SN8F2270B Series

USER'S MANUAL

SN8F2271B
SN8F22711B
SN8F22721B

SONiX 8-Bit Micro-Controller

AMENDMENT HISTORY

Version	Date	Description
VER1.0	2009/3/23	version 1.0
VER1.1	2009/6/17	Modify SN8F22721S/X/K to SN8F22721S/X/P
VER1.2	2009/7/9	Modify PWM output pin to p5.3.

Table of Content

1 PRODUCT OVERVIEW

1.1 FEATURES

- **Memory configuration**
  Flash ROM size: 5K x 16 bits, including in system programming function and EEPROM enumeration. 20000 erase/write cycles.
  RAM size: 192 x 8 bits.

- **8 levels stack buffer**

- **I/O pin configuration**
  Bi-directional: P0, P1, P5
  Wake-up: P0/P1 level change.
  Pull-up resistors: P0, P1, P5.
  External interrupt: P0.0, P0.1 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, 1 control endpoint
  2 interrupt IN/OUT endpoints, each has 8 bytes FIFO

- **Powerful instructions**
  One clocks per instruction cycle (1T)
  Most of instructions are one cycle only.
  All ROM area JMP instruction.
  All ROM area CALL address instruction.
  All ROM area lookup table function (MOVC)

- **7 interrupt sources.**
  Five internal interrupts: T0, TC0, USB, SPI, Wakeup
  Two external interrupt: INT0, INT1.

- **One SIO function for data transfer**

- **Two 8 bits timer counter (T0, TC0)**
  TC0 has 8 bit PWM function (duty/cycle programmable).

- **Two system clocks.**
  Internal low clock: RC type 24KHz which Fosc = 24KHz.
  Internal High clock: Fosc = 6MHz.

- **Four operating modes.**
  Normal mode: Both high and low clocks active.
  Slow mode: Low clock only.
  Sleep: Both high and low clocks stop.
  Green mode: Periodical wakeup by timer.

- **Package**
  SOP14, QFN16, SOP20, SSOP20, DIP20

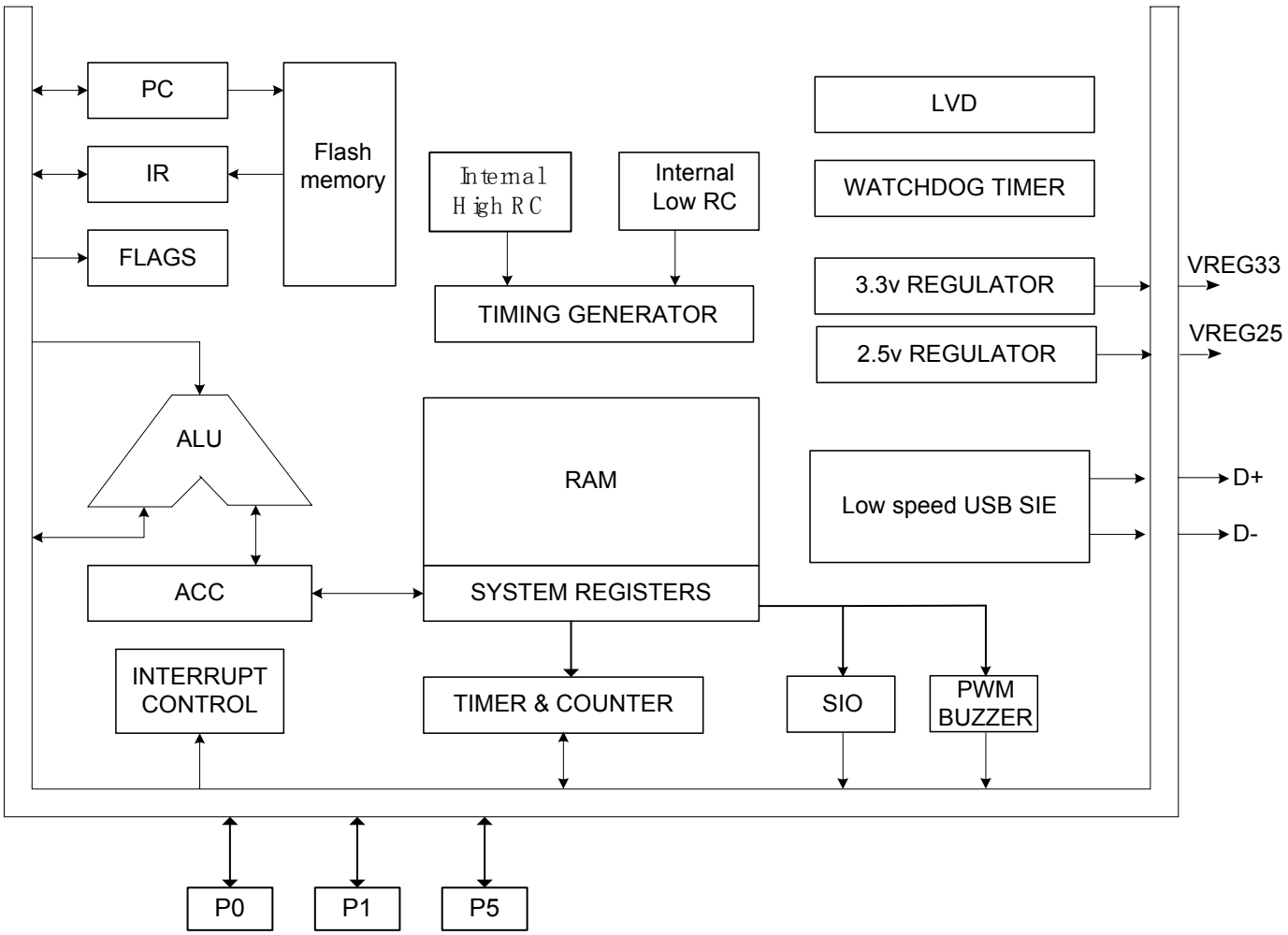
- **On chip watchdog timer.**

- **In-system re-programmability**
  Allows easy firmware update

☞ **Features Selection Table**

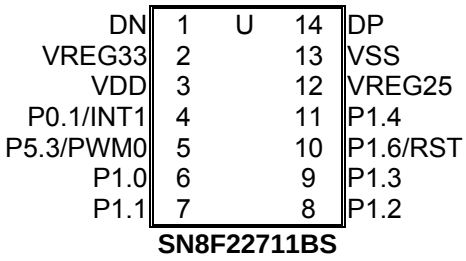
CHIP	ROM	RAM	STACK	TIMER		SIO	PWM	WAKE-UP PIN NO.	I/O pin	PACKAGE
				T0	TC0					
SN8F2271B	5K*16	192*8	8	V	V	V	-	7	10	QFN
SN8F22711B	5K*16	192*8	8	V	V	-	V	7	8	SOP
SN8F22721B	5K*16	192*8	8	V	V	V	V	10	14	DIP/SOP/SSOP

1.2 SYSTEM BLOCK DIAGRAM



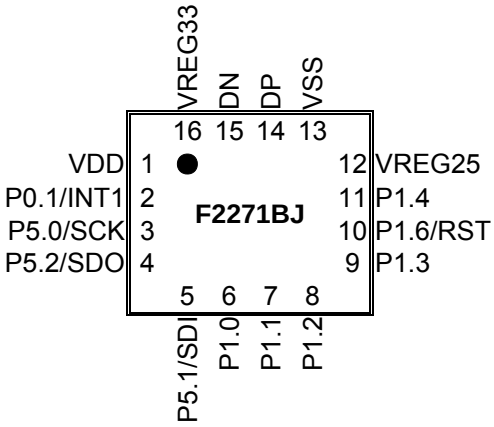
1.3 PIN ASSIGNMENT

SN8F22711BS (SOP 14 pins)

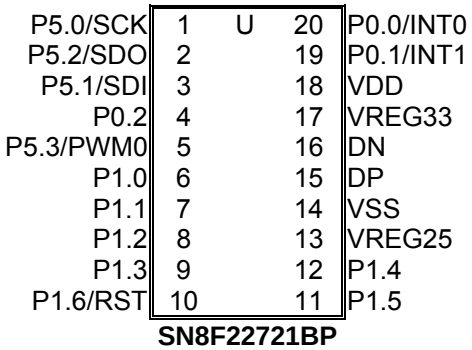


SN8F22711BS

SN8F2271BJ (QFN 16 pins)



SN8F22721BP (DIP 20 pins)
SN8F22721BS (SOP 20 pins)
SN8F22721BX (SSOP 20 pins)



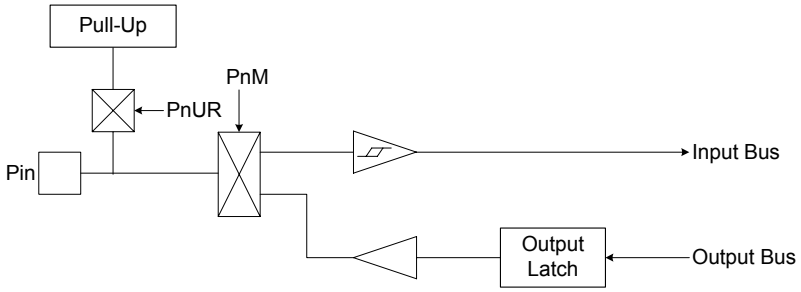
SN8F22721BP
SN8F22721BS
SN8F22721BX

1.4 PIN DESCRIPTIONS

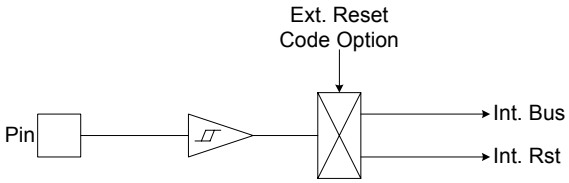
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital circuit.
P0.0/INT0	I/O	P0.0: Port 0.0 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. Built wakeup function. INT0: External interrupt 0 input pin.
P0.1/INT1	I/O	P0.1: Port 0.1 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. Built wakeup function. INT1: External interrupt 1 input pin.
P0.2	I/O	P0.2: Port 0.2 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. Built wakeup function.
P1[5:0]	I/O	P1: Port 1 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. Built wakeup function.
P1.6/RST	I/O	RST is system external reset input pin under Ext_RST mode, Schmitt trigger structure, active “low”, and normal stay to “high”. P1.6: Port 1.6 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. Built wakeup function.
P5.0/SCK	I/O	P5.0: Port 5.0 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. SCK: SIO output clock pin.
P5.1 /SDI	I/O	P5.1: Port 5.1 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. SDI: SIO data input pin.
P5.2/SDO	I/O	P5.2: Port 5.2 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. SDO: SIO data output pin.
P5.3/PWM0	I/O	P5: Port 5 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. PWM0: PMW output pin.
VREG25	P	2.5V power pin. Please connect 1uF capacitor to GND.
VREG33	P	3.3V power pin. Please connect XuF capacitor to GND. X=1~10.
D+, D-	I/O	USB differential data line.

1.5 PIN CIRCUIT DIAGRAMS

Port 0, 1, 5 structures:



Pin RST structure:

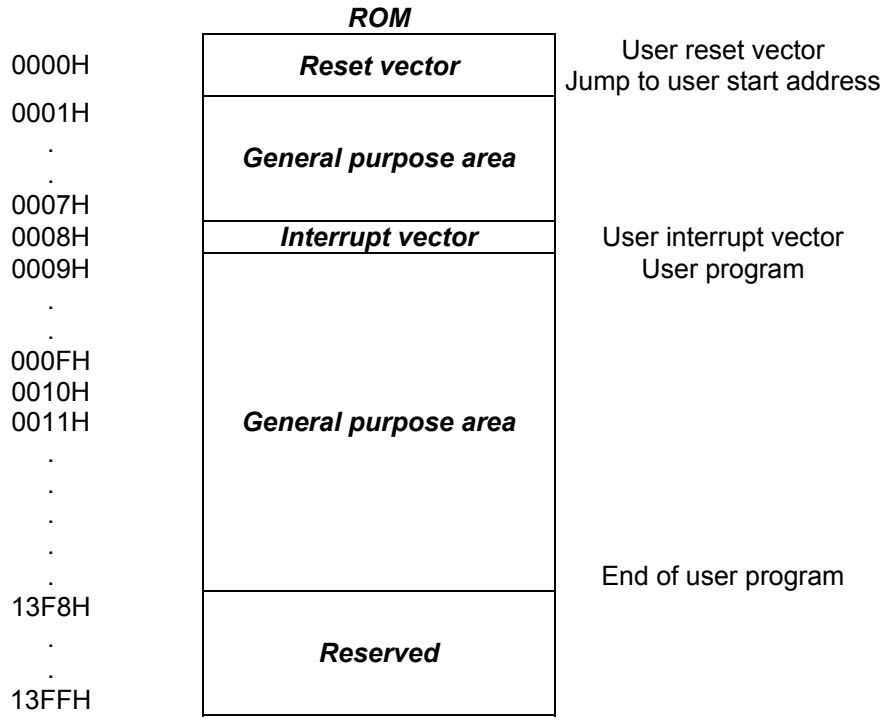


2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ ***5K words ROM***



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.
                …

                ORG     10H
START:                                  ; 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.

                ORG     10H
START:                                  ; 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 loop-up table errors. If Z register overflows, 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 loop-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 calculated Checksum from 00H to the end of user's code.**

```
                MOV     A,#END_USER_CODE$L
                B0MOV   END_ADDR1, A            ; Save low end address to end_addr1
                MOV     A,#END_USER_CODE$M
                B0MOV   END_ADDR2, A            ; Save middle end address to end_addr2
                CLR     Y                       ; Set Y to 00H
                CLR     Z                       ; Set Z to 00H
@@:
                MOVC
                B0BSET  FC                      ; Clear C flag
                ADD     DATA1, A                ; Add A to Data1
                MOV     A, R
                ADC     DATA2, A                ; Add R to Data2
                JMP     END_CHECK               ; Check if the YZ address =  the end of code
AAA:
                INCMS   Z                       ; Z=Z+1
                JMP     @B                      ; If Z != 00H calculate to next address
                JMP     Y_ADD_1                 ; If Z = 00H increase Y
END_CHECK:
                MOV     A, END_ADDR1
                CMPRS   A, Z                    ; Check if Z = low end address
                JMP     AAA                     ; If Not jump to checksum calculate
                MOV     A, END_ADDR2
                CMPRS   A, Y                    ; If Yes, check if Y = middle end address
                JMP     AAA                     ; If Not jump to checksum calculate
                JMP     CHECKSUM_END            ; If Yes checksum calculated is done.

Y_ADD_1:
                INCMS   Y                       ; Increase Y
                NOP
www.DataSheet4U.com
                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 with pull up resistor.
	P16	Enable P1.6 I/O function.
Security0	Enable	Enable ROM code Security function. Lock Address (0x1380~0x13FF) Security function.
	Disable	Disable ROM code Security function.

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

2.1.3 DATA MEMORY (RAM)

☞ ***192 X 8-bit RAM***

	Address	RAM location	
BANK 0	000h “ “ “ “ “ 07Fh	***General purpose area***	BANK 0
	080h “ “ “ “ “ 0FFh	***System register***	80h~FFh of Bank 0 store system registers (128 bytes).
		End of bank 0 area	
BANK1	100h “ “ “ “ “ 140h	***General purpose area***	BANK1

☞ ***24 x 8-bit RAM for USB DATA FIFO***

	24 x 8 RAM (USB FIFO)
00h ~ 07h	***Endpoint 0 RAM (8 byte)***
10h ~ 17h	***Endpoint 1 RAM (8 byte)***
18h ~ 1Fh	***Endpoint 2 RAM (8 byte)***

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	-	-	R	Z	Y	-	PFLAG	RBANK	TC0M	TC0C	TC0R	-	-	-	-	-
9	UDA	USTATUS	EP0OUT_CNT	USB_INT_EN	EP_ACK	EP_NAK	UE0R	UE1R	UE2R	-	-	-	-	-	-	-
A	-	-	-	UDP0_L	UDP0_H	UDR0_R	UDR0_W	EP1OUT_CNT	EP2OUT_CNT	-	-	UPID	UToggle	-	-	-
B	-	-	-	-	SIOM	SIOR	SIOB	-	P0M	-	PECMD	PEROML	PEROMH	PERAML	PERAMCNT	PEDGE
C	P1W	P1M	-	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	-	PCL	PCH
D	P0	P1	-	-	-	P5	-	-	T0M	T0C	-	-	-	-	-	STKP
E	P0UR	P1UR	-	-	-	P5UR	-	@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.
UDP0 = USB FIFO address pointer.
UDR0_W = USB FIFO write data buffer by UDP0 point to.
EP_ACK = Endpoint ACK flag register.
UToggle = USB endpoint toggle bit control register.
USTATUS = USB status register.
EPXOUT_CNT = USB endpoint 1~3 OUT token data byte counter
SIOM = SIO mode control register.
SIOB = SIO's data buffer.
PnM = Port n input/output mode register.
INTRQ = Interrupt request register.
INTRQ1 = Interrupt1 request register.
OSCM = Oscillator mode register.
TC0R = TC0 auto-reload data buffer.
Pn = Port n data buffer.
TnC = T0 counting register. n = 0, C0
PnUR = Port n pull-up resister control register.
P1W = Port 1 wakeup control register.
PEROM = ISP ROM address.
PERAMCNT = ISP RAM programming counter register.

Y, Z = Working, @YZ and ROM addressing register.
RBANK = RAM bank selection register.
UE0R~UE2R = Endpoint 0~2 control registers.
UDR0_R = USB FIFO read data buffer by UDP0 point to.
EP_NAK = Endpoint NAK flag register.
UDR0_W = USB FIFO write data buffer by UDP1 point to.
UPID = USB bus control register.
USB_INT_EN = USB interrupt enable/disable control register.
SIOR = SIO's clock reload buffer
PEDGE = P0.0, P0.1 edge direction register.
INTEN = Interrupt enable register.
INTEN1 = Interrupt1 enable register.
WDTR = Watchdog timer clear register.
PCH, PCL = Program counter.
TnM = Tn mode register. n = 0, C0
TnR = Tn register. n = C0
STKP = Stack pointer buffer.
@YZ = RAM YZ indirect addressing index pointer.
STK0~STK7 = Stack 0 ~ stack 7 buffer.
PECMD = ISP command register.
PERAM = ISP RAM mapping address.

2.1.4.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD				C	DC	Z	R/W	PFLAG
087H								RBNKS0	R/W	RBANK
088H	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	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	EP0_SETUP	EP0_IN	EP0_OUT	R/W	USTATUS
092H						UEP0OC2	UEP0OC1	UEP0OC0	R/W	EP0OUT_CNT
093H	REG_EN	DN_PU_EN					EP2NAK _INT_EN	EP1NAK _INT_EN	R/W	USB_INT_EN
094H							EP2_ACK	EP1_ACK	R/W	EP_ACK
095H							EP2_NAK	EP1_NAK	R/W	EP_NAK
096H		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
0A3H	UDP07	UDP06	UDP05	UDP04	UDP03	UDP02	UDP01	UDP00	R/W	UDP0_L
0A4H	WE0	RD0							R/W	UDP0_H
0A5H	UDR0_R7	UDR0_R6	UDR0_R5	UDR0_R4	UDR0_R3	UDR0_R2	UDR0_R1	UDR0_R0	R/W	UDR0_R
0A6H	UDR0_W7	UDR0_W6	UDR0_W5	UDR0_W4	UDR0_W3	UDR0_W2	UDR0_W1	UDR0_W0	R/W	UDR0_W
0A7H					UEP1OC3	UEP1OC2	UEP1OC1	UEP1OC0	R/W	EP1OUT_CNT
0A8H					UEP2OC3	UEP2OC2	UEP2OC1	UEP2OC0	R/W	EP2OUT_CNT
0ABH						UBDE	DDP	DDN	R/W	UPID
0ACH							EP2_DATA0 /1	EP1_DATA0 /1	R/W	Utoggle
0B4H	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	SEDGE	SP	R/W	SIOM
0B5H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	W	SIOR
0B6H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB
0B8H						P02M	P01M	P00M	R/W	P0M
0BAH	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0	W	PECMD
0BBH	PEROML7	PEROML6	PEROML5	PEROML4	PEROML3	PEROML2	PEROML1	PEROML0	R/W	PEROML
0BCH	PEORMH7	PEORMH6	PEORMH5	PEORMH4	PEORMH3	PEORMH2	PEORMH1	PEORMH0	R/W	PEORMH
0BDH	PERAML7	PERAML6	PERAML5	PERAML4	PERAML3	PERAML2	PERAML1	PERAML0	R/W	PERAML
0BEH	PERAMCNT 4	PERAMCNT 3	PERAMCNT 2	PERAMCNT 1	PERAMCNT 0			PERAML8	R/W	PERAMCNT
0BFH					P01G1	P01G0	P00G1	P00G0	R/W	PEDGE
0C0H		P16W	P15W	P14W	P13W	P12W	P11W	P10W	R/W	P1W
0C1H		P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C5H					P53M	P52M	P51M	P50M	R/W	P5M
0C8H		USBIRQ	TC0IRQ	T0IRQ	SIOIRQ	WAKEIRQ	P01IRQ	P00IRQ	R/W	INTRQ
0C9H		USBIEN	TC0IEN	T0IEN	SIOIEN	WAKEIEN	P01IEN	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						P02	P01	P00	R/W	P0
0D1H		P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D5H					P53	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H						P02R	P01R	P00R	W	P0UR
0E1H		P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E5H					P53R	P52R	P51R	P50R	W	P5UR
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H				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.

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

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

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

Bit 2 **C:** Carry flag
1 = Addition with carry, subtraction without borrowing, rotation with shifting out logic "1", comparison result ≥ 0.
0 = Addition without carry, subtraction with borrowing signal, rotation with shifting out logic "0", comparison result < 0.

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

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

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

2.1.4.6 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
                …
www.DataSheet4U.com …
```

2.1.4.7 Y, Z REGISTERS

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

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

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

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

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

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

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

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

CLR_YZ_BUF:
        CLR     @YZ         ; Clear @YZ to be zero

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

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


2.1.4.8 R REGISTERS

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

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

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

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

2.2 ADDRESSING MODE

2.2.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.2.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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


2.2.3 INDIRECTLY ADDRESSING MODE

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

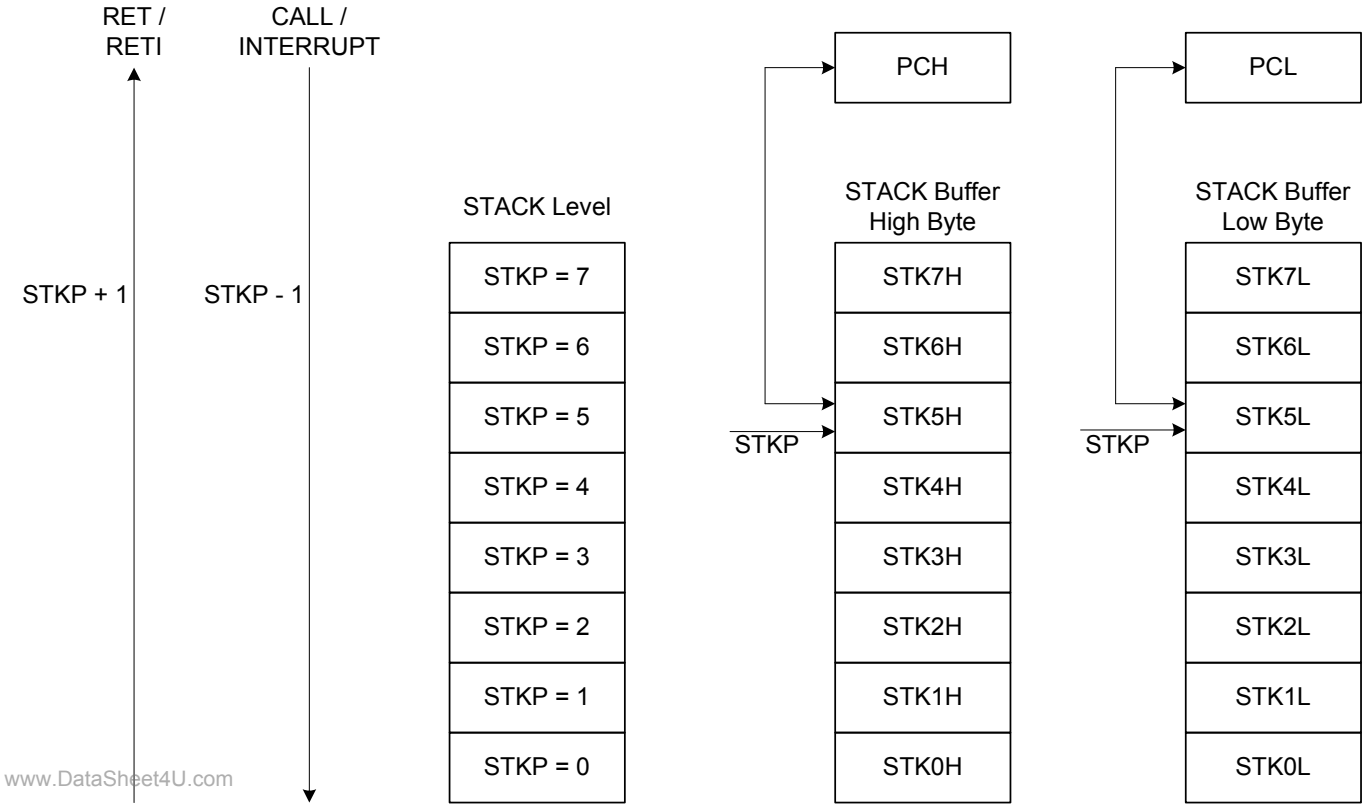
- **Example: Indirectly addressing mode with @YZ register.**

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

2.3 STACK OPERATION

2.3.1 OVERVIEW

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



2.3.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 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)

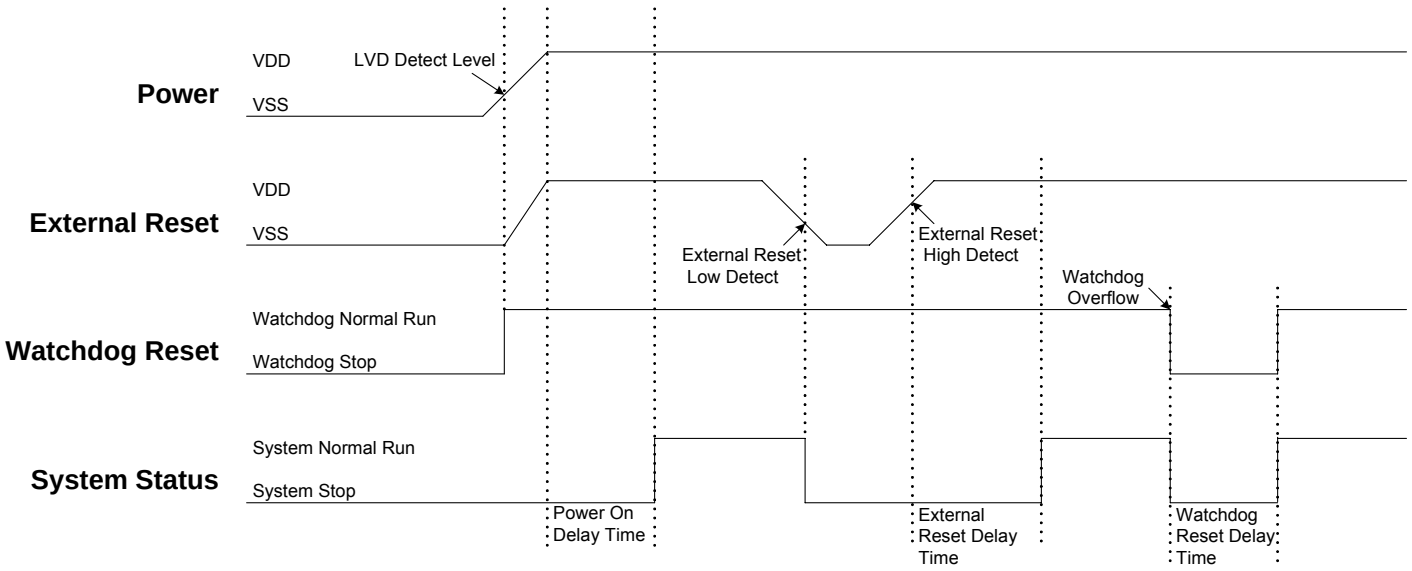
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

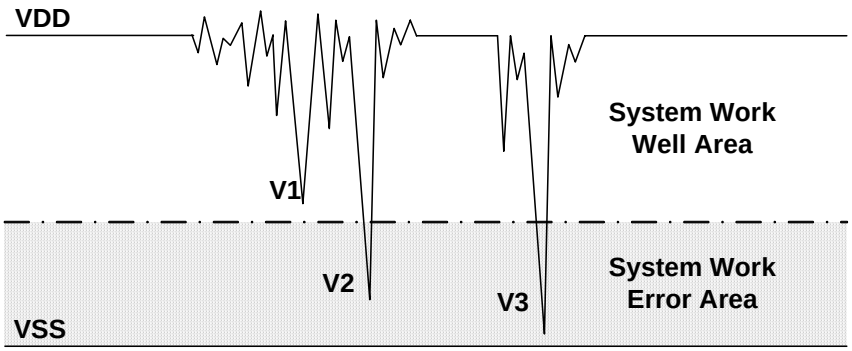
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

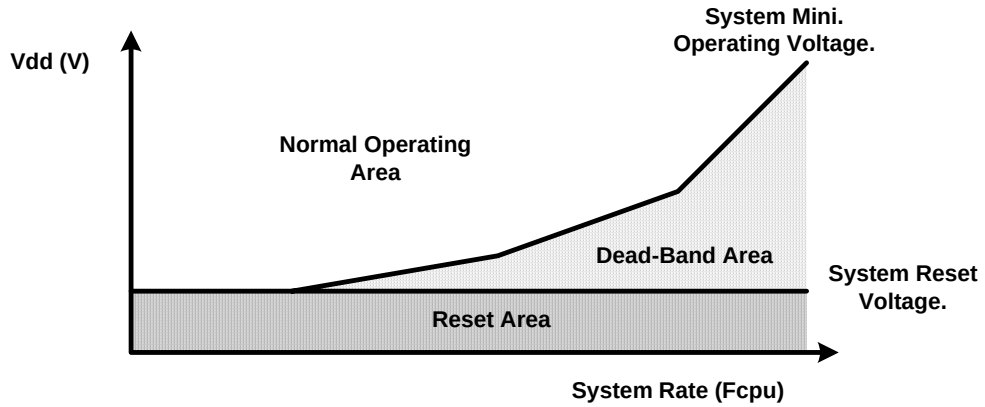
AC application:

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

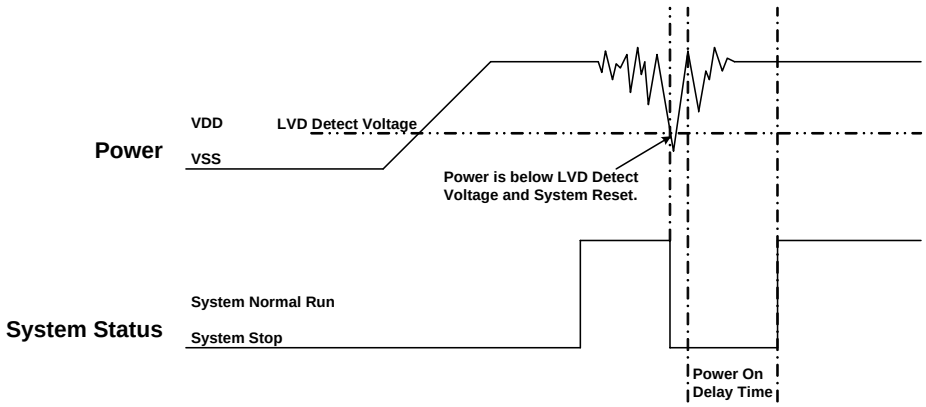
How to improve the brown reset condition? There are some methods to improve brown out reset as following.

- **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

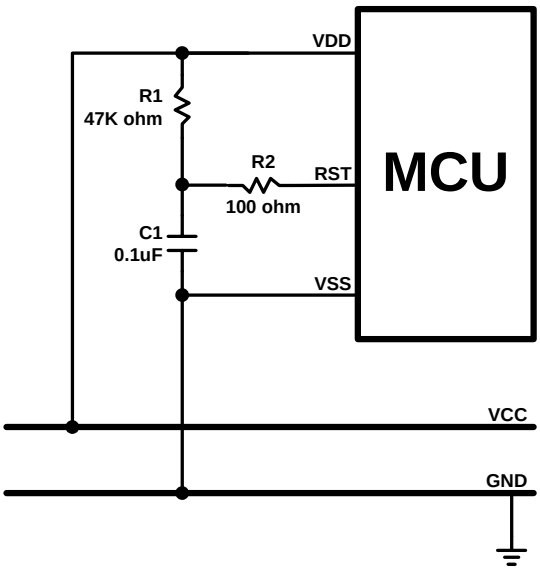
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

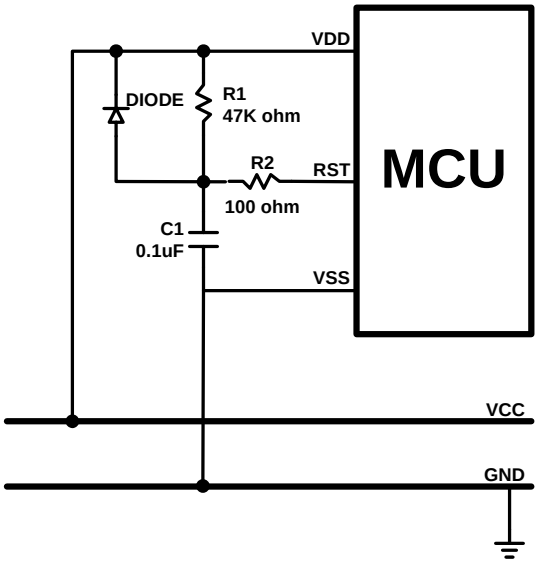
3.6.1 Simply RC Reset Circuit



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.***

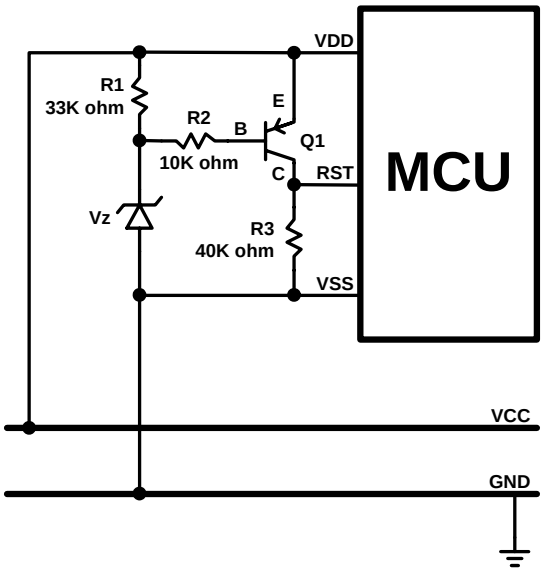
3.6.2 Diode & RC Reset Circuit



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.

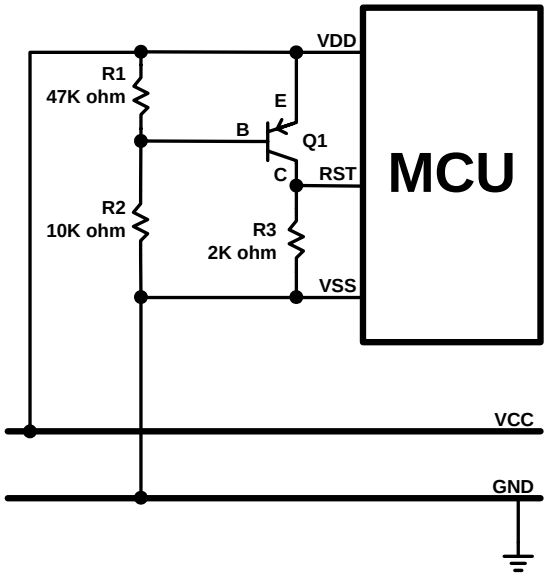
✷ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

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 "Vz + 0.7V", the C terminal of the PNP transistor outputs high voltage and MCU operates normally. When VDD is below "Vz + 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

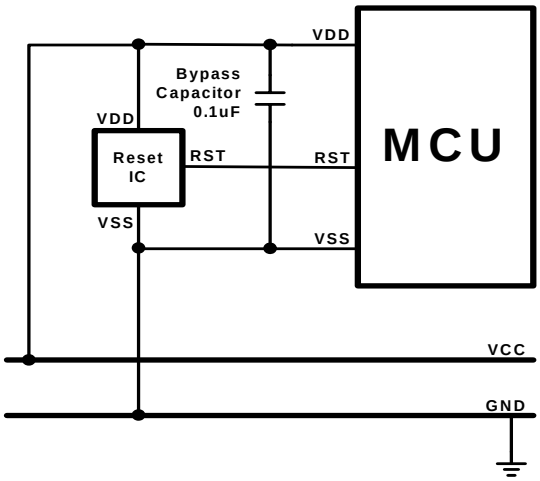


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to "0.7V x (R1 + R2) / R1", the C terminal of the PNP transistor outputs high voltage and MCU operates normally. When VDD is below "0.7V x (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 24 KHz).

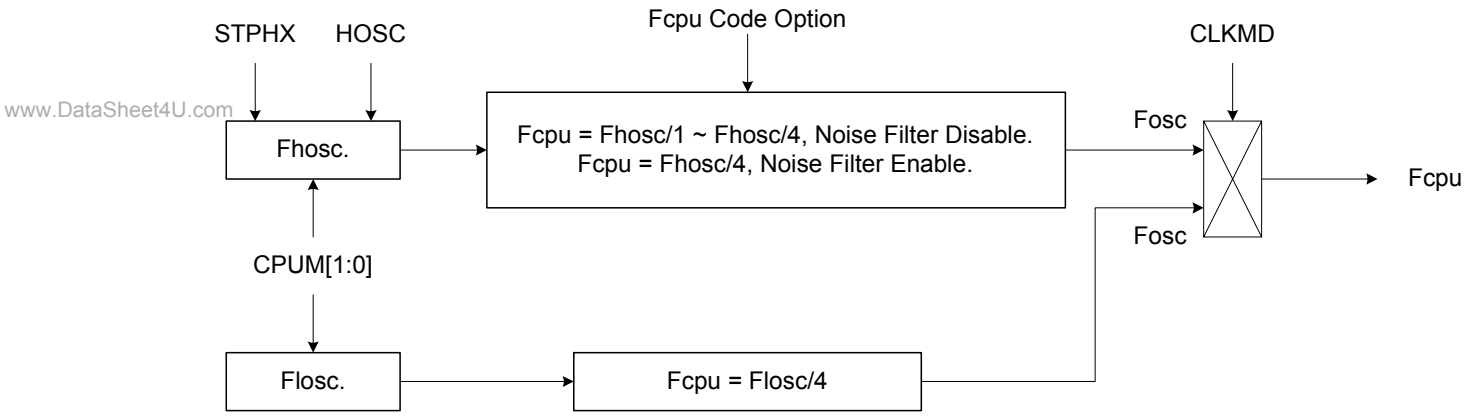
Both the high-speed clock and the low-speed clock can be system clock (Fosc). The system clock in slow mode is divided by 4 to be the instruction cycle (Fcpu).

☞ **Normal Mode (High Clock):** **Fcpu = Fhosc / N**, N = 1 ~ 4, Select N by Fcpu code option.

☞ **Slow Mode (Low Clock):** **Fcpu = Flosc/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 **Fhosc/4** when noise filter enable.

4.2 CLOCK BLOCK DIAGRAM



- HOSC: High_Clk code option.
- Fhosc: Internal high-speed clock.
- Flosc: Internal low-speed RC clock (Typical 24 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 24KHz.

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

☞ ***Flosc = Internal low RC oscillator (24KHz).***

☞ ***Slow mode Fcpu = Flosc / 4***

There are two conditions to stop internal low RC. One is power down mode, and the other is green mode of 24KHz mode and watchdog disable. If system is in 24KHz mode and watchdog disable, only 24KHz 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 (24 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
```

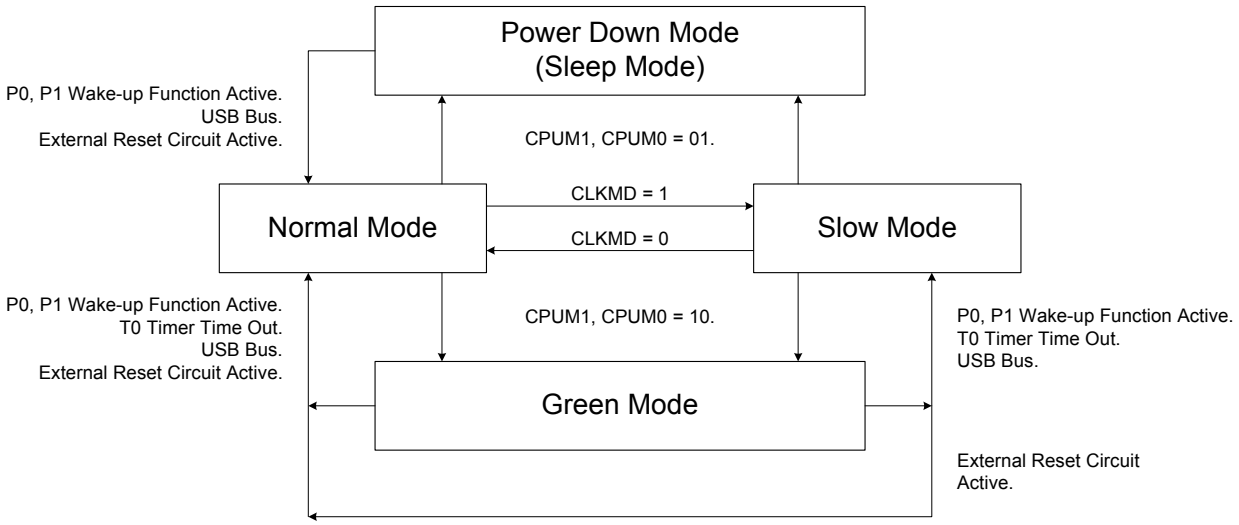
✸ ***Note: Do not measure the RC frequency directly from XIN; the probe impendence will affect the RC frequency.***

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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



System Mode Switching Diagram

Operating mode description

MODE	NORMAL	SLOW	GREEN	POWER DOWN (SLEEP)	REMARK
IHRC	Running	By STPHX	By STPHX	Stop	
ILRC	Running	Running	Running	Stop	
CPU instruction	Executing	Executing	Stop	Stop	
T0 timer	*Active	*Active	*Active	Inactive	* Active if T0ENB=1
TC0 timer	*Active	*Active	Inactive	Inactive	* Active if TC0ENB=1
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	T0	All inactive	
External interrupt	All active	All active	All active	All inactive	
Wakeup source	-	-	P0, P1, T0, Reset	P0, P1, Reset	

- **IHRC:** Internal high clock (6MHz RC oscillator)
- **ILRC:** Internal low clock (24KHz 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=24KHz (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.***

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

```
; Set T0 timer wakeup function.
        B0BCLR      FT0IEN      ; To disable T0 interrupt service
        B0BCLR      FT0ENB      ; To disable T0 timer
        MOV         A,#20H      ;
        B0MOV       T0M,A       ; To set T0 clock = Fcpu / 64
        MOV         A,#74H
        B0MOV       T0C,A       ; To set T0C initial value = 74H (To set T0 interval = 10 ms)
        B0BCLR      FT0IEN      ; To disable T0 interrupt service
        B0BCLR      FT0IRQ      ; To clear T0 interrupt request
        B0BSET      FT0ENB      ; To enable T0 timer
; Go into green mode
        B0BCLR      FCPUM0      ;To set CPUMx = 10
        B0BSET      FCPUM1
```

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

5.3 WAKEUP

5.3.1 OVERVIEW

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

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

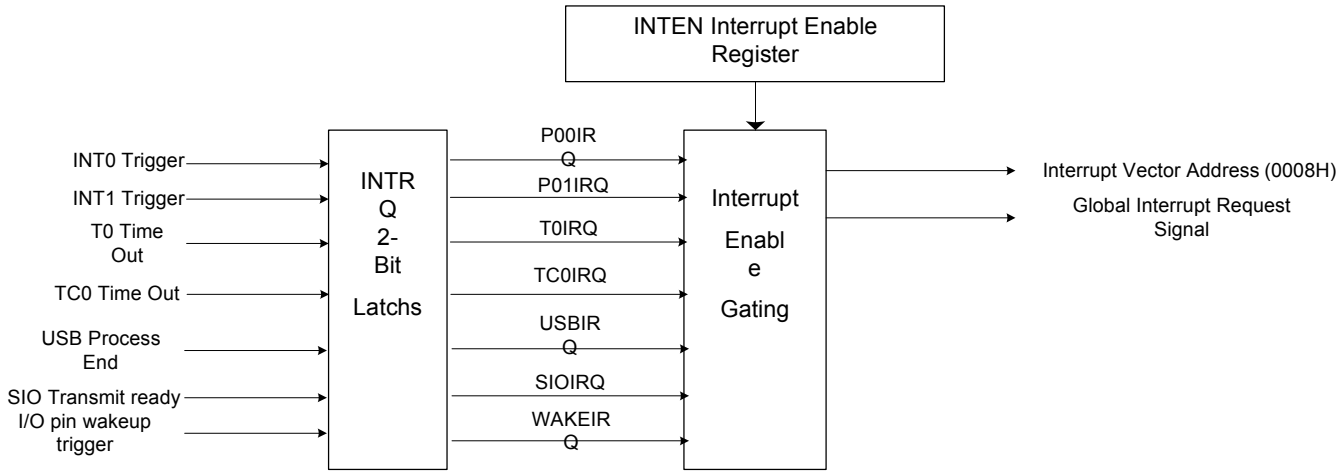
The wakeup time = 1/Fosc * 2048 = 0.341 ms (Fosc = 6MHz)

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

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 7 interrupt sources, including 4 internal interrupt (T0/TC0/USB/SIO) and two external interrupt (INT0/INT1). The external interrupt can wakeup the chip while the system is switched from power down mode to high-speed normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to “0” for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to “1” to accept the next interrupts’ request. All of the interrupt request signals are stored in INTRQ register.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN		USBIEN	TC0IEN	T0IEN	SIOIEN	WAKEIEN	P01IEN	P00IEN
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 0 **P00IEN:** External P0.0 interrupt (INT0) control bit.
0 = Disable INT0 interrupt function.
1 = Enable INT0 interrupt function.

Bit 1 **P01IEN:** External P0.1 interrupt (INT1) control bit.
0 = Disable INT1 interrupt function.
1 = Enable INT1 interrupt function.

Bit 2 **WAKEIEN:** I/O PORT0 & PORT 1 WAKEUP interrupt control bit.
0 = Disable WAKEUP interrupt function.
1 = Enable WAKEUP interrupt function.

Bit 3 **SIOIEN:** SIO interrupt control bit.
0 = Disable SIO interrupt function.
1 = Enable SIO interrupt function.

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

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

Bit 6 **USBIEN:** USB interrupt control bit.
0 = Disable USB interrupt function.
1 = Enable USB 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		USBIRQ	TC0IRQ	T0IRQ	SIOIRQ	WAKEIRQ	P01IRQ	P00IRQ
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 0 **P00IRQ:** External P0.0 interrupt (INT0) request flag.
0 = None INT0 interrupt request.
1 = INT0 interrupt request.

Bit 1 **P01IRQ:** External P0.1 interrupt (INT1) request flag.
0 = None 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 3 **SIOIRQ:** SIO interrupt request flag.
0 = None SIO interrupt request.
1 = SIO interrupt request.

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

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

Bit 6 **USBIRQ:** USB interrupt request flag.
0 = None USB interrupt request.
1 = USB interrupt request.

6.4 GIE GLOBAL INTERRUPT OPERATION

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

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

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

Example: Set global interrupt control bit (GIE).

```
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

            ORG     10H
START:
            …

INT_SERVICE:
            PUSH                ; Save ACC and PFLAG to buffers.
            …
            …
            POP                 ; Load ACC and PFLAG from buffers.

            RETI                ; Exit interrupt service vector
            …
            ENDP
```

6.6 INT0 (P0.0) & INT1 (P0.1) INTERRUPT OPERATION

When the INT0/INT1 trigger occurs, the P00IRQ/P01IRQ will be set to "1" no matter the P00IEN/P01IEN is enable or disable. If the P00IEN/P01IEN = 1 and the trigger event P00IRQ/P01IRQ is also set to be "1". As the result, the system will execute the interrupt vector (ORG 8). If the P00IEN/P01IEN = 0 and the trigger event P00IRQ/P01IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the P00IRQ/P01IRQ 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/INT1 interrupt request flag (INT0IRQ/INT1IRQ) 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: INT1 interrupt request can be latched by P0.1 wake-up trigger.***

✸ ***Note: The interrupt trigger direction of P0.0/P0.1 is control by PEDGE register.***

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE					P01G1	P01G0	P00G1	P00G0
Read/Write					R/W	R/W	R/W	R/W
After reset					1	0	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).

Bit[3:2] **P01G[1:0]:** P0.1 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, #18H
B0MOV   PEDGE, A        ; Set INT0 interrupt trigger as bi-direction edge.

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

Example: INT0 interrupt service routine.

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

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

        B0BCLR  FP00IRQ         ; Reset P00IRQ
        …                       ; INT0 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.7 T0 INTERRUPT OPERATION

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

➢ **Example: T0 interrupt request setup.**

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

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

        B0BSET      FGIE            ; Enable GIE
```

Example: T0 interrupt service routine.

```
        ORG         8               ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

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

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

        B0BCLR      FT0IRQ          ; Reset T0IRQ
        MOV         A, #74H
        B0MOV       T0C, A          ; Reset T0C.
        …                           ; T0 interrupt service routine
        …
EXIT_INT:
        …                           ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

6.8 TC0 INTERRUPT OPERATION

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

➢ **Example: TC0 interrupt request setup.**

```
        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
```

6.9 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
```

6.10 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
```

6.11 SIO INTERRUPT OPERATION

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

➢ **Example: SIO interrupt request setup.**

```
        B0BSET      FSIOIEN     ; Enable SIO interrupt service
        B0BCLR      FSIOIRQ     ; Clear SIO interrupt request flag
        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: SIO interrupt service routine.**

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

        B0BTS1      FSIOIRQ     ; Check SIOIRQ
        JMP         EXIT_INT    ; SIOIRQ = 0, exit interrupt vector

        B0BCLR      FSIOIRQ     ; Reset SIOIRQ
        …                       ; SIO interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.12 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
T0IRQ	T0C overflow
USBIRQ	USB process finished
WAEKIRQ	I/O port0 & port1 wakeup MCU
SIOIRQ	SIO process finished

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

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

```
                ORG         8               ; Interrupt vector
                JMP         INT_SERVICE
INT_SERVICE:

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

INTP00CHK:                                  ; Check INT0 interrupt request
                B0BTS1      FP00IEN         ; Check P00IEN
                JMP         INTT0CHK        ; Jump check to next interrupt
                B0BTS0      FP00IRQ         ; Check P00IRQ
                JMP         INTP00
INTT0CHK:                                   ; Check T0 interrupt request
                B0BTS1      FT0IEN          ; Check T0IEN
                JMP         INTUSBCHK       ; Jump check to next interrupt
                B0BTS0      FT0IRQ          ; Check T0IRQ
                JMP         INTT0           ; Jump to T0 interrupt service routine
INTUSBCHK:                                  ; Check USB interrupt request
                B0BTS1      FUSBIEN         ; Check USBIEN
                JMP         INTWAKECHK      ; Jump check to next interrupt
                B0BTS0      FUSBIRQ         ; Check USBIRQ
                JMP         INTUSB          ; Jump to USB interrupt service routine
INTWAKECHK:                                 ; Check USB interrupt request
                B0BTS1      FWAKEIEN        ; Check WAKEIEN
                JMP         INTSIOCHK       ; Jump check to next interrupt
                B0BTS0      FWAKEIRQ        ; Check WAKEIRQ
                JMP         INTWAKEUP       ; Jump to WAKEUP interrupt service routine
INTSIOCHK:                                  ; Check SIO interrupt request
                B0BTS1      FSIOIEN         ; Check SIOIEN
                JMP         INT_EXIT        ; Jump check to next interrupt
                B0BTS0      FSIOIRQ         ; Check SIOIRQ
                JMP         INTSIO          ; Jump to SIO 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	-	-	-	-	-	P02M	P01M	P00M
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	-	P16M	P15M	P14M	P13M	P12M	P11M	P10M
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	-	-	-	-	P53M	P52M	P51M	P50M
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

Bit[7:0] **PnM[7:0]:** Pn mode control bits. (n = 0~3).
0 = Pn is input mode.
1 = Pn is output mode.

✸ ***Note:***
1. 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	-	-	-	-	-	P02R	P01R	P00R
Read/Write	-	-	-	-	-	W	W	W
After reset	-	-	-	-	-	0	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	-	P16R	P15R	P14R	P13R	P12R	P11R	P10R
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	-	-	-	-	P53R	P52R	P51R	P50R
Read/Write	-	-	-	-	W	W	W	W
After reset	-	-	-	-	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	-	-	-	-	P03	P02	P01	P00
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	-	P16	P15	P14	P13	P12	P11	P10
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

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	-	-	P53	P52	P51	P50
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	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 P5.3to be "1".
B0BSET  P5.3

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

7.4 I/O PORT1 WAKEUP CONTROL REGISTER

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

Bit [7:0] **P1nW:** Port 1 wakeup function control bit.
0 = Disable port 1 wakeup function.
1 = Enable port 1 wakeup function.

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 (24KHz).

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

VDD	*Internal Low RC Freq.*	*Watchdog Overflow Time*
5V	24KHz	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.

0CCH	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
```

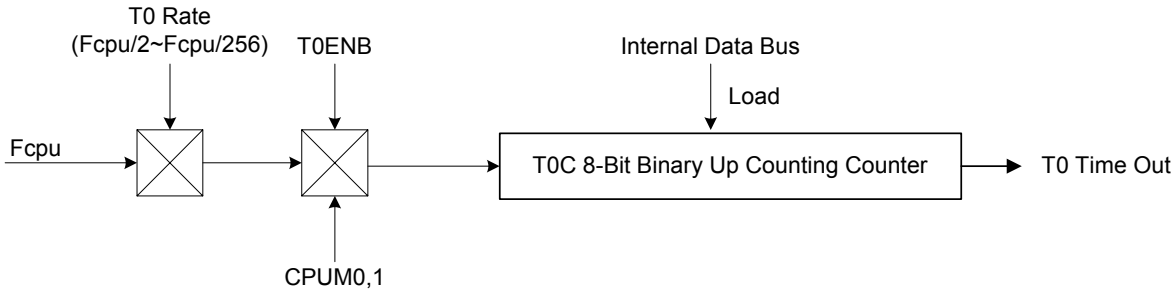
8.2 TIMER 0 (T0)

8.2.1 OVERVIEW

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

The main purpose of the T0 timer is as following.

- **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **Green mode wakeup function:** T0 can be green mode wake-up time as T0ENB = 1. System will be wake-up by T0 time out.



8.2.2 T0M MODE REGISTER

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

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

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

8.2.3 T0C COUNTING REGISTER

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

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

The equation of T0C initial value is as following.

T0C initial value = 256 - (T0 interrupt interval time * input clock)

Example: To set 1ms interval time for T0 interrupt. High clock is 6MHz. Fcpu=Fosc/1. Select T0RATE=010 (Fcpu/64).

*T0C initial value = 256 - (T0 interrupt interval time * input clock)*
*= 256 - (1ms * 6MHz / 1 / 64)*
*= 256 - (10^{-3} * 6 * 10^{6} / 1 / 64)*
= 162
= A2H

The basic timer table interval time of T0.

T0RATE	T0CLOCK	High speed mode (Fcpu = 6MHz)	
		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 T0 TIMER OPERATION SEQUENCE

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

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

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

☞ **Set T0 timer rate.**

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

☞ **Set T0 interrupt interval time.**

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

☞ **Set T0 timer function mode.**

```
B0BSET      FT0IEN          ; Enable T0 interrupt function.
```

☞ **Enable T0 timer.**

```
B0BSET      FT0ENB          ; Enable T0 timer.
```

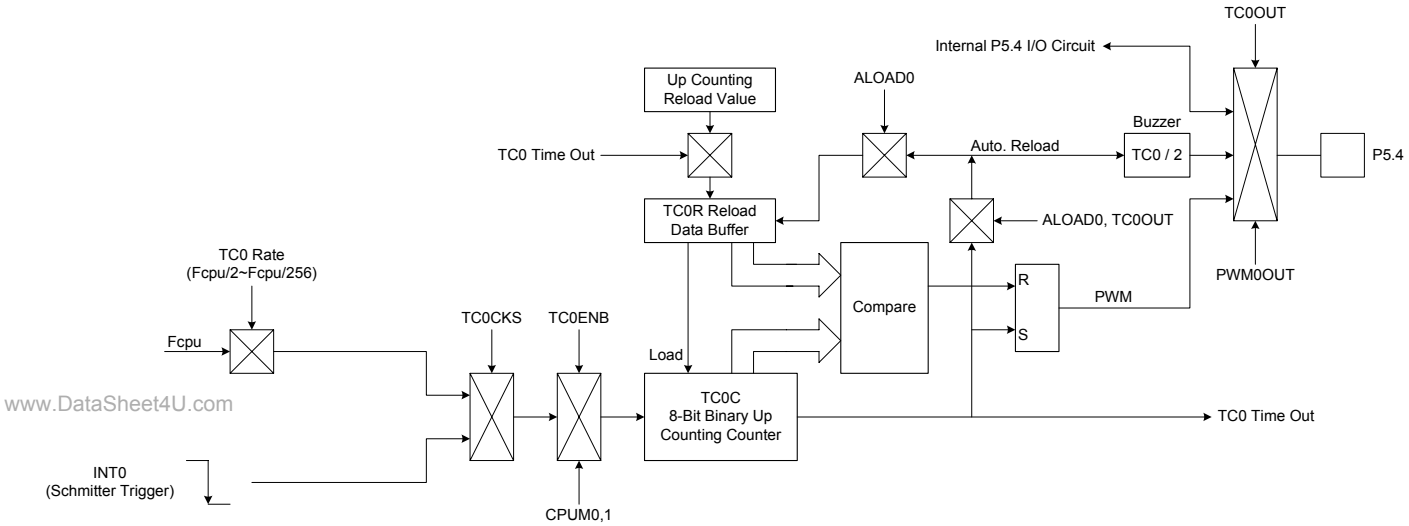
8.3 TIMER/COUNTER 0 (TC0)

8.3.1 OVERVIEW

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

The main purposes of the TC0 timer is as following.

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



8.3.2 TC0M MODE REGISTER

088H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **PWM0OUT:** PWM output control bit.
0 = Disable PWM output.
1 = Enable PWM output. PWM duty controlled by TC0OUT, ALOAD0 bits.

Bit 1 **TC0OUT:** TC0 time out toggle signal output control bit. **Only valid when PWM0OUT = 0.**
0 = Disable, P5.3 is I/O function.
1 = Enable, P5.3 is output TC0OUT signal.

Bit 2 **ALOAD0:** Auto-reload control bit. **Only valid when PWM0OUT = 0.**
0 = Disable TC0 auto-reload function.
1 = Enable TC0 auto-reload function.

Bit 3 **TC0CKS:** TC0 clock source select bit.
0 = Internal clock (Fcpu or Fosc).
1 = External clock from P0.0/INT0 pin.

Bit [6:4] **TC0RATE[2:0]:** TC0 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
…
110 = fcpu/4.
111 = fcpu/2.

Bit 7 **TC0ENB:** TC0 counter control bit.
0 = Disable TC0 timer.
1 = Enable TC0 timer.

✸ ***Note: When TC0CKS=1, TC0 became an external event counter and TC0RATE is useless. No more P0.0 interrupt request will be raised. (P0.0IRQ will be always 0).***

8.3.3 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 initial value = N - (TC0 interrupt interval time * 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) and no PWM output (PWM0=0). High clock is internal 6MHz. Fcpu=Fosc/1. Select TC0RATE=010 (Fcpu/64).**

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

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

8.3.4 TC0R AUTO-LOAD REGISTER

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

TC0 is double buffer design. If new TC0R value is set by program, the new value is stored in 1st buffer. Until TC0 overflow occurs, the new value moves to real TC0R buffer. This way can avoid TC0 interval time error and glitch in PWM and Buzzer output.

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

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 initial value = N - (TC0 interrupt interval time * input clock)

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

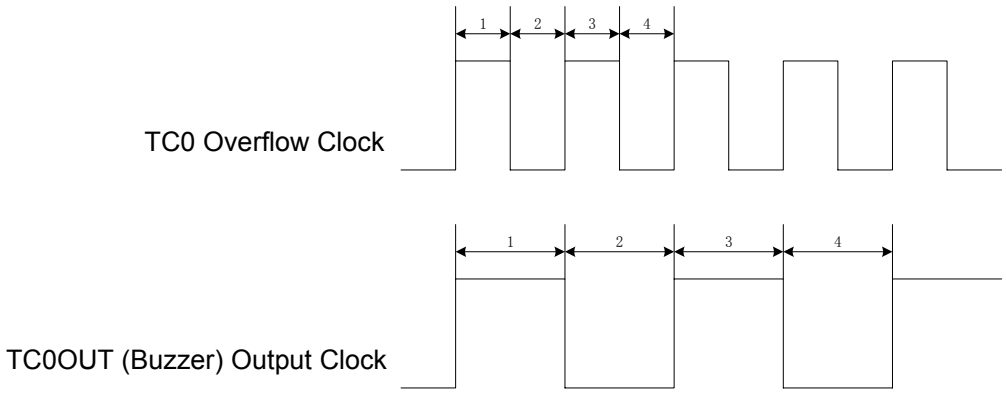
TC0CKS	PWM0	ALOAD0	TC0OUT	N	TC0R valid value	TC0R value binary type
0	0	x	x	256	0x00~0xFF	00000000b~11111111b
	1	0	0	256	0x00~0xFF	00000000b~11111111b
	1	0	1	64	0x00~0x3F	xx000000b~xx111111b
	1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
	1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
1	-	-	-	256	0x00~0xFF	00000000b~11111111b

➢ **Example: To set 1ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=0) and no PWM output (PWM0=0). High clock is internal 6MHz. Fcpu=Fosc/1. Select TC0RATE=010 (Fcpu/64).**

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

8.3.5 TC0 CLOCK FREQUENCY OUTPUT (BUZZER)

Buzzer output (TC0OUT) is from TC0 timer/counter frequency output function. By setting the TC0 clock frequency, the clock signal is output to P5.3 and the P5.3 general purpose I/O function is auto-disable. The TC0OUT frequency is divided by 2 from TC0 interval time. TC0OUT frequency is 1/2 TC0 frequency. The TC0 clock has many combinations and easily to make difference frequency. The TC0OUT frequency waveform is as following.



➢ **Example: Setup TC0OUT output from TC0 to TC0OUT (P5.3). The external high-speed clock is 4MHz. The TC0OUT frequency is 0.5KHz. Because the TC0OUT signal is divided by 2, set the TC0 clock to 1KHz. The TC0 clock source is from external oscillator clock. T0C rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 131.**

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

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



```
B0BSET    FTC0OUT         ; Enable TC0 output to P5.3 and disable P5.3 I/O function
B0BSET    FALOAD1         ; Enable TC0 auto-reload function
B0BSET    FTC0ENB         ; Enable TC0 timer
```

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

8.3.6 TC0 TIMER OPERATION SEQUENCE

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

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

☞

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

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

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

☞ **Set TC0 timer clock source.**

```
; Select TC0 internal / external clock source.
B0BCLR    FTC0CKS     ; Select TC0 internal clock source.
or
B0BSET    FTC0CKS     ; Select TC0 external clock source.
```

☞ **Set TC0 timer auto-load mode.**

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

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

```
; Set TC0 interrupt interval time, TC0OUT (Buzzer) frequency or PWM duty.
MOV       A,#7FH      ; TC0C and TC0R value is decided by TC0 mode.
B0MOV     TC0C,A      ; Set TC0C value.
B0MOV     TC0R,A      ; Set TC0R value under auto reload mode or PWM mode.
```



```
; In PWM mode, set PWM cycle.
B0BCLR    FALOAD0     ; ALOAD0, TC0OUT = 00, PWM cycle boundary is
B0BCLR    FTC0OUT     ; 0~255.
or
B0BCLR    FALOAD0     ; ALOAD0, TC0OUT = 01, PWM cycle boundary is
B0BSET    FTC0OUT     ; 0~63.
or
B0BSET    FALOAD0     ; ALOAD0, TC0OUT = 10, PWM cycle boundary is
B0BCLR    FTC0OUT     ; 0~31.
or
B0BSET    FALOAD0     ; ALOAD0, TC0OUT = 11, PWM cycle boundary is
B0BSET    FTC0OUT     ; 0~15.
```

☞ **Set TC0 timer function mode.**

```
B0BSET    FTC0IEN     ; Enable TC0 interrupt function.
or
B0BSET    FTC0OUT     ; Enable TC0OUT (Buzzer) function.
or
B0BSET    FPWM0OUT    ; Enable PWM function.
```

☞ **Enable TC0 timer.**

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

8.4 PWM0 MODE

8.4.1 OVERVIEW

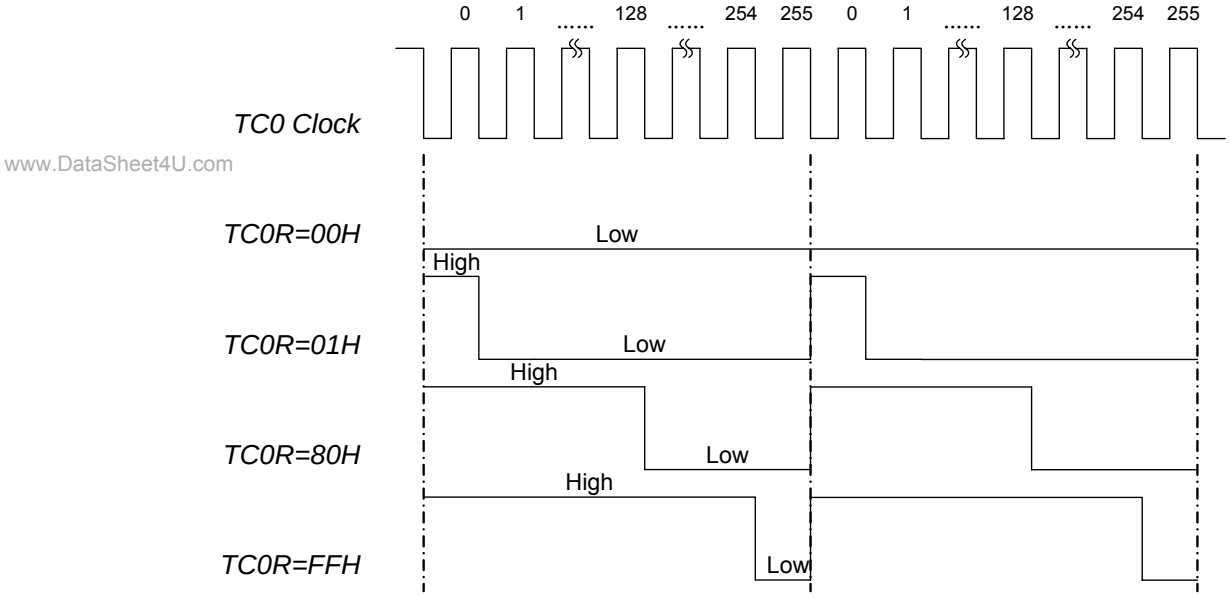
PWM function is generated by TC0 timer counter and output the PWM signal to PWM0OUT pin (P5.3). The 8-bit counter counts modulus 256, 64, 32, 16 controlled by ALOAD0, TC0OUT bits. The value of the 8-bit counter (TC0C) is compared to the contents of the reference register (TC0R). When the reference register value (TC0R) is equal to the counter value (TC0C), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWM0 output is TC0R/256, 64, 32, 16.

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

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

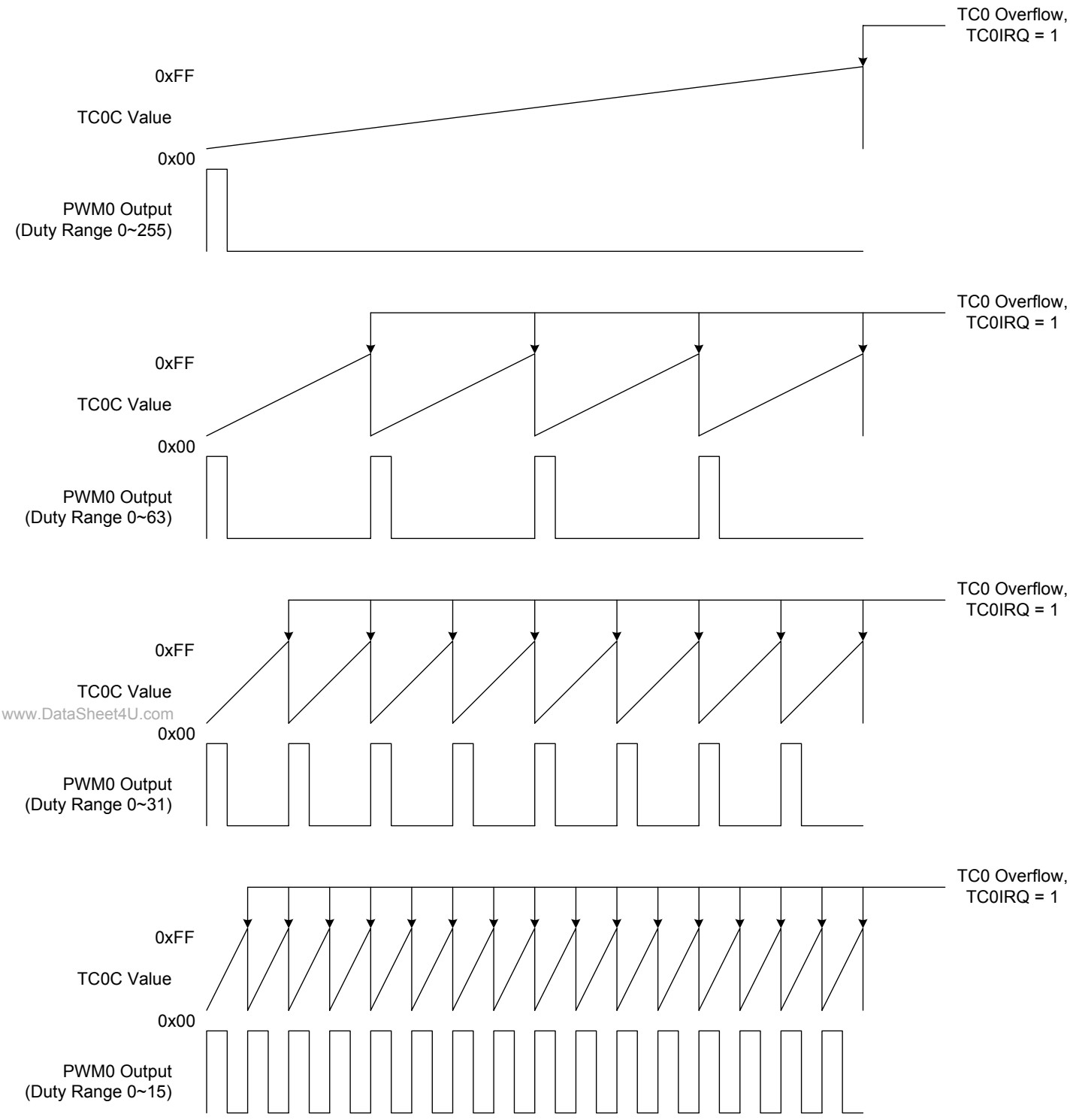
ALOAD0	TC0OUT	PWM duty range	TC0C valid value	TC0R valid bits value	MAX. PWM Frequency (Fcpu = 6MHz)	Remark
0	0	0/256~255/256	0x00~0xFF	0x00~0xFF	11.719K	Overflow per 256 count
0	1	0/64~63/64	0x00~0x3F	0x00~0x3F	46.875K	Overflow per 64 count
1	0	0/32~31/32	0x00~0x1F	0x00~0x1F	93.75K	Overflow per 32 count
1	1	0/16~15/16	0x00~0x0F	0x00~0x0F	187.5K	Overflow per 16 count

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



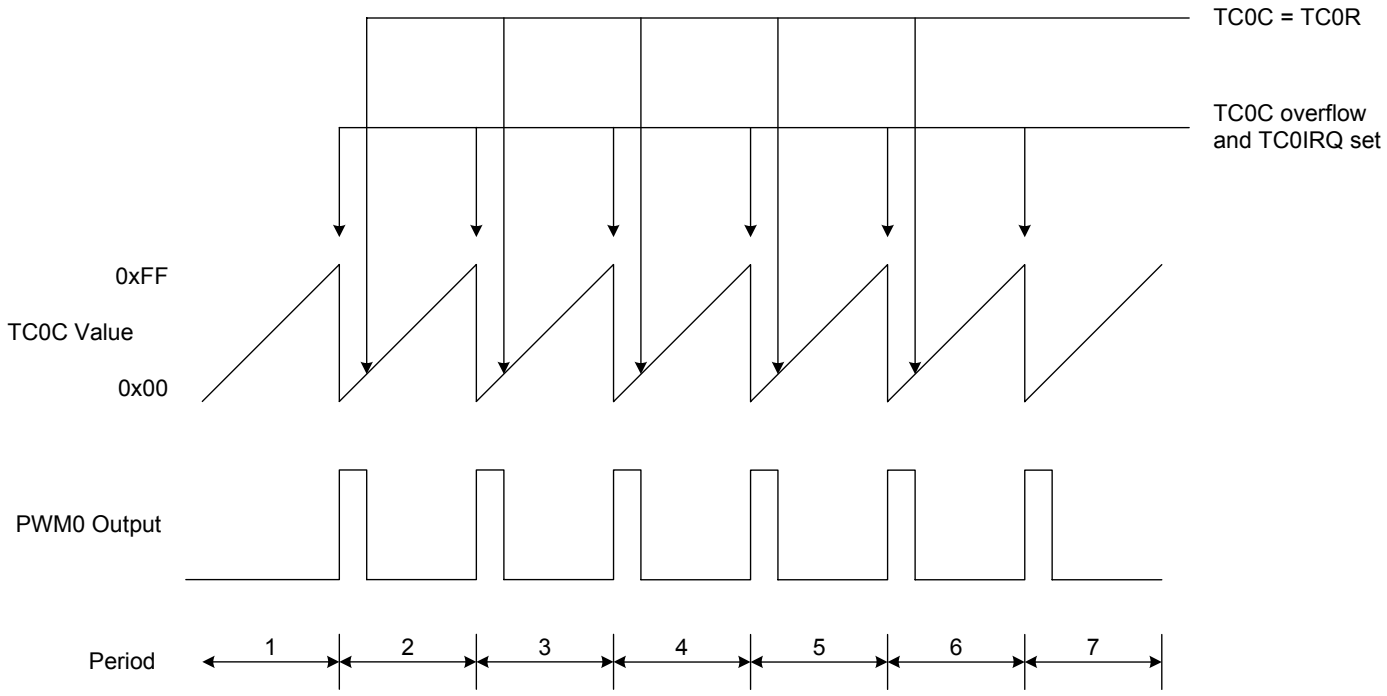
8.4.2 TCxIRQ and PWM Duty

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

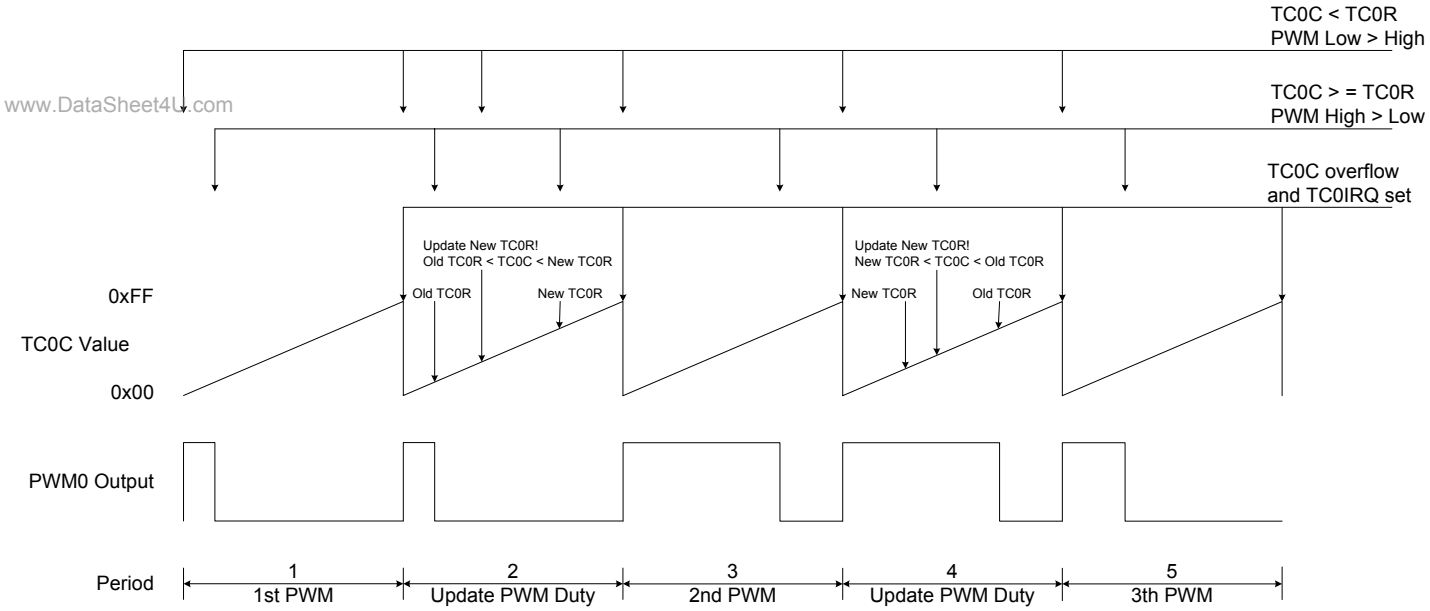


8.4.3 PWM Duty with TCxR Changing

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



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



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

8.4.4 PWM PROGRAM EXAMPLE

➢ **Example: Setup PWM0 output from TC0 to PWM0OUT (P5.3). The clock source is internal 6MHz. Fcpu = Fosc/1. The duty of PWM is 30/256. The PWM frequency is about 6KHz. The PWM clock source is from external oscillator clock. TC0 rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 30.**

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

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

B0BCLR  FTC0OUT         ; Set duty range as 0/256~255/256.
B0BCLR  FALOAD0
B0BSET  FPWM0OUT        ; Enable PWM0 output to P5.3 and disable P5.3 I/O function
B0BSET  FTC0ENB         ; Enable TC0 timer
```

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

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

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

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


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

9 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 can not 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					EP2NAK _INT_EN	EP1NAK _INT_EN
Read/Write	R/W	R/W					R/W	R/W
After reset	1	0					0	0

Bit [1:0] **EPnNAK_INT_EN [1:0]:** EP1~EP3 NAK transaction interrupts enable control bits. n = 1, 2, 3.
0 = Disable NAK transaction interrupt request.
1 = Enable NAK transaction interrupt request.

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 and I/O port 0, port5.

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_ACK	EP1_ACK
Read/Write							R/W	R/W
After reset							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) complete with an ACK.

9.5.6 USB endpoint's NAK handshaking flag REGISTER

The status of endpoint's NAK transaction.

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP_NAK							EP2_NAK	EP1_NAK
Read/Write							R/W	R/W
After reset							0	0

Bit [1:0] **EPn_NAK [1:0]:** EP1~EP3 NAK transaction. n = 1, 2. The bit is set whenever the endpoint that completes with an NAK received.
0 = the EPnNAK_INT_EN = 0 or the endpoint (interrupt pipe) doesn't complete with an NAK.
1 = the EPnNAK_INT_EN = 1 and the endpoint (interrupt pipe) complete with an NAK.

9.5.7 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	-	UE0M1	UE0M0	-	UE0C3	UE0C	UE0C1	UE0C0
Read/Write	-	R/W	R/W	-	R/W	R/W	R/W	R/W
After reset	-	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

9.5.8 USB ENDPOINT 1 ENABLE REGISTER

The communication with the USB host using endpoint 1, endpoint 1's FIFO is implemented as 16 bytes of dedicated RAM. The endponit1 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.

9.5.9 USB ENDPOINT 2 ENABLE REGISTER

The communication with the USB host using endpoint 2, endpoint 2's FIFO is implemented as 16 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.

9.5.10 USB DATA POINTER REGISTER

USB FIFO address pointer. Use the point to set the FIFO address for reading data from USB FIFO and writing data to USB FIFO.

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UDP0	UDP07	UDP06	UDP05	UDP04	UDP03	UDP02	UDP01	UDP00
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

Address [07]~address [00]: data buffer for endpoint 0.
Address [17]~address [10]: data buffer for endpoint 1.
Address [1F]~address [18]: data buffer for endpoint 2.

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UDP0_H	WE0	RD0						
Read/Write	R/W	R/W						
After reset	0	0						

Bit [6] **RE0:** Read data from USB FIFO's control bit.
0 = Read disable.
1 = Read enable.

Bit [7] **WE0:** Write data to USB FIFO's control bit.
0 = Write disable.
1 = Write enable.

9.5.11 USB DATA READ/WRITE REGISTER

0A5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UDR0_R	UDR0_R7	UDR0_R6	UDR0_R5	UDR0_R4	UDR0_R3	UDR0_R2	UDR0_R1	UDR0_R0
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

UDR0_R: Read the data from USB FIFO which UDP0 register point to.

0A6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UDR0_W	UDR0_W7	UDR0_W6	UDR0_W5	UDR0_W4	UDR0_W3	UDR0_W2	UDR0_W1	UDR0_W0
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

UDR0_W: Write the data to USB FIFO which UDP0 register point to.

9.5.12 USB ENDPOINT OUT TOKEN DATA BYTES COUNTER

Endpoint 1's OUT TOKEN DATA BYTES COUNTER.

0A7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP1OUT_CNT	-	-	-	-	UEP1OC3	UEP1OC2	UEP1OC1	UEP1OC0
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

Bit [4:0] **UEP1Cn:** Bytes counter of EP1 token data. Reset by firmware.

Endpoint 2's OUT TOKEN DATA BYTES COUNTER.

0A8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP2OUT_CNT	-	-	-	-	UEP2OC3	UEP2OC2	UEP2OC1	UEP2OC0
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

Bit [4:0] **UEP2Cn:** Bytes counter of EP2 token data. Reset by firmware.

9.5.13 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 direct drive USB bus.
0 = disable.
1 = enable.

Bit 7 **EP0OUT_EN:** Enable EP0 control data out.
0 = disable.
1 = enable to receive the EP0 continuous OUT TOKEN data over 8 bytes.

9.5.14 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 [2:0] 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.

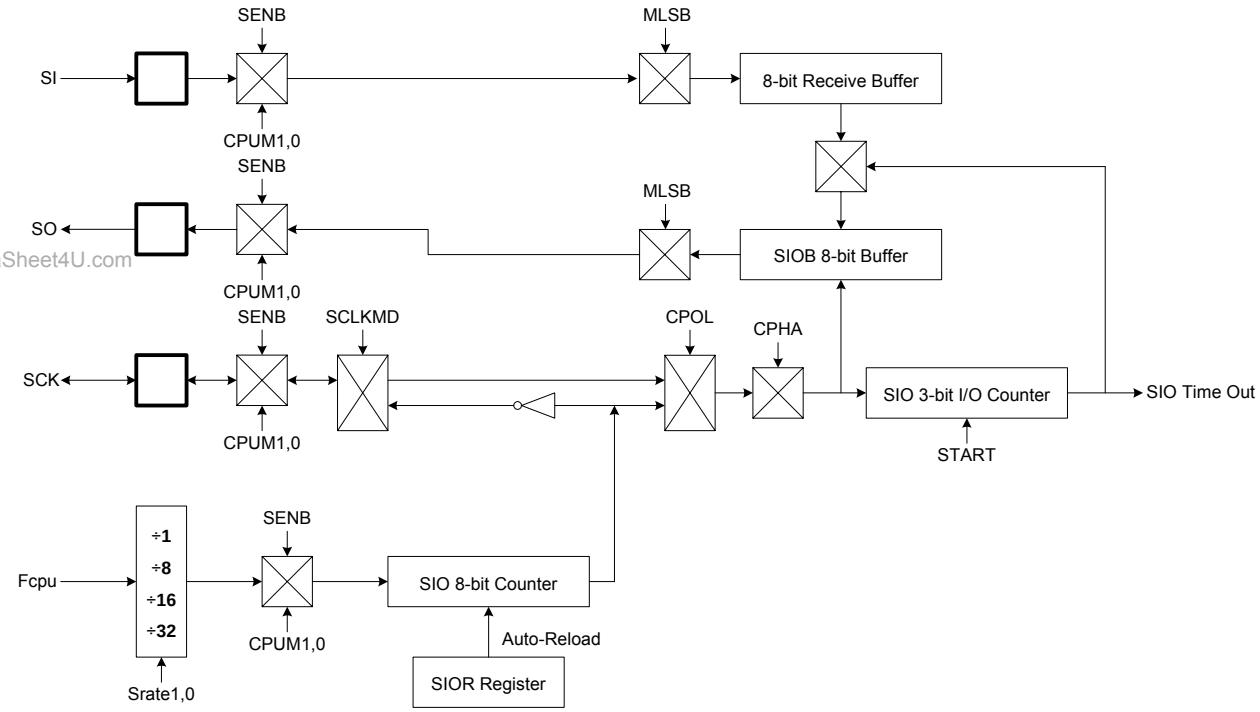
10 SERIAL INPUT/OUTPUT TRANSCEIVER

10.1 OVERVIEW

The SIO (serial input/output) transceiver allows high-speed synchronous data transfer between the SN8F2270B series MCU and peripheral devices or between several SN8F2270B devices. These peripheral devices may be Serial EEPROMs, shift registers, display drivers, etc. The SN8F2270B SIO features include the following:

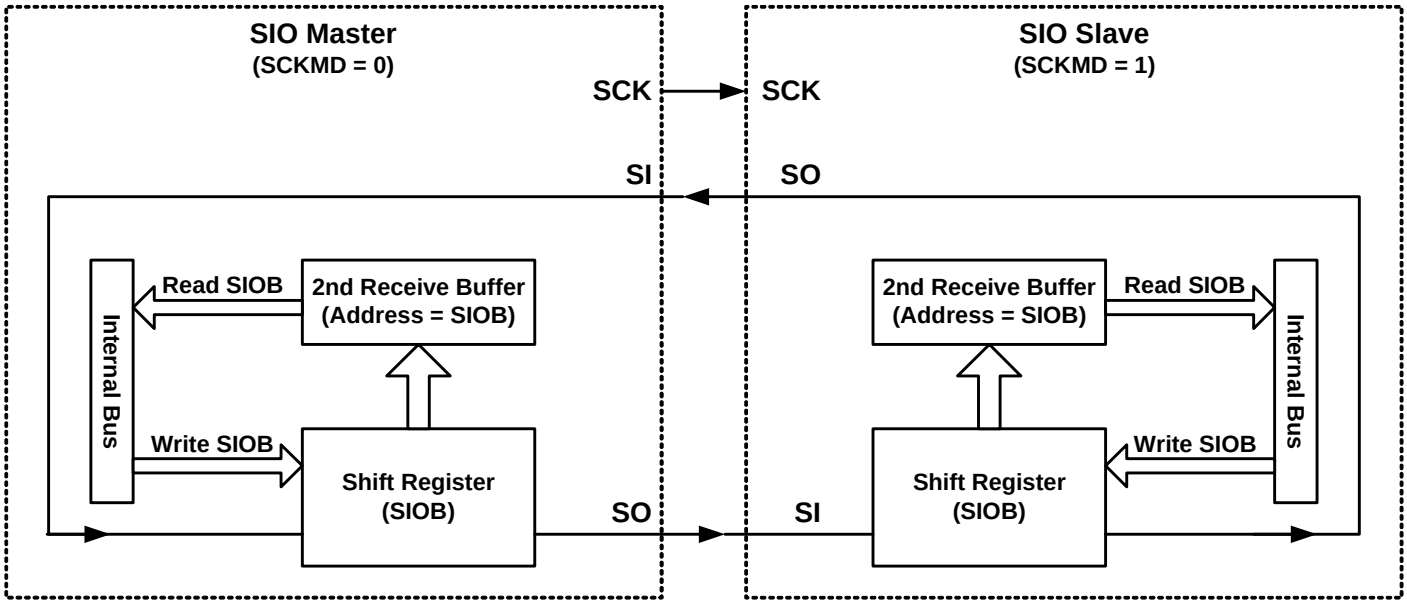
- **Full-duplex, 3-wire synchronous data transfer**
- **TX/RX or TX Only mode**
- **Master (SCK is clock output) or Slave (SCK is clock input) operation**
- **MSB/LSB first data transfer**
- **SDO (P5.2) is programmable open-drain output pin for multiple salve devices application**
- **Two programmable bit rates (Only in master mode)**
- **End-of-Transfer interrupt**

The SIOM register can control SIO operating function, such as: transmit/receive, clock rate, transfer edge and starting this circuit. This SIO circuit will transmit or receive 8-bit data automatically by setting SENB and START bits in SIOM register. The SIOB is an 8-bit buffer, which is designed to store transfer data. SIOC and SIOR are designed to generate SIO's clock source with auto-reload function. The 3-bit I/O counter can monitor the operation of SIO and announce an interrupt request after transmitting/receiving 8-bit data. After transferring 8-bit data, this circuit will be disabled automatically and re-transfer data by programming SIOM register.



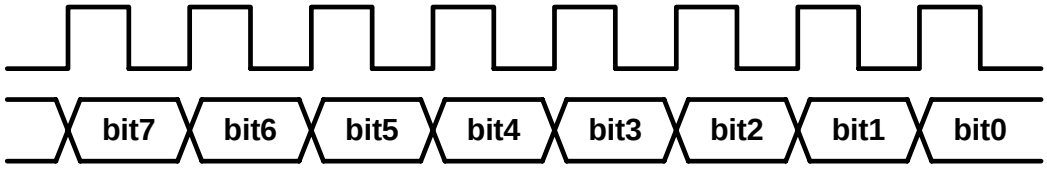
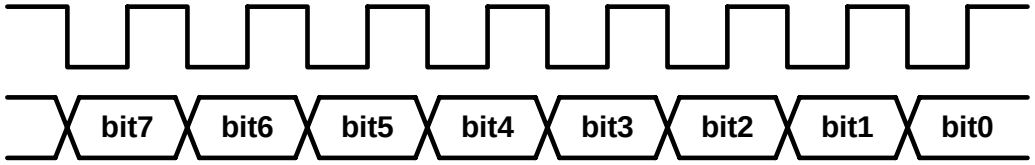
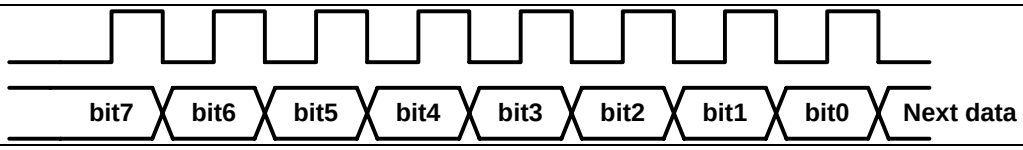
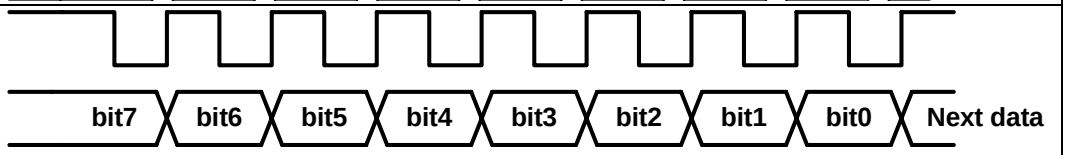
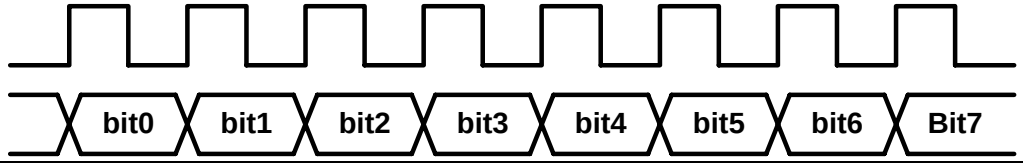
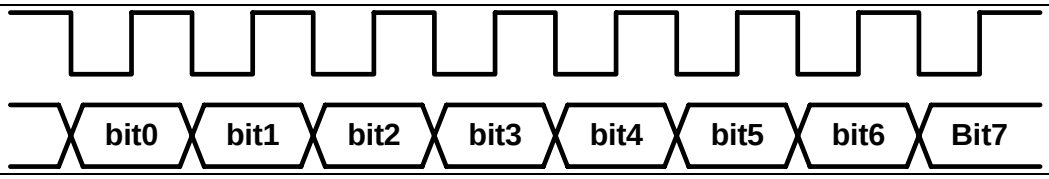
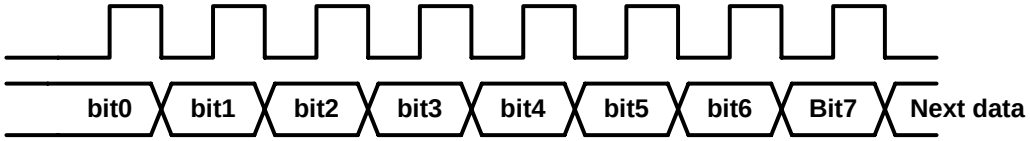
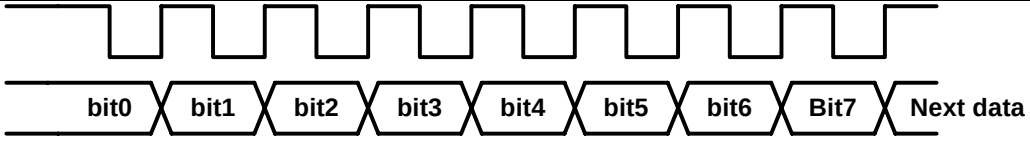
SIO Interface Circuit Diagram

The system is single-buffered in the transmit direction and double-buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SIOB Data Register before the entire shift cycle is completed. When receiving data, however, a received byte must be read from the SIOB Data Register before the next byte has been completely shifted in. Otherwise, the first byte is lost. Following figure shows a typical SIO transfer between two SN8F2270B micro-controllers. Master MCU sends SCK for initial the data transfer. Both master and slave MCU must work in the same clock edge direction, and then both controllers would send and receive data at the same time.



SIO Data Transfer Diagram

The SIO data transfer timing as following figure:

MLSB	CPOL	CPHA	SCK Idle Status	Diagrams
0	0	1	Low	
0	1	1	High	
0	0	0	Low	
0	1	0	High	
1	0	1	Low	
1	1	1	High	
1	0	0	Low	
1	1	0	High	

SIO Data Transfer Timing

10.2 SIOM MODE REGISTER

0B4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOM	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA
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 7 **SENB:** SIO function control bit.
0 = Disable (P5.0~P5.2 is general purpose I/O port).
1 = Enable (P5.0~P5.2 is SIO pins).

Bit 6 **START:** SIO progress control bit.
0 = End of transfer.
1 = Progressing.

Bit [5:4] **SRATE1:0:** SIO's transfer rate select bit. **These 2-bits are workless when SCKMD=1.**
00 = Fcpu.
01 = Fcpu/32
10 = Fcpu/16
11 = Fcpu/8.

Bit 3 **MLSB:** MSB/LSB transfer first.
0 = MSB transmit first.
1 = LSB transmit first.

Bit 2 **SCKMD:** SIO's clock mode select bit.
0 = Internal. (Master mode)
1 = External. (Slave mode)

Bit 1 **CPOL:** SIO's transfer clock edge select bit.
0 = SCK idle status is low status
1 = SCK idle status is high status

Bit 0 **CPHA:** The Clock Phase bit controls the phase of the clock on which data is sampled.
0 = Data receive at the fisrt clock phase.
1 = Data receive at the second clock phase.

✷ ***Note: 1. If SCKMD=1 for external clock, the SIO is in SLAVE mode. If SCKMD=0 for internal clock, the SIO is in MASTER mode.***
2. Don't set SENB and START bits in the same time. That makes the SIO function error.

Because SIO function is shared with Port5 for P5.0 as SCK, P5.1 as SDI and P5.2 as SDO.
The following table shown the Port5[2:0] I/O mode behavior and setting when SIO function enable and disable.

SENB=1 (SIO Function Enable)		
P5.0/SCK	(SCKMD=1) SIO source = External clock	P5.0 will change to Input mode automatically, no matter what P5M setting
	(SCKMD=0) SIO source = Internal clock	P5.0 will change to Output mode automatically, no matter what P5M setting
P5.1/SDI	P5.1 must be set as Input mode in P5M ,or the SIO function will be abnormal	
P5.2/SDO	SIO = Transmitter/Receiver	P5.2 will change to Output mode automatically, no matter what P5M setting
SENB=0 (SIO Function Disable)		
P5.0/P5.1/P5.2	Port5[2:0] I/O mode are fully controlled by P5M when SIO function is disable	

10.3 SIOB DATA BUFFER

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOB	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0
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

SIOB is the SIO data buffer register. It stores serial I/O transmit and receive data.

10.4 SIOR REGISTER DESCRIPTION

0B5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOR	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The SIOR is designed for the SIO counter to reload the counted value when end of counting. It is like a post-scaler of SIO clock source and let SIO has more flexible to setting SCK range. Users can set the SIOR value to setup SIO transfer time. **The valid SIOR value = 0x0 to 0xFD.** To setup SIOR value equation to desire transfer time is as following.

SCK frequency = SIO rate / (256 - SIOR);

SIOR = 256 - (1 / (SCK frequency) * SIO rate)

Example: Setup the SIO clock to be 1MHz. Fosc = 6MHz. SIO's rate = Fcpu/2. Fcpu = Fosc/1 = 6MHz.

SIOR = 256 – (1/(2MHz) * 12MHz/2)
= 256 – 3
= 253
= 0xFD

Example: Master, duplex transfer and transmit data on rising edge

```
        MOV     A,TXDATA        ; Load transmitted data into SIOB register.
        B0MOV   SIOB,A
        MOV     A,#0FEH         ; Set SIO clock
        B0MOV   SIOR,A
        MOV     A,#10000000B    ; Setup SIOM and enable SIO function.
        B0MOV   SIOM,A
        B0BSET  FSTART          ; Start transfer and receiving SIO data.
CHK_END:
        B0BTS0  FSTART          ; Wait the end of SIO operation.
        JMP     CHK_END
        B0MOV   A,SIOB          ; Save SIOB data into RXDATA buffer.
        MOV     RXDATA,A
```

Example: Slave, duplex transfer and transmit data on rising edge

```
        MOV     A,TXDATA        ; Load transfer data into SIOB register.
        B0MOV   SIOB,A
        MOV     A,# 10000100B   ; Setup SIOM and enable SIO function.
        B0MOV   SIOM,A
        B0BSET  FSTART          ; Start transfer and receiving SIO data.
CHK_END:
        B0BTS0  FSTART          ; Wait the end of SIO operation.
        JMP     CHK_END
        B0MOV   A,SIOB          ; Save SIOB data into RXDATA buffer.
        MOV     RXDATA,A
```

11 Flash

11.1 OVERVIEW

The SN8F2270B series USB MCU integrated device feature in-system programmable (ISP) FLASH memory for convenient, upgradeable code storage. The FLASH memory may be programmed via the SONiX 8 bit MCU programming interface or by application code and USB interface for maximum flexibility. The SN8F2270B provides security options at the disposal of the designer to prevent unauthorized access to information stored in FLASH memory.

- The MCU is stalled during Flash write (program) and erase operations, although peripherals (USB, Timers, WDT, I/O, PWM, etc.) remain active.
- Interrupts will disable by firmware during a Flash write or erase operation.
- The Flash page containing the boot loader and code option (ROM address 0x1380 ~ 0x13FF) cannot be erased from application code when the code option's security1 enable.
- Watch dog timer should be clear before the Flash write or erase operation.
- The erase operation sets all the bits in the Flash page to logic 1.
- Hardware will hold system clock and automatically move out data from RAM and do programming, after programming finished, hardware will release system clock and let MCU execute the next instruction.(Recommend add two NOP instructions after this active).

11.2 FLASH PROGRAMMING/ERASE CONTROL REGISTER

0BAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PECMD	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **PECMD[7:0]: 0x5A: Page Program (32 words/page)**, **0xC3: Page Erase (128 words/page)**

11.3 PROGRAMMING/ERASE ADDRESS REGISTER

0BBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEROML	PEROML7	PEROML6	PEROML5	PEROML4	PEROML3	PEROML2	PEROML1	PEROML0
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 [7:0] **PEROML[7:0]:** Define the target starting low byte address [7:0] of Flash memory (10K x 16) which is going to be programmed or erased.

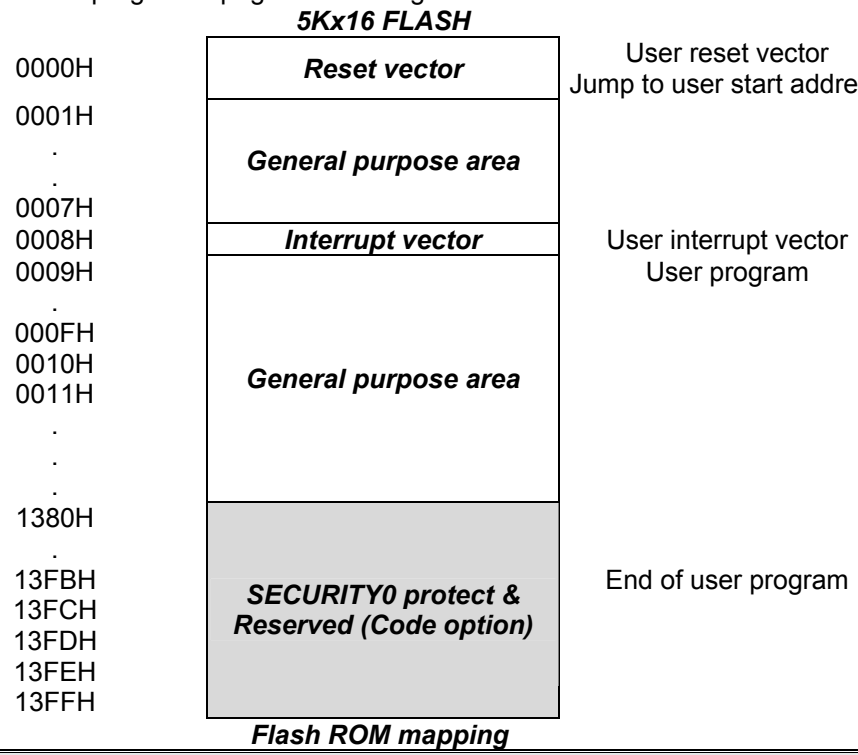
0BCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEROMH	PEROMH7	PEROMH6	PEROMH5	PEROMH4	PEROMH3	PEROMH2	PEROMH1	PEROMH0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **PEROMH [7:0]:** Define the target starting high address [15:8] of Flash memory (10K x 16) which is going to be programmed or erased.

The valid **PAGE ERASE** starting addresses are **0x0, 0x80, 0x100, 0x180, 0x 200, 0x280, 0x300, 0x380 … 0x1380.** The page erase function is used to erase a page of 128 contiguous words in Flash ROM.

Note: If the code option SECURITY0 = 0 (SECURITY1 disable), the code option address 0x13FC ~ 0x13FF will NOT be protected by hardware. And the code option can be "erase and program" by the in-system-programming function. To avoid the error occur, when SECUIRTY0 = 0, please DO NOT set the PAGE ERASE starting address at 0x1380.

The valid **PAGE PROGRAM** starting addresses are **0x0, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, 0xE0 … 0x13E0.** The page program function is used to program a page of 32 contiguous words in Flash ROM.



Flash ROM mapping

Note: 1. If the code option SECURITY0 = 1 (SECURITY enable), the FLASH ROM ADDRESS = 0x1380 ~ 0x13FF will not allow to do the "page erase and page program".
2. If the code option SECURITY0 = 0 (SECURITY disable), the code option address 0x13FC ~ 0x13FF will not be protected by hardware. And the code option can be "erase and program" by the in-system-programming function.

11.4 PROGRAMMING/ERASE DATA REGISTER

0BDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAML	PERAML7	PERAML6	PERAML5	PERAML4	PERAML3	PERAML2	PERAML1	PERAML0
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

0BEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAMCNT	PERAMCNT4	PERAMCNT3	PERAMCNT2	PERAMCNT1	PERAMCNT0	-	-	PERAML8
Read/Write	R/W	R/W	R/W	R/W	R/W	-	-	R/W
After reset	0	0	0	0	0	-	-	0

{PERAMCNT [0], PERAML [7:0]}**:** Define the starting RAM address [8:0], which stores the data wanted to be programmed. **The valid RAM addresses are 00H ~ 07FH.**

PERAMCNT [7:3]**:** Defines the number of words wanted to be programmed. The maximum PERAMCNT [7:3] is 01FH, which program **32 words (64 bytes RAM)** to the Flash. The minimum PERAMCNT [7:3] is 00H, which program only 1 word to the Flash.

Note: 1. If the code option SECURITY0 = 1 (SECURITY enable), the FLASH ROM ADDRESS = 0x1380 ~ 0x13FF will not allow to do the "page erase and page program".
2. If the code option SECURITY0 = 0 (SECURITY disable), the code option address 0x13FC ~ 0x13FF will not be protected by hardware. And the code option can be "erase and program" by the in-system-programming function.

11.4.1 Flash In-system-programming mapping address

RAM (byte)

	bit7 ~ bit0
X	DATA0
X+1	DATA1
X+2	DATA2
X+3	DATA3
…	
X+N	DATAN

=>

Flash ROM (word)

	bit15 ~ bit8	bit7 ~ bit0
Y	DATA1	DATA0
Y+1	DATA3	DATA2
Y+2		
Y+3		
…		
Y+M	DATAN	DATAN-1

12 INSTRUCTION TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I, "M" only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z…)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
ARITHMEIC	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1+N
	ADD	A,M	A (A + M, if occur carry, then C=1, else C=0	√	√	√	1
	ADD	M,A	M (A + M, if occur carry, then C=1, else C=0	√	√	√	1+N
	B0ADD	M,A	M (bank 0) (M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
	ADD	A,I	A (A + I, if occur carry, then C=1, else C=0	√	√	√	1
	SBC	A,M	A (A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
	SBC	M,A	M (A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
	SUB	A,M	A (A - M, if occur borrow, then C=0, else C=1	√	√	√	1
	SUB	M,A	M (A - M, if occur borrow, then C=0, else C=1	√	√	√	1+N
	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
LOGIC	AND	A,M	A ← A and M	-	-	√	1
	AND	M,A	M ← A and M	-	-	√	1+N
	AND	A,I	A ← A and I	-	-	√	1
	OR	A,M	A ← A or M	-	-	√	1
	OR	M,A	M ← A or M	-	-	√	1+N
	OR	A,I	A ← A or I	-	-	√	1
	XOR	A,M	A ← A xor M	-	-	√	1
	XOR	M,A	M ← A xor M	-	-	√	1+N
	XOR	A,I	A ← A xor I	-	-	√	1
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1+N
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1+N
	CLR	M	M ← 0	-	-	-	1
	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
BRANCH	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1+ S
	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1+ S
	DECMS	M	M ← 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 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
MISC	RET		PC ← Stack	-	-	-	2
	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
	PUSH		To push ACC and PFLAG (except NT0, NPD bit) into buffers.	-	-	-	1
	POP		To pop ACC and PFLAG (except NT0, NPD bit) from buffers.	√	√	√	1
	NOP		No operation	-	-	-	1

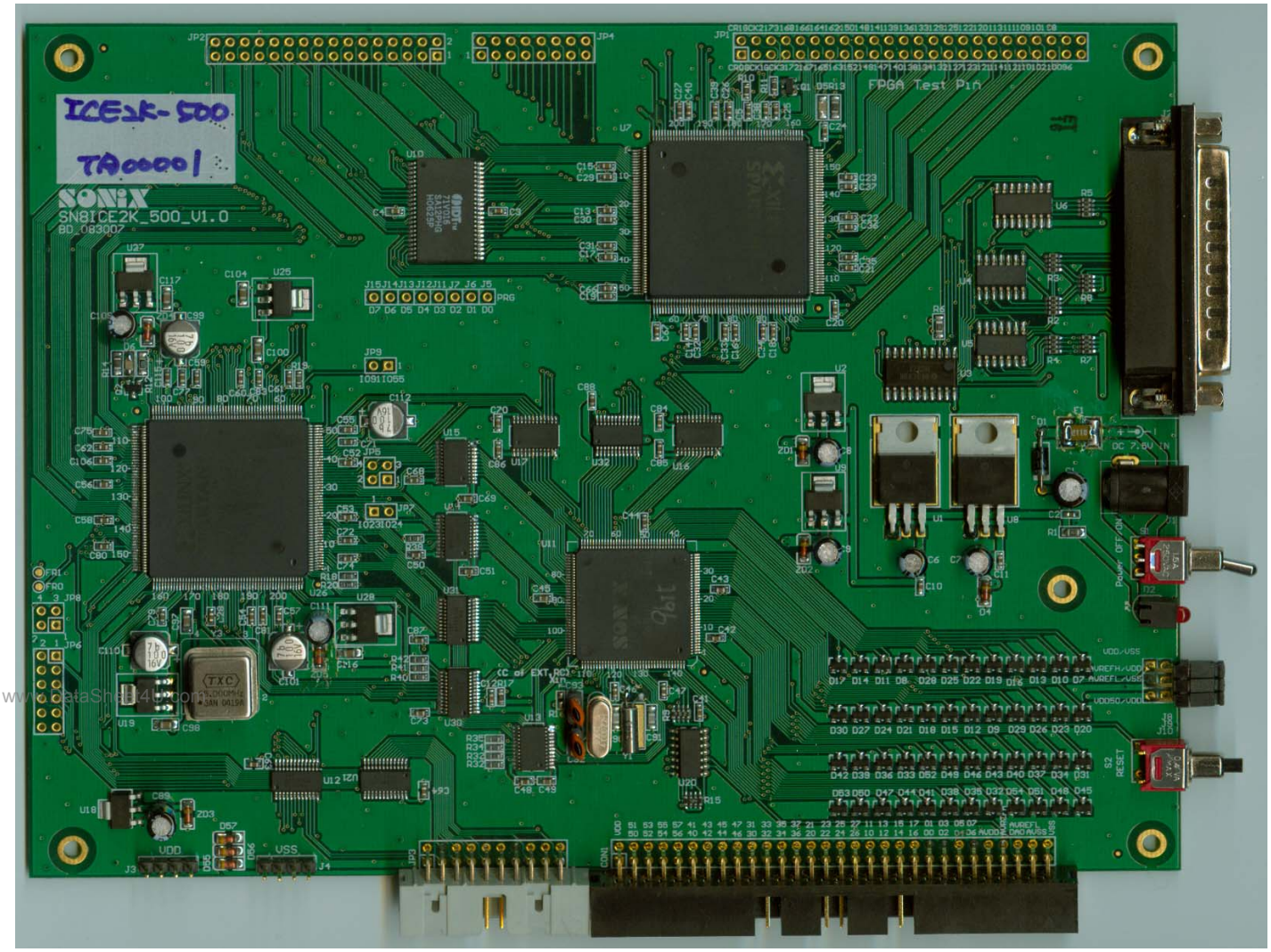
***Note:** 1. "M" is system register or RAM. If "M" is system registers then "N" = 0, otherwise "N" = 1.*
2. If branch condition is true then "S = 1", otherwise "S = 0".

13 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.

13.1 ICE (In Circuit Emulation)

The ICE called “SN8ICE2K Plus”



13.2 SN8F2270B EV-kit

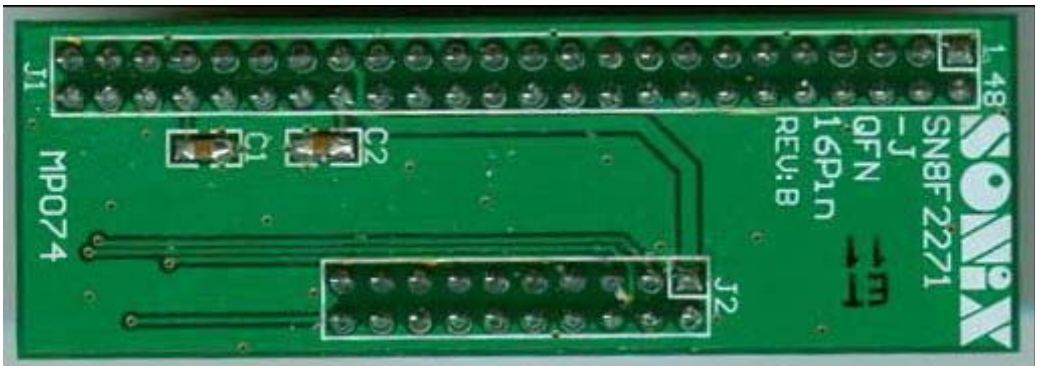
SN8F2270B EV-kit includes ICE interface, GPIO interface, USB interface, and VREG 3.3V power supply.

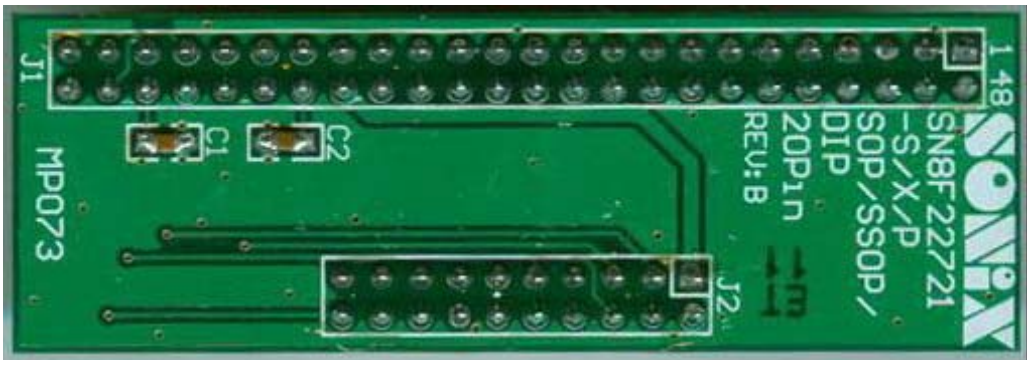
The outline of SN8F2270B EV-kit is as following.

- CON2 : ICE Interface: Interface connected to SN8ICE2K_FSUSB_V2.1
- J2: Jumper to connect between the 5V VDD from SN8ICE2K Plus and VDD_IC on SN8F2271B/ 721B package form socket.
- J1: USB Mini-B connector.
- U4: SN8P2212 to supply 3.3V power for VREG pin and USB PHY.
- U8-U9: SN8F2271B/ 721B connector for user's target board.

13.3 SN8F2270B Transition Board

SN8F2270B Transition Boards includes total 2 models. The following shows the transition board outline for SN8F2271B and SN8F22721B. **Among the board, both C1 and C2 MUST be welded by 1uF capacitor.**





14 ELECTRICAL CHARACTERISTIC

14.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8F22711BS, SN8F2271BJ, SN8F22721BS, SN8F22721BX, SN8F22721BP ..... 0°C ~ + 70°C
Storage ambient temperature (Tstor) ..... –30°C ~ + 125°C

14.2 ELECTRICAL CHARACTERISTIC

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

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd1	Normal mode except USB transmitter specifications, Vpp = Vdd		4.0	5	5.5	V
	Vdd2	USB mode		4.25	5	5.25	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	P1, P5.3 input ports		Vss	-	0.3Vdd	V
	ViL2	P0, P5.0, P5.1, P5.2 input ports		Vss	-	0.2 VREG33	V
Input High Voltage	ViH1	P1, P5.3 input ports		0.7Vdd	-	Vdd	V
	ViH2	P0, P5.0, P5.1, P5.2 input ports		0.8 VREG33	-	VREG33	V
Input Voltage	Vin1	P1, P5.3 I/O port's input voltage range		-0.5	-	Vdd+0.5	V
	Vin2	P0, P5.0, P5.1, P5.2 I/O port's input voltage range		-0.3	-	VREG33 +0.3	V
Output Voltage	Voh1	P1, P5.3 output ports		0	-	Vdd	V
	Voh2	P0, P5.0, P5.1, P5.2 output ports		0	-	VREG33	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port pull-up resistor Rup1	Rup1	P1, P5.3 's Vin = Vss, Vdd = 5V		25	40*	70	KΩ
I/O port pull-up resistor Rup2	Rup2	P0, P5.0, P5.1, P5.2's Vin = Vss, Vdd = 5V		40	60*	80	KΩ
D- pull-up resistor	Rd-	Vdd = 5V, VREG = 3.3V		1.35	1.5	1.65	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O output source current P1, P5.3	IoH1	Vop = Vdd – 1V		15	20*		mA
sink current P1, P5.3	IoL1	Vop = Vss + 0.4V			15*	20	
I/O output source current P0, P5.0~P5.2	IoH2	Vop1 = VREG33 – 1V		1	2*		
sink current P0, P5.0~P5.2	IoL2	Vop1 = Vss + 0.4V			2*	3	
INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Page erase (128 words)	Terase	Flash ROM page erase time		-	25*	TBD	ms
Page program (32 words)	Tpg	Flash ROM page program time (program 32 words)		-	1*	TBD	ms
VREG33 Regulator current	IVREG33	VREG33 Max Regulator Output Current, Vcc > 4.35 volt with 10uF to GND		-	-	60	mA
VREG33 Regulator GND current	Ivreg33 _gnl	No loading. VREG33 pin output 3.3V ((Regulator enable)		-	70	100	uA
VREG25 Regulator GND current	Ivreg25 _gnl	No loading.VREG25 pin output 2.5V ((Regulator enable)		-	120	150	uA
VREG33 Regulator Output voltage	Vreg1	VCC > 4.35V, 0 < temp < 40°C, IVREG ≦ 60 mA with 10uF to GND		3.0	-	3.6	V
	Vreg2	VCC > 4.35V, 0 < temp < 40°C, IVREG ≦ 25 mA with 10uF to GND		3.1	-	3.6	V
Supply Current	Idd1	normal Mode (No loading, Fcpu = Fosc/1)	Vdd= 5V, 6Mhz	-	4	6	mA
	Idd2	Slow Mode (Internal low RC)	Vdd= 5V, 24Khz	-	190	250	uA
	Idd3	Sleep Mode	Vdd= 5V	-	190	250	uA
	Idd4	Green Mode (No loading, Fcpu = Fosc/4 Watchdog Disable)	Vdd= 5V,6Mhz	-	1	2	mA
			Vdd=5V, ILRC 24Khz	-	190	250	uA
LVD Voltage	Vdet	Low voltage reset level.		2.0	2.4	2.9	V

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

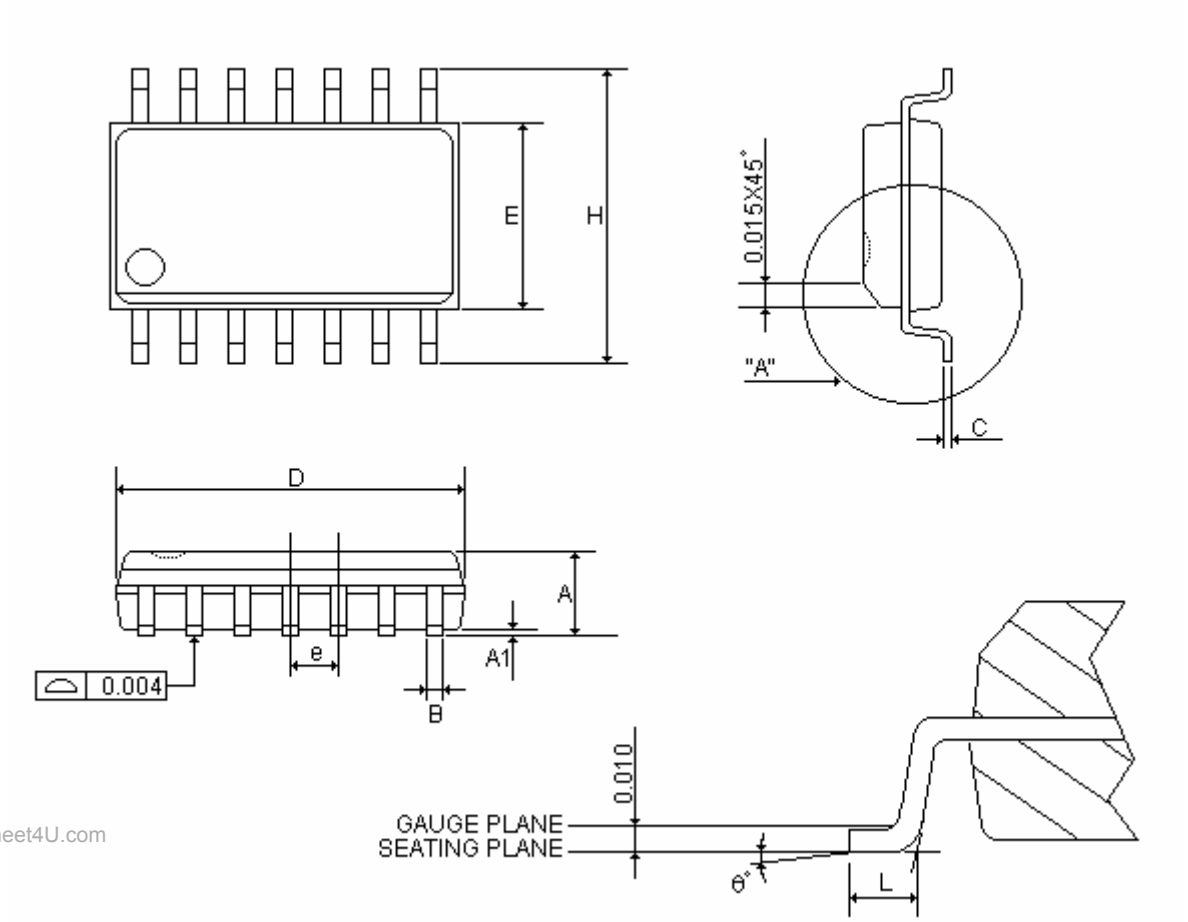
15 Flash ROM PROGRAMMING PIN

Programming Information of SN8F2270B Series											
Chip Name		SN8F2271BJ		SN8F22721BS/X/P		SN8F22711BS					
EZ Writer / MP Writer Connector		Flash IC / JP3 Pin Assigment									
Number	Name	Number	Pin	Number	Pin	Number	Pin	Number	Pin	Number	Pin
1	VDD	1	VDD	18	VDD	3	VDD				
2	GND	13	VSS	14	VSS	13	VSS				
3	CLK	8	P1.2	8	P1.2	8	P1.2				
4	CE										
5	PGM	6	P1.0	6	P1.0	6	P1.0				
6	OE	9	P1.3	9	P1.3	9	P1.3				
7	D1										
8	D0										
9	D3										
10	D2										
11	D5										
12	D4										
13	D7										
14	D6										
15	VDD										
16	-										
17	HLS										
18	RST										
19	-										
20	ALSB/PDB	7, 11	P1.1 P1.4	7, 12	P1.1 P1.4	7, 11	P1.1 P1.4				

Note: Please also check the chapter 13.3 about the description of the SN8F2270B transition boards.

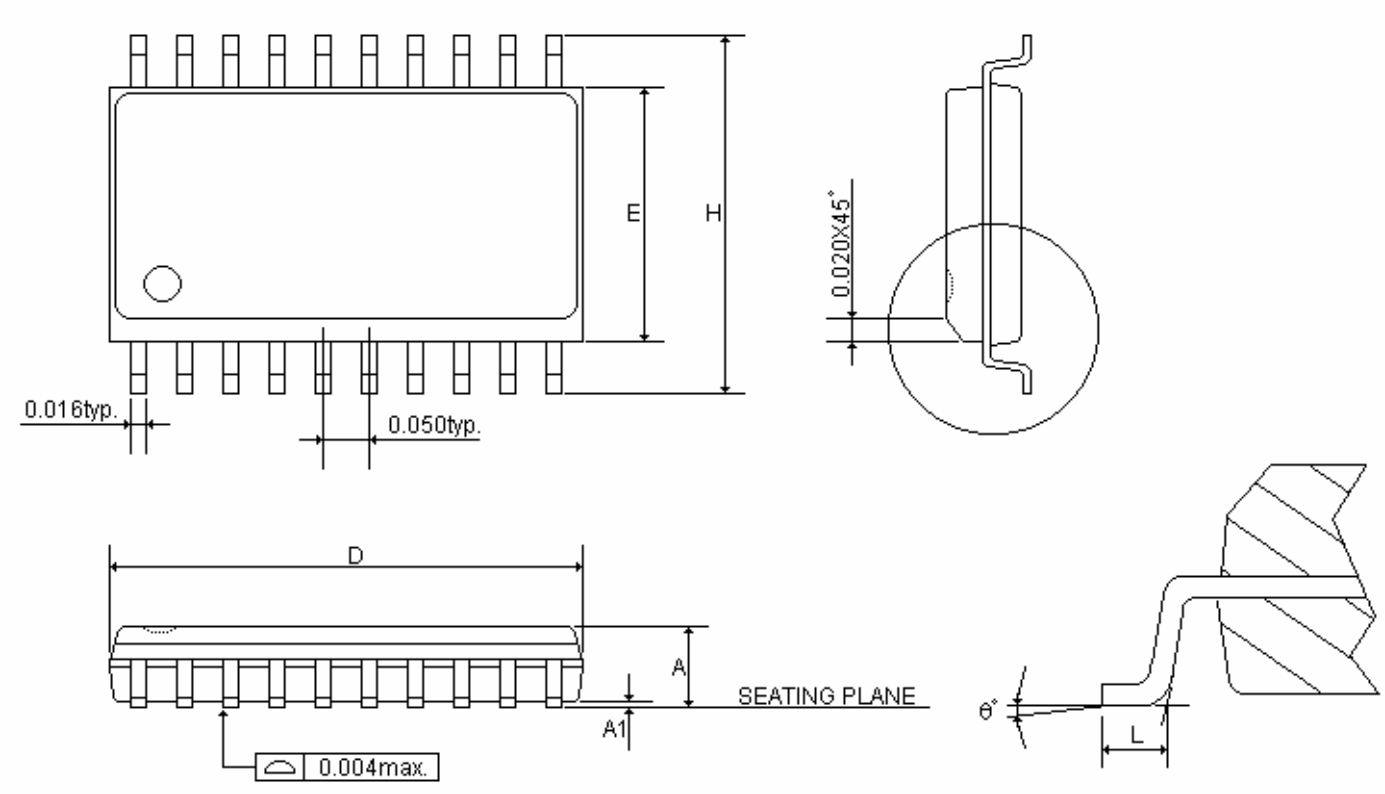
16 PACKAGE INFORMATION

16.1 SOP 14 PIN



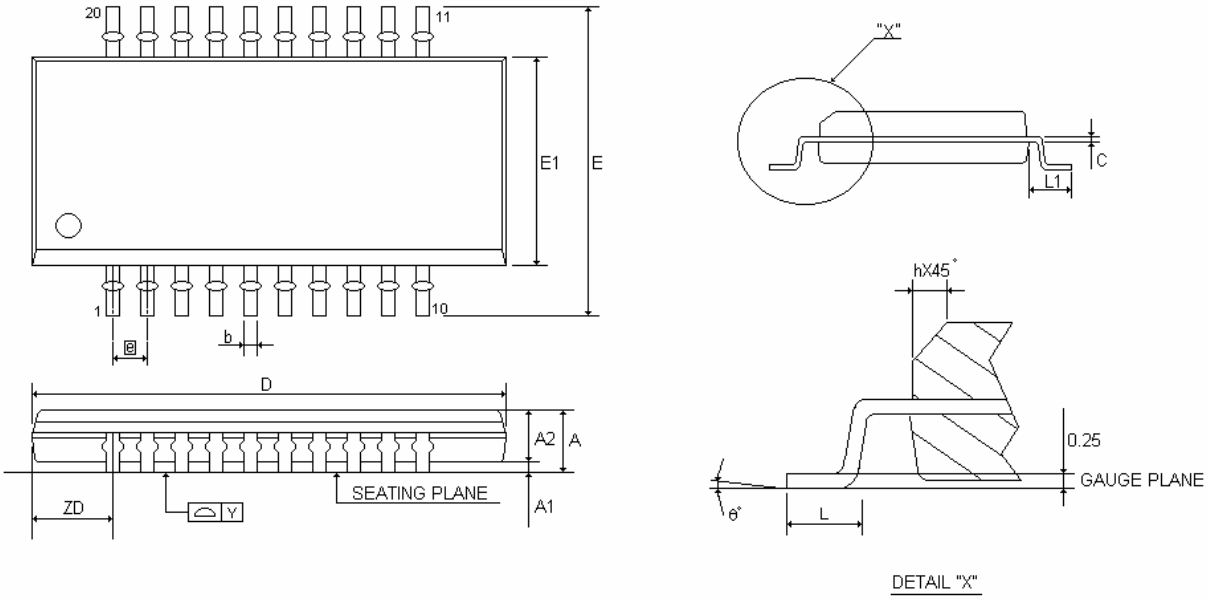
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.058	0.064	0.068	1.4732	1.6256	1.7272
A1	0.004	-	0.010	0.1016	-	0.254
B	0.013	0.016	0.020	0.3302	0.4064	0.508
C	0.0075	0.008	0.0098	0.1905	0.2032	0.2490
D	0.336	0.341	0.344	8.5344	8.6614	8.7376
E	0.150	0.154	0.157	3.81	3.9116	3.9878
e	-	0.050	-	-	1.27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

16.2 SOP 20 PIN



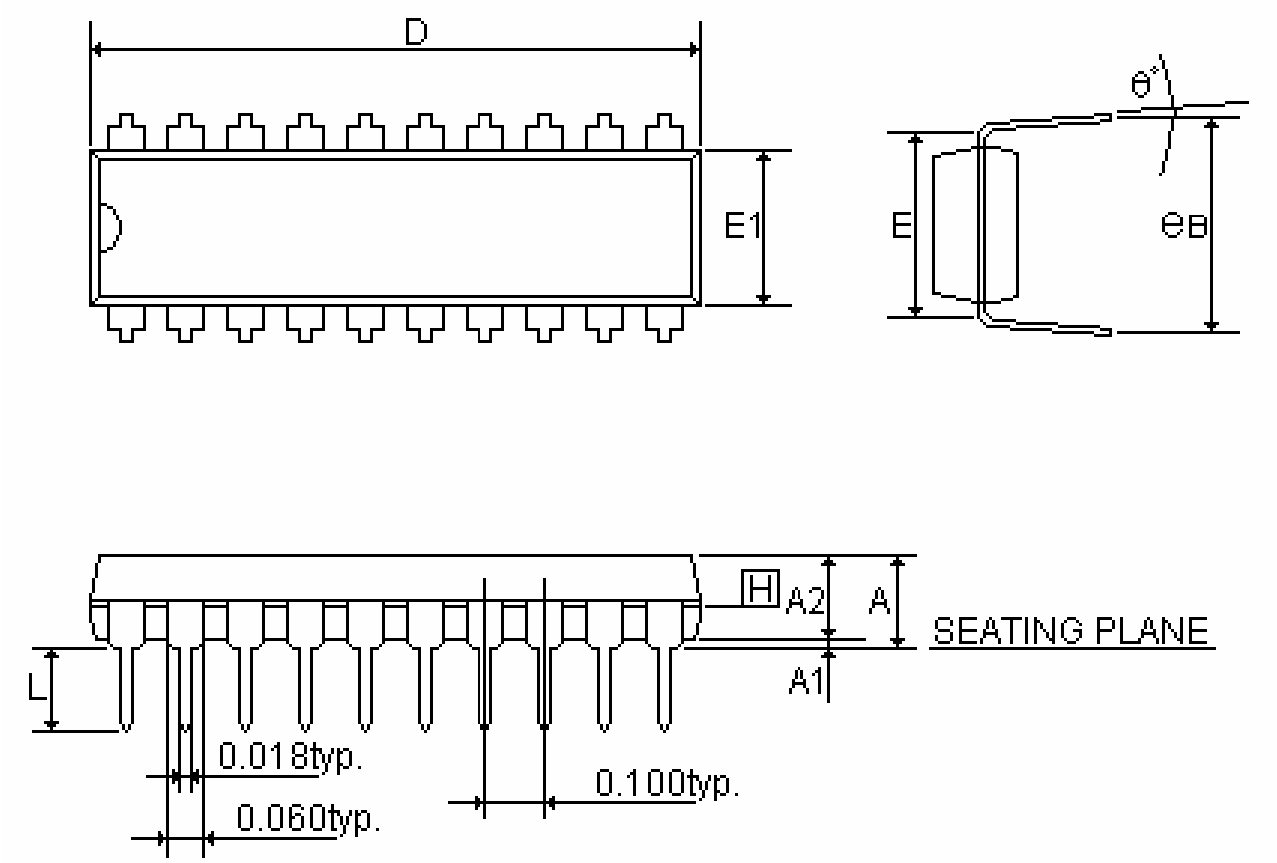
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.496	0.502	0.508	12.598	12.751	12.903
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
θ°	0°	4°	8°	0°	4°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

16.3 SSOP 20 PIN



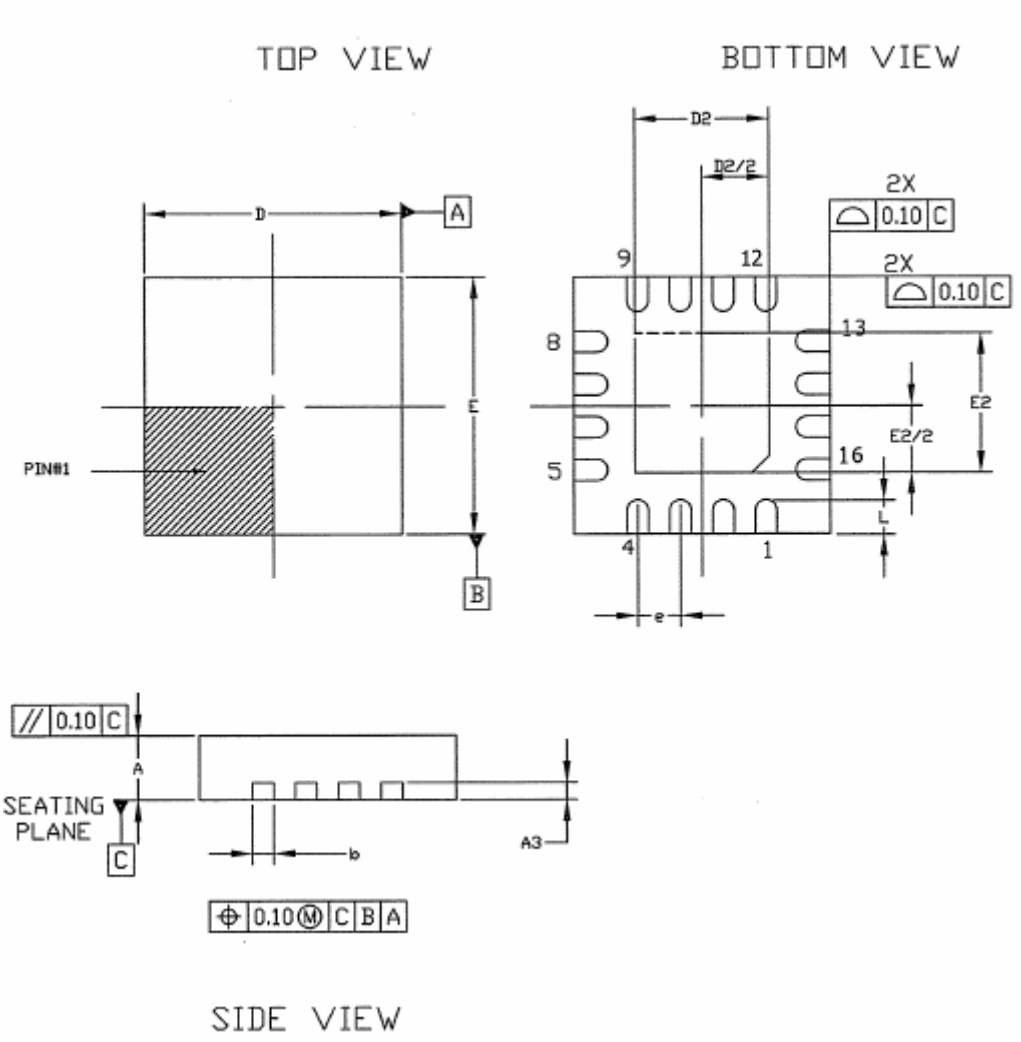
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.053	0.063	0.069	1.350	1.600	1.750
A1	0.004	0.006	0.010	0.100	0.150	0.250
A2	-	-	0.059	-	-	1.500
b	0.008	0.010	0.012	0.200	0.254	0.300
c	0.007	0.008	0.010	0.180	0.203	0.250
D	0.337	0.341	0.344	8.560	8.660	8.740
E	0.228	0.236	0.244	5.800	6.000	6.200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

16.4 P-DIP 20 PIN



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

16.5 QFN 16 PIN



SYMBOL	COMMON					
	DIMENSIONS MILLIMETER			DIMENSIONS INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	SEE VARIATIONS					
A3	0.195	0.203	0.211	0.0077	0.008	0.0083
b	0.18	0.23	0.30	0.007	0.009	0.012
D	2.95	3.0	3.05	0.116	0.118	0.120
E	2.95	3.0	3.05	0.116	0.118	0.120
e	0.50 BSC			0.020 BSC		
L	0.35	0.40	0.45	0.014	0.016	0.018

SYMBOL	VARIATIONS "A"					
	DIMENSIONS MILLIMETER			DIMENSIONS INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
OPTION 1	0.70	0.75	0.80	0.027	0.029	0.031

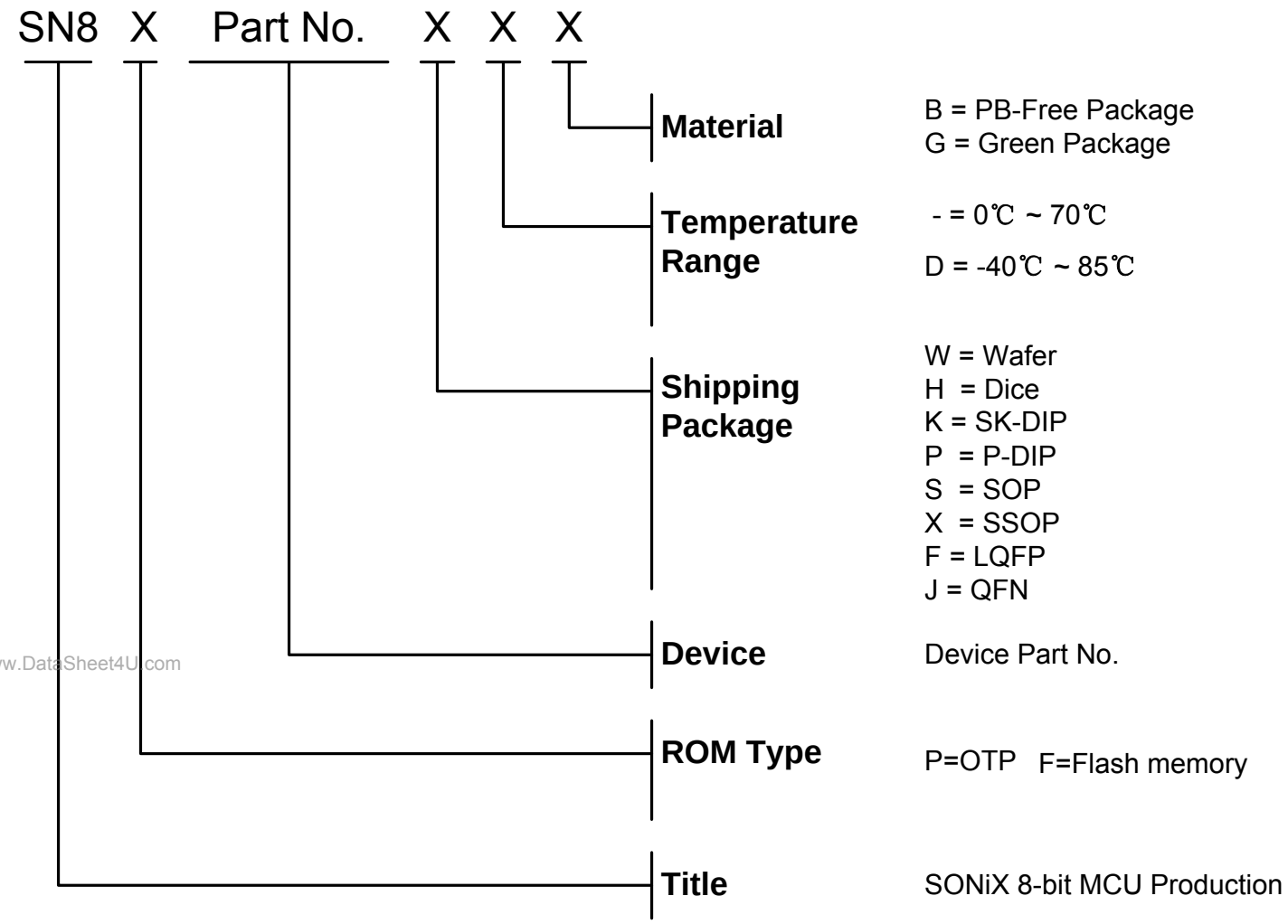
SYMBOL	D2/E2			D2/E2		
	DIMENSIONS MILLIMETER			DIMENSIONS INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
OPTION 1	1.50/1.50	1.625/1.625	1.75/1.75	0.059/0.059	0.064/0.064	0.069/0.069

17 Marking Definition

17.1 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.

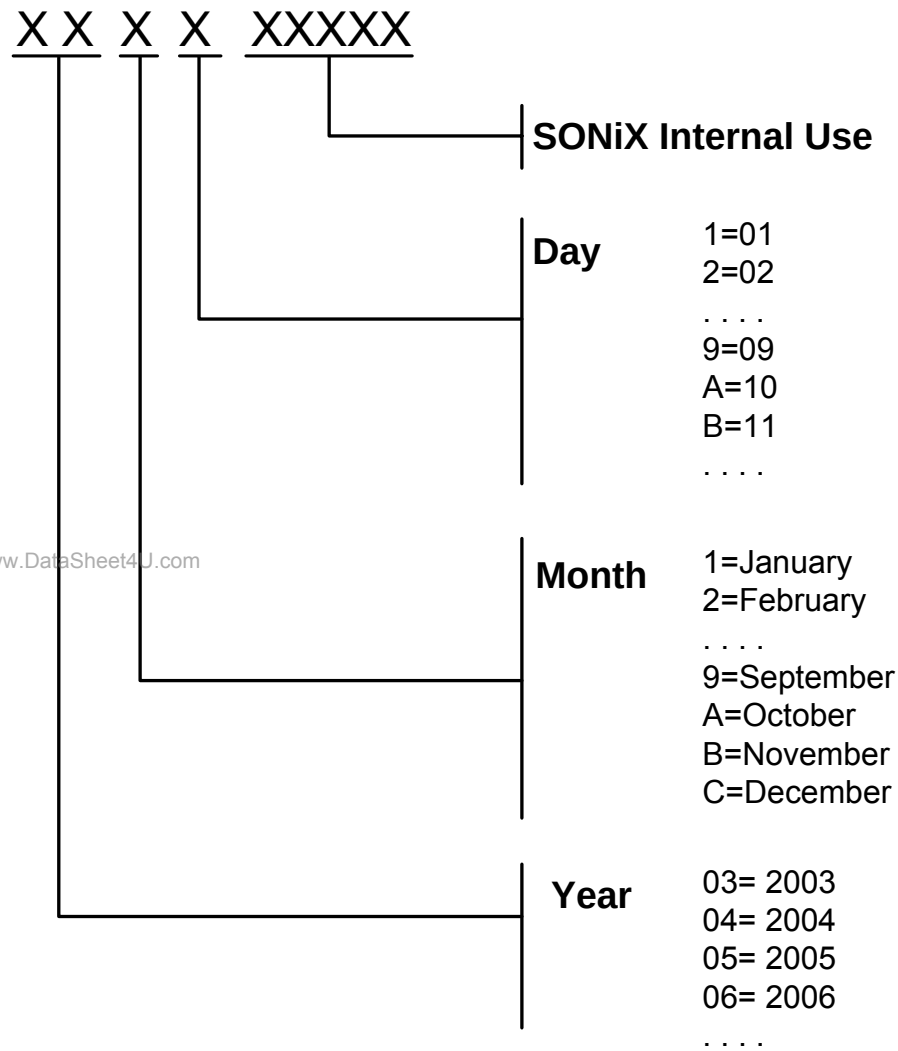
17.2 MARKING INDETIFICATION SYSTEM



17.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8F22721BPB	Flash memory	22721B	SK-DIP	0℃~70℃	PB-Free Package
SN8F22721BSB	Flash memory	22721B	SOP	0℃~70℃	PB-Free Package
SN8F22721BXB	Flash memory	22721B	SSOP	0℃~70℃	PB-Free Package
SN8F22721BPG	Flash memory	22721B	P-DIP	0℃~70℃	Green Package
SN8F22721BSG	Flash memory	22721B	SOP	0℃~70℃	Green Package
SN8F22721BXG	Flash memory	22721B	SSOP	0℃~70℃	Green Package
F2271BJ	Flash memory	2271B	QFN	0℃~70℃	Green Package
SN8F22721BW	Flash memory	22721B	Wafer	0℃~70℃	-
SN8F22721BH	Flash memory	22721B	Dice	0℃~70℃	-

17.4 DATECODE SYSTEM



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