

Bit[2:0] **CHS[2:0]**: ADC input channels select bit.
000 = AIN0, 001 = AIN1, 010 = AIN2, 011 = AIN3, 100 = AIN4

* **Note:** If **ADENB = 1**, users should set **P4.n/AINn** as input mode without pull-up. System doesn't set automatically. If **P4CON.n** is set, the **P4.n/AINn**'s digital I/O function including pull-up is isolated.

9.3 ADR REGISTERS

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

Bit 6,4 **ADCKS [1:0]**: ADC's clock source select bit.

| ADCKS1 | ADCKS0 | ADC Clock Source |
|--------|--------|------------------|
| 0 | 0 | Fcpu/16 |
| 0 | 1 | Fcpu/8 |
| 1 | 0 | Fcpu/1 |
| 1 | 1 | Fcpu/2 |

Bit [3:0] **ADB [3:0]**: ADC data buffer.
ADB11~ADB4 bits for 8-bit ADC data.
ADB11~ADB0 bits for 12-bit ADC data.

* **Note:** ADC buffer **ADR [3:0]** initial value after reset is unknown.

9.4 ADB REGISTERS

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

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

ADB is ADC data buffer to store AD converter result. The ADB is only 8-bit register including bit 4~bit11 ADC data. To combine ADB register and the low-nibble of ADR will get full 12-bit ADC data buffer. The ADC buffer is a read-only register. In 8-bit ADC mode, the ADC data is stored in ADB register. In 12-bit ADC mode, the ADC data is stored in ADB and ADR registers.

The AIN's input voltage v.s. ADB's output data

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

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

| ADC Resolution | ADB | | | | | | | | ADR | | | |
|----------------|-------|-------|------|------|------|------|------|------|------|------|------|------|
| | ADB11 | ADB10 | ADB9 | ADB8 | ADB7 | ADB6 | ADB5 | ADB4 | ADB3 | ADB2 | ADB1 | ADB0 |
| 8-bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | x | x | x | x |
| 9-bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | x | x | x |
| 10-bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | x | x |
| 11-bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | x |
| 12-bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

○ = Selected, x = Delete

* **Note: ADC buffer ADB initial value after reset is unknown.**

9.5 P4CON REGISTERS

The Port 4 is shared with ADC input function. Only one pin of port 4 can be configured as ADC input in the same time by ADM register. The other pins of port 4 are digital I/O pins. Connect an analog signal to COMS digital input pin, especially the analog signal level is about 1/2 VDD will cause extra current leakage. In the power down mode, the above leakage current will be a big problem. Unfortunately, if users connect more than one analog input signal to port 4 will encounter above current leakage situation. P4CON is Port4 Configuration register. Write "1" into P4CON [7:0] will configure related port 4 pin as pure analog input pin to avoid current leakage.

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

Bit[4:0] **P4CON[4:0]**: P4.n configuration control bits.
 0 = P4.n can be an analog input (ADC input) or digital I/O pins.
 1 = P4.n is pure analog input, can't be a digital I/O pin.

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

9.6 ADC CONVERTING TIME

$$12\text{-bit ADC conversion time} = 1/(\text{ADC clock}/4) * 16 \text{ sec}$$

Fcpu = 4MHz (High clock, Fosc is 16MHz and Fcpu = Fosc/4)

| ADLEN | ADCKS1 | ADCKS0 | ADC Clock | ADC conversion time |
|------------|--------|--------|-----------|--|
| 1 (12-bit) | 0 | 0 | Fcpu/16 | $1/(4\text{MHz}/16/4) * 16 = 256 \text{ us}$ |
| | 0 | 1 | Fcpu/8 | $1/(4\text{MHz}/8/4) * 16 = 128 \text{ us}$ |
| | 1 | 0 | Fcpu/1 | $1/(4\text{MHz}/4) * 16 = 16 \text{ us}$ |
| | 1 | 1 | Fcpu/2 | $1/(4\text{MHz}/2/4) * 16 = 32 \text{ us}$ |

9.7 ADC ROUTINE EXAMPLE

➤ Example : Configure AIN0 as 12-bit ADC input and start ADC conversion then enter power down mode.

ADC0:

```

BOBSET      FADENB          ; Enable ADC circuit
CALL        Delay100uS    ; Delay 100uS to wait ADC circuit ready for conversion
MOV         A, #0FEh
BOBMOV     P4UR, A        ; Disable P4.0 pull-up resistor
BOBCLR     FP40M         ; Set P4.0 as input pin
MOV         A, #01h
BOBMOV     P4CON, A      ; Set P4.0 as pure analog input
MOV         A, #60H
BOBMOV     ADM, A        ; To set 12-bit ADC and ADC clock = Fosc.
MOV         A, #90H
BOBMOV     FADS          ; To enable ADC and set AIN0 input
BOBSET     FADS          ; To start conversion

```

WADC0:

```

BOBTS1     FEOC          ; To skip, if end of converting =1
JMP        WADC0        ; else, jump to WADC0
BOBMOV     A,ADB         ; To get AIN0 input data bit11 ~ bit4
BOBMOV     Adc_Buf_Hi, A
BOBMOV     A,ADR         ; To get AIN0 input data bit3 ~ bit0
AND        A, 0Fh
BOBMOV     Adc_Buf_Low, A

```

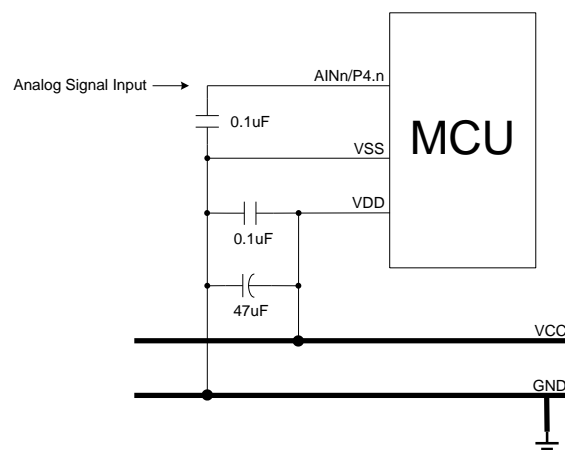
Power_Down

```

BOBCLR     FADENB        ; Disable ADC circuit
BOBCLR     FCPUM1
BOBSET     FCPUM0        ; Enter sleep mode

```

9.8 ADC CIRCUIT

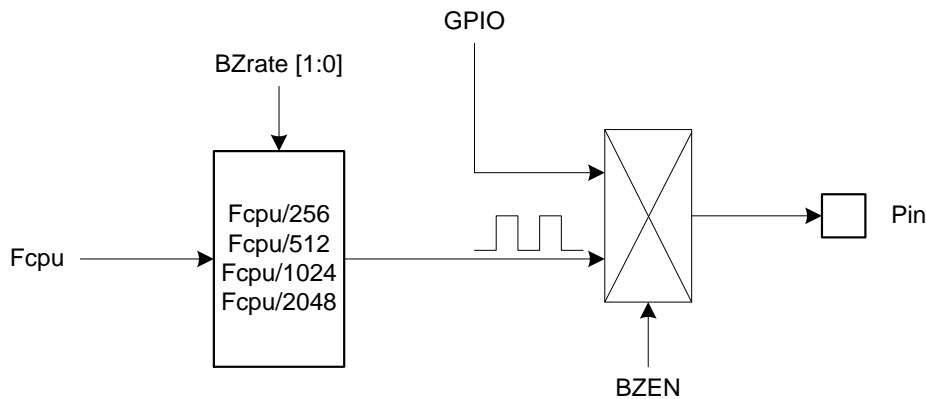


The 0.1uF capacitor near by ADC input pin is necessary to filter the power noise.

10_{2K/4K} BUZZER GENERATOR

10.1 OVERVIEW

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



The buzzer frequency is divided from Fcpu (instruction cycle) controlled by BZrate bits, and Fcpu decides the buzzer frequency. The selection table is as following.

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

The buzzer target frequency is 2KHz and 4KHz. It is important to choice a good Fcpu rate to obtain the correct buzzer frequency. The above table shows 2KHz/4KHz buzzer frequency configurations.

10.2 BZM REGISTER

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

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

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

* **Note:**

1. If **BZEN=1**, the P0.4 is buzzer output pin and isolates the GPIO function.
2. If **BZEN=0**, the P0.4 is GPIO mode and returns to last status after disabling buzzer output.

11 INSTRUCTION TABLE

| Field | Mnemonic | Description | C | DC | Z | Cycle |
|--------|---|--|----------------------------|----|-----|-------|
| MOV | A,M | $A \leftarrow M$ | - | - | √ | 1 |
| | M,A | $M \leftarrow A$ | - | - | - | 1 |
| | A,M | $A \leftarrow M$ (bank 0) | - | - | √ | 1 |
| | M,A | M (bank 0) $\leftarrow A$ | - | - | - | 1 |
| | A,I | $A \leftarrow I$ | - | - | - | 1 |
| | M,I | $M \leftarrow I$, "M" only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z....) | - | - | - | 1 |
| | A,M | $A \leftrightarrow M$ | - | - | - | 1+N |
| | A,M | $A \leftrightarrow M$ (bank 0) | - | - | - | 1+N |
| | | MOV _C | R, $A \leftarrow ROM[Y,Z]$ | - | - | - |
| ARITH | A,M | $A \leftarrow A + M + C$, if occur carry, then C=1, else C=0 | √ | √ | √ | 1 |
| | M,A | $M \leftarrow A + M + C$, if occur carry, then C=1, else C=0 | √ | √ | √ | 1+N |
| | A,M | $A \leftarrow A + M$, if occur carry, then C=1, else C=0 | √ | √ | √ | 1 |
| | M,A | $M \leftarrow A + M$, if occur carry, then C=1, else C=0 | √ | √ | √ | 1+N |
| | M,A | M (bank 0) $\leftarrow M$ (bank 0) + A, if occur carry, then C=1, else C=0 | √ | √ | √ | 1+N |
| | A,I | $A \leftarrow A + I$, if occur carry, then C=1, else C=0 | √ | √ | √ | 1 |
| | A,M | $A \leftarrow A - M - /C$, if occur borrow, then C=0, else C=1 | √ | √ | √ | 1 |
| | M,A | $M \leftarrow A - M - /C$, if occur borrow, then C=0, else C=1 | √ | √ | √ | 1+N |
| | A,M | $A \leftarrow A - M$, if occur borrow, then C=0, else C=1 | √ | √ | √ | 1 |
| | M,A | $M \leftarrow A - M$, if occur borrow, then C=0, else C=1 | √ | √ | √ | 1+N |
| | A,I | $A \leftarrow A - I$, if occur borrow, then C=0, else C=1 | √ | √ | √ | 1 |
| | A,M | R, $A \leftarrow A * M$, The LB of product stored in Acc and HB stored in R register. ZF affected by Acc. | - | - | √ | 2 |
| | LOGIC | A,M | $A \leftarrow A$ and M | - | - | √ |
| M,A | | $M \leftarrow A$ and M | - | - | √ | 1+N |
| A,I | | $A \leftarrow A$ and I | - | - | √ | 1 |
| A,M | | $A \leftarrow A$ or M | - | - | √ | 1 |
| M,A | | $M \leftarrow A$ or M | - | - | √ | 1+N |
| A,I | | $A \leftarrow A$ or I | - | - | √ | 1 |
| A,M | | $A \leftarrow A$ xor M | - | - | √ | 1 |
| M,A | | $M \leftarrow A$ xor M | - | - | √ | 1+N |
| A,I | | $A \leftarrow A$ xor I | - | - | √ | 1 |
| PROM | M | $A(b3-b0, b7-b4) \leftarrow M(b7-b4, b3-b0)$ | - | - | - | 1 |
| | M | $M(b3-b0, b7-b4) \leftarrow M(b7-b4, b3-b0)$ | - | - | - | 1+N |
| | M | $A \leftarrow RRC M$ | √ | - | - | 1 |
| | M | $M \leftarrow RRC M$ | √ | - | - | 1+N |
| | M | $A \leftarrow RLC M$ | √ | - | - | 1 |
| | M | $M \leftarrow RLC M$ | √ | - | - | 1+N |
| | M | $M \leftarrow 0$ | - | - | - | 1 |
| | M.b | $M.b \leftarrow 0$ | - | - | - | 1+N |
| | M.b | $M.b \leftarrow 1$ | - | - | - | 1+N |
| | M.b | $M(bank\ 0).b \leftarrow 0$ | - | - | - | 1+N |
| M.b | $M(bank\ 0).b \leftarrow 1$ | - | - | - | 1+N | |
| BRANCH | A,I | ZF,C $\leftarrow A - I$, If A = I, then skip next instruction | √ | - | √ | 1 + S |
| | A,M | ZF,C $\leftarrow A - M$, If A = M, then skip next instruction | √ | - | √ | 1 + S |
| | M | $A \leftarrow M + 1$, If A = 0, then skip next instruction | - | - | - | 1 + S |
| | M | $M \leftarrow M + 1$, If M = 0, then skip next instruction | - | - | - | 1+N+S |
| | M | $A \leftarrow M - 1$, If A = 0, then skip next instruction | - | - | - | 1 + S |
| | M | $M \leftarrow M - 1$, If M = 0, then skip next instruction | - | - | - | 1+N+S |
| | M.b | If M.b = 0, then skip next instruction | - | - | - | 1 + S |
| | M.b | If M.b = 1, then skip next instruction | - | - | - | 1 + S |
| | M.b | If M(bank 0).b = 0, then skip next instruction | - | - | - | 1 + S |
| | M.b | If M(bank 0).b = 1, then skip next instruction | - | - | - | 1 + S |
| | d | PC15/14 \leftarrow RomPages1/0, PC13-PC0 \leftarrow d | - | - | - | 2 |
| | d | Stack \leftarrow PC15-PC0, PC15/14 \leftarrow RomPages1/0, PC13-PC0 \leftarrow d | - | - | - | 2 |
| | RET | PC \leftarrow Stack | - | - | - | 2 |
| RETI | PC \leftarrow Stack, and to enable global interrupt | - | - | - | 2 | |
| PUSH | To push ACC and PFLAG (except NT0, NPD bit) into buffers. | - | - | - | 1 | |
| 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".

12 ELECTRICAL CHARACTERISTIC

12.1 ABSOLUTE MAXIMUM RATING

| | |
|--|-------------------------|
| Supply voltage (Vdd)..... | - 0.3V ~ 6.0V |
| Input in voltage (Vin)..... | Vss – 0.2V ~ Vdd + 0.2V |
| Operating ambient temperature (Topr) | |
| SN8P2722P, SN8P2722S, SN8P2722X, SN8P2722T..... | -20°C ~ + 85°C |
| SN8P2722PD, SN8P2722SD, SN8P2722XD, SN8P2722TD | -40°C ~ + 85°C |
| Storage ambient temperature (Tstor) | -40°C ~ + 125°C |

12.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, fosc = 4MHz, 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 | 2.4 | 5.0 | 5.5 | V | |
| | | Normal mode, Vpp = Vdd, -40°C~85°C | 2.5 | 5.0 | 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 | |
| | ViL3 | P4 ADC shared pin | Vss | - | 0.5Vdd | V | |
| Input High Voltage | ViH1 | All input ports | 0.7Vdd | - | Vdd | V | |
| | ViH2 | Reset pin | 0.9Vdd | - | Vdd | V | |
| | ViH3 | P4 ADC shared pin | 0.5Vdd | - | Vdd | V | |
| Reset pin leakage current | Ilekg | 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 port input leakage current | Ilekg | Pull-up resistor disable, Vin = Vdd | - | - | 2 | uA | |
| I/O output source current | IoH | Vop = Vdd – 0.5V | 8 | 12 | - | mA | |
| sink current | IoL | Vop = Vss + 0.5V | 8 | 15 | - | | |
| INTn trigger pulse width | Tint0 | INT0 interrupt request pulse width | 2/fcpu | - | - | cycle | |
| Supply Current (Disable ADC) | Idd1 | Run Mode (No loading, Fcpu = Fosc/4) | Vdd= 5V, 4Mhz | - | 2.5 | 5 | mA |
| | | | Vdd= 3V, 4Mhz | - | 1 | 2 | |
| | Idd2 | Slow Mode (Internal low RC, Stop high clock) | Vdd= 5V, 32Khz | - | 20 | 40 | uA |
| | | | Vdd= 3V, 16Khz | - | 5 | 10 | |
| | Idd3 | Sleep Mode | Vdd= 5V, 25°C | - | 0.8 | 1.6 | uA |
| | | | Vdd= 3V, 25°C | - | 0.7 | 1.4 | |
| | | | Vdd= 5V, -40°C~ 85°C | - | 10 | 21 | |
| | | | Vdd= 3V, -40°C~ 85°C | - | 10 | 21 | |
| | Idd4 | Green Mode (No loading, Fcpu = Fosc/4 Watchdog Disable) | Vdd= 5V, 4Mhz | - | 0.6 | 1.2 | mA |
| | | | Vdd= 3V, 4Mhz | - | 0.25 | 0.5 | |
| Vdd=5V, ILRC 32Khz | | | - | 15 | 30 | | |
| Vdd=3V, ILRC 16Khz , | | | - | 3 | 6 | | |
| Internal High Oscillator Freq. | Fihrc | Internal Hihg RC (IHRC) | 25°C, Vdd= 5V, Fcpu = 1MHz | 15.68 | 16 | 16.32 | Mhz |
| | | | -40°C~85°C, Vdd= 2.4V~5.5V, Fcpu = 1MHz~16 MHz | 13 | 16 | 19 | Mhz |
| LVD Voltage | Vdet0 | Low voltage reset level. | 1.6 | 2.0 | 2.3 | V | |
| | Vdet1 | Low voltage reset level. Fcpu = 1 MHz. Low voltage indicator level. Fcpu = 1 MHz. | 2.0 | 2.3 | 3 | V | |
| | Vdet2 | Low voltage indicator level. Fcpu = 1 MHz | 2.7 | 3.3 | 4.5 | V | |
| AIN0 ~ AIN5 input voltage | Vani | Vdd = 5.0V | 0 | - | Vrefh1~ 5 | V | |
| ADC enable time | Tast | Ready to start convert after set ADENB = "1" | 100 | - | - | us | |
| ADC current consumption | IADC | Vdd=5.0V | - | 0.6 | - | mA | |
| | | Vdd=3.0V | - | 0.4 | - | | |
| ADC Clock Frequency | FADCLK | VDD=5.0V | 32K | | 8M | Hz | |
| | | VDD=3.0V | 32K | | 5M | | |
| ADC Conversion Cycle Time | FADCYL | VDD=2.4V~5.5V | 64 | | | 1/FADCLK | |

| | | | | | | |
|---|--------------------|--|----|----|-----|-------|
| ADC Sampling Rate (Set FADS=1 Frequency) | F _{ADSMP} | VDD=5.0V | | | 125 | K/sec |
| | | VDD=3.0V | | | 80 | K/sec |
| Differential Nonlinearity | DNL | VDD=5.0V , AVREFH=3.2V, F _{ADSMP} =7.8K | ±1 | ±2 | ±16 | LSB |
| Integral Nonlinearity | INL | VDD=5.0V , AVREFH=3.2V, F _{ADSMP} =7.8K | ±2 | ±4 | ±16 | LSB |
| No Missing Code | NMC | VDD=5.0V , AVREFH=3.2V, F _{ADSMP} =7.8K | 8 | 10 | 12 | Bits |

*These parameters are for design reference, not tested.

➤ **Internal 16MHz Oscillator RC Type Temperature Characteristic.**

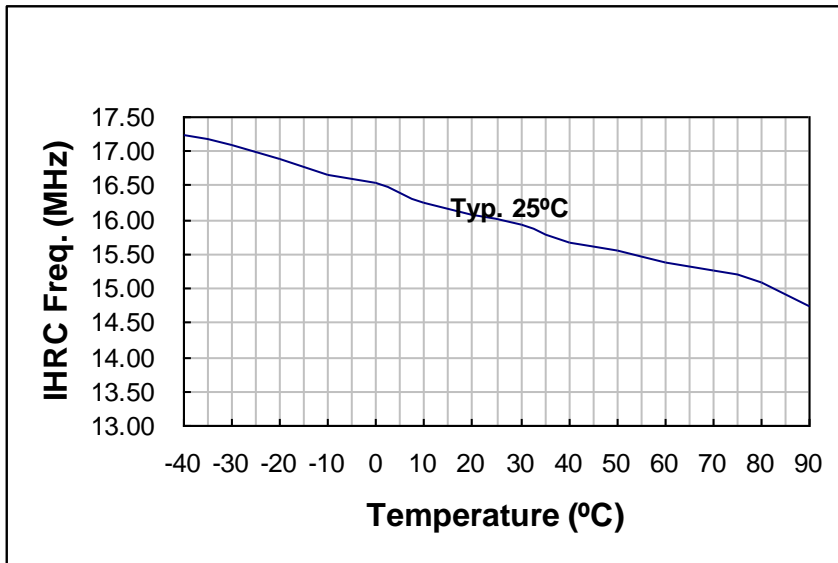
Power Voltage (VDD) = 5V.

Machine Cycle (F_{cpu}) = F_{osc}/4.

Typical Temperature = 25°C.

Typical Internal 16MHz Oscillator RC Type Frequency = 16MHz.

Testing Temperature Range = -40°C ~ + 90°C



➤ **Internal 16MHz Oscillator RC Type Power Voltage and Machine Cycle Characteristic.**

Temperature = 25°C.

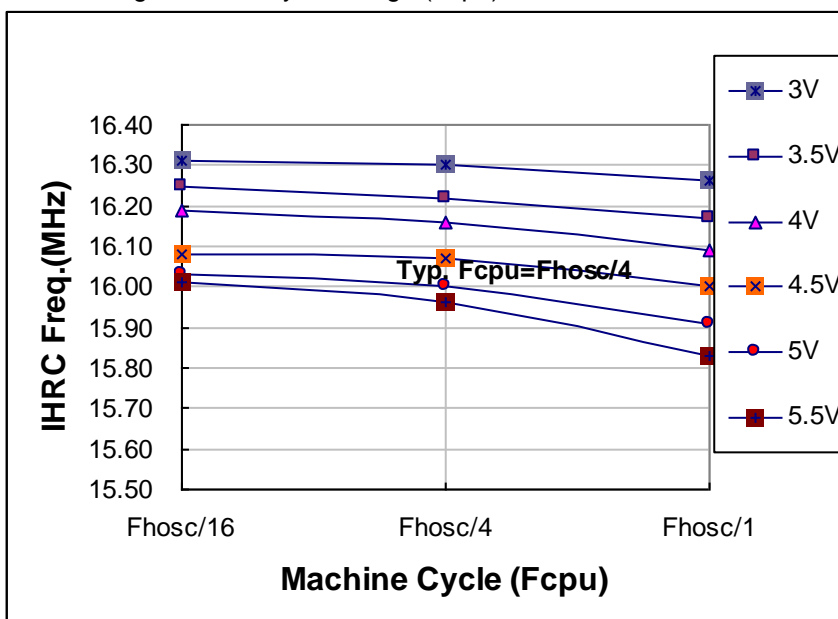
Typical Power Voltage (VDD) = 5V.

Typical Machin Cycle (F_{cpu}) = F_{osc} / 4.

Typical Internal 16MHz Oscillator RC Type Frequency = 16MHz.

Testing Power Voltage Range (VDD) = 3V~5.5V.

Testing Machine Cycle Range (F_{cpu}) = F_{osc}/1~F_{osc}/16.



13 SN8P2722 DEVELOPMENT TOOL

SONiX provides ICE (in circuit emulation), IDE (Integrated Development Environment) and EV-kit for SN8P2722 development. ICE and EV-kit are external hardware devices, and IDE is a friendly user interface for firmware development and emulation. These development tools' version is as following.

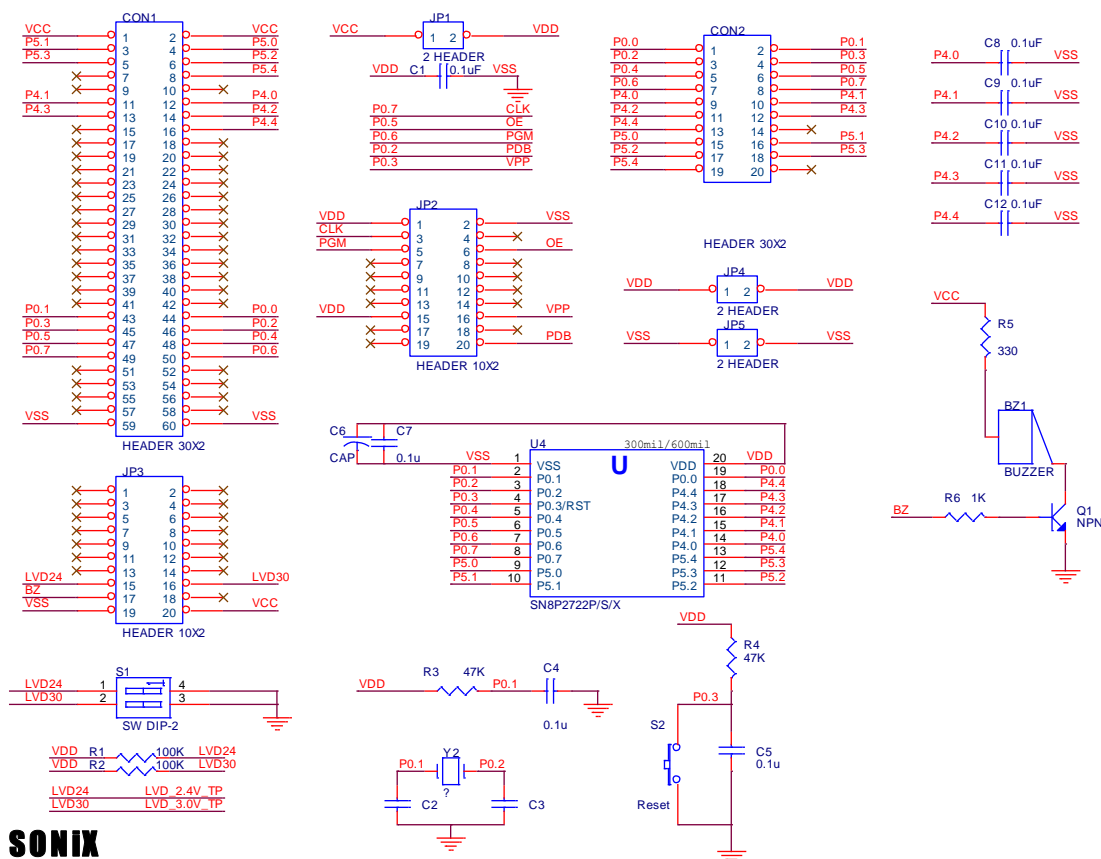
- ICE: SN8ICE2K
- EV-kit: SN8P2722 EV-kit Rev. A.
- IDE: SONiX IDE M2IDE 070604. This is brief version.
- Writer: MPIII writer.

13.1 SN8P2722 EV-kit

SN8P2722 EV-kit includes ICE interface, GPIO interface and Buzzer module.

- Buzzer module: .Emulate 2KHz/4KHz buzzer output function.

The schematic of SN8P2722 EV-kit is as following.



SONiX
SN8P2722 V3 Transition Board WRxxxx
SN8P2722 EV Kit REV:A

- CON1, JP3: ICE interface connected to SN8ICE2K .
- U4: SN8P2722 DIP form connector for connecting to user's target board.
- CON2: GPIO connector.
- JP2: Writer interface connected to MPIII writer.
- S1: LVD24 and LVD36 emulating switch.
- BZ1: Buzzer module.

13.2 ICE and EV-KIT APPLICATION NOTIC

SN8P2722 Buzzer output pin is shared with P0.4 GPIO pin. In ICE environment, the P0.4 GPIO pin isn't connected with buzzer output pin. The Buzzer module is independent in the SN8P2722 EV-kit.

- The Buzzer emulation is from the buzzer module of SN8P2722 EV-kit. The P0.4 pin of EV-kit doesn't output buzzer signal.
- The P0.4 emulation is from P0.4 pin of SN8P2722-EV-kit.

14 OTP PROGRAMMING PIN

14.1 EASY WRITER TRANSITION BOARD SOCKET PIN ASSIGNMENT

Easy Writer JP1/JP2

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

JP1 for MP transition board

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

| | | | |
|-------|----|----|-------|
| DIP1 | 1 | 48 | DIP48 |
| DIP2 | 2 | 47 | DIP47 |
| DIP3 | 3 | 46 | DIP46 |
| DIP4 | 4 | 45 | DIP45 |
| DIP5 | 5 | 44 | DIP44 |
| DIP6 | 6 | 43 | DIP43 |
| DIP7 | 7 | 42 | DIP42 |
| DIP8 | 8 | 41 | DIP41 |
| DIP9 | 9 | 40 | DIP40 |
| DIP10 | 10 | 39 | DIP39 |
| DIP11 | 11 | 38 | DIP38 |
| DIP12 | 12 | 37 | DIP38 |
| DIP13 | 13 | 36 | DIP36 |
| DIP14 | 14 | 35 | DIP35 |
| DIP15 | 15 | 34 | DIP34 |
| DIP16 | 16 | 33 | DIP33 |
| DIP17 | 17 | 32 | DIP32 |
| DIP18 | 18 | 31 | DIP31 |
| DIP19 | 19 | 30 | DIP30 |
| DIP20 | 20 | 29 | DIP29 |
| DIP21 | 21 | 28 | DIP28 |
| DIP22 | 22 | 27 | DIP27 |
| DIP23 | 23 | 26 | DIP26 |
| DIP24 | 24 | 25 | DIP25 |

JP3 for MP transition board

14.2 PROGRAMMING PIN MAPPING:

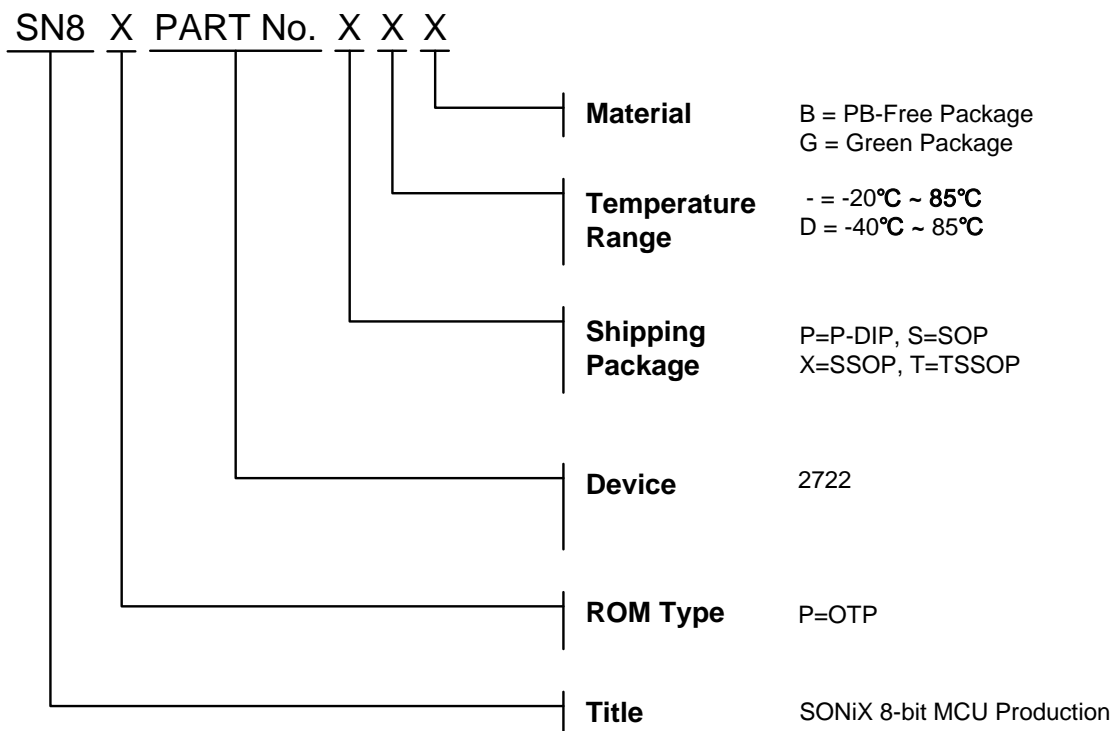
| Programming Information of SN8P2722 Series | | | | | | | | | |
|--|----------|----------------------------|------|--------|-----|--|--|--|--|
| Chip Name | | SN8P2722P,S,X,T | | | | | | | |
| EZ Writer Connector | | OTP IC / JP3 Pin Assigment | | | | | | | |
| Number | Name | Number | Pin | Number | Pin | | | | |
| 1 | VDD | 20 | VDD | | | | | | |
| 2 | GND | 1 | VSS | | | | | | |
| 3 | CLK | 8 | P0.7 | | | | | | |
| 4 | CE | | - | | | | | | |
| 5 | PGM | 7 | P0.6 | | | | | | |
| 6 | OE | 6 | P0.5 | | | | | | |
| 7 | D1 | | - | | | | | | |
| 8 | D0 | | - | | | | | | |
| 9 | D3 | | - | | | | | | |
| 10 | D2 | | - | | | | | | |
| 11 | D5 | | - | | | | | | |
| 12 | D4 | | - | | | | | | |
| 13 | D7 | | - | | | | | | |
| 14 | D6 | | - | | | | | | |
| 15 | VDD | | - | | | | | | |
| 16 | VPP | 4 | RST | | | | | | |
| 17 | HLS | | - | | | | | | |
| 18 | RST | | - | | | | | | |
| 19 | - | | - | | | | | | |
| 20 | ALSB/PDB | 3 | P0.2 | | | | | | |

15 Marking Definition

15.1 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.

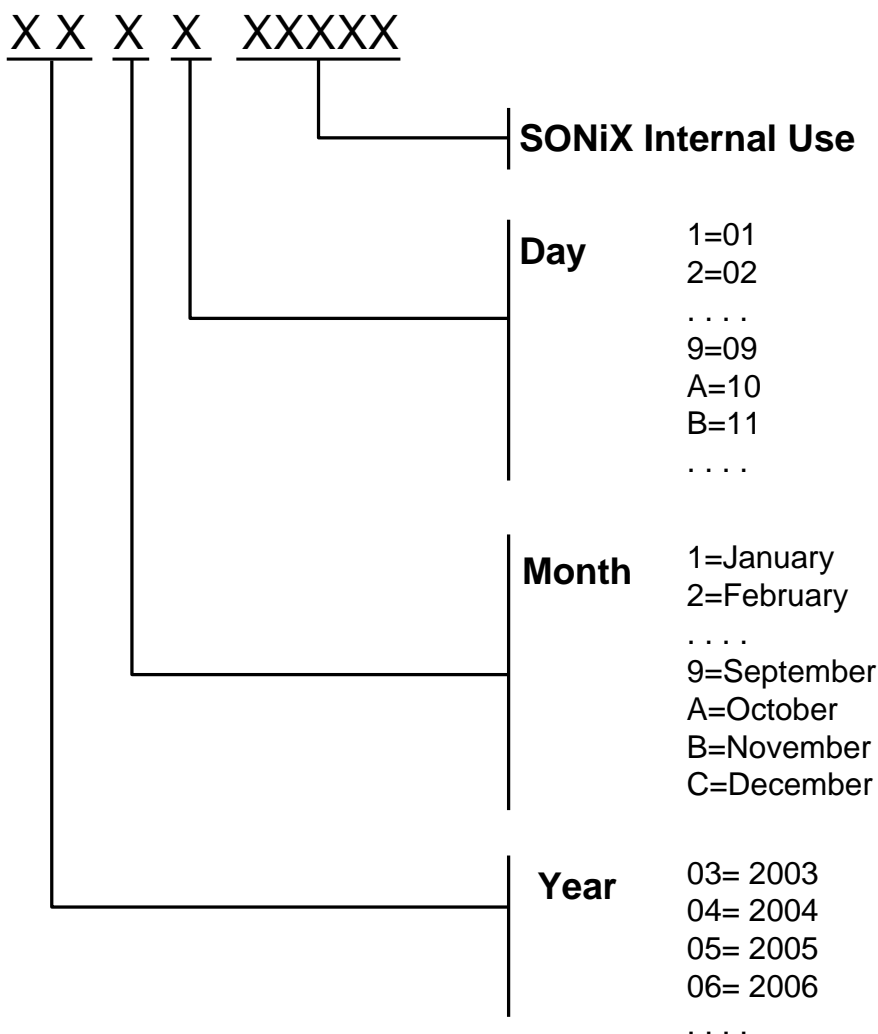
15.2 MARKING INDETFICATION SYSTEM



15.3 MARKING EXAMPLE

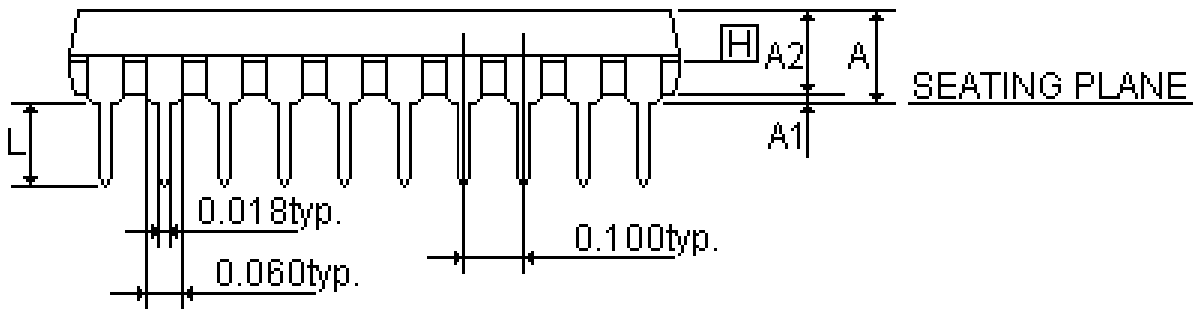
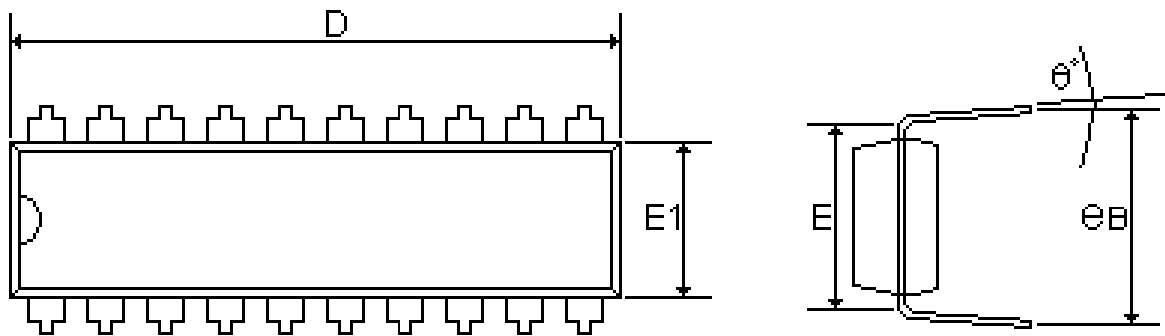
| Name | ROM Type | Device | Package | Temperature | Material |
|-------------|----------|--------|---------|-------------|-----------------|
| SN8P2722PB | OTP | 2722 | P-DIP | -20°C~85°C | PB-Free Package |
| SN8P2722SB | OTP | 2722 | SOP | -20°C~85°C | PB-Free Package |
| SN8P2722XB | OTP | 2722 | SSOP | -20°C~85°C | PB-Free Package |
| SN8P2722TB | OTP | 2722 | TSSOP | -20°C~85°C | PB-Free Package |
| SN8P2722PG | OTP | 2722 | P-DIP | -20°C~85°C | Green Package |
| SN8P2722SG | OTP | 2722 | SOP | -20°C~85°C | Green Package |
| SN8P2722XG | OTP | 2722 | SSOP | -20°C~85°C | Green Package |
| SN8P2722TG | OTP | 2722 | TSSOP | -20°C~85°C | Green Package |
| SN8P2722PDB | OTP | 2722 | P-DIP | -40°C~85°C | PB-Free Package |
| SN8P2722SDB | OTP | 2722 | SOP | -40°C~85°C | PB-Free Package |
| SN8P2722XDB | OTP | 2722 | SSOP | -40°C~85°C | PB-Free Package |
| SN8P2722TDB | OTP | 2722 | TSSOP | -40°C~85°C | PB-Free Package |
| SN8P2722PDG | OTP | 2722 | P-DIP | -40°C~85°C | Green Package |
| SN8P2722SDG | OTP | 2722 | SOP | -40°C~85°C | Green Package |
| SN8P2722XDG | OTP | 2722 | SSOP | -40°C~85°C | Green Package |
| SN8P2722TDG | OTP | 2722 | TSSOP | -40°C~85°C | Green Package |

15.4 DATECODE SYSTEM



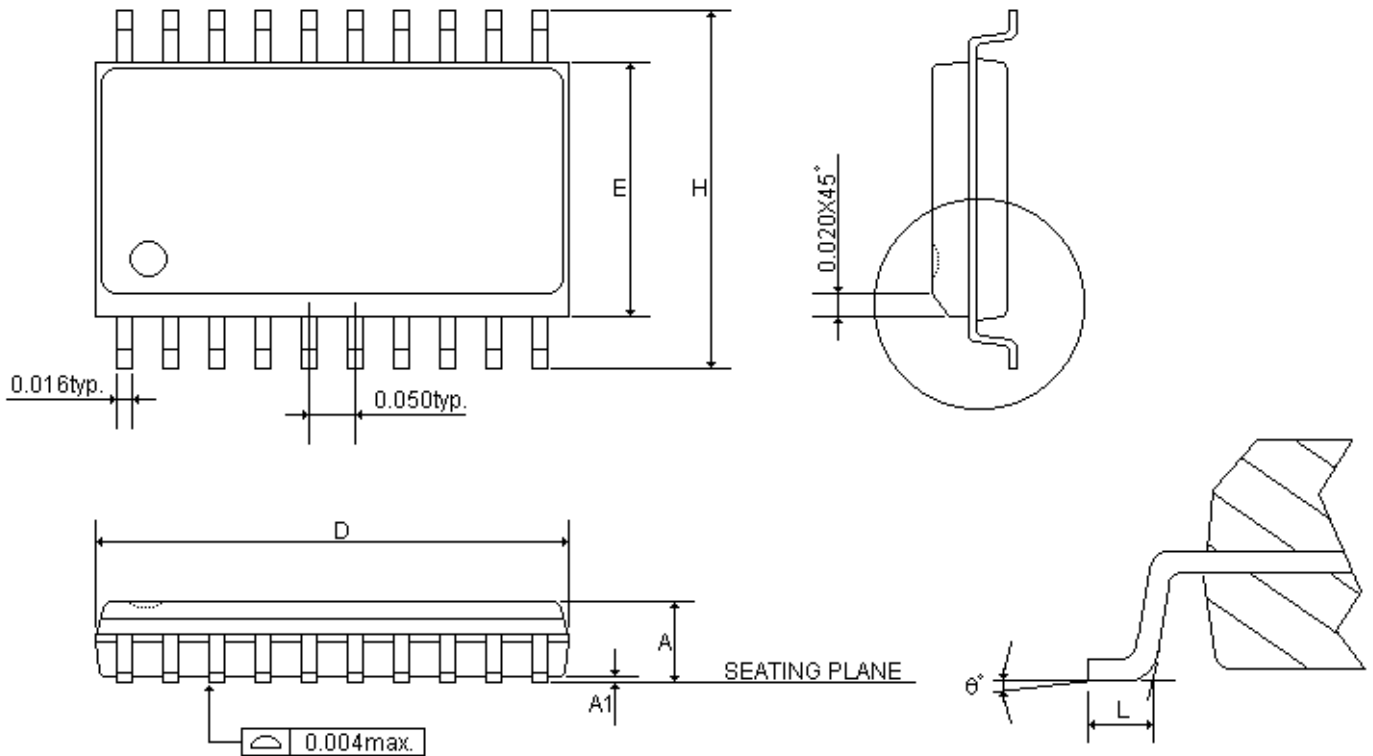
16 PACKAGE INFORMATION

16.1 P-DIP 20 PIN



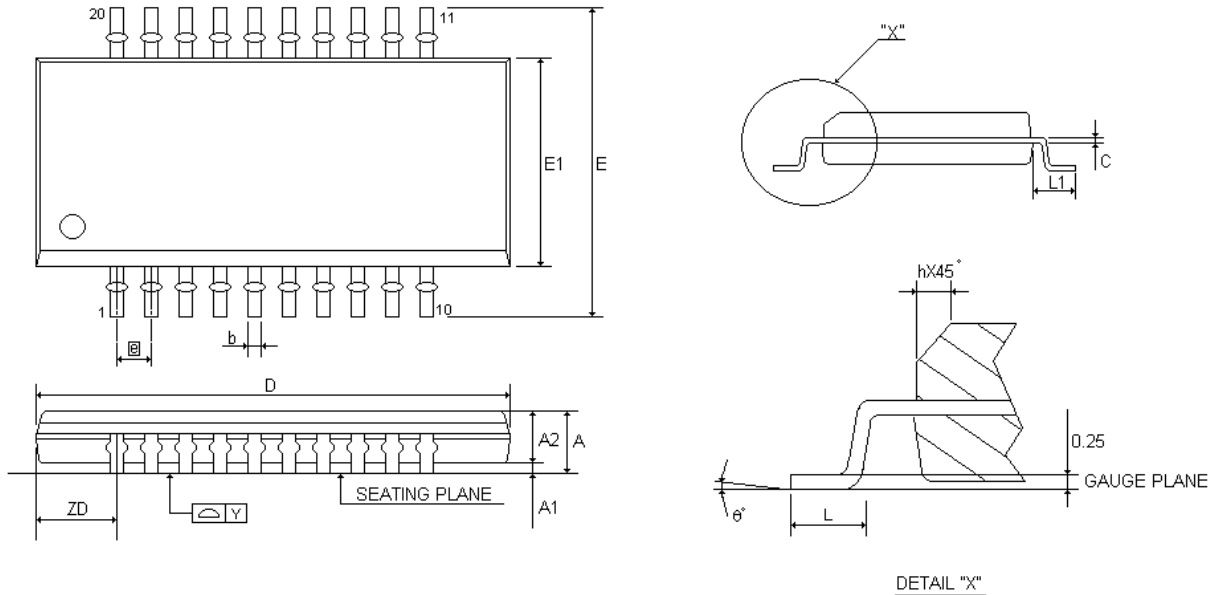
| SYMBOLS | MIN | NOR | MAX | MIN | NOR | MAX |
|---------|--------|-------|-------|--------|--------|--------|
| | (inch) | | | (mm) | | |
| A | - | - | 0.210 | - | - | 5.334 |
| A1 | 0.015 | - | - | 0.381 | - | - |
| A2 | 0.125 | 0.130 | 0.135 | 3.175 | 3.302 | 3.429 |
| D | 0.980 | 1.030 | 1.060 | 24.892 | 26.162 | 26.924 |
| E | 0.300 | | | 7.620 | | |
| E1 | 0.245 | 0.250 | 0.255 | 6.223 | 6.350 | 6.477 |
| L | 0.115 | 0.130 | 0.150 | 2.921 | 3.302 | 3.810 |
| eB | 0.335 | 0.355 | 0.375 | 8.509 | 9.017 | 9.525 |
| θ° | 0° | 7° | 15° | 0° | 7° | 15° |

16.2 SOP 20 PIN



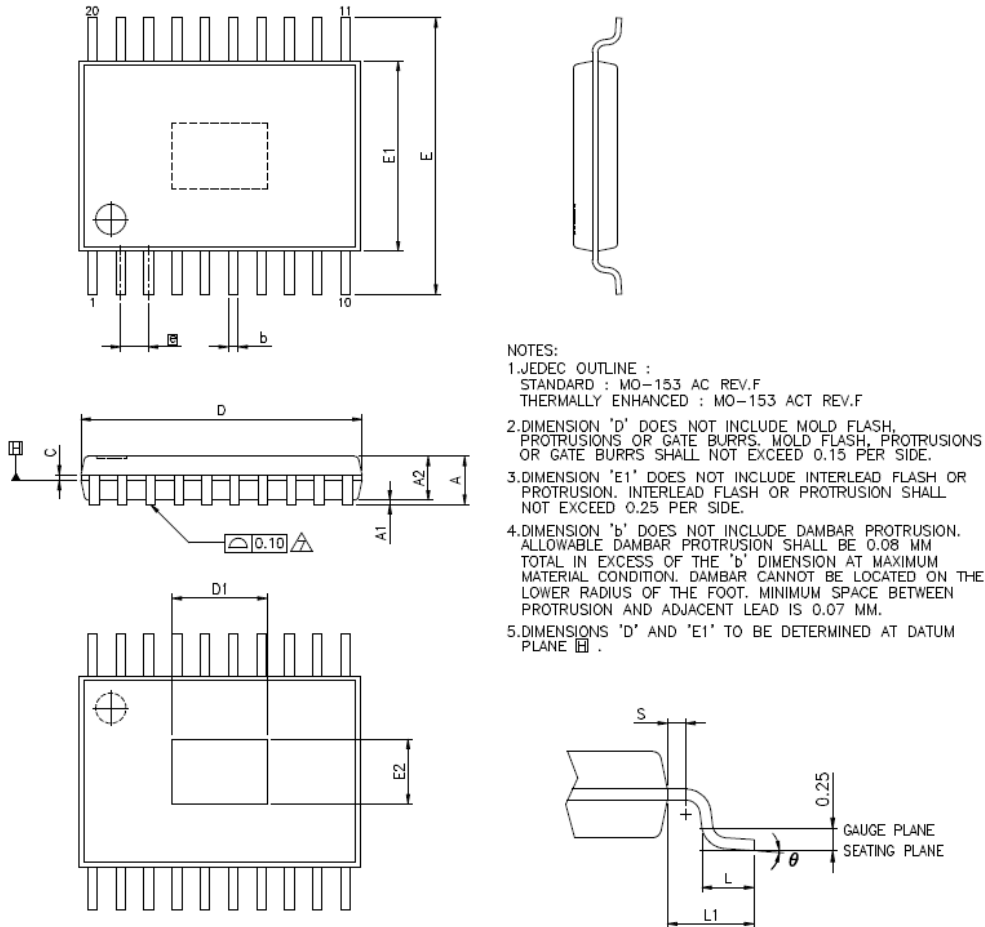
| SYMBOLS | MIN | NOR | MAX | MIN | NOR | MAX |
|----------------|--------|-------|-------|--------|--------|--------|
| | (inch) | | | (mm) | | |
| A | 0.093 | 0.099 | 0.104 | 2.362 | 2.502 | 2.642 |
| A1 | 0.004 | 0.008 | 0.012 | 0.102 | 0.203 | 0.305 |
| D | 0.496 | 0.502 | 0.508 | 12.598 | 12.751 | 12.903 |
| E | 0.291 | 0.295 | 0.299 | 7.391 | 7.493 | 7.595 |
| H | 0.394 | 0.407 | 0.419 | 10.008 | 10.325 | 10.643 |
| L | 0.016 | 0.033 | 0.050 | 0.406 | 0.838 | 1.270 |
| θ° | 0° | 4° | 8° | 0° | 4° | 8° |

16.3 SSOP 20 PIN



| SYMBOLS | MIN | NOR | MAX | MIN | NOR | MAX |
|---------|--------|-------|-------|-------|-------|-------|
| | (inch) | | | (mm) | | |
| A | 0.053 | 0.063 | 0.069 | 1.350 | 1.600 | 1.750 |
| A1 | 0.004 | 0.006 | 0.010 | 0.100 | 0.150 | 0.250 |
| A2 | - | - | 0.059 | - | - | 1.500 |
| b | 0.008 | 0.010 | 0.012 | 0.200 | 0.254 | 0.300 |
| c | 0.007 | 0.008 | 0.010 | 0.180 | 0.203 | 0.250 |
| D | 0.337 | 0.341 | 0.344 | 8.560 | 8.660 | 8.740 |
| E | 0.228 | 0.236 | 0.244 | 5.800 | 6.000 | 6.200 |
| E1 | 0.150 | 0.154 | 0.157 | 3.800 | 3.900 | 4.000 |
| [e] | 0.025 | | | 0.635 | | |
| h | 0.010 | 0.017 | 0.020 | 0.250 | 0.420 | 0.500 |
| L | 0.016 | 0.025 | 0.050 | 0.400 | 0.635 | 1.270 |
| L1 | 0.039 | 0.041 | 0.043 | 1.000 | 1.050 | 1.100 |
| ZD | 0.059 | | | 1.500 | | |
| Y | - | - | 0.004 | - | - | 0.100 |
| θ° | 0° | - | 8° | 0° | - | 8° |

16.4 TSSOP 20 PIN



| SYMBOLS | MIN | NOR | MAX | MIN | NOR | MAX |
|----------------|-----------|-------|-------|----------|------|------|
| | (inch) | | | (mm) | | |
| A | - | - | 0.047 | - | - | 1.2 |
| A1 | 0.002 | - | 0.006 | 0.05 | - | 0.15 |
| A2 | 0.031 | 0.035 | 0.041 | 0.80 | 0.90 | 1.05 |
| b | 0.007 | - | 0.012 | 0.19 | - | 0.30 |
| C | 0.004 | - | 0.008 | 0.09 | - | 0.20 |
| D | 0.252 | 0.256 | 0.260 | 6.40 | 6.50 | 6.60 |
| E | 0.252 BSC | | | 6.40 BSC | | |
| E1 | 0.169 | 0.173 | 0.177 | 4.30 | 4.40 | 4.50 |
| [e] | 0.026 BSC | | | 0.65 BSC | | |
| L | 0.020 | 0.024 | 0.030 | 0.50 | 0.60 | 0.75 |
| L1 | 0.039 REF | | | 1.00 REF | | |
| S | 0.008 | - | - | 0.2 | - | - |
| θ° | 0° | - | 8° | 0° | - | 8° |

| PAD SIZE | E2 | | | D1 | | |
|----------|------|------|------|------|------|------|
| | MIN | NOR | MAX | MIN | NOR | MAX |
| 118X16E | 2.60 | 2.80 | 3.00 | 3.79 | 3.99 | 4.19 |

SONIX reserves the right to make change without further notice to any products herein to improve reliability, function or design. SONIX does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. SONIX products are not designed, intended, or authorized for use as components in systems intended, for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SONIX product could create a situation where personal injury or death may occur. Should Buyer purchase or use SONIX products for any such unintended or unauthorized application. Buyer shall indemnify and hold SONIX and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that SONIX was negligent regarding the design or manufacture of the part.

Main Office:

Address: 10F-1, NO. 36, Taiyuan Stree., Chupei City, Hsinchu, Taiwan R.O.C.

Tel: 886-3-5600 888

Fax: 886-3-5600 889

Taipei Office:

Address: 15F-2, NO. 171, Song Ted Road, Taipei, Taiwan R.O.C.

Tel: 886-2-2759 1980

Fax: 886-2-2759 8180

Hong Kong Office:

Unit 1519, Chevalier Commercial Center, No.8 Wang Hoi Road, Kowloon Bay, Kowloon. Hong Kong.

Tel: 852-2723-8086

Fax: 852-2723-9179

Technical Support by Email:

Sn8fae@sonix.com.tw

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for [32-bit Microcontrollers - MCU category](#):

Click to view products by [SONIX manufacturer](#):

Other Similar products are found below :

[MCF51AC256AVFUE](#) [MCF51AC256BCFUE](#) [MCF51AC256BVFUE](#) [MB91F464AAPMC-GSE2](#) [R5S726B0D216FP#V0](#) [MB91F248PFV-GE1](#) [MB91243PFV-GS-136E1](#) [SAK-TC1782F-320F180HR BA](#) [TC364DP64F300WAAKXUMA1](#) [R5F566NNDDFP#30](#)
[R5F566NNDDFC#30](#) [R5F566NNDDBD#20](#) [MC96F8216ADBN](#) [A96G181HDN](#) [A96G140KNN](#) [A96G174FDN](#) [A31G213CL2N](#)
[A96G148KNN](#) [A96G174AEN](#) [AC33M3064TLBN-01](#) [V3s](#) [T3](#) [A40i-H](#) [V526](#) [A83T](#) [R11](#) [V851s](#) [A133](#) [V833](#) [F1C100S](#) [T3L](#) [T507](#) [A33](#)
[A63](#) [T113-i](#) [H616](#) [V853](#) [V533](#) [V536-H](#) [A64-H](#) [V831](#) [V3LP](#) [T113-S3](#) [F1C200S](#) [F133-A](#) [R128-S2](#) [ADUCM360BCPZ128-TR](#)
[APT32S003F8PT](#) [AT32F435VMT7](#) [AT32F435CGT7](#)