

SN8PC21

USER'S MANUAL

Preliminary Version 0.1

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	Jun. 2009	First issue.

Table of Content

AMENDMENT HISTORY.....	2
1 PRODUCT OVERVIEW.....	6
1.1 FEATURES.....	6
1.2 SYSTEM BLOCK DIAGRAM.....	7
1.3 PIN ASSIGNMENT.....	8
1.4 PIN DESCRIPTIONS.....	8
1.5 PIN CIRCUIT DIAGRAMS.....	9
2 CENTRAL PROCESSOR UNIT (CPU).....	10
2.1 PROGRAM MEMORY (ROM).....	10
2.1.1 RESET VECTOR (0000H).....	11
2.1.2 LOOK-UP TABLE DESCRIPTION.....	12
2.1.3 JUMP TABLE DESCRIPTION.....	14
2.1.4 CHECKSUM CALCULATION.....	16
2.2 DATA MEMORY (RAM).....	17
2.2.1 SYSTEM REGISTER.....	17
2.2.1.1 SYSTEM REGISTER TABLE.....	17
2.2.1.2 SYSTEM REGISTER DESCRIPTION.....	17
2.2.1.3 BIT DEFINITION of SYSTEM REGISTER.....	18
2.2.1.4 ACCUMULATOR.....	19
2.2.2 PROGRAM FLAG.....	20
2.2.2.1 PROGRAM COUNTER.....	21
2.2.2.2 Y, Z REGISTERS.....	24
2.2.2.3 R REGISTERS.....	25
2.3 ADDRESSING MODE.....	26
2.3.1 IMMEDIATE ADDRESSING MODE.....	26
2.3.2 DIRECTLY ADDRESSING MODE.....	26
2.3.3 INDIRECTLY ADDRESSING MODE.....	26
2.4 STACK OPERATION.....	27
2.4.1 OVERVIEW.....	27
2.4.2 STACK REGISTERS.....	28
2.4.3 STACK OPERATION EXAMPLE.....	29
2.5 CODE OPTION TABLE.....	30
2.5.1 Reset_Pin code option.....	30
2.5.2 Security code option.....	30
3 RESET.....	31

3.1	OVERVIEW	31
3.2	POWER ON RESET	32
3.3	WATCHDOG RESET	32
3.4	BROWN OUT RESET	33
3.5	THE SYSTEM OPERATING VOLTAGE	34
3.6	LOW VOLTAGE DETECTOR (LVD)	34
3.7	BROWN OUT RESET IMPROVEMENT	35
3.8	EXTERNAL RESET	36
3.9	EXTERNAL RESET CIRCUIT	36
3.9.1	<i>Simply RC Reset Circuit</i>	36
3.9.2	<i>Diode & RC Reset Circuit</i>	37
3.9.3	<i>Zener Diode Reset Circuit</i>	37
3.9.4	<i>Voltage Bias Reset Circuit</i>	38
3.9.5	<i>External Reset IC</i>	39
4	SYSTEM CLOCK	40
4.1	OVERVIEW	40
4.2	CLOCK BLOCK DIAGRAM	40
4.3	OSCM REGISTER	40
4.4	SYSTEM HIGH CLOCK	41
4.4.1	<i>INTERNAL HIGH RC</i>	41
4.4.2	<i>EXTERNAL HIGH CLOCK</i>	41
4.4.2.1	CRYSTAL/CERAMIC	42
4.4.2.2	RC	42
4.4.3	<i>SYSTEM CLOCK MEASUREMENT</i>	43
4.5	SYSTEM CLOCK TIMING	44
5	SYSTEM OPERATION MODE	46
5.1	OVERVIEW	46
5.2	NORMAL MODE	47
5.3	POWER DOWN MDOE	47
5.4	OPERATING MODE CONTROL MACRO	47
5.5	WAKEUP	48
5.5.1	<i>OVERVIEW</i>	48
5.5.2	<i>WAKEUP TIME</i>	48
6	I/O PORT	49
6.1	OVERVIEW	49
6.2	I/O PORT MODE	50
6.3	I/O PORT DATA REGISTER	51

7	TIMERS	52
7.1	WATCHDOG TIMER.....	52
7.2	TIMER 0 (T0).....	54
7.2.1	OVERVIEW.....	54
7.2.2	T0M MODE REGISTER.....	54
7.2.3	T0C COUNTING REGISTER.....	55
7.2.4	T0 TIMER OPERATION SEQUENCE.....	56
8	IR OUTPUT	57
8.1	OVERVIEW	57
8.2	IRM MODE REGISTER.....	58
8.3	IRC COUNTING REGISTER.....	58
8.4	IRR AUTO-LOAD REGISTER.....	59
8.5	IRD IR DUTY CONTROL REGISTER.....	60
8.6	IR OUTPUT OPERATION SEQUENCE.....	61
9	INSTRUCTION TABLE	62
10	ELECTRICAL CHARACTERISTIC	63
10.1	ABSOLUTE MAXIMUM RATING	63
10.2	ELECTRICAL CHARACTERISTIC.....	63
10.3	CHARACTERISTIC GRAPHS.....	64
11	DEVELOPMENT TOOL	65
11.1	ICE AND EV-KIT APPLICATION NOTIC	65
12	OTP PROGRAMMING PIN	66
12.1	THE PIN ASSIGNMENT OF EASY WRITER TRANSITION BOARD SOCKET:	66
12.2	PROGRAMMING PIN MAPPING:	67
13	PACKAGE INFORMATION	68
13.1	DIP16 PIN.....	68
13.2	SOP 16 PIN	69
14	MARKING DEFINITION	70
14.1	INTRODUCTION.....	70
14.2	MARKING INDETIFICATION SYSTEM	70
14.3	MARKING EXAMPLE.....	71
14.4	DATECODE SYSTEM	71

1 PRODUCT OVERVIEW

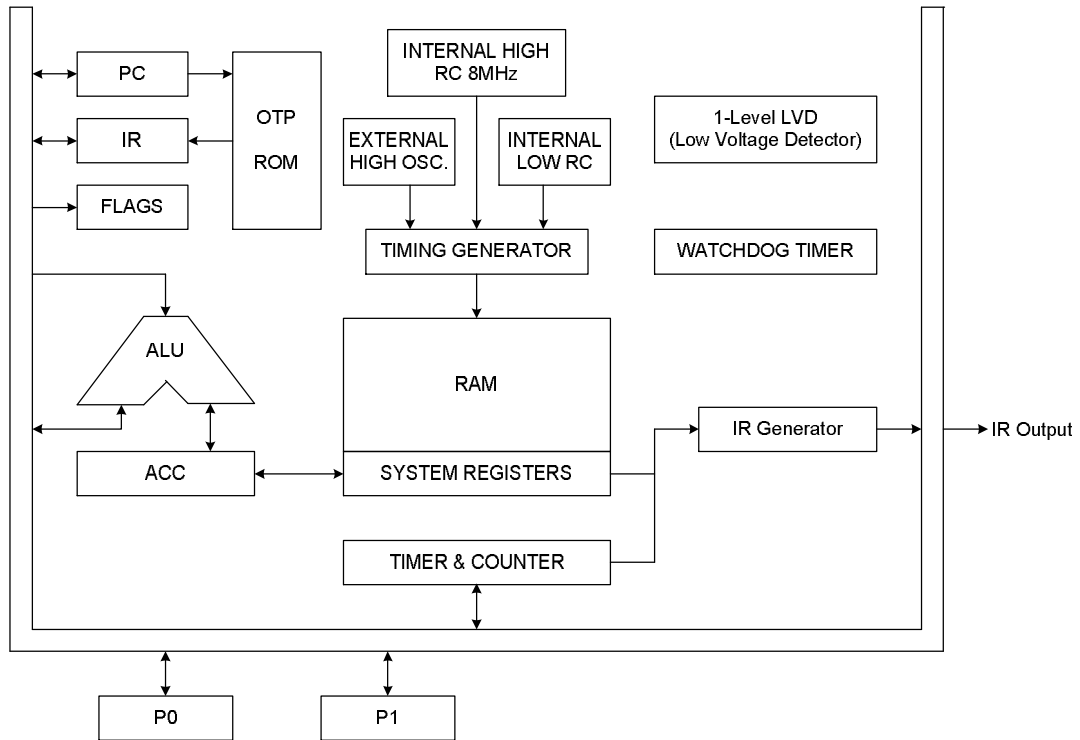
1.1 FEATURES

- ◆ **Memory configuration**
OTP ROM size: 1K * 16 bits.
RAM size: 32 * 8 bits.
- ◆ **4 levels stack buffer.**
- ◆ **I/O pin configuration**
Bi-directional: P0, P1.
400mA IR output pin: IROUT
Wakeup: P0, P1 level change trigger.
Pull-up resistors: P0, P1.
- ◆ **1-Level LVD.**
- ◆ **Fcpu (Instruction cycle)**
Fcpu = Fosc/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).
- ◆ **No interrupt function.**
- ◆ **One 8-bit timer. (T0).**
T0: Basic timer with flag function.
- ◆ **One channel duty/cycle programmable IR output.**
- ◆ **Five 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 16KHz(3V)
- ◆ **Two operating modes**
Normal mode: Both high and low clock active
Sleep mode: Both high and low clock stop
- ◆ **Package (Chip form support)**
PDIP 16 pin
SOP 16 pin

F Features Selection Table

CHIP	ROM	RAM	Stack	T0	Oscillator			I/O	IR Output	Wakeup Pin No.	Package
					Ext. 455K	Ext. 4M	Int. 8M				
SN8PC01	0.5K*16	32	4	-	V	-	-	16	Fix 38KHz, Normal current	9	PDIP20/SOP20
SN8PC13	2K*16	48	4	V	-	V	-	16	Duty/cycle programmable, Normal current	15	PDIP20/SOP20/ SSOP20
SN8PC20	2K*16	56	4	V	V	V	V	18	Duty/cycle programmable, 400mA sink current	16	PDIP20/SOP20/ SSOP20
SN8PC21	1K*16	32	4	V	-	V	V	13	Duty/cycle programmable, 400mA sink current	13	PDIP16/SOP16

1.2 SYSTEM BLOCK DIAGRAM



1.3 PIN ASSIGNMENT

SN8PC21P (PDIP 16 pins)
SN8PC21S (SOP 16 pins)

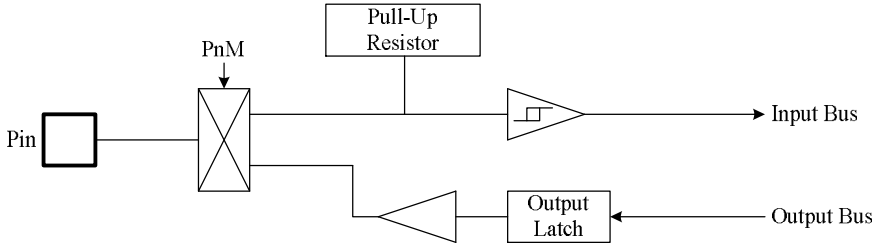
P0.0	1	U	16	VDD
P0.1	2		15	VSS
P0.2/RST/VPP	3		14	IROUT
P0.3/XIN	4		13	P1.5
P0.4/XOUT	5		12	P1.4
P0.5	6		11	P1.3
P0.6	7		10	P1.2
P1.0	8		9	P1.1

1.4 PIN DESCRIPTIONS

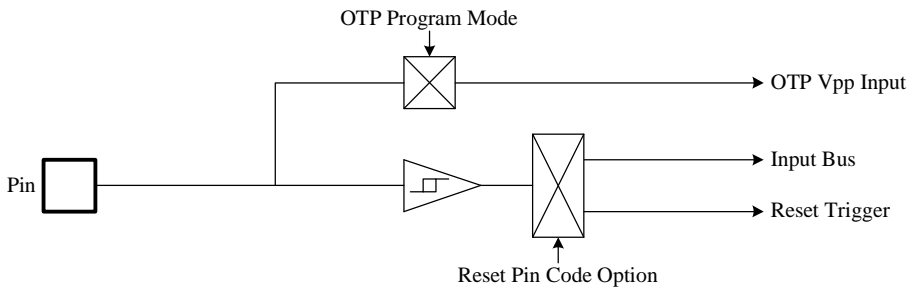
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital and analog circuit.
P0.2/RST/ VPP	I, P	RST: System external reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”.
		VPP: OTP power input pin in programming mode.
		P0.2: Input only pin with Schmitt trigger 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[6:0]	I/O	P0: Bi-direction pin. Schmitt trigger structure. Built-in pull-up resistors and level change wake-up function as input mode.
P1[5:0]	I/O	P1: Bi-direction pin. Schmitt trigger structure. Built-in pull-up resistors and level change wake-up function as input mode.
IROUT	O	IROUT: Duty/cycle programmable IR signal output pin.

1.5 PIN CIRCUIT DIAGRAMS

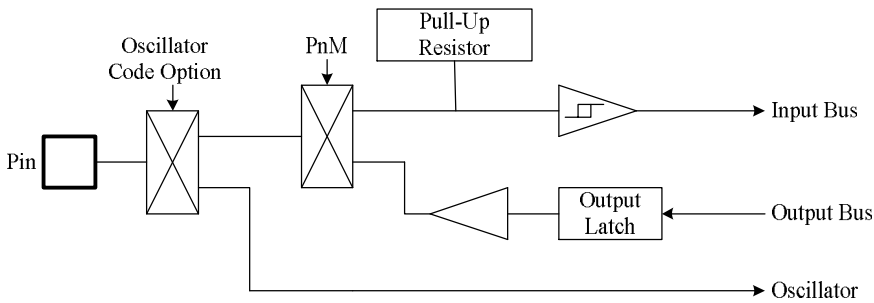
I I/O with Schmitt-trigger ($V_{IH}=0.7*V_{dd}$, $V_{IL}=0.3*V_{dd}$) and Pull-up Resistor(200K Ω @3V):



I Input Only with Schmitt-trigger ($V_{IH}=0.7*V_{dd}$, $V_{IL}=0.3*V_{dd}$) and Vpp High Voltage:



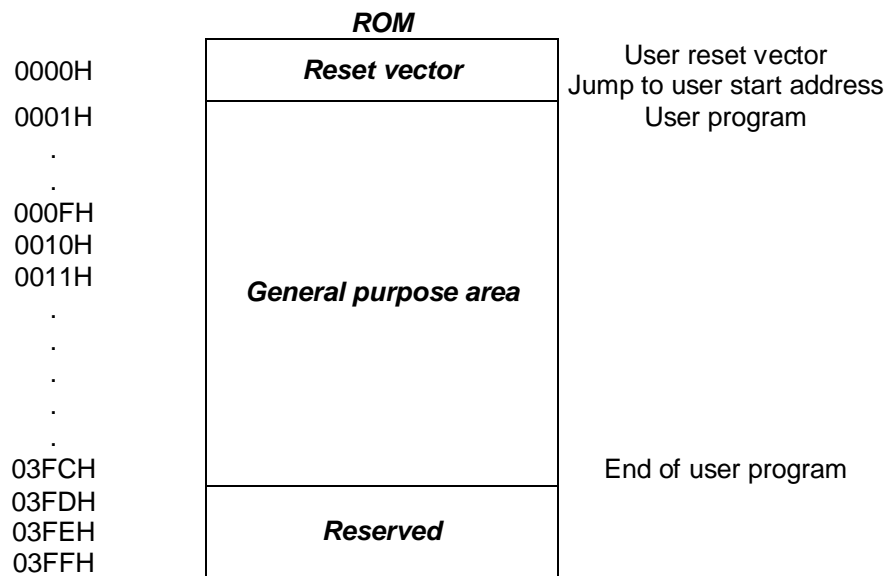
I Oscillator shared I/O with Schmitt-trigger ($V_{IH}=0.7*V_{dd}$, $V_{IL}=0.3*V_{dd}$) and Pull-up Resistor(200K Ω @3V):



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (ROM)

F 1K words ROM



The ROM includes Reset vector, General purpose area and Reserved area. The Reset vector is program beginning address. The General purpose area is main program area including main loop, sub-routines and data table.

2.1.1 RESET VECTOR (0000H)

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

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

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

Ø Example: Defining Reset Vector

```

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

```

2.1.2 LOOK-UP TABLE DESCRIPTION

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

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

```

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

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

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

```

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

Ø Example: INC_YZ macro.

```

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

        INCMS    Y                ; Y+1
        NOP     ; Not overflow

@@:
        ENDM

```

Ø Example: Modify above example by "INC_YZ" macro.

```

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

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

```

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

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

```

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

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

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

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

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

```

2.1.3 JUMP TABLE DESCRIPTION

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

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

Ø Example: Jump table.

```

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

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

```

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

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

```

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

```

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

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

```

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

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

Ø **Example: “@JMP_A” operation.**

; Before compiling program.

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

; After compiling program.

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

2.1.4 CHECKSUM CALCULATION

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

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

```

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

@@:
MOV     MOVC
B0BSET  FC                ; Clear C flag
ADD     DATA1, A         ; Add A to Data1
MOV     A, R
ADC     DATA2, A         ; Add R to Data2
JMP     END_CHECK        ; Check if the YZ address = the end of code

AAA:
INCMS   Z                 ; Z=Z+1
JMP     @B                ; If Z != 00H calculate to next address
JMP     Y_ADD_1          ; If Z = 00H increase Y

END_CHECK:
MOV     A, END_ADDR1
CMPRS   A, Z              ; Check if Z = low end address
JMP     AAA              ; If Not jump to checksum calculate
MOV     A, END_ADDR2
CMPRS   A, Y              ; If Yes, check if Y = middle end address
JMP     AAA              ; If Not jump to checksum calculate
JMP     CHECKSUM_END     ; If Yes checksum calculated is done.

Y_ADD_1:
INCMS   Y                 ; Increase Y
NOP
JMP     @B                ; Jump to checksum calculate

CHECKSUM_END:
...
...
END_USER_CODE:          ; Label of program end

```


2.2 DATA MEMORY (RAM)

F 32 X 8-bit RAM

BANK 0	Address	RAM Location	
	000h	General Purpose Area	RAM Bank 0
	“		
	“		
	“		
	01Fh	System Register	080h~0FFh of Bank 0 store system registers (128 bytes).
	080h		
	“		
	“		
	“		
0FFh	End of Bank 0		

Sonix provides “Bank 0” type instructions (e.g. b0mov, b0add, b0bts1, b0bset...) to control Bank 0 RAM directly.

2.2.1 SYSTEM REGISTER

2.2.1.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	-	-	R	Z	Y	-	PFLAG	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	-
C	-	P1M	-	-	-	-	-	-	INTRQ	-	OSCM	-	WDTR	IRR	PCL	PCH
D	P0	P1	-	-	-	-	-	-	T0M	T0C	IRM	IRC	-	-	-	STKP
E	-	-	-	-	-	-	-	@YZ	IRD	-	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 SYSTEM REGISTER DESCRIPTION

R = Working register and ROM look-up data buffer.
PnM = Port n input/output mode register.
INTRQ = Interrupt request register.
OSCM = Oscillator mode register.
IRR = IR counter auto-reload data buffer.
Pn = Port n data buffer.
T0C = T0 counting register.
IRC = IR counting register.
@YZ = RAM YZ indirect addressing index pointer.

Y, Z = Working, @YZ and ROM addressing register.
PFLAG = ROM page and special flag register.
WDTR = Watchdog timer clear register.
PCH, PCL = Program counter.
T0M = T0 mode register.
IRM = IR mode register.
STKP = Stack pointer buffer.
IRD = IR duty control register.
STK0~STK3 = Stack 0 ~ stack 3 buffer.

2.2.1.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD				C	DC	Z	R/W	PFLAG
0B8H		P06M	P05M	P04M	P03M		P01M	P00M	R/W	P0M
0C1H			P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C8H				T0IRQ					R/W	INTRQ
0CAH					CPUM0				R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0	W	IRR
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH							PC9	PC8	R/W	PCH
0D0H		P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H			P15	P14	P13	P12	P11	P10	R/W	P1
0D8H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH								CREN	R/W	IRM
0DBH	IRC7	IRC6	IRC5	IRC4	IRC3	IRC2	IRC1	IRC0	R/W	IRC
0DFH	GIE						STKPB1	STKPB0	R/W	STKP
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E8H	IRD7	IRD6	IRD5	IRD4	IRD3	IRD2	IRD1	IRD0	W	IRD
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H							S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH							S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH							S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH							S0PC9	S0PC8	R/W	STK0H

Note:

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

2.2.1.4 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) occurrence, then these flags will be set to PFLAG register.

Ø Example: Read and write ACC value.

; Read ACC data and store in BUF data memory

```
MOV     BUF, A
```

; Write a immediate data into ACC

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

; Write ACC data from BUF data memory

```
MOV     A, BUF
```

2.2.2 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.**

2.2.2.1 PROGRAM COUNTER

The program counter (PC) is a 10-bit binary counter separated into the high-byte 2 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

Besides, it can be replaced with specific address by executing CALL or JMP instruction. When JMP or CALL instruction is executed, the destination address will be inserted to bit 0 ~ bit 9.

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

F ONE ADDRESS SKIPPING

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

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

```

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

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

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

```

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

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

INCS instruction:

```

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

```

C0STEP: NOP

INCMS instruction:

```

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

```

C0STEP: NOP

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

DECS instruction:

```

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

```

C0STEP: NOP

DECMS instruction:

```

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

```

C0STEP: NOP

F MULTI-ADDRESS JUMPING

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

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

∅ **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
      ...
      ...
```

2.2.2.2 Y, Z REGISTERS

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

- I can be used as general working registers
- I can be used as RAM data pointers with @YZ register
- I 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:
...

```


2.2.2.3 R REGISTERS

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

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

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

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

2.3 ADDRESSING MODE

2.3.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.3.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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

2.3.3 INDIRECTLY ADDRESSING MODE

The indirectly addressing mode is to access the memory by the data pointer registers (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.
```

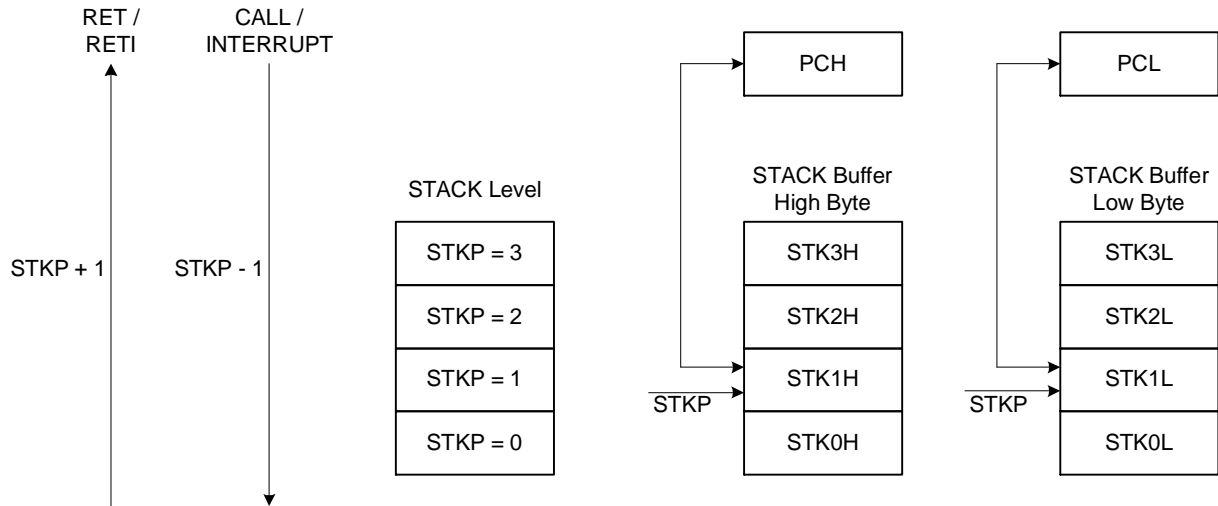
Example: Indirectly addressing mode with @YZ register

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



2.4.2 STACK REGISTERS

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

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

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

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

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

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

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

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

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

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

2.4.3 STACK OPERATION EXAMPLE

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

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

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

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

2.5 CODE OPTION TABLE

The code option is the system hardware configurations including 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.
	RC	Low cost RC for external high clock oscillator. XIN pin is connected to RC oscillator. XOUT pin is bi-direction GPIO mode.
	4M X'tal	Standard crystal /resonator (e.g. 4M~8M) for external high clock oscillator.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
Reset_Pin	Reset	Enable External reset pin.
	P02	Enable P0.2 input only without pull-up resistor.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.

2.5.1 Reset_Pin code option

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

- I **Reset:** The reset pin is external reset function. When falling edge trigger occurring, the system will be reset.
- I **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.

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

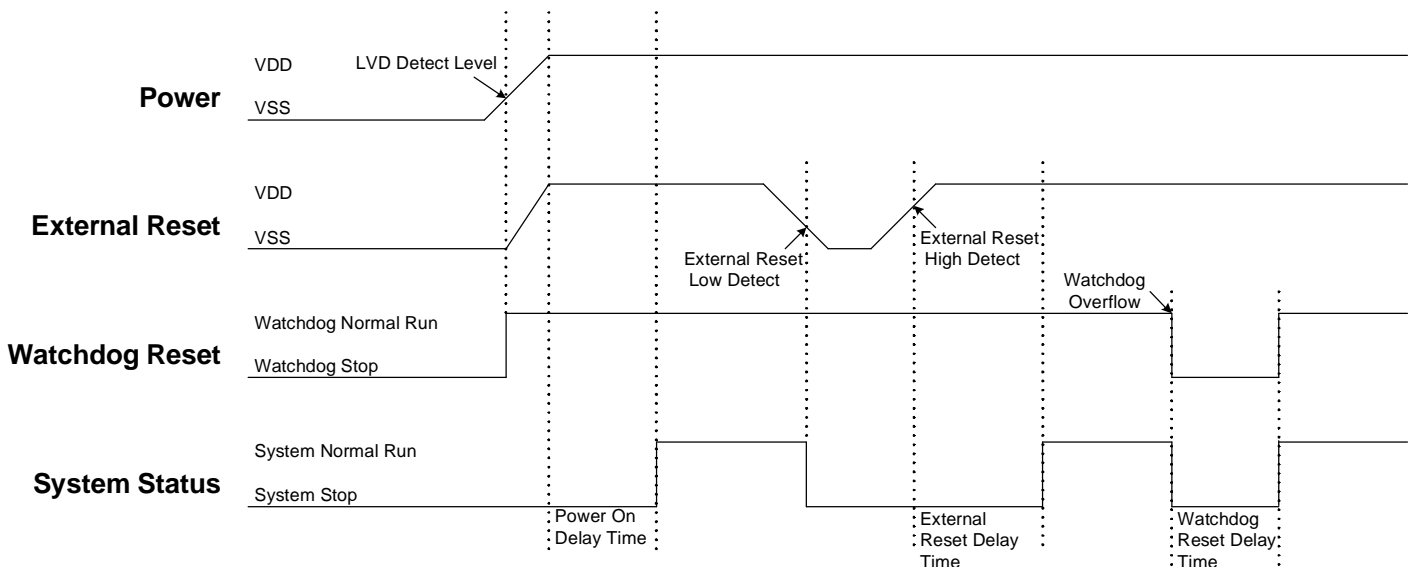
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
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.

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

3.3 WATCHDOG RESET

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

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

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

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

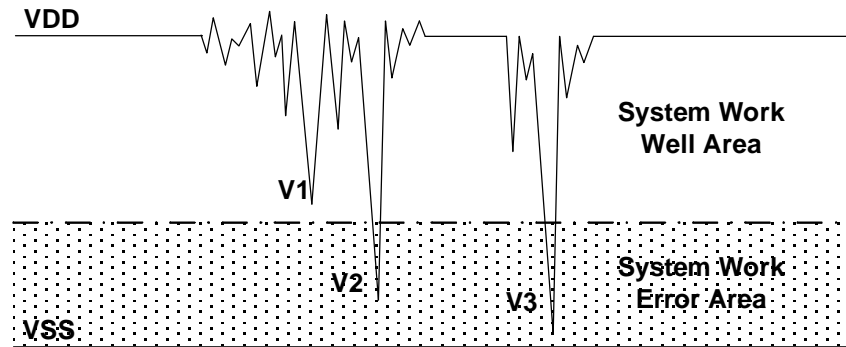
Watchdog timer application note is as following.

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

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

3.4 BROWN OUT RESET

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



Brown Out Reset Diagram

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

DC application:

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

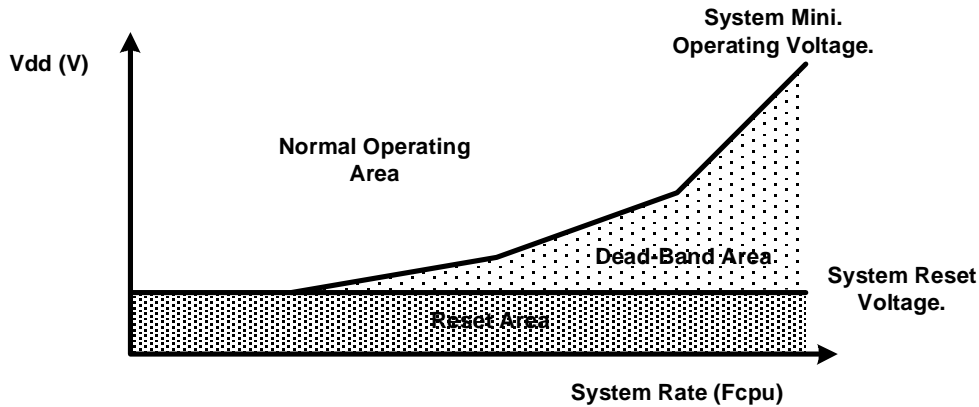
AC application:

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

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

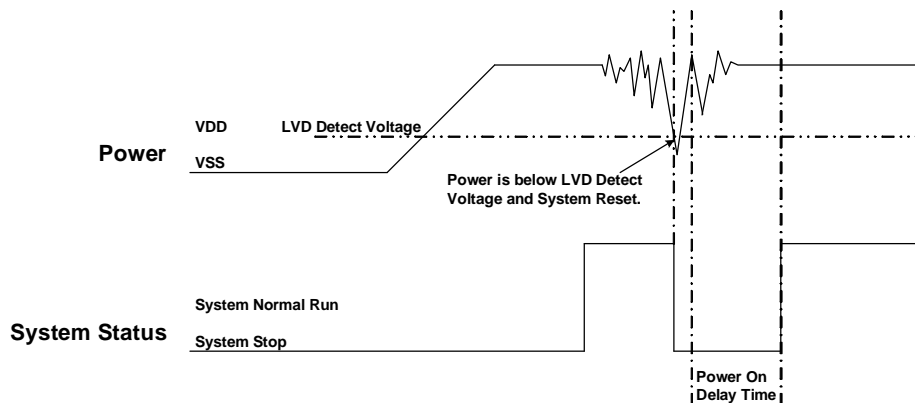
3.5 THE SYSTEM OPERATING VOLTAGE

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



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

3.6 LOW VOLTAGE DETECTOR (LVD)



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

3.7 BROWN OUT RESET IMPROVEMENT

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

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

- **Note:**

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

Watchdog reset:

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

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

Reduce the system executing rate:

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

External reset circuit:

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

3.8 EXTERNAL RESET

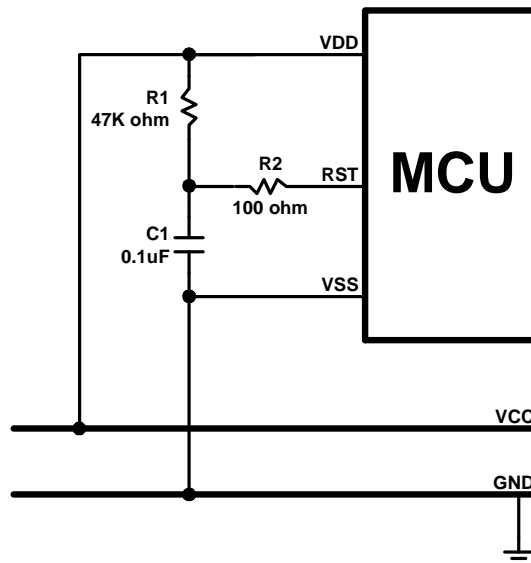
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

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

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

3.9 EXTERNAL RESET CIRCUIT

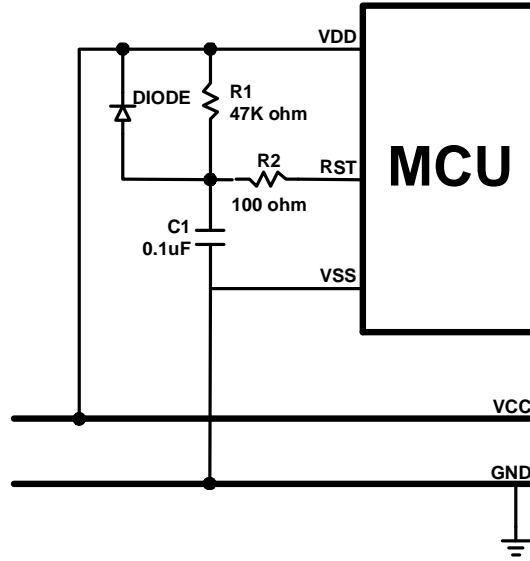
3.9.1 Simply RC Reset Circuit



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

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

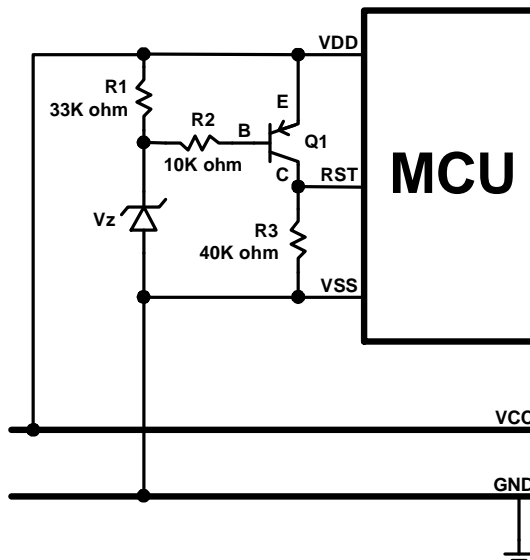
3.9.2 Diode & RC Reset Circuit



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

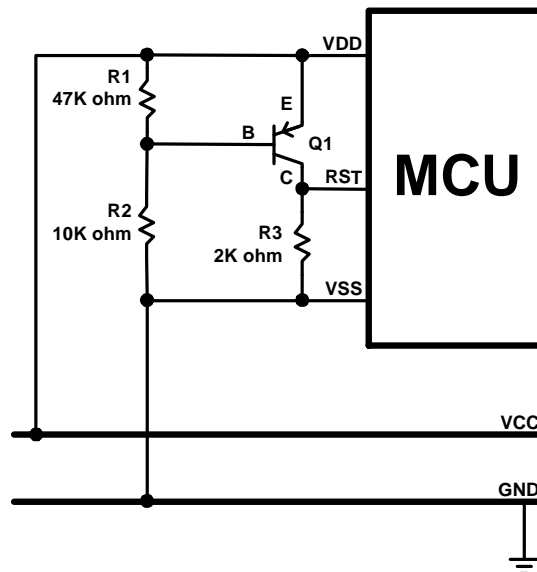
- **Note:** The R2 100 ohm resistor of “Simply reset circuit” and “Diode & RC reset circuit” is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).

3.9.3 Zener Diode Reset Circuit



The zener diode reset circuit is a simple low voltage detector and can **improve brown out reset condition completely**. Use zener voltage to be the active level. When VDD voltage level is above “ $V_z + 0.7V$ ”, 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.

3.9.4 Voltage Bias Reset Circuit

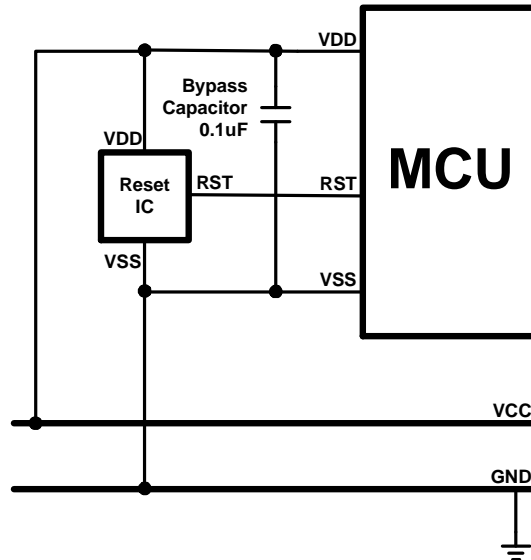


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to $0.7V \times (R1 + R2)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

3.9.5 External Reset IC



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

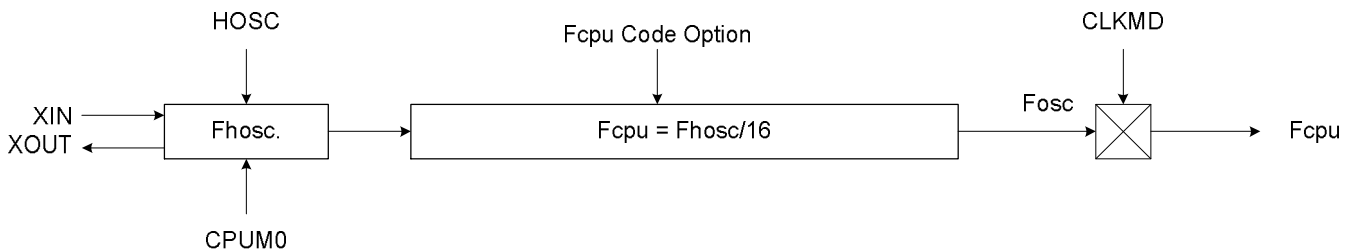
4 SYSTEM CLOCK

4.1 OVERVIEW

The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit or on-chip 8MHz high-speed RC oscillator circuit (IHRC 8MHz). The low-speed clock is generated from on-chip low-speed RC oscillator circuit (ILRC 16KHz @3V). The high-speed clock can be system clock (Fosc). The low-speed clock only supports watchdog timer.

F Normal Mode (High Clock): $F_{cpu} = F_{osc} / 16$

4.2 CLOCK BLOCK DIAGRAM



- I HOSC: High_Clk code option.
- I Fosc: External high-speed clock / Internal high-speed RC clock.
- I Fosc: System clock source.
- I Fcpu: Instruction cycle.

4.3 OSCM REGISTER

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

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

Bit[4:3] **CPUM0**: CPU operating mode control bits.
 0 = normal mode.
 1 = sleep (power down) mode.

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

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

4.4 SYSTEM HIGH CLOCK

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

High_Clk Code Option	Description
IHRC_8M	The high clock is internal 8MHz oscillator RC type. XIN and XOUT pins are general purpose I/O pins.
RC	The high clock is external RC type oscillator. XOUT pin is general purpose I/O pin.
4M	The high clock is external oscillator. The typical frequency is 4MHz.

4.4.1 INTERNAL HIGH RC

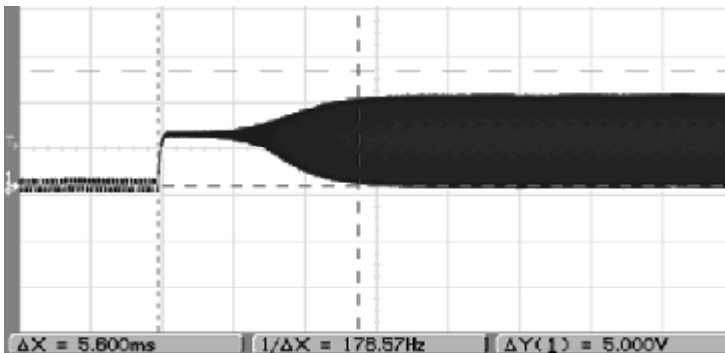
The chip is built-in RC type internal high clock (8MHz) controlled by "IHRC_8M" code options. In "IHRC_8M" mode, the system clock is from internal 8MHz RC type oscillator and XIN/XOUT pins are general-purpose I/O pins. |

I **IHRC:** High clock is internal 8MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.

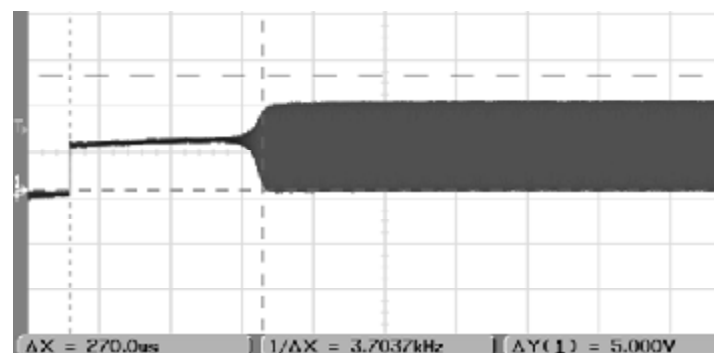
4.4.2 EXTERNAL HIGH CLOCK

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

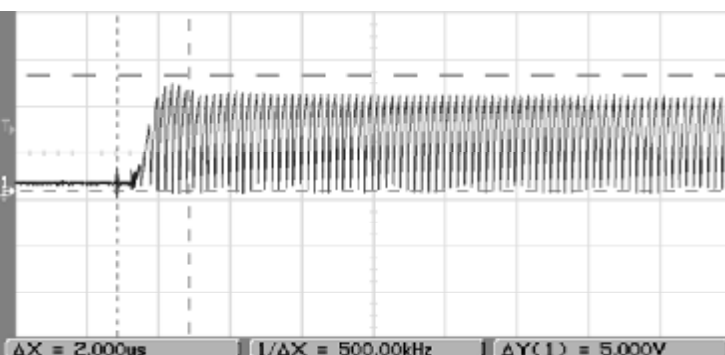
4MHz Crystal



4MHz Ceramic

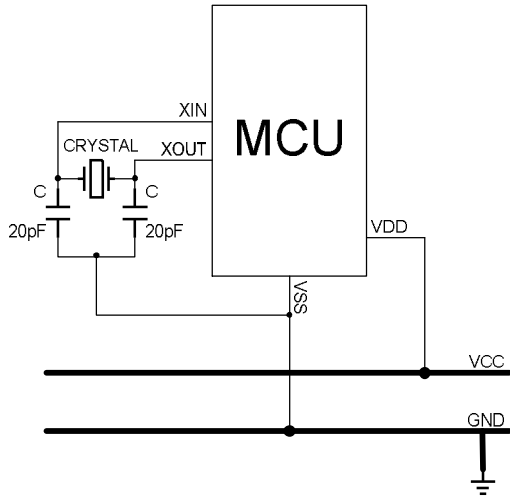


RC



4.4.2.1 CRYSTAL/CERAMIC

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

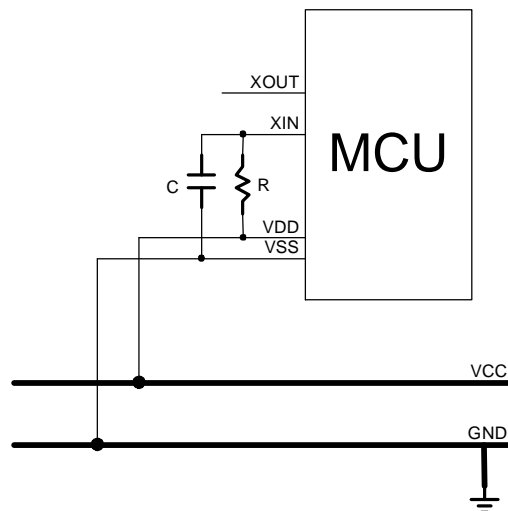


I 4M/8M code option crystal's capacitors are 20pF.

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

4.4.2.2 RC

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



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

4.4.3 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.
```

@ @:

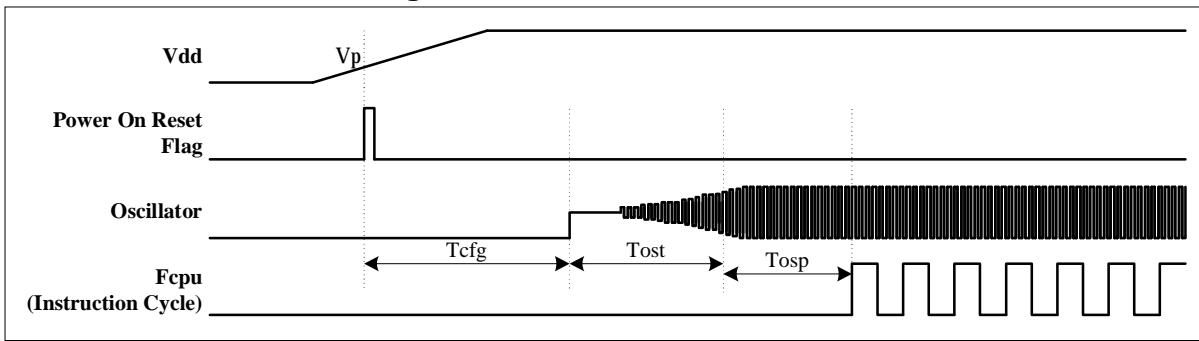
```
B0BSET    P0.0        ; Output Fcpu toggle signal in low-speed clock mode.  
B0BCLR    P0.0        ; Measure the Fcpu frequency by oscilloscope.  
JMP       @B
```

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

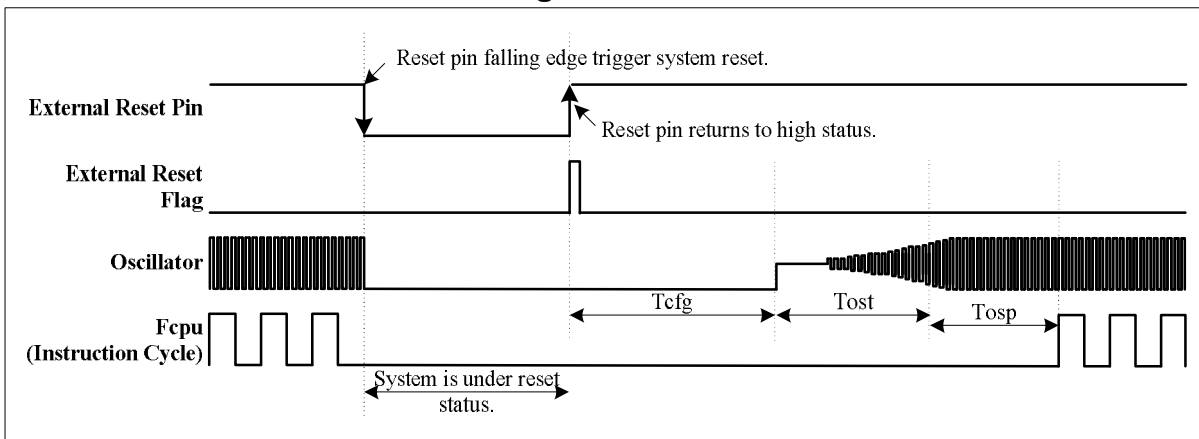
4.5 SYSTEM CLOCK TIMING

Parameter	Symbol	Description	Typical
Hardware configuration time	Tcfg	$2048 * F_{ILRC}$	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. $2048 * F_{hosc}$ (Power on reset, LVD reset, watchdog reset, external reset pin active.)	512us @ $F_{hosc} = 4\text{MHz}$ 256us @ $F_{hosc} = 8\text{MHz}$
		Oscillator warm-up time of power down mode wake-up condition. $2048 * F_{hosc}$Crystal/resonator type oscillator, e.g. 4MHz crystal, 16MHz crystal... $32 * F_{hosc}$RC type oscillator, e.g. external RC type oscillator, internal high-speed RC type oscillator.	512us @ $F_{hosc} = 4\text{MHz}$ 256us @ $F_{hosc} = 8\text{MHz}$

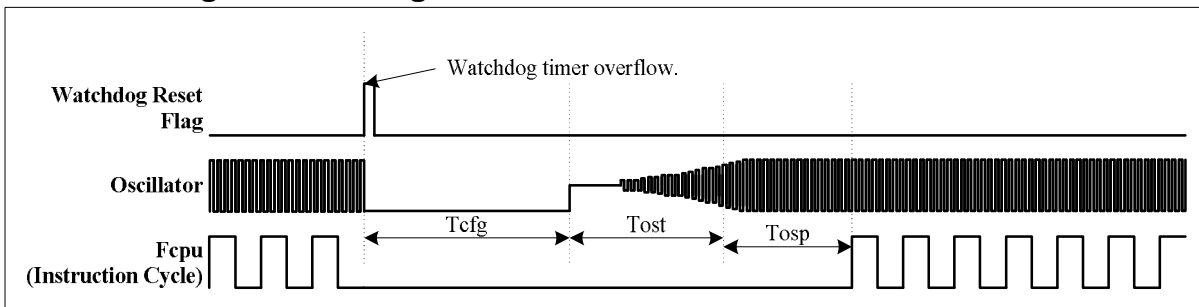
I Power On Reset Timing



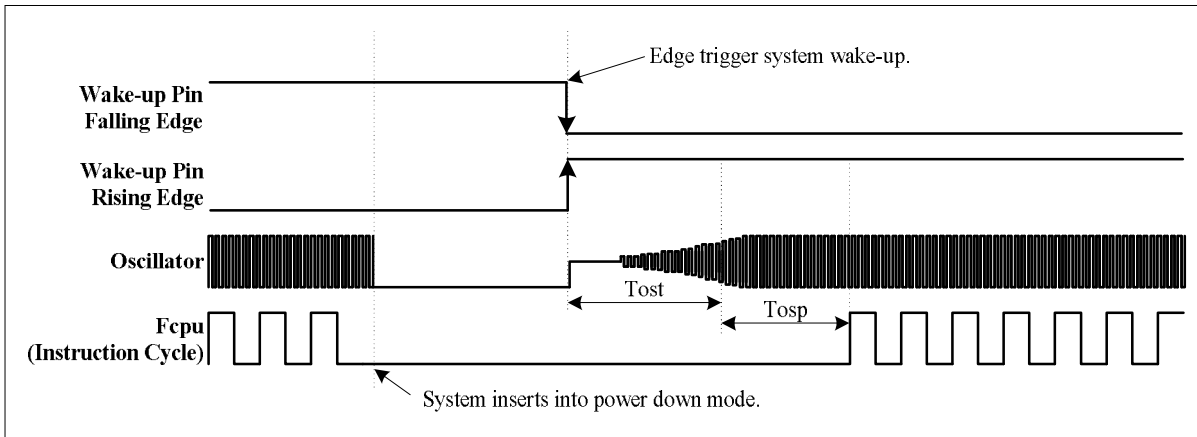
I External Reset Pin Reset Timing



I Watchdog Reset Timing

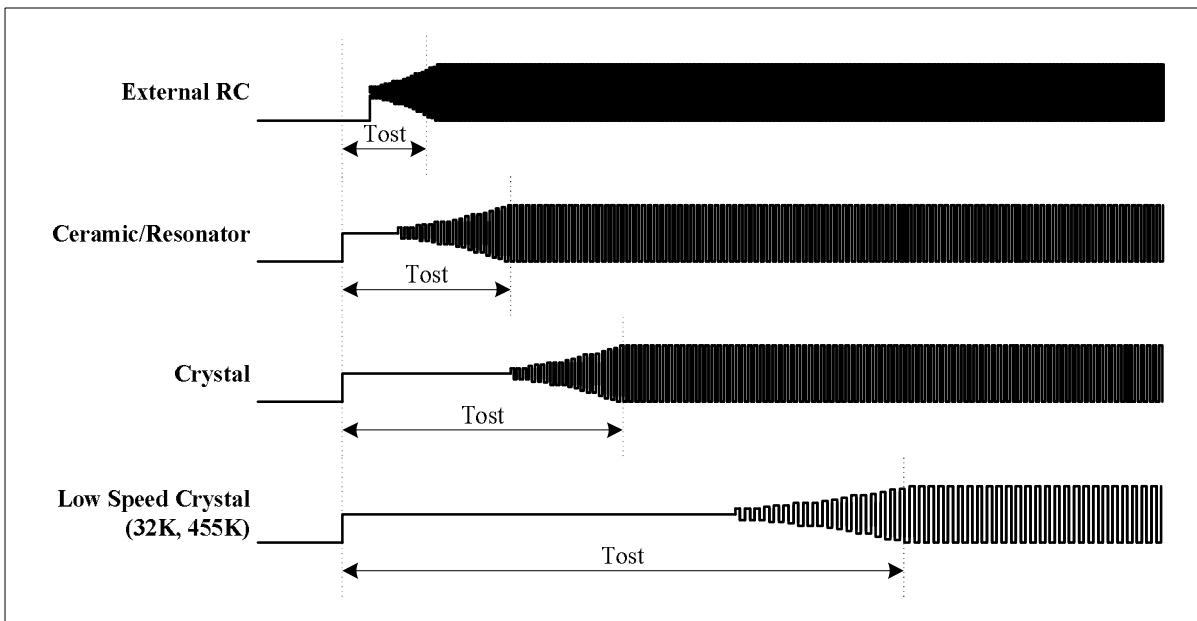


I Power Down Mode Wake-up Timing



I 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.

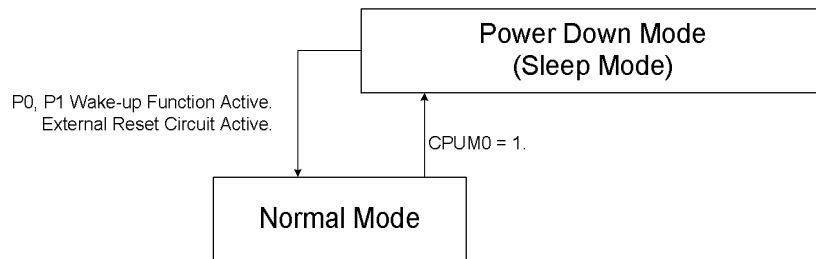


5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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



System Mode Switching Diagram

Operating mode description

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

- I **EHOSC**: External high clock
- I **IHRC**: Internal high clock (8M RC oscillator)
- I **ILRC**: Internal low clock (16KHz RC oscillator at 3V)

5.2 NORMAL MODE

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

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

5.3 POWER DOWN MDOE

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

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

5.4 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).

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

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

5.5 WAKEUP

5.5.1 OVERVIEW

Under power down mode (sleep mode), program doesn't execute. The wakeup trigger can wake the system up to normal mode. The wakeup trigger sources is external trigger (P0 level change).

I Power down mode is waked up to normal mode. The wakeup trigger is only external trigger (P0 level change)

5.5.2 WAKEUP TIME

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

The value of the wakeup time is as the following.

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

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

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

$$\begin{aligned} \text{The wakeup time} &= 1/F_{osc} * 2048 = 256\mu\text{s} \quad (F_{osc} = 4\text{MHz}) \\ \text{The total wakeup time} &= 256\mu\text{s} + \text{oscillator start-up time} \end{aligned}$$

6 I/O PORT

6.1 OVERVIEW

The micro-controller builds in 13 pin I/O including P0 and P1. These ports are GPIO purpose and build in internal pull-up resistor when I/O pin be set as input mode.

- **Note: GPIO's pull-up resistors are enabled when GPIO be set as input mode.**

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	
P0.2	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P0.4	I/O	XOUT	AC	High_CLK code option = 455K, 4M, 8M
P0.3	I/O	XIN	AC	High_CLK code option = RC, 455K, 4M, 8M

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

6.2 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	-	P06M	P05M	P04M	P03M	-	P01M	P00M
Read/Write	-	R/W	R/W	R/W	R/W	-	R/W	R/W
After reset	-	0	0	0	0	-	0	0

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

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

Ø Example: I/O mode selecting

```

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

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

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

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

6.3 I/O PORT DATA REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	-	P06	P05	P04	P03	P02	P01	P00
Read/Write	-	R/W	R/W	R/W	R/W	R	R/W	R/W
After reset	-	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	-	-	P15	P14	P13	P12	P11	P10
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

- **Note: The P02 keeps "1" when external reset enable by code option.**

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

```
B0MOV    A, P0           ; Read data from Port 0
B0MOV    A, P1           ; Read data from Port 4
```

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

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

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

```
B0BSET   P1.0           ; Set P1.0 to be "1".
```

```
B0BCLR   P1.0           ; Set P1.0 to be "0".
```

7 TIMERS

7.1 WATCHDOG TIMER

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

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

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

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

- I **Disable:** Disable watchdog timer function.
- I **Enable:** Enable watchdog timer function. Watchdog timer actives in normal mode and slow mode. In power down mode and green mode, the watchdog timer stops.
- I **Always_On:** Enable watchdog timer function. The watchdog timer 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

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

Main:

```

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

```

Ø Example: Clear watchdog timer by @RST_WDT macro.

Main:

```

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

Watchdog timer application note is as following.

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

Main:

```

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

Correct:

```

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

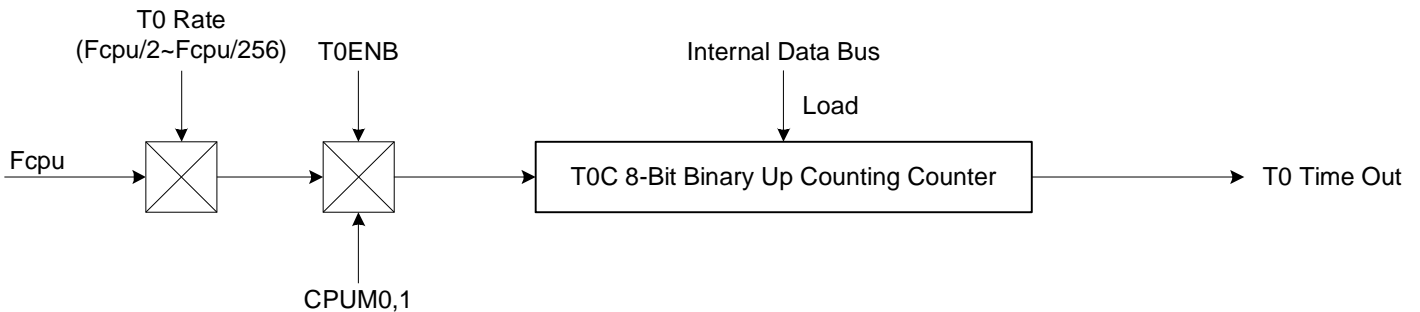
7.2 TIMER 0 (T0)

7.2.1 OVERVIEW

The T0 is an 8-bit binary up timer without interrupt function. If T0 timer occurs an overflow (from FFH to 00H), it will continue counting and issue a time-out signal to trigger T0 interrupt request flag and set T0IRQ (T0IRQ=1). Tracking T0IRQ status is to get right timer period by polling program.

The main purposes of the T0 timer is as following.

- F 8-bit programmable up counting timer:** Generates timer overflow request at specific time intervals based on the selected clock frequency.



7.2.2 T0M MODE REGISTER

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

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

000 = fcpu/256.

001 = fcpu/128.

...

110 = fcpu/4.

111 = fcpu/2.

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

0 = Disable T0 timer.

1 = Enable T0 timer.

7.2.3 T0C COUNTING REGISTER

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

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

The equation of T0C initial value is as following.

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

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

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

The basic timer table interval time of T0.

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

7.2.4 T0 TIMER OPERATION SEQUENCE

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

F Stop T0 timer counting, clear T0 interrupt request flag.

```
BCLR    FT0ENB    ; T0 timer.
BCLR    FT0IRQ    ; T0 time out request flag is cleared.
```

F Set T0 timer rate.

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

F Set T0 interrupt interval time.

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

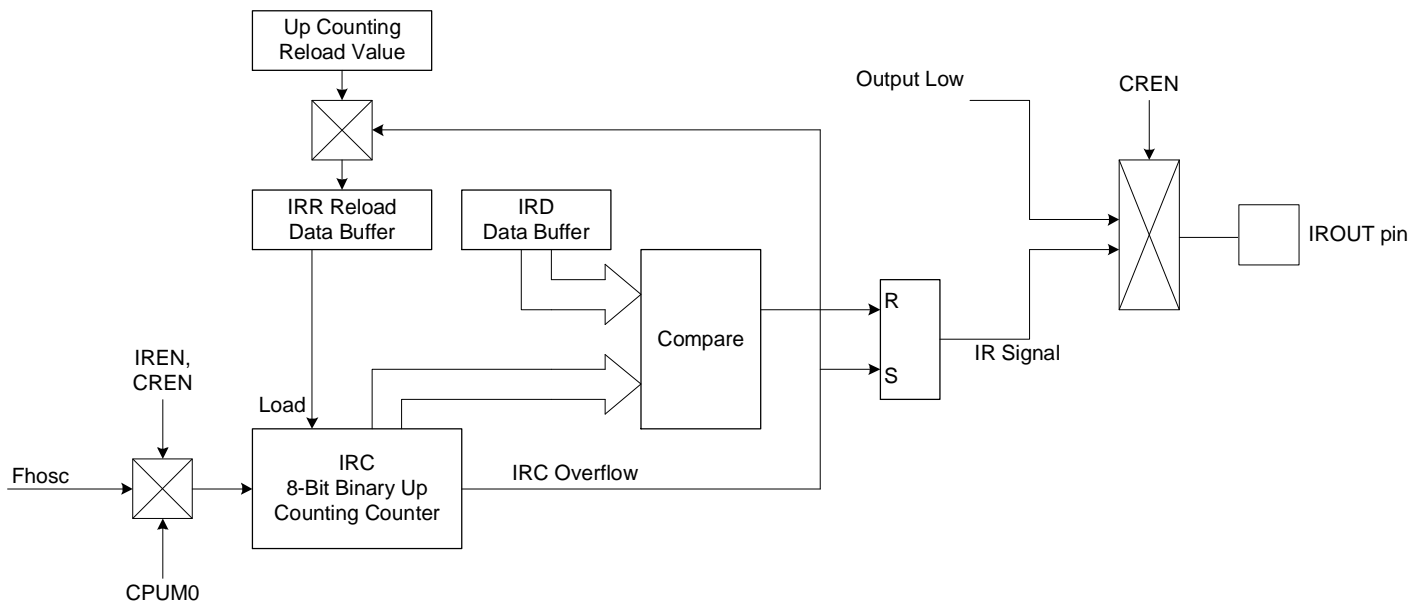
F Enable T0 timer.

```
BSET    FT0ENB    ; Enable T0 timer.
```


8 IR OUTPUT

8.1 OVERVIEW

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



- **Note:** Before system inserting power down mode, disable IR carry output (CREN=0) is necessary to avoid IR carry signal error. The system forces IROUT exchanging to low status in power down mode automatically

8.2 IRM MODE REGISTER

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

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

8.3 IRC COUNTING REGISTER

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

ODBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRC	IRC7	IRC6	IRC5	IRC4	IRC3	IRC2	IRC1	IRC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

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

8.4 IRR AUTO-LOAD REGISTER

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

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

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

The equation of IRRinitial value is as following.

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

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

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

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

IR interval time = 1/38KHz = 26.3us

Input clock = external oscillator 4MHz.

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

8.5 IRD IR DUTY CONTROL REGISTER

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

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

The equation of IRD initial value is as following.

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

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

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

IRR of 38KHz = 151

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

Common IR signal table. System clock is 4MHz.

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

8.6 IR OUTPUT OPERATION SEQUENCE

F Set IRC and IRR for IR cycle.

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

F Set IRD for IR duty.

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

F Enable IR output.

```
BSET     FCREN             ; Set IR carry signal output.
```

9 INSTRUCTION TABLE

Field	Mnemonic	Description	C	DC	Z	Cycle
MOV	MOV A,M	$A \leftarrow M$	-	-	√	1
	MOV M,A	$M \leftarrow A$	-	-	-	1
	B0MOV A,M	$A \leftarrow M$ (bank 0)	-	-	√	1
	B0MOV M,A	M (bank 0) $\leftarrow A$	-	-	-	1
	MOV A,I	$A \leftarrow I$	-	-	-	1
	B0MOV M,I	$M \leftarrow I$, "M" only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z...)	-	-	-	1
	XCH A,M	$A \leftrightarrow M$	-	-	-	1+N
	B0XCH A,M	$A \leftrightarrow M$ (bank 0)	-	-	-	1+N
MOV	MOV R,A	$R, A \leftarrow ROM [Y,Z]$	-	-	-	2
ADD	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	
AND	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
SWAP	SWAP M	$A (b3\sim b0, b7\sim b4) \leftarrow M(b7\sim b4, b3\sim b0)$	-	-	-	1
	SWAPM M	$M(b3\sim b0, b7\sim b4) \leftarrow M(b7\sim b4, b3\sim b0)$	-	-	-	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	
CMPRS	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	$PC15/14 \leftarrow RomPages1/0, PC13\sim PC0 \leftarrow d$	-	-	-	2
	CALL d	$Stack \leftarrow PC15\sim PC0, PC15/14 \leftarrow RomPages1/0, PC13\sim PC0 \leftarrow d$	-	-	-	2
RET	RET	$PC \leftarrow Stack$	-	-	-	2
	NOP	No operation	-	-	-	1

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

10 ELECTRICAL CHARACTERISTIC

10.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd).....	- 0.3V ~ 6.0V
Input in voltage (Vin).....	Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr) SN8PC21P, SN8PC21S,.....	0°C ~ + 70°C
Storage ambient temperature (Tstor)	-40°C ~ + 125°C

10.2 ELECTRICAL CHARACTERISTIC

I DC CHARACTERISTIC

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

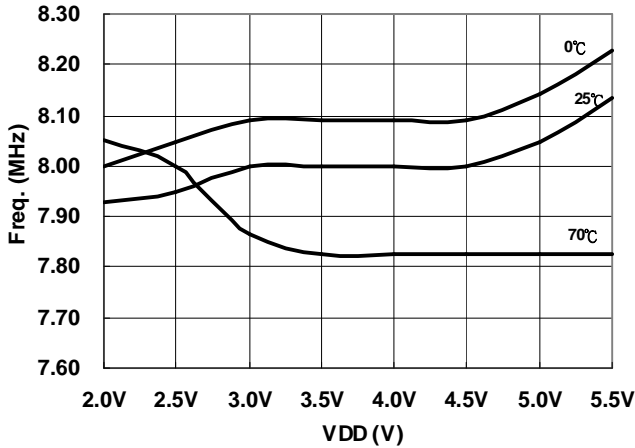
PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, 25°C, Fcpu = 2mips.	2.0	-	5.5	V
RAM Data Retention voltage	Vdr		1.5	-	-	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset	0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports	Vss	-	0.3Vdd	V
	ViL2	Reset pin	Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports	0.7Vdd	-	Vdd	V
	ViH2	Reset pin	0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd	-	-	2	uA
I/O port 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
	Idd1	Run Mode, IHRC 8MHz.	-	0.5	1	
Supply Current	Idd2	Sleep Mode	-	1	2	uA
			Vdd= 3V Vdd= 5V			
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	7.84	8	8.16	Mhz
		25°C, Vdd= 2V~5V				
*LVD Voltage	Vdet0	Low voltage reset level.	-	-	1.8	V

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

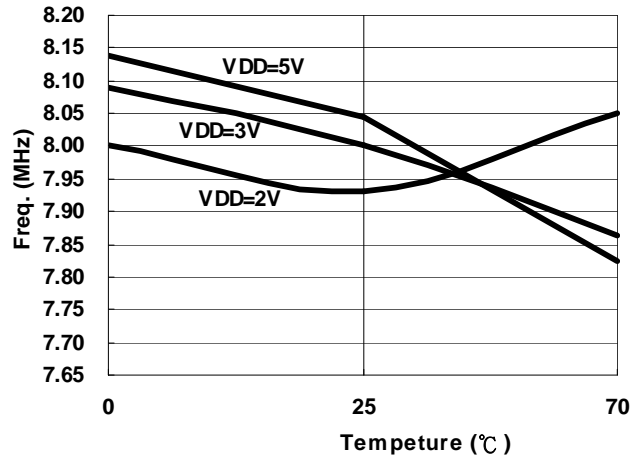
10.3 CHARACTERISTIC GRAPHS

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

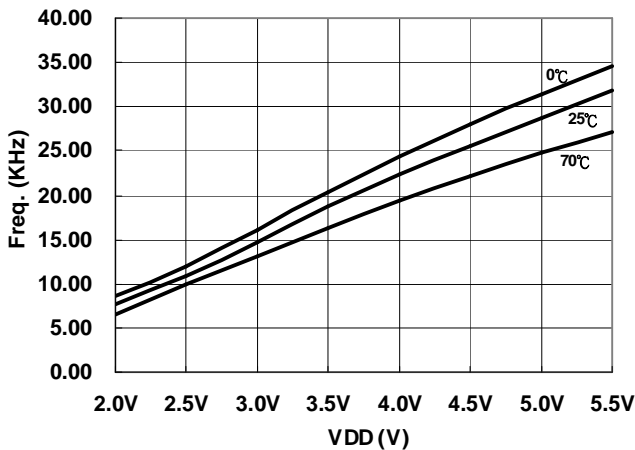
Internal High RC Oscillator (MHz)



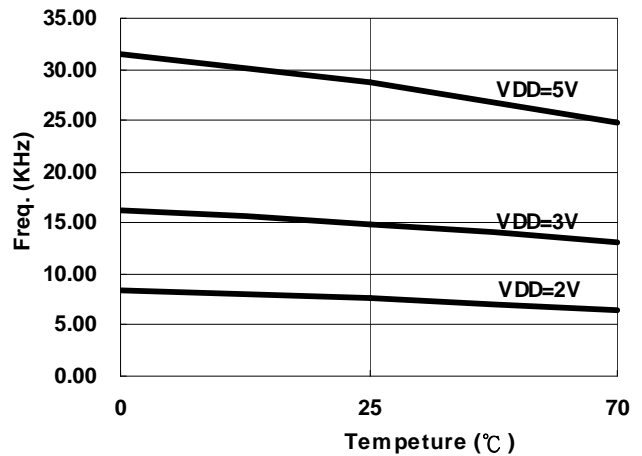
Internal High RC Oscillator (MHz)
(Fcpu=HRC/16)



Internal Low RC Oscillator (KHz)



Internal Low RC Oscillator (KHz)



11 DEVELOPMENT TOOL

SONiX provides ICE (in circuit emulation), IDE (Integrated Development Environment) and EV-kit for SN8PC21 development. ICE and EV-kit are external hardware devices, and IDE is a friendly user interface for firmware development and emulation. These development tools' version is as following.

- I ICE: SN8ICE2K
- I EV-kit: SN8PC21 EV-kit V1.0.
- I IDE: SONiX IDE M2IDE_Vxxx.
- I Writer: MPIII WRITER.

11.1 ICE AND EV-KIT APPLICATION NOTIC

SN8PC21 EV-kit includes matrix type key map and IR driving circuit module.

- I The IR driving circuit module includes two types. One is for ICE emulation to drive IR carry signal. The other one is for SN8PC21 real chip test and disconnected with ICE. The EV-kit is like a real remote controller. The R3 resistance is over 3.75Ω better for 400mA sink current @Vdd=3V.
- I If the SN8PC21 real chip is set as crystal mode, the C4 (XIN) capacitance and C5 (XOUT) capacitance are 20pF.

12 OTP PROGRAMMING PIN

12.1 THE PIN ASSIGNMENT OF EASY WRITER TRANSITION BOARD SOCKET:

Easy Writer JP1/JP2

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

JP1 for MP transition board

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

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

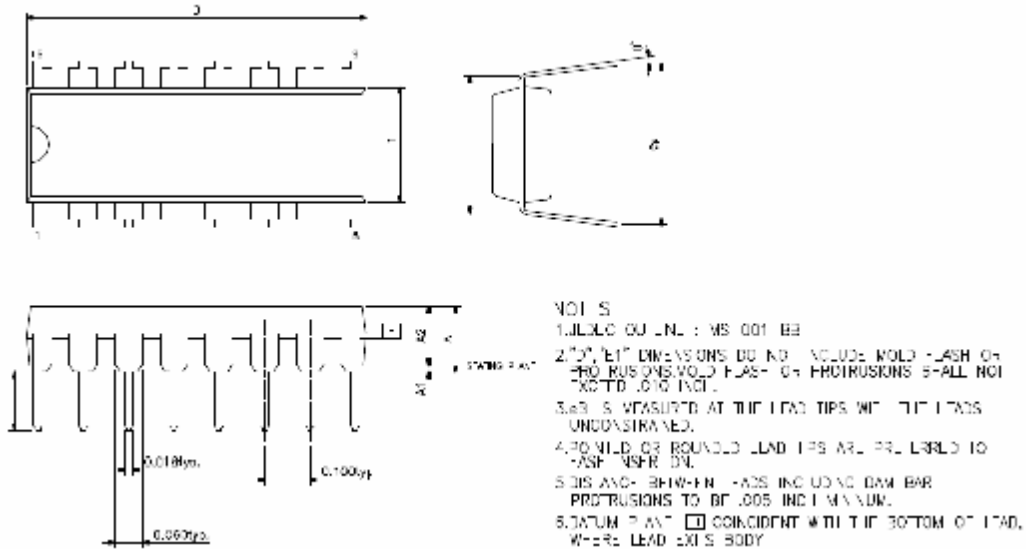
JP3 for MP transition board

12.2 PROGRAMMING PIN MAPPING:

Programming Information of SN8PC21									
Chip Name		SN8PC21P/S							
EZ Writer Connector		OTP IC / JP3 Pin Assigment							
Number	Name	Number	Pin	Number	Pin	Number	Pin		
1	VDD	16	VDD						
2	GND	15	VSS						
3	CLK	10	P1.2						
4	CE	-	-						
5	PGM	8	P1.0						
6	OE	11	P1.3						
7	D1	-	-						
8	D0	-	-						
9	D3	-	-						
10	D2	-	-						
11	D5	-	-						
12	D4	-	-						
13	D7	-	-						
14	D6	-	-						
15	VDD	-	-						
16	VPP	3	RST						
17	HLS	-	-						
18	RST	-	-						
19	-	-	-						
20	ALSB/PDB	9	P1.1						

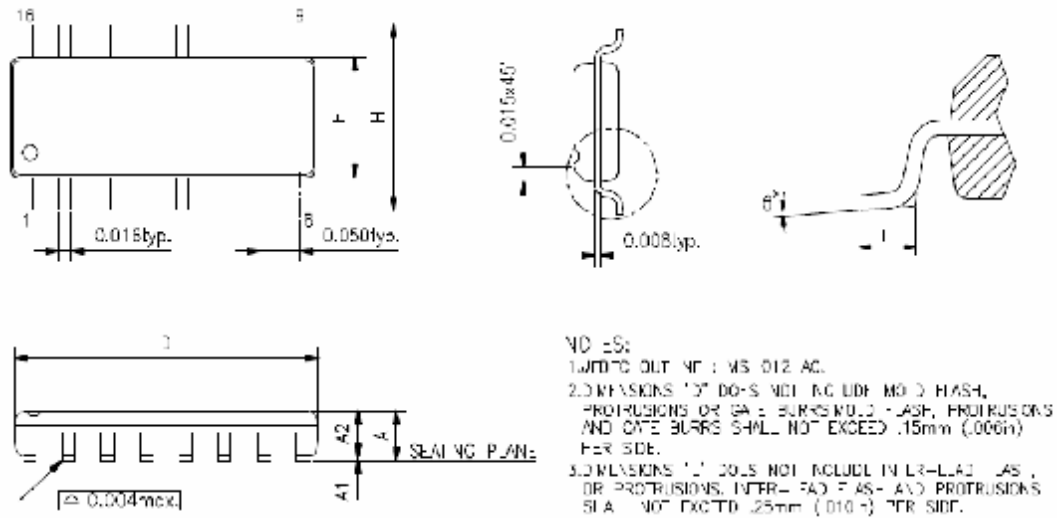
13 Package information

13.1 DIP16 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-		0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.735	0.755	0.775	18.669	19.177	19.685
E	0.30 BSC			7.620 BSC		
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°

13.2 SOP 16 PIN



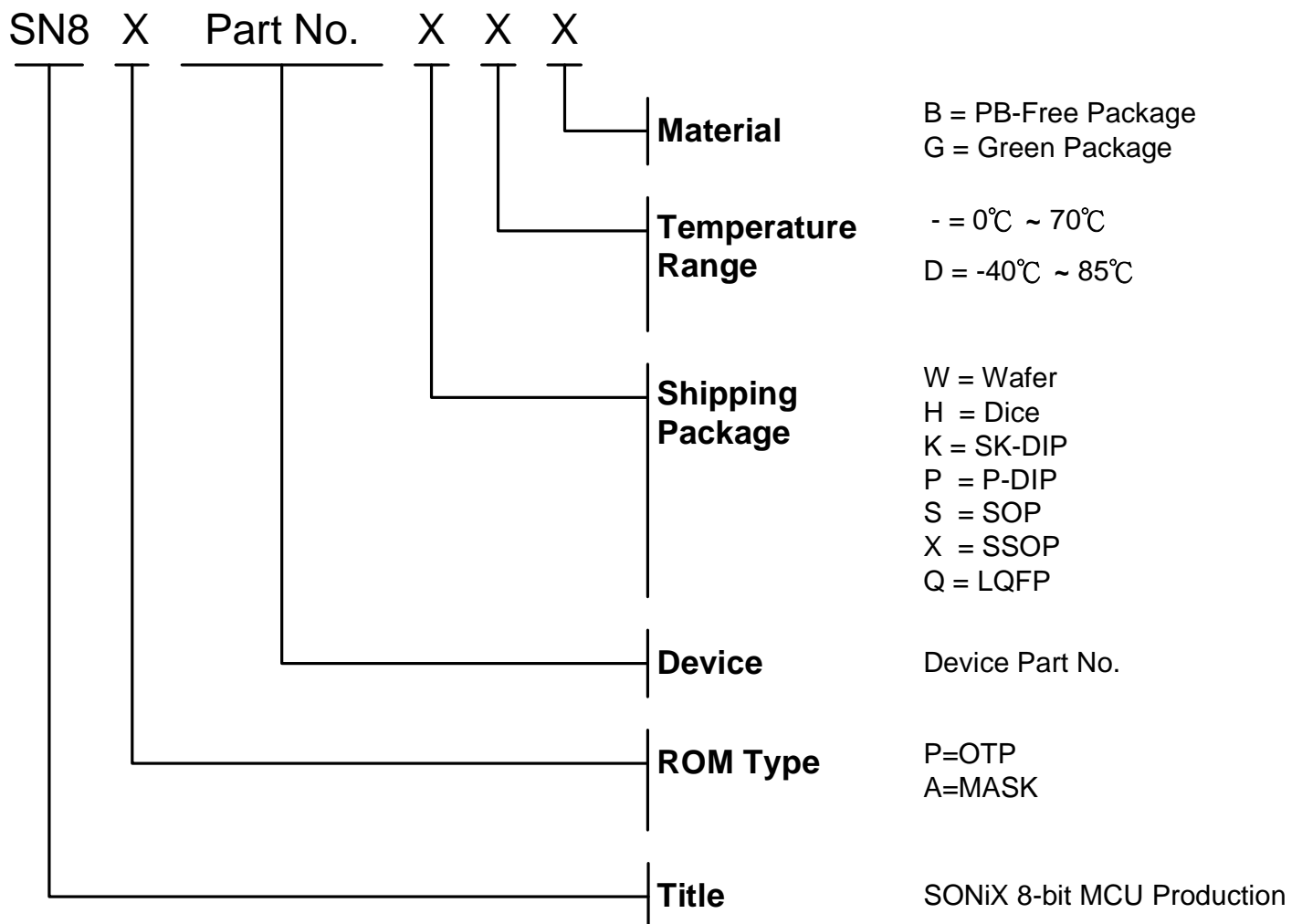
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.053	-	0.069	1.346	-	1.753
A1	0.004	-	0.010	0.102	-	0.254
D	0.386	-	0.394	9.804	-	10.008
E	0.150	-	0.157	3.810	-	3.988
H	0.228	-	0.244	5.791	-	6.198
L	0.016	-	0.050	0.406	-	1.270
θ°	0°	-	8°	0°	-	8°

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

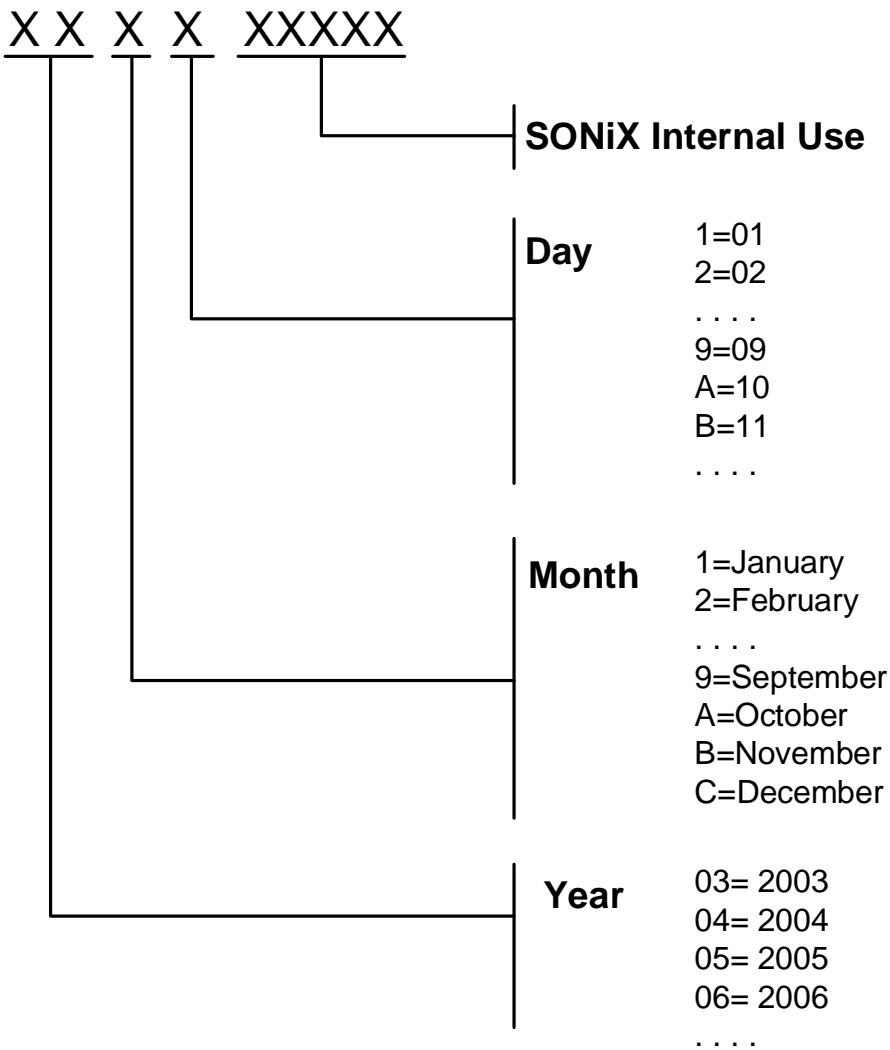


- Note: SN8PC21 doesn't support -40°C~85°C temperature range.

14.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8PC21PG	OTP	C21	P-DIP	0°C~70°C	Green Package
SN8PC21SG	OTP	C21	SOP	0°C~70°C	Green Package
SN8PC21W	OTP	C21	Wafer	0°C~70°C	-
SN8PC21H	OTP	C21	Dice	0°C~70°C	-

14.4 DATECODE SYSTEM



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