

SN8PC22

USER'S MANUAL

Preliminary Version 0.2

SONiX 8-Bit Micro-Controller

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

AMENDENT HISTORY

Version	Date	Description
VER 0.1	Dec. 2010	First issue.
VER 0.2	Jan. 2011	Modify programming pin mapping.

Table of Content

AMENDENT HISTORY	2
1 PRODUCT OVERVIEW	6
1.1 FEATURES	6
1.2 SYSTEM BLOCK DIAGRAM	7
1.3 PIN ASSIGNMENT	8
1.4 PIN DESCRIPTIONS	8
1.5 PIN CIRCUIT DIAGRAMS	9
2 CENTRAL PROCESSOR UNIT (CPU)	11
2.1 PROGRAM MEMORY (ROM)	11
2.1.1 RESET VECTOR (0000H)	12
2.1.2 INTERRUPT VECTOR (0008H)	13
2.1.3 LOOK-UP TABLE DESCRIPTION	15
2.1.4 JUMP TABLE DESCRIPTION	17
2.1.5 CHECKSUM CALCULATION	19
2.2 DATA MEMORY (RAM)	20
2.2.1 SYSTEM REGISTER	21
2.2.1.1 SYSTEM REGISTER TABLE	21
2.2.1.2 SYSTEM REGISTER DESCRIPTION	21
2.2.1.3 BIT DEFINITION of SYSTEM REGISTER	22
2.2.2 ACCUMULATOR	23
2.2.3 PROGRAM FLAG	24
2.2.4 PROGRAM COUNTER	25
2.2.5 H, L REGISTERS	28
2.2.6 Y, Z REGISTERS	29
2.2.7 R REGISTER	29
2.3 ADDRESSING MODE	30
2.3.1 IMMEDIATE ADDRESSING MODE	30
2.3.2 DIRECTLY ADDRESSING MODE	30
2.3.3 INDIRECTLY ADDRESSING MODE	30
2.4 STACK OPERATION	31
2.4.1 OVERVIEW	31
2.4.2 STACK REGISTERS	32
2.4.3 STACK OPERATION EXAMPLE	33
2.5 CODE OPTION TABLE	34
2.5.1 Reset_Pin code option	34
2.5.2 Security code option	34
3 RESET	35
3.1 OVERVIEW	35
3.2 POWER ON RESET	36
3.3 WATCHDOG RESET	36
3.4 BROWN OUT RESET	37
3.4.1 THE SYSTEM OPERATING VOLTAGE	38
3.4.2 LOW VOLTAGE DETECTOR (LVD)	38
3.4.3 BROWN OUT RESET IMPROVEMENT	39
3.5 EXTERNAL RESET	40
3.6 EXTERNAL RESET CIRCUIT	40
3.6.1 Simply RC Reset Circuit	40

3.6.2	Diode & RC Reset Circuit	41
3.6.3	Zener Diode Reset Circuit	41
3.6.4	Voltage Bias Reset Circuit	42
3.6.5	External Reset IC	42
4	SYSTEM CLOCK	43
4.1	OVERVIEW	43
4.2	FCPU (INSTRUCTION CYCLE)	43
4.3	SYSTEM HIGH-SPEED CLOCK	43
4.3.1	HIGH_CLK CODE OPTION	44
4.3.2	INTERNAL HIGH-SPEED OSCILLATOR RC TYPE (IHRC)	44
4.3.3	EXTERNAL HIGH-SPEED OSCILLATOR	44
4.3.4	EXTERNAL OSCILLATOR APPLICATION CIRCUIT	44
4.4	SYSTEM LOW-SPEED CLOCK	45
4.5	OSCM REGISTER	46
4.6	SYSTEM CLOCK MEASUREMENT	46
4.7	SYSTEM CLOCK TIMING	47
5	SYSTEM OPERATION MODE	49
5.1	OVERVIEW	49
5.2	NORMAL MODE	50
5.3	SLOW MODE	50
5.4	POWER DOWN MDOE	50
5.5	GREEN MODE	51
5.6	OPERATING MODE CONTROL MACRO	52
5.7	WAKEUP	53
5.7.1	OVERVIEW	53
5.7.2	WAKEUP TIME	53
6	INTERRUPT	54
6.1	OVERVIEW	54
6.2	INTEN INTERRUPT ENABLE REGISTER	54
6.3	INTRQ INTERRUPT REQUEST REGISTER	55
6.4	GIE GLOBAL INTERRUPT OPERATION	55
6.5	PUSH, POP ROUTINE	56
6.6	EXTERNAL INTERRUPT OPERATION (INT0)	57
6.7	T0 INTERRUPT OPERATION	58
6.8	TC0 INTERRUPT OPERATION	59
6.9	MULTI-INTERRUPT OPERATION	60
7	I/O PORT	61
7.1	OVERVIEW	61
7.2	I/O PORT MODE	62
7.3	I/O PULL UP REGISTER	63
7.4	I/O PORT DATA REGISTER	64
8	TIMERS	65
8.1	WATCHDOG TIMER	65
8.2	T0 8-BIT BASIC TIMER	67
8.2.1	OVERVIEW	67
8.2.2	T0 Timer Operation	68
8.2.3	T0M MODE REGISTER	69
8.2.4	T0C COUNTING REGISTER	69
8.2.5	T0 TIMER OPERATION EXPLAME	70
8.3	TC0 8-BIT TIMER/DUTY-CYCLE PROGRAMMABLE IR DRIVER	71
8.3.1	OVERVIEW	71

8.3.2	TC0 TIMER OPERATION	72
8.3.3	TC0M MODE REGISTER.....	73
8.3.4	TC0C COUNTING REGISTER	73
8.3.5	TC0R AUTO-RELOAD REGISTER.....	74
8.3.6	TC0D PWM DUTY REGISTER	74
8.3.7	PULSE WIDTH MODULATION (PWM)	75
8.3.8	TC0 CAPTURE TIMER	76
8.3.8.1	High Pulse Width Measurement.....	76
8.3.8.2	Low Pulse Width Measurement	76
8.3.8.3	Input Cycle Measurement.....	76
8.3.9	CAPTURE TIMER CONTROL REGISTER.....	77
8.3.10	TC0 TIMER OPERATION EXAMPLE	77
9	IR TRANSMITTER/RECEIVER.....	79
9.1	OVERVIEW	79
9.2	IR CONTROL REGISTER.....	79
9.3	IR TRANSMITTER/RECEIVER OPERATION SEQUENCE.....	80
9.4	IR APPLICATION CIRCUIT	81
10	INSTRUCTION TABLE	82
11	ELECTRICAL CHARACTERISTIC	83
11.1	ABSOLUTE MAXIMUM RATING	83
11.2	ELECTRICAL CHARACTERISTIC.....	83
11.3	CHARACTERISTIC GRAPHS	84
12	DEVELOPMENT TOOL	85
12.1	SN8PC22 EV-KIT	85
12.2	ICE AND EV-KIT APPLICATION NOTIC	86
13	OTP PROGRAMMING PIN.....	87
13.1	PROGRAMMING PIN MAPPING	87
14	MARKING DEFINITION.....	88
14.1	INTRODUCTION	88
14.2	MARKING INDETIFICATION SYSTEM.....	88
14.3	MARKING EXAMPLE	89
14.4	DATECODE SYSTEM	89
15	PACKAGE INFORMATION	90
15.1	P-DIP 20 PIN	90
15.2	SOP 20 PIN.....	91
15.3	SSOP 20 PIN.....	92

1 PRODUCT OVERVIEW

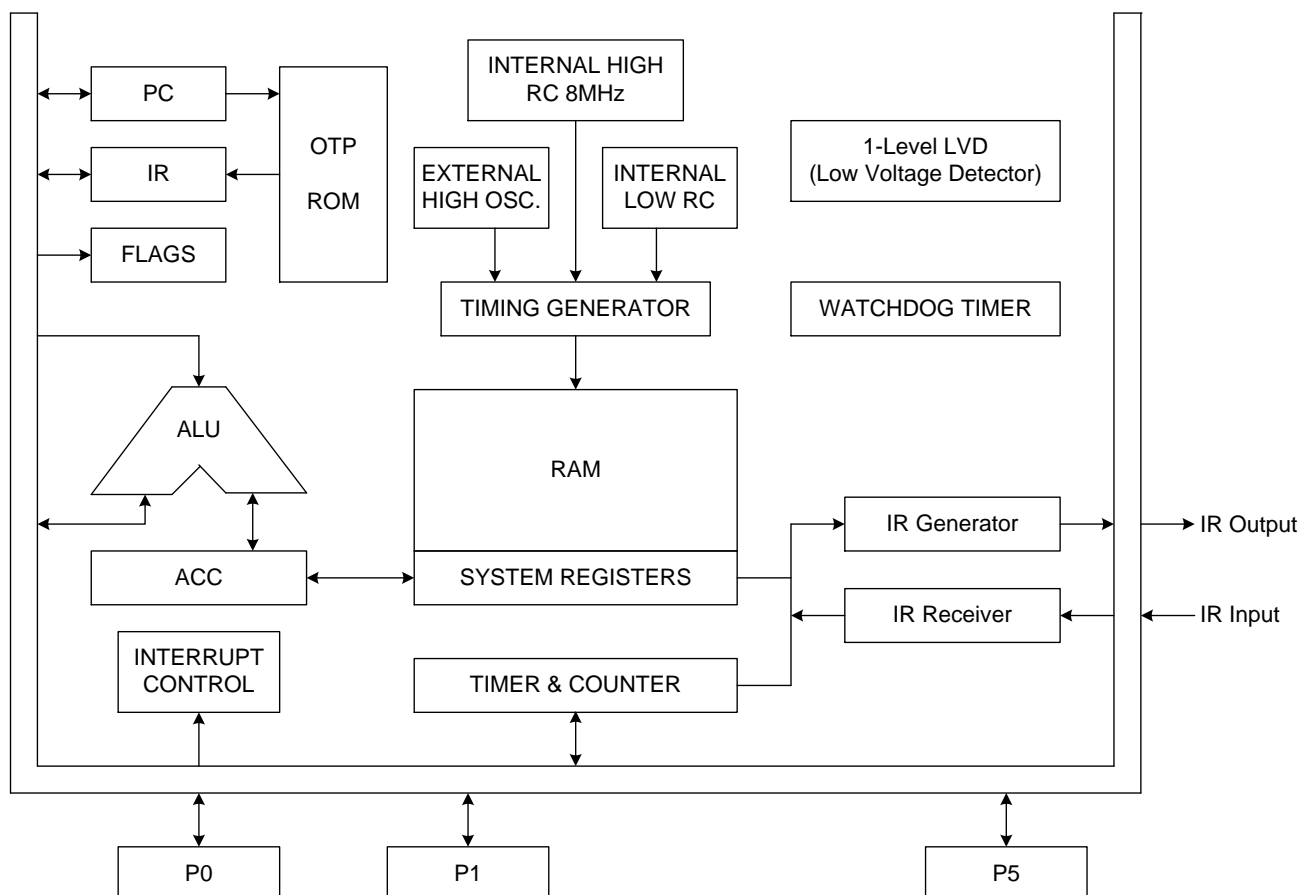
1.1 FEATURES

- ◆ **Memory configuration**
OTP ROM size: 16K * 16 bits.
RAM size: 240 * 8 bits.
- ◆ **8 levels stack buffer.**
- ◆ **I/O pin configuration**
Bi-directional: P0, P1, P5.
400mA IR output pin: P5.4/IROUT
Wakeup: P0, P1 level change trigger.
Pull-up resisters: P0, P1
External Interrupt trigger edge: P0.0 controlled by PEDGE register.
- ◆ **1-Level LVD.**
- ◆ **Fcpu (Instruction cycle)**
 $F_{cpu} = F_{osc}/2, F_{psc}/4, F_{osc}/8, F_{osc}/16$
- ◆ **Powerful instructions**
One clocks per instruction cycle (1T)
Most of instructions are one cycle only.
All ROM area JMP instruction.
All ROM area CALL address instruction.
All ROM area lookup table function (MOVC).
- ◆ **3 interrupt sources**
2 internal interrupts: T0, TC0
1 external interrupts: INT0
- ◆ **One 8-bit basic timer. (T0).**
- ◆ **One 8-bit timer/capture. (TC0).**
1-ch Duty/cycle programmable PWM for IR transmitter.
- ◆ **Build in IR receiver.**
- ◆ **On chip watchdog timer and clock source is Internal low clock RC type (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)**
PDIP 20 pin
SOP 20 pin
SSOP 20 pin

Features Selection Table

CHIP	ROM	RAM	Stack	I/O	Oscillator			IR Receiver	IR Output	Wakeup Pin No.	Package
					Ext. 455K	Ext. 4M	Int. 8M				
SN8PC20	2K*16	56	4	18	V	V	V	-	Duty/cycle programmable, 400mA sink current	16	DIP20/SOP20/SSOP20
SN8PC21	1K*16	32	4	13	-	V	V	-	Duty/cycle programmable, 400mA sink current	13	DIP16/SOP16
SN8PC22	16K*16	240	8	18	-	V	V	V	Duty/cycle programmable, 400mA sink current	16	DIP20/SOP20/SSOP20

1.2 SYSTEM BLOCK DIAGRAM



1.3 PIN ASSIGNMENT

SN8PC22P (PDIP 20 pins)

SN8PC22S (SOP 20 pins)

SN8PC22X (SSOP 20 pins)

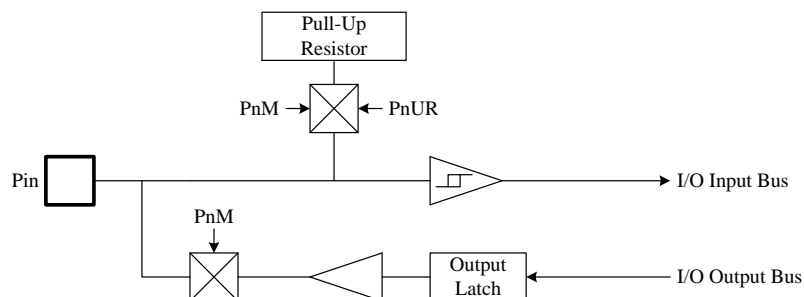
VDD	1	U	20	VSS
P0.0/INT0	2		19	P5.4/IROUT/IRIN
P0.1	3		18	P5.0/IRXO
P0.2/RST/VPP	4		17	P1.7
P0.3/XIN	5		16	P1.6
P0.4/XOUT	6		15	P1.5
P0.5	7		14	P1.4
P0.6	8		13	P1.3
P0.7	9		12	P1.2
P1.0	10		11	P1.1

1.4 PIN DESCRIPTIONS

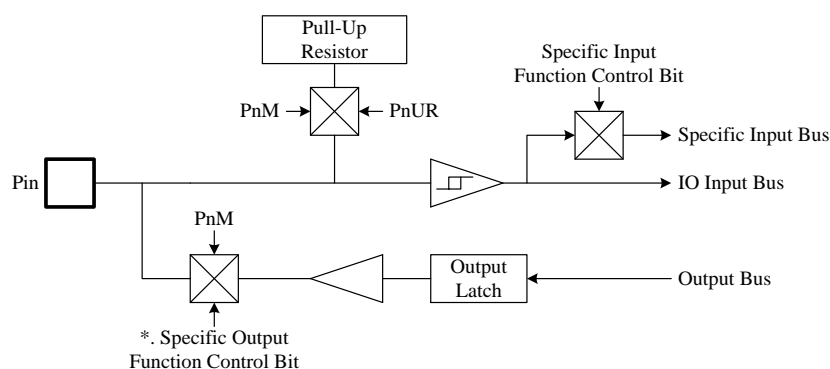
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital and analog circuit.
P0.2/RST/ VPP	I, P	RST: System external reset input pin. Schmitt trigger structure, active "low", normal stay to "high".
		VPP: OTP power input pin in programming mode.
		P0.2: Input only pin with Schmitt trigger structure and no pull-up resistor.
XIN/P0.3	I/O	XIN: Oscillator input pin while external oscillator enable (crystal and RC).
		P0.3: Bi-direction pin. Schmitt trigger structure. Built-in pull-up resistors and level change wake-up function as input mode.
XOUT/P0.4	I/O	XOUT: Oscillator output pin while external crystal enable.
		P0.4: Bi-direction pin. Schmitt trigger structure. Built-in pull-up resistors and level change wake-up function as input mode.
P0.0/INT0	I/O	P0.0: Bi-direction pin. Schmitt trigger structure. Built-in pull-up resistors and level change wake-up function as input mode.
		INT0: External interrupt 0 input pin.
P0.1, P0[7:5]	I/O	P0: Bi-direction pin. Schmitt trigger structure. Built-in pull-up resistors and level change wake-up function as input mode.
P1[7:0]	I/O	P1: Bi-direction pin. Schmitt trigger structure. Built-in pull-up resistors and level change wake-up function as input mode.
P5.0/IRXO	I/O	P5.0: Bi-direction pin. Schmitt trigger structure. Built-in pull-up resistors as input mode.
		IRXO: IR receiver output pin.
P5.4/IROUT/ IRIN	I/O	P5.4: Bi-direction pin. Schmitt trigger structure. Built-in pull-up resistors as input mode.
		IROUT: Duty/cycle programmable IR signal output pin.
		IRIN: IR signal input pin.

1.5 PIN CIRCUIT DIAGRAMS

- Normal bi-direction I/O pin.

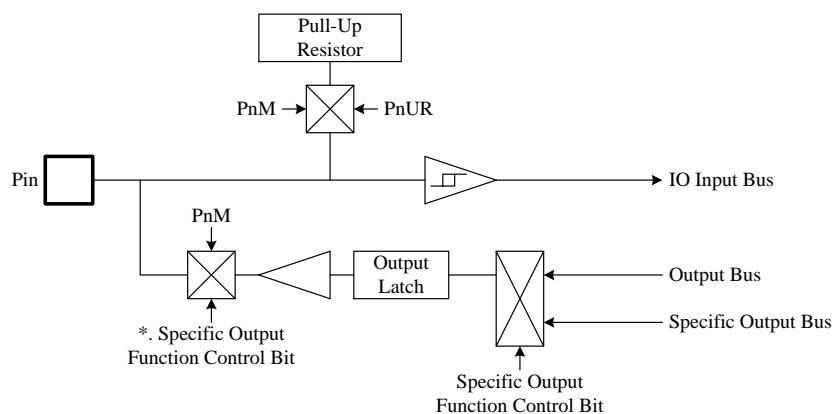


- Bi-direction I/O pin shared with specific digital input function, e.g. INT0, event counter...



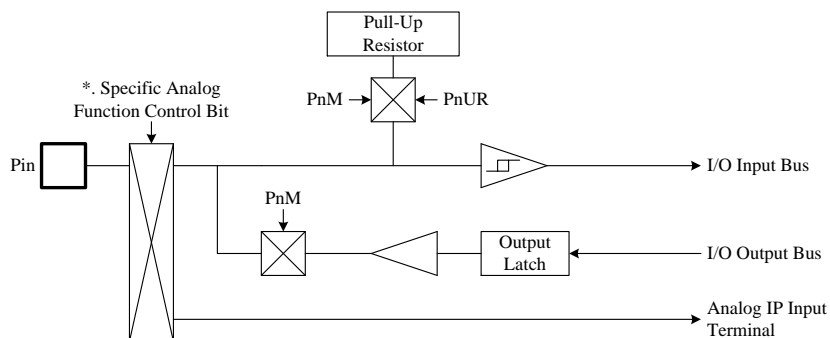
*. Some specific functions switch I/O direction directly, not through PnM register.

- Bi-direction I/O pin shared with specific digital output function, e.g. IR Output...



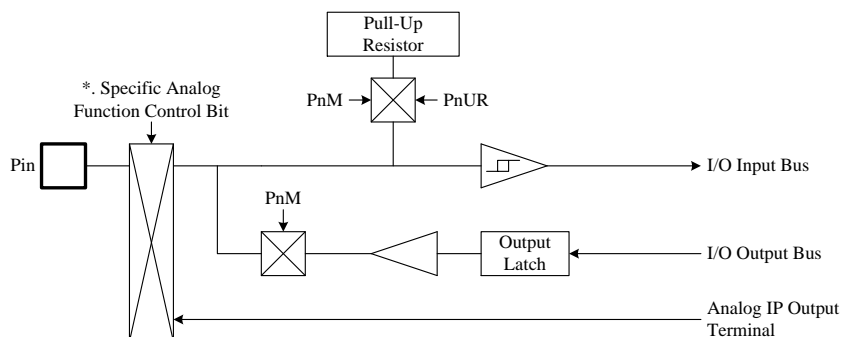
*. Some specific functions switch I/O direction directly, not through PnM register.

- **Bi-direction I/O pin shared with specific analog input function, e.g. XIN...**



*. Some specific functions switch I/O direction directly, not through PnM register.

- **Bi-direction I/O pin shared with specific analog output function, e.g. XOUT...**



*. Some specific functions switch I/O direction directly, not through PnM register.

2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (ROM)

☞ 16K words ROM

ROM			
0000H	Reset vector	User reset vector Jump to user start address	
0001H	General purpose area		
.			
0007H			
0008H	Interrupt vector	User interrupt vector	
0009H	General purpose area	User program	
.			
.			
000FH			
0010H			
0011H			
.			
.			
.			
.			
3FFCH		End of user program	
3FFDH	Reserved		
3FFEH			
3FFFH			

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

2.1.1 RESET VECTOR (0000H)

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

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

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

➤ **Example: Defining Reset Vector**

```

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

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

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

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

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

    ENDP              ; End of program
```

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

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

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

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

    ENDP                ; End of program.
```

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

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

2.1.3 LOOK-UP TABLE DESCRIPTION

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

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

```

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

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

```

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

➤ **Example: INC_YZ macro.**

```

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

        INCMS    Y                ; Y+1
        NOP      ; Not overflow

@@:
        ENDM

```

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

```

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

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

```

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

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

```

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

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

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

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

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

```


2.1.4 JUMP TABLE DESCRIPTION

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

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

➤ **Example: Jump table.**

```

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

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

```

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

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

```

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

```

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

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

```

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

```

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

➤ **Example: “@JMP_A” operation.**

; Before compiling program.

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

; After compiling program.

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

2.1.5 CHECKSUM CALCULATION

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

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

```

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

@@:
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)

240 X 8-bit RAM

	Address	RAM Location	
BANK 0	000h	General Purpose Area	RAM Bank 0
	"		
	"		
	"		
	07Fh	System Register	080h~0FFh of Bank 0 store system registers (128 bytes).
	080h		
BANK 1	"		
	"		
	0FFh	General Purpose Area	End of Bank 0 RAM Bank 1
	100h		
	"		
	"		
	16Fh		End of Bank 1

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.

- **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 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	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	-	P1M	-	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	-	-	-	P5	-	-	T0M	T0C	TC0M	TC0C	-	-	-	STKP
E	P0UR	P1UR	-	-	-	P5UR	@HL	@YZ	TC0D	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 SYSTEM REGISTER DESCRIPTION

H, L = Working, @HL addressing register.
 R = Working register and ROM look-up data buffer.
 PnM = Port n input/output mode register.
 INTRQ = Interrupt request register.
 OSCM = Oscillator mode register.
 TC0R = TC0 counter auto-reload data buffer.
 Pn = Port n data buffer.
 T0C = T0 counting register.
 TC0C = TC0 counting register.
 PnUR = Port n pull-up resistor control register.
 @YZ = RAM YZ indirect addressing index pointer.
 STK0~STK7 = Stack 0 ~ stack 7 buffer.

Y, Z = Working, @YZ and ROM addressing register.
 PFLAG = ROM page and special flag register.
 PEDGE = P0.0 edge direction register.
 INTEN = Interrupt enable register.
 WDTR = Watchdog timer clear register.
 PCH, PCL = Program counter.
 T0M = T0 mode register.
 TC0M = TC0 mode register.
 STKP = Stack pointer buffer.
 @HL = RAM HL indirect addressing index pointer.
 TC0D = TC0 duty control register.

2.2.1.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD				C	DC	Z	R/W	PFLAG
087H								RBANKS0	R/W	RBANK
0B8H	P07M	P06M	P05M	P04M	P03M		P01M	P00M	R/W	P0M
0BFH				P00G1	P00G0		IRXO	PEDGS	R/W	PEDGE
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C5H				P54M				P50M	R/W	P5M
0C8H			TC0IRQ	T0IRQ				P00IRQ	R/W	INTRQ
0C9H			TC0IEN	T0IEN				P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH			PC13	PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D5H				P54				P50	R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0	T0TB	CPTS1	CPTS0	IRXEN	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	IRSTS	IREN	CREN	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	P07R	P06R	P05R	P04R	P03R		P01R	P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E5H				P54R				P50R	W	P5UR
0E6H	@HL7	@ HL 6	@ HL5	@ HL4	@ HL3	@ HL2	@ HL1	@ HL0	R/W	@ HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E8H	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0	R/W	TC0D
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	1	1	S7PC13	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	1	1	S6PC13	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	1	1	S5PC13	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	1	1	S4PC13	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	1	1	S3PC13	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	1	1	S2PC13	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	1	1	S1PC13	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	1	1	S0PC13	S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

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

2.2.2 ACCUMULATOR

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

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

```

; Read ACC data and store in BUF data memory.

```

MOV BUF, A

; Write a immediate data into ACC.

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

```
; Write ACC data from BUF data memory.
```

MOV A, BUF

; or

B0MOV A, BUF

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

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

INT_SERVICE:

PUSH ; Save ACC and PFLAG to buffers.

...

POP ; Load ACC and PFLAG from buffers.

RETI ; Exit interrupt service vector

2.2.3 PROGRAM FLAG

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

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	-	-	-	C	DC	Z
Read/Write	R/W	R/W	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	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 2 **C**: Carry flag
 1 = Addition with carry, subtraction without borrowing, rotation with shifting out logic "1", comparison result ≥ 0 .
 0 = Addition without carry, subtraction with borrowing signal, rotation with shifting out logic "0", comparison result < 0 .

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

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

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

2.2.4 PROGRAM COUNTER

The program counter (PC) is a 14-bit binary counter separated into the high-byte 6 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 13.

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	PC13	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	0
	PCH								PCL							

☞ ONE ADDRESS SKIPPING

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

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

```

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

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

```

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

```

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

```

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

INCS instruction:

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

...

...

C0STEP: NOP

INCMS instruction:

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

...

...

C0STEP: NOP

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

DECS instruction:

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

...

...

C0STEP: NOP

DECMS instruction:

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

...

...

C0STEP: NOP

✎ MULTI-ADDRESS JUMPING

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

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

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

; PC = 0323H

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

; PC = 0328H

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

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

; PC = 0323H

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

2.2.5 H, L REGISTERS

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

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

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

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

- **Example: If want to read a data from RAM address 20H of bank_0, it can use indirectly addressing mode to access data as following.**

```

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

```

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

```

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

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

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
Y	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

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

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

```

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

```

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

```

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

```

CLR_YZ_BUF:

```

CLR      @YZ            ; Clear @YZ to be zero

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

```

```

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

2.2.7 R REGISTER

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

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

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

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

2.3 ADDRESSING MODE

2.3.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.3.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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

2.3.3 INDIRECTLY ADDRESSING MODE

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

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

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

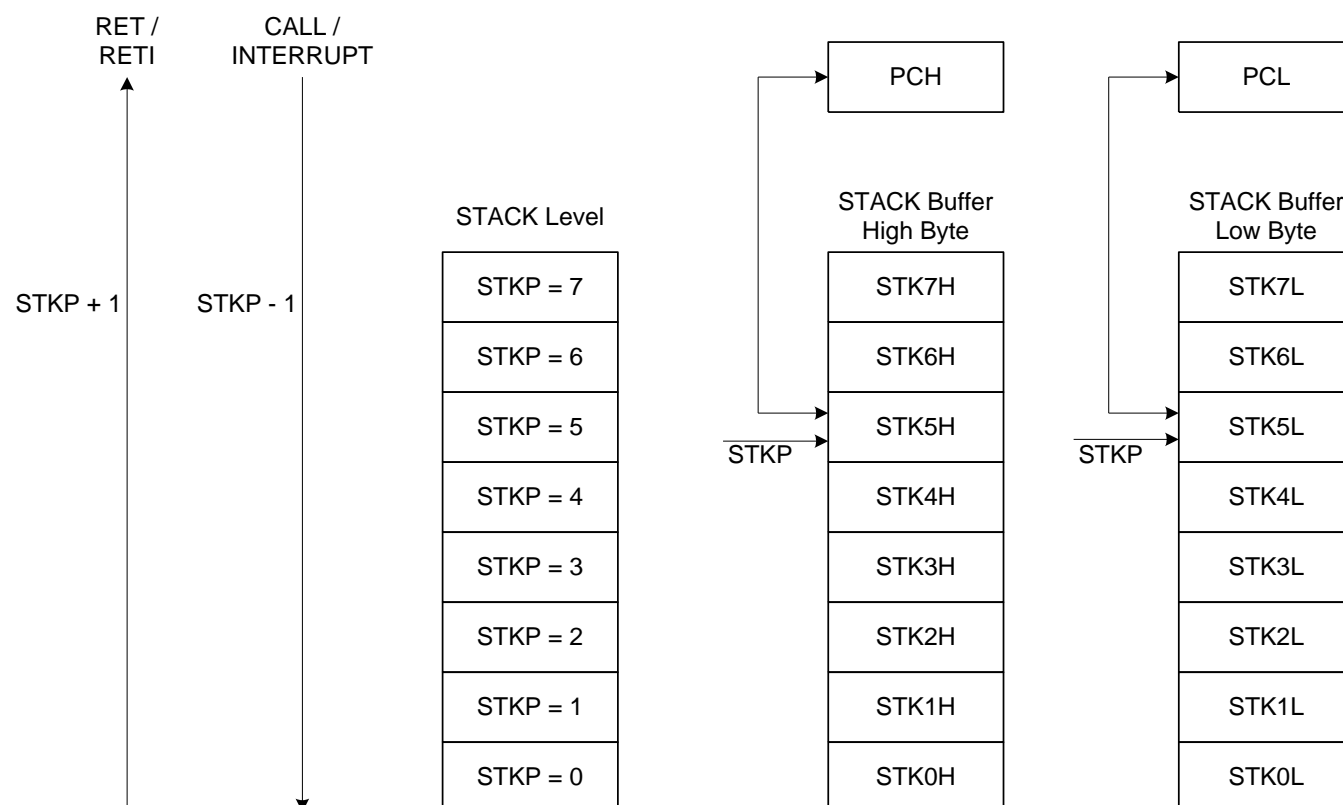
- **Example: Indirectly addressing mode with @YZ register**

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

2.4 STACK OPERATION

2.4.1 OVERVIEW

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



2.4.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 14-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

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

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

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

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

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

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

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

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	SnPC13	SnPC12	SnPC11	SnPC10	SnPC9	SnPC8
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

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

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

2.4.3 STACK OPERATION EXAMPLE

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

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

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

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

2.5 CODE OPTION TABLE

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

Code Option	Content	Function Description
High_Clk	IHRC_8M	High speed internal 8MHz RC. XIN/XOUT pins are bi-direction GPIO mode.
	IHRC_RTC	High speed internal 8MHz RC with 0.5sec RTC. XIN/XOUT pins are bi-direction GPIO mode.
	Ext_RC	Low cost RC for external high clock oscillator and XOUT is bi-direction GPIO mode.
	32K X'tal	Low frequency, power saving crystal (e.g. 32.768KHz) for external high clock oscillator.
	12M X'tal	High-speed crystal /resonator (e.g. 12M) for external high clock oscillator.
	4M X'tal	Standard crystal /resonator (e.g. 4M) for external high clock oscillator.
Fcpu	Fhosc/2	Instruction cycle is 2 oscillator clocks.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
	Fhosc/16	Instruction cycle is 16 oscillator clocks.
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.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.

2.5.1 Reset_Pin code option

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

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

2.5.2 Security code option

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

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

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

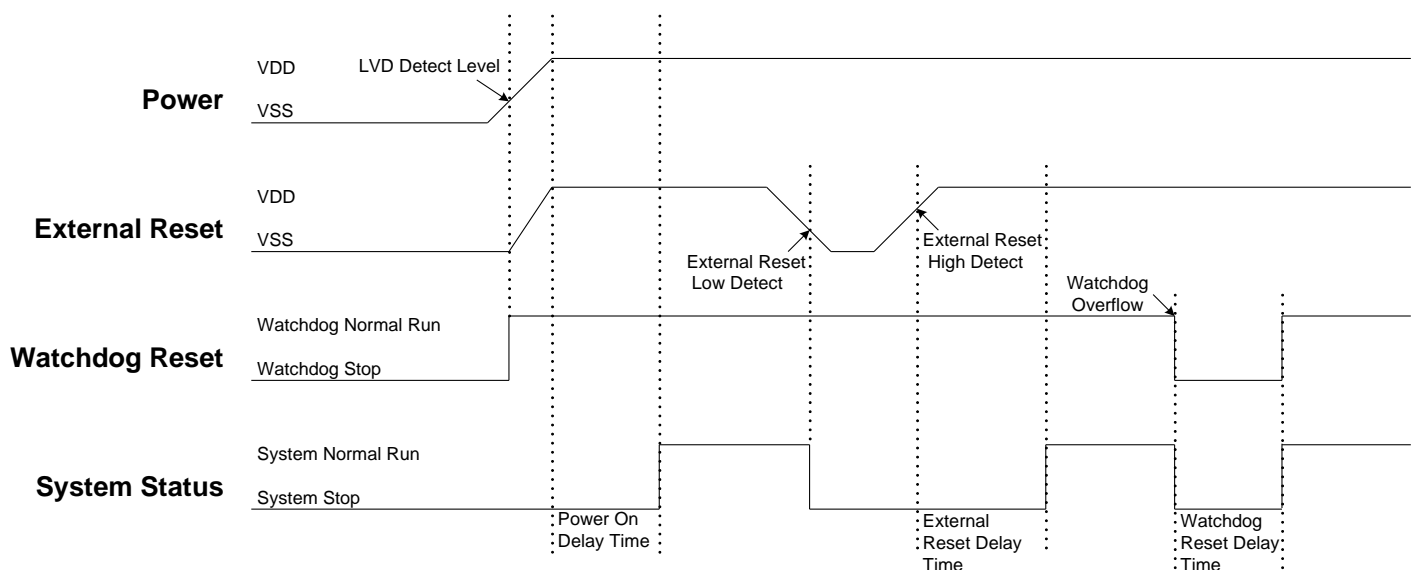
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

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

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

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

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



3.2 POWER ON RESET

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

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

3.3 WATCHDOG RESET

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

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

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

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

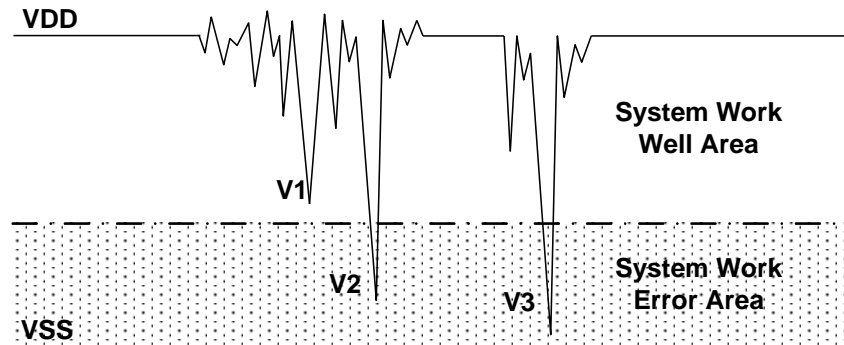
Watchdog timer application note is as following.

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

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

3.4 BROWN OUT RESET

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



Brown Out Reset Diagram

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

DC application:

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

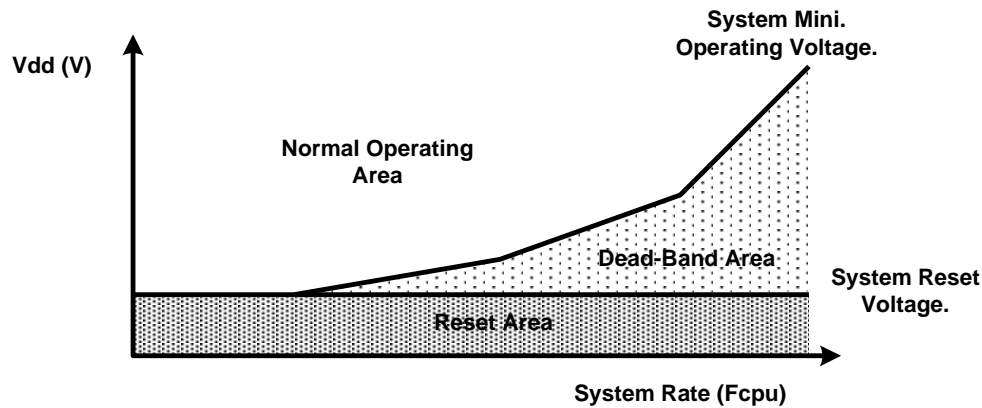
AC application:

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

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

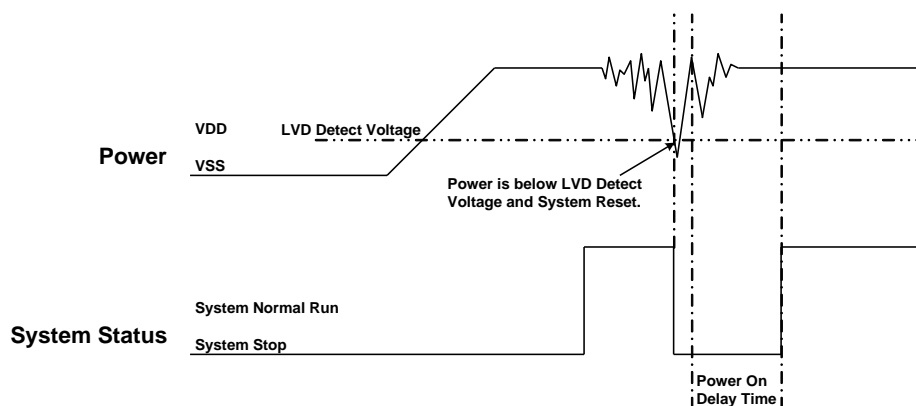
3.4.1 THE SYSTEM OPERATING VOLTAGE

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



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

3.4.2 LOW VOLTAGE DETECTOR (LVD)



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

3.4.3 BROWN OUT RESET IMPROVEMENT

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

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

*** Note:**

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

Watchdog reset:

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

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

Reduce the system executing rate:

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

External reset circuit:

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

3.5 EXTERNAL RESET

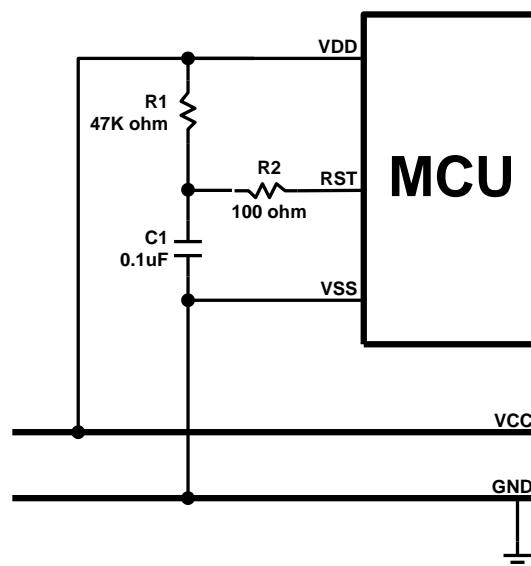
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

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

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

3.6 EXTERNAL RESET CIRCUIT

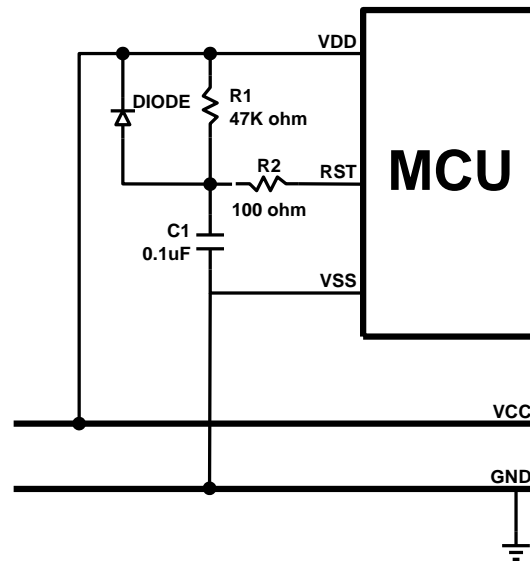
3.6.1 Simply RC Reset Circuit



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

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

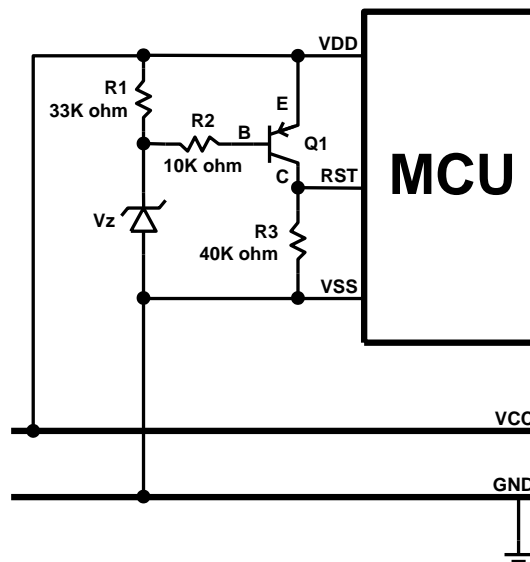
3.6.2 Diode & RC Reset Circuit



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

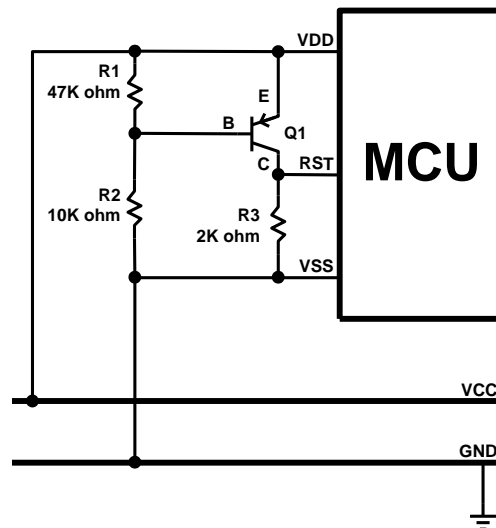
* **Note:** The R2 100 ohm resistor of “Simply reset circuit” and “Diode & RC reset circuit” is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).

3.6.3 Zener Diode Reset Circuit



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

3.6.4 Voltage Bias Reset Circuit

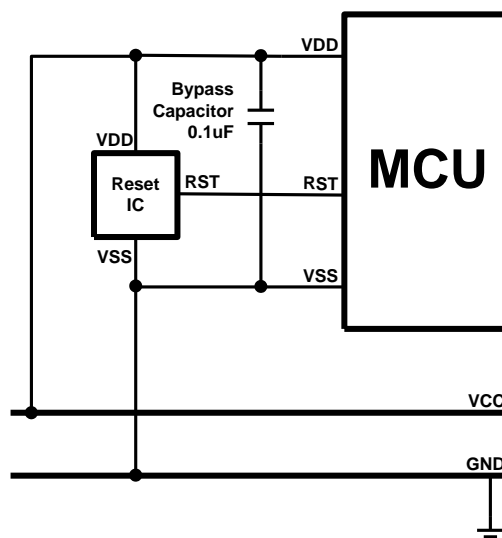


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to $0.7V \times (R1 + R2)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

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

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

3.6.5 External Reset IC



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

4 SYSTEM CLOCK

4.1 OVERVIEW

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

- **High-speed oscillator**

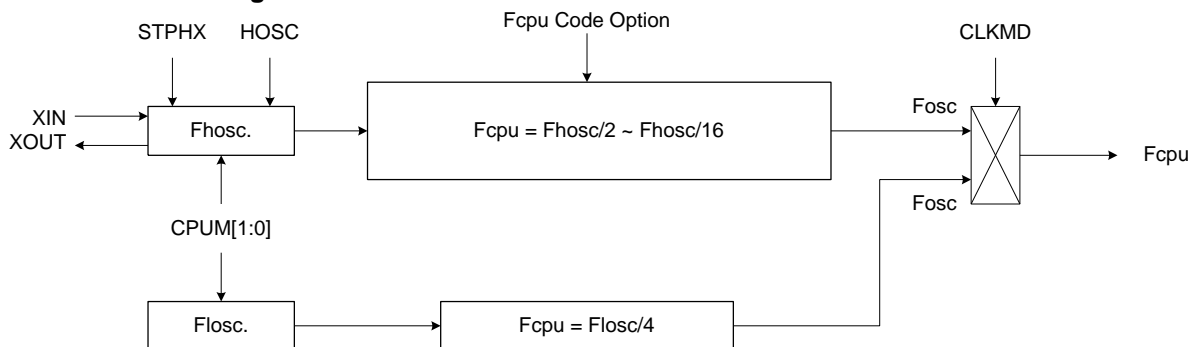
Internal high-speed oscillator is 8MHz RC type called “IHRC”.

External high-speed oscillator includes crystal/ceramic (4MHz, 12MHz, 32KHz) and RC type.

- **Low-speed oscillator**

Internal low-speed oscillator is 10KHz @3V RC type called “ILRC”.

- **System clock block diagram**



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

4.2 FCPU (INSTRUCTION CYCLE)

The system clock rate is instruction cycle called “Fcpu” which is divided from the system clock source and decides the system operating rate. Fcpu rate is selected by Fcpu code option and the range is Fhosc/2~Fhosc/16 under system normal mode. If the system high clock source is external 4MHz crystal, and the Fcpu code option is Fhosc/4, the Fcpu frequency is 4MHz/4 = 1MHz. Under system slow mode, the Fcpu is fixed Fosc/4, 10KHz/4=2.5KHz @3V.

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

4.3 SYSTEM HIGH-SPEED CLOCK

The system high-speed clock has internal and external two-type. The external high-speed clock includes 4MHz, 12MHz, 32KHz crystal/ceramic and RC type. These high-speed oscillators are selected by “High_CLK” code option. The internal high-speed clock supports real time clock (RTC) function. Under “IHRC_RTC” mode, the internal high-speed clock and external 32KHz oscillator active. The internal high-speed clock is the system clock source, and the external 32KHz oscillator is the RTC clock source to supply a accurately real time clock rate.

4.3.1 HIGH_CLK CODE OPTION

For difference clock functions, Sonix provides multi-type system high clock options controlled by “High_CLK” code option. The High_CLK code option defines the system oscillator types including IHRC_8M, IHRC_RTC, RC, 32K X’tal, 12M X’tal and 4M X’tal. These oscillator options support different bandwidth oscillator.

- **IHRC_8M:** The system high-speed clock source is 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 device.
- **IHRC_RTC:** The system high-speed clock source is internal high-speed 8MHz RC type oscillator. The RTC clock source is external low-speed 32768Hz crystal. The XIN and XOUT pins are defined to drive external 32768Hz crystal and disables GPIO function.
- **RC:** The system high-speed clock source is external low cost RC type oscillator. The RC oscillator circuit only connects to XIN pin, and the XOUT pin is bi-direction GPIO mode.
- **32K X’tal:** The system high-speed clock source is external low-speed 32768Hz crystal. The option only supports 32768Hz crystal and the RTC function is workable.
- **12M X’tal:** The system high-speed clock source is external high-speed crystal/ceramic. The oscillator bandwidth is 10MHz~16MHz.
- **4M X’tal:** The system high-speed clock source is external high-speed crystal/resonator. The oscillator bandwidth is 1MHz~10MHz.

For power consumption under “IHRC_RTC” mode, the internal high-speed oscillator and internal low-speed oscillator stops and only external 32KHz crystal actives under green mode. The condition is the watchdog timer can’t be “Always_On” option, or the internal low-speed oscillator actives.

4.3.2 INTERNAL HIGH-SPEED OSCILLATOR RC TYPE (IHRC)

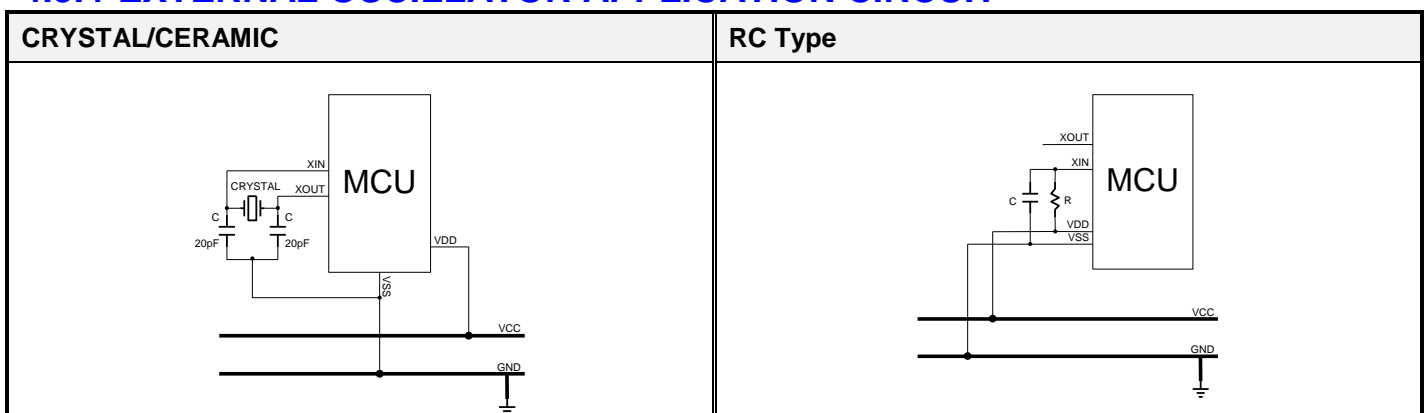
The internal high-speed oscillator is 8MHz RC type. The accuracy is $\pm 2\%$ under commercial condition. When the “High_CLK” code option is “IHRC_8M” or “IHRC_RTC”, the internal high-speed oscillator is enabled.

- **IHRC_8M:** The system high-speed clock is internal 8MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.
- **IHRC_RTC:** The system high-speed clock is internal 8MHz oscillator RC type, and the real time clock is external 32768Hz crystal. XIN/XOUT pins connect with external 32768Hz crystal.

4.3.3 EXTERNAL HIGH-SPEED OSCILLATOR

The external high-speed oscillator includes 4MHz, 12MHz, 32KHz and RC type. The 4MHz, 12MHz and 32KHz oscillators support crystal and ceramic types connected to XIN/XOUT pins with 20pF capacitors to ground. The RC type is a low cost RC circuit only connected to XIN pin. The capacitance is not below 100pF, and use the resistance to decide the frequency.

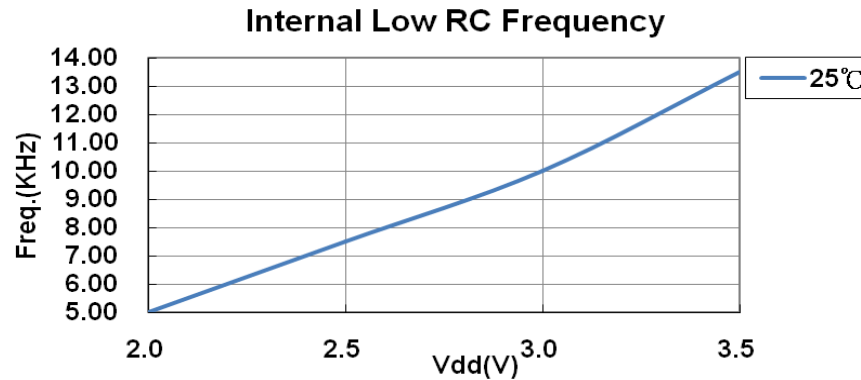
4.3.4 EXTERNAL OSCILLATOR APPLICATION CIRCUIT



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

4.4 SYSTEM LOW-SPEED CLOCK

The system low clock source is the internal low-speed oscillator built in the micro-controller. The low-speed oscillator uses RC type oscillator circuit. The frequency is affected by the voltage and temperature of the system. In common condition, the frequency of the RC oscillator is about 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” bit of OSCM register.

- ***F_{osc}* = Internal low RC oscillator (about 10KHz @3V).**
- ***Slow mode F_{cpu}* = *F_{osc}* / 4**

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

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

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

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

4.5 OSCM REGISTER

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

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

- Bit 1 **STPHX**: 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.

- **IHRC_8M**: “STPHX=1” disables internal high speed RC type oscillator.
- **IHRC_RTC**: “STPHX=1” disables internal high speed RC type oscillator and external 32768Hz crystal.
- **RC, 4M, 12M, 32K**: “STPHX=1” disables external oscillator.

4.6 SYSTEM CLOCK MEASUREMENT

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

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

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

@ @:

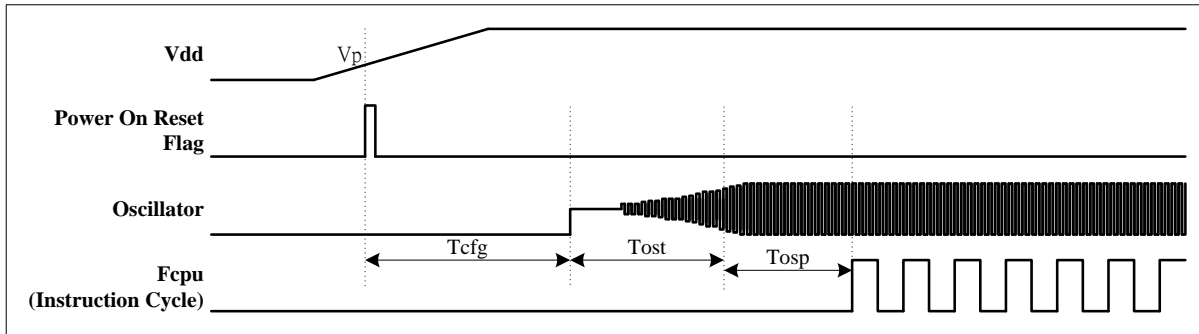
```
B0BSET    P0.0           ; Output Fcpu toggle signal in low-speed clock mode.
B0BCLR    P0.0           ; Measure the Fcpu frequency by oscilloscope.
JMP       @B
```

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

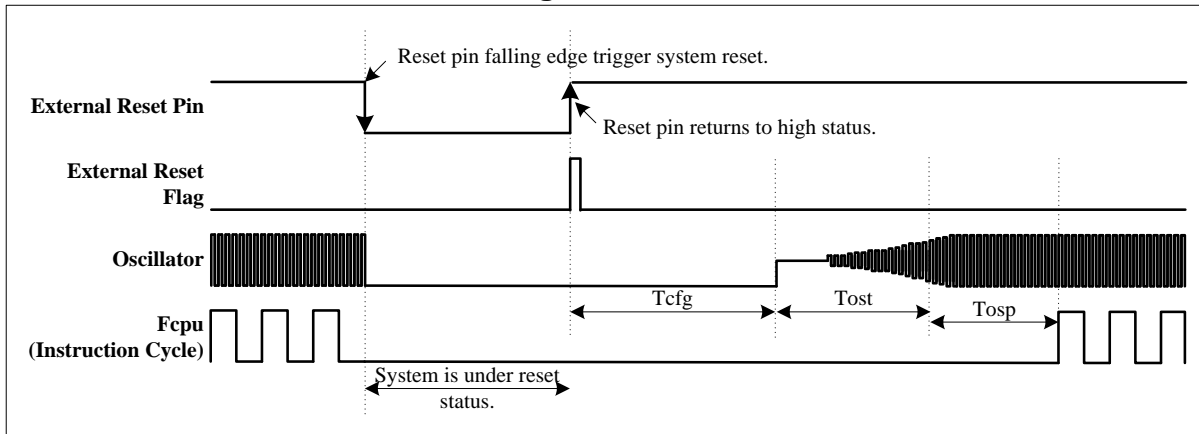
4.7 SYSTEM CLOCK TIMING

Parameter	Symbol	Description	Typical
Hardware configuration time	Tcfg	$2048 \cdot F_{ILRC}$	64ms @ $F_{ILRC} = 32\text{KHz}$ 128ms @ $F_{ILRC} = 16\text{KHz}$
Oscillator start up time	Tost	The start-up time is depended on oscillator's material, factory and architecture. Normally, the low-speed oscillator's start-up time is lower than high-speed oscillator. The RC type oscillator's start-up time is faster than crystal type oscillator.	-
Oscillator warm-up time	Tosp	Oscillator warm-up time of reset condition. $2048 \cdot F_{hosc}$ (Power on reset, LVD reset, watchdog reset, external reset pin active.)	64ms @ $F_{hosc} = 32\text{KHz}$ 512us @ $F_{hosc} = 4\text{MHz}$ 256us @ $F_{hosc} = 8\text{MHz}$
		Oscillator warm-up time of power down mode wake-up condition. $2048 \cdot F_{hosc}$Crystal/resonator type oscillator, e.g. 32768Hz crystal, 4MHz crystal, 16MHz crystal... $32 \cdot F_{hosc}$RC type oscillator, e.g. external RC type oscillator, internal high-speed RC type oscillator.	64ms @ $F_{hosc} = 32\text{KHz}$ 512us @ $F_{hosc} = 4\text{MHz}$ 256us @ $F_{hosc} = 8\text{MHz}$

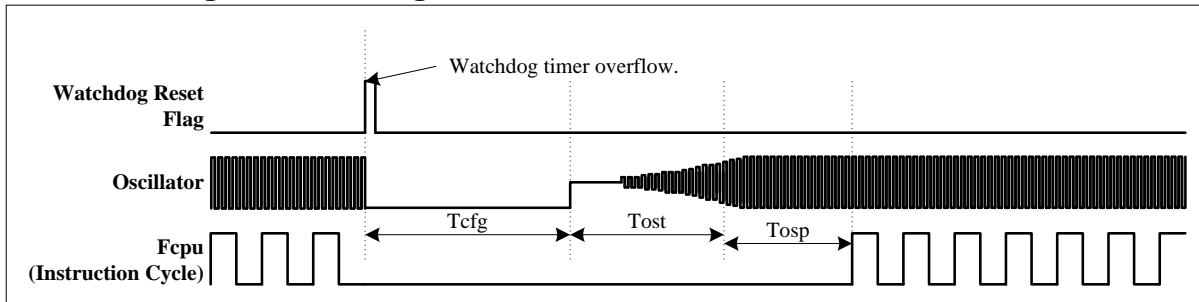
● Power On Reset Timing



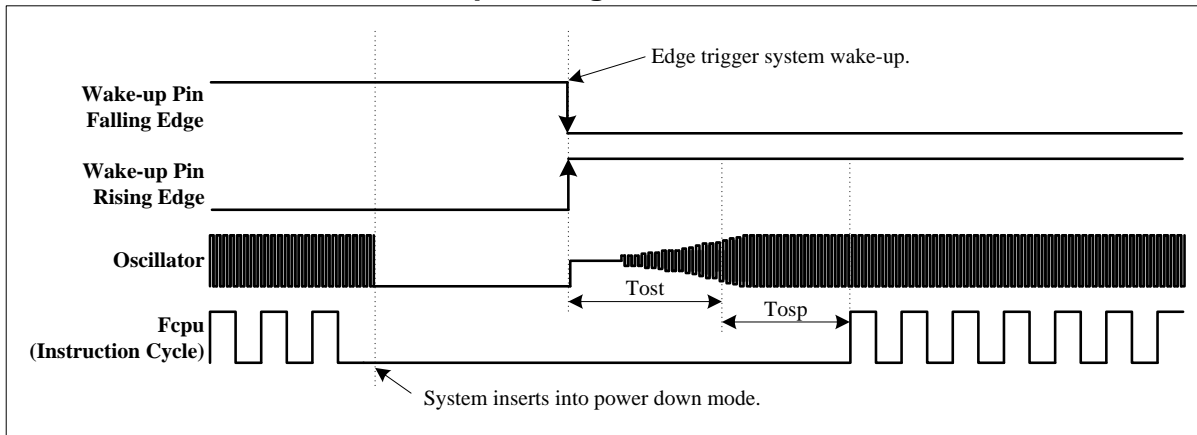
● External Reset Pin Reset Timing



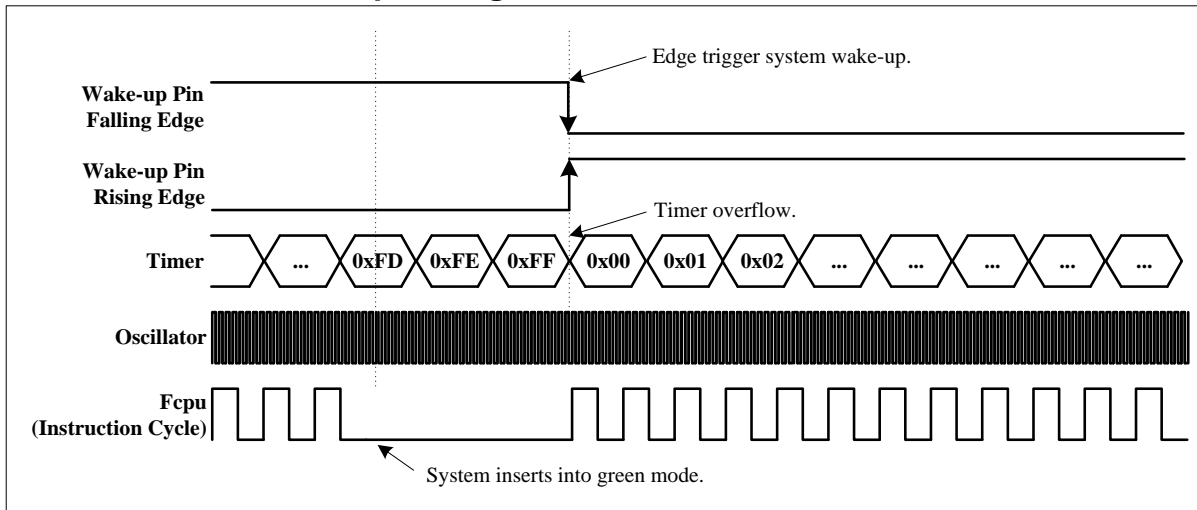
● Watchdog Reset Timing



● Power Down Mode Wake-up Timing

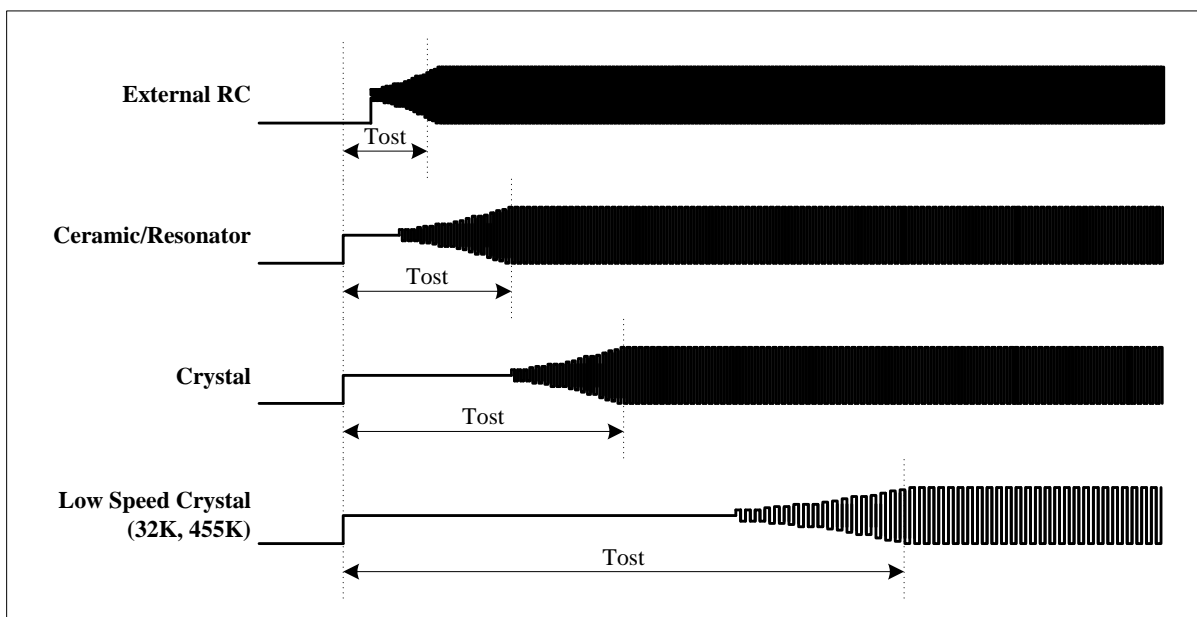


● Green Mode Wake-up Timing



● Oscillator Start-up Time

The start-up time is depended on oscillator's material, factory and architecture. Normally, the low-speed oscillator's start-up time is lower than high-speed oscillator. The RC type oscillator's start-up time is faster than crystal type oscillator.



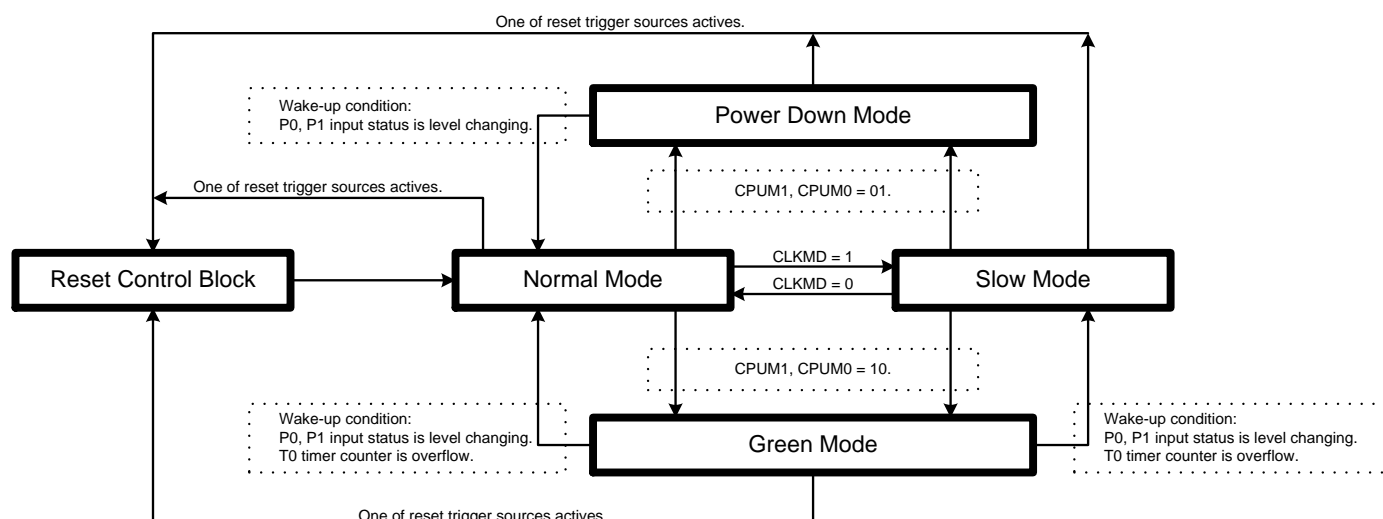
5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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

Operating Mode Control Block



Operating Mode Clock Control Table

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

- **EHOSC**: External high-speed oscillator (XIN/XOUT).
- **IHRC**: Internal high-speed oscillator RC type.
- **ILRC**: Internal low-speed oscillator RC type.

5.2 NORMAL MODE

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

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

5.3 SLOW MODE

The slow mode is system low clock operating mode. The system clock source is from internal low speed RC type oscillator. The slow mode is controlled by CLKMD bit of OSCM register. When CLKMD=0, the system is in normal mode. When CLKMD=1, the system inserts into slow mode. The high speed oscillator won't be disabled automatically after switching to slow mode, and must be disabled by SPTHX bit to reduce power consumption. In slow mode, the system rate is fixed $F_{osc}/4$ (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 actives, 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. 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 activates as enabled, and the timer clock source is the non-stop system clock. The green mode has 2 wake-up sources. One is the P0, P1 level change trigger wake-up. The other one is internal timer with wake-up function occurring overflow. That's mean users can setup one fix period to timer, and the system is waked up until the time out. Inserting green mode is controlled by CPUM1 bit of OSCM register. When CPUM1=1, the system inserts into green mode. After system wake-up from green mode, the CPUM1 bit is disabled (zero status) automatically.

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

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

5.6 OPERATING MODE CONTROL MACRO

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

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

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

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

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

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

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

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

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

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

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

; Set T0 timer wakeup function.

```
BOBCLR    FT0IEN    ; To disable T0 interrupt service
BOBCLR    FT0ENB    ; To disable T0 timer
MOV        A,#20H    ;
BOMOV     T0M,A      ; To set T0 clock = Fcpu / 64
MOV        A,#74H    ;
BOMOV     T0C,A      ; To set T0C initial value = 74H (To set T0 interval = 10 ms)
BOBCLR    FT0IEN    ; To disable T0 interrupt service
BOBCLR    FT0IRQ    ; To clear T0 interrupt request
BOBSET    FT0ENB    ; To enable T0 timer
```

; Go into green mode

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

5.7 WAKEUP

5.7.1 OVERVIEW

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

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

5.7.2 WAKEUP TIME

When the system is in power down mode (sleep mode), the high clock oscillator stops. When waked up from power down mode, MCU waits for 2048 external high-speed oscillator clocks and 32 internal high-speed oscillator clocks as the wakeup time to stable the oscillator circuit. After the wakeup time, the system goes into the normal mode.

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

The value of the external high clock oscillator wakeup time is as the following.

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

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

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

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

$$\text{The Wakeup time} = 1/F_{\text{hosc}} * 32 \text{ (sec)} + \text{high clock start-up time}$$

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

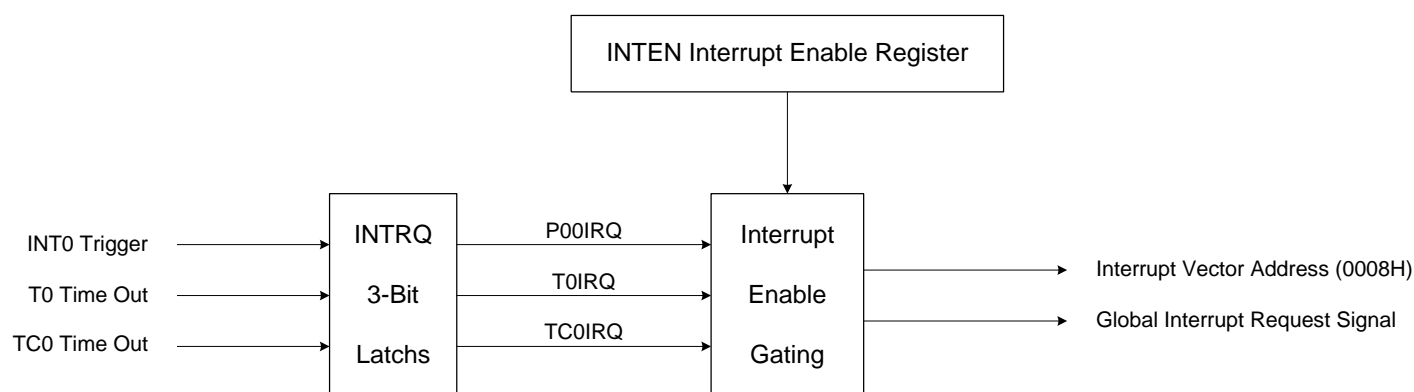
$$\text{The wakeup time} = 1/F_{\text{hosc}} * 32 = 2 \text{ us} \quad (F_{\text{hosc}} = 16\text{MHz})$$

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

6 INTERRUPT

6.1 OVERVIEW

This MCU provides three interrupt sources, including two internal interrupt (T0/TC0) and one external interrupt (INT0). The external interrupt can wakeup the chip while the system is switched from power down mode to high-speed normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to “0” for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to “1” to accept the next interrupts’ request. All of the interrupt request signals are stored in INTRQ register.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	-	-	TC0IEN	T0IEN	-	-	-	P00IEN
Read/Write	-	-	R/W	R/W	-	-	-	R/W
After reset	-	-	0	0	-	-	-	0

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

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

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

6.3 INTRQ INTERRUPT REQUEST REGISTER

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

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	-	-	TC0IRQ	T0IRQ	-	-	-	P00IRQ
Read/Write	-	-	R/W	R/W	-	-	-	R/W
After reset	-	-	0	0	-	-	-	0

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

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

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

6.4 GIE GLOBAL INTERRUPT OPERATION

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

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

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

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

BOBSET FGIE ; Enable GIE

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

6.5 PUSH, POP ROUTINE

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

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

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

```

                                ORG      0
                                JMP      START

                                ORG      8
                                JMP      INT_SERVICE

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)

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 first after wake-up.

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	-	P00G1	P00G0	-	IRXO	PEDGS
Read/Write	-	-	-	R/W	R/W	-	R/W	R/W
After reset	-	-	-	1	0	-	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, #18H
B0MOV    PEDGE, A      ; Set INT0 interrupt trigger as bi-direction edge.

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

- Example: INT0 interrupt service routine.**

```

ORG      8              ; Interrupt vector
JMP      INT_SERVICE

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

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

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

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

6.7 T0 INTERRUPT OPERATION

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

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

B0BCLR	FT0IEN	; Disable T0 interrupt service
B0BCLR	FT0ENB	; Disable T0 timer
MOV	A, #20H	;
B0MOV	T0M, A	; Set T0 clock = Fcpu / 64
MOV	A, #64H	; Set T0C initial value = 64H
B0MOV	T0C, A	; Set T0 interval = 10 ms
B0BSET	FT0IEN	; Enable T0 interrupt service
B0BCLR	FT0IRQ	; Clear T0 interrupt request flag
B0BSET	FT0ENB	; Enable T0 timer
B0BSET	FGIE	; Enable GIE

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

	ORG	8	; Interrupt vector
	JMP	INT_SERVICE	
INT_SERVICE:			
	...		; Push routine to save ACC and PFLAG to buffers.
	B0BTS1	FT0IRQ	; Check T0IRQ
	JMP	EXIT_INT	; T0IRQ = 0, exit interrupt vector
	B0BCLR	FT0IRQ	; Reset T0IRQ
	MOV	A, #64H	
	B0MOV	T0C, A	; Reset T0C.
	...		; T0 interrupt service routine
	...		
EXIT_INT:			
	...		; Pop routine to load ACC and PFLAG from buffers.
	RETI		; Exit interrupt vector

6.8 TC0 INTERRUPT OPERATION

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

➤ **Example: TC0 interrupt request setup. Fcpu = 16MHz / 16.**

B0BCLR	FTC0IEN	; Disable TC0 interrupt service
B0BCLR	FTC0ENB	; Disable TC0 timer
MOV	A, #20H	;
B0MOV	TC0M, A	; Set TC0 clock = Fcpu / 64
MOV	A, #64H	; Set TC0C initial value = 64H
B0MOV	TC0C, A	; Set TC0 interval = 10 ms
B0BSET	FTC0IEN	; Enable TC0 interrupt service
B0BCLR	FTC0IRQ	; Clear TC0 interrupt request flag
B0BSET	FTC0ENB	; Enable TC0 timer
B0BSET	FGIE	; Enable GIE

➤ **Example: TC0 interrupt service routine.**

	ORG	8	; Interrupt vector
	JMP	INT_SERVICE	
INT_SERVICE:			
	...		; Push routine to save ACC and PFLAG to buffers.
	B0BTS1	FTC0IRQ	; Check TC0IRQ
	JMP	EXIT_INT	; TC0IRQ = 0, exit interrupt vector
	B0BCLR	FTC0IRQ	; Reset TC0IRQ
	MOV	A, #64H	; Reset TC0C.
	B0MOV	TC0C, A	; TC0 interrupt service routine
	...		
	...		
EXIT_INT:			
	...		; Pop routine to load ACC and PFLAG from buffers.
	RETI		; Exit interrupt vector

6.9 MULTI-INTERRUPT OPERATION

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

<i>Interrupt Name</i>	<i>Trigger Event Description</i>
P00IRQ	P0.0 trigger controlled by PEDGE
T0IRQ	T0C overflow
TC0IRQ	TC0C overflow

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

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

```

                                ORG            8                ; Interrupt vector
                                JMP            INT_SERVICE
INT_SERVICE:

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

INTP00CHK:                      ; Check INT0 interrupt request
                                B0BTS1         FP00IEN        ; Check P00IEN
                                JMP            INTT0CHK        ; Jump check to next interrupt
                                B0BTS0         FP00IRQ        ; Check P00IRQ
                                JMP            INTP00          ; Jump to INT0 interrupt service routine
INTT0CHK:                       ; Check T0 interrupt request
                                B0BTS1         FT0IEN         ; Check T0IEN
                                JMP            INTTC0CHK       ; Jump check to next interrupt
                                B0BTS0         FT0IRQ        ; Check T0IRQ
                                JMP            INTT0           ; Jump to T0 interrupt service routine
INTTC0CHK:                     ; Check TC0 interrupt request
                                B0BTS1         FTC0IEN        ; Check TC0IEN
                                JMP            INT_EXIT        ; Jump to exit of IRQ
                                B0BTS0         FTC0IRQ        ; Check TC0IRQ
                                JMP            INTTC0          ; Jump to TC0 interrupt service routine
INT_EXIT:

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

                                RETI                          ; Exit interrupt vector

```

7 I/O PORT

7.1 OVERVIEW

The micro-controller builds in 18 pin I/O. Some 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
P0.2	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P0.4	I/O	XOUT	AC	High_CLK code option = 32K, 4M, 12M
P0.3	I/O	XIN	AC	High_CLK code option = RC, 32K, 4M, 12M
P5.0	I/O	IRXO	DC	IRXEN=1, IRXO=1.
P5.4	I/O	IROUT	DC	IREN = 1.
		IRIN	DC	IRXEN=1.

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

7.2 I/O PORT MODE

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

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	P03M	-	P01M	P00M
Read/Write	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
After reset	0	0	0	0	0	-	0	0

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

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	-	P54M	-	-	-	P50M
Read/Write	-	-	-	R/W	-	-	-	R/W
After reset	-	-	-	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 input only pin, and the P0M.2 is undefined.

➤ Example: I/O mode selecting

```

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

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

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

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

7.3 I/O PULL UP REGISTER

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

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07R	P06R	P05R	P04R	P03R	-	P01R	P00R
Read/Write	W	W	W	W	W	-	W	W
After reset	0	0	0	0	0	-	0	0

0E10H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

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

➤ Example: I/O Pull up Register

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

7.4 I/O PORT DATA REGISTER

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

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

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	-	P54	-	-	-	P50
Read/Write	-	-	-	R/W	-	-	-	R/W
After reset	-	-	-	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 4
B0MOV      A, P5      ; Read data from Port 5

```

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

```

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

```

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

```

B0BSET     P1.0        ; Set P1.0 and P5.4 to be "1".
B0BSET     P5.4

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

```


8 TIMERS

8.1 WATCHDOG TIMER

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

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

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

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

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

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

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

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

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

Main:

```

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

```

- **Example: Clear watchdog timer by “@RST_WDT” macro of Sonix IDE.**

Main:

```

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

```

Watchdog timer application note is as following.

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

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

Main:

```
... ; Check I/O.
```

```
... ; Check RAM
```

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

Correct:

```
MOV A, #5AH ; I/O and RAM are correct. Clear watchdog timer and
B0MOV WDTR, A ; execute program.
; Clear the watchdog timer.
```

```
...
CALL SUB1
```

```
CALL SUB2
```

```
...
```

```
...
```

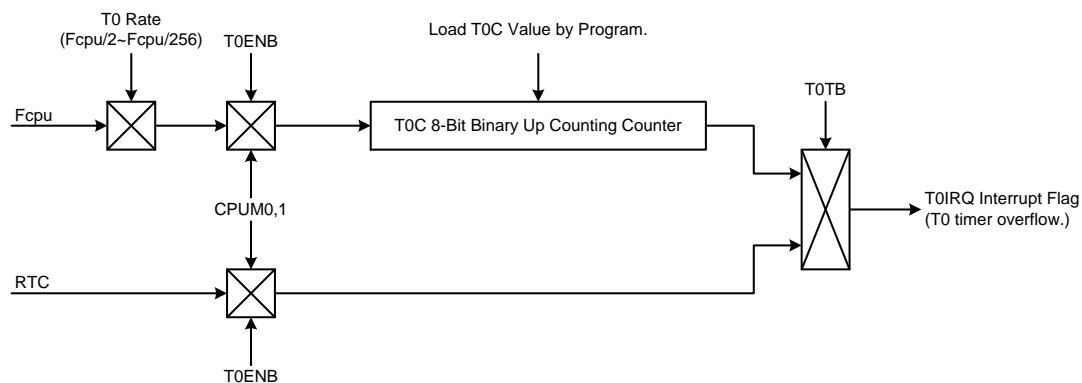
```
...
JMP MAIN
```

8.2 T0 8-BIT BASIC TIMER

8.2.1 OVERVIEW

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

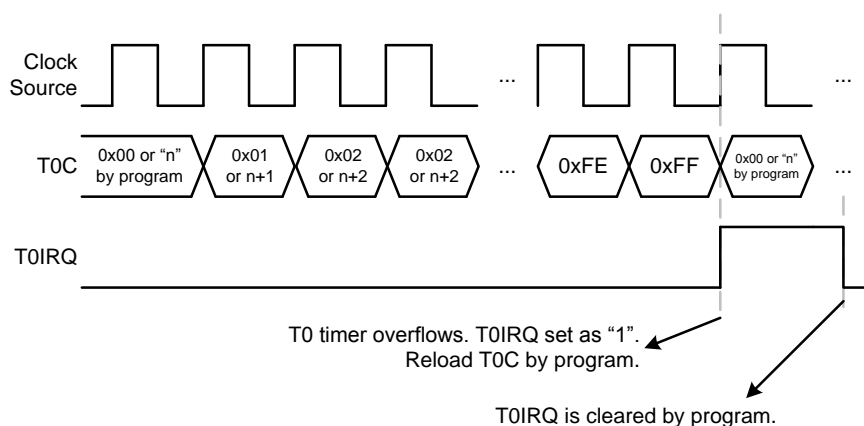
- ☞ **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- ☞ **Interrupt function:** T0 timer function supports interrupt function. When T0 timer occurs overflow, the T0IRQ activates and the system points program counter to interrupt vector to do interrupt sequence.
- ☞ **RTC function:** T0 supports RTC function. The RTC clock source is from external low speed 32K oscillator when T0TB=1. **RTC function is only available in High_Clk code option = "IHRC_RTC".**
- ☞ **Green mode function:** T0 timer keeps running in green mode and wakes up system when T0 timer overflows.



* **Note:** In RTC mode, the T0 interval time is fixed at 0.5 sec and T0C is 256 counts.

8.2.2 T0 Timer Operation

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



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

T0rate[2:0]	T0 Clock	T0 Interval Time					
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4		IHRC_RTC mode	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)	max. (sec)	Unit (ms)
000b	Fcpu/256	16.384	64	65.536	256	-	-
001b	Fcpu/128	8.192	32	32.768	128	-	-
010b	Fcpu/64	4.096	16	16.384	64	-	-
011b	Fcpu/32	2.048	8	8.192	32	-	-
100b	Fcpu/16	1.024	4	4.096	16	-	-
101b	Fcpu/8	0.512	2	2.048	8	-	-
110b	Fcpu/4	0.256	1	1.024	4	-	-
111b	Fcpu/2	0.128	0.5	0.512	2	-	-
-	32768Hz/64	-	-	-	-	0.5	1.953

8.2.3 T0M MODE REGISTER

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

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	T0TB	CPTS1	CPTS0	IRXEN
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 3 **T0TB**: RTC clock source control bit.
0 = Disable RTC (T0 clock source from Fcpu).
1 = Enable RTC.

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

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

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

8.2.4 T0C COUNTING REGISTER

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

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

The equation of T0C initial value is as following.

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

➤ **Example: To calculation T0C to obtain 10ms T0 interval time. T0 clock source is Fcpu = 4MHz/4 = 1MHz. Select T0RATE=001 (Fcpu/128).**

T0 interval time = 10ms. T0 clock rate = 4MHz/4/128

$$\begin{aligned} T0C \text{ initial value} &= 256 - (T0 \text{ interval time} * \text{input clock}) \\ &= 256 - (10\text{ms} * 4\text{MHz} / 4 / 128) \\ &= 256 - (10^{-2} * 4 * 10^6 / 4 / 128) \\ &= B2H \end{aligned}$$

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

8.2.5 T0 TIMER OPERATION EXPLAME

- **T0 TIMER CONFIGURATION:**

; Reset T0 timer.

```
MOV      A, #0x00      ; Clear T0M register.
BOBMV    T0M, A
```

; Set T0 clock source and T0 rate.

```
MOV      A, #0nnn0000b
BOBMV    T0M, A
```

; Set T0C register for T0 Interval time.

```
MOV      A, #value
BOBMV    T0C, A
```

; Clear T0IRQ

```
BOBCLR    FT0IRQ
```

; Enable T0 timer and interrupt function.

```
BOBSET    FT0IEN      ; Enable T0 interrupt function.
BOBSET    FT0ENB      ; Enable T0 timer.
```

- **T0 works in RTC mode:**

; Reset T0 timer.

```
MOV      A, #0x00      ; Clear T0M register.
BOBMV    T0M, A
```

; Set T0 RTC function.

```
BOBSET    FT0TB
```

; Clear T0C.

```
CLR      T0C
```

; Clear T0IRQ

```
BOBCLR    FT0IRQ
```

; Enable T0 timer and interrupt function.

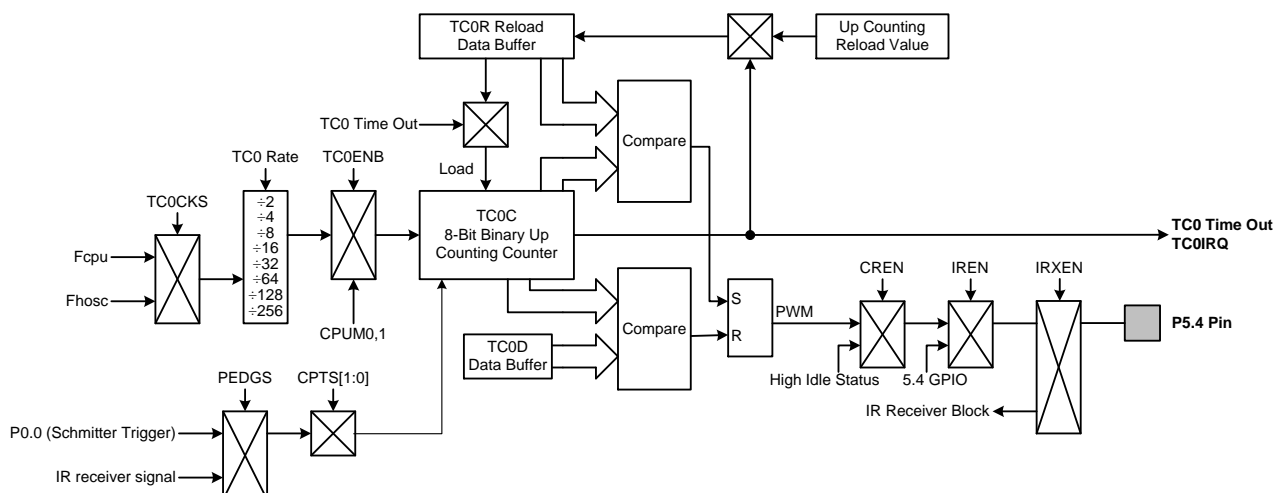
```
BOBSET    FT0IEN      ; Enable T0 interrupt function.
BOBSET    FT0ENB      ; Enable T0 timer.
```

8.3 TC0 8-BIT TIMER/DUTY-CYCLE PROGRAMMABLE IR DRIVER

8.3.1 OVERVIEW

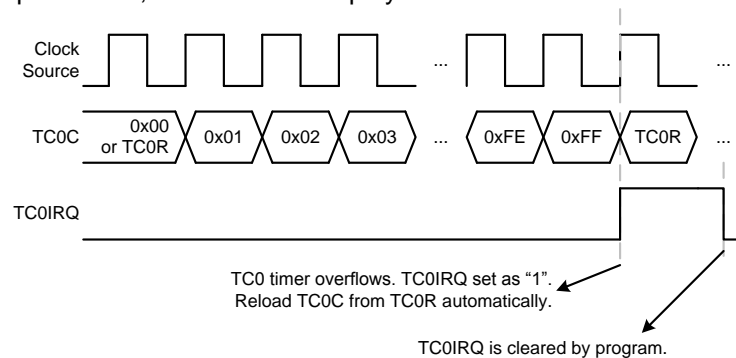
The TC0 timer is an 8-bit binary up timer and supports normal 8-bit timer, capture timer and PWM functions. If TC0 timer occurs an overflow (from 0xFF to 0x00), it will continue counting and issue a time-out signal to indicate TC0 time out event. TC0 builds in capture timer function to measure input signal's high pulse, low pulse or cycle. TC0 builds in PWM function. The PWM is duty/cycle programmable controlled by TC0R and TC0D registers. It is easy to implement buzzer, PWM and IR carry signal. TC0 counter supports auto-reload function which always enabled. When TC0 timer overflow occurs, the TC0C will be reloaded from TC0R automatically. The TC0 doesn't build in green mode wake-up function. The main purposes of the TC0 timer are as following.

- ☞ **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- ☞ **Interrupt function:** TC0 timer function supports interrupt function. When TC0 timer occurs overflow, the TC0IRQ activates and the system points program counter to interrupt vector to do interrupt sequence.
- ☞ **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by TC0R and TC0D registers. The PWM can be IR carry signal generator. **If IREN=1, TC0 is forced to PWM function, even CPTS[1:0] isn't zero.**
- ☞ **Capture Timer:** The capture timer function is an auto-start-stop timer function triggered by signal's edge changing. The capture timer can measure IR input carry signal.
- ☞ **Green mode function:** All TC0 functions (Timer, PWM, capture timer, auto-reload) keeps running in green mode, but no wake-up function. Timer IRQ actives as any IRQ trigger occurrence, e.g. timer overflow...



8.3.2 TC0 TIMER OPERATION

TC0 timer is controlled by TC0ENB bit. When TC0ENB=0, TC0 timer stops. When TC0ENB=1, TC0 timer starts to count. Before enabling TC0 timer, setup TC0 timer's configurations to select timer function modes, e.g. basic timer, interrupt function...TC0C increases "1" by timer clock source. When TC0 overflow event occurs, TC0IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is TC0C count from full scale (0xFF) to zero scale (0x00). In difference function modes, TC0C value relates to operation. If TC0C value changing effects operation, the transition of operations would make timer function error. So TC0 builds in double buffer to avoid these situations happen. The double buffer concept is to flash TC0C during TC0 counting, to set the new value to TC0R (reload buffer), and the new value will be loaded from TC0R to TC0C after TC0 overflow occurrence automatically. In the next cycle, the TC0 timer runs under new conditions, and no any transitions occur. The auto-reload function is no any control interface and always actives as TC0 enables. If TC0 timer interrupt function is enabled (TC0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 0008H) and executes interrupt service routine after TC0 overflow occurrence. Clear TC0IRQ by program is necessary in interrupt procedure. TC0 timer can works in normal mode, slow mode and green mode. But in green mode, TC0 keep counting, set TC0IRQ and outputs PWM, but can't wake-up system.



TC0 provides different clock sources to implement different applications and configurations. TC0 clock source includes Fcpu (instruction cycle) and Fhosc (high speed oscillator) controlled by TC0CKS bit. TC0CKS bit selects the clock source is from Fcpu or Fhosc. If TC0CKS=0, TC0 clock source is Fcpu through TC0rate[2:0] pre-scalar to decide Fcpu/2~Fcpu/256. If TC0CKS=1, TC0 clock source is Fhosc through TC0rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/16. TC0 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

TC0CKS	TC0rate[2:0]	TC0 Clock	TC0 Interval Time			
			Fhosc=8MHz, Fcpu=Fhosc/2		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/256	16.384	64	65.536	256
0	001b	Fcpu/128	8.192	32	32.768	128
0	010b	Fcpu/64	4.096	16	16.384	64
0	011b	Fcpu/32	2.048	8	8.192	32
0	100b	Fcpu/16	1.024	4	4.096	16
0	101b	Fcpu/8	0.512	2	2.048	8
0	110b	Fcpu/4	0.256	1	1.024	4
0	111b	Fcpu/2	0.128	0.5	0.512	2
1	000b	Fhosc/1	0.032	0.125	0.064	0.25
1	001b	Fhosc/2	0.064	0.25	0.128	0.5
1	010b	Fhosc/4	0.128	0.5	0.256	1
1	011b	Fhosc/8	0.256	1	0.512	2
1	100b	Fhosc/16	0.512	2	1.024	4

8.3.3 TC0M MODE REGISTER

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

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	IRSTS	IREN	CREN
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 **TC0ENB**: TC0 timer control bit.

0 = Disable.

1 = Enable.

Bit [6:4] **TC0RATE [2:0]**: TC0 timer clock source select bits.

TC0CKS=0 -> 000=Fcpu/256, 001=Fcpu/128, 010=Fcpu/64, 011=Fcpu/32, 100=Fcpu/16, 101=Fcpu/8, 110=Fcpu/4, 111=Fcpu/2.

TC0CKS=1 -> 000=Fhosc/1, 001=Fhosc/2, 010=Fhosc/4, 011=Fhosc/8, 100~111=Fhosc/16.

Bit 3 **TC0CKS**: TC0 timer clock source control bit.

0 = Fcpu.

1 = Fhosc.

Bit 2 **IRSTS**: IR signal status indicator bit. (**IRXEN must be enable**)

0 = Logic low.

1 = Logic high.

Bit 1 **IREN**: IR output function control bit.

0 = Disable. P5.4 is GPIO mode. TC0 is normal timer or capture functions.

1 = Enable. P5.4 is IR output mode, and the idle status is output high. TC0ENB=1 is necessary and starts to count by CREN.

Bit 0 **CREN**: IR carry signal output control bit. (**IREN must be enable**)

0 = Disable. P5.4 is keeps IR idle status.

1 = Enable. P5.4 outputs IR carry signal.

8.3.4 TC0C COUNTING REGISTER

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

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

The equation of TC0C initial value is as following.

$$TC0C \text{ initial value} = 256 - (TC0 \text{ interrupt interval time} * TC0 \text{ clock rate})$$

8.3.5 TC0R AUTO-RELOAD REGISTER

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

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

The equation of TC0R initial value is as following.

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

- **Example: To calculation TC0C and TC0R value to obtain 10ms TC0 interval time. TC0 clock source is Fcpu = 4MHz/4= 1MHz. Select TC0RATE=001 (Fcpu/128).**

TC0 interval time = 10ms. TC0 clock rate = 4MHz/4/128

$$\begin{aligned}
 TC0C/TC0R \text{ initial value} &= 256 - (TC0 \text{ interval time} * \text{input clock}) \\
 &= 256 - (10\text{ms} * 4\text{MHz} / 4 / 128) \\
 &= 256 - (10^{-2} * 4 * 10^6 / 4 / 128) \\
 &= B2H
 \end{aligned}$$

8.3.6 TC0D PWM DUTY REGISTER

TC0D register's purpose is to decide PWM duty. In PWM mode, TC0R controls PWM's cycle, and TC0D controls the duty of PWM. The operation is base on timer counter value. When TC0C = TC0D, the PWM low duty finished and exchange to high level. It is easy to configure TC0D to choose the right PWM's duty for application.

0E8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0D	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.

$$TC0D \text{ initial value} = TC0R + (PWM \text{ high pulse width period} / TC0 \text{ clock rate})$$

- **Example: To calculate TC0D value to obtain 1/3 duty PWM signal. The TC0 clock source is Fcpu = 8MHz/1= 8MHz. Select TC0RATE=111 (Fcpu/2).**

TC0R = 97H. TC0 interval time = 26.25us. So the PWM cycle is 38 KHz. In 1/3 duty condition, the low pulse width is about 8.75us.

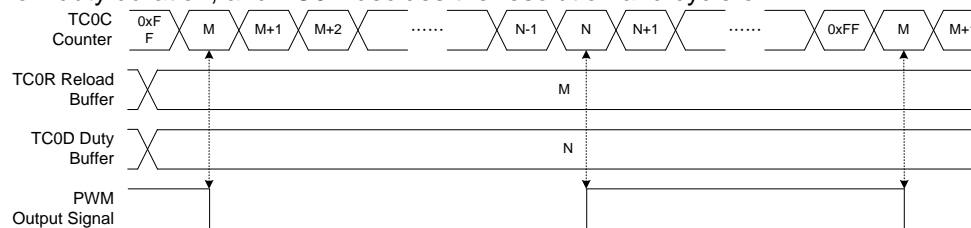
$$\begin{aligned}
 TC0D \text{ initial value} &= 97H + (PWM \text{ high pulse width period} / TC0 \text{ clock rate}) \\
 &= 97H + (8.75\text{us} * 8\text{MHz} / 1 / 2) \\
 &= 97H + 23H \\
 &= BAH
 \end{aligned}$$

Common IR signal table. TC0 clock rate is 4MHz.

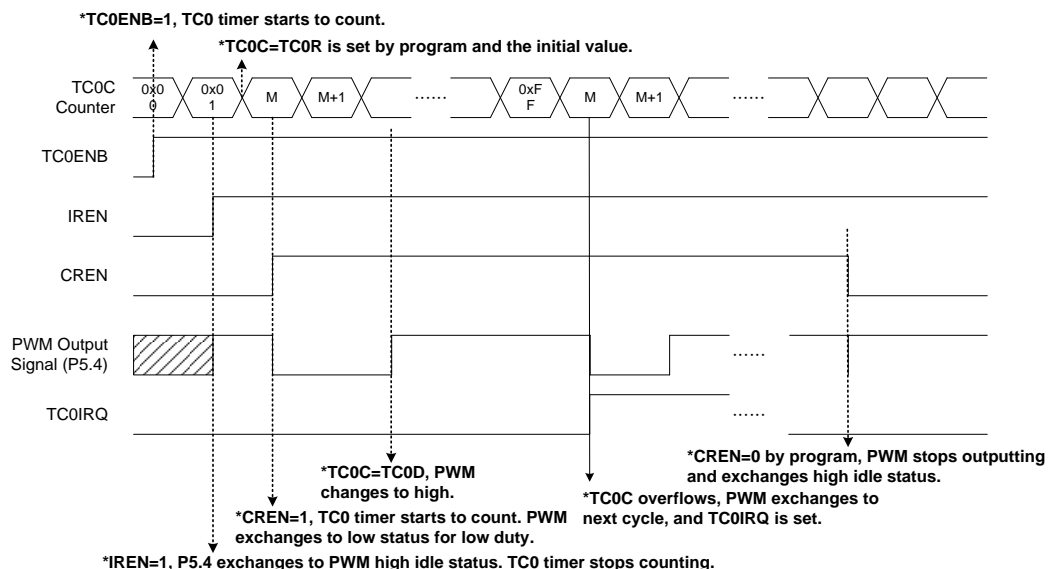
IR Freq. (KHz)	TC0C TC0R		TC0D						Freq.Error Rate
			1/2 duty		1/3 duty		1/4 duty		
	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	
32	131	83	193.50	C1	172.67	AC	162.25	A2	0.00%
36	145	91	200.50	C8	182.00	B6	172.75	AC	0.10%
38	151	97	203.50	CB	186.00	BA	177.25	B1	0.25%
39.2	154	9A	205.00	CD	188.00	BC	179.50	B3	0.04%
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%

8.3.7 PULSE WIDTH MODULATION (PWM)

TC0 timer builds in PWM function, and the PWM idle status is high. PWM output pin is P5.4. TC0's PWM is controlled by IREN bit and CREN bit. When IREN=1, the PWM function is enabled, and P5.4 is switched from GPIO to PWM output status (idle status = high). When IREN=0, PWM output pin returns to GPIO last status. PWM signal is generated from the result of TC0C, TC0R and TC0D comparison. When CREN=1 or TC0C counts from 0xFF to 0x00 (overflow), the PWM outputs low status which is the PWM initial status. TC0C is loaded new data from TC0R register to decide PWM cycle and resolution. TC0C keeps counting, and the system compares TC0C and TC0D. When TC0C=TC0D, the PWM output status exchanges to high. TC0C keeps counting. When TC0 timer overflow occurs, and one cycle of PWM signal finishes. TC0C is reloaded from TC0R automatically, and PWM output status exchanges to low for next cycle. TC0D decides the low duty duration, and TC0R decides the resolution and cycle of PWM.



The resolution of PWM is decided by TC0R. TC0R range is from 0x00~0xFF. If TC0R = 0x00, PWM's resolution is 1/256. If TC0R = 0x80, PWM's resolution is 1/128. TC0D controls the low pulse width of PWM for PWM's duty. When TC0C = TC0D, PWM output exchanges to high status. TC0D must be greater than TC0R, or the PWM signal keeps high status. When PWM outputs, TC0IRQ still activates as TC0 overflows, and TC0 interrupt function activates as TC0IEN = 1. But strongly recommend be careful to use PWM and TC0 timer together, and make sure both functions work well. The PWM output pin is shared with GPIO and switch to output PWM signal as IREN=1 automatically. If IREN bit is cleared to disable PWM, the output pin exchanges to last GPIO mode automatically. It easily to implement carry signal on/off operation, not to control TC0ENB bit.



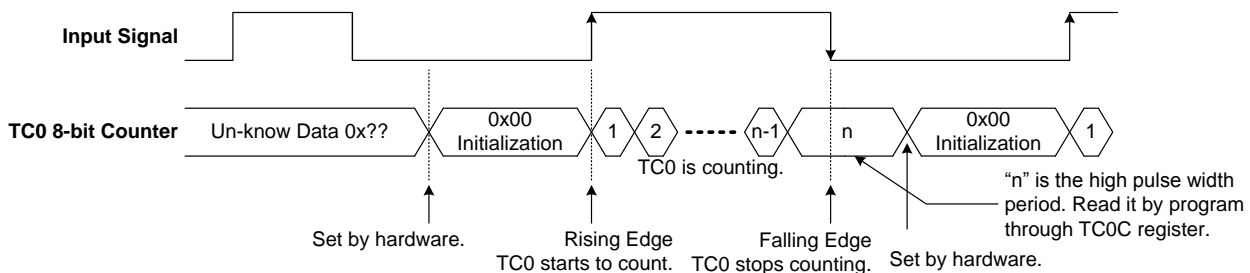
8.3.8 TC0 CAPTURE TIMER

TC0 timer builds in capture timer function. It is a simple method to measure high pulse width, low pulse width and cycle. TC0 capture timer is controlled by CPTS[1:0] bits. When CPTS[1:0]=00b, TC0 capture timer is disabled. When CPTS[1:0] isn't zero, TC0 capture timer is enabled and TC0 counter is controlled by the edge changing of input signal. The measured signal for capture timer has two sources controlled by PEDGS. If PEDGS = 0, capture timer signal is P0.0. If PEDGS = 1, capture timer signal is from IR receiver.

- CPTS[1:0] = 01: High pulse width measurement
- CPTS[1:0] = 10: Low pulse width measurement.
- CPTS[1:0] = 11: Input cycle measurement.

8.3.8.1 High Pulse Width Measurement

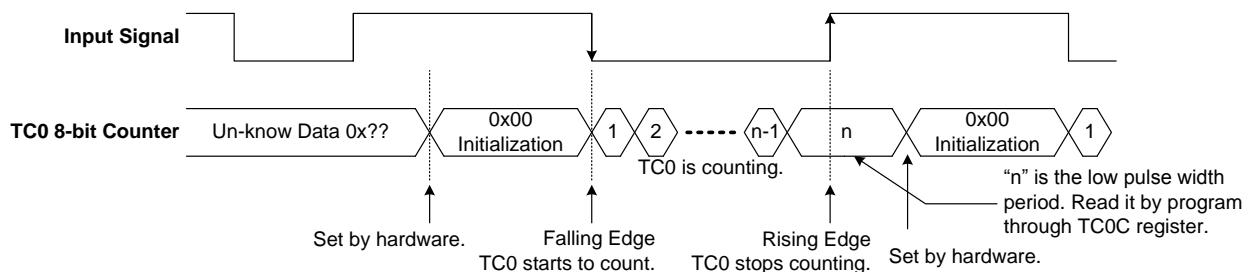
TC0ENB = 1. CPTS[1:0] = 01.



The high pulse width measurement is using rising edge to start TC0 8-bit timer and falling edge to stop TC0 8-bit timer. If set CPTS[1:0] = 01, the capture timer will measure high pulse until the rising edge occurrence. When the end of measuring high pulse width and TC0 timer stops, the TC0IRQ sets as "1", and the TC0 interrupt executes as TC0IEN = 1.

8.3.8.2 Low Pulse Width Measurement

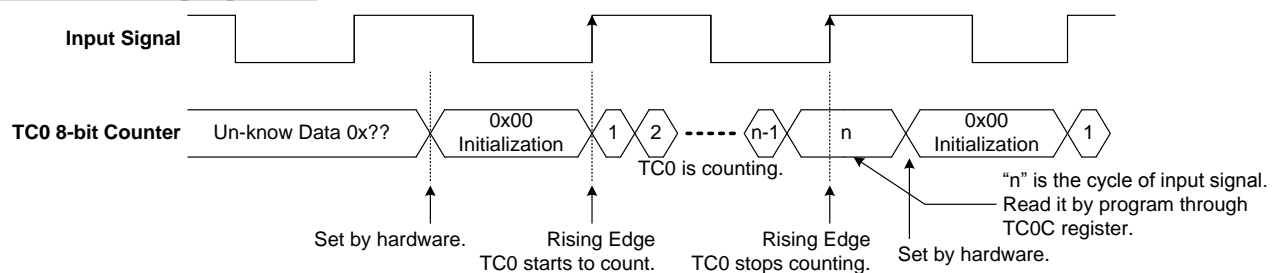
TC0ENB = 1. CPTS[1:0] = 10.



The low pulse width measurement is using falling edge to start TC0 8-bit timer and rising edge to stop TC0 8-bit timer. If set CPTS[1:0] = 10, the capture timer will measure low pulse until the falling edge occurrence. When the end of measuring low pulse width and TC0 timer stops, the TC0IRQ sets as "1", and the TC0 interrupt executes as TC0IEN = 1.

8.3.8.3 Input Cycle Measurement

TC0ENB = 1. CPTS[1:0] = 11.



The cycle measurement is using rising to start TC0 8-bit timer and rising edge to stop TC0 8-bit timer. If set CPTS[1:0] = 11, the capture timer will measure cycle until the rising edge occurrence. When the end of measuring cycle and TC0 timer stops, the TC0IRQ sets as "1", and the TC0 interrupt executes as TC0IEN = 1.

8.3.9 CAPTURE TIMER CONTROL REGISTER

D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	T0TB	CPTS1	CPTS0	IRXEN
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 [2:1] **CPTS[1:0]**: TC0 capture function control bit. **The capture source is controlled by PEDGS bit.**
 00 = Disable. 01=High pulse width measurement. 10=Low pulse width measurement. 11=Cycle measurement.
- Bit 0 **IRXEN**: IR receiver control bit.
 0 = Disable. P5.4 is GPIO mode.
 1 = Enable. P5.4 is input mode.

8.3.10 TC0 TIMER OPERATION EXAMPLE

● TC0 TIMER CONFIGURATION:

; Reset TC0 timer.

```
CLR          TC0M          ; Clear TC0M register.
```

; Set TC0 clock source and TC0 rate.

```
MOV          A, #0nnn0n00b
B0MOV        TC0M, A
```

; Set TC0C and TC0R register for TC0 Interval time.

```
MOV          A, #value      ; TC0C must be equal to TC0R.
B0MOV        TC0C, A
B0MOV        TC0R, A
```

; Clear TC0IRQ

```
B0BCLR       FTC0IRQ
```

; Enable TC0 timer and interrupt function.

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

● TC0 PWM CONFIGURATION:

; Reset TC0 timer.

```
CLR          TC0M          ; Clear TC0M register.
```

; Set TC0 clock source and TC0 rate.

```
MOV          A, #0nnn0n00b
B0MOV        TC0M, A
```

; Set TC0C and TC0R register for PWM cycle.

```
MOV          A, #value1     ; TC0C must be equal to TC0R.
B0MOV        TC0C, A
B0MOV        TC0R, A
```

; Set TC0D register for PWM duty.

```
MOV          A, #value2     ; TC0D must be greater than TC0R.
B0MOV        TC0D, A
```

; Set PWM carry signal output.

```
B0BSET       FIREN         ; Set IROUT pin to PWM carry output function.
B0BSET       FCREN         ; Set PWM carry signal output.
```

; Enable PWM and TC0 timer.

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

● TC0 CAPTURE TIMER CONFIGURATION:

; Reset TC0 timer.

```
CLR          TC0M          ; Clear TC0M register.
```

; Set TC0 clock rate and clock source.

```
MOV          A, #0nnnm000b ; "nnn" is TC0rate[2:0] for TC0 clock rate selection.
B0MOV        TC0M, A        ; "m" is TC0 clock source control bit.
MOV          A, #00000mm0b  ; "mm" is CPTS[1:0] for TC0 capture timer function selection.
B0MOV        T0M, A         ; CPTS[1:0] = 01b/10b/11b, enable high pulse width, low
                             pulse width or cycle measurement.
```

; Select capture timer source.

```
B0BCLR       FPDGGS        ; Source is P0.0.
```

; or

```
B0BSET       FPDGGS        ; Source is IR input signal.
```

; Clear TC0IRQ

```
B0BCLR       FTC0IRQ
```

; Enable TC0 capture timer function and interrupt function.

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

9 IR TRANSMITTER/RECEIVER

9.1 OVERVIEW

IR signal is generated by TC0 timer. The IR output pin is 400mA @Vss+0.5V sink type. When IREN bit of TC0M is set as logic "1", IROUT pin exchanges from GPIO to IR output mode. If CREN = 0 or system is in power down mode, IROUT pin is tied to high status. The TC0 timer 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's cycle and TC0D decides IR's duty. When enable IR output function (CREN=1), IR output status is low level. TC0C initial value is TC0R and starts to count. When TC0C=TC0D, IR output status change to high level and finishes low duty operation. When TC0C overflow occurs (TC0C changes from 0xFF to 0x00), IR output high duty operation stops. System loads TC0R into TC0C automatically and next cycle starts. SN8PC22 also builds in IR receiver circuit for receiving IR carry signal and IR diode must be parallel 33K ohm resistor. IR receiver function is controlled by IRXEN bit and receiver signal is from IRIN pin. When IRXEN = 0, IR receiver function is disabled and IRIN pin exchanges to GPIO or PWM mode. When IRXEN = 1, IR receiver function is enabled and IRIN pin is receiver signal pin.

*** Note:**

1. The system forces IROUT exchanging to high status in power down mode automatically.
2. Set TC0C=TC0R before IR output enable to make sure the first cycle correct.
3. In IR receiver mode, IR diode must be parallel 33K ohm resistor, but in IR transmitter mode does not need.

9.2 IR CONTROL REGISTER

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	IRSTS	IREN	CREN
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 2 **IRSTS:** IR signal status indicator bit. (**IRXEN must be enable**)

0 = Logic low.

1 = Logic high.

Bit 1 **IREN:** IR output function control bit.

0 = Disable. P5.4 is GPIO mode. TC0 is normal timer or capture functions.

1 = Enable. P5.4 is IR output mode, and the idle status is output high. TC0ENB=1 is necessary and starts to count by CREN.

Bit 0 **CREN:** IR carry signal output control bit. (**IREN must be enable**)

0 = Disable. P5.4 is keeps IR idle status.

1 = Enable. P5.4 outputs IR carry signal.

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

D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	T0TB	CPTS1	CPTS0	IRXEN
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 **IRXEN**: IR receiver control bit.
0 = Disable. P5.4 is GPIO mode.
1 = Enable. P5.4 is input mode.

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

Bit 1 **IRXO**: IR receiver output pin control bit. (**IRXEN must be enable**)
0 = Disable. P5.0 is GPIO mode.
1 = Enable. P5.0 is IR receiver output signal.

Bit 0 **PEDGS**: Trigger edge source control bit.
0 = P0.0.
1 = IR input signal.

9.3 IR TRANSMITTER/RECEIVER OPERATION SEQUENCE

➤ Example: IR transmitter operation sequence.

```
; Set TC0C and TC0R for IR cycle.
    MOV        A, #IRCYCVAL      ; TC0C, TC0R value for IR cycle.
    MOV        TC0C, A           ;
    MOV        TC0R, A
; Set TC0D for IR duty.
    MOV        A, #IRDUTYVAL     ; TC0D value for IR duty.
    MOV        TC0D, A
; Enable IR output.
    B0BSET     FIREN             ; Set IROUT pin to IR carry output function.
    B0BSET     FCREN            ; Set IR carry signal output.
    B0BSET     FTC0ENB          ; Enable TC0 timer.
```

➤ Example: IR receiver operation sequence.

```
; Set P5.0 is GPIO or IR receiver output signal.
    B0BCLR     FIRXO             ; P5.0 is GPIO mode.
; or
    B0BSET     FIRXO             ; P5.0 is IR receiver output signal.
; Enable IR input.
    B0BSET     FIRXEN            ; Set IRIN pin to IR carry input function.

; Set TC0 Capture timer.
    MOV        A, #0nnnm000b     ; "nnn" is TC0rate[2:0] for TC0 clock rate selection.
    B0MOV      TC0M, A           ; "m" is TC0 clock source control bit.

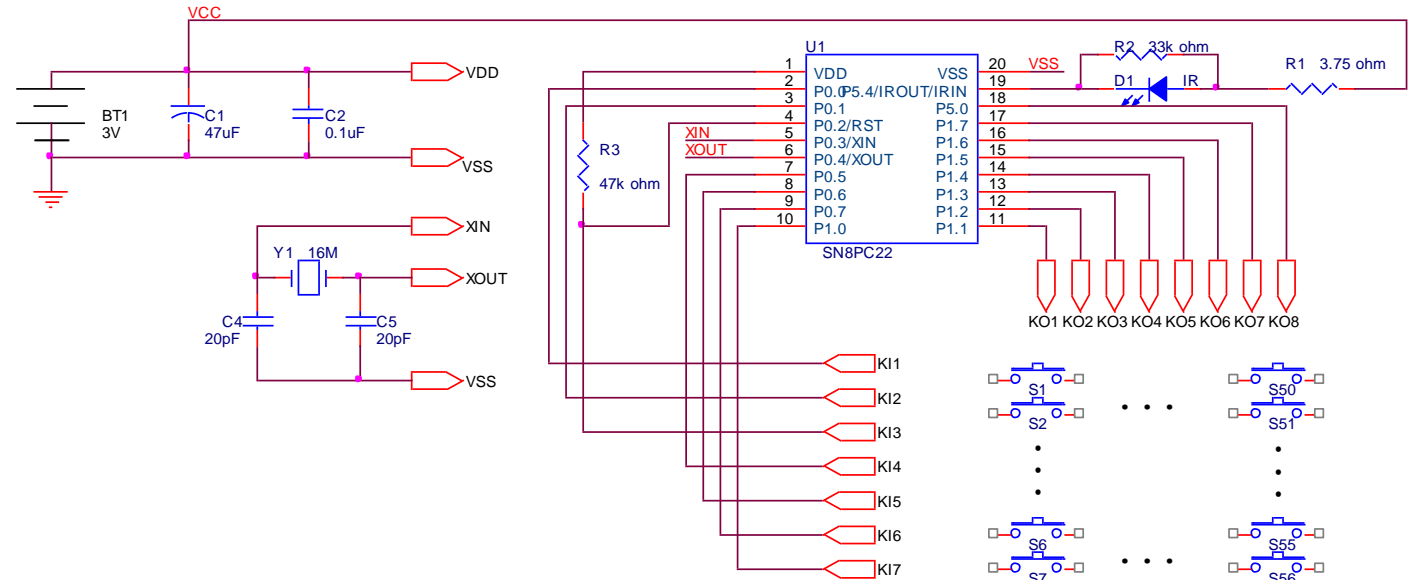
    MOV        A, #00000mm0b     ; "mm" is CPTS[1:0] for TC0 capture timer function selection.
    OR         T0M, A            ; CPTS[1:0] = 01b/10b/11b, enable high pulse width, low
                                ; pulse width or cycle measurement.
; Source is IR input signal.
    B0BSET     FPEDGS            ; Source is IR input signal.
; Clear TC0IRQ
    B0BCLR     FTC0IRQ

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

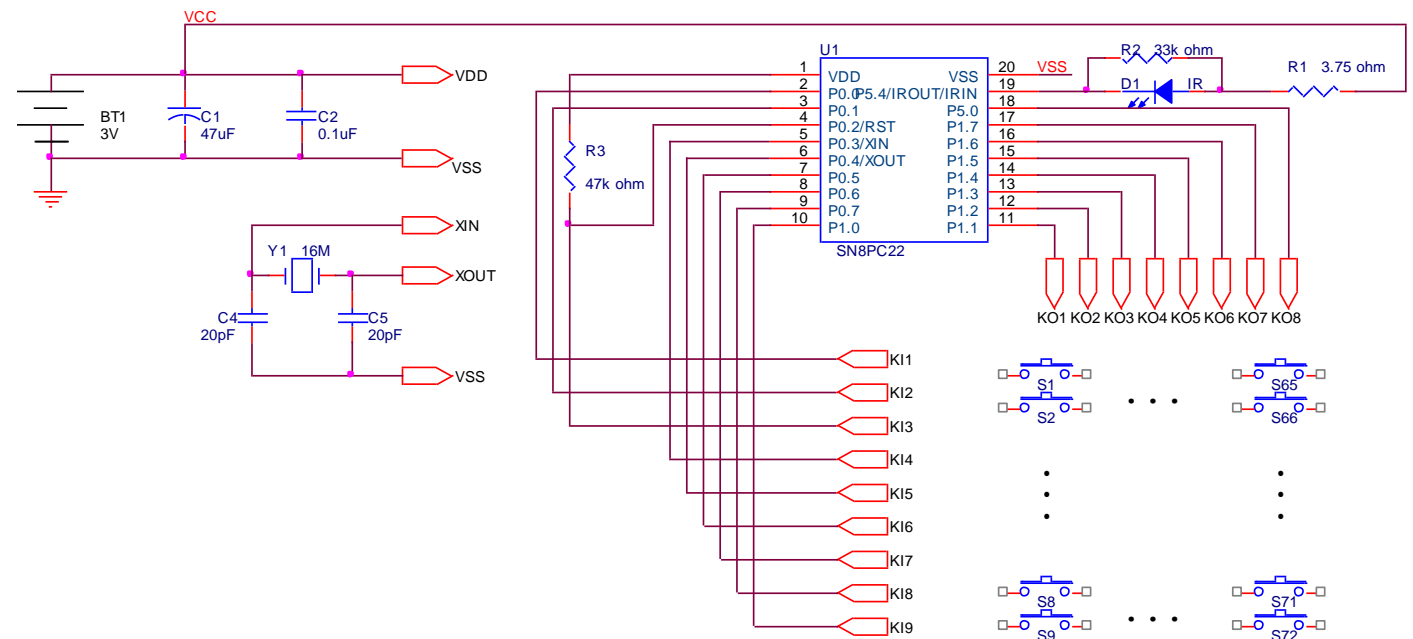

9.4 IR APPLICATION CIRCUIT

The IR sink current is 400mA @Vss+1.5V. The resistance of IR drive circuit is 3.75 ohm @Vdd=3V. The resistance can't lower than 3.75 ohm, or the sink current is over specification. The R2 33K ohm is necessary for IR receiver mode, but in transmitter mode isn't necessary.

Crystal type, Max. 56 keys:



IHRC_8M type, Max. 72 keys:



10 INSTRUCTION TABLE

Field	Mnemonic	Description	C	DC	Z	Cycle
MOV O V E	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
A R I T H M E T I C	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
L O G I C	AND A,M	$A \leftarrow A$ and M	-	-	✓	1
	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
P R O C E S S	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
B R A N C H	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	$PC15/14 \leftarrow RomPages1/0, PC13-PC0 \leftarrow d$	-	-	-	2
	CALL d	Stack $\leftarrow PC15-PC0, PC15/14 \leftarrow RomPages1/0, PC13-PC0 \leftarrow d$	-	-	-	2
M I S C	RET	$PC \leftarrow Stack$	-	-	-	2
	RETI	$PC \leftarrow Stack$, and to enable global interrupt	-	-	-	2
	PUSH	To push ACC and PFLAG (except NT0, NPD bit) into buffers.	-	-	-	1
	POP	To pop ACC and PFLAG (except NT0, NPD bit) from buffers.	✓	✓	✓	1
	NOP	No operation	-	-	-	1

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

11 ELECTRICAL CHARACTERISTIC

11.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd).....	- 0.3V ~ 6.0V
Input in voltage (Vin).....	Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr) SN8PC22P, SN8PC22S, SN8PC22X.....	0°C ~ + 70°C
Storage ambient temperature (Tstor)	–40°C ~ + 125°C

11.2 ELECTRICAL CHARACTERISTIC

● DC CHARACTERISTIC

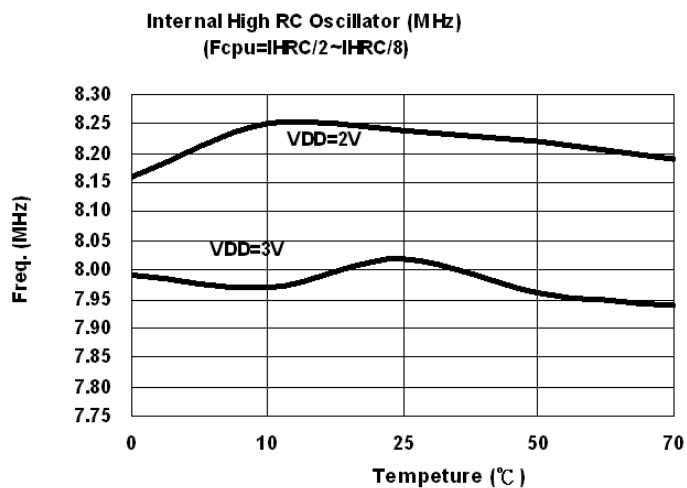
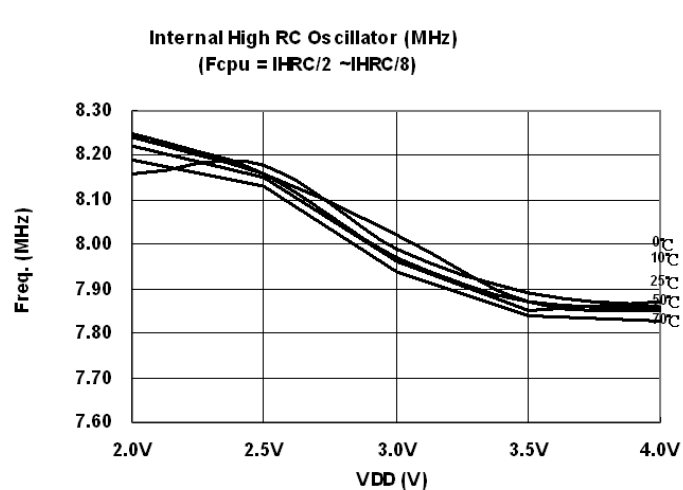
(All of voltages refer to Vss, Vdd = 3.0V, fosc = 4MHz, fcpu=1MHz, ambient temperature is 25°C unless otherwise note.)

(All of voltages refer to Vss, Vdd = 3.0V, Iosc = 4mA, Icpu = 1mA, 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	Xin/Xout		Vss	-	0.4Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Xin/Xout		0.6Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
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
	IoL1	Vop = Vss + 0.5V		8	12	-	
sink current	IoL2	Vop = Vss + 1.5V, IR output pin		300	400	-	mA
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, 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 Hihg RC (IHRC)	25°C, Vdd= 3V, Fcpu = 1MHz	7.84	8	8.16	Mhz
LVD Voltage	Vdet0	Low voltage reset level.		-	-	1.8	V

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

11.3 CHARACTERISTIC GRAPHS

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



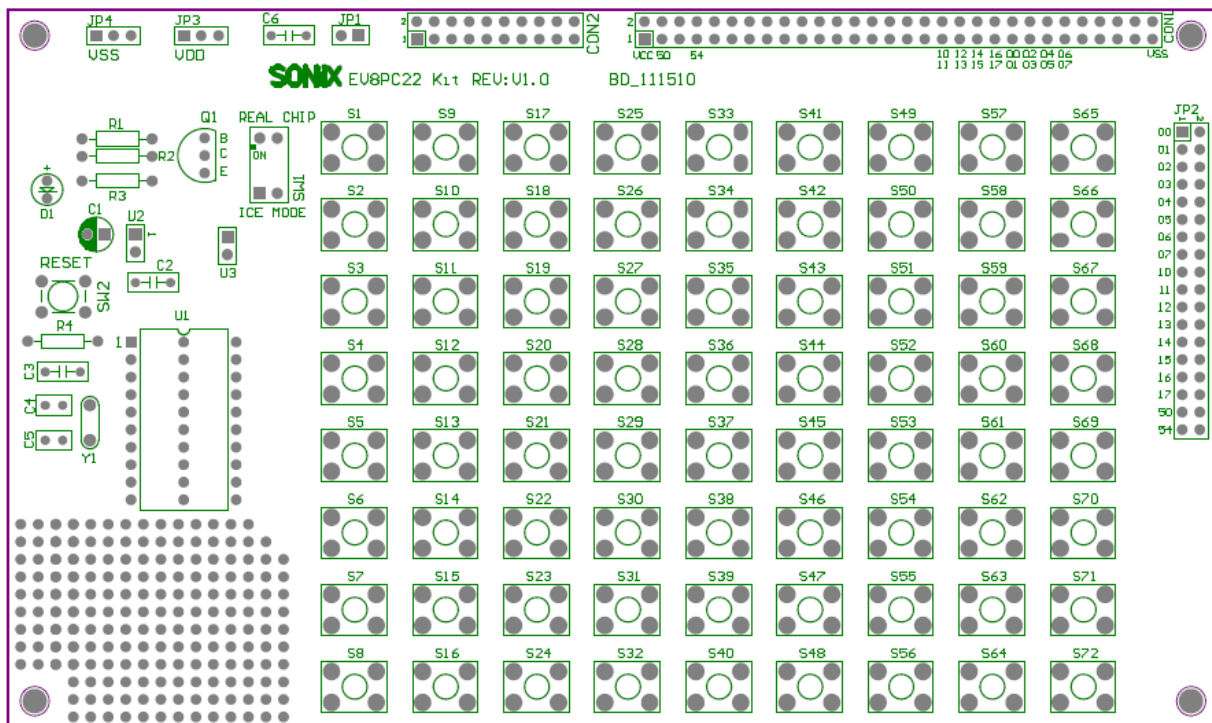
12 DEVELOPMENT TOOL

SONiX provides ICE (in circuit emulation), IDE (Integrated Development Environment) and EV-KIT for SN8PC22 development. ICE and EV-KIT are external hardware devices, and IDE is a friendly user interface for firmware development and emulation. These development tools' version is as following.

- ICE: SN8ICE2K Plus 2.
- EV-KIT: EV8PC22 KIT V1.0.
- IDE: SONiX IDE M2IDE_V129.
- Writer: MP PRO.

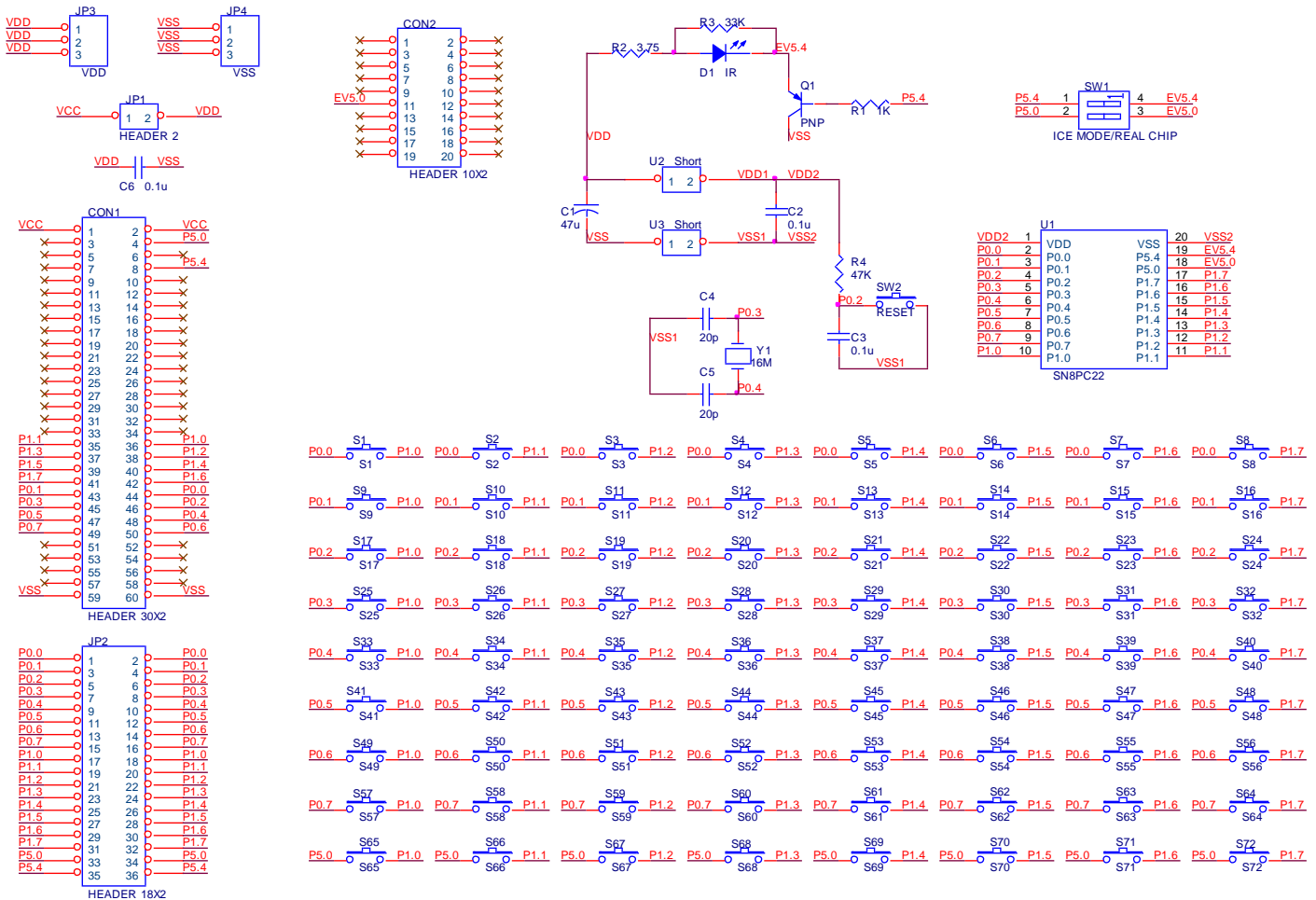
12.1 SN8PC22 EV-KIT

SN8PC22 EV-KIT includes ICE interface, GPIO interface and IR transmitter/receiver driver module. The schematic of SN8PC22 EV-KIT is as following.



- CON1, CON2: ICE interface connected to SN8ICE2K Plus 2.
- JP1: EV-KIT power connector between VCC and VDD. VCC is the power source from SN8ICE2K Plus 2. VDD is the power of KV-KIT.
- JP2: GPIO connector.
- U1: SN8PC22 EV-Chip for IR receiver emulation or SN8PC22 DIP and SSOP type connector for connecting to user's target board.
- U2/U3: Internal short connector. Don't care them as emulating.
- S1~S72: Remote key map.
- D1/R1/R2/R3/Q1: IR transmitter/receiver driving circuit for ICE emulation and real chip test.
- SW1: ICE mode and real chip mode control switch.

● SN8PC22 EV-KIT Circuit



12.2 ICE AND EV-KIT APPLICATION NOTIC

SN8PC22 EV-KIT includes matrix type key map and IR transmitter/receiver driving circuit module.

- The IR transmitter/receiver driving circuit module includes two types. One is for ICE emulation to drive IR carry signal and SW1 must switch to ICE mode. The other one is for SN8PC22 real chip test, SW1 must switch to real chip mode and disconnected with ICE.
- The EV-KIT is like a real remote controller. The R2 resistance is over 3.75Ω better for 400mA sink current @Vdd=3V. The R3 resistance typical value is 33KΩ .

13 OTP PROGRAMMING PIN

13.1 PROGRAMMING PIN MAPPING

Programming Pin Information of SN8PC22									
Chip Name		SN8PC22P/S/X							
MP PRO Writer Connector		OTP IC Pin Assignment							
JP3 Number	JP3 Name	Number	Pin	Number	Pin	Number	Pin		
1	VDD	1	VDD						
2	GND	20	VSS						
3	CLK	12	P1.2						
4	CE	-	-						
5	PGM	10	P1.0						
6	OE	13	P1.3						
7	D1	-	-						
8	D0	-	-						
9	D3	-	-						
10	D2	-	-						
11	D5	-	-						
12	D4	-	-						
13	D7	-	-						
14	D6	-	-						
15	VDD	-	-						
16	VPP	4	RST						
17	HLS	-	-						
18	RST	-	-						
19	-	-	-						
20	ALSB/PDB	11	P1.1						

*** Note:**

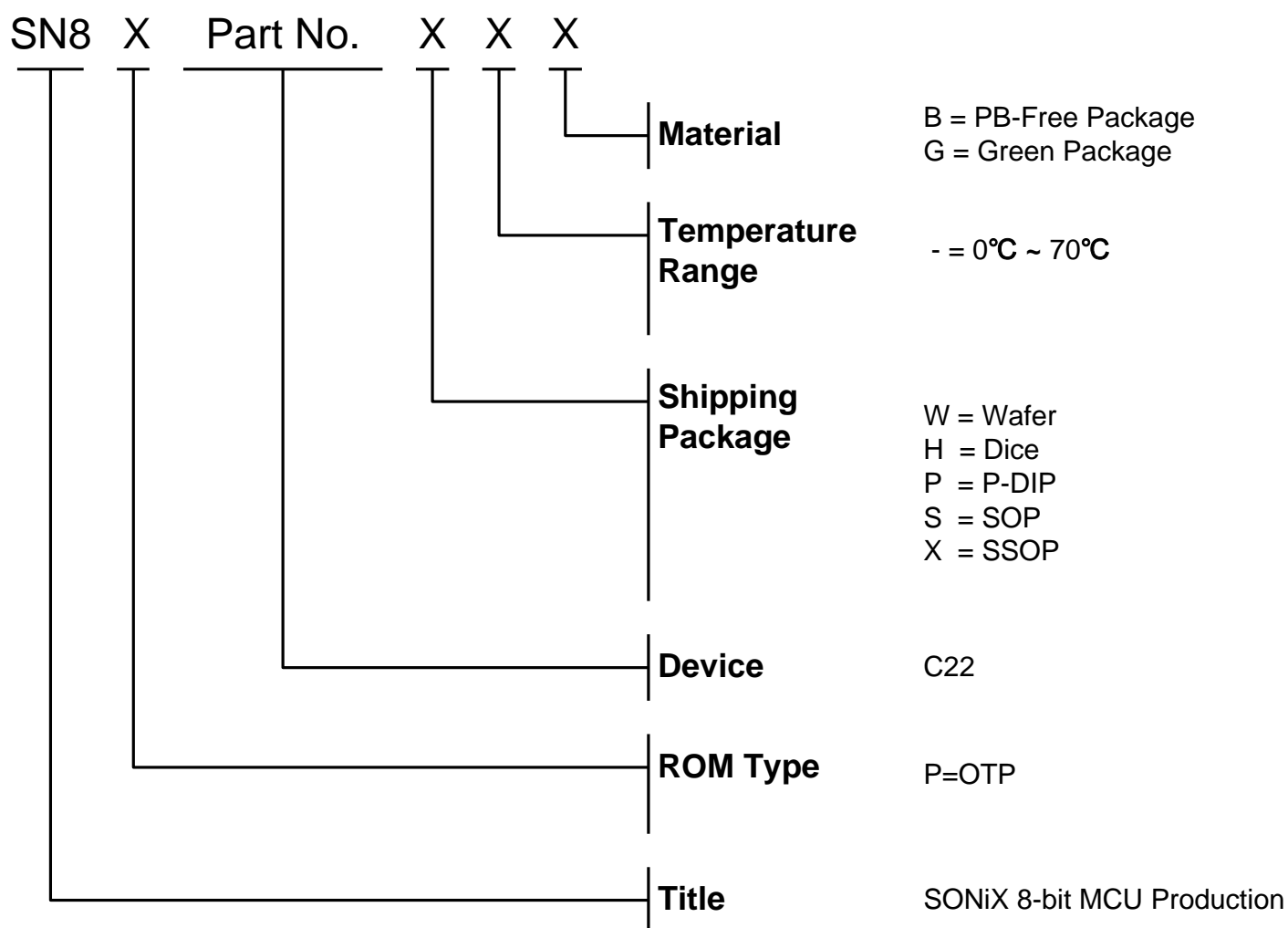
1. SN8PC22 only support MP PRO Writer not support MPIII Writer-LV.
2. Refer to MP-PRO Writer Easy Guide.

14 Marking Definition

14.1 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.

14.2 MARKING INDETIFICATION SYSTEM



* Note: SN8PC22 doesn't support -40°C~85°C temperature range.

14.3 MARKING EXAMPLE

- **Wafer, Dice:**

Name	ROM Type	Device	Package	Temperature	Material
S8PC22W	OTP	C22	Wafer	0°C ~70°C	-
SN8PC22H	OTP	C22	Dice	0°C ~70°C	-

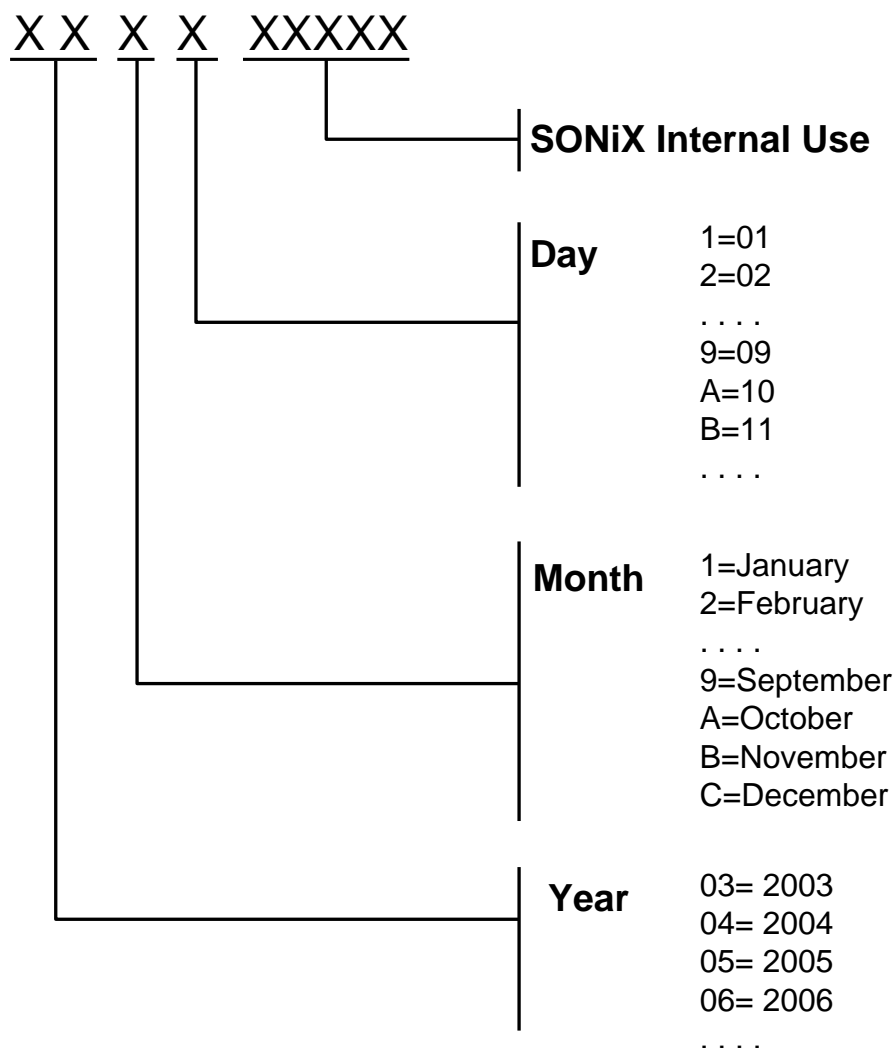
- **Green Package:**

Name	ROM Type	Device	Package	Temperature	Material
SN8PC22PG	OTP	C22	P-DIP	0°C ~70°C	Green Package
SN8PC22SG	OTP	C22	SOP	0°C ~70°C	Green Package
SN8PC22XG	OTP	C22	SSOP	0°C ~70°C	Green Package

- **PB-Free Package:**

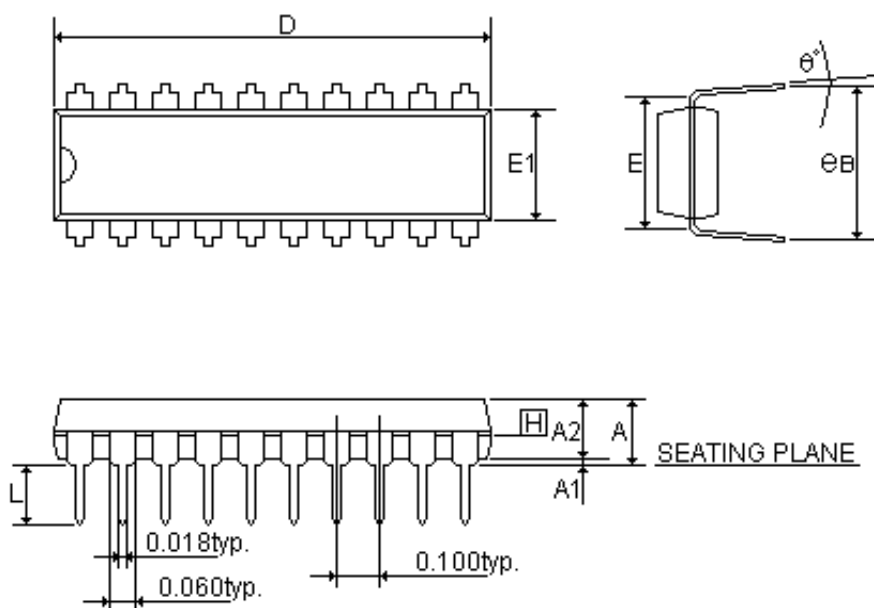
Name	ROM Type	Device	Package	Temperature	Material
SN8PC22PB	OTP	C22	P-DIP	0°C ~70°C	PB-Free Package
SN8PC22SB	OTP	C22	SOP	0°C ~70°C	PB-Free Package
SN8PC22XB	OTP	C22	SSOP	0°C ~70°C	PB-Free Package

14.4 DATECODE SYSTEM



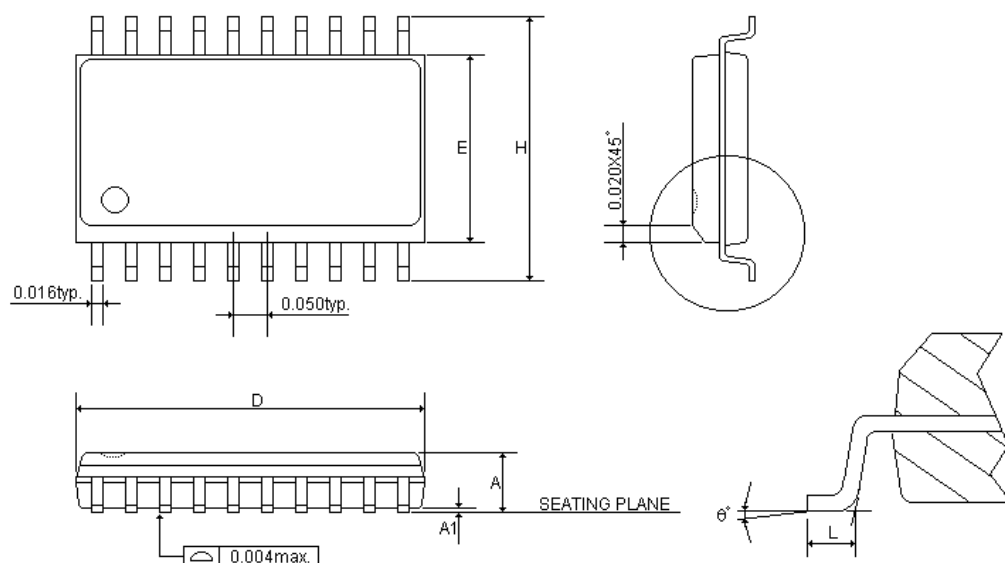
15 PACKAGE INFORMATION

15.1 P-DIP 20 PIN



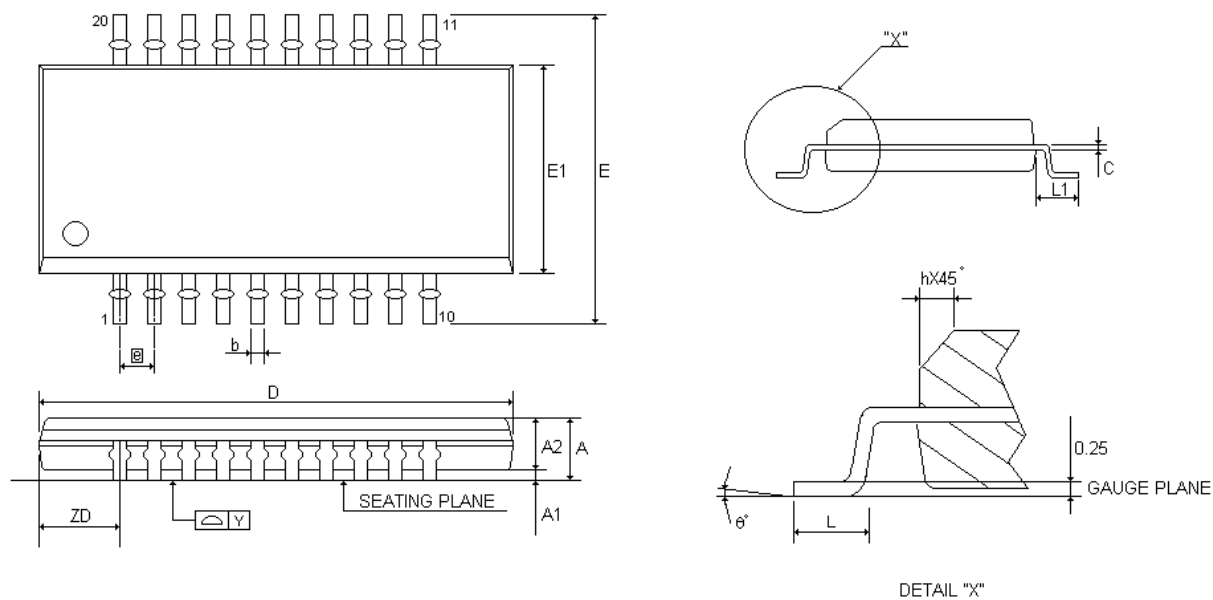
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.980	1.030	1.060	24.892	26.162	26.924
E	0.300			7.620		
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017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

15.2 SOP 20 PIN



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

15.3 SSOP 20 PIN



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

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 Street, Chupei City, Hsinchu, Taiwan.
Tel: 886-3-5600 888
Fax: 886-3-5600 889

Taipei Office:

Address: 15F-2, No.171 Song Ted Road, Taipei, Taiwan
Tel: 886-2-2759 1980
Fax: 886-2-2759 8180

Hong Kong Office:

Address: 9F-3 Energy Plaza 92 Granville Road, Tsimshatsui East, Kowloon, Hong Kong
Tel: 852-2723 8086
Fax: 852-2723 9179

Technical Support by Email:

Sn8fae@sonix.com.tw