

SN8P2267C Series

USER'S MANUAL

SN8P2267CF
SN8P2267CJ

SONiX 8-Bit Micro-Controller

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

AMENDMENT HISTORY

Version	Date	Description
VER0.1	2011/09/20	version 0.1
VER0.2	2011/11/24	1. Modified the I/O port pull-up resistor.
VER0.3	2012/09/04	1. Modified the CH2 content title.

Table of Content

AMENDMENT HISTORY	2
1 PRODUCT OVERVIEW	7
1.1 FEATURES	7
1.2 SYSTEM BLOCK DIAGRAM	8
1.3 PIN ASSIGNMENT	9
1.4 PIN DESCRIPTIONS	10
1.5 PIN CIRCUIT DIAGRAMS	11
2 CENTRAL PROCESSOR UNIT (CPU)	12
2.1 MEMORY MAP	12
2.1.1 PROGRAM MEMORY (ROM)	12
2.1.1.1 RESET VECTOR(0000H)	13
2.1.1.2 INTERRUPT VECTOR(0008H)	14
2.1.1.3 LOOK-UP TABLE DESCRIPTION	16
2.1.1.4 JUMP TABLE DESCRIPTION	18
2.1.1.5 CHECKSUM CALCULATION	20
2.1.2 CODE OPTION TABLE	21
2.1.3 DATA MEMORY (RAM)	22
2.1.4 SYSTEM REGISTER	23
2.1.4.1 SYSTEM REGISTER TABLE	23
2.1.4.2 SYSTEM REGISTER DESCRIPTION	23
2.1.4.3 BIT DEFINITION of SYSTEM REGISTER	24
2.1.4.4 ACCUMULATOR	26
2.1.4.5 PROGRAM FLAG	27
2.1.4.6 PROGRAM COUNTER	28
2.1.5 Y/H,Z/L REGISTERS	31
2.1.6 R REGISTERS	32
2.2 ADDRESSING MODE	33
2.2.1 IMMEDIATE ADDRESSING MODE	33
2.2.2 DIRECTLY ADDRESSING MODE	33
2.2.3 INDIRECTLY ADDRESSING MODE	33
2.3 STACK OPERATION	34
2.3.1 OVERVIEW	34
2.3.2 STACK REGISTERS	35
2.3.3 STACK OPERATION EXAMPLE	36
3 RESET	37

3.1	OVERVIEW	37
3.2	POWER ON RESET	38
3.3	WATCHDOG RESET	38
3.4	BROWN OUT RESET	39
3.4.1	BROWN OUT DESCRIPTION	39
3.4.2	THE SYSTEM OPERATING VOLTAGE DECSRIPTION	40
3.4.3	BROWN OUT RESET IMPROVEMENT	40
3.5	EXTERNAL RESET	42
3.6	EXTERNAL RESET CIRCUIT	42
3.6.1	Simply RC Reset Circuit	42
3.6.2	Diode & RC Reset Circuit	43
3.6.3	Zener Diode Reset Circuit	43
3.6.4	Voltage Bias Reset Circuit	44
3.6.5	External Reset IC	45
4	SYSTEM CLOCK	46
4.1	OVERVIEW	46
4.2	CLOCK BLOCK DIAGRAM	46
4.3	OSCM REGISTER	47
4.4	SYSTEM HIGH CLOCK	48
4.4.1	INTERNAL HIGH RC	48
4.5	SYSTEM LOW CLOCK	48
4.5.1	SYSTEM CLOCK MEASUREMENT	49
5	SYSTEM OPERATION MODE	50
5.1	OVERVIEW	50
5.2	SYSTEM MODE SWITCHING EXAMPLE	51
5.3	WAKEUP	52
5.3.1	OVERVIEW	52
5.3.2	WAKEUP TIME	52
6	INTERRUPT	53
6.1	OVERVIEW	53
6.2	INTEN INTERRUPT ENABLE REGISTER	54
6.3	INTRQ INTERRUPT REQUEST REGISTER	54
6.4	GIE GLOBAL INTERRUPT OPERATION	55
6.5	PUSH, POP ROUTINE	56
6.6	INT0 (P0.0) INTERRUPT OPERATION	57
6.7	TC0 INTERRUPT OPERATION	59
6.8	USB INTERRUPT OPERATION	60
6.9	WAKEUP INTERRUPT OPERATION	61

6.10	MULTI-INTERRUPT OPERATION	62
7	I/O PORT	63
7.1	I/O PORT MODE	63
7.2	I/O PULL UP REGISTER	64
7.3	I/O PORT DATA REGISTER	65
8	TIMERS	66
8.1	WATCHDOG TIMER.....	66
8.2	TIMER/COUNTER 0 (TC0)	68
8.2.1	OVERVIEW	68
8.2.2	TC0M MODE REGISTER	69
8.2.3	TC0C COUNTING REGISTER	70
8.2.4	TC0R AUTO-LOAD REGISTER	71
8.2.5	TC0 TIMER OPERATION SEQUENCE	72
9	UNIVERSAL SERIAL BUS (USB).....	73
9.1	OVERVIEW	73
9.2	USB MACHINE.....	73
9.3	USB INTERRUPT.....	74
9.4	USB ENUMERATION	74
9.5	USB REGISTERS	75
9.5.1	USB DEVICE ADDRESS REGISTER	75
9.5.2	USB STATUS REGISTER.....	75
9.5.3	USB DATA COUNT REGISTER	75
9.5.4	USB ENABLE CONTROL REGISTER.....	76
9.5.5	USB endpoint's ACK handshaking flag REGISTER	76
9.5.6	USB ENDPOINT 0 ENABLE REGISTER	77
9.5.7	USB ENDPOINT 1 ENABLE REGISTER	77
9.5.8	USB ENDPOINT 2 ENABLE REGISTER	78
9.5.9	UPID REGISTER	78
9.5.10	ENDPOINT TOGGLE BIT CONTROL REGISTER.....	79
9.5.11	NDT BIT CONTROL REGISTER	79
9.5.12	ENDPOINT CONTROL REGISTER	79
10	INSTRUCTION TABLE	80
11	DEVELOPMENT TOOL	82
11.1	ICE (IN CIRCUIT EMULATION).....	82
11.2	SN8P2267C EV-KIT	83
11.3	SN8P2267C TRANSITION BOARD	83

12	ELECTRICAL CHARACTERISTIC	84
12.1	ABSOLUTE MAXIMUM RATING	84
12.2	ELECTRICAL CHARACTERISTIC	84
13	OTP ROM PROGRAMMING PIN	85
14	PACKAGE INFORMATION	86
14.1	LQFP 48 PIN	86
14.2	QFN 46 PIN	87
15	MARKING DEFINITION.....	88
15.1	INTRODUCTION	88
15.2	MARKING INDETIFICATION SYSTEM.....	88
15.3	MARKING EXAMPLE	89
15.4	DATECODE SYSTEM	89

1 PRODUCT OVERVIEW

1.1 FEATURES

Memory configuration.

OTP ROM size: 4K x 16 bits.

RAM size: 160 x 8 bits.

◆ 8 levels stack buffer.

◆ I/O pin configuration.

Bi-directional: P0, P1, P2, P3, P5

Wake-up: P0/P1/P2/P3/P5 level change.

Pull-up resistors: P0, P1, P2, P3, P5

External interrupt: P0.0 controlled by PEDGE.

◆ Low Speed USB 2.0

Conforms to USB Specification, Version 2.0

3.3V regulator output for USB D- pin internal

1.5k ohm pull-up resistor.

Integrated USB transceiver.

Supports 1 Low speed USB device address and

1 control endpoint has 8 bytes FIFO

2 interrupt endpoints, each has 8 bytes FIFO

◆ Powerful instructions.

Instruction cycle controlled by code option.

Instruction's length is one word.

Most of instructions are one cycle only.

Maximum instruction cycle is two.

All ROM area JMP instruction.

All ROM area lookup table function (MOVC)

◆ USB mode only support.

◆ 5 interrupt sources.

4 internal interrupts: TC0, USB, NDT, Wakeup.

1 external interrupts: INT0

◆ 8 bits timer counter (TC0)

◆ On chip watchdog timer.

◆ Two system clocks.

Internal high clock: Fcpu(max) = 6MHz

Internal low clock: RC type 32KHz (5V).

◆ Four operating modes.

Normal mode: Both high and low clock active

Slow mode: Low clock only

Sleep mode: Both high and low clock stop

Green mode: Periodical wakeup by timer

◆ Package (Chip form support).

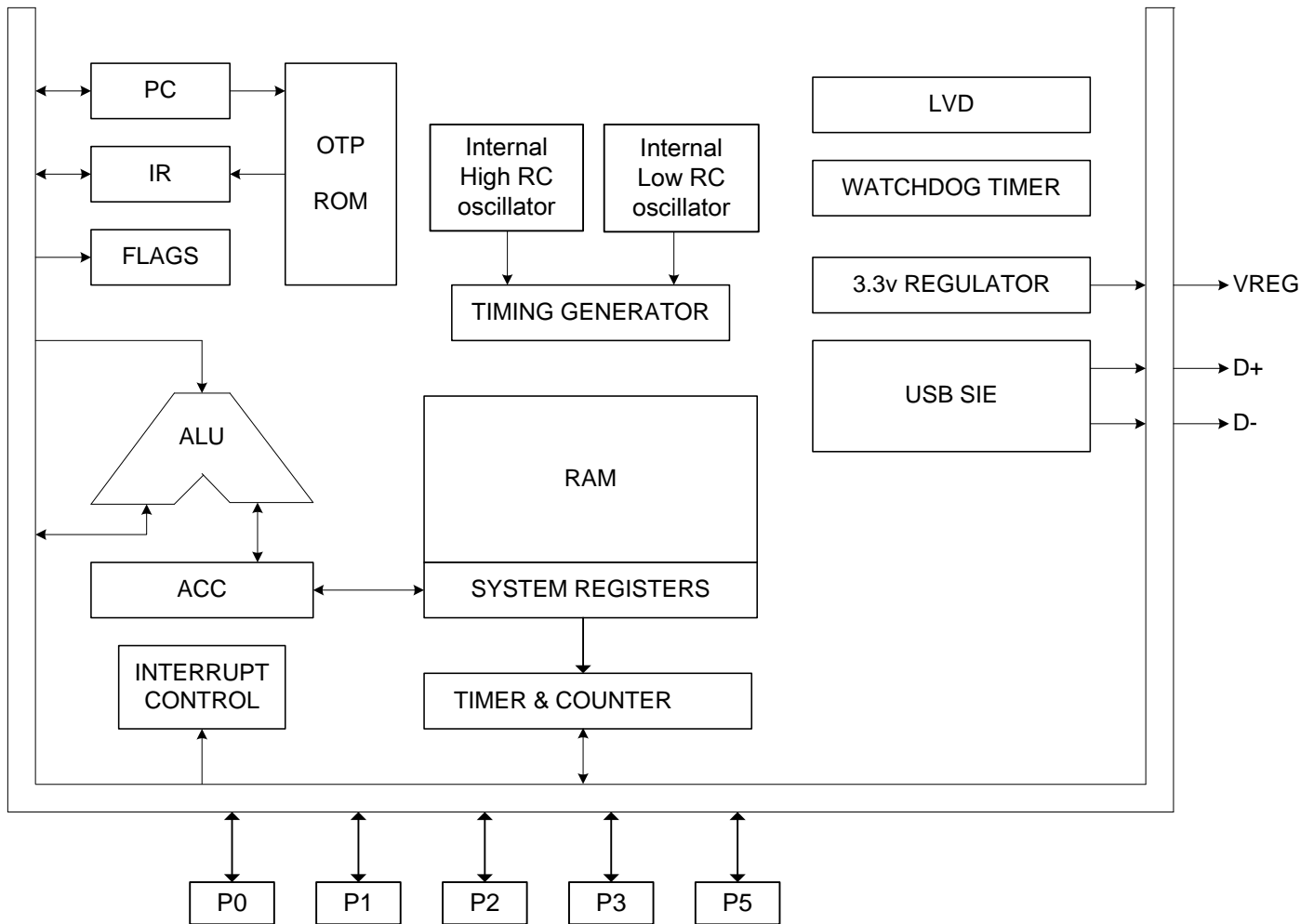
LQFP 48 pin

QFN 46 pin

Features Selection Table

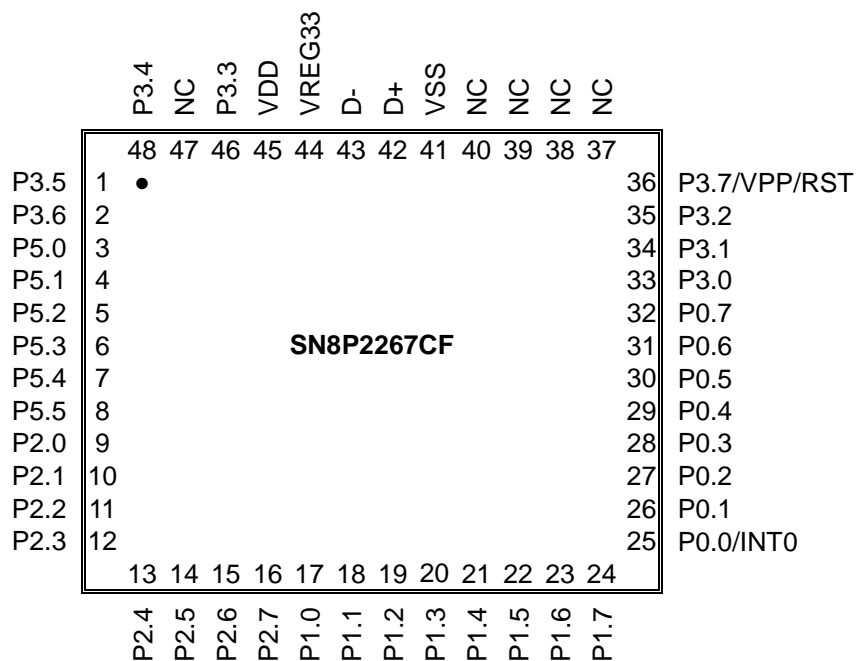
CHIP	ROM	RAM	STACK	TC0 TIMER	USB	I/O	WAKE-UP PIN NO.	PACKAGE
SN8P2267C	4K*16	160*8	8	v	v	38	37	LQFP48
SN8P2267C	4K*16	160*8	8	v	v	38	37	QFN46

1.2 SYSTEM BLOCK DIAGRAM

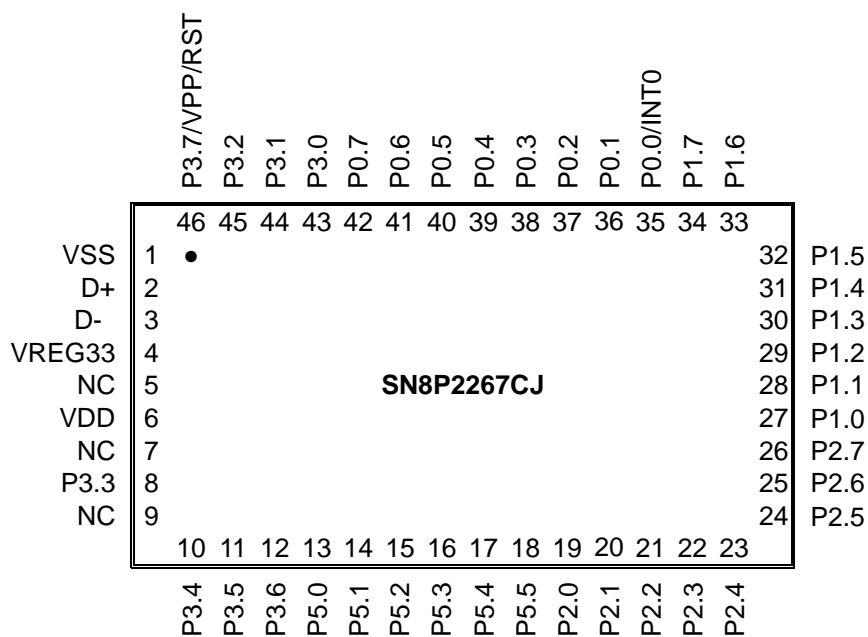


1.3 PIN ASSIGNMENT

SN8P2267CF (LQFP 48 pins)



SN8P2267CJ (QFN 46 pins)

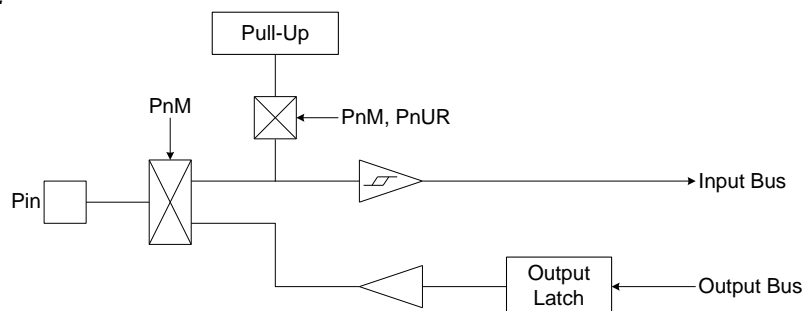


1.4 PIN DESCRIPTIONS

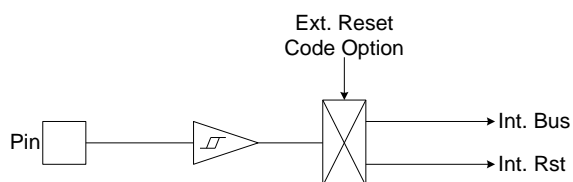
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital circuit.
P0.0/INT0	I/O	P0.0: Port 0.0 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function. INT0: External interrupt 0 input pin.
P0[7:1]	I/O	P0: Port 0 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function.
P1[7:0]	I/O	P1: Port 1 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function.
P2[7:0]	I/O	P2: Port 2 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function.
P3.7/RST/VPP	I, P	RST is system external reset input pin under Ext_RST mode. Schmitt trigger structure, active "low", normal stay to "high". P3.7 is input only pin without pull-up resistor under P3.7 mode. Built wakeup function. OTP 12.3V power input pin in programming mode.
P3[6:0]	I/O	P3: Port 3 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function.
P5[5:0]	I/O	P5: Port 5 bi-direction pin. Schmitt trigger structure and built-in pull-up resistors as input mode. Built wakeup function.
VREG33	O	3.3V voltage output from USB 3.3V regulator.
D+, D-	I/O	USB differential data line.

1.5 PIN CIRCUIT DIAGRAMS

Port 0, 1, 2, 3, 5 structures:



Port 3.7 structure:



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

 4K words ROM

ROM		
0000H	<i>Reset vector</i>	User reset vector Jump to user start address
0001H	<i>General purpose area</i>	User interrupt vector User program
.		
.		
0007H		
0008H	<i>Interrupt vector</i>	End of user program
0009H	<i>General purpose area</i>	
.		
.		
000FH		
0010H		
0011H		
.		
.		
.		
.		
0FF8H	<i>Reserved</i>	
.		
0FFFH		

2.1.1.1 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
```

2.1.1.2 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.

2.1.1.3 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 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
        ...

```

2.1.1.4 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
ADD      PCL, A
ENDM

```

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

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

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

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

➤ **Example: “@JMP_A” operation.**

; Before compiling program.

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

; After compiling program.

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

2.1.1.5 CHECKSUM CALCULATION

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

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

```

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

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

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

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

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

CHECKSUM_END:
...
...

END_USER_CODE:             ; Label of program end

```

2.1.2 CODE OPTION TABLE

Code Option	Content	Function Description
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
Fcpu	Fhosc/1	Instruction cycle is 6 MHz clock.
	Fhosc/2	Instruction cycle is 3 MHz clock.
	Fhosc/4	Instruction cycle is 1.5 MHz clock.
Reset_Pin	Reset	Enable External reset pin without pull up resistor.
	P37	Enable P3.7 I/O function.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
External Reset Length	No	Disable External reset de-bounce time.
	128 * ILRC	Enable External reset de-bounce time.

* **Note:** Fcpu code option is only available for High Clock. Fcpu of slow mode is Fhosc/4.

2.1.3 DATA MEMORY (RAM)

☞ 160 X 8-bit RAM

	Address	RAM location	
BANK 0	000h	General purpose area	BANK 0
	"		
	"		
	"		
	"		
	07Fh		
	080h	System register	80h~FFh of Bank 0 store system registers (128 bytes).
	"		
	"		
	"		
	"		
	0FFh	End of bank 0 area	
BANK1	100h	General purpose area	BANK1
	"		
	"		
	"		
	"		
	11Fh		
	120h	Endpoint 0 RAM (8 byte)	USB FIFO RAM in BANK1
	127h		
		Reserved	
	130h	Endpoint 1 RAM (8 byte)	
	137h		
	138h	Endpoint 2 RAM (8 byte)	
	13Fh		

2.1.4 SYSTEM REGISTER

2.1.4.1 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	TC0M	TC0C	TC0R	-	-	-	-	-
9	UDA	USTAT US	EP0OUT _CNT	USB_IN T_EN	EP _ACK		UE0R	UE1R	UE2R	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	UPID	UToggle	-	UNDT	-
B	EP0_S TALL	-	-	-	-	-	-	--	P0M	-	-	-	-	-	-	PEDGE
C		P1M	P2M	P3M	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	-	PCL	PCH
D	P0	P1	P2	P3	-	P5	-	-	-	-	-	-	-	-	-	STKP
E	P0UR	P1UR	P2UR	P3UR	-	P5UR	@HL	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.1.4.2 SYSTEM REGISTER DESCRIPTION

R = Working register and ROM look-up data buffer.
PFLAG = ROM page and special flag register.
UDA = USB control register.
UE0R~UE2R = Endpoint 0~2 control registers.
EP_ACK = Endpoint ACK and NAK flag register.
UToggle = USB endpoint toggle bit control register.
USTATUS = USB status register.
EP0OUT_CNT = USB endpoint 0 OUT token data byte counter
PnM = Port n input/output mode register.
INTRQ = Interrupt request register.
OSCM = Oscillator mode register.
TC0R = TC0 auto-reload data buffer.
Pn = Port n data buffer.
TnC = T0 counting register. n = C0
PnUR = Port n pull-up resistor control register.
@HL = RAM HL indirect addressing index pointer.

H, L = Working, @HL register.
Y, Z = Working, @YZ and ROM addressing register.
RBANK = RAM bank selection register.
UNDT_EN= NDT control registers
UNDT= NDT flag registers
UPID = USB bus control register.
USB_INT_EN = USB interrupt enable/disable control register.
PEDGE = P0.0 edge direction register.
INTEN = Interrupt enable register.
WDTR = Watchdog timer clear register.
PCH, PCL = Program counter.
TnM = Tn mode register. n = C0
TnR = Tn register. n = C0
STKP = Stack pointer buffer.
@YZ = RAM YZ indirect addressing index pointer.
STK0~STK7 = Stack 0 ~ stack 7 buffer.

2.1.4.3 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	LVD36	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	-	-	RBNKS0	R/W	RBANK
088H	TC0ENB	TC0rate2	TC0rate1	TC0rate0	-	ALOAD0	-	-	R/W	TC0M
089H	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
08AH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	R/W	TC0R
090H	UDE	UDA6	UDA5	UDA4	UDA3	UDA2	UDA1	UDA0	R/W	UDA
091H	-	-	-	BUS_RST	SUSPEND	EP0SETUP	EP0IN	EP0OUT	R/W	USTATUS
092H	-	-	-	UEP0OC4	UEP0OC3	UEP0OC2	UEP0OC1	UEP0OC0	R/W	EP0OUT_CNT
093H	REG_EN	DN_PU_EN	-	-	EP2_NAK_E N	EP1_NAK_E N	UE2D	UE1D	R/W	USB_INT_EN
094H	-	-	-	-	EP2_NAK	EP1_NAK	EP2_ACK	EP1_ACK	R/W	EP_ACK
096H	UE0E	UE0M1	UE0M0	-	UE0C3	UE0C2	UE0C1	UE0C0	R/W	UE0R
097H	UE1E	UE1M1	UE1M0	-	UE1C3	UE1C2	UE1C1	UE1C0	R/W	UE1R
098H	UE2E	UE2M1	UE2M0	-	UE2C3	UE2C2	UE2C1	UE2C0	R/W	UE2R
0ABH	EP0OUT_E N	-	-	-	-	UBDE	DDP	DDN	R/W	UPID
0ACH	-	-	-	-	-	-	EP2_DATA0 /1	EP1_DATA0 /1	R/W	Utoggle
0AEH	-	-	-	-	-	-	-	NDT	R/W	UNDT
0B0H	-	-	-	-	-	-	EP0_OUT_S TALL	EP0_IN_ST ALL	R/W	EP0_STALL
0B8H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0BFH	-	-	-	-	-	-	P00G1	P00G0	R/W	PEDGE
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M
0C3H	-	P36M	P35M	P34M	P33M	P32M	P31M	P30M	R/W	P3M
0C5H	-	-	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H	NDTIRQ	USBIRQ	TC0IRQ	-	-	WAKEIRQ	-	P00IRQ	R/W	INTRQ
0C9H	NDTIEN	USBIEN	TC0IEN	-	-	WAKEIEN	-	P00IEN	R/W	INTEN
0CAH	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	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
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2
0D3H	P37	P36	P35	P34	P33	P32	P31	P30	R/W	P3
0D5H	-	-	P55	P54	P53	P52	P51	P50	R/W	P5
0DFH	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2UR
0E3H	-	P36R	P35R	P34R	P33R	P32R	P31R	P30R	W	P3UR
0E5H	-	-	P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	@HL7	@HL6	@HL	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	-	-	-	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H

0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	-	-	-	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	-	-	-	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	-	-	-	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	-	-	-	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	-	-	-	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	-	-	-	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	-	-	-	S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

*** Note:**

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

2.1.4.4 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 or DC) occurrence, then these flags will be set to PFLAG register. ACC is not in data memory (RAM), so ACC can't be access by "B0MOV" instruction during the instant addressing mode.

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

; Read ACC data and store in BUF data memory.

```
MOV      BUF, A
```

; Write a immediate data into ACC.

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

; Write ACC data from BUF data memory.

```
MOV      A, BUF
```

; or

```
B0MOV    A, BUF
```

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

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

INT_SERVICE:

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

```
...
```

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

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

2.1.4.5 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. LVD36 bit indicates LVD detecting power voltage status.

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

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

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

Bit 5 **LVD36**: LVD 3.6V operating flag.

0 = Inactive ($VDD > 3.6V$).

1 = Active ($VDD \leq 3.6V$).

Bit 2 **C**: Carry flag

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

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

Bit 1 **DC**: Decimal carry flag

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

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

Bit 0 **Z**: Zero flag

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

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

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

2.1.4.6 PROGRAM COUNTER

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

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

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

☞ 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

✎ MULTI-ADDRESS JUMPING

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

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

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

; PC = 0323H

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

; PC = 0328H

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

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

; PC = 0323H

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

2.1.5 Y/H,Z/L REGISTERS

The Y/H and Z/L 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, @HL register
- can be used as ROM data pointer with the MOVC instruction for look-up table

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	-	-	-	-	-	-	-	-

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	-	-	-	-	-	-	-	-

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	-	-	-	-	-	-	-	-

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	-	-	-	-	-	-	-	-

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_CLR: ; End of clear general purpose data memory area of bank 0

...

2.1.6 R REGISTERS

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

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

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

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

2.2 ADDRESSING MODE

2.2.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.2.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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

2.2.3 INDIRECTLY ADDRESSING MODE

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

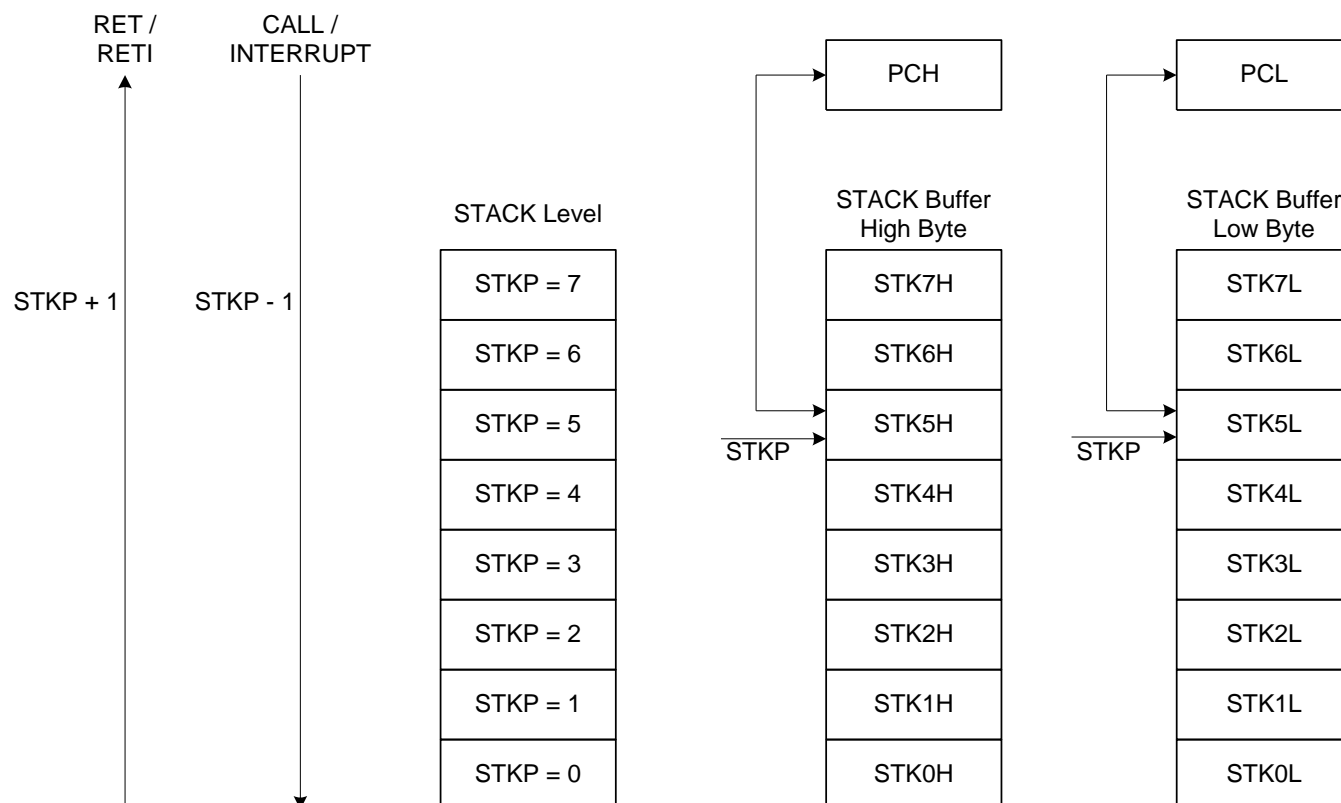
Example: Indirectly addressing mode with @YZ register

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.

2.3 STACK OPERATION

2.3.1 OVERVIEW

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



2.3.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 13-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

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

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

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

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

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

- **Example: Stack pointer (STKP) reset, we strongly recommended to clear the stack 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	-	-	-	SnPC12	SnPC11	SnPC10	SnPC9	SnPC8
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

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

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

2.3.3 STACK OPERATION EXAMPLE

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

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

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

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

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

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

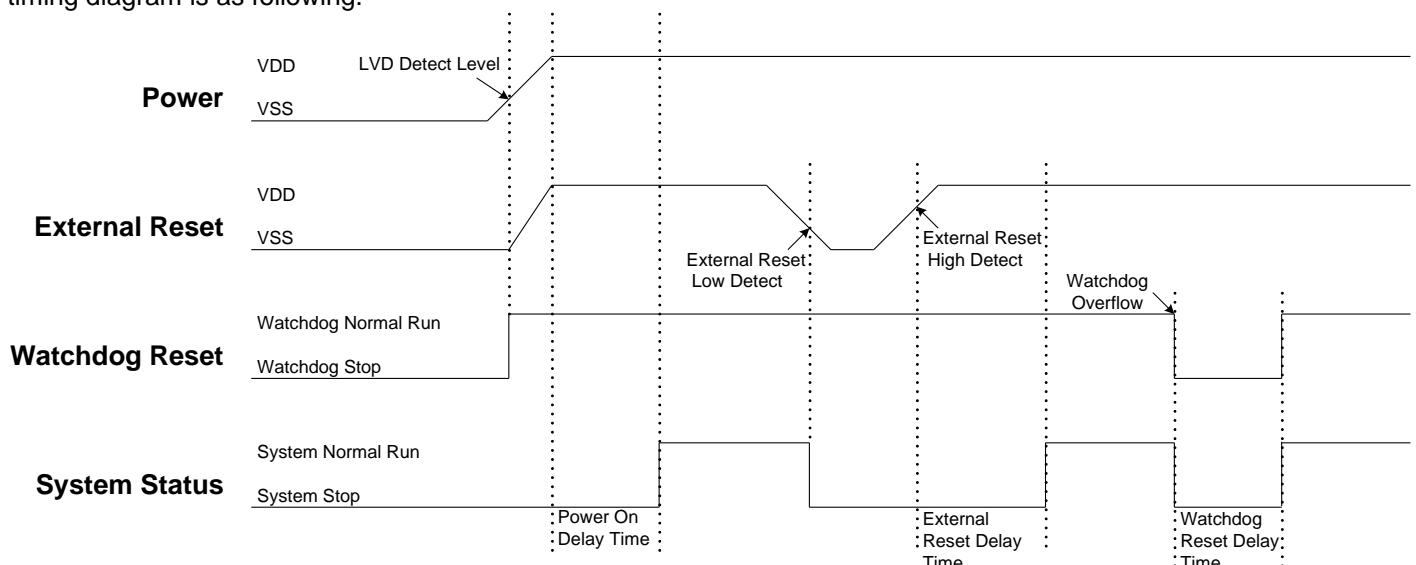
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	-	-	-	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 timer application note is as following.

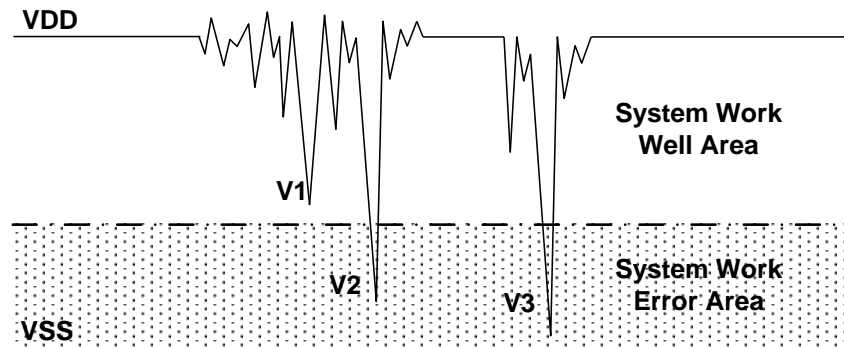
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

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

3.4 BROWN OUT RESET

3.4.1 BROWN OUT DESCRIPTION

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



Brown Out Reset Diagram

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

DC application:

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

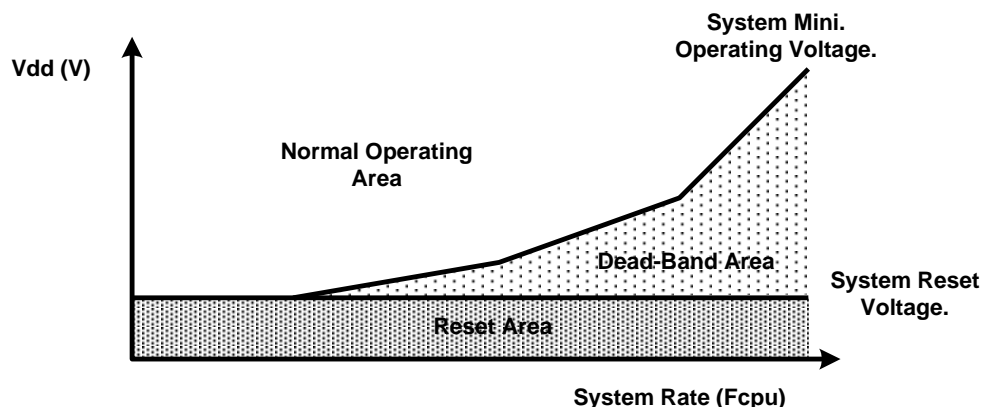
AC application:

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

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

3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION

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



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

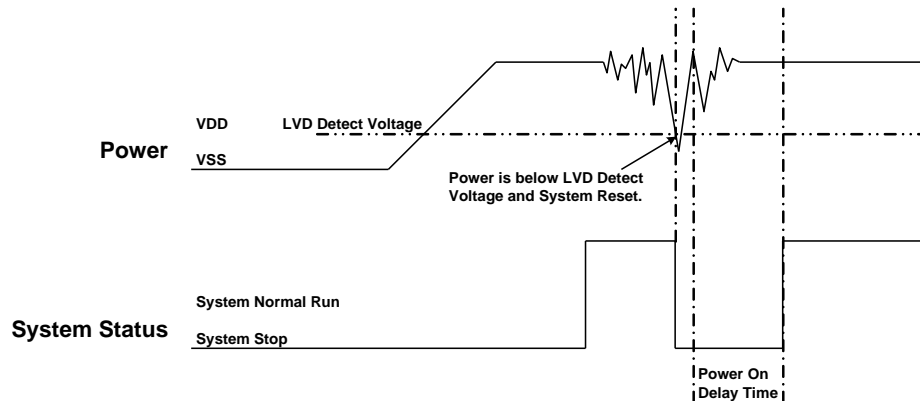
3.4.3 BROWN OUT RESET IMPROVEMENT

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.

LVD reset:

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.

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

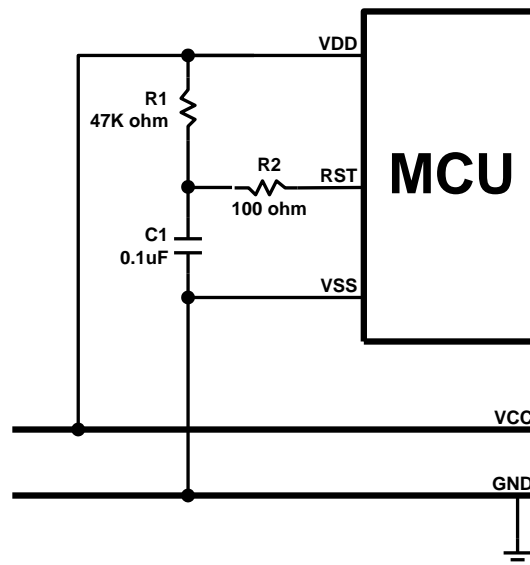
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

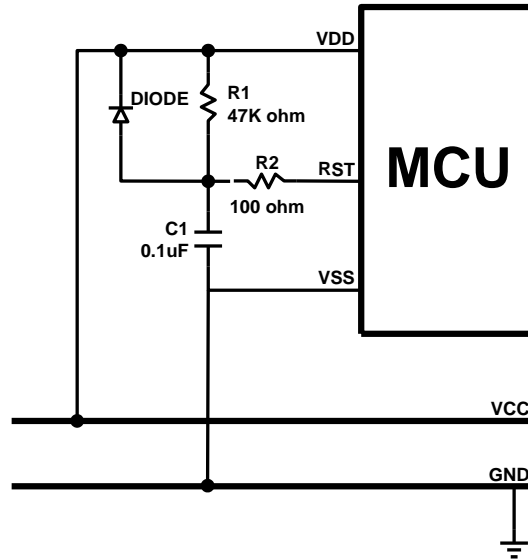
3.6.1 Simply RC Reset Circuit



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.

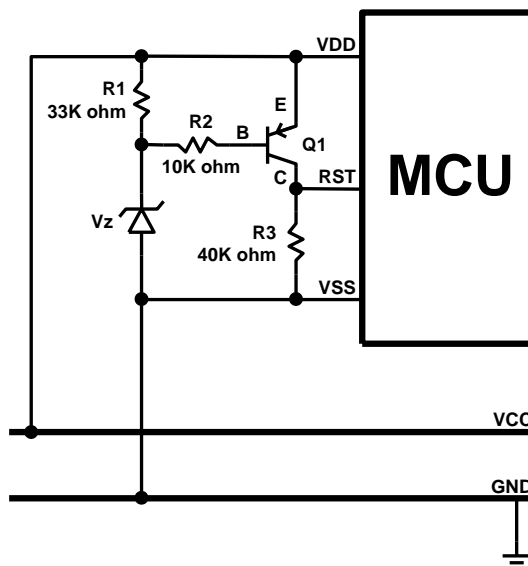
3.6.2 Diode & RC Reset Circuit



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.

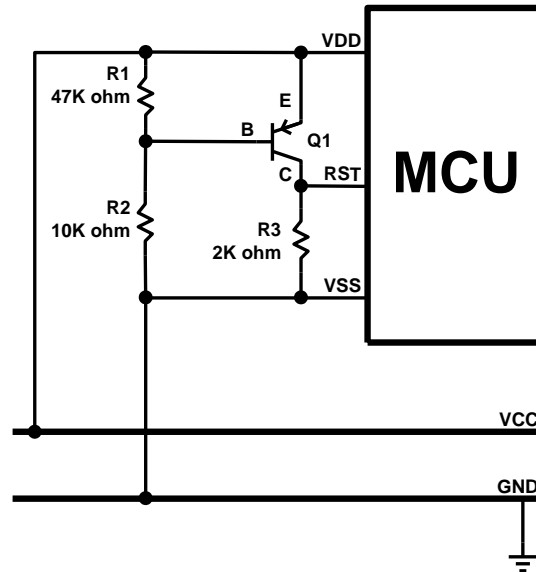
* **Note:** The R2 100 ohm resistor of “Simply reset circuit” and “Diode & RC reset circuit” is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).

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

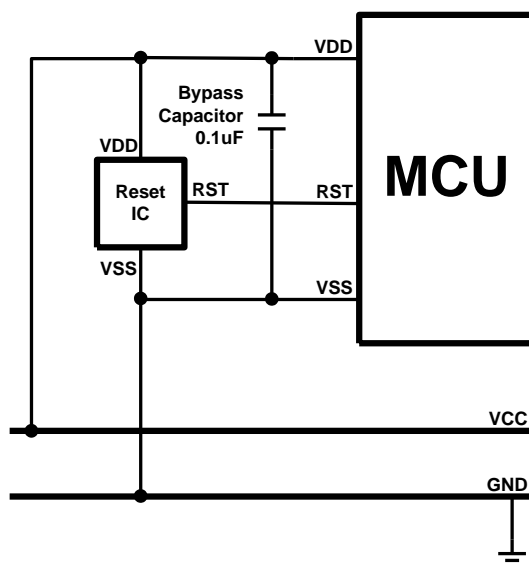


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to $0.7V \times (R1 + R2) / R1$, 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. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator & on-chip PLL circuit. The low-speed clock is generated from on-chip low-speed RC oscillator circuit (ILRC 32KHz).

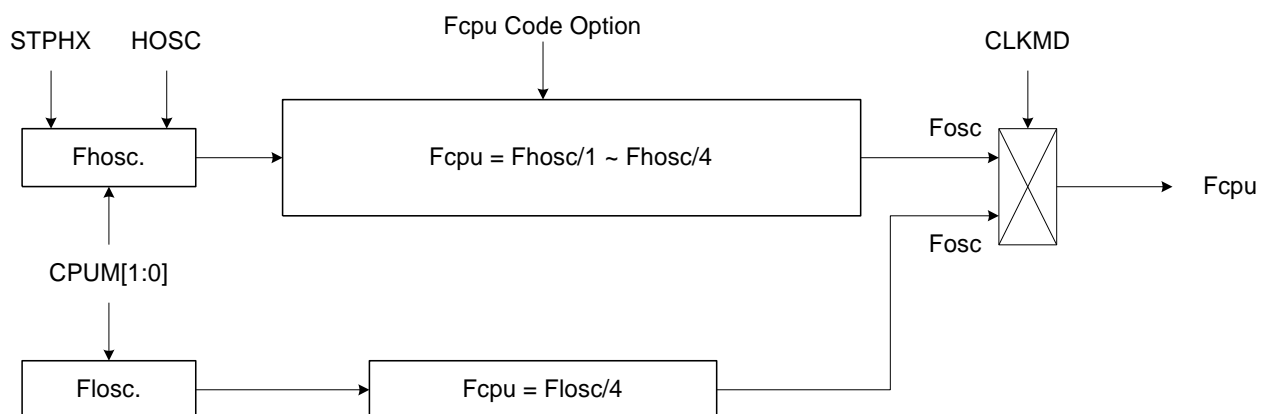
Both the high-speed clock and the low-speed clock can be system clock (Fosc). The system clock in slow mode is divided by 4 to be the instruction cycle (Fcpu).

☞ **Normal Mode (High Clock):** $F_{cpu} = F_{osc} / N$, $N = 1 \sim 4$, Select N by Fcpu code option.

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

SONiX provides a “**Noise Filter**” controlled by code option. In high noisy situation, the noise filter can isolate noise outside and protect system works well. The minimum Fcpu of high clock is limited at **Fosc/4** when noise filter enable.

4.2 CLOCK BLOCK DIAGRAM



- HOSC: High_Clk code option.
- Fosc: Internal high-speed clock.
- Fosc: Internal low-speed RC clock (Typical 32 KHz).
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.3 OSCM REGISTER

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

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

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

➤ **Example: Stop high-speed oscillator and PLL circuit.**

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

Example: When entering the power down mode (sleep mode), both high-speed external oscillator, PLL circuit 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 6MHz oscillator.

4.4.1 INTERNAL HIGH RC

The chip is built-in RC type internal high clock (6MHz). The system clock is from internal 6MHz RC type oscillator.

IHRC: High clock is internal 6MHz oscillator RC type.

4.5 SYSTEM LOW CLOCK

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

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

☞ ***Fosc = Internal low RC oscillator (32KHz).***

☞ ***Slow mode Fcpu = Fosc / 4***

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

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

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

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

4.5.1 SYSTEM CLOCK MEASUREMENT

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

Example: Fcpu instruction cycle of external oscillator.

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

@ @:

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

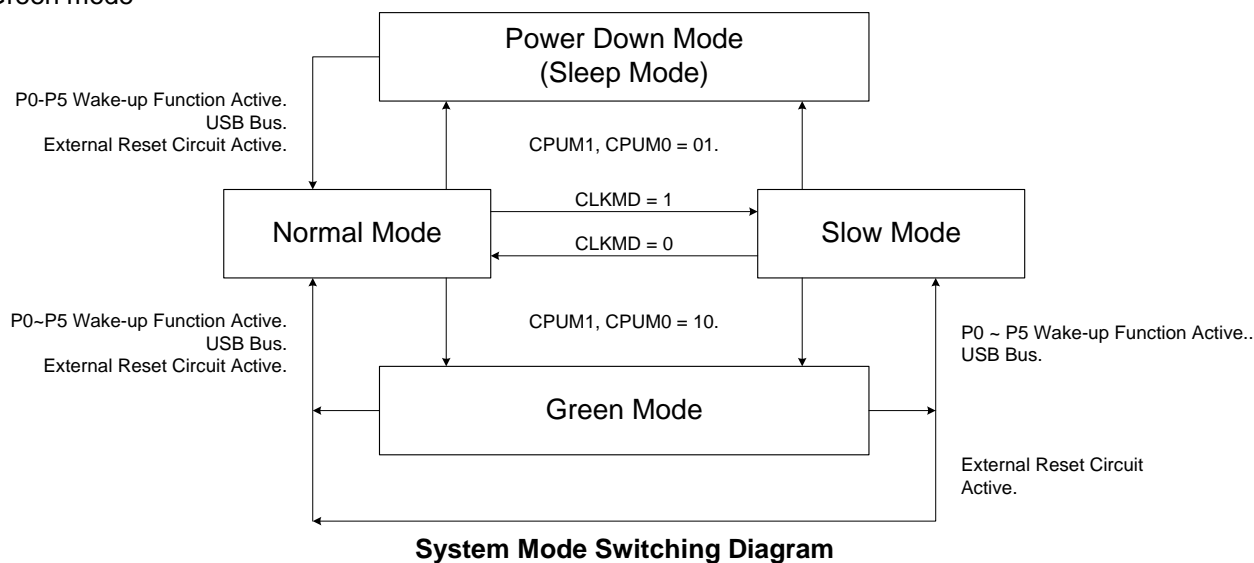
* **Note: Do not measure the RC frequency directly from XIN; the probe impedance will affect the RC frequency.**

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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



Operating mode description

MODE	NORMAL	SLOW	GREEN	POWER DOWN (SLEEP)	REMARK
IHRC	Running	By STPHX	By STPHX	Stop	
ILRC	Running	Running	Running	Stop	
CPU instruction	Executing	Executing	Stop	Stop	
TC0 timer	*Active	*Active	Inactive	Inactive	* Active if TC0ENB=1
USB	Running	Inactive	Inactive	Inactive	* Active if USBE=1
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	Refer to code option description
Internal interrupt	All active	All active	All inactive	All inactive	
External interrupt	All active	All active	All active	All inactive	
Wakeup source	-	-	P0, P1, P2, P3, P5, Reset	P0, P1, P2,P3, P5 Reset	

- **IHRC:** Internal high clock (6MHz RC oscillator)
- **ILRC:** Internal low clock (32KHz RC oscillator)

5.2 SYSTEM MODE SWITCHING EXAMPLE

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

```
B0BSET      FCPUM0      ; Set CPUM0 = 1.
```

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

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

```
B0BSET      FCLKMD      ;To set CLKMD = 1, Change the system into slow mode
B0BSET      FSTPHX      ;To stop external high-speed oscillator for power saving.
```

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

```
B0BCLR      FCLKMD      ;To set CLKMD = 0
```

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

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

```

B0BCLR      FSTPHX      ; Turn on the external high-speed oscillator.

MOV         A, #20      ; internal RC=32KHz (typical) will delay
B0MOV       Z, A
@@: DECMS    Z           ; 0.33ms X 30 ~ 10ms for external clock stable
    JMP      @B
;
B0BCLR      FCLKMD      ; Change the system back to the normal mode

```

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

```
B0BSET      FCPUM1      ; Set CPUM1 = 1.
```

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

5.3 WAKEUP

5.3.1 OVERVIEW

Under power down mode (sleep mode) or green mode, program doesn't execute. The wakeup trigger can wake the system up to normal mode or slow mode. The wakeup trigger sources are external trigger (P0, P1, P2, P3, P5 level change) and USB bus toggle.

- Power down mode is waken up to normal mode. The wakeup trigger is only external trigger (P0, P1, P2, P3, P5 level change and USB bus toggle)
- Green mode is waken up to last mode (normal mode or slow mode). The wakeup triggers are external trigger (P0, P1, P2, P3, P5 level change) and USB bus toggle.

5.3.2 WAKEUP TIME

When the system is in power down mode (sleep mode), the high clock oscillator stops. When waked up from power down mode, MCU waits for 4 internal 6MHz clock or 2048 external 6MHz clocks as the wakeup time to stable the oscillator circuit. After the wakeup time, the system goes into the normal mode.

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

The value of the wakeup time is as the following.

“6MHz IHRC” mode:

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

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

Example: In 6MHz IHRC mode and power down mode (sleep mode), the system is waked up. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.

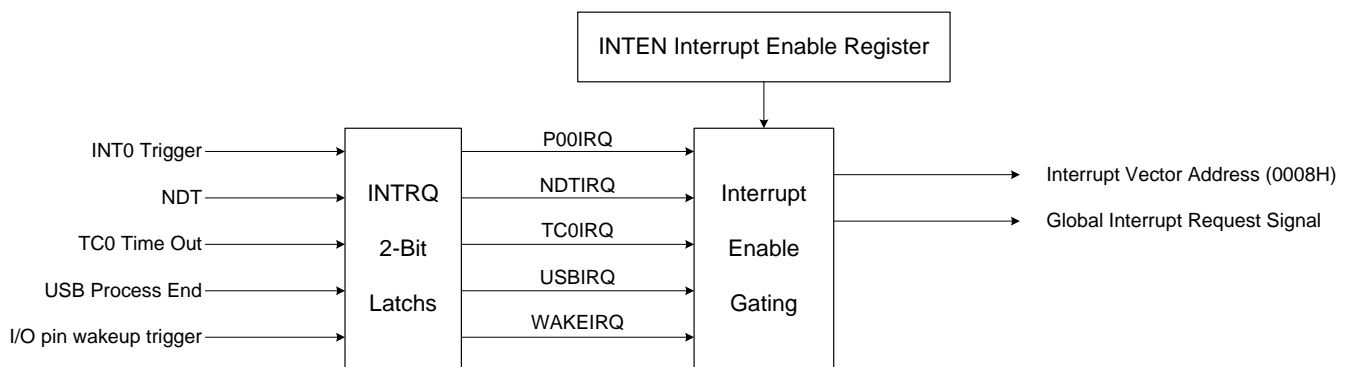
The wakeup time = $1/F_{osc} * 2048 = 0.341$ ms ($F_{osc} = 6\text{MHz}$)

The total wakeup time = 0.1705 ms + internal high RC oscillator start-up time

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 5 interrupt sources, including 4 internal interrupts (TC0/USB/WAKE/NDT) and one external interrupt (INT0). The external interrupt can wake up 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 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	NDTIEN	USBIEN	TC0IEN	-	-	WAKEIEN	-	P00IEN
Read/Write	R/W	R/W	R/W	-	-	R/W	-	R/W
After reset	0	0	0	-	-	0	-	0

- Bit 0 **P00IEN:** External P0.0 interrupt (INT0) control bit.
0 = Disable INT0 interrupt function.
1 = Enable INT0 interrupt function.
- Bit 2 **WAKEIEN:** I/O PORT0 & PORT 1 WAKEUP interrupt control bit.
0 = Disable WAKEUP interrupt function.
1 = Enable WAKEUP interrupt function.
- Bit 5 **TC0IEN:** TC0 timer interrupt control bit.
0 = Disable TC0 interrupt function.
1 = Enable TC0 interrupt function.
- Bit 6 **USBIEN:** USB interrupt control bit.
0 = Disable USB interrupt function.
1 = Enable USB interrupt function.
- Bit 7 **NDTIEN:** Noise Detector interrupt control bit.
0 = Disable Noise Detector interrupt function.
1 = Enable Noise Detector 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	NDTIRQ	USBIRQ	TC0IRQ	-	-	WAKEIRQ	-	P00IRQ
Read/Write	R/W	R/W	R/W	-	-	R/W	-	R/W
After reset	0	0	0	-	-	0	-	0

- Bit 0 **P00IRQ:** External P0.0 interrupt (INT0) request flag.
0 = None INT0 interrupt request.
1 = INT0 interrupt request.
- Bit 2 **WAKEIRQ:** I/O PORT0 & PORT1 WAKEUP interrupt request flag.
0 = None WAKEUP interrupt request.
1 = WAKEUP interrupt request.
- Bit 5 **TC0IRQ:** TC0 timer interrupt request flag.
0 = None TC0 interrupt request.
1 = TC0 interrupt request.

- Bit 6 **USBIRQ**: USB interrupt request flag.
0 = None USB interrupt request.
1 = USB interrupt request.
- Bit 7 **NDTIRQ**: Noise Detector interrupt request flag.
0 = None Noise Detector interrupt request.
1 = Noise Detector 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).

```
B0BSET     FGIE                   ; Enable GIE
```

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

6.5 PUSH, POP ROUTINE

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

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

➤ **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 INT0 (P0.0) INTERRUPT OPERATION

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

If the interrupt trigger direction is identical with wake-up trigger direction, the INT0 interrupt request flag (INT0IRQ) is latched while system wake-up from power down mode or green mode by P0.0 wake-up trigger. System inserts to interrupt vector (ORG 8) after wake-up immediately.

* **Note:** INT0 interrupt request can be latched by P0.0 wake-up trigger.

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

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

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

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

```

MOV      A, #03H
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.

```
INT_SERVICE:  ORG      8      ; Interrupt vector
               JMP      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 TC0 INTERRUPT OPERATION

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

Example: TC0 interrupt request setup.

```

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

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

B0BSET    FGIE       ; Enable GIE

```

Example: TC0 interrupt service routine.

```

ORG       8          ; Interrupt vector
JMP      INT_SERVICE

INT_SERVICE:

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

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

B0BCLR    FTC0IRQ    ; Reset TC0IRQ
MOV       A, #74H
B0MOV     TC0C, A     ; Reset TC0C.
...          ; TC0 interrupt service routine
...

EXIT_INT:

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

RETI       ; Exit interrupt vector

```

6.8 USB INTERRUPT OPERATION

When the USB process finished, the USBIRQ will be set to “1” no matter the USBIEN is enable or disable. If the USBIEN and the trigger event USBIRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the USBIEN = 0, the trigger event USBIRQ is still set to be “1”. Moreover, the system won't execute interrupt vector. Users need to be cautious with the operation under multi-interrupt situation.

➤ **Example: USB interrupt request setup.**

B0BCLR	FUSBIEN	; Disable USB interrupt service
B0BCLR	FUSBIRQ	; Clear USB interrupt request flag
B0BSET	FUSBIEN	; Enable USB interrupt service
...		; USB initializes.
...		; USB operation.
B0BSET	FGIE	; Enable GIE

Example: USB interrupt service routine.

	ORG	8	; Interrupt vector
	JMP	INT_SERVICE	
INT_SERVICE:			
	PUSH		; Push routine to save ACC and PFLAG to buffers.
	B0BTS1	FUSBIRQ	; Check USBIRQ
	JMP	EXIT_INT	; USBIRQ = 0, exit interrupt vector
	B0BCLR	FUSBIRQ	; Reset USBIRQ
	...		; USB interrupt service routine
	...		
EXIT_INT:			
	POP		; Pop routine to load ACC and PFLAG from buffers.
	RETI		; Exit interrupt vector

6.9 WAKEUP INTERRUPT OPERATION

When the I/O port 1 or I/O port 0 wakeup the MCU from the sleep mode, the WAKEIRQ will be set to “1” no matter the WAKEIEN is enable or disable. If the WAKEIEN and the trigger event WAKEIRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the WAKEIEN = 0, the trigger event WAKEIRQ is still set to be “1”. Moreover, the system won't execute interrupt vector. Users need to be cautious with the operation under multi-interrupt situation.

➤ **Example: WAKE interrupt request setup.**

B0BCLR	FWAKEIEN	; Disable WAKE interrupt service
B0BCLR	FWAKEIRQ	; Clear WAKE interrupt request flag
B0BSET	FWAKEIEN	; Enable WAKE interrupt service
...		; Pin WAKEUP initialize.
...		; Pin WAKEUP operation.
B0BSET	FGIE	; Enable GIE

Example: WAKE interrupt service routine.

	ORG	8	; Interrupt vector
	JMP	INT_SERVICE	
INT_SERVICE:			
	PUSH		; Push routine to save ACC and PFLAG to buffers.
	B0BTS1	FWAKEIRQ	; Check WAKEIRQ
	JMP	EXIT_INT	; WAKEIRQ = 0, exit interrupt vector
	B0BCLR	FWAKEIRQ	; Reset WAKEIRQ
	...		; WAKE interrupt service routine
	...		
EXIT_INT:			
	POP		; Pop routine to load ACC and PFLAG from buffers.
	RETI		; Exit interrupt vector

6.10 MULTI-INTERRUPT OPERATION

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

Interrupt Name	Trigger Event Description
P00IRQ	P0.0 trigger controlled by PEDGE
TC0IRQ	TC0 overflow
USBIRQ	USB process finished
NDTIRQ	Noise detected
WAEKIRQ	I/O port0 & port1 wakeup MCU

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

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

```

ORG      8                      ; Interrupt vector
JMP      INT_SERVICE

INT_SERVICE:

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

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

INTT0CHK:                      ; Check TC0 interrupt request
    B0BTS1    FTC0IEN           ; Check TC0IEN
    JMP      INTUSBCHK          ; Jump check to next interrupt
    B0BTS0    FTC0IRQ           ; Check TC0IRQ
    JMP      INTTC0            ; Jump to TC0 interrupt service routine

INTUSBCHK:                    ; Check USB interrupt request
    B0BTS1    FUSBIEN           ; Check USBIEN
    JMP      INTNDTCHK          ; Jump check to next interrupt
    B0BTS0    FUSBIRQ           ; Check USBIRQ
    JMP      INTUSB            ; Jump to USB interrupt service routine

INTNDTCHK:                    ; Check NDT interrupt request
    B0BTS1    FNDTIEN           ; Jump check to next interrupt
    JMP      INTWAKECHK         ; Check NDTIRQ
    B0BTS0    FNDTIRQ           ; Jump to NDT interrupt service routine
    JMP      INTNDT

INTWAKECHK:                    ; Check USB interrupt request
    B0BTS1    FWAKEIEN          ; Check WAKEIEN
    JMP      INT_EXIT           ; Jump check to next interrupt
    B0BTS0    FWAKEIRQ          ; Check WAKEIRQ
    JMP      INTWAKEUP          ; Jump to WAKEUP interrupt service routine

INT_EXIT:

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

    RETI                        ; Exit interrupt vector

```

7 I/O PORT

7.1 I/O PORT MODE

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

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
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

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

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

0C3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3M	-	P36M	P35M	P34M	P33M	P32M	P31M	P30M
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

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	P55M	P54M	P53M	P52M	P51M	P50M
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 = 0~5).
 0 = Pn is input mode.
 1 = Pn is output mode.

*** Note:**

- Users can program them by bit control instructions (**B0BSET**, **B0BCLR**).

Example: I/O mode selecting

```

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

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

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

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

7.2 I/O PULL UP REGISTER

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0E1H	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

0E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0E3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3UR	-	P36R	P35R	P34R	P33R	P32R	P31R	P30R
Read/Write	-	W	W	W	W	W	W	W
After reset	-	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	-	-	P55R	P54R	P53R	P52R	P51R	P50R
Read/Write	-	-	W	W	W	W	W	W
After reset	-	-	0	0	0	0	0	0

Example: I/O Pull up Register

```

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

```


7.3 I/O PORT DATA REGISTER

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

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

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

0D3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3	P37	P36	P35	P34	P33	P32	P31	P30
Read/Write	R	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	-	-	P55	P54	P53	P52	P51	P50
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

* **Note:** The P1.6 keeps “1” when external reset enable by code option.

Example: Read data from input port.

```

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

```

Example: Write data to output port.

```

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

```

Example: Write one bit data to output port.

```

B0BSET     P1.3           ; Set P1.3 and P5.3 to be “1”.
B0BSET     P5.3

B0BCLR     P1.3           ; Set P1.3 and P5.3 to be “0”.
B0BCLR     P5.3

```

8 TIMERS

8.1 WATCHDOG TIMER

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

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

VDD	Internal Low RC Freq.	Watchdog Overflow Time
5V	32KHz	341ms

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

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

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

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

Main:

```

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

```

Watchdog timer application note is as following.

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

Main:

... ; Check I/O.

... ; Check RAM

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

Correct: ; I/O and RAM are correct. Clear watchdog timer and
; execute program.

```
MOV      A,#5AH
B0MOV    WDTR,A
CALL     SUB1
CALL     SUB2
...
...
...
JMP      MAIN
```

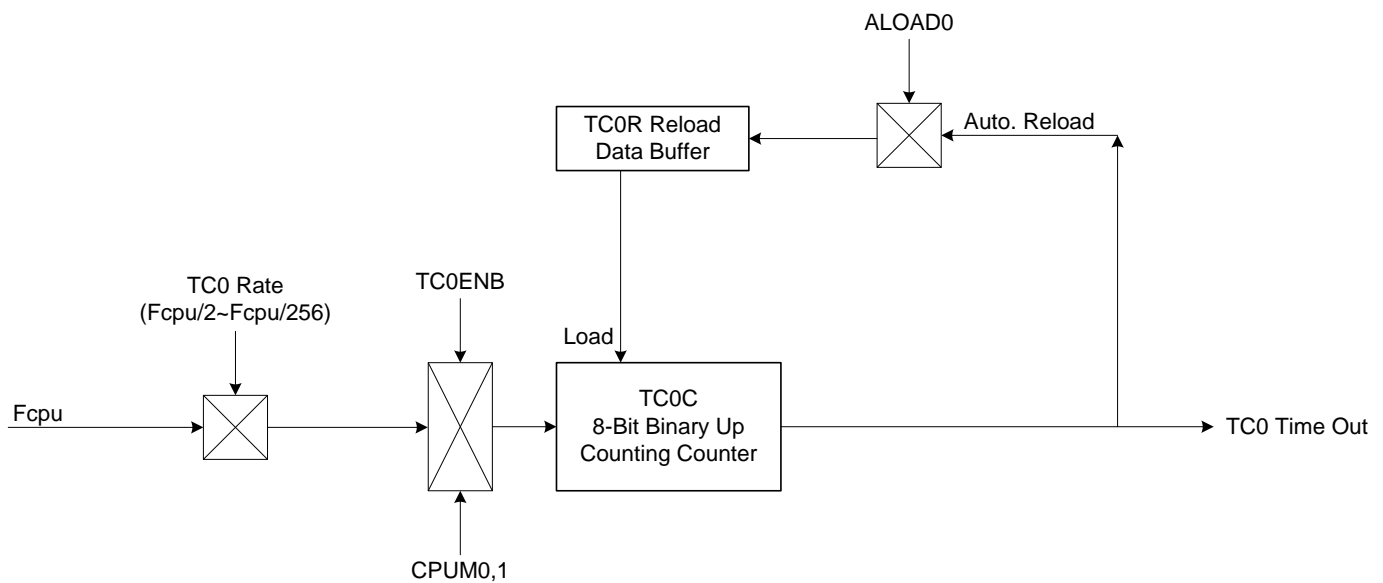
8.2 TIMER/COUNTER 0 (TC0)

8.2.1 OVERVIEW

The TC0 is an 8-bit binary up counting timer with double buffers. TC0 uses internal clock for counting a precision time. The internal clock source is from Fcpu. If TC0 timer overflow occurs, it will continue counting and issue a time-out signal to trigger TC0 interrupt to request interrupt service. TC0 overflow time is 0xFF to 0x00 normally.

The main purpose of the TC0 timer is as following.

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



8.2.2 TC0M MODE REGISTER

088H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	-	ALOAD0	-	-
Read/Write	R/W	R/W	R/W	R/W	-	R/W	-	-
After reset	0	0	0	0	-	0	-	-

- Bit 2 **ALOAD0:** Auto-reload control bit.
0 = Disable TC0 auto-reload function.
1 = Enable TC0 auto-reload function.
- Bit [6:4] **TC0RATE[2:0]:** TC0 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
...
110 = fcpu/4.
111 = fcpu/2.
- Bit 7 **TC0ENB:** TC0 counter control bit.
0 = Disable TC0 timer.
1 = Enable TC0 timer.

8.2.3 TC0C COUNTING REGISTER

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

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0C	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of TC0C initial value is as following.

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

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

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

Example: To set 1ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=0). High clock is internal 6MHz. Fcpu=Fosc/1. Select TC0RATE=010 (Fcpu/64).

$$\begin{aligned}
 TC0C \text{ initial value} &= N - (TC0 \text{ interrupt interval time} * \text{input clock}) \\
 &= 256 - (1\text{ms} * 6\text{MHz} / 1 / 64) \\
 &= 256 - (10^{-3} * 6 * 10^6 / 1 / 64) \\
 &= 162 \\
 &= A2H
 \end{aligned}$$

The basic timer table interval time of TC0.

TC0RATE	TC0CLOCK	High speed mode (Fcpu = 6MHz / 1)	
		Max overflow interval	One step = max/256
000	Fcpu/256	10.923 ms	42.67 us
001	Fcpu/128	5.461 ms	21.33 us
010	Fcpu/64	2.731 ms	10.67 us
011	Fcpu/32	1.365 ms	5.33 us
100	Fcpu/16	0.683 ms	2.67 us
101	Fcpu/8	0.341 ms	1.33 us
110	Fcpu/4	0.171 ms	0.67 us
111	Fcpu/2	0.085 ms	0.33 us

8.2.4 TC0R AUTO-LOAD REGISTER

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

TC0 is double buffer design. If new TC0R value is set by program, the new value is stored in 1st buffer. Not until TC0 overflow occurs, the new value moves to real TC0R buffer.

08AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of TC0R initial value is as following.

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

N is TC0 overflow boundary number.

Example: To set 1ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=0). High clock is internal 6MHz. Fcpu=Fosc/1. Select TC0RATE=010 (Fcpu/64).

$$\begin{aligned}
 TC0R \text{ initial value} &= N - (TC0 \text{ interrupt interval time} * \text{input clock}) \\
 &= 256 - (1ms * 6MHz / 1 / 64) \\
 &= 256 - (10^{-3} * 6 * 10^6 / 1 / 64) \\
 &= 162 \\
 &= A2H
 \end{aligned}$$

8.2.5 TC0 TIMER OPERATION SEQUENCE

The sequence of setup TC0 timer is as following.

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

B0BCLR	FTC0ENB	; TC0 timer stop.
B0BCLR	FTC0IEN	; TC0 interrupt function is disabled.
B0BCLR	FTC0IRQ	; TC0 interrupt request flag is cleared.

☞ **Set TC0 timer rate.**

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

☞ **Set TC0 timer auto-load mode.**

B0BCLR	FALOAD0	; Enable TC0 auto reload function.
or		
B0BSET	FALOAD0	; Disable TC0 auto reload function.

☞ **Enable TC0 timer.**

B0BSET	FTC0ENB	; Enable TC0 timer.
--------	---------	---------------------

9 UNIVERSAL SERIAL BUS (USB)

9.1 OVERVIEW

The USB is the answer to connectivity for the PC architecture. A fast, bi-directional interrupt pipe, low-cost, dynamically attachable serial interface is consistent with the requirements of the PC platform of today and tomorrow. The SONiX USB microcontrollers are optimized for human-interface computer peripherals such as a mouse, joystick, game pad.

USB Specification Compliance

- Conforms to USB specifications, Version 2.0.
- Supports 1 Low-speed USB device address.
- Supports 1 control endpoint, 3 interrupt endpoints.
- Integrated USB transceiver.
- 5V to 3.3V regulator output for D- 1.5K ohm internal resistor pull up.

9.2 USB MACHINE

The USB machine allows the microcontroller to communicate with the USB host. The hardware handles the following USB bus activity independently of the microcontroller.

The USB machine will do:

- Translate the encoded received data and format the data to be transmitted on the bus.
- CRC checking and generation by hardware. If CRC is not correct, hardware will not send any response to USB host.
- Send and update the data toggle bit (Data1/0) automatically by hardware.
- Send appropriate ACK/NAK/STALL handshakes.
- SETUP, IN, or OUT Token type identification. Set the appropriate bit once a valid token is received.
- Place valid received data in the appropriate endpoint FIFOs.
- Bit stuffing/unstuffing.
- Address checking. Ignore the transactions not addressed to the device.
- Endpoint checking. Check the endpoint's request from USB host, and set the appropriate bit of registers.

Firmware is required to handle the rest of the following tasks:

- Coordinate enumeration by decoding USB device requests.
- Fill and empty the FIFOs.
- Suspend/Resume coordination.
- Remote wake up function.
- Determine the right interrupt request of USB communication.

9.3 USB INTERRUPT

The USB function will accept the USB host command and generate the relative interrupts, and the program counter will go to 0x08 vector. Firmware is required to check the USB status bit to realize what request comes from the USB host.

The USB function interrupt is generated when:

- The endpoint 0 is set to accept a SETUP token.
- The device receives an ACK handshake after a successful read transaction (IN) from the host.
- If the endpoint is in ACK OUT modes, an interrupt is generated when data is received.
- The USB host send USB suspend request to the device.
- USB bus reset event occurs.
- The USB endpoints interrupt after a USB transaction complete is on the bus.
- The NAK handshaking when the NAK interrupt enable.

The following examples show how to avoid the error of reading or writing the endpoint FIFOs and to do the right USB request routine according to the flag.

9.4 USB ENUMERATION

A typical USB enumeration sequence is shown below.

1. The host computer sends a SETUP packet followed by a DATA packet to USB address 0 requesting the Device descriptor.
2. Firmware decodes the request and retrieves its Device descriptor from the program memory tables.
3. The host computer performs a control read sequence and Firmware responds by sending the Device descriptor over the USB bus, via the on-chip FIFO.
4. After receiving the descriptor, the host sends a SETUP packet followed by a DATA packet to address 0 assigning a new USB address to the device.
5. Firmware stores the new address in its USB Device Address Register after the no-data control sequence completes.
6. The host sends a request for the Device descriptor using the new USB address.
7. Firmware decodes the request and retrieves the Device descriptor from program memory tables.
8. The host performs a control read sequence and Firmware responds by sending its Device descriptor over the USB bus.
9. The host generates control reads from the device to request the Configuration and Report descriptors.
10. Once the device receives a Set Configuration request, its functions may now be used.
11. Firmware should take appropriate action for Endpoint 0~2 transactions, which may occur from this point.

9.5 USB REGISTERS

9.5.1 USB DEVICE ADDRESS REGISTER

The USB Device Address Register (UDA) contains a 7-bit USB device address and one bit to enable the USB function. This register is cleared during a reset, setting the USB device address to zero and disable the USB function.

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UDA	UDE	UDA6	UDA5	UDA4	UDA3	UDA2	UDA1	UDA0
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 [6:0] **UDA [6:0]:** These bits must be set by firmware during the USB enumeration process (i.e., SetAddress) to the non-zero address assigned by the USB host.

Bit 7 **UDE: Device Function Enable.** This bit must be enabled by firmware to enable the USB device function.
0 = Disable USB device function.
1 = Enable USB device function.

9.5.2 USB STATUS REGISTER

The USB status register indicates the status of USB.

091H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USTATUS	-	-	-	BUS_RST	SUSPEND	EP0_SETUP	EP0_IN	EP0_OUT
Read/Write	-	-	-	R	R	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

Bit 0 **EP0_OUT:** Endpoint 0 OUT Token Received.
0 = Endpoint 0 has no OUT token received.
1 = A valid OUT packet has been received. The bit is set to 1 after the last received packet in an OUT transaction.

Bit 1 **EP0_IN:** Endpoint 0 IN Token Received.
0 = Endpoint 0 has no IN token received.
1 = A valid IN packet has been received. The bit is set to 1 after the last received packet in an IN transaction.

Bit 2 **EP0_SETUP:** Endpoint 0 SETUP Token Received.
0 = Endpoint 0 has no SETUP token received.
1 = A valid SETUP packet has been received. The bit is set to 1 after the last received packet in an SETUP transaction. While the bit is set to 1, the HOST cannot write any data in to EP0 FIFO. This prevents SIE from overwriting an incoming SETUP transaction before firmware has a chance to read the SETUP data

Bit 3 **SUSPEND:** indicate USB suspend status.
0 = Non-suspend status. When MCU wakeup from sleep mode by USB resume wakeup request, the bit will changes from 1 to 0 automatically.
1 = Set to 1 by hardware when USB suspend request.

Bit 4 **BUS_RST:** USB bus reset.
0 = Non-USB bus reset.
1 = Set to 1 by hardware when USB bus reset request.

9.5.3 USB DATA COUNT REGISTER

The USB EP0 OUT token data byte counter.

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP0OUT_CNT	-	-	-	UEP0OC4	UEP0OC3	UEP0OC2	UEP0OC1	UEP0OC0

Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

Bit [4:0] **UEP0C [4:0]**: USB endpoint 0 OUT token data counter.

9.5.4 USB ENABLE CONTROL REGISTER

The register control the regulator output 3.3 volts enable, SOF packet receive interrupt, NAK handshaking interrupt and D- internal 1.5k ohm pull up.

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USB_INT_EN	REG_EN	DN_UP_EN	-	EP2_NAK_EN	EP1_NAK_EN	-	UE2D	UE1D
Read/Write	R/W	R/W	-	R/W	R/W	-	R/W	R/W
After reset	1	0	-	0	0	-	0	0

- Bit 0 **UE1D**: The IN/OUT direction enable bit.
0 = EP1 only handshakes with IN token.
1 = EP1 only handshakes with OUT token.
- Bit 1 **UE2D**: The IN/OUT direction enable bit.
0 = EP2 only handshakes with IN token.
1 = EP2 only handshakes with OUT token.
- Bit 2 **EP1_NAK_EN**: EP1 NAK transaction interrupt enable bit.
0 = Disable EP1_NAK interrupt & flag
1 = Enable EP1_NAK interrupt & flag
- Bit 3 **EP2_NAK_EN**: EP2 NAK transaction interrupt enable bit.
0 = Disable EP2_NAK interrupt & flag
1 = Enable EP2_NAK interrupt & flag
- Bit 6 **DN_UP_EN**: D- internal 1.5k ohm pull up resistor control bit.
0 = Disable D- pull up 1.5k ohm to 3.3volts.
1 = Enable D- pull up 1.5k ohm to 3.3volts.
- Bit 7 **REG_EN**: 3.3volts Regulator control bit.
0 = Disable regulator output 3.3volts.
1 = Enable regulator output 3.3volts. This bit must enable when using USB function.

9.5.5 USB endpoint's ACK handshaking flag REGISTER

The status of endpoint's ACK transaction.

094H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP_ACK	-	-	-	-	EP2_NAK	EP1_NAK	EP2_ACK	EP1_ACK
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

- Bit [1:0] **EPn_ACK [1:0]**: EP1~EP2 ACK transaction. n= 1, 2. The bit is set whenever the endpoint that completes with an ACK received.
0 = The endpoint (interrupt pipe) doesn't complete with an ACK.
1 = The endpoint (interrupt pipe) completes with an ACK.
- Bit 2 **EP1_NAK**: Endpoint 1's NAK Transaction. The bit is set whenever the endpoint 1 that completes with an NAK packet received
0 = The endpoint 1's transaction doesn't complete with an NAK.
1 = The endpoint 1's transaction completes with an NAK.
- Bit 3 **EP2_NAK**: Endpoint 2's NAK Transaction. The bit is set whenever the endpoint 2 that completes with an

NAK packet received

0 = The endpoint 2's transaction doesn't complete with an NAK.

1 = The endpoint 2's transaction complete with an NAK.

9.5.6 USB ENDPOINT 0 ENABLE REGISTER

An endpoint 0 (EP0) is used to initialize and control the USB device. EP0 is bi-directional (Control pipe), as the device, can both receive and transmit data, which provides to access the device configuration information and allows generic USB status and control accesses.

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE0R	UE0E	UE0M1	UE0M0	-	UE0C3	UE0C	UE0C1	UE0C0
Read/Write	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
After reset	0	0	0	-	0	0	0	0

Bit [3:0] **UE0C [3:0]**: Indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint 0 FIFO.

Bit [6:5] **UE0M [1:0]**: The endpoint 0 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 0. For example, if the endpoint 0's mode bit is set to 00 that is NAK IN/OUT mode as shown in Table, The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 0. The bit 5 UE0M0 will auto reset to zero when the ACK transaction complete.

USB endpoint 0's mode table

UE0M1	UE0M0	IN/OUT Token Handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

Bit 7 **UE0E**: EP0 Enable bit.
0 = EP0 enable.
1 = EP0 disable (No handshake for any EP0 USB packet).

9.5.7 USB ENDPOINT 1 ENABLE REGISTER

The communication with the USB host using endpoint 1, endpoint 1's FIFO is implemented as 8 bytes of dedicated RAM. The endpoint1 is an interrupt endpoint.

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE1R	UE1E	UE1M1	UE1M0	-	UE1C3	UE1C2	UE1C1	UE1C0
Read/Write	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
After reset	0	0	0	-	0	0	0	0

Bit [3:0] **UE1C [3:0]**: Indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint 1 FIFO.

Bit [6:5] **UE1M [1:0]**: The endpoint 1 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 1. For example, if the endpoint 1's mode bit is set to 00 that is NAK IN/OUT mode as shown in Table, The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 1. The bit 5 UE1M0 will auto reset to zero when the ACK transaction complete.

USB endpoint 1's mode table

UE1M1	UE1M0	IN/OUT Token Handshake
-------	-------	------------------------

0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

Bit 7 **UE1E:** USB endpoint 1 function enable bit.
0 = Disable USB endpoint 1 function.
1 = Enable USB endpoint 1 function.

9.5.8 USB ENDPOINT 2 ENABLE REGISTER

The communication with the USB host using endpoint 2, endpoint 2's FIFO is implemented as 8 bytes of dedicated RAM. The endpoint 2 is an interrupt endpoint.

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE2R	UE2E	UE2M1	UE2M0	-	UE2C3	UE2C2	UE2C1	UE2C0
Read/Write	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
After reset	0	0	0	-	0	0	0	0

Bit [3:0] **UE2C [3:0]:** Indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint 2 FIFO.

Bit [6:5] **UE2M [1:0]:** The endpoint 2 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 2. For example, if the endpoint 2's mode bit is set to 00 that is NAK IN/OUT mode as shown in Table, The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 2. The bit 5 UE2M0 will auto reset to zero when the ACK transaction complete.

USB endpoint 2's mode table

UE2M1	UE2M0	IN/OUT Token Handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

Bit 7 **UE2E:** USB endpoint 2 function enable bit.
0 = Disable USB endpoint 2 function.
1 = Enable USB endpoint 2 function.

9.5.9 UPID REGISTER

Forcing bits allow firmware to directly drive the D+ and D- pins.

0ABH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UPID	EP0OUT_EN	-	-	-	-	UBDE	DDP	DDN
Read/Write	R/W	-	-	-	-	R/W	R/W	R/W
After reset	0	-	-	-	-	0	0	0

Bit 0 **DDN:** Drive D- on the USB bus.
0 = Drive D- low.
1 = Drive D- high.

Bit 1 **DDP:** drive D+ on the USB bus.
0 = Drive D+ low.
1 = Drive D+ high.

- Bit 2 **UBDE:** Enable to directly drive USB bus.
0 = Disable.
1 = Enable.
- Bit 7 **EP0OUT_EN:** Enable SIE to received EP0 OUT over 8 bytes
0 = Disable.
1 = Enable.

9.5.10 ENDPOINT TOGGLE BIT CONTROL REGISTER

0ACH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UTOGGLE	-	-	-	-	-	-	EP2 _DATA0/1	EP1 _DATA0/1
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	1	1

- Bit [1:0] Endpoint 1~2's DATA0/1 toggle bit control.
0 = Clear the endpoint 1~2's toggle bit to DATA0.
1 = hardware set toggle bit automatically.

9.5.11 NDT BIT CONTROL REGISTER

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UNDT	-	-	-	-	-	-	-	NDT
Read/Write	-	-	-	-	-	-	-	R/W
After reset	-	-	-	-	-	-	-	0

- Bit 0 **NDT:** Indicates the power noise is detected.
0 = Cleared by firmware.
1 = Power noise is detected. Set by H/W automatically.

9.5.12 ENDPOINT CONTROL REGISTER

0B0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP0_STALL	-	-	-	-	-	-	EP0_OUT_ STALL	EP0 _IN_STALL
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

- Bit 0 **EP0_IN_STALL:** The IN STALL function enable bit.
0 = Disable EP0 IN STALL function.
1 = Enable EP0 IN STALL function. If this function is enable, EP0 IN token always handshakes STALL. The EP0 OUT token handshakes depend on UE0R. This flag will clear at next SETUP token.
- Bit 1 **EP0_OUT_STALL:** The OUT STALL function enable bit.
0 = Disable EP0 OUT STALL function.
1 = Enable EP0 OUT STALL function. If this function is enable, EP0 OUT token always handshakes STALL. The EP0 IN token handshakes depend on UE0R. This flag will clear at next SETUP token

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
L O G I C	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 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
P R O C E S S	XOR M,A	$M \leftarrow A$ xor M	-	-	√	1+N
	XOR A,I	$A \leftarrow A$ xor I	-	-	√	1
	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
B R A N C H	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
	CMPS A,I	$ZF, C \leftarrow A - I$, If $A = I$, then skip next instruction	√	-	√	1 + S
C O N D I T I O N A L	CMPS 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
	RET	$PC \leftarrow Stack$	-	-	-	2
	RETI	$PC \leftarrow Stack$, and to enable global interrupt	-	-	-	2
M I S C	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.

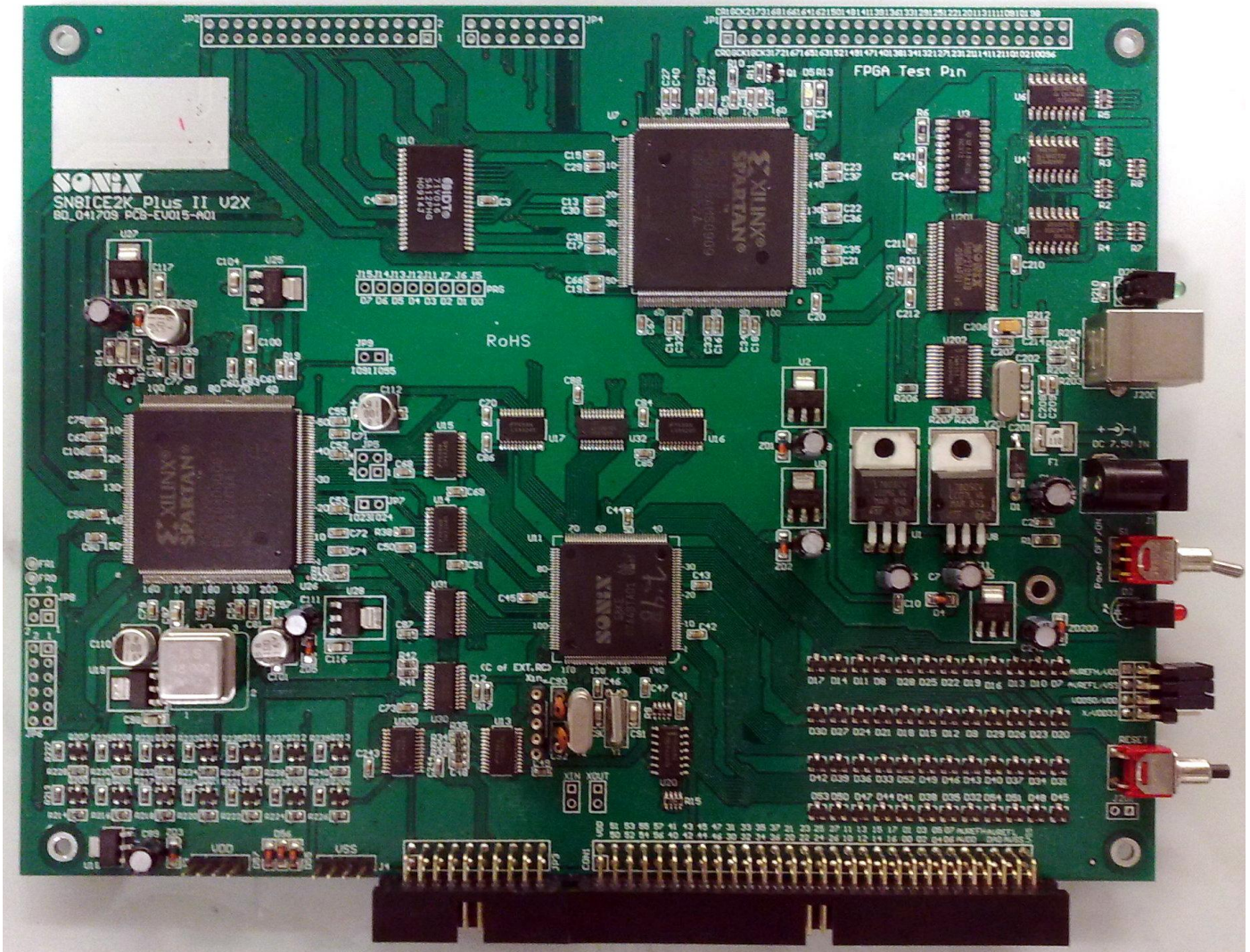
2. If branch condition is true then "S = 1", otherwise "S = 0".

11 DEVELOPMENT TOOL

SONiX provides ICE (in circuit emulation), IDE (Integrated Development Environment), EV-kit and firmware library for USB application development. ICE and EV-kit are external hardware device and IDE is a friendly user interface for firmware development and emulation.

11.1 ICE (In Circuit Emulation)

The ICE called "SN8ICE2K Plus II"



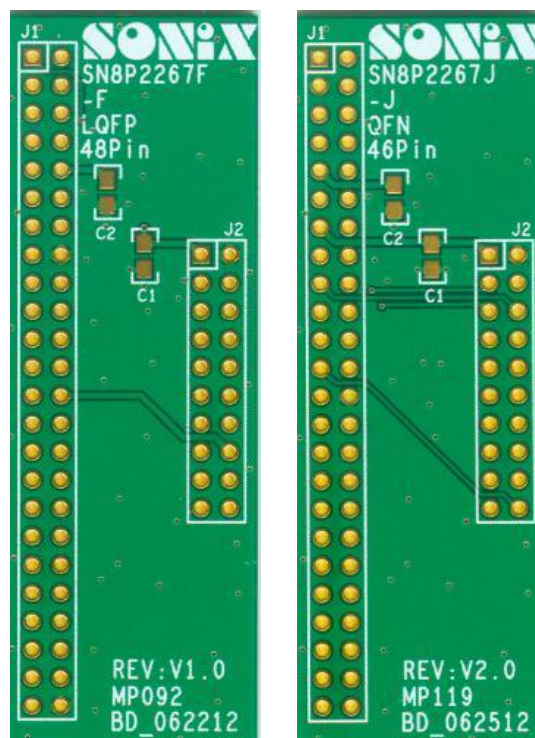
11.2 SN8P2267C EV-kit

The EV-kit for SN8P2267C is equal to that for SN8P2267. The EV-kit includes ICE interface, GPIO interface, USB interface, and VREG 3.3V power supply.

The outline of SN8P2267C EV-kit is as following.



11.3 SN8P2267C Transition Board



Note: C1 = C2 = 1uF

12 ELECTRICAL CHARACTERISTIC

12.1 ABSOLUTE MAXIMUM RATING

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

12.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, fosc = 6MHz, ambient temperature is 25°C unless otherwise note.)

(All of voltages refer to Vss, Vdd = 5.0V, Iosc = 0mHz, ambient temperature is 25 °C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd1	Normal mode except USB transmitter specifications, Vpp = Vdd		4.1	5	5.5	V
RAM Data Retention voltage	Vdr			-	1.5*	-	V
Vdd rise rate	Vpor	Vdd rise rate to ensure 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		-	-	5	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 5V		250	360	550	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O P0/P1/P2/P5/P3.0~P3.6, output source current	IoH	Vop = Vdd – 0.5V		2	4	6	mA
I/O P0/P1/P2/P5/P3.0~P3.6, output source current (connect LED)	IoH	Vop = 2V		10	12	14	
I/O P0/P1/P2/P5/P3.0~P3.6, output sink current	IoL	Vop = Vss + 0.5V		2	4	6	
I/O P0/P1/P2/P5/P3.0~P3.6, output sink current (connect LED)	IoL	Vop = 2.9V		10	12	14	
INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Regulator output voltage	Vreg	Regulator output voltage, Vin = Vdd		3.0		3.6	V
Regulator GND current	IvREGn1	No loading. Vreg pin output 3.3V ((Regulator enable)			80	100	uA
Supply Current	Idd1	normal Mode (No loading, Fcpu = Fosc/1)	Vdd= 5V, 6Mhz	-	2.5	5	mA
	Idd2	Slow Mode (Internal low RC)	Vdd= 5V, 32Khz	-	100	150	uA
	Idd3	Sleep Mode	Vdd= 5V	-	100	150	uA
	Idd4	Green Mode (No loading, Fcpu = Fosc/4 Watchdog Disable)	Vdd= 5V,6Mhz	-	1	2	mA
			Vdd=5V, ILRC 32Khz	-	100	150	uA
LVD36 detect level	VLVD2	Internal 3.6V detect level		3.2	3.6	4.0	V
LVD Voltage	Vdet0	Low voltage reset level.		2.0	2.2	2.4	V

* These parameters are for design reference, not tested.

13

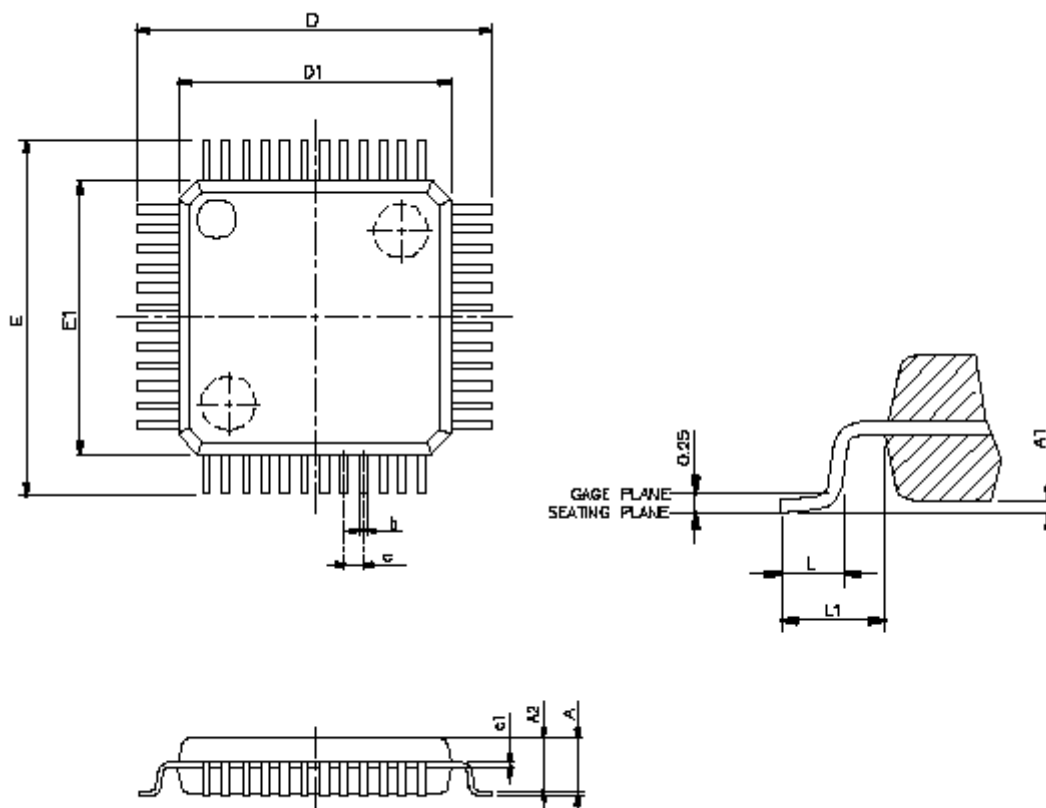
OTP ROM PROGRAMMING PIN

Programming Information of SN8P2267C Series											
Chip Name		SN8P2267CF		SN8P2267CJ							
EZ Writer / MP Writer Connector		OTP IC / JP3 Pin Assignment									
Number	Name	Number	Pin	Number	Pin	Number	Pin	Number	Pin	Number	Pin
1	VDD	45	VDD	6	VDD						
2	GND	41	VSS	1	VSS						
		25	P0.0	35	P0.0						
3	CLK	48	P3.4	10	P3.4						
4	CE										
5	PGM	2	P3.6	12	P3.6						
6	OE	46	P3.3	8	P3.3						
7	D1										
8	D0										
9	D3										
10	D2										
11	D5										
12	D4										
13	D7										
14	D6										
15	VDD										
16	VPP	36	P3.7	46	P3.7						
17	HLS										
18	RST										
19	-										
20	ALSB/PDB	1	P3.5	11	P3.5						

Note: Please also check the chapter 11.3 about the description of the SN8P2267C transition boards.

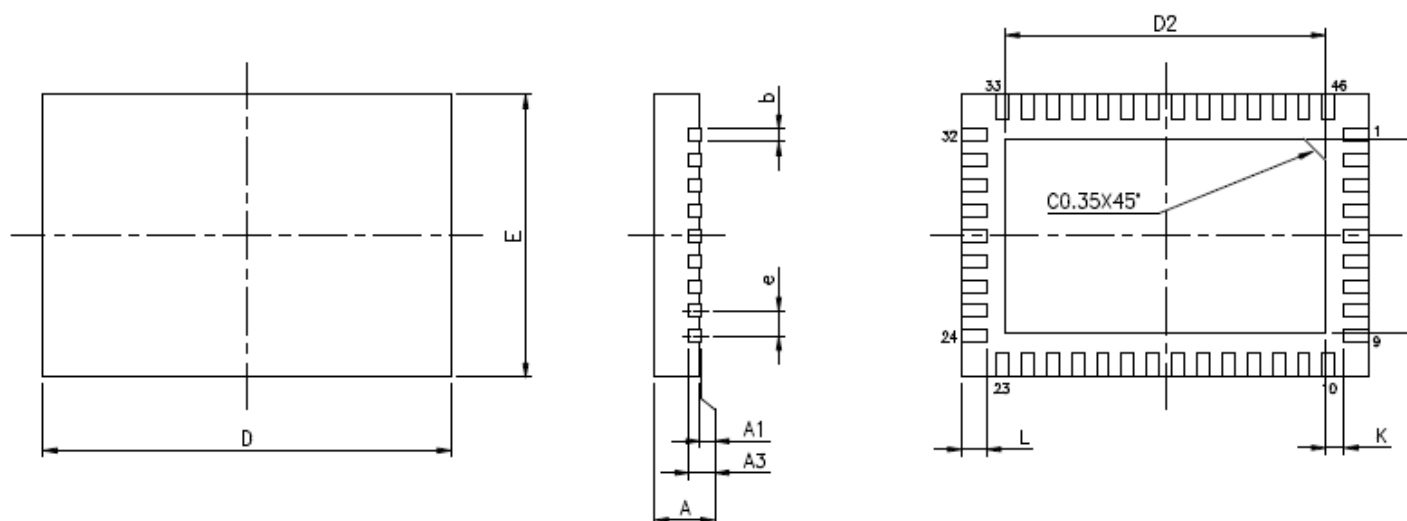
14 PACKAGE INFORMATION

14.1 LQFP 48 PIN



SYMBOLS	MIN	NOR	MAX
	(mm)		
A	-	-	1.6
A1	0.05	-	0.15
A2	1.35	-	1.45
c1	0.09	-	0.16
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
e	0.5 BSC		
B	0.17	-	0.27
L	0.45	-	0.75
L1	1 REF		

14.2 QFN 46 PIN



SYMBOLS	MIN.	NOM.	MAX.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	0.203 REF.		
b	0.18	0.25	0.30
D	6.40	6.50	6.60
E	4.40	4.50	4.60
e	0.40 BSC.		
D2	5.00	5.10	5.20
E2	3.00	3.10	3.20
L	0.35	0.40	0.45
K	0.20	—	—

UNIT : mm

NOTES :

1. JEDEC OUTLINE : N/A.
2. DIMENSION b APPLIES TO METALLIZED TERMINAL AND IS MEASURED BETWEEN 0.15mm AND 0.30mm FROM THE TERMINAL TIP. IF THE TERMINAL HAS THE OPTIONAL RADIUS ON THE OTHER END OF THE TERMINAL, THE DIMENSION b SHOULD NOT BE MEASURED IN THAT RADIUS AREA.
3. THE MINIMUM "K" VALUE OF 0.20mm APPLIES.
4. BILATERAL COPLANARITY ZONE APPLIES TO THE EXPOSED HEAT SINK SLUG AS WELL AS THE TERMINALS.

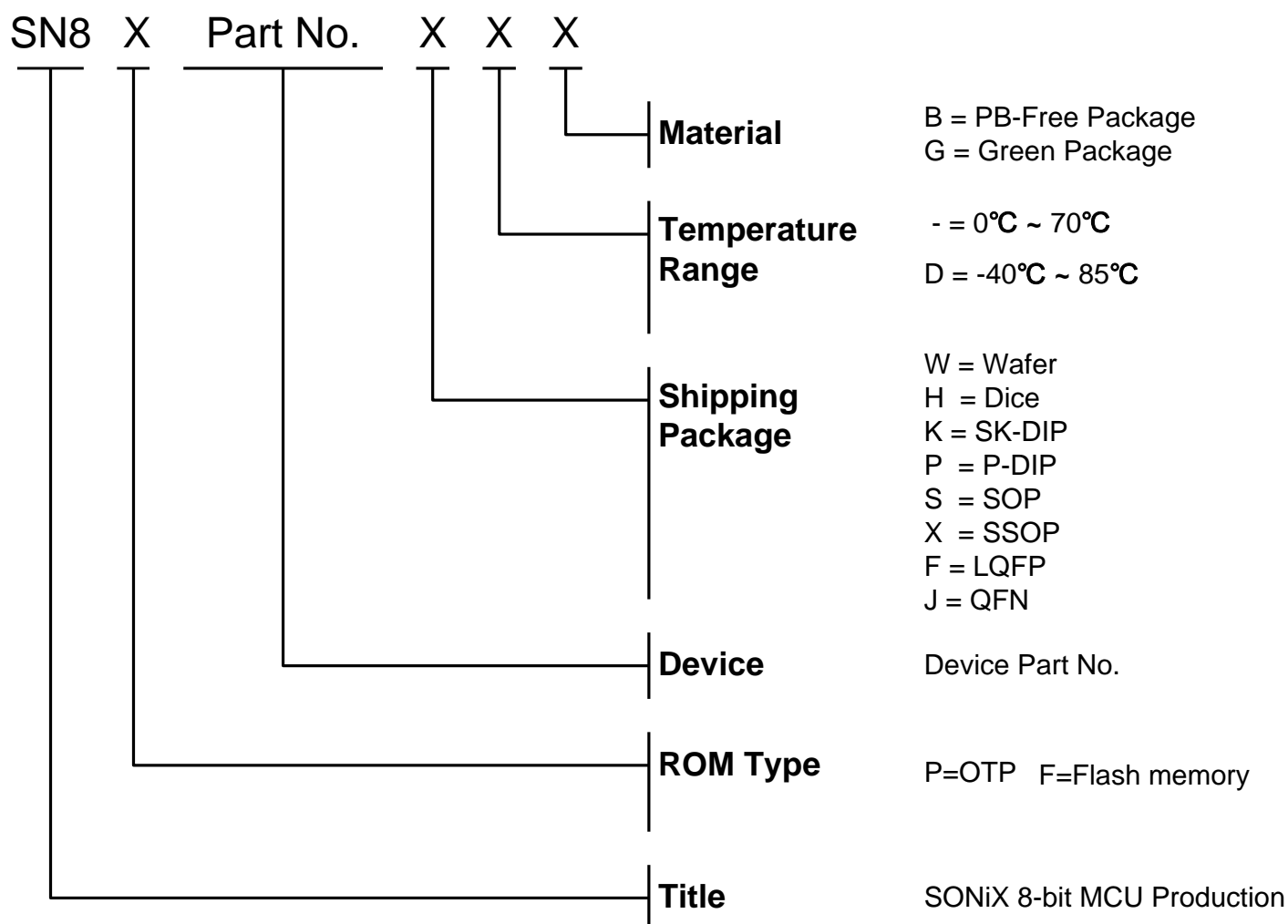
15

Marking Definition

15.1 INTRODUCTION

There are many different types in Sonix 8-bit MCU production line. This note listed the production definition of all 8-bit MCU for order or obtain information.

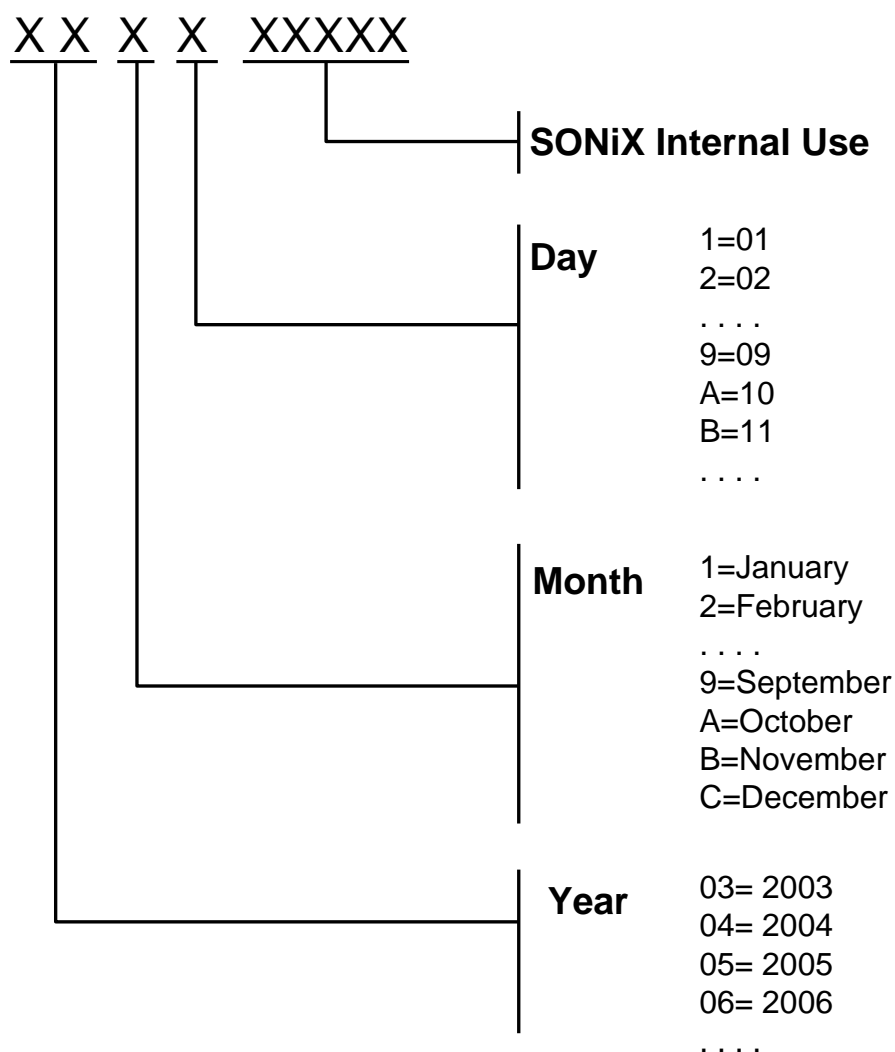
15.2 MARKING INDETIFICATION SYSTEM



15.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P2267CPB	OTP memory	2267C	SK-DIP	0°C ~70°C	PB-Free Package
SN8P2267CSB	OTP memory	2267C	SOP	0°C ~70°C	PB-Free Package
SN8P2267CXB	OTP memory	2267C	SSOP	0°C ~70°C	PB-Free Package
SN8P2267CPG	OTP memory	2267C	P-DIP	0°C ~70°C	Green Package
SN8P2267CSG	OTP memory	2267C	SOP	0°C ~70°C	Green Package
SN8P2267CXG	OTP memory	2267C	SSOP	0°C ~70°C	Green Package
SN8P2260CW	OTP memory	2260C	Wafer	0°C ~70°C	-
SN8P2260CH	OTP memory	2260C	Dice	0°C ~70°C	-

15.4 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 Stree., Chupei City, Hsinchu, Taiwan R.O.C.
Tel: 886-3-5600 888
Fax: 886-3-5600 889

Taipei Office:

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

Hong Kong Office:

Unit No.705,Level 7 Tower 1,Grand Central Plaza 138 Shatin Rural Committee Road,Shatin,New Territories,Hong Kong.
Tel: 852-2723-8086
Fax: 852-2723-9179

Technical Support by Email:

Sn8fae@sonix.com.tw