

# **SN8P26L00 Series**

## **USER'S MANUAL**

Version 1.1

SN8P26L34

SN8P26L32

SN8P26L321

SN8P26L31

## **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.

## AMENDMENT HISTORY

Version	Date	Description
VER 0.1	Dec. 2006	First Issue.
VER 0.2	Mar. 2007	1. Add SN8P26L321 pin assignment. 2. Add IR section.
VER 0.3	Mar. 2007	Modify SN8P26L32 pin assignment. Comparator 1 pins are replaced by Comparator 0. Don't modify PCB layout. Only modify the firmware comparator definition from comparator 1 to comparator 0.
VER 0.4	Jun. 2007	1. Modify comparator interrupt request flag description. 2. Add development tool section. 3. Modify electrical characteristic table. 4. Add development tool section.
VER 0.5	Oct. 2007	Modify Programming Pin Mapping.
VER 0.6	Nov. 2007	1. Add IR section. 2. Modify system operating mode section.
VER 0.7	Dec. 2007	Add SN8P26L34/32/321 EV-kit circuit.
VER 0.8	Feb. 2008	Modify internal low RC frequency to 10KHz.
VER 0.9	Otc. 2008	1. Add SN8P26L34 Green Package Marking Example. 2. Add SN8P26L31 pin assignment.
VER 1.0	Aug. 2009	1. Modify Bank 0 RAM address range from 0x00~0x2F to 0x00~0x7F. 2. Modify P.97 comparator pin assignment description.
VER 1.1	Oct. 2009	Add SN8P26L321J TQFN 20L package type.

# Table of Content

AMENDMENT HISTORY .....	2
1 PRODUCT OVERVIEW .....	8
1.1 FEATURES.....	8
1.2 SYSTEM BLOCK DIAGRAM.....	9
1.3 PIN ASSIGNMENT .....	10
1.4 PIN DESCRIPTIONS.....	12
1.5 PIN CIRCUIT DIAGRAMS .....	13
2 CENTRAL PROCESSOR UNIT (CPU).....	15
2.1 PROGRAM MEMORY (ROM).....	15
2.1.1 RESET VECTOR (0000H).....	16
2.1.2 INTERRUPT VECTOR (0008H).....	17
2.1.3 LOOK-UP TABLE DESCRIPTION.....	19
2.1.4 JUMP TABLE DESCRIPTION.....	21
2.1.5 CHECKSUM CALCULATION .....	23
2.2 DATA MEMORY (RAM) .....	24
2.2.1 SYSTEM REGISTER.....	25
2.2.1.1 SYSTEM REGISTER TABLE .....	25
2.2.1.2 SYSTEM REGISTER DESCRIPTION.....	25
2.2.1.3 BIT DEFINITION of SYSTEM REGISTER.....	26
2.2.2 ACCUMULATOR .....	28
2.2.3 PROGRAM FLAG .....	29
2.2.4 PROGRAM COUNTER.....	30
2.2.5 H, L REGISTERS.....	33
2.2.6 Y, Z REGISTERS.....	34
2.2.7 R REGISTERS.....	35
2.3 ADDRESSING MODE.....	36
2.3.1 IMMEDIATE ADDRESSING MODE .....	36
2.3.2 DIRECTLY ADDRESSING MODE .....	36
2.3.3 INDIRECTLY ADDRESSING MODE .....	36
2.4 STACK OPERATION.....	37
2.4.1 OVERVIEW.....	37
2.4.2 STACK REGISTERS .....	38
2.4.3 STACK OPERATION EXAMPLE.....	39
2.5 CODE OPTION TABLE .....	40
2.5.1 RESET_PIN CODE OPTION.....	40
2.5.2 SECURITY CODE OPTION.....	40

---

---

3 RESET .....	41
3.1 OVERVIEW .....	41
3.2 POWER ON RESET .....	42
3.3 WATCHDOG RESET .....	42
3.4 BROWN OUT RESET .....	43
3.4.1 THE SYSTEM OPERATING VOLTAGE.....	44
3.4.2 LOW VOLTAGE DETECTOR (LVD).....	44
3.4.3 BROWN OUT RESET IMPROVEMENT .....	46
3.5 EXTERNAL RESET .....	47
3.6 EXTERNAL RESET CIRCUIT .....	47
3.6.1 Simply RC Reset Circuit.....	47
3.6.2 Diode & RC Reset Circuit .....	48
3.6.3 Zener Diode Reset Circuit.....	48
3.6.4 Voltage Bias Reset Circuit.....	49
3.6.5 External Reset IC.....	50
4 SYSTEM CLOCK .....	51
4.1 OVERVIEW .....	51
4.2 CLOCK BLOCK DIAGRAM.....	51
4.3 F <sub>CPU</sub> (INSTRUCTION CYCLE).....	51
4.4 OSCM REGISTER.....	52
4.5 SYSTEM HIGH CLOCK.....	53
4.5.1 INTERNAL HIGH RC .....	53
4.5.2 EXTERNAL HIGH CLOCK .....	53
4.5.2.1 CRYSTAL/CERAMIC .....	54
4.5.2.2 RC .....	54
4.5.2.3 EXTERNAL CLOCK SIGNAL.....	55
4.6 SYSTEM LOW CLOCK .....	56
4.6.1 SYSTEM CLOCK MEASUREMENT.....	57
5 SYSTEM OPERATION MODE .....	58
5.1 OVERVIEW .....	58
5.2 NORMAL MODE .....	59
5.3 SLOW MODE.....	59
5.4 POWER DOWN MDOE.....	59
5.5 GREEN MODE.....	60
5.6 OPERATING MODE CONTROL MACRO .....	60
5.7 WAKEUP.....	62
5.7.1 OVERVIEW.....	62
5.7.2 WAKEUP TIME .....	62

---

---

5.7.3 P1W WAKEUP CONTROL REGISTER .....	63
6 INTERRUPT .....	64
6.1 OVERVIEW .....	64
6.2 INTEN INTERRUPT ENABLE REGISTER .....	65
6.3 INTRQ INTERRUPT REQUEST REGISTER .....	66
6.4 GIE GLOBAL INTERRUPT OPERATION .....	66
6.5 PUSH, POP ROUTINE .....	67
6.6 EXTERNAL INTERRUPT OPERATION (INT0) .....	68
6.7 INT1 (P0.1) INTERRUPT OPERATION .....	69
6.8 T0 INTERRUPT OPERATION .....	70
6.9 TC1 INTERRUPT OPERATION .....	72
6.10 MULTI-INTERRUPT OPERATION .....	73
7 I/O PORT .....	74
7.1 OVERVIEW .....	74
7.2 I/O PORT MODE .....	75
7.3 I/O PULL UP REGISTER .....	76
7.4 I/O OPEN-DRAIN REGISTER .....	77
7.5 I/O PORT DATA REGISTER .....	78
8 TIMERS .....	79
8.1 WATCHDOG TIMER .....	79
8.2 TIMER 0 (T0) .....	81
8.2.1 OVERVIEW .....	81
8.2.2 T0M MODE REGISTER .....	82
8.2.3 T0C COUNTING REGISTER .....	83
8.2.4 T0 TIMER OPERATION SEQUENCE .....	84
8.3 TIMER/COUNTER 0 (TC1) .....	85
8.3.1 OVERVIEW .....	85
8.3.2 TC1M MODE REGISTER .....	86
8.3.3 TC1C COUNTING REGISTER .....	87
8.3.4 TC1R AUTO-LOAD REGISTER .....	88
8.3.5 TC1 CLOCK FREQUENCY OUTPUT (BUZZER) .....	89
8.3.6 TC1 TIMER OPERATION SEQUENCE .....	90
8.4 PWM1 MODE .....	91
8.4.1 OVERVIEW .....	91
8.4.2 TC1IRQ AND PWM DUTY .....	92
8.4.3 PWM PROGRAM EXAMPLE .....	93
8.4.4 PWM1 DUTY CHANGING NOTICE .....	94

---

9 ANALOG COMPARAOTR.....	96
9.1 OVERVIEW .....	96
9.2 CMP0M REGISTER .....	98
9.3 CMP1M REGISTER .....	99
9.4 ANALOG COMPARATOR APPLICATION .....	100
10 IR OUTPUT.....	102
10.1 OVERVIEW.....	102
10.2 IR CONTROL REGISTER .....	103
10.2.1 TC0M MODE REGISTER .....	103
10.2.2 TC0C COUNTING REGISTER .....	103
10.2.3 TC0R AUTO-LOAD REGISTER.....	104
10.2.4 TC0D IR DUTY CONTROL REGISTER.....	105
10.2.5 IR OUTPUT OPERATION SEQUENCE .....	106
11 INSTRUCTION TABLE.....	107
12 ELECTRICAL CHARACTERISTIC .....	108
12.1 ABSOLUTE MAXIMUM RATING .....	108
12.2 ELECTRICAL CHARACTERISTIC.....	108
13 DEVELOPMENT TOOL.....	109
13.1 SN8P26L34/32/321 EV-KIT .....	109
13.2 ICE AND EV-KIT APPLICATION NOTIC .....	110
14 OTP PROGRAMMING PIN .....	111
14.1 THE PIN ASSIGNMENT OF EASY WRITER TRANSITION BOARD SOCKET: .....	111
14.2 PROGRAMMING PIN MAPPING: .....	112
15 MARKING DEFINITION .....	113
15.1 INTRODUCTION.....	113
15.2 MARKING INDETIFICATION SYSTEM .....	113
15.3 MARKING EXAMPLE.....	114
15.4 DATECODE SYSTEM .....	114
16 PACKAGE INFORMATION.....	115
16.1 SK-DIP 28 PIN .....	115
16.2 SOP 28 PIN .....	116
16.3 SSOP 28 PIN .....	117
16.4 P-DIP 18 PIN.....	118
16.5 SOP 18 PIN .....	119
16.6 P-DIP 20 PIN.....	120

---

16.7 SOP 20 PIN .....	121
16.8 SSOP 20 PIN .....	122
16.9 TQFN 20 PIN .....	123
16.10 SOP 14 PIN.....	124

# 1 PRODUCT OVERVIEW

## 1.1 FEATURES

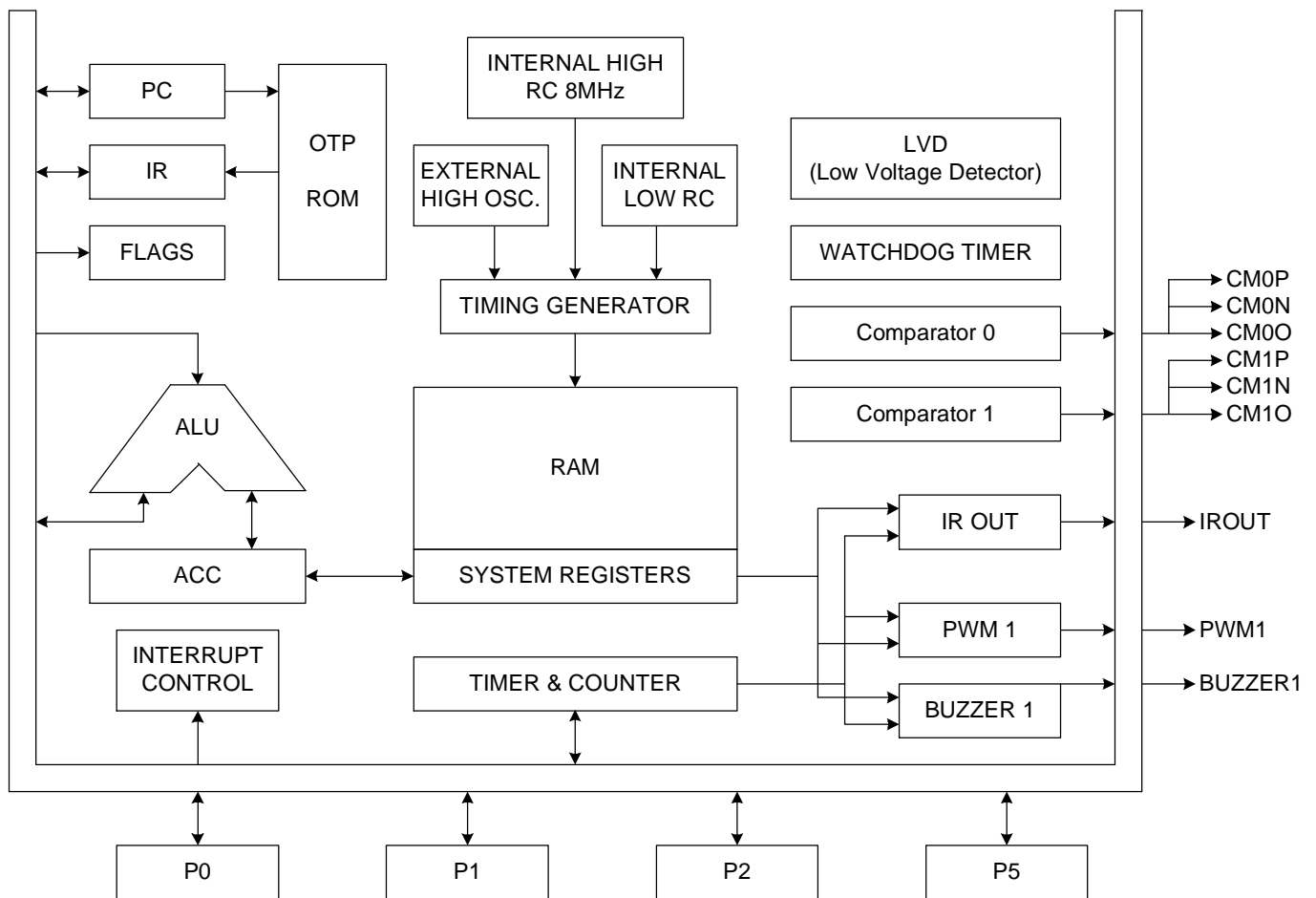
- ◆ **Memory configuration**  
OTP ROM size: 8K \* 16 bits.  
RAM size: 240 \* 8 bits.
- ◆ **8 levels stack buffer**
- ◆ **I/O pin configuration**  
Bi-directional: P0, P1, P2, P5  
Programmable open-drain: P1.0, P1.1  
Wakeup: P0, P1 level change trigger.  
P1 wake-up function controlled by P1W.  
Pull-up resistors: P0, P1, P2, P5  
External interrupt input: P0.0, P0.1  
External Interrupt trigger edge:  
P0.0 controlled by PEDGE register
- ◆ **3-Level LVD.**  
Reset system and power monitor.
- ◆ **2-ch analog comparators with internal selectable reference voltage 0.9V/1.0V/1.1V/1.2V and external reference input.**
- ◆ **Six interrupt sources**  
Four internal interrupts: T0, TC1, CM0, CM1  
One external interrupts: INT0 INT1
- ◆ **Powerful instructions**  
**One clock per instruction cycle (1T)**  
All ROM area JMP instruction.  
All ROM area CALL address instruction.  
All ROM area lookup table function (MOVC)
- ◆ **Two 8-bit Timer/Counter**  
T0: Basic timer.  
TC1: Auto-reload timer/counter.
- ◆ **One RTC timer (T0).**
- ◆ **One channels PWM output.**
- ◆ **One channels buzzer output.**
- ◆ **One channel IR output (duty/cycle programmable PWM, TC0).**
- ◆ **On chip watchdog timer and clock source is internal low clock RC type (about 10KHz @3V).**
- ◆ **Four system clocks**  
External high clock: RC type up to 8 MHz  
External high clock: Crystal type up to 8 MHz  
Internal high clock: RC type 8MHz.  
Internal low clock: RC type 10KHz(3V).
- ◆ **Four operating modes**  
Normal mode: Both high and low clock active  
Slow mode: Low clock only  
Sleep mode: Both high and low clock stop  
Green mode: Periodical wakeup by timer
- ◆ **Package (Chip form support)**  
SK-DIP 28 pins  
SOP 28 pins  
SSOP 28 pins  
P-DIP 18 pins  
SOP 18 pins  
SSOP 20 pins  
DIP 20 pins  
SOP20 pins  
TQFN 20 pins

**F Features Selection Table**

CHIP	ROM (word)	RAM (byte)	Stack	Timer		RTC	IR Out	Int. High Osc.	I/O	Comp arator	PWM	Wakeup Pin No.	Package
				T0	TC1						Buzzer		
SN8P26L31	8K	240	8	V	V	V	-	8M	12	1-ch	V	5	SOP14
SN8P26L32	8K	240	8	V	V	V	V	8M	16	1-ch	V	8	DIP18/ SOP18
SN8P26L321	8K	240	8	V	V	V	V	8M	18	2-ch	V	8	DIP20/ SOP20/ SSOP20/ TQFN20
SN8P26L34	8K	240	8	V	V	V	V	8M	26	2-ch	V	13	SKDIP28/ SOP28/ SSOP28



## 1.2 SYSTEM BLOCK DIAGRAM



## 1.3 PIN ASSIGNMENT

**SN8P26L34K (SK-DIP 28 pins)**

**SN8P26L34S (SOP 28 pins)**

**SN8P26L34X (SSOP 28 pins)**

P0.1/INT1	1	U	28	P2.7/CM1O
VDD	2		27	P2.6/CM1P
P5.4/IROUT	3		26	P2.5/CM1N
VSS	4		25	P2.4/CM0O
P0.0/INT0	5		24	P2.3/CM0P
P5.0	6		23	P2.2/CM0N
P5.1	7		22	XIN/P0.3
P5.2	8		21	XOUT/P0.4
P5.3/BZ1/PWM1	9		20	P2.1
P0.2/RST/VPP	10		19	P2.0
P1.0	11		18	P1.7
P1.1	12		17	P1.6
P1.2	13		16	P1.5
P1.3	14		15	P1.4

SN8P26L34K

SN8P26L34S

SN8P26L34X

**SN8P26L32P (P-DIP 18 pins)**

**SN8P26L32S (SOP 18 pins)**

P0.1/INT1	1	U	18	P2.4/CM0O
VDD	2		17	P2.3/CM0P
P5.4/IROUT	3		16	P2.2/CM0N
VSS	4		15	XIN/P0.3
P5.0	5		14	XOUT/P0.4
P5.1	6		13	P1.3
P5.2	7		12	P1.2
P5.3/BZ1/PWM1	8		11	P1.1
P0.2/RST/VPP	9		10	P1.0

SN8P26L32P

SN8P26L32S

**SN8P26L321P (P-DIP 20 pins)**

**SN8P26L321S (SOP 20 pins)**

**SN8P26L321X (SSOP 20 pins)**

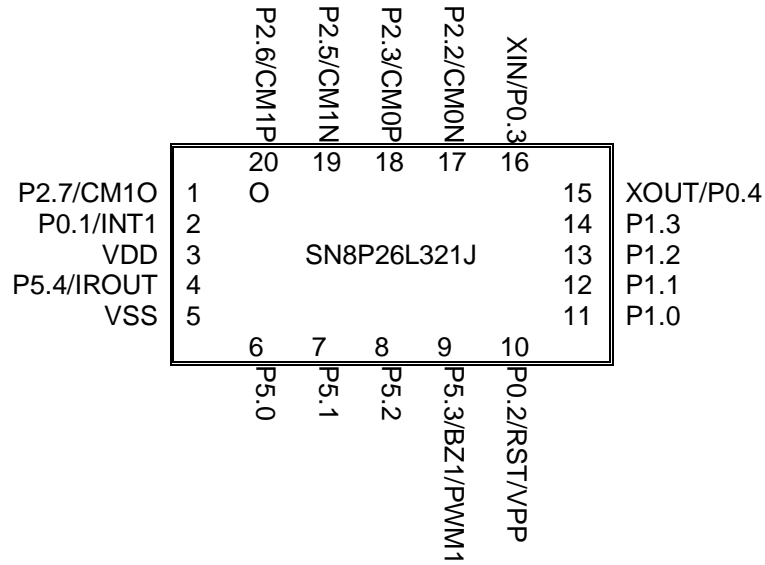
P2.7/CM1O	1	U	20	P2.6/CM1P
P0.1/INT1	2		19	P2.5/CM1N
VDD	3		18	P2.3/CM0P
P5.4/IROUT	4		17	P2.2/CM0N
VSS	5		16	XIN/P0.3
P5.0	6		15	XOUT/P0.4
P5.1	7		14	P1.3
P5.2	8		13	P1.2
P5.3/BZ1/PWM1	9		12	P1.1
P0.2/RST/VPP	10		11	P1.0

SN8P26L321P

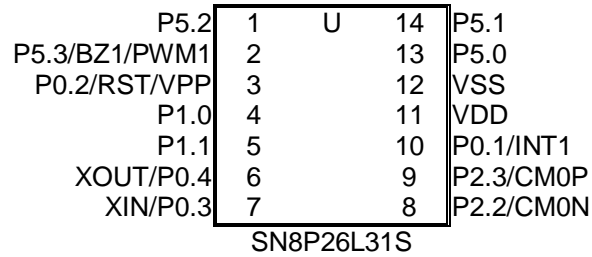
SN8P26L321S

SN8P26L321X

**SN8P26L321J (TQFN 20 pins)**



**SN8P26L31S (SOP 14 pins)**



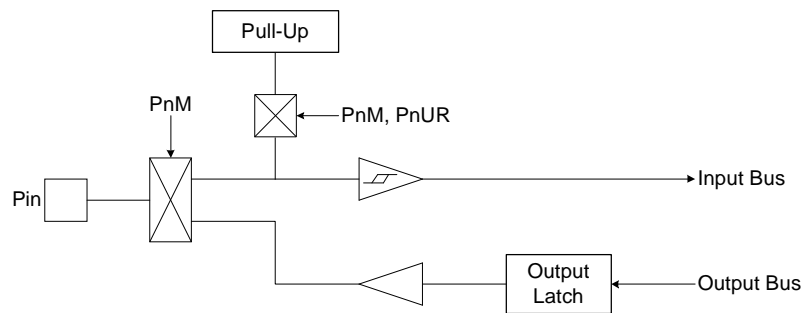
## 1.4 PIN DESCRIPTIONS

PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital circuit.
P0.0/INT0	I/O	P0.0: Port 0.0 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function. INT0: External interrupt 0 input pin.
P0.1/INT1	I/O	P0.1: Port 0.1 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function. INT1: External interrupt 1 input pin.
P0.2/RST/VPP	I, P	RST is system external reset input pin under Ext_RST mode. P0.2 is input only pin without pull-up resistor under P0.2 mode. Schmitt trigger structure, active “low”, normal stay to “high”. Built wakeup function. OTP 12.3V power input pin in programming mode.
P0.3/XIN	I/O	XIN: Oscillator input pin while external oscillator enable (crystal and RC). P0.3: Port 0.3 bi-direction pin under internal 16M RC. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function.
P0.4/XOUT	I/O	XOUT: Oscillator output pin while external crystal enable. P0.4: Port 0.4 bi-direction pin under internal 16M RC and external RC. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function.
P1.0	I/O	P1.0: Port 1.0 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Open-Drain function controlled by “P1OC” register. Built wakeup function.
P1.1	I/O	P1.1: Port 1.1 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Open-Drain function controlled by “P1OC” register. Built wakeup function.
P1[7:2]	I/O	P1: Port 1 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function.
P2[1:0]	I/O	P2: Port 2 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode.
P2.2/CM0N	I/O	P2.2: Port 2.2 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. CM0N: Analog comparator 0 negative input pin.
P2.3/CM0P	I/O	P2.3: Port 2.3 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. CM0P: Analog comparator 0 positive input pin.
P2.4/CM0O	I/O	P2.4: Port 2.4 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. CM0O: Analog comparator 0 output pin. When the CM0N input level is less than the level of CM0P, the CM0O output Vdd. When the CM0N input level is higher than the level of CM0P, the CM0O output Vss.
P2.5/CM1N	I/O	P2.5: Port 2.5 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. CM1N: Analog comparator 1 negative input pin.
P2.6/CM1P	I/O	P2.6: Port 2.6 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. CM1P: Analog comparator 1 positive input pin.
P2.7/CM1O	I/O	P2.7: Port 2.7 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. CM1O: Analog comparator 1 output pin.

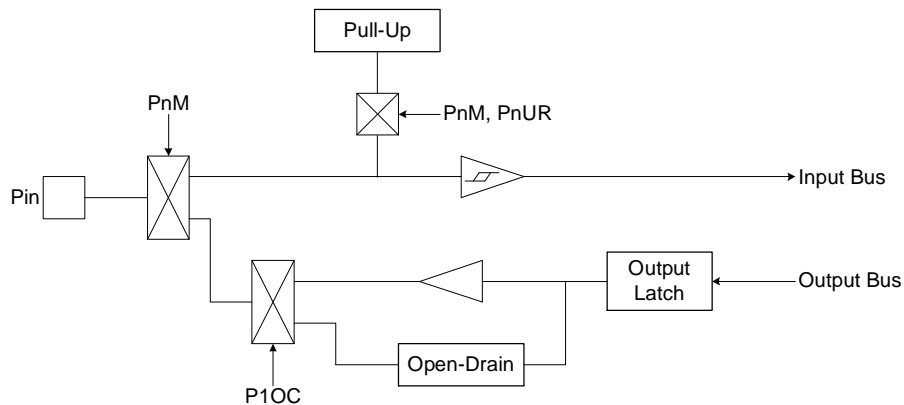
		When the CM1N input level is less than the level of CM1P, the CM1O output Vdd. When the CM1N input level is higher than the level of CM1P, the CM1O output Vss.
P5[2:0]	I/O	P5: Port 5 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode.
P5.3/BZ1/PWM1	I/O	P5.3: Port 5.3 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. BZ1: 1/2 TC1 counter output pin. PWM1: PWM1 output.
P5.4/IROUT	I/O	P5.4: Port 5.4 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. IROUT: IR output pin.

## 1.5 PIN CIRCUIT DIAGRAMS

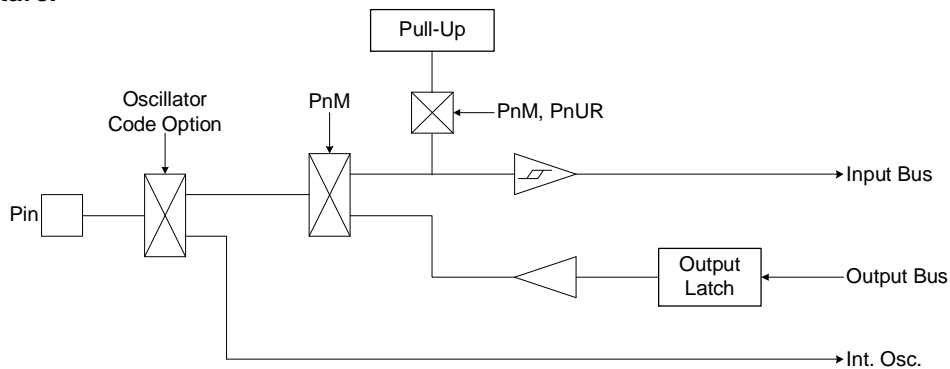
**Port 0, 1, 2, 5 structure:**



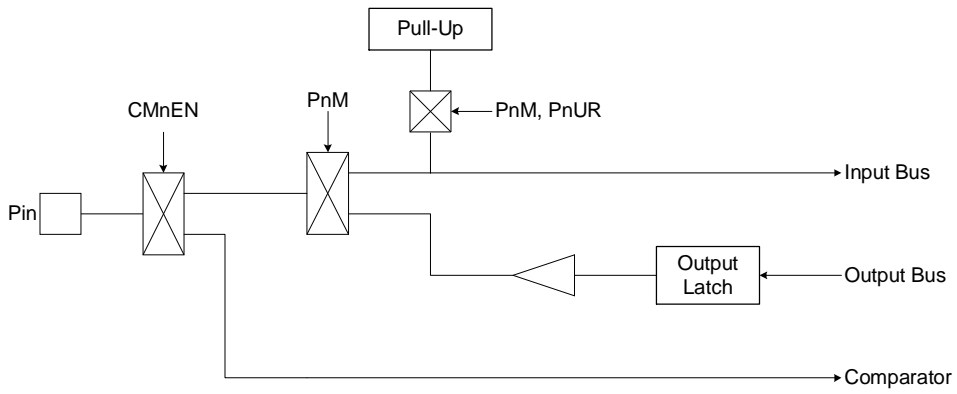
**Port 1.0, P1.1 structure:**



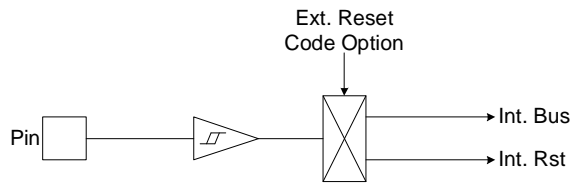
**Port 0.3, 0.4 structure:**



**Port 2 structure:**



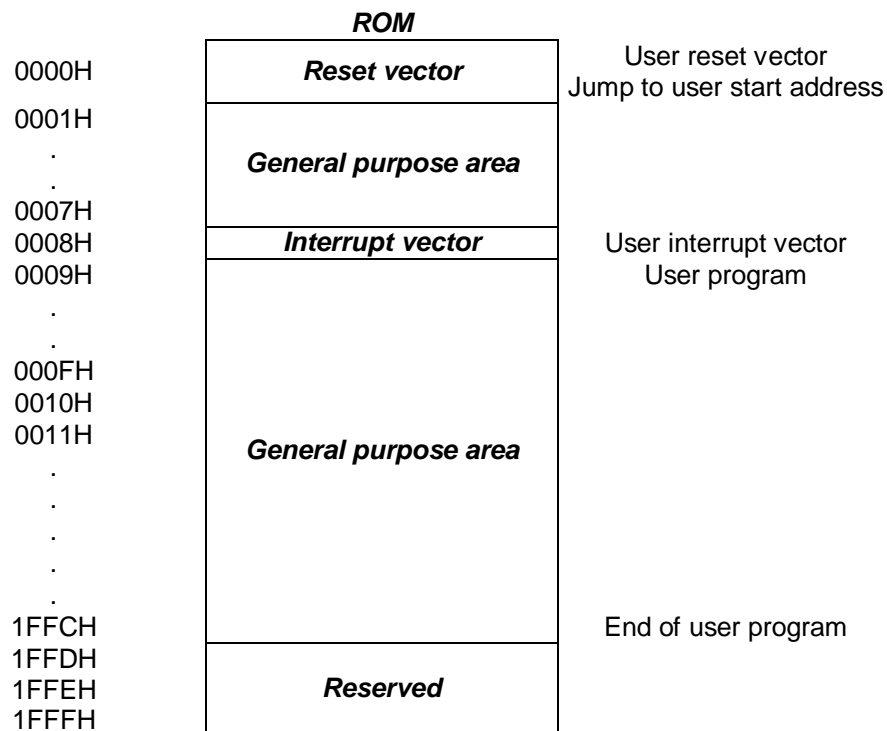
**Port 0.2 structure:**



# 2 CENTRAL PROCESSOR UNIT (CPU)

## 2.1 PROGRAM MEMORY (ROM)

F 8K words ROM



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

## 2.1.1 RESET VECTOR (0000H)

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

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

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

### Ø Example: Defining Reset Vector

```

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

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

                ENDP             ; End of program

```



## 2.1.2 INTERRUPT VECTOR (0008H)

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

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

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

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

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

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

    ENDP             ; End of program
```

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

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

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                   ; Save ACC and PFLAG register to buffers.
    ...
    ...
    POP                    ; Load ACC and PFLAG register from buffers.
    RETI                   ; End of interrupt service routine.
    ...

    ENDP              ; End of program.
```

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

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

### 2.1.3 LOOK-UP TABLE DESCRIPTION

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

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

```

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

                                ; Increment the index address for next address.
        INCMS    Z                ; Z+1
        JMP     @F                ; Z is not overflow.
        INCMS    Y                ; Z overflow (FFH à 00), à Y=Y+1
        NOP

@@:    MOVC     ; To lookup data, R = 51H, ACC = 05H.
        ...
TABLE1: DW     0035H            ; To define a word (16 bits) data.
        DW     5105H
        DW     2012H
        ...

```

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

Ø Example: INC\_YZ macro.

```

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

        INCMS    Y                ; Y+1
        NOP     ; Not overflow

@@:
        ENDM

```

Ø Example: Modify above example by "INC\_YZ" macro.

```

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

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

```

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

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

```

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

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

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

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

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

```

## 2.1.4 JUMP TABLE DESCRIPTION

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

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

### Ø Example: Jump table.

```

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

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

```

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

### Ø Example: If “jump table” crosses over ROM boundary will cause errors.

```

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

```

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

Ø Example: “@JMP\_A” application in SONIX macro file called “MACRO3.H”.

```

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

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

Ø Example: “@JMP\_A” operation.

**; Before compiling program.**

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

**; After compiling program.**

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

## 2.1.5 CHECKSUM CALCULATION

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

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

```

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

@@:
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

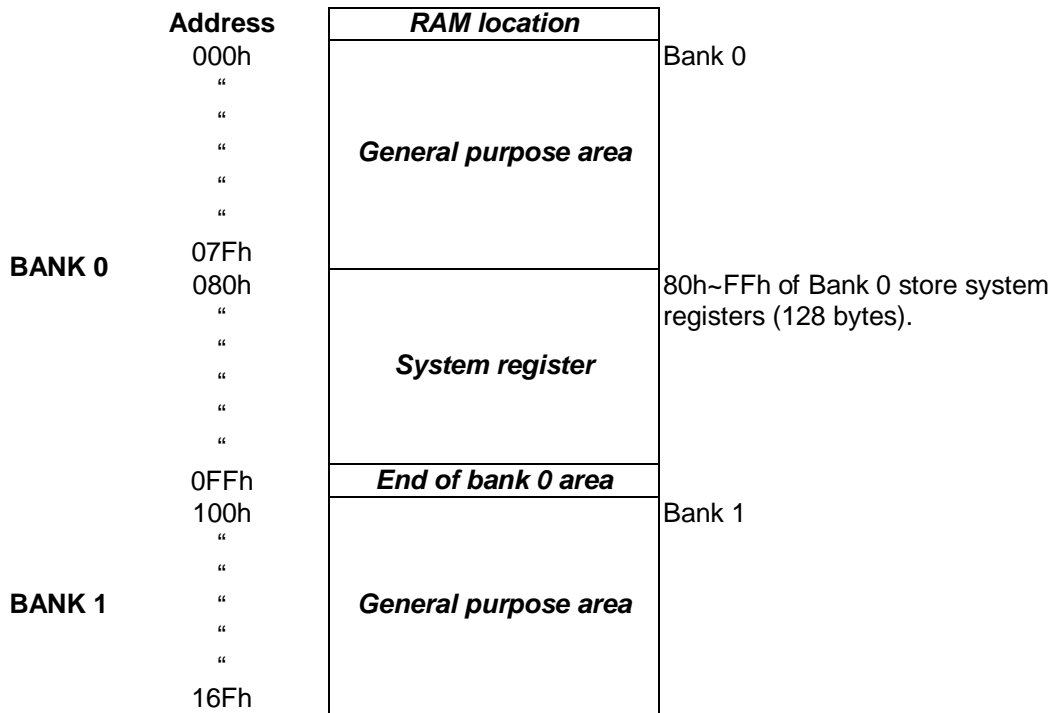
CHECKSUM_END:
...
...

END_USER_CODE:           ; Label of program end

```

## 2.2 DATA MEMORY (RAM)

### F 240 X 8-bit RAM



The 240-byte general purpose RAM is separated into Bank 0 and Bank 1. Accessing the two banks' RAM is controlled by "RBANK" register. When RBANK = 0, the program controls Bank 0 RAM directly. When RBANK = 1, the program controls Bank 1 RAM directly. Under one bank condition and need to access the other bank RAM, setup the RBANK register is necessary. Sonix provides "Bank 0" type instructions (e.g. b0mov, b0add, b0bts1, b0bset...) to control Bank 0 RAM in non-zero RAM bank condition directly.

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

; Bank 1 (RBANK = 1)

**B0MOV** A, WK00  
**MOV** WK01,A

; Use Bank 0 type instruction to access Bank 0 RAM.

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



## 2.2.1 SYSTEM REGISTER

### 2.2.1.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	-	PFLAG	RBANK	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	CMP0M	CMP1M	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	P2	-	-	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	P2UR	-	-	P5UR	@HL	@YZ	TC0D	P1OC	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

### 2.2.1.2 SYSTEM REGISTER DESCRIPTION

PFLAG = ROM page and special flag register.

H, L = Working, @HL and ROM addressing register.

P1W = Port 1 wakeup register.

CMPnM = Comparator control register.

PEDGE = P0.0 edge direction register.

PnM = Port n input/output mode register.

P1OC = Port 1 open-drain control register.

INTRQ = Interrupt request register.

OSCM = Oscillator mode register.

T0M = T0 mode register.

TC0M = IR output control register.

TC0D = IR duty control register.

TC1C = TC1 counter register.

WDTR = Watchdog timer clear register.

STKP = Stack pointer buffer.

R = Working register and ROM look-up data buffer.

Y, Z = Working, @YZ and ROM addressing register.

RBANK = Ram bank selection register.

@HL = RAM HL indirect addressing index pointer.

@YZ = RAM YZ indirect addressing index pointer.

Pn = Port n data buffer.

PnUR = Port n pull-up resistor control register.

INTEN = Interrupt enable register.

PCH, PCL = Program counter.

T0C = T0 counting register.

TC0C = IR cycle control register.

TC0R = IR auto-reload register.

TC1M = TC1 mode control register.

TC1R = TC1 auto-reload buffer.

STK0~STK7 = Stack 0 ~ stack 7 buffer.

**2.2.1.3BIT DEFINITION of SYSTEM REGISTER**

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD	LVD30	LVD24	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	-	-	RBANKS0	R/W	RBANK
09CH	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CMS1	CMS0	R/W	CMP0M
09DH	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	-	-	R/W	CMP1M
0B8H	-	-	-	P04M	P03M	-	P01M	P00M	R/W	P0M
0BFH	-	-	-	P00G1	P00G0	-	-	-	R/W	PEDGE
0C0H	P17W	P16W	P15W	P14W	13W	P12W	P11W	P10W	W	P1W wakeup register
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M I/O direction
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M I/O direction
0C5H	-	-	-	P54M	P53M	P52M	P51M	P50M	R/W	P5M I/O direction
0C8H	-	TC1IRQ	-	TOIRQ	-	-	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	-	TC1IEN	-	TOIEN	-	-	P01IEN	P00IEN	R/W	INTEN
0CAH	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	-	-	-	P04	P03	P02	P01	P00	R/W	P0 data buffer
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1 data buffer
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2 data buffer
0D5H	-	-	-	P54	P53	P52	P51	P50	R/W	P5 data buffer
0D8H	TOENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	-	-	-	-	-	-	IREN	CREN	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0	R/W	STKP stack pointer
0E0H	-	-	-	P04R	P03R	-	P01R	P00R	W	P0 pull-up register
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1 pull-up register
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2 pull-up register
0E5H	-	-	-	P54R	P53R	P52R	P51R	P50R	W	P5 pull-up register
0E6H	@HL7	@ HL 6	@ HL5	@ HL4	@ HL3	@ HL2	@ HL1	@ HL0	R/W	@HL index pointer
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ index pointer
0E8H	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0	W	TC0D
0E9H	-	-	-	-	-	-	P11OC	P10OC	W	P10C
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	1	1	1	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	1	1	1	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	1	1	1	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	1	1	1	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	1	1	1	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	1	1	1	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	1	1	1	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	1	1	1	S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

- **Note:**

1. *To avoid system error, please be sure to put all the “0” and “1” as it indicates in the above table.*
2. *All of register names had been declared in SN8ASM assembler.*
3. *One-bit name had been declared in SN8ASM assembler with “F” prefix code.*
4. *“b0bset”, “b0bclr”, “bset”, “bclr” instructions are only available to the “R/W” registers.*
5. *For detail description, please refer to the “System Register Quick Reference Table”.*

## 2.2.2 ACCUMULATOR

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

### Ø Example: Read and write ACC value.

; Read ACC data and store in BUF data memory.

```
MOV     BUF, A
```

; Write a immediate data into ACC.

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

; Write ACC data from BUF data memory.

```
MOV     A, BUF
```

; or

```
B0MOV   A, BUF
```

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

### Ø Example: Protect ACC and working registers.

INT\_SERVICE:

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

```
...
```

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

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

### 2.2.3 PROGRAM FLAG

The PFLAG register contains the arithmetic status of ALU operation, system reset status and LVD detecting status. NT0, NPD bits indicate system reset status including power on reset, LVD reset, reset by external pin active and watchdog reset. C, DC, Z bits indicate the result status of ALU operation. LVD24, LVD28 bits indicate LVD detecting power voltage status.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PFLAG</b>	NT0	NPD	LVD28	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

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

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

Bit 5 **LVD30:** LVD 2.8V operating flag and only support LVD code option is LVD\_H.

0 = Inactive ( $V_{DD} > 2.8V$ ).

1 = Active ( $V_{DD} \leq 2.8V$ ).

Bit 4 **LVD24:** LVD 2.4V operating flag and only support LVD code option is LVD\_M.

0 = Inactive ( $V_{DD} > 2.4V$ ).

1 = Active ( $V_{DD} \leq 2.4V$ ).

Bit 2 **C:** Carry flag

1 = Addition with carry, subtraction without borrowing, rotation with shifting out logic "1", comparison result  $\geq 0$ .

0 = Addition without carry, subtraction with borrowing signal, rotation with shifting out logic "0", comparison result  $< 0$ .

Bit 1 **DC:** Decimal carry flag

1 = Addition with carry from low nibble, subtraction without borrow from high nibble.

0 = Addition without carry from low nibble, subtraction with borrow from high nibble.

Bit 0 **Z:** Zero flag

1 = The result of an arithmetic/logic/branch operation is zero.

0 = The result of an arithmetic/logic/branch operation is not zero.

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

## 2.2.4 PROGRAM COUNTER

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

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

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

### F ONE ADDRESS SKIPPING

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

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

```

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

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

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

```

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

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

**INCS instruction:**

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

...

...

C0STEP:      **NOP**

**INCMS instruction:**

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

...

...

C0STEP:      **NOP**

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

**DECS instruction:**

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

...

...

C0STEP:      **NOP**

**DECMS instruction:**

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

...

...

C0STEP:      **NOP**

## F MULTI-ADDRESS JUMPING

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

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

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

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

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

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

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



## 2.2.5 H, L REGISTERS

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

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

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

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

**Example: If want to read a data from RAM address 20H of bank\_0, it can use indirectly addressing mode to access data as following.**

```

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

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

```

CLR      H              ; H = 0, bank 0
B0MOV    L, #07FH       ; L = 7FH, the last address of the data memory area

CLR_HL_BUF:
CLR      @HL            ; Clear @HL to be zero
DECMS    L              ; L - 1, if L = 0, finish the routine
JMP      CLR_HL_BUF     ; Not zero

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

## 2.2.6 Y, Z REGISTERS

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

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

084H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>Y</b>	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
<b>Z</b>	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
END_CLR:
...
    
```

## 2.2.7 R REGISTERS

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

- I Can be used as working register
- I 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
<b>R</b>	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

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

## 2.3 ADDRESSING MODE

### 2.3.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

### 2.3.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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

### 2.3.3 INDIRECTLY ADDRESSING MODE

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

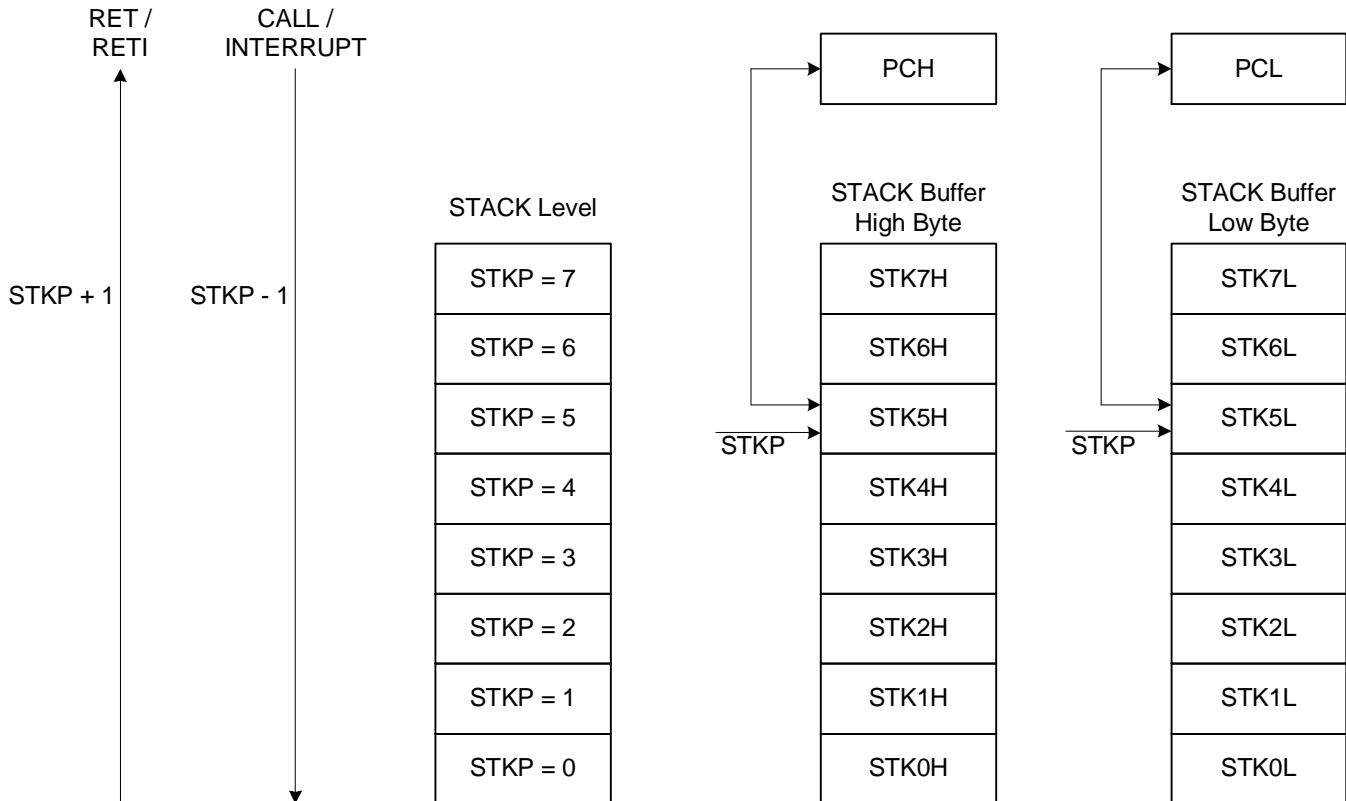
Ø **Example: Indirectly addressing mode with @YZ register.**

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

## 2.4 STACK OPERATION

### 2.4.1 OVERVIEW

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



## 2.4.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 13-bit data memory (STK<sub>n</sub>H and STK<sub>n</sub>L) 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 (STK<sub>n</sub>H and STK<sub>n</sub>L) 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
<b>STKP</b>	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0
Read/Write	R/W	-	-	-	-	R/W	R/W	R/W
After reset	0	-	-	-	-	1	1	1

Bit[2:0] **STKPB<sub>n</sub>**: Stack pointer (n = 0 ~ 2)

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

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

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

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>STK<sub>n</sub>H</b>	-	-	-	SnPC12	SnPC11	SnPC10	SnPC9	SnPC8
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>STK<sub>n</sub>L</b>	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

**STK<sub>n</sub>** = STK<sub>n</sub>H , STK<sub>n</sub>L (n = 7 ~ 0)

### 2.4.3 STACK OPERATION EXAMPLE

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

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

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

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

## 2.5 CODE OPTION TABLE

The code option is the system hardware configurations including oscillator type, watchdog timer operation, LVD option, reset pin option and OTP ROM security control. The code option items are as following table:

Code Option	Content	Function Description
Fcpu	Fhosc/1	Instruction cycle is oscillator clock.
	Fhosc/2	Instruction cycle is 2 oscillator clocks.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
High_Clk	IHRC_8M	High speed internal 8MHz RC. XIN/XOUT become to P0.3/P0.4 bi-direction I/O pins.
	IHRC_RTC	High speed internal 8MHz RC with 0.5sec RTC. XIN/XOUT become to P0.3/P0.4 bit-direction I/O pins.
	RC	Low cost RC for external high clock oscillator and XOUT becomes to P0.4 bit-direction I/O pin.
	32K X'tal	Low frequency, power saving crystal (e.g. 32.768KHz) for external high clock oscillator.
	8M X'tal	High speed crystal /resonator (e.g. 8MHz) for external high clock oscillator.
	4M X'tal	Standard crystal /resonator (e.g. 4M) for external high clock oscillator.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
Reset_Pin	Reset	Enable External reset pin.
	P02	Enable P0.2 input only without pull-up resistor.
LVD	LVD_L	LVD will reset chip if VDD is below 1.8V
	LVD_M	LVD will reset chip if VDD is below 1.8V Enable LVD24 bit of PFLAG register for 2.4V low voltage indicator.
	LVD_H	LVD will reset chip if VDD is below 2.4V Enable LVD28 bit of PFLAG register for 2.8V low voltage indicator.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.

### 2.5.1 RESET\_PIN CODE OPTION

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

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

### 2.5.2 SECURITY CODE OPTION

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



# 3 RESET

## 3.1 OVERVIEW

The system would be reset in three conditions as following.

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

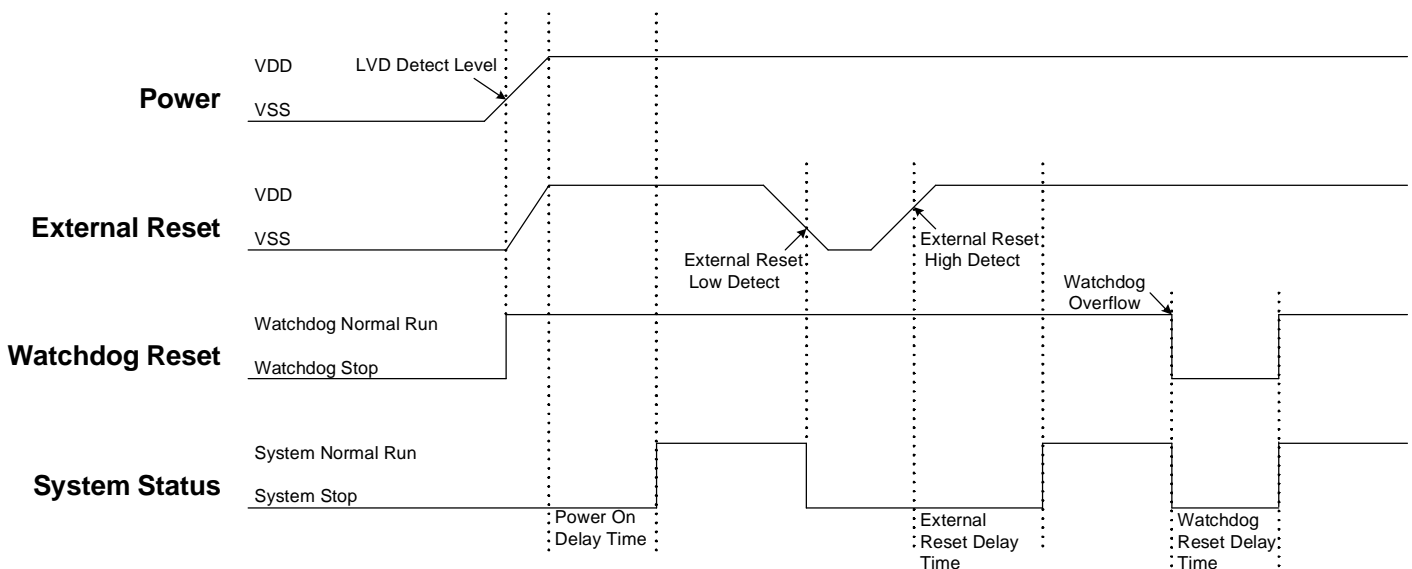
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PFLAG</b>	NT0	NPD	LVD28	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

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

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

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



## 3.2 POWER ON RESET

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

- I **Power-up:** System detects the power voltage up and waits for power stable.
- I **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.
- I **System initialization:** All system registers is set as initial conditions and system is ready.
- I **Oscillator warm up:** Oscillator operation is successfully and supply to system clock.
- I **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.

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

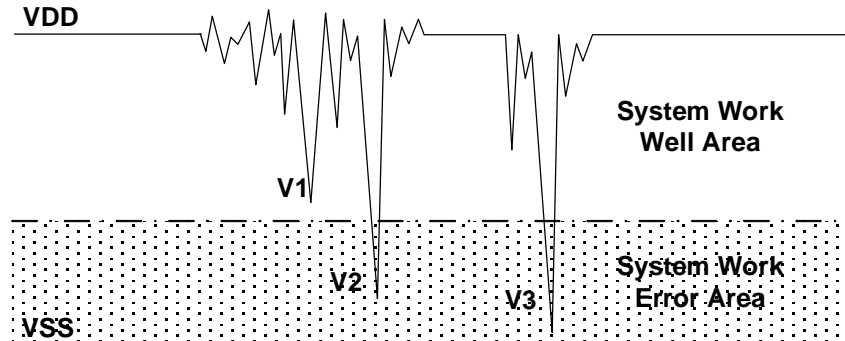
Watchdog timer application note is as following.

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

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

### 3.4 BROWN OUT RESET

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



**Brown Out Reset Diagram**

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

**DC application:**

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

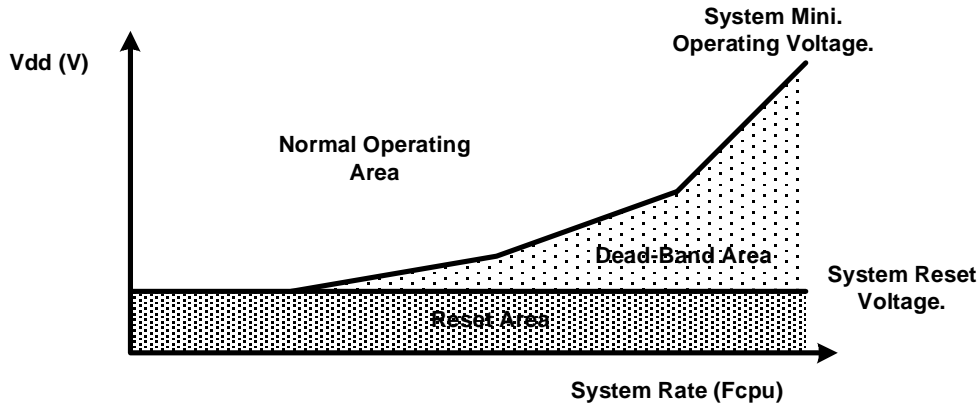
**AC application:**

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

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

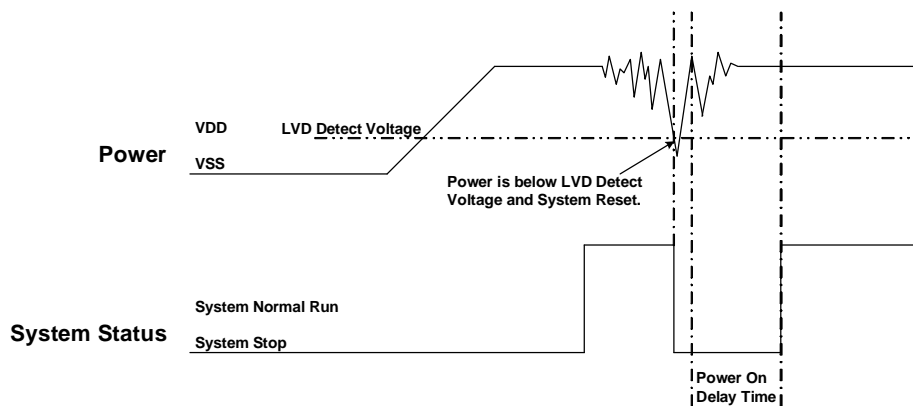
### 3.4.1 THE SYSTEM OPERATING VOLTAGE

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



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

### 3.4.2 LOW VOLTAGE DETECTOR (LVD)



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

The LVD is three levels design (1.8V/2.4V/2.8V) and controlled by LVD code option. The 1.8V LVD is always enable for power on reset and Brown Out reset. The 2.4V LVD includes LVD reset function and flag function to indicate VDD status function. The 2.8V includes flag function to indicate VDD status. LVD flag function can be an **easy low battery detector**. LVD24, LVD28 flags indicate VDD voltage level. For low battery detect application, only checking LVD24, LVD28 status to be battery status. This is a cheap and easy solution.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PFLAG</b>	NT0	NPD	LVD28	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

Bit 5 **LVD28:** LVD 2.8V operating flag and only support LVD code option is LVD\_H.  
 0 = Inactive (VDD > 2.8V).  
 1 = Active (VDD ≤ 2.8V).

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

LVD	LVD Code Option		
	LVD_L	LVD_M	LVD_H
1.8V Reset	Available	Available	Available
2.4V Flag	-	Available	-
2.4V Reset	-	-	Available
2.8V Flag	-	-	Available

#### **LVD\_L**

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

#### **LVD\_M**

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

#### **LVD2\_H**

If VDD < 2.4V, system will be reset.  
 Enable LVD24 bit of PFLAG register. If VDD > 2.4V, LVD24 is "0". If VDD ≤ 2.4V, LVD24 flag is "1"  
 Enable LVD28 bit of PFLAG register. If VDD > 2.8V, LVD28 is "0". If VDD ≤ 2.8V, LVD28 flag is "1"

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

### 3.4.3 BROWN OUT RESET IMPROVEMENT

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

- I LVD reset
- I Watchdog reset
- I Reduce the system executing rate
- I 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

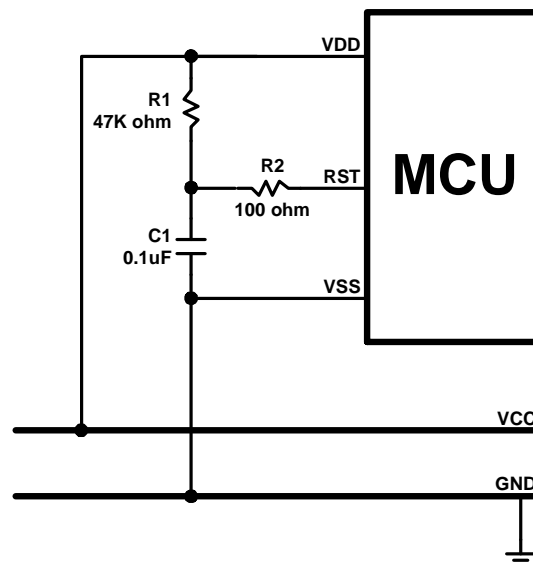
External reset function is controlled by “Reset\_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the 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.

- I **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.
- I **System initialization:** All system registers is set as initial conditions and system is ready.
- I **Oscillator warm up:** Oscillator operation is successfully and supply to system clock.
- I **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

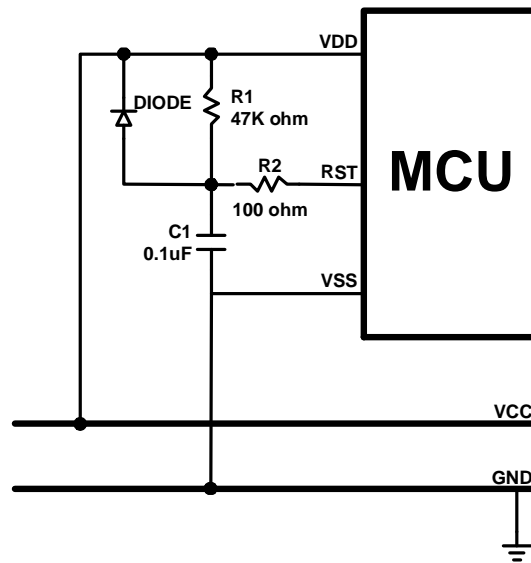
### 3.6.1 Simply RC Reset Circuit



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.**

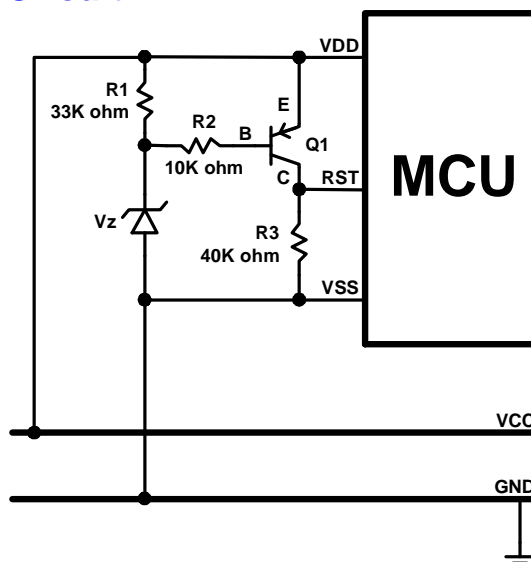
### 3.6.2 Diode & RC Reset Circuit



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).

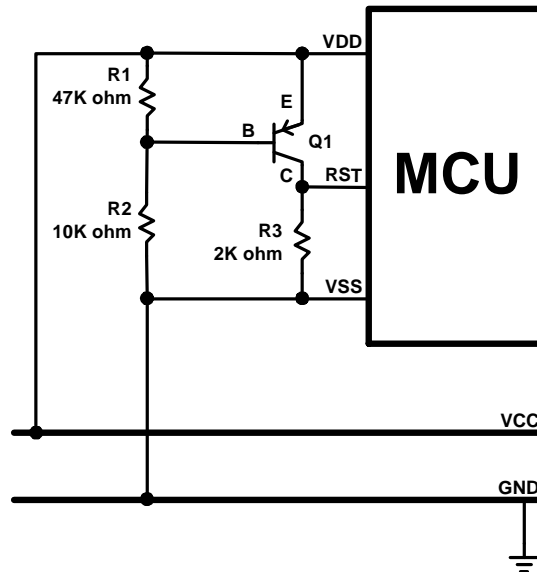
### 3.6.3 Zener Diode Reset Circuit



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

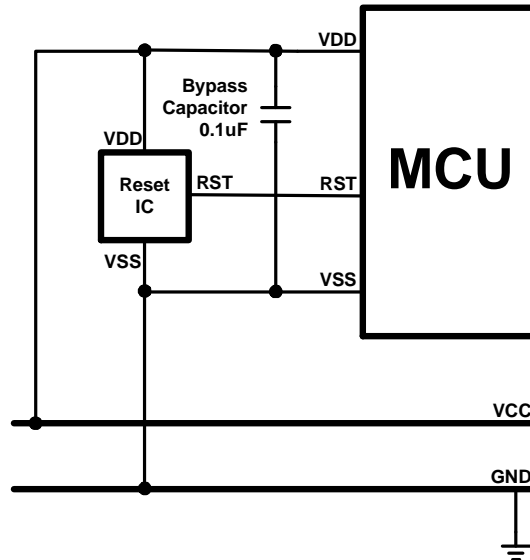


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to  $0.7V \times (R1 + R2) / R1$ , 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

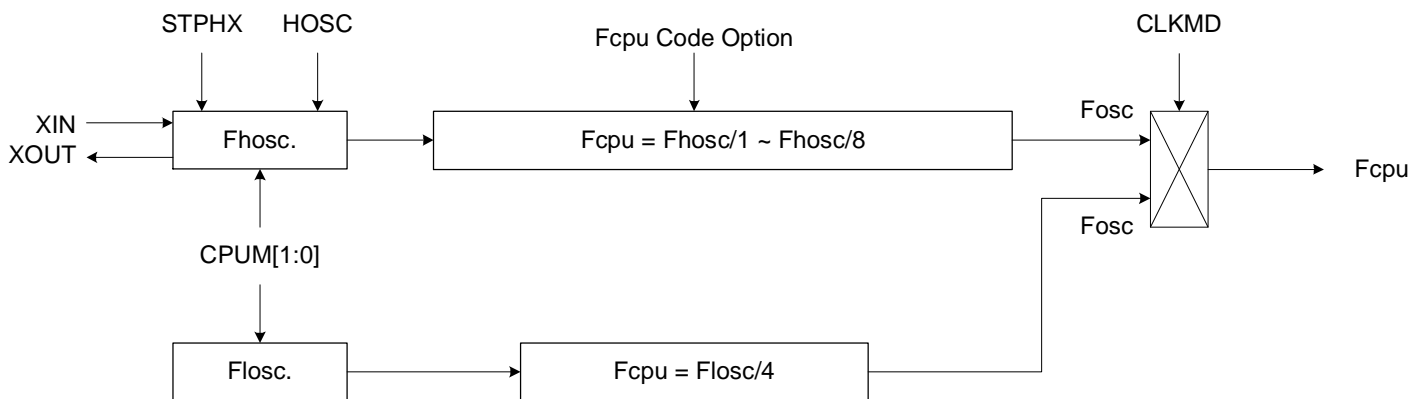
The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit or on-chip 8MHz high-speed RC oscillator circuit (IHRC 8MHz). The low-speed clock is generated from on-chip low-speed RC oscillator circuit (ILRC 10KHz @3V).

Both the high-speed clock and the low-speed clock can be system clock (Fosc). The system clock in slow mode is divided by 4 to be the instruction cycle (Fcpu).

**F Normal Mode (High Clock):**  $F_{cpu} = F_{osc} / N$ ,  $N = 1 \sim 8$ , Select N by Fcpu code option.

**F Slow Mode (Low Clock):**  $F_{cpu} = F_{osc}/4$ .

## 4.2 CLOCK BLOCK DIAGRAM



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

## 4.3 Fcpu (INSTRUCTION CYCLE)

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

**In high noisy environment, below “Fosc/4” of Fcpu code option is the strongly recommendation to reduce high frequency noise effect.**

## 4.4 OSCM REGISTER

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

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

- Bit 1     **STPHX**: External high-speed oscillator control bit.  
 0 = External high-speed oscillator free run.  
 1 = External high-speed oscillator free run stop. Internal low-speed RC oscillator is still running.
- Bit 2     **CLKMD**: System high/Low clock mode control bit.  
 0 = Normal (dual) mode. System clock is high clock.  
 1 = Slow mode. System clock is internal low clock.
- Bit[4:3]   **CPUM[1:0]**: CPU operating mode control bits.  
 00 = normal.  
 01 = sleep (power down) mode.  
 10 = green mode.  
 11 = reserved.

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

**High\_CLK = IHRC\_8M**: “STPHX=1” disables internal high speed RC type oscillator.

**High\_CLK = IHRC\_RTC**: “STPHX=1” disables internal high speed RC type oscillator and external 32768Hz crystal.

**High\_CLK = RC, 4M, 8M, 32K**: “STPHX=1” disables external oscillator.

## 4.5 SYSTEM HIGH CLOCK

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

- I **IHRC\_8M:** The system clock source is from internal high speed 8MHz RC type oscillator. In the mode, XIN and XOUT pins are bi-direction GPIO mode, and not to connect any external oscillator.
- I **IHRC\_RTC:** The system clock source is from internal high speed 8MHz RC type oscillator, and includes external low speed 32768Hz crystal for RTC function. The XIN and XOUT pins are defined to drive external 32768Hz crystal and not GPIO mode.
- I **RC:** The system clock source is from external low cost RC type oscillator. The RC oscillator circuit only connects to XIN pin, and the XOUT pin is bi-direction GPIO mode.
- I **32K X'tal:** The system clock source is from external low speed 32768Hz crystal. The option only supports 32768Hz crystal.
- I **8M X'tal:** The system clock source is from external high speed crystal/resonator. The oscillator bandwidth is 8MHz~16MHz.
- I **4M X'tal:** The system clock source is from external high speed crystal/resonator. The oscillator bandwidth is 1MHz~10MHz.

### 4.5.1 INTERNAL HIGH RC

The chip is built-in RC type internal high clock (8MHz) controlled by "IHRC\_8M" or "IHRC\_RTC" code options. In "IHRC\_8M" mode, the system clock is from internal 8MHz RC type oscillator and XIN / XOUT pins are general-purpose I/O pins. In "IHRC\_RTC" mode, the system clock is from internal 8MHz RC type oscillator and XIN / XOUT pins are connected with external 32768 crystal for real time clock (RTC).

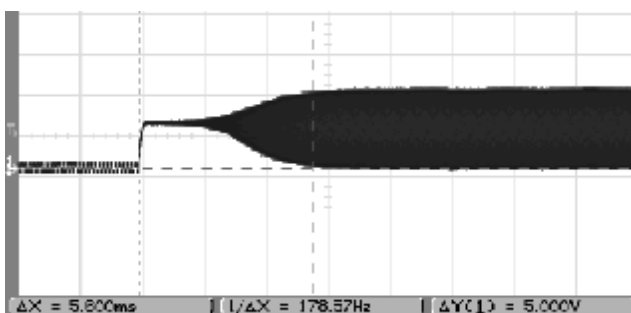
- I **IHRC:** High clock is internal 8MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.
- I **IHRC\_RTC:** High clock is internal 8MHz oscillator RC type. XIN/XOUT pins are connected with external 32768Hz crystal/ceramic oscillator for RTC clock source.

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

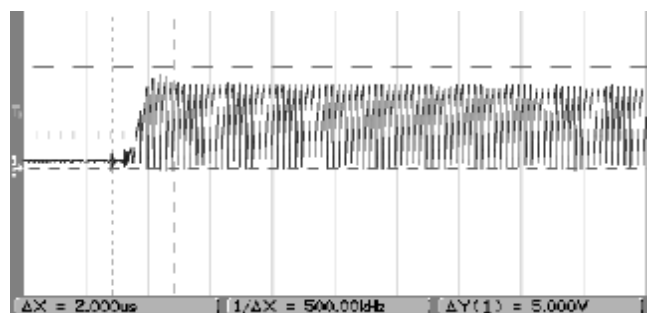
### 4.5.2 EXTERNAL HIGH CLOCK

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.

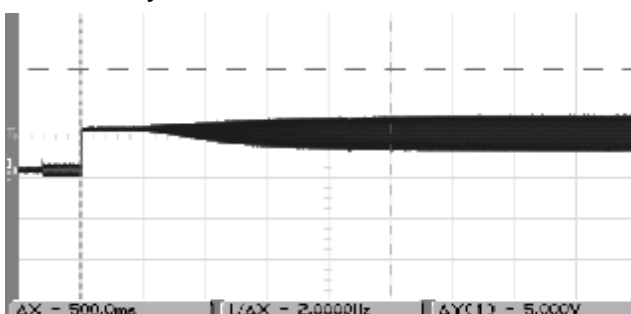
4MHz Crystal



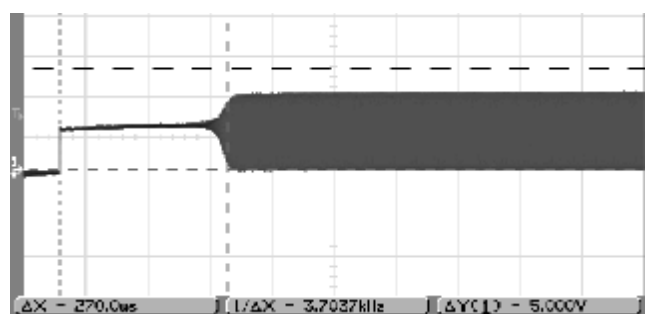
RC



32768Hz Crystal

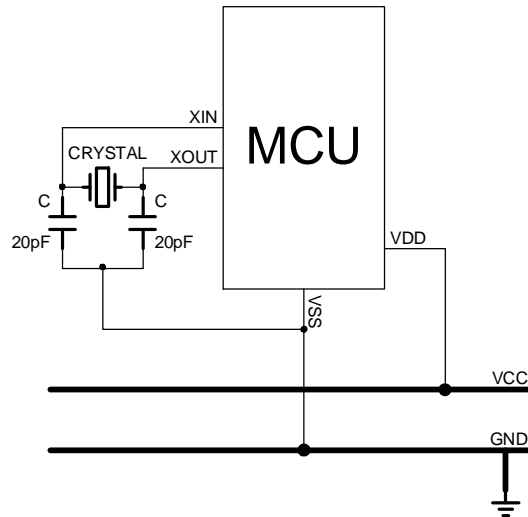


4MHz Ceramic



#### 4.5.2.1 CRYSTAL/CERAMIC

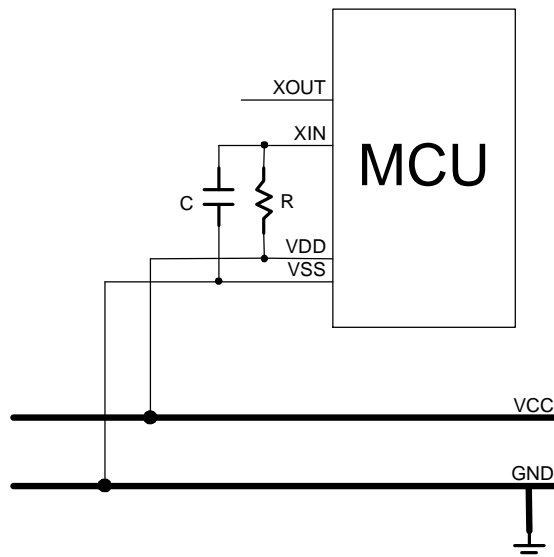
Crystal/Ceramic devices are driven by XIN, XOUT pins. For high/normal/low frequency, the driving currents are different. High\_Clk code option supports different frequencies. 8M option is for high speed (ex. 8MHz). 4M option is for normal speed (ex. 4MHz). 32K option is for low speed (ex. 32768Hz).



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

#### 4.5.2.2 RC

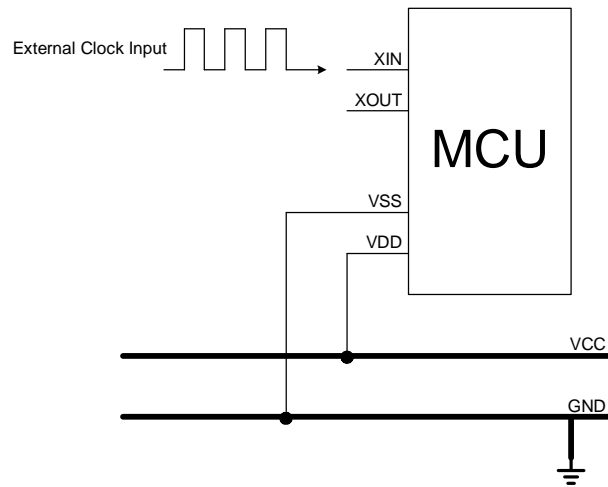
Selecting RC oscillator is by RC option of High\_Clk code option. RC type oscillator's frequency is up to 10MHz. Using "R" value is to change frequency. 50P~100P is good value for "C". XOUT pin is general purpose I/O pin.



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

### 4.5.2.3 EXTERNAL CLOCK SIGNAL

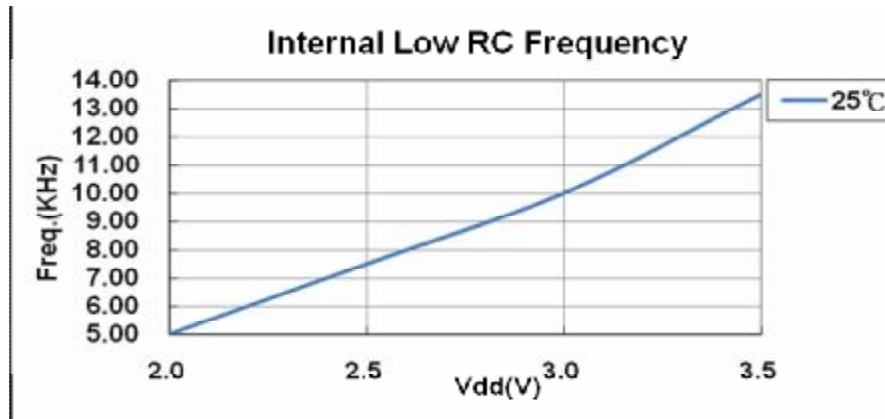
Selecting external clock signal input to be system clock is by RC option of High\_Clk code option. The external clock signal is input from XIN pin. XOUT pin is general purpose I/O pin.



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

## 4.6 SYSTEM LOW CLOCK

The system low clock source is the internal low-speed oscillator built in the micro-controller. The low-speed oscillator uses RC type oscillator circuit. The frequency is affected by the voltage and temperature of the system. In common condition, the frequency of the RC oscillator is about 10KHz at 3V. The relation between the RC frequency and voltage is as the following figure.



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

F  $F_{osc} = \text{Internal low RC oscillator (about 10KHz @3V)}$ .

F  $\text{Slow mode } F_{cpu} = F_{osc} / 4$

There are two conditions to stop internal low RC. One is power down mode, and the other is green mode of 455K mode and watchdog disable. If system is in 455K mode and watchdog disable, only 455K oscillator activates and system is under low power consumption.

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

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

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



#### 4.6.1 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
```

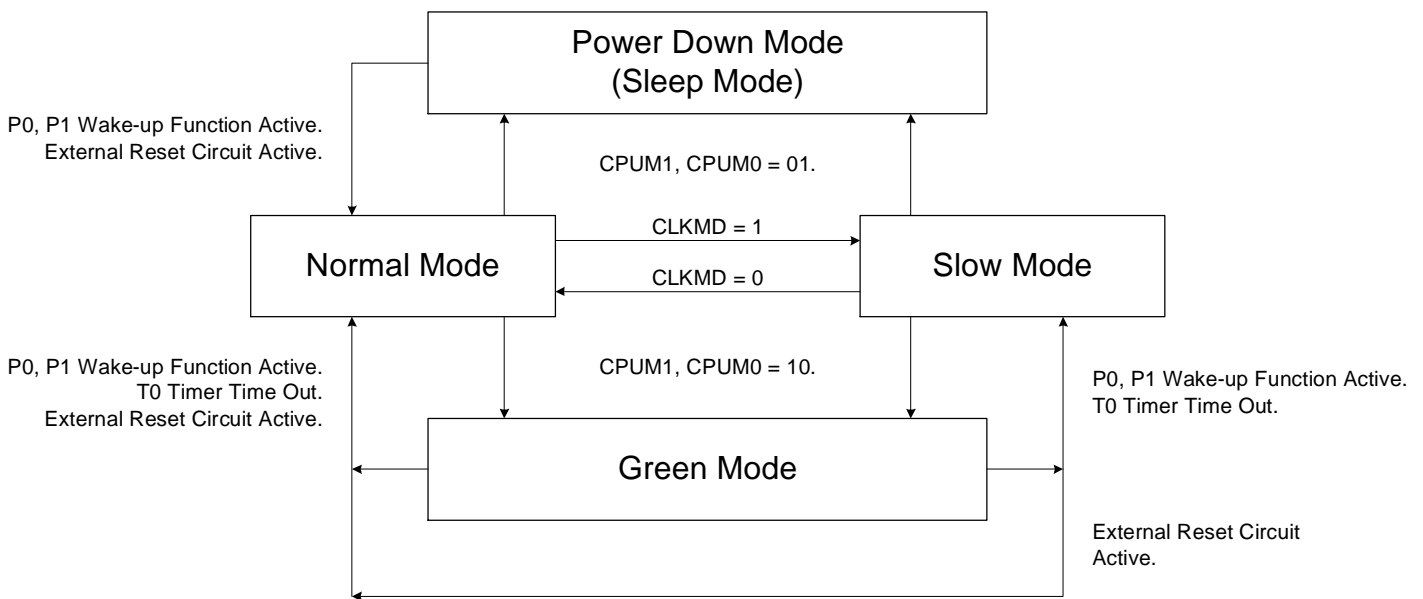
- ***Note: Do not measure the RC frequency directly from XIN; the probe impedance will affect the RC frequency.***

# 5 SYSTEM OPERATION MODE

## 5.1 OVERVIEW

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

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



System Mode Switching Diagram

### Operating mode description

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

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

## 5.2 NORMAL MODE

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

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

## 5.3 SLOW MODE

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

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

## 5.4 POWER DOWN MDOE

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

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

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

## 5.5 GREEN MODE

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

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

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

## 5.6 OPERATING MODE CONTROL MACRO

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

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

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

**SleepMode** ; Declare "SleepMode" macro directly.

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

**SlowMode** ; Declare "SlowMode" macro directly.

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

**Slow2Normal** ; Declare "Slow2Normal" macro directly.

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

**GreenMode** ; Declare "GreenMode" macro directly.

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

; Set T0 timer wakeup function.

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

; Go into green mode

**GreenMode**

; Declare "GreenMode" macro directly.

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

CLR	T0C	; Clear T0 counter.
<b>B0BSET</b>	<b>FT0ENB</b>	<b>; To enable T0 timer</b>

; Go into green mode

**GreenMode**

; Declare "GreenMode" macro directly.

## 5.7 WAKEUP

### 5.7.1 OVERVIEW

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

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

### 5.7.2 WAKEUP TIME

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

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

The value of the wakeup time is as the following.

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

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

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

***The wakeup time =  $1/F_{osc} * 2048 = 0.512$  ms ( $F_{osc} = 4$ MHz)  
The total wakeup time = 0.512 ms + oscillator start-up time***

### 5.7.3 P1W WAKEUP CONTROL REGISTER

Under power down mode (sleep mode) and green mode, the I/O ports with wakeup function are able to wake the system up to normal mode. The Port 0 and Port 1 have wakeup function. Port 0 wakeup function always enables, but the Port 1 is controlled by the P1W register.

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

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

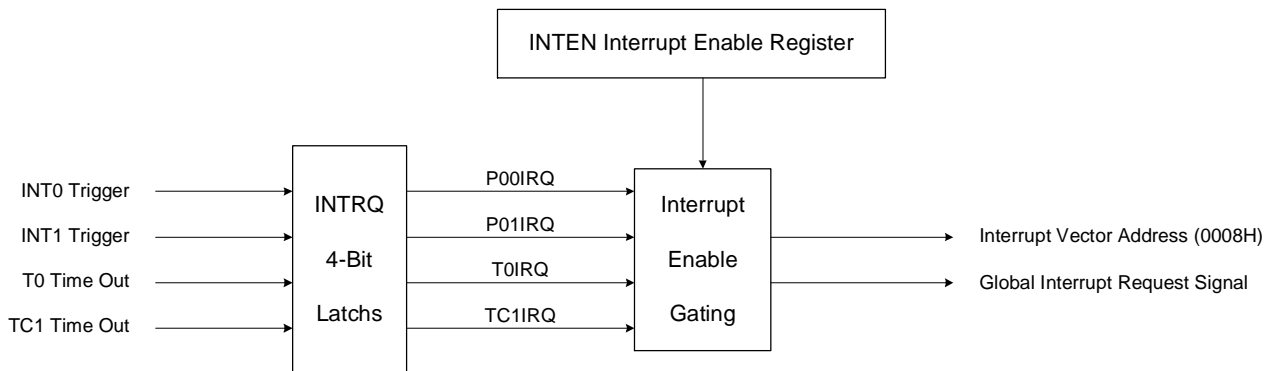
0 = Disable P1n wakeup function.

1 = Enable P1n wakeup function.

# 6 INTERRUPT

## 6.1 OVERVIEW

This MCU provides three interrupt sources, including two internal interrupt (T0/TC1) and two external interrupt (INT0, INT1). The external interrupt can wakeup the chip while the system is switched from power down mode to high-speed normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to "0" for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to "1" to accept the next interrupts' request. All of the interrupt request signals are stored in INTRQ register.



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



## 6.2 INTEN INTERRUPT ENABLE REGISTER

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>INTEN</b>	-	TC1IEN	-	TOIEN	-	-	P01IEN	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 1     **P01IEN:** External P0.1 interrupt (INT1) control bit.  
 0 = Disable INT1 interrupt function.  
 1 = Enable INT1 interrupt function.

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

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

## 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
<b>INTRQ</b>	-	TC1IRQ	-	T0IRQ	-	-	P01IRQ	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 1     **P01IRQ:** External P0.1 interrupt (INT1) request flag.  
 0 = None INT1 interrupt request.  
 1 = INT1 interrupt request.

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

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

## 6.4 GIE GLOBAL INTERRUPT OPERATION

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

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>STKP</b>	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.

Ø     **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

START:         ORG      10H
                ...

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
<b>PEDGE</b>	-	-	-	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
BOMOV   PEDGE, A      ; Set INT0 interrupt trigger as bi-direction edge.

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

**Example: INT0 interrupt service routine.**

```

ORG      8          ; Interrupt vector
JMP     INT_SERVICE

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

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

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

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

## 6.7 INT1 (P0.1) INTERRUPT OPERATION

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

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

**Example: INT1 interrupt request setup.**

```

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

```

**Example: INT1 interrupt service routine.**

```

ORG          8            ; Interrupt vector
INT_SERVICE:
  JMP        INT_SERVICE

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

  B0BTS1     FP01IRQ      ; Check P01IRQ
  JMP        EXIT_INT     ; P01IRQ = 0, exit interrupt vector

  B0BCLR     FP01IRQ      ; Reset P01IRQ
  ...                    ; INT1 interrupt service routine
EXIT_INT:
  ...                    ; Pop routine to load ACC and PFLAG from buffers.

  RETI         ; Exit interrupt vector

```

## 6.8 T0 INTERRUPT OPERATION

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

### Ø Example: T0 interrupt request setup.

```

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

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

B0BSET    FGIE          ; Enable GIE

```

### Example: T0 interrupt service routine as no RTC function.

```

ORG       8              ; Interrupt vector
INT_SERVICE:
JMP      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

```

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

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

```

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

                                RETI          ; Exit interrupt vector

```

## 6.9 TC1 INTERRUPT OPERATION

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

### Example: TC1 interrupt request setup.

```

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

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

B0BSET    FGIE       ; Enable GIE

```

### Example: TC1 interrupt service routine.

```

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

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

B0BCLR   FTC1IRQ    ; Reset TC1IRQ
MOV      A, #74H    ; Reset TC1C.
B0MOV    TC1C, A    ; TC1 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.
P01IRQ	P0.1 falling edge trigger.
T0IRQ	T0C overflow.
TC1IRQ	TC1C overflow.

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

### Ø Example: Check the interrupt request under multi-interrupt operation

```

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

INTP00CHK:
                B0BTS1       FP00IEN    ; Check INT0 interrupt request
                JMP           INTP01CHK  ; Check P00IEN
                B0BTS0       FP00IRQ    ; Jump check to next interrupt
                JMP           INTP00     ; Check P00IRQ
                ...           ; Jump to INT0 interrupt service routine
INTP01CHK:
                B0BTS1       FP01IEN    ; Check INT0 interrupt request
                JMP           INTT0CHK   ; Check P01IEN
                B0BTS0       FP01IRQ    ; Jump check to next interrupt
                JMP           INTP01     ; Check P01IRQ
                ...           ; Jump to INT1 interrupt service routine
INTT0CHK:
                B0BTS1       FT0IEN     ; Check T0 interrupt request
                JMP           INTTC1CHK  ; Check T0IEN
                B0BTS0       FT0IRQ     ; Jump check to next interrupt
                JMP           INTT0      ; Check T0IRQ
                ...           ; Jump to T0 interrupt service routine
INTTC1CHK:
                B0BTS1       FTC1IEN    ; Check TC1 interrupt request
                JMP           INT_EXIT   ; Check TC1IEN
                B0BTS0       FTC1IRQ    ; Jump to exit of IRQ
                JMP           INTTC1     ; Check TC1IRQ
                ...           ; Jump to TC1 interrupt service routine
INT_EXIT:
                ...           ; Pop routine to load ACC and PFLAG from buffers.

                RETI          ; Exit interrupt vector
    
```

# 7 I/O PORT

## 7.1 OVERVIEW

The micro-controller builds in 26 pin I/O. Most of the I/O pins are mixed with analog pins and special function pins. The I/O shared pin list is as following.

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

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

## 7.2 I/O PORT MODE

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

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

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

0C2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P2M</b>	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P5M</b>	-	-	-	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. **P0.2** is input only pin, and the **P0M.2** keeps "1".

Ø **Example: I/O mode selecting**

```

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

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

B0BCLR  P1M.2    ; Set P1.2 to be input mode.

B0BSET  P1M.2    ; Set P1.2 to be output mode.
    
```

## 7.3 I/O PULL UP REGISTER

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P0UR</b>	-	-	-	P04R	P03R	-	P01R	P00R
Read/Write	-	-	-	W	W	-	W	W
After reset	-	-	-	0	0	-	0	0

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

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

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

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

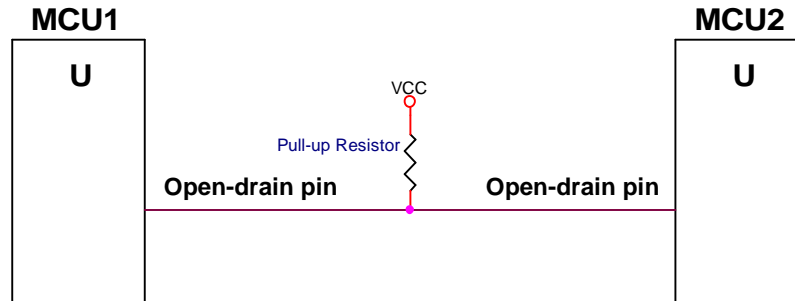
### Ø Example: I/O Pull up Register

```

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

## 7.4 I/O OPEN-DRAIN REGISTER

P1.0/P1.1 are built-in open-drain function. P1.0/P1.1 must be set as output mode when enable open-drain function. Open-drain external circuit is as following.



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

0E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P10C</b>	-	-	-	-	-	-	P110C	P100C
Read/Write	-	-	-	-	-	-	W	W
After reset	-	-	-	-	-	-	0	0

Bit 1 **P110C**: P1.1 open-drain control bit  
0 = Disable open-drain mode  
1 = Enable open-drain mode

Bit 0 **P100C**: P1.0 open-drain control bit  
0 = Disable open-drain mode  
1 = Enable open-drain mode

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

```

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

B0BSET    P10M           ; Enable P1.0 output mode.
MOV       A, #01H        ; Enable P1.0 open-drain function.
B0MOV     P10C, A
    
```

- **Note: P10C is write only register. Setting P100C must be used "MOV" instructions.**

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

```

MOV       A, #0           ; Disable P1.0 open-drain function.
B0MOV     P10C, A
    
```

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

## 7.5 I/O PORT DATA REGISTER

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

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

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

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P5</b>	-	-	-	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 P02 keeps "1" when external reset enable by code option.

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

```

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

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

```

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

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

```

B0BSET     P1.3           ; Set P1.3 and P5.4 to be "1".
B0BSET     P5.4

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

# 8 TIMERS

## 8.1 WATCHDOG TIMER

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

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

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

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

- I **Disable:** Disable watchdog timer function.
- I **Enable:** Enable watchdog timer function. Watchdog timer activates in normal mode and slow mode. In power down mode and green mode, the watchdog timer stops.
- I **Always\_On:** Enable watchdog timer function. The watchdog timer activates and not stop in power down mode and green mode.

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

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

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

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

Main:

```

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

```

Ø **Example: Clear watchdog timer by @RST\_WDT macro.**

```

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

Watchdog timer application note is as following.

- I Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- I Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- I 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.
                                ; Clear the watchdog timer.
        MOV          A, #5AH
        B0MOV        WDTR, A
        ...
        CALL          SUB1
        CALL          SUB2
        ...
        ...
        JMP           MAIN
    
```



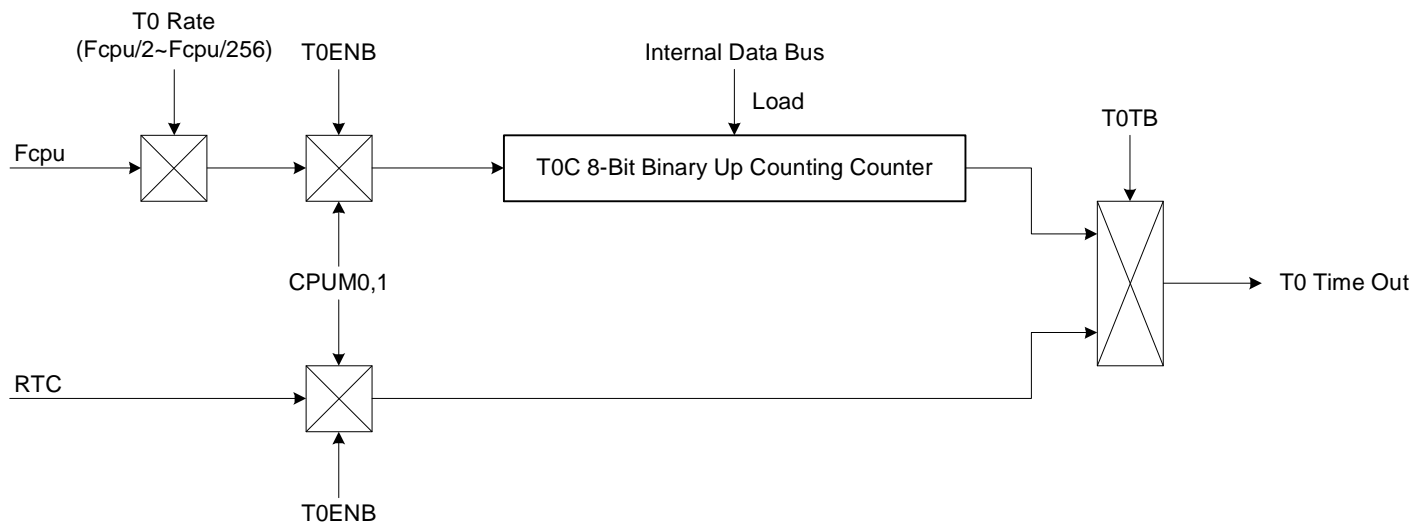
## 8.2 TIMER 0 (T0)

### 8.2.1 OVERVIEW

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

The main purposes of the T0 timer is as following.

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



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

## 8.2.2 T0M MODE REGISTER

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

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

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

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

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

### 8.2.3 T0C COUNTING REGISTER

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

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

The equation of T0C initial value is as following.

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

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

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

#### The basic timer table interval time of 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

- **Note:** In RTC mode, T0C is 256 counts and generates T0 0.5 sec interval time. Don't change T0C value in RTC mode.

## 8.2.4 T0 TIMER OPERATION SEQUENCE

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

### F Stop T0 timer counting, disable T0 interrupt function and clear T0 interrupt request flag.

```

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

```

### F Set T0 timer rate.

```

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

```

### F Set T0 clock source from Fcpu or RTC.

```

BOBCLR    FT0TB        ; Select T0 Fcpu clock source.
or
BOBSET    FT0TB        ; Select T0 RTC clock source.

```

### F Set T0 interrupt interval time.

```

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

```

### F Set T0 timer function mode.

```

BOBSET    FT0IEN       ; Enable T0 interrupt function.

```

### F Enable T0 timer.

```

BOBSET    FT0ENB       ; Enable T0 timer.

```

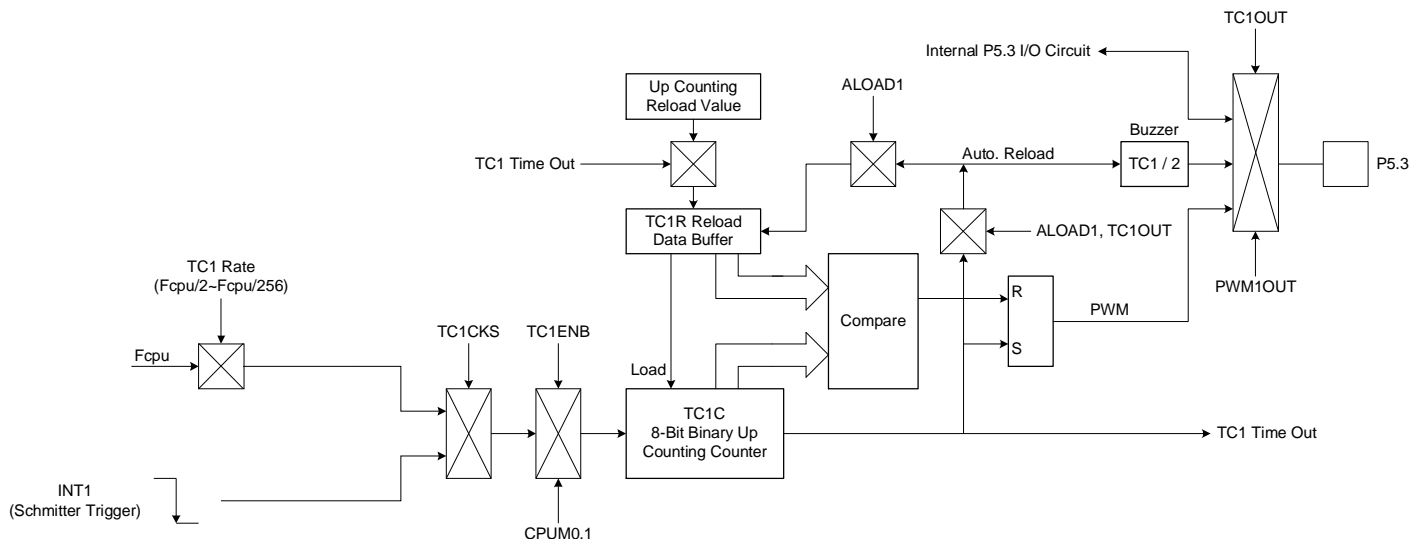
## 8.3 TIMER/COUNTER 0 (TC1)

### 8.3.1 OVERVIEW

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

The main purposes of the TC1 timer is as following.

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



### 8.3.2 TC1M MODE REGISTER

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

- Bit 0     **PWM1OUT:** PWM output control bit.  
0 = Disable PWM output.  
1 = Enable PWM output. PWM duty controlled by TC1OUT, ALOAD1 bits.
- Bit 1     **TC1OUT:** TC1 time out toggle signal output control bit. **Only valid when PWM1OUT = 0.**  
0 = Disable, P5.3 is I/O function.  
1 = Enable, P5.3 is output TC1OUT signal.
- Bit 2     **ALOAD1:** Auto-reload control bit. **Only valid when PWM1OUT = 0.**  
0 = Disable TC1 auto-reload function.  
1 = Enable TC1 auto-reload function.
- Bit 3     **TC1CKS:** TC1 clock source select bit.  
0 = Internal clock (Fcpu or Fosc).  
1 = External clock from P0.1/INT1 pin.
- Bit [6:4] **TC1RATE[2:0]:** TC1 internal clock select bits.  
000 = fcpu/256.  
001 = fcpu/128.  
...  
110 = fcpu/4.  
111 = fcpu/2.
- Bit 7     **TC1ENB:** TC1 counter control bit.  
0 = Disable TC1 timer.  
1 = Enable TC1 timer.

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

### 8.3.3 TC1C COUNTING REGISTER

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

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

The equation of TC1C initial value is as following.

$$TC1C \text{ initial value} = N - (TC1 \text{ interrupt interval time} * \text{input clock})$$

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

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

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

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

#### The basic timer table interval time of TC1.

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

### 8.3.4 TC1R AUTO-LOAD REGISTER

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

TC1 is double buffer design. If new TC1R value is set by program, the new value is stored in 1<sup>st</sup> buffer. Until TC1 overflow occurs, the new value moves to real TC1R buffer. This way can avoid TC1 interval time error and glitch in PWM and Buzzer output.

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

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

The equation of TC1R initial value is as following.

$$TC1R \text{ initial value} = N - (TC1 \text{ interrupt interval time} * \text{input clock})$$

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

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

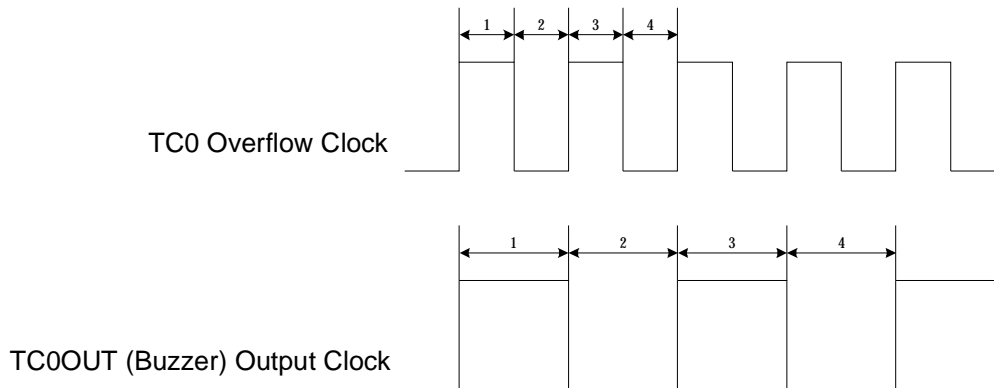
**Example: To set 10ms interval time for TC1 interrupt. TC1 clock source is Fcpu (TC1KS=0) and no PWM output (PWM1=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC1RATE=010 (Fcpu/64).**

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



### 8.3.5 TC1 CLOCK FREQUENCY OUTPUT (BUZZER)

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



**Example: Setup TC1OUT output from TC1 to TC1OUT (P5.3). The external high-speed clock is 4MHz. The TC1OUT frequency is 0.5KHz. Because the TC1OUT signal is divided by 2, set the TC1 clock to 1KHz. The TC1 clock source is from external oscillator clock. T0C rate is  $F_{cpu}/8$ . The  $TC1RATE2-TC1RATE1 = 101$ .  $TC1C = TC1R = 131$ .**

```

MOV      A,#01010000B
B0MOV    TC1M,A           ; Set the TC1 rate to  $F_{cpu}/4$ 

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

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

```

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

### 8.3.6 TC1 TIMER OPERATION SEQUENCE

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

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

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

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

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

**F Set TC1 timer clock source.**

*; Select TC1 internal / external clock source.*

```
B0BCLR    FTC1CKS    ; Select TC1 internal clock source.
```

*or*

```
B0BSET    FTC1CKS    ; Select TC1 external clock source.
```

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

```
B0BCLR    FALOAD1    ; Enable TC1 auto reload function.
```

*or*

```
B0BSET    FALOAD1    ; Disable TC1 auto reload function.
```

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

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

```
MOV       A,#7FH      ; TC1C and TC1R value is decided by TC1 mode.
B0MOV     TC1C,A      ; Set TC1C value.
B0MOV     TC1R,A      ; Set TC1R value under auto reload mode or PWM mode.
```

*; In PWM mode, set PWM cycle.*

```
B0BCLR    FALOAD1    ; ALOAD1, TC1OUT = 00, PWM cycle boundary is
B0BCLR    FTC1OUT    ; 0~255.
```

*or*

```
B0BCLR    FALOAD1    ; ALOAD1, TC1OUT = 01, PWM cycle boundary is
B0BSET    FTC1OUT    ; 0~63.
```

*or*

```
B0BSET    FALOAD1    ; ALOAD1, TC1OUT = 10, PWM cycle boundary is
B0BCLR    FTC1OUT    ; 0~31.
```

*or*

```
B0BSET    FALOAD1    ; ALOAD1, TC1OUT = 11, PWM cycle boundary is
B0BSET    FTC1OUT    ; 0~15.
```

**F Set TC1 timer function mode.**

```
B0BSET    FTC1IEN    ; Enable TC1 interrupt function.
```

*or*

```
B0BSET    FTC1OUT    ; Enable TC1OUT (Buzzer) function.
```

*or*

```
B0BSET    FPWM1OUT   ; Enable PWM function.
```

**F Enable TC1 timer.**

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

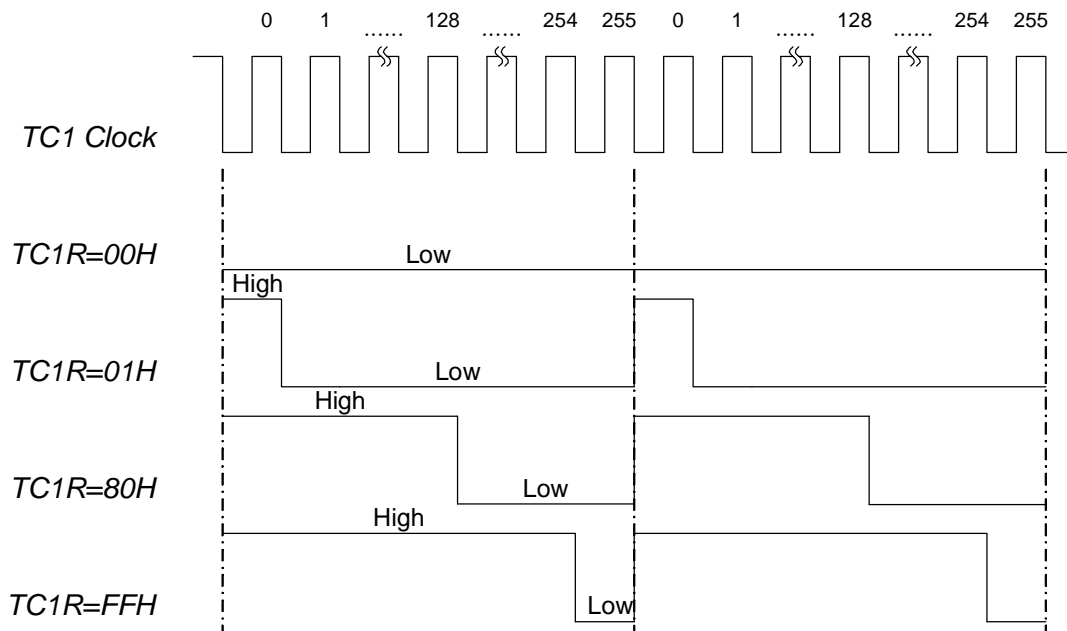
## 8.4 PWM1 MODE

### 8.4.1 OVERVIEW

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

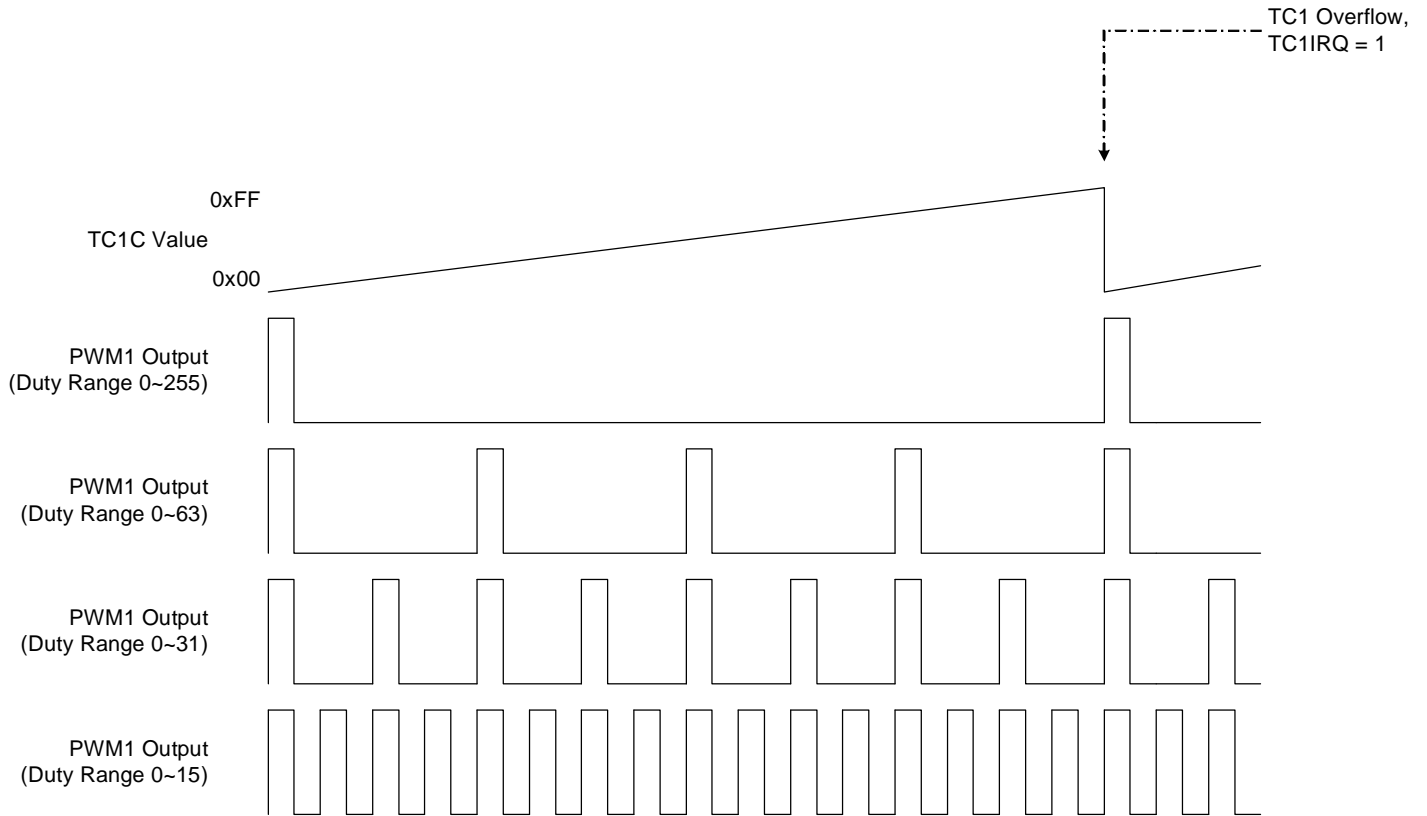
ALOAD1	TC1OUT	PWM duty range	TC1C valid value	TC1R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0	0	0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count
0	1	0/64~63/64	0x00~0x3F	0x00~0x3F	31.25K	Overflow per 64 count
1	0	0/32~31/32	0x00~0x1F	0x00~0x1F	62.5K	Overflow per 32 count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

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



### 8.4.2 TC1IRQ AND PWM DUTY

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



### 8.4.3 PWM PROGRAM EXAMPLE

**Example: Setup PWM1 output from TC1 to PWM1OUT (P5.3). 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. TC1 rate is  $F_{cpu}/4$ . The  $TC1RATE2-TC1RATE1 = 110$ .  $TC1C = TC1R = 30$ .**

```

MOV      A,#01100000B
B0MOV   TC1M,A           ; Set the TC1 rate to  $F_{cpu}/4$ 

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

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

```

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

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

```

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

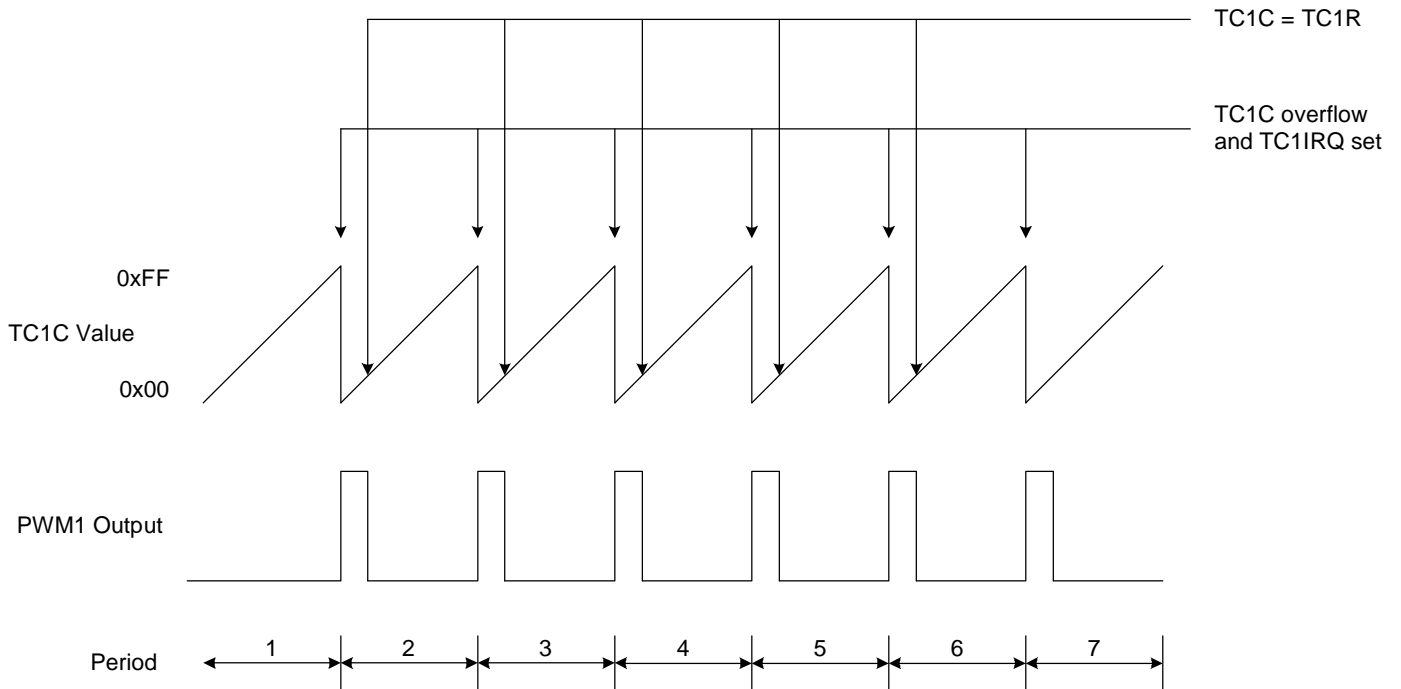
INCMS   BUF0              ; Get the new TC1R value from the BUF0 buffer defined by
NOP                                           ; programming.
B0MOV   A, BUF0
B0MOV   TC1R, A

```

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

### 8.4.4 PWM1 DUTY CHANGING NOTICE

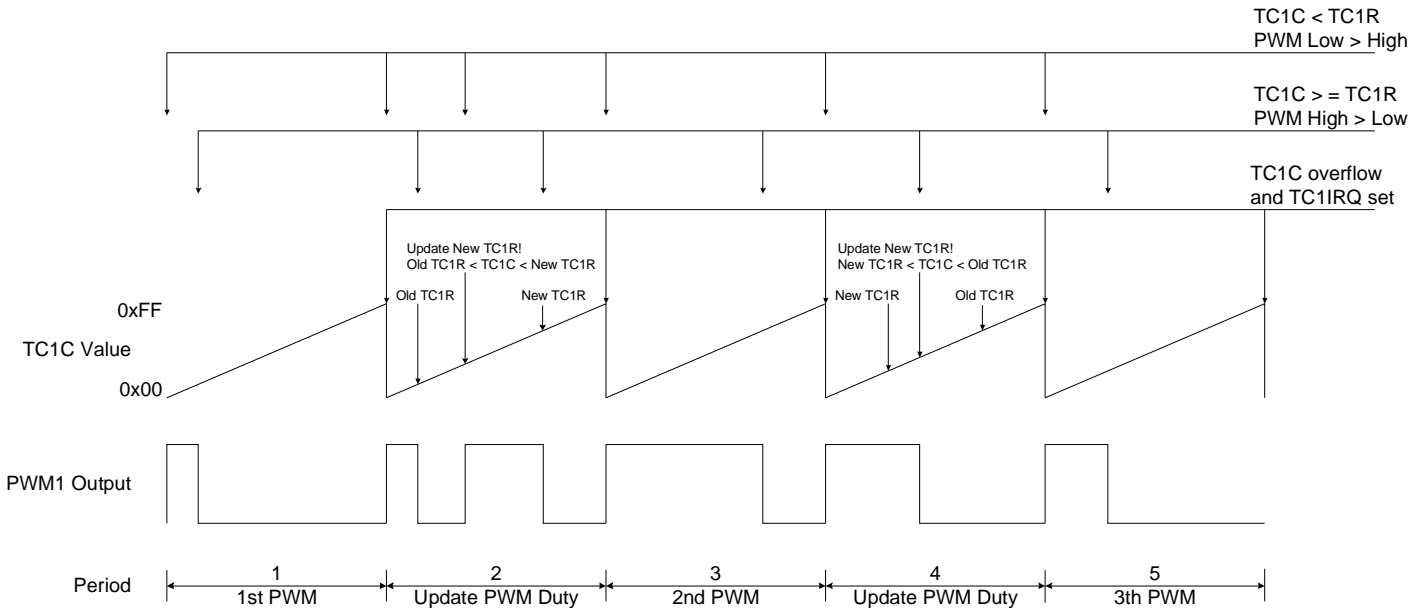
In PWM mode, the system will compare TC1C and TC1R all the time. When  $TC1C < TC1R$ , the PWM will output logic “High”, when  $TC1C \geq TC1R$ , the PWM will output logic “Low”. If TC1C is changed in certain period, the PWM duty will change immediately. If TC1R is fixed all the time, the PWM waveform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



Above diagram is shown the waveform with fixed TC1R. In every TC1C overflow PWM output “High, when  $TC1C \geq TC1R$  PWM output ”Low”.

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

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



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

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

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

MAIN:

```

...
B0MOV      TC1RBUF, A      ; Load new PWM duty setting value into TC1RBUF.
...

```

INT\_SER:

```

... ; Push routine to save ACC and PFLAG to buffers.
...
B0BTS1    FTC1IRQ
JMP      INT_SER90
B0MOV     A, TC1RBUF ; When TC1 Interrupt occurs, update TC1R.
B0MOV     TC1R, A
...

```

INT\_SER90:

```

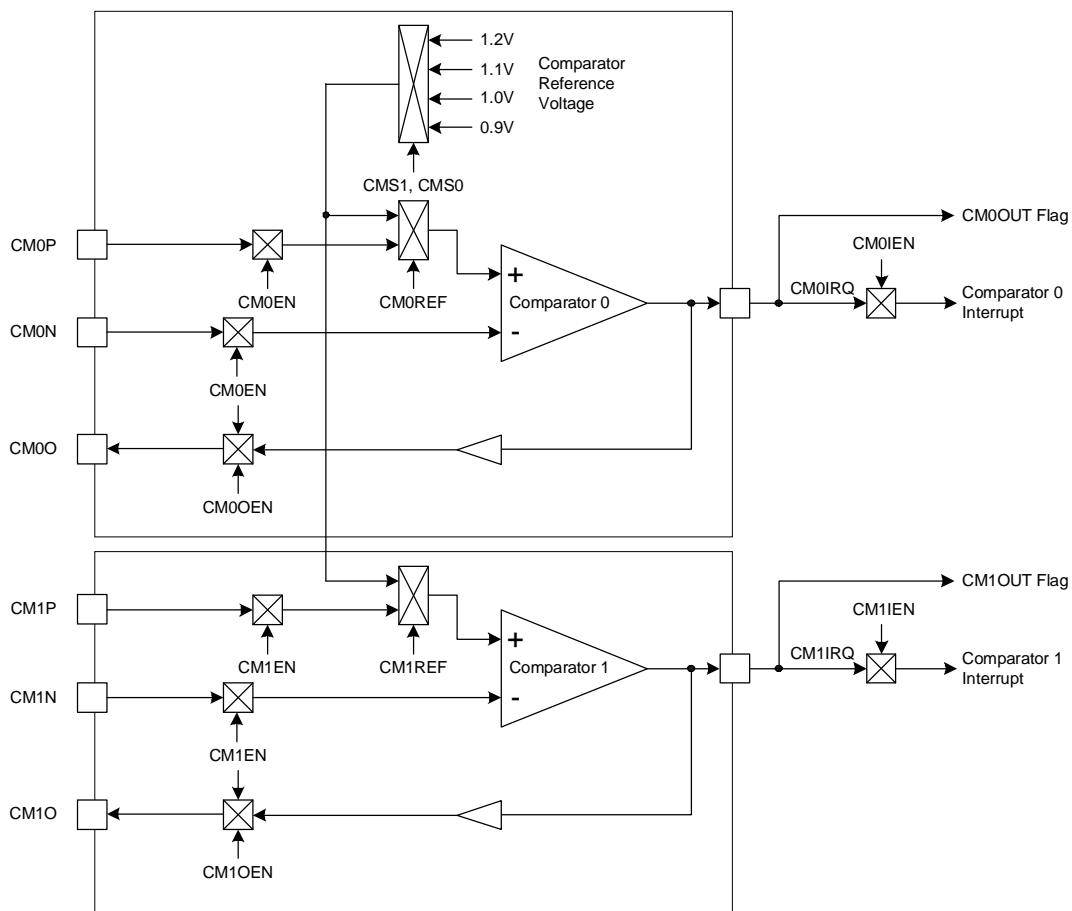
... ; Pop routine to load ACC and PFLAG from buffers.
RETI

```

# 9 ANALOG COMPARAOTR

## 9.1 OVERVIEW

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



### Comparator 0 Pin assignment:

CM0P: Comparator 0 positive input pin shared with P2.3. CM0P enables when CM0EN=1 and CM0REF=0.

CM0N: Comparator 0 negative input pin shared with P2.2. CM0N enables when CM0EN=1.

CM0O: Comparator 0 output pin shared with P2.4. CM0O enables when CM0EN=1 and CM0OEN = 1.

### Comparator 1 Pin assignment:

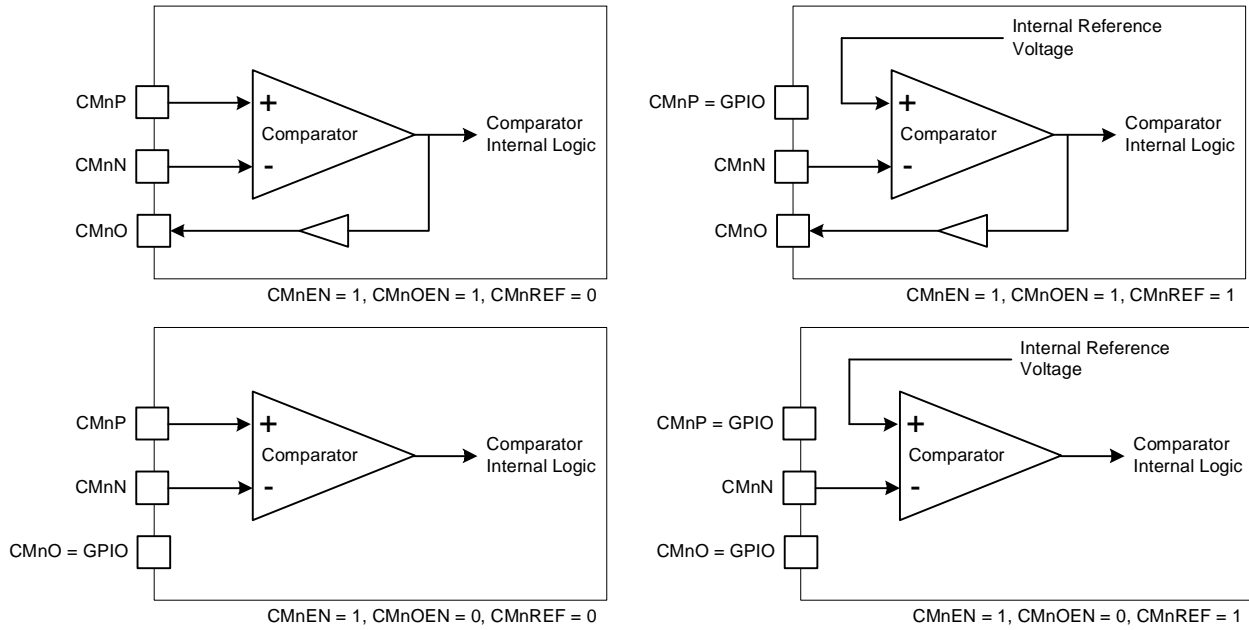
CM1P: Comparator 1 positive input pin shared with P2.6. CM1P enables when CM1EN=1 and CM1REF=0.

CM1N: Comparator 1 negative input pin shared with P2.5. CM1N enables when CM1EN=1.

CM1O: Comparator 1 output pin shared with P2.7. CM1O enables when CM1EN=1 and CM1OEN = 1.

**The comparator pins are GPIO mode except above conditions.**





- **Note:**

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

## 9.2 CMP0M REGISTER

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

- Bit 7     **CM0EN**: Comparator 0 control bit.  
0 = Disable. All comparator pins are GPIO.  
1 = Enable. CM0N pin is analog input pin.
- Bit 6     **CM0IEN**: Comparator 0 interrupt control bit.  
0 = Disable.  
1 = Enable.
- Bit 5     **CM0IRQ**: Comparator 0 interrupt request flag. CM0IRQ is latched to “1” as occurring comparator interrupt request, and it is cleared by program.  
0 = No comparator interrupt request.  
1 = Comparator interrupt request occurs when CM0P voltage or comparator 0 reference voltage is larger than CM0N voltage.
- Bit 4     **CM0OEN**: Comparator 0 output pin control bit.  
0 = Disable. CM0O pin is GPIO.  
1 = Enable. CM0O pin is comparator output pin.
- Bit 3     **CM0REF**: Comparator 0 internal reference voltage control bit.  
0 = Disable. CM0P pin is analog input pin.  
1 = Enable. CM0P pin is GPIO.
- Bit 2     **CM0OUT**: Comparator 0 raw output flag.  
0 = CM0P voltage or comparator 0 reference voltage is less than CM0N voltage.  
1 = CM0P voltage or comparator 0 reference voltage is larger than CM0N voltage.
- Bit[1:0]   **CMS[1:0]**: Comparator internal reference voltage select bit.  
00 = 0.9V, 01 = 1.0V, 10 = 1.1V, 11 = 1.2V

## 9.3 CMP1M REGISTER

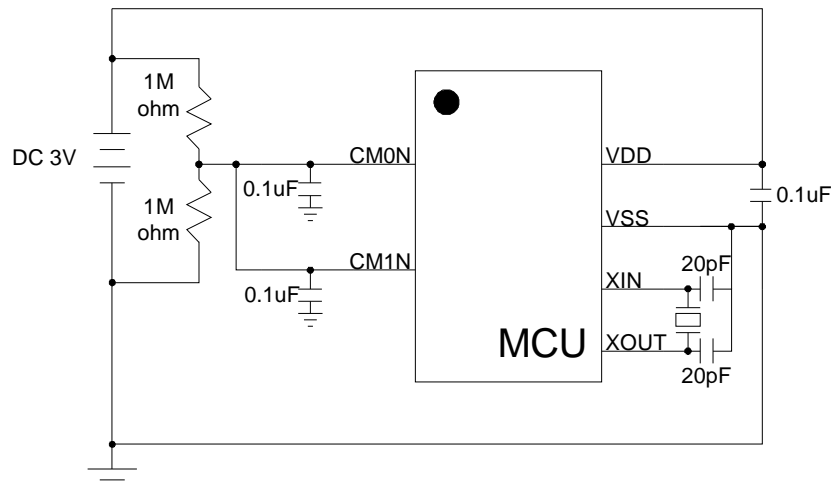
09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>CMP1M</b>	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	-	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R	-	-
After reset	0	0	0	0	0	0	-	-

- Bit 7     **CM1EN:** Comparator 1 control bit.  
0 = Disable. All comparator pins are GPIO.  
1 = Enable. CM1N pin is analog input pin.
- Bit 6     **CM1IEN:** Comparator 1 interrupt control bit.  
0 = Disable.  
1 = Enable.
- Bit 5     **CM1IRQ:** Comparator 1 interrupt request flag. CM1IRQ is latched to “1” as occurring comparator interrupt request, and it is cleared by program.  
0 = No comparator interrupt request.  
1 = Comparator interrupt request occurs when CM1P voltage or comparator 1 reference voltage is larger than CM1N voltage.
- Bit 4     **CM1OEN:** Comparator 1 output pin control bit.  
0 = Disable. CM1O pin is GPIO.  
1 = Enable. CM1O pin is comparator output pin.
- Bit 3     **CM1REF:** Comparator 1 internal reference voltage control bit.  
0 = Disable. CM1P pin is analog input pin.  
1 = Enable. CM1P pin is GPIO.
- Bit 2     **CM1OUT:** Comparator 1 raw output flag.  
0 = CM1P voltage or comparator 1 reference voltage is less than CM1N voltage.  
1 = CM1P voltage or comparator 1 reference voltage is larger than CM1N voltage.

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

## 9.4 ANALOG COMPARATOR APPLICATION

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



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

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

; The comparator initialize.

MOV	A, #00001010b	; Enable internal reference 1.1V.
B0MOV	CMP0M, A	; Disable comparator output pin.
B0MOV	CMP1M, A	; Disable comparator interrupt.
B0BSET	FCM0EN	; Enable comparator 0.
B0BSET	FCM1EN	; Enable comparator 1.

; Check 2.2V battery voltage.

CMP0\_CHK:

B0BSET	FCMS1	; Set internal reference voltage = 1.1V.
B0BCLR	FCMS0	
NOP		; Delay 2 instructions cycle to be the internal band-gap
NOP		; setup time.
B0BTS1	FCM0OUT	; Check comparator 0 status through CM0OUT flag.
JMP	Main	; Not low battery, return to main loop.
JMP	CMP1_CHK	; Is low battery status, go to check comparator 1.

; Check 2.0V battery voltage.

CMP1\_CHK:

B0BCLR	FCMS1	; Set internal reference voltage = 1.0V.
B0BSET	FCMS0	
NOP		; Delay 2 instructions cycle to be the internal band-gap
NOP		; setup time.
B0BTS1	FCM1OUT	; Check comparator 1 status through CM1OUT flag.

---

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

; Low battery process.  
LowBat:

```
...  
...  
JMP      Main          ; Return to main loop.
```

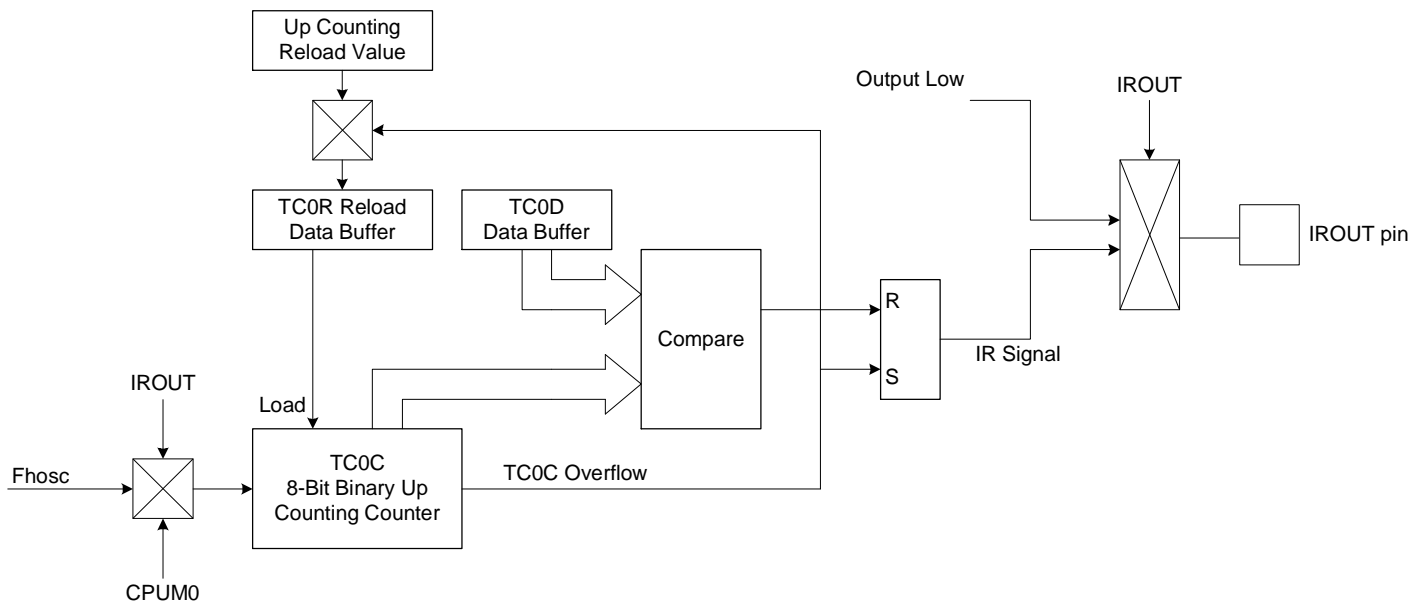
; No battery process.  
NoBat:

```
...  
...  
JMP      Main          ; Return to main loop.
```

# 10 IR OUTPUT

## 10.1 OVERVIEW

IR signal is generated by TC0 timer. The IR output pin is IROUT pin. When IROUT bit of TC0M is logic "1", IROUT pin outputs IR signal. If IROUT = 0 or system is in power down mode, IROUT pin is tied to low status. The TC0 is an 8-bit binary up counting timer for IR signal generator. The IR signal is duty/cycle changeable type controlled by TC0R and TC0D. TC0R decides IR cycle and TC0D decides IR duty. TC0 clock source is only from Fosc (external high clock source), eg. 4MHz crystal. If external oscillator is 4MHz, the TC0 clock rate is 4MHz. TC0 only generate IR output and no interrupt function. When enable IR output function (IROUT=1), IR output status is high level. TC0C initial value is TC0R and starts to count. When TC0C=TC0D, IR output status change to low level and finishes high duty operation. When TC0C overflow occurs (TC0C changes from 0xFF to 0x00), IR output low duty operation stops. System loads TC0R into TC0C automatically and next cycle starts.



## 10.2 IR CONTROL REGISTER

### 10.2.1 TC0M MODE REGISTER

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0M</b>	-	-	-	-	-	-	IREN	CREN
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

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

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

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

### 10.2.2 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
<b>TC0C</b>	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

- **Note:** Set TC0C=TC0R before IR output enable to make sure the first cycle correct.

### 10.2.3 TC0R AUTO-LOAD REGISTER

**TC0R decides IR signal frequency.** TC0 timer is with auto-load function. When TC0C overflow occurs, TC0R value will load to TC0C. It is easy to generate an accurate time for IR signal cycle, 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 1<sup>st</sup> buffer. Until TC0 overflow occurs, the new value moves to real TC0R buffer.

OCDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0R</b>	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.

$$TC0R \text{ initial value} = 256 - (TC0 \text{ interrupt interval time} * \text{input clock})$$

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

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

$$TC0R \text{ initial value} = 256 - (TC0 \text{ interrupt interval time} * \text{input clock})$$

*TC0 interval time = 1/38KHz = 26.3us*  
*Input clock = external oscillator 4MHz.*

$$\begin{aligned}
 TC0R &= 256 - (26.3us * 4MHz) \\
 &= 150.8 \\
 &\approx 151 \\
 &= 97h
 \end{aligned}$$



## 10.2.4 TC0D IR DUTY CONTROL REGISTER

The IR signal is duty changeable by TC0D. TC0D decides the IR output signal high pulse width length. When TC0C=TC0D, the IR signal changes from high pulse to low pulse. The low pulse stops when TC0C overflow. The high pulse width is TC0D-TC0R, and the low pulse width is 256-TC0D. It is easy to modulate IR duty/cycle by TC0R and TC0D registers.

0E8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0D</b>	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of TC0D initial value is as following.

$$TC0D \text{ initial value} = TC0R + (256-TC0R) / (1/IR \text{ duty})$$

**Example: Set TC0D for 38KHz IR and duty is 1/3. Input clock is 4MHz.**

$$TC0D \text{ initial value} = TC0R + (256-TC0R) / (1/IR \text{ duty})$$

TC0R of 38KHz = 151

$$\begin{aligned}
 TC0D &= 151 + (256-151)/(1/(1/3)) \\
 &= 186 \\
 &= \mathbf{BAh}
 \end{aligned}$$

**Common IR signal table. System clock is 4MHz.**

IR Freq. (KHz)	TC0C TC0D		TC0D						Freq. Error Rate
			1/2 duty		1/3 duty		1/4duty		
	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	
32	131	83	193.50	C1	172.67	AC	162.25	A2	0.00%
36	145	91	200.50	C8	182.00	B6	172.75	AC	0.10%
38	151	97	203.50	CB	186.00	BA	177.25	B1	0.25%
39.2	154	9A	205.00	CD	188.00	BC	179.50	B3	0.04%
40	156	9C	206.00	CE	189.33	BD	181.00	B5	0.00%
56	185	B9	220.50	DC	208.67	D0	202.75	CA	0.60%

## 10.2.5 IR OUTPUT OPERATION SEQUENCE

### F Set TC0C and TC0R for IR cycle.

```
MOV    A, #IRCYCVAL    ;TC0C, TC0R value for IR cycle.
MOV    TC0C, A
MOV    TC0R, A
```

### F Set TC0D for IR duty.

```
MOV    A, #IRDUTYVAL  ;TC0D value for IR duty.
MOV    TC0D, A
```

### F Enable IR output.

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

# 11 INSTRUCTION TABLE

Field	Mnemonic	Description	C	DC	Z	Cycle
MOV	MOV A,M	$A \leftarrow M$	-	-	√	1
	MOV M,A	$M \leftarrow A$	-	-	-	1
	B0MOV A,M	$A \leftarrow M$ (bank 0)	-	-	√	1
	B0MOV M,A	$M$ (bank 0) $\leftarrow A$	-	-	-	1
	MOV A,I	$A \leftarrow I$	-	-	-	1
	B0MOV M,I	$M \leftarrow I$ , "M" only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z...)	-	-	-	1
	XCH A,M	$A \leftrightarrow M$	-	-	-	1+N
	B0XCH A,M	$A \leftrightarrow M$ (bank 0)	-	-	-	1+N
	MOVC	$R, A \leftarrow ROM [Y,Z]$	-	-	-	2
ARITH	ADC A,M	$A \leftarrow A + M + C$ , if occur carry, then C=1, else C=0	√	√	√	1
	ADC M,A	$M \leftarrow A + M + C$ , if occur carry, then C=1, else C=0	√	√	√	1+N
	ADD A,M	$A \leftarrow A + M$ , if occur carry, then C=1, else C=0	√	√	√	1
	ADD M,A	$M \leftarrow A + M$ , if occur carry, then C=1, else C=0	√	√	√	1+N
	B0ADD M,A	$M$ (bank 0) $\leftarrow M$ (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
	ADD A,I	$A \leftarrow A + I$ , if occur carry, then C=1, else C=0	√	√	√	1
	SBC A,M	$A \leftarrow A - M - /C$ , if occur borrow, then C=0, else C=1	√	√	√	1
	SBC M,A	$M \leftarrow A - M - /C$ , if occur borrow, then C=0, else C=1	√	√	√	1+N
	SUB A,M	$A \leftarrow A - M$ , if occur borrow, then C=0, else C=1	√	√	√	1
	SUB M,A	$M \leftarrow A - M$ , if occur borrow, then C=0, else C=1	√	√	√	1+N
	SUB A,I	$A \leftarrow A - I$ , if occur borrow, then C=0, else C=1	√	√	√	1
	LOGIC	AND A,M	$A \leftarrow A$ and M	-	-	√
AND M,A		$M \leftarrow A$ and M	-	-	√	1+N
AND A,I		$A \leftarrow A$ and I	-	-	√	1
OR A,M		$A \leftarrow A$ or M	-	-	√	1
OR M,A		$M \leftarrow A$ or M	-	-	√	1+N
OR A,I		$A \leftarrow A$ or I	-	-	√	1
XOR A,M		$A \leftarrow A$ xor M	-	-	√	1
XOR M,A		$M \leftarrow A$ xor M	-	-	√	1+N
XOR A,I		$A \leftarrow A$ xor I	-	-	√	1
SHIFT	SWAP M	$A (b3-b0, b7-b4) \leftarrow M(b7-b4, b3-b0)$	-	-	-	1
	SWAPM M	$M(b3-b0, b7-b4) \leftarrow M(b7-b4, b3-b0)$	-	-	-	1+N
	RRC M	$A \leftarrow RRC M$	√	-	-	1
	RRCM M	$M \leftarrow RRC M$	√	-	-	1+N
	RLC M	$A \leftarrow RLC M$	√	-	-	1
	RLCM M	$M \leftarrow RLC M$	√	-	-	1+N
	CLR M	$M \leftarrow 0$	-	-	-	1
	BCLR M.b	$M.b \leftarrow 0$	-	-	-	1+N
	BSET M.b	$M.b \leftarrow 1$	-	-	-	1+N
	B0BCLR M.b	$M$ (bank 0).b $\leftarrow 0$	-	-	-	1+N
B0BSET M.b	$M$ (bank 0).b $\leftarrow 1$	-	-	-	1+N	
BRANCH	CMPRS A,I	ZF,C $\leftarrow A - I$ , If A = I, then skip next instruction	√	-	√	1 + S
	CMPRS A,M	ZF,C $\leftarrow A - M$ , If A = M, then skip next instruction	√	-	√	1 + S
	INCS M	$A \leftarrow M + 1$ , If A = 0, then skip next instruction	-	-	-	1 + S
	INCMS M	$M \leftarrow M + 1$ , If M = 0, then skip next instruction	-	-	-	1+N+S
	DECS M	$A \leftarrow M - 1$ , If A = 0, then skip next instruction	-	-	-	1 + S
	DECMS M	$M \leftarrow M - 1$ , If M = 0, then skip next instruction	-	-	-	1+N+S
	BTS0 M.b	If M.b = 0, then skip next instruction	-	-	-	1 + S
	BTS1 M.b	If M.b = 1, then skip next instruction	-	-	-	1 + S
	B0BTS0 M.b	If M(bank 0).b = 0, then skip next instruction	-	-	-	1 + S
	B0BTS1 M.b	If M(bank 0).b = 1, then skip next instruction	-	-	-	1 + S
	JMP d	$PC_{15/14} \leftarrow RomPages1/0, PC_{13-PC0} \leftarrow d$	-	-	-	2
	CALL d	$Stack \leftarrow PC_{15-PC0}, PC_{15/14} \leftarrow RomPages1/0, PC_{13-PC0} \leftarrow d$	-	-	-	2
MISC	RET	$PC \leftarrow Stack$	-	-	-	2
	NOP	No operation	-	-	-	1

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

# 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)	
SN8P26L34K, SN8P2L634S, SN8P26L34X .....	0°C ~ + 70°C
SN8P26L32P, SN8P26L32S, .....	0°C ~ + 70°C
SN8P26L321P, SN8P2L6321S, SN8P26L321X.....	0°C ~ + 70°C
SN8P26L31S, .....	0°C ~ + 70°C
Storage ambient temperature (Tstor) .....	-40°C ~ + 125°C

## 12.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 3.0V, 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, Fcpu = 2mips.	1.8	3.0	3.6	V	
RAM Data Retention voltage	Vdr		1.5	-	-	V	
Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset	0.05	-	-	V/ms	
Input Low Voltage	ViL1	All input ports	Vss	-	0.3Vdd	V	
	ViL2	Reset pin	Vss	-	0.2Vdd	V	
Input High Voltage	ViH1	All input ports	0.7Vdd	-	Vdd	V	
	ViH2	Reset pin	0.9Vdd	-	Vdd	V	
Reset pin leakage current	Ilekg	Vin = Vdd, 25°C	-	-	2	uA	
I/O port pull-up resistor	Rup	Vin = Vss, Vdd = 3V	100	200	300	KΩ	
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd	-	-	2	uA	
I/O output source current	IoH	Vop = Vdd – 0.5V	8	10	-	mA	
	IoL	Vop = Vss + 0.5V	8	12	-		
INTn trigger pulse width	Tint0	INT0 interrupt request pulse width	2/fcpu	-	-	cycle	
Supply Current	Idd1	Run Mode (No loading, Fcpu = Fosc/4)	Vdd= 3V, 4Mhz	-	1	2	mA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd=3V, ILRC 10Khz	-	5	10	uA
	Idd3	Sleep Mode	Vdd= 3V, 25°C	-	1	2	uA
	Idd4	Green Mode (No loading, Fcpu = Fosc/4, Watchdog Disable)	Vdd= 3V, 4Mhz	-	0.25	0.5	mA
Vdd=3V, ILRC 10Khz			-	3	6	uA	
Internal High Oscillator Freq.	Fihrc	Internal High RC (IHRC)	25°C, Vdd= 3V, Fcpu = 1MHz	7.84	8	8.16	Mhz
Band-gap Output Voltage	Vbd1	CMS0, CMS1 = 00	-	0.9	-	V	
	Vbd2	CMS0, CMS1 = 01	-	1.0	-		
	Vbd3	CMS0, CMS1 = 10	-	1.1	-		
	Vbd4	CMS0, CMS1 = 11	-	1.2	-		
Comparator Current	Icm1	Fcpu = 1MHz, Vdd=3V. Disable internal reference.	-	50	-	uA	
	Icm2	Fcpu = 1MHz, Vdd=3V. Enable internal reference.	-	150	-		
Comparator Input Offset Voltage	Vcmoff	Fcpu = 1MHz, Vdd=3V,	-	±5	-	mV	
LVD Voltage	Vdet0	Low voltage reset level.	-	1.7	-	V	
	Vdet1	Low voltage reset level. Fcpu = 1 MHz.	-	2.4	-	V	
		Low voltage indicator level. Fcpu = 1 MHz.	-	2.8	-		
Vdet2	Low voltage indicator level. Fcpu = 1 MHz	-	2.8	-	V		

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

# 13 DEVELOPMENT TOOL

SONiX provides ICE (in circuit emulation), IDE (Integrated Development Environment) and EV-kit for SN8P26L34/32/321 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.

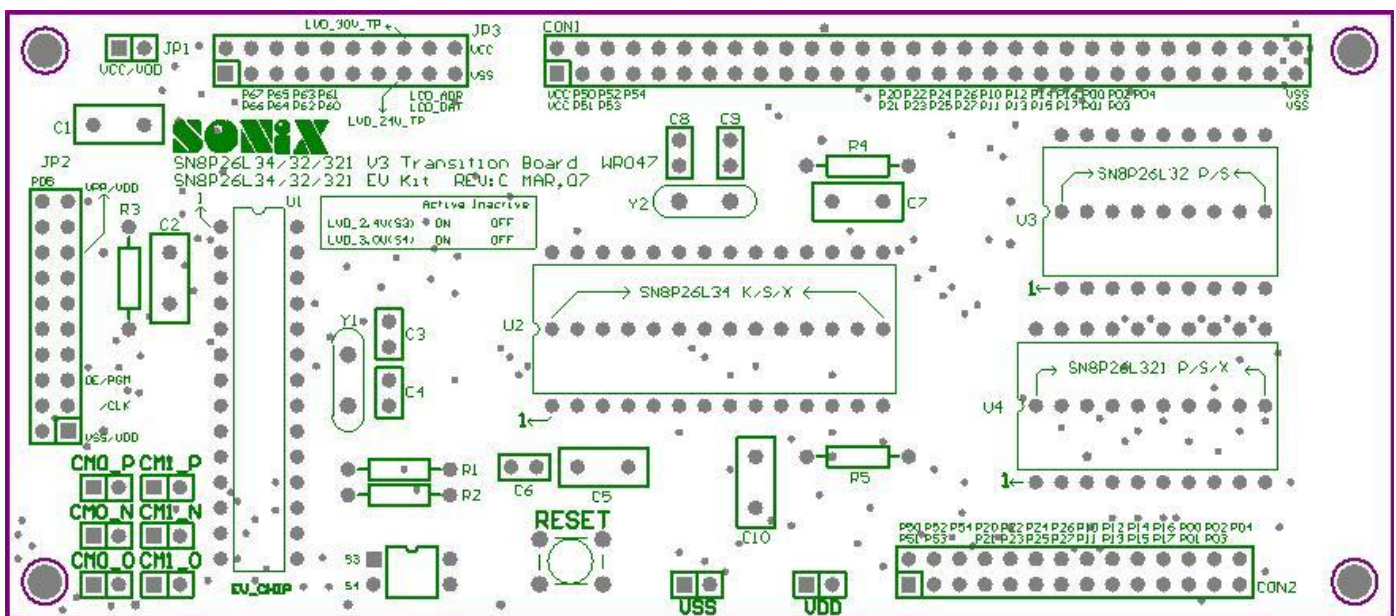
- I ICE: SN8ICE2K
- I EV-kit: SN8P26L34/32 EV-kit Rev. A. (Only support SN8P26L34)  
SN8P26L34/32/321 EV-kit Rev. B. (Support SN8P26L34/321)  
SN8P26L34/32/321 EV-kit Rev. C. (Support SN8P26L34/32/321)
- I IDE: SONiX IDE M2IDE\_V115.
- I Writer: MPIII WRITE-LV.

## 13.1 SN8P26L34/32/321 EV-kit

SN8P26L34/32/321 EV-kit includes ICE interface, GPIO interface and EV-chip module.

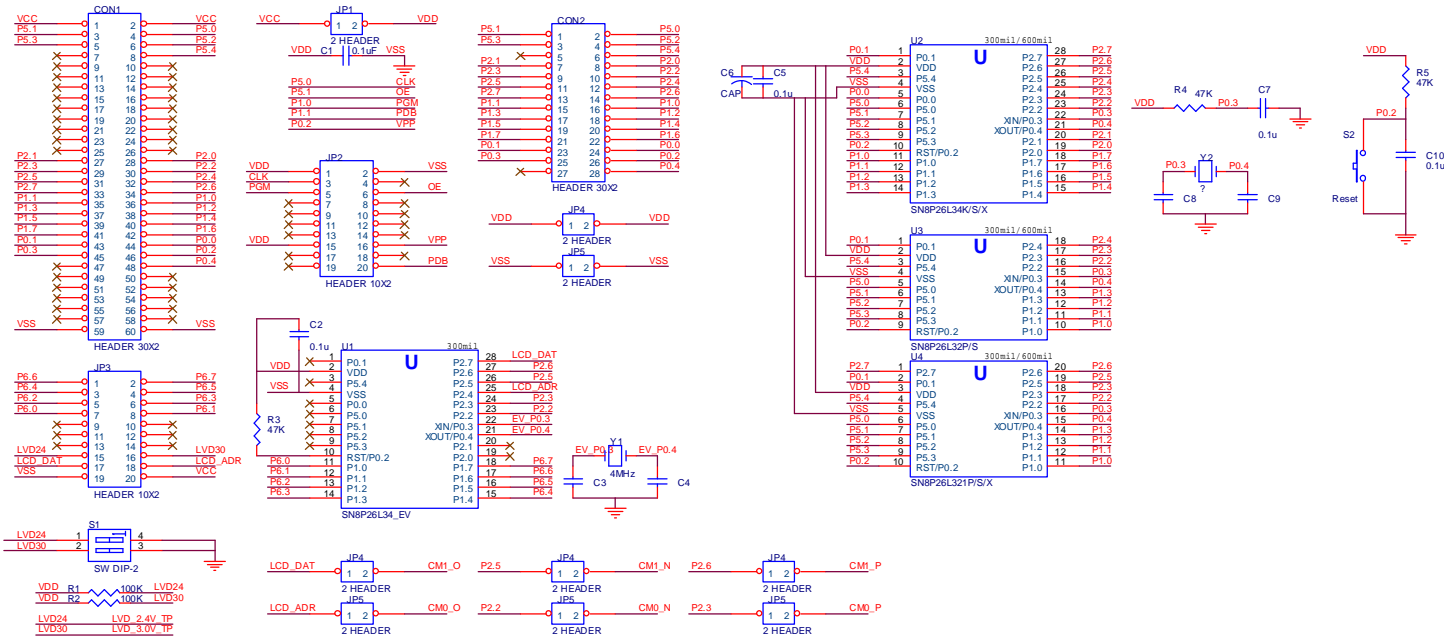
- I EV-chip module: .Emulate comparator function.

The schematic of SN8P26L34/32/321 EV-kit is as following.



- I CON1, JP3: ICE interface connected to SN8ICE2K .
- I CON2: GPIO connector.
- I JP2: Writer interface connected to MPIII writer.
- I U1: SN8P26L34 EV-chip for comparator emulation.
- I U2: SN8P26L34 DIP form connector for connecting to user's target board.
- I U3: SN8P26L32 DIP form connector for connecting to user's target board.
- I U4: SN8P26L321 DIP form connector for connecting to user's target board.
- I CM0\_P: Comparator 0 positive input pin.
- I CM0\_N: Comparator 0 negative input pin.
- I CM0\_O: Comparator 0 output pin.
- I CM1\_P: Comparator 1 positive input pin.
- I CM1\_N: Comparator 1 negative input pin.
- I CM1\_O: Comparator 1 output pin.
- I S1: LVD24 and LVD28 emulating switch.

**SN8P26L34/32/321 EV-kit Circuit**



SONiX  
SN8P26L34/32/321 V3 Transition Board WRxxxx  
SN8P26L34/32/321 EV Kit REV:C

**13.2 ICE and EV-KIT APPLICATION NOTIC**

SN8P26L34/32/321 EV-kit includes comparator emulation module. There is a SN8P26L34P chip programmed emulating code to emulate comparator function. The SN8P26L34/32/321 comparator pins are shared with P2 GPIO pins. In ICE environment, the comparator pins isn't connected with GPIO pin.

- I The Comparator emulation is from the SN8P26L34 EV-chip of SN8P26L34/32/321 EV-kit. For comparator emulation, input and output comparator signals from these pins.
- I The P2 comparator shared pin GPIO emulation is from P2 pins of SN8P26L34/32/321 EV-kit.
- I The SN8P26L34/32/321 EV-kit power level must be external 3V. Don't using ICE internal\_5V power. Disconnect internal\_5V pin of SN8ICE2K ICE and supply 3V power from external power source.

# 14 OTP PROGRAMMING PIN

## 14.1 The pin assignment of Easy Writer transition board socket:

### Easy Writer JP1/JP2

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

JP1 for MP transition board

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

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

JP3 for MP transition board

## 14.2 Programming Pin Mapping:

Programming Information of SN8P26L34/26L32/26L321									
Chip Name		SN8P26L34KSX	SN8P26L32PS	SN8P26L321P/S/X	SN8P26L321J				
EZ Writer Connector		OTP IC / JP3 Pin Assignment							
Number	Name	Number	Pin	Number	Pin	Number	Pin	Number	Pin
1	VDD	2	VDD	2	VDD	3	VDD	3	VDD
2	GND	4	VSS	4	VSS	5	VSS	5	VSS
3	CLK	6	P5.0	5	P5.0	6	P5.0	6	P5.0
4	CE	-	-	-	-	-	-	-	-
5	PGM	11	P1.0	10	P1.0	11	P1.0	11	P1.0
6	OE	7	P5.1	6	P5.1	7	P5.1	7	P5.1
7	D1	-	-	-	-	-	-	-	-
8	D0	-	-	-	-	-	-	-	-
9	D3	-	-	-	-	-	-	-	-
10	D2	-	-	-	-	-	-	-	-
11	D5	-	-	-	-	-	-	-	-
12	D4	-	-	-	-	-	-	-	-
13	D7	-	-	-	-	-	-	-	-
14	D6	-	-	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-	-	-
16	VPP	10	RST	9	RST	10	RST	10	RST
17	HLS	-	-	-	-	-	-	-	-
18	RST	-	-	-	-	-	-	-	-
19	-	-	-	-	-	-	-	-	-
20	ALSB/PDB	12, 8	P1.1, P5.2	11, 7	P1.1, P5.2	12, 8	P1.1, P5.2	12, 8	P1.1, P5.2

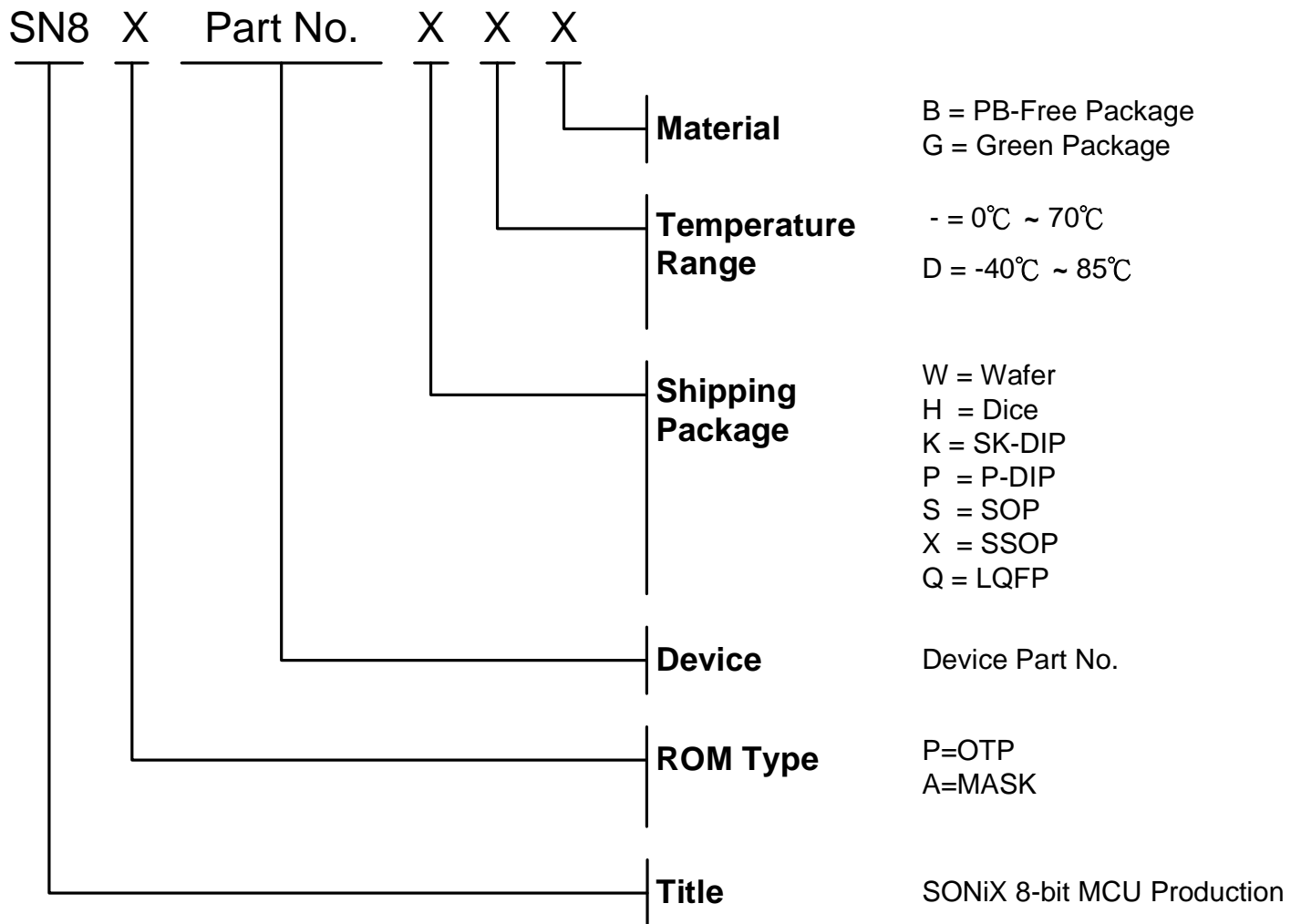


# 15 Marking Definition

## 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 INDETIFICATION SYSTEM

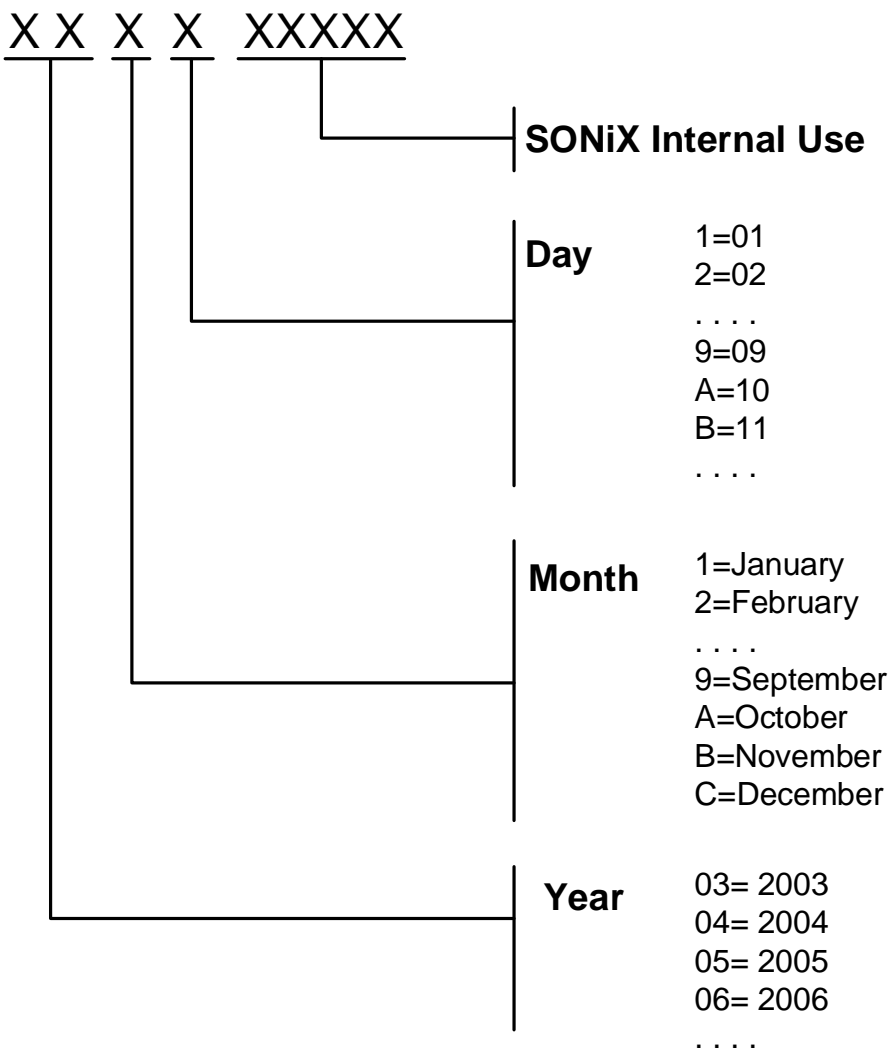


- *Note: SN8P26L34/26L32/26L321 doesn't support -40°C~85°C temperature range, LQFP package type and MASK ROM type.*

### 15.3 MARKING EXAMPLE

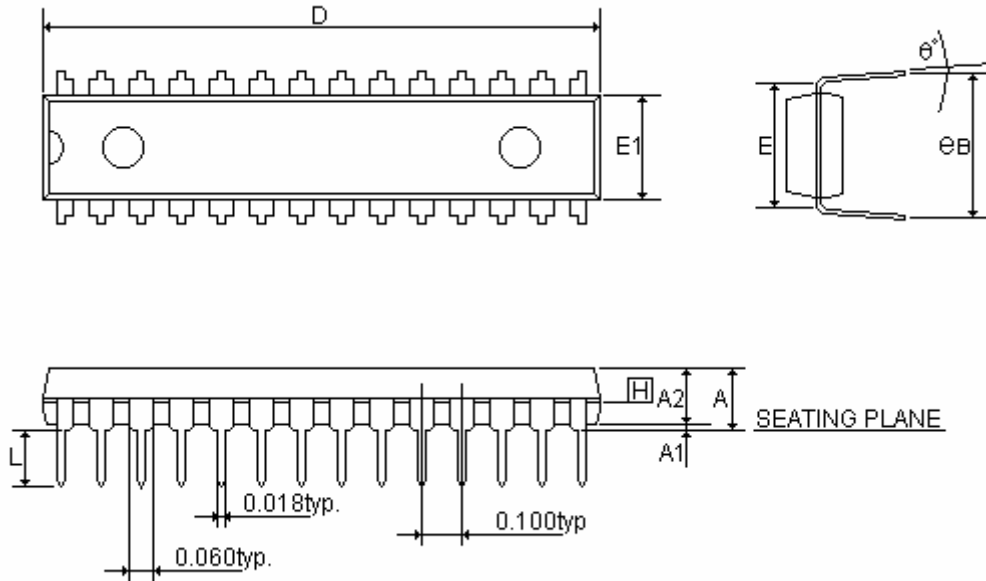
Name	ROM Type	Device	Package	Temperature	Material
SN8P26L34KB	OTP	26L34	SK-DIP	0°C~70°C	PB-Free Package
SN8P26L34SB	OTP	26L34	SOP	0°C~70°C	PB-Free Package
SN8P26L34XB	OTP	26L34	SSOP	0°C~70°C	PB-Free Package
SN8P26L34KG	OTP	26L34	SK-DIP	0°C~70°C	Green Package
SN8P26L34SG	OTP	26L34	SOP	0°C~70°C	Green Package
SN8P26L34XG	OTP	26L34	SSOP	0°C~70°C	Green Package
SN8P26L32PG	OTP	26L32	P-DIP	0°C~70°C	Green Package
SN8P26L32SG	OTP	26L32	SOP	0°C~70°C	Green Package
SN8P26L321PG	OTP	26L321	P-DIP	0°C~70°C	Green Package
SN8P26L321SG	OTP	26L321	SOP	0°C~70°C	Green Package
SN8P26L321XG	OTP	26L321	SSOP	0°C~70°C	Green Package
SN8P26L321JG	OTP	26L321	TQFN	0°C~70°C	Green Package
SN8P26L31SG	OTP	26L31	SOP	0°C~70°C	Green Package
SN8P26L34W	OTP	26L34	Wafer	0°C~70°C	-
SN8P26L34H	OTP	26L34	Dice	0°C~70°C	-

### 15.4 DATECODE SYSTEM



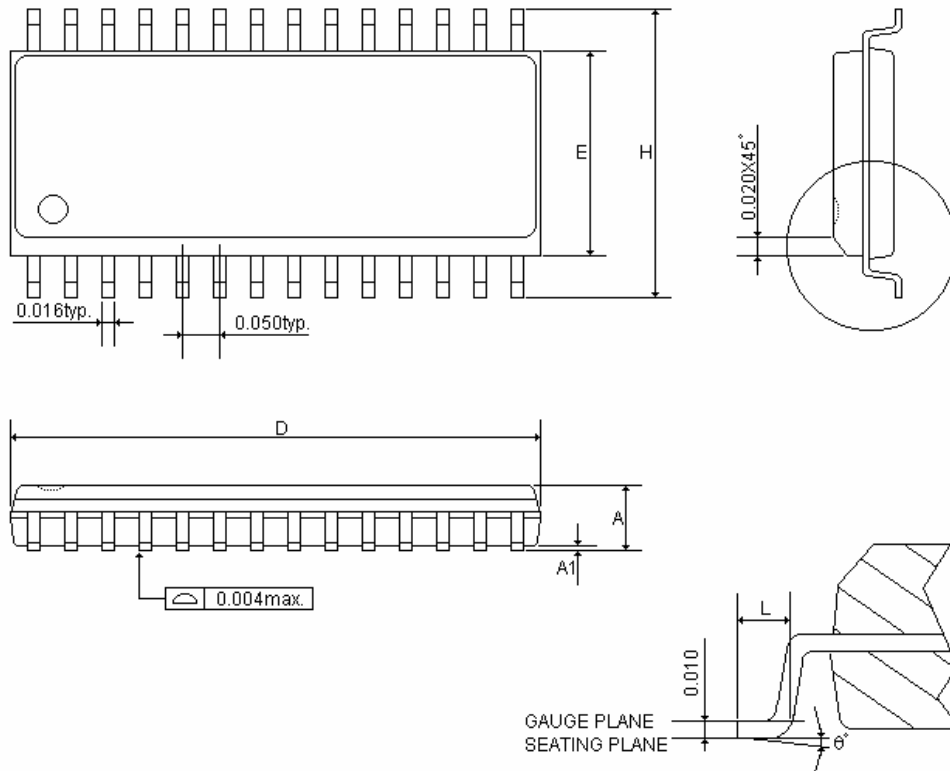
# 16 PACKAGE INFORMATION

## 16.1 SK-DIP 28 PIN



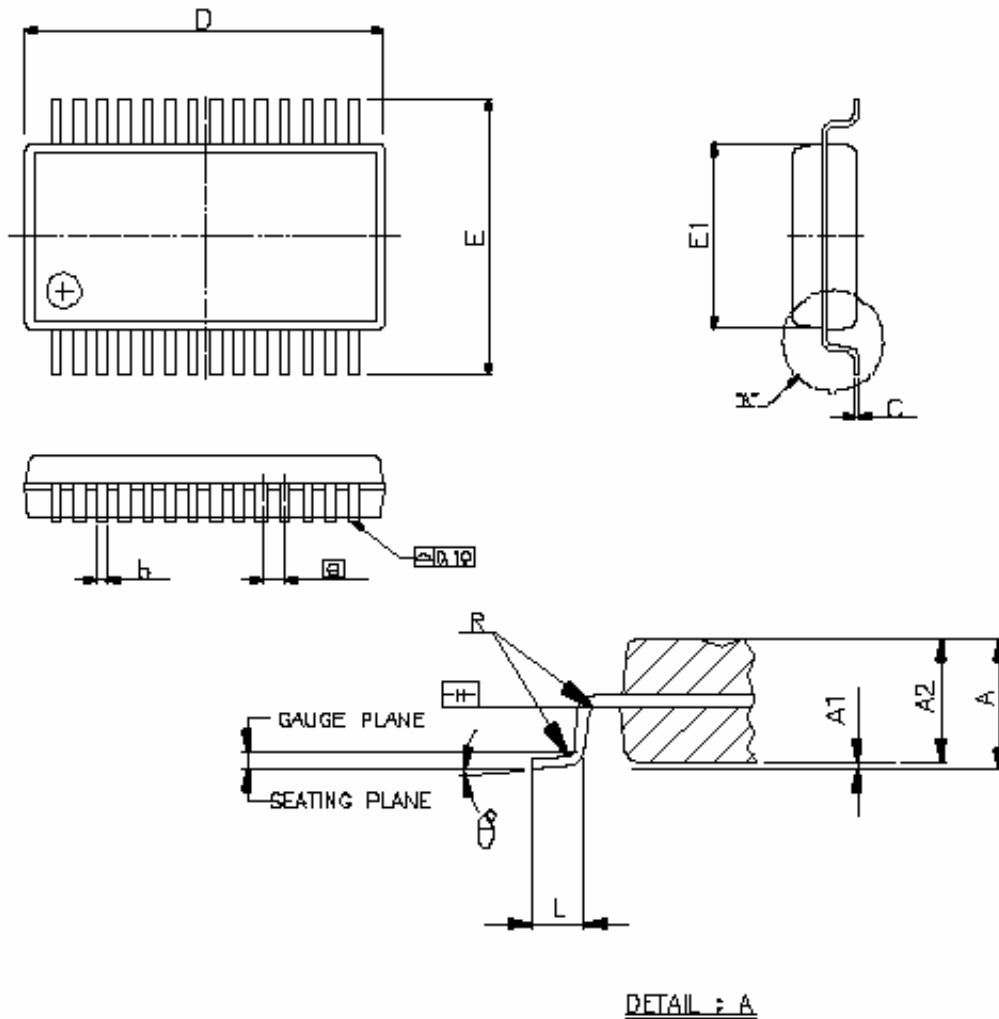
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.114	0.130	0.135	2.896	3.302	3.429
D	1.390	1.390	1.400	35.306	35.306	35.560
E	0.310			7.874		
E1	0.283	0.288	0.293	7.188	7.315	7.442
L	0.115	0.130	0.150	2.921	3.302	3.810
e B	0.330	0.350	0.370	8.382	8.890	9.398
$\theta^\circ$	0°	7°	15°	0°	7°	15°

## 16.2 SOP 28 PIN



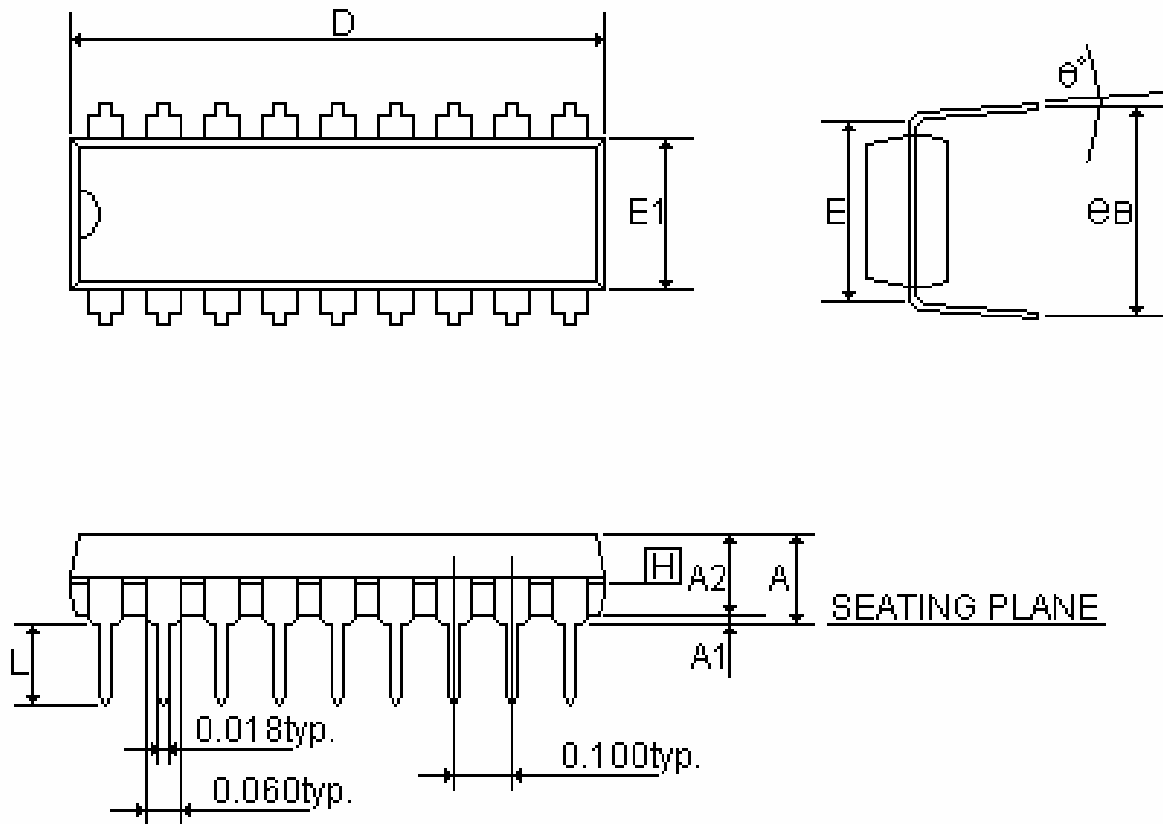
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.697	0.705	0.713	17.704	17.907	18.110
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
$\theta^\circ$	0°	4°	8°	0°	4°	8°

### 16.3 SSOP 28 PIN



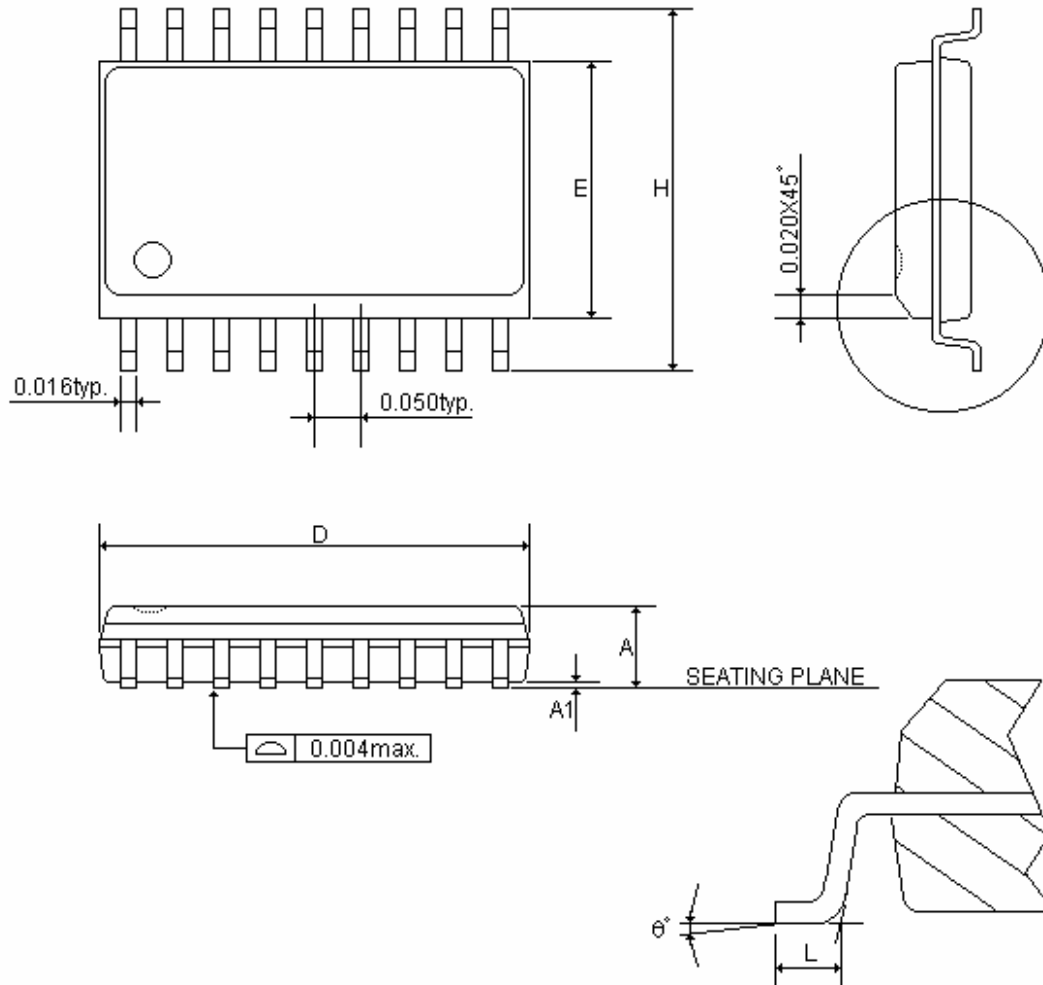
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.08	-	-	2.13
A1	0.00	-	0.01	0.05	-	0.25
A2	0.06	0.07	0.07	1.63	1.75	1.88
b	0.01	-	0.01	0.22	-	0.38
C	0.00	-	0.01	0.09	-	0.20
D	0.39	0.40	0.41	9.90	10.20	10.50
E	0.29	0.31	0.32	7.40	7.80	8.20
E1	0.20	0.21	0.22	5.00	5.30	5.60
[e]	0.0259BSC			0.65BSC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

## 16.4 P-DIP 18 PIN



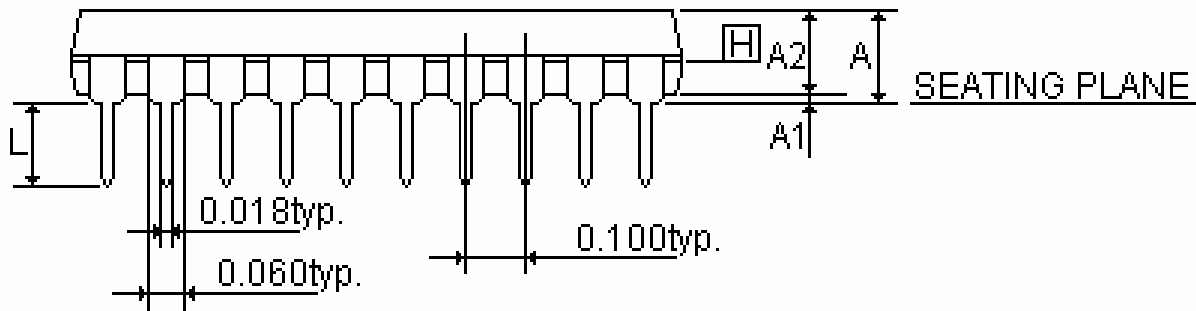
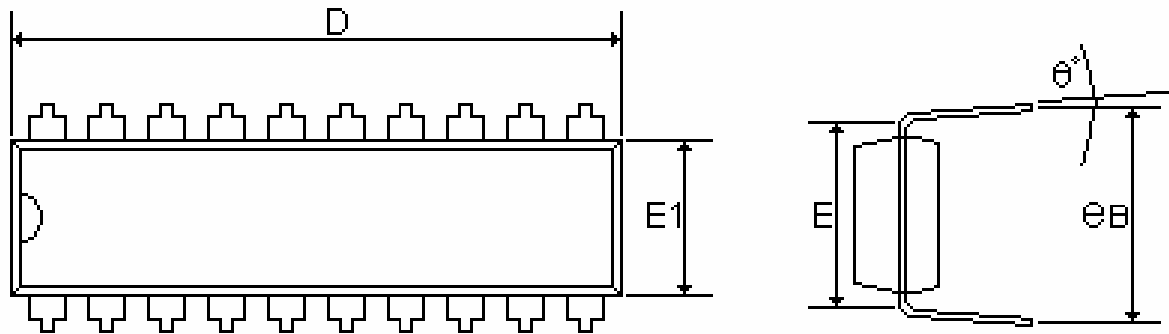
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.880	0.900	0.920	22.352	22.860	23.368
E	0.300			7.620		
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017	9.525
$\theta^\circ$	0°	7°	15°	0°	7°	15°

## 16.5 SOP 18 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	0.447	0.455	0.463	11.354	11.557	11.760
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
$\theta^\circ$	0°	4°	8°	0°	4°	8°

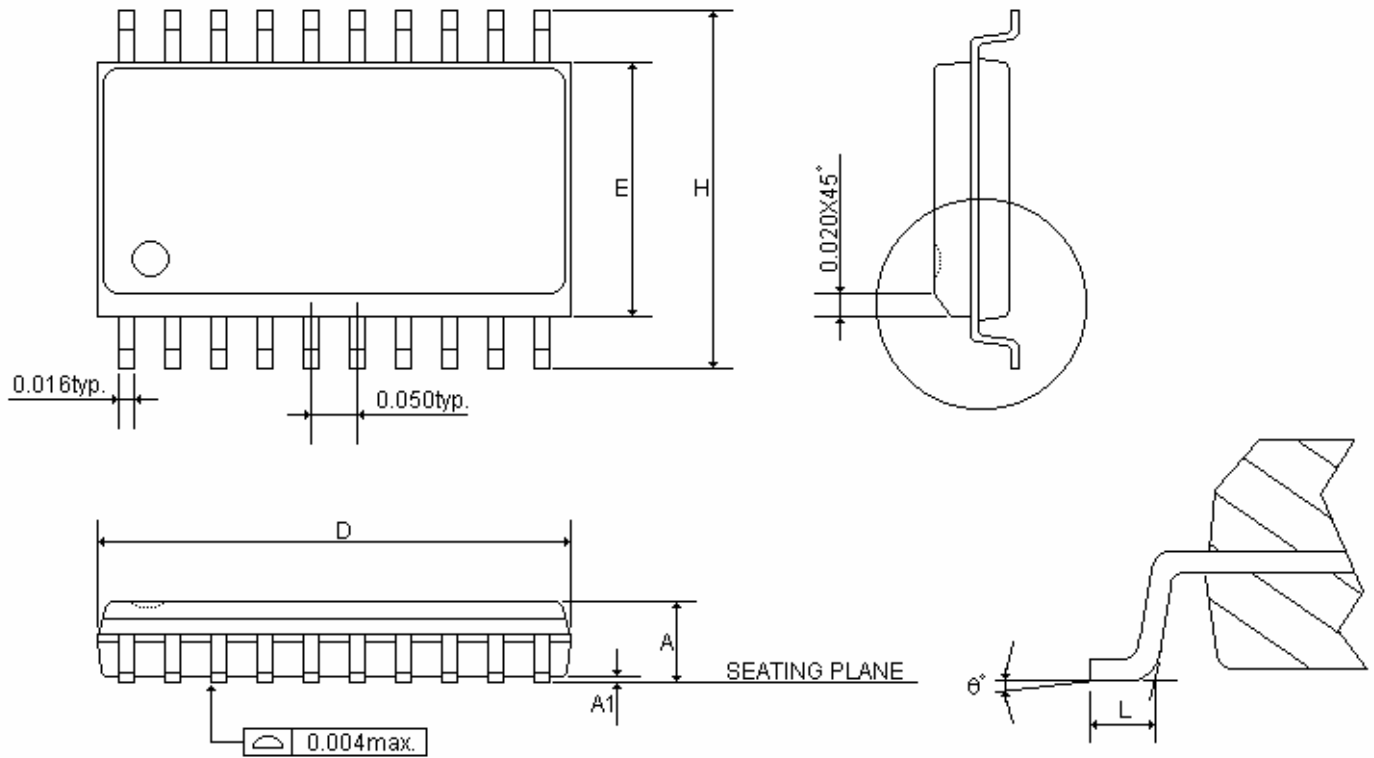
## 16.6 P-DIP 20 PIN



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
$\theta^\circ$	0°	7°	15°	0°	7°	15°

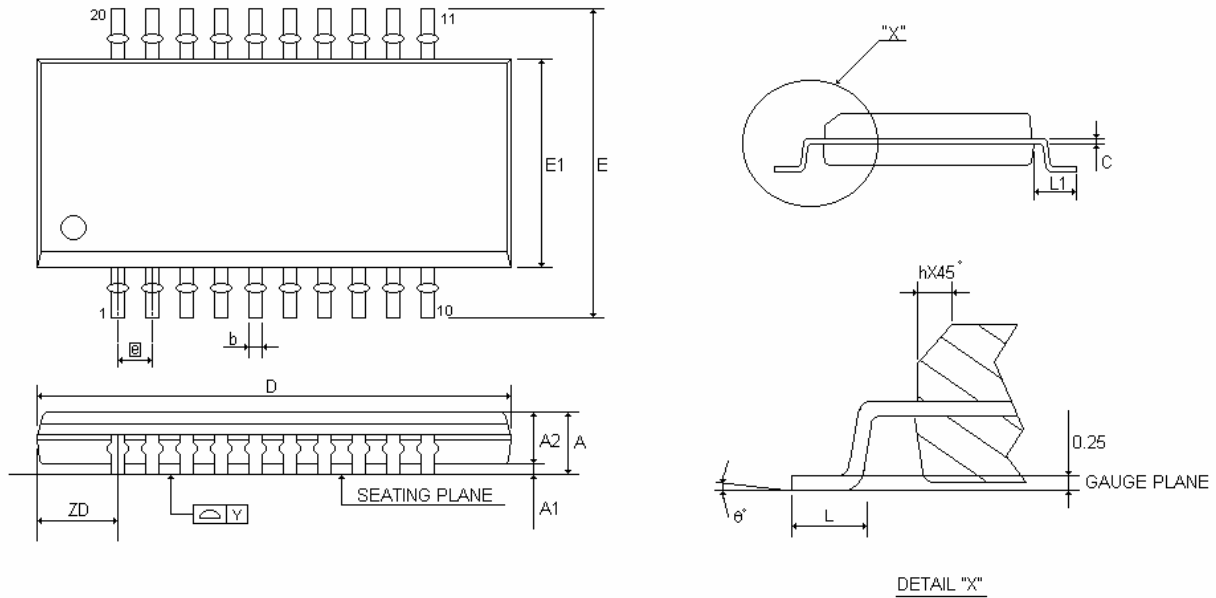


## 16.7 SOP 20 PIN



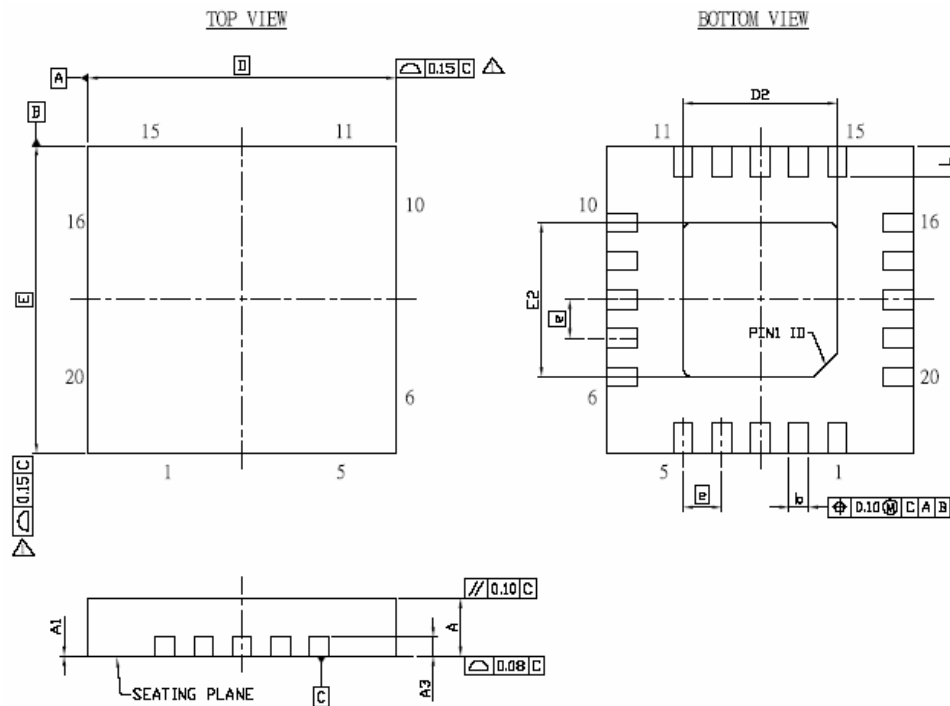
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.496	0.502	0.508	12.598	12.751	12.903
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
$\theta^\circ$	0°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

## 16.8 SSOP 20 PIN



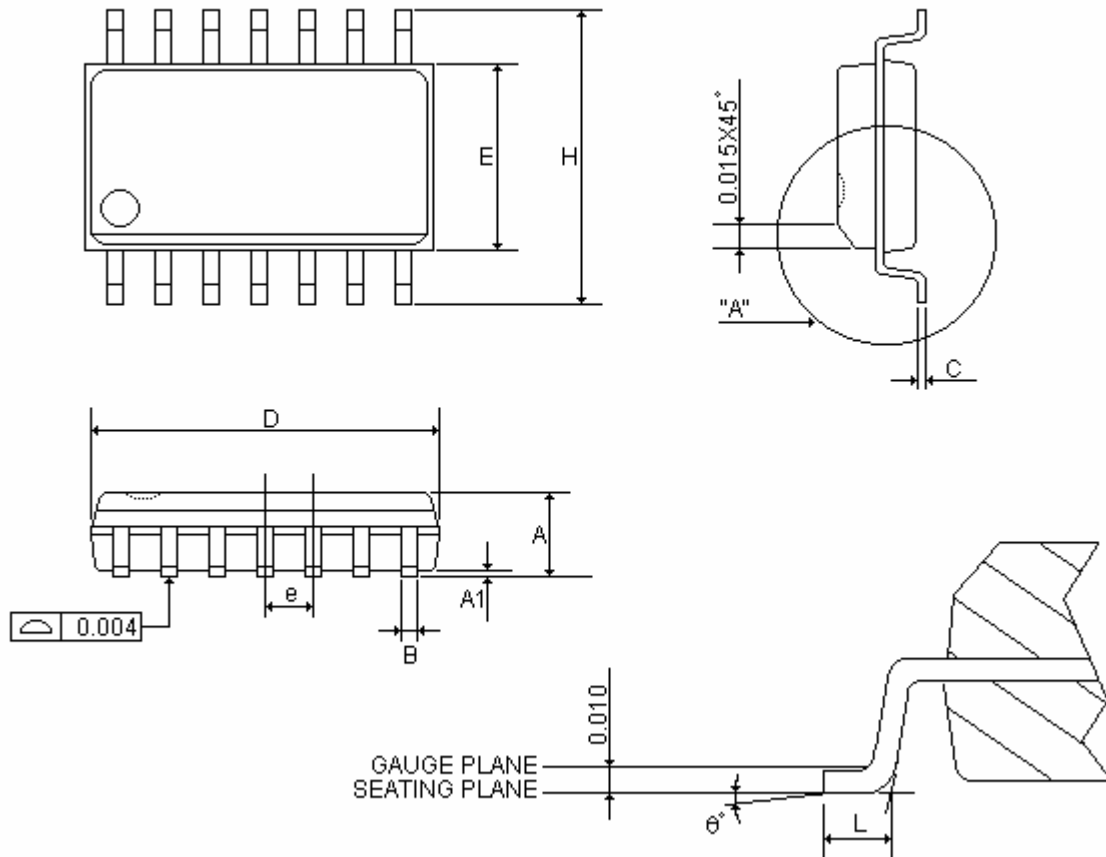
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
<b>A</b>	<b>0.053</b>	<b>0.063</b>	<b>0.069</b>	<b>1.350</b>	<b>1.600</b>	<b>1.750</b>
<b>A1</b>	<b>0.004</b>	<b>0.006</b>	<b>0.010</b>	<b>0.100</b>	<b>0.150</b>	<b>0.250</b>
<b>A2</b>	-	-	<b>0.059</b>	-	-	<b>1.500</b>
<b>b</b>	<b>0.008</b>	<b>0.010</b>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

## 16.9 TQFN 20 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.028	0.030	0.031	0.70	0.75	0.80
A1	0.000	0.001	0.002	0.00	0.02	0.05
A3	0.0009 REF			0.203 REF		
b	0.007	0.010	0.012	0.18	0.25	0.30
D	0.154	0.157	0.161	3.90	4.00	4.10
D2	0.075	0.079	0.083	1.90	2.00	2.10
E	0.154	0.157	0.161	3.90	4.00	4.10
E2	0.075	0.079	0.083	1.90	2.00	2.10
[e]	0.0020 BSC			0.05 BSC		
L	0.012	0.016	0.020	0.30	0.40	0.50

## 16.10 SOP 14 PIN



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

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](mailto:Sn8fae@sonix.com.tw)