SN8F2280 Series

USER'S MANUAL

SN8F2288
SN8F2283

SONiX 8-Bit Micro-Controller

AMENDMENT HISTORY

Version	Date	Description
VER 1.0	2009/03/06	1. First version is released.
VER 1.1	2009/08/11	1. Remove ADC external high reference voltage source from P40. 2. Modify wakeup time's description. 3. Modify UART baud rate pre-scalar's description. 4. Modify SIO's transfer rate select bit.
VER 1.2	2009/10/20	1. Add SN8F2283 related information. 2. Modify the minimum of Low voltage reset level middle in 17.2 ELECTRONICAL CHARACTERISTIC.
VER 1.3	2009/11/05	1. Update the package information of QFN 24Pin in 19.3 QFN 24 PIN.
VER 1.4	2009/11/16	1. Update Low voltage reset level in 17.2 ELECTRONICAL CHARACTERISTIC. 2. Remove LVD_L selection in 2.1.2 CODE OPTION TABLE. 3. Replace Ext_OSC with High_CLK in 2.1.2 CODE OPTION TABLE. 4. Add information of SN8ICE 2K Plus II and EV-Kit V3.0.
VER 1.5	2010/01/05	1. Update the package information D2 of QFN 48Pin in 19.2 QFN 48 PIN.

Table of Content

1 PRODUCT OVERVIEW

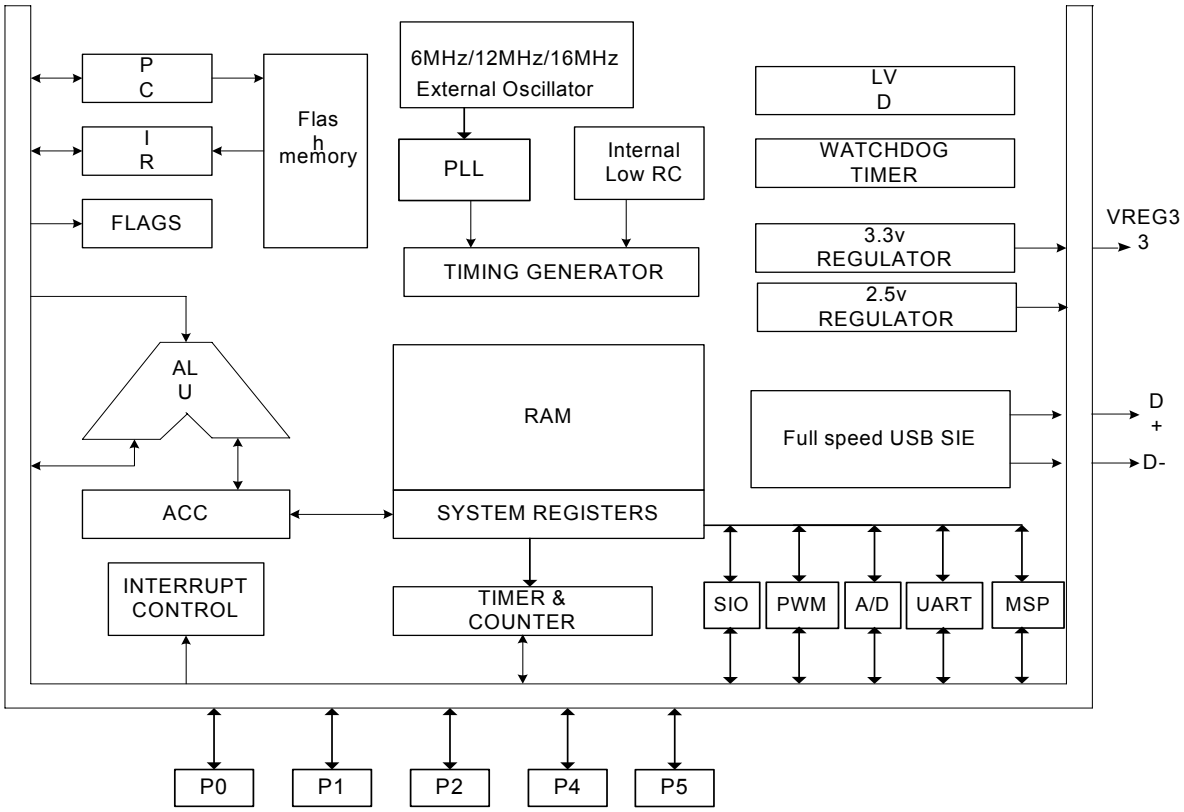
1.1 FEATURES

- **Memory configuration**
  Flash ROM size: 12K x 16 bits, including in system programming function.
  20000 erase/write cycles.
  RAM size: 512 x 8 bits.
- **8 levels stack buffer**
- **I/O pin configuration**
  Bi-directional: P0, P1, P2, P4, P5
  Wakeup: P0/P1 level change.
  P0, P1, P2, P4, P5 with pull up function
  External interrupt: P0, P1 (Level change)
  P0.5, P0.6, P1.0, P1.1 with open drain function.
- **Full Speed USB 2.0**
  Conforms to USB Specification, Version 2.0
  3.3V regulated output/Driving 60mA
  Internal D+ 1.5k ohm pull-up resistor.
  1 control endpoint.
  2 bi-directional INT endpoints
  1 bi-directional INT/BULK IN endpoint.
  1 bi-directional INT/BULK OUT endpoints.
  Programmable EP1~EP4 FIFO depth
- **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)
- **In-system re-programmability**
  Allows easy firmware update
- **On chip watchdog timer.**
- **15 interrupt sources.**
  11 internal interrupts: T0, T1, TC0, TC1, TC2, USB, SIO, I/O pin wakeup, UART, MSP, A/D
  4 external interrupts: INT0, INT1, P0, P1
- **One SIO function for data transfer (Serial Peripheral Interface)**
- **One 8 bits timer counter (T0).**
- **Three 8 bits timer counter (TC0, TC1, TC2)**
  TC0, TC1, TC2. Each has 8 bit PWM function (duty/cycle programmable).
- **One 16 bits timer counter (T1).**
- **One channel UART function**
- **One channel MSP function.**
- **8 channel 12 bit A/D function.**
- **Two system clocks.**
  External high clock: Crystal type 6MHz/12MHz/16MHz
  Internal low clock: RC type 12KHz
- **Four operating modes.**
  Normal mode: Both high and low clock active
  Slow mode: Low clock only
  Sleep mode: Both high and low clock stop
  Green mode: Periodical wakeup by timer
- **Package**
  LQFP48/QFN48/QFN24

☞ **Features Selection Table**

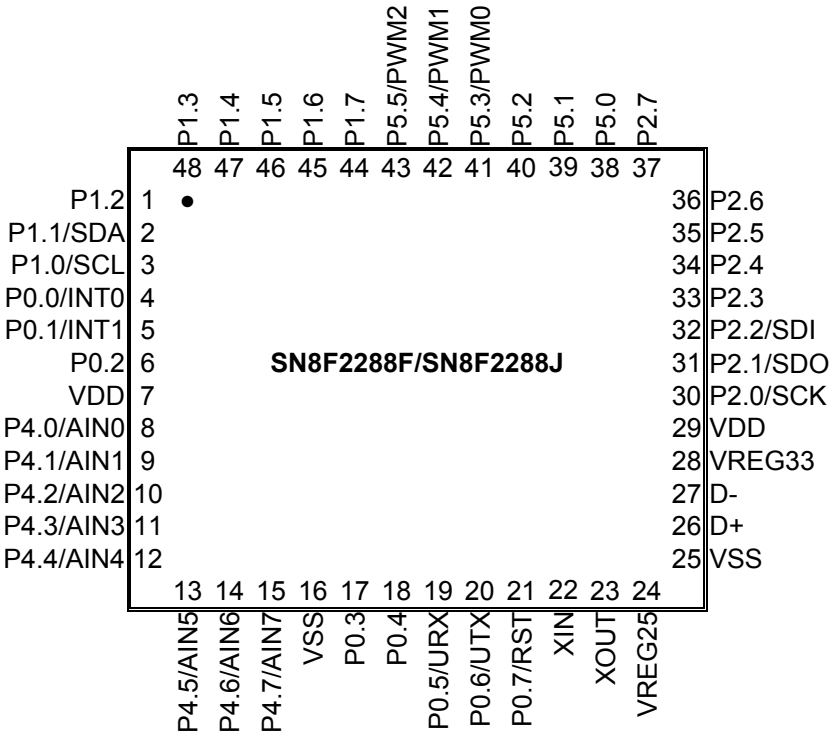
CHIP	ROM	RAM	TIMER					SIO	UART	PWM	A/D	USB	MSP	Wakeup pin.	I/O pin	Package
			T0	T1	TC0	TC1	TC2									
SN8F2288	12K*16	512*8	V	V	V	V	V	V	V	V	12bit	V	V	16	38	LQFP/QFN
SN8F2283	12K*16	512*8	V	V	V	V	V	X	V	X	X	V	X	10	12	QFN

1.2 SYSTEM BLOCK DIAGRAM

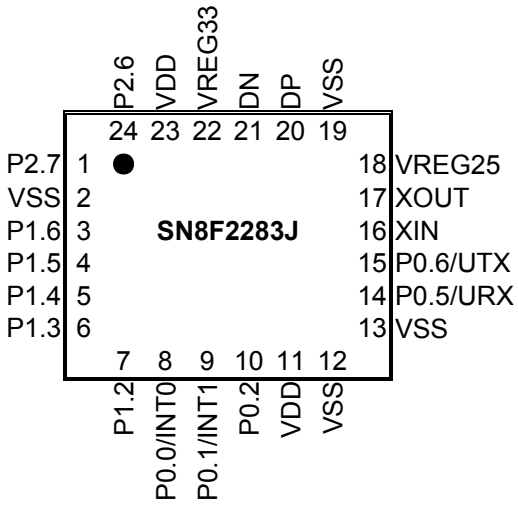


1.3 PIN ASSIGNMENT

SN8F2288F (LQFP 48 pins)
SN8F2288J (QFN 48 pins)



SN8F2283J (QFN 24 pins)



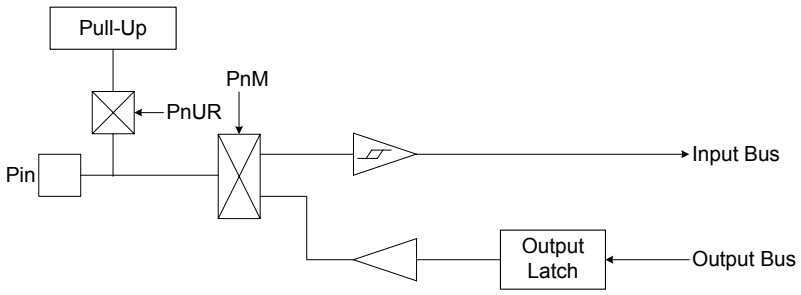
1.4 PIN DESCRIPTIONS

PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital circuit.
P0.0/INT0	I/O	P0.0: Port 0.0 bi-direction pin. Schmitt trigger structure and built-in pull-up 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[4:2]	I/O	P0: Port 0 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. Built wakeup function.
P0.5/URX	I/O	P0.5: Port 0 bi-direction pin Schmitt trigger structure and built-in pull-up resistor as input mode. Built wakeup function. UART function: URX pin Open drain function by control register P1OC
P0.6/UTX	I/O	P0.6: Port 0 bi-direction pin Schmitt trigger structure and built-in pull-up resistor as input mode. Built wakeup function. UART function: UTX pin Open drain function by control register P1OC
P0.7/RST	I/O	RST is system external reset input pin under Ext_RST mode, Schmitt trigger structure, active “low”, and normal stay to “high”. Built wakeup function.
P1.0/SCL	I/O	P1.0: Port 1.0 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. Built wakeup function. Open drain function by control register P1OC MSP Function: SCL function
P1.1/SDA	I/O	P1.1: Port 1.1 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. Built wakeup function. Open drain function by control register P1OC MSP Function: SDA function
P1[7:2]	I/O	P1: Port 1 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. Built wakeup function.
P2.0/SCK	I/O	P2.0: Port 2.0 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. SCK: SIO output clock pin.
P2.1/SDO	I/O	P2.1: Port 2.1 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. SDO: SIO data output pin.
P2.2/SDI	I/O	P2.2: Port 2.2 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. SDI: SIO data input pin.
P2[5:3]	I/O	P2: Port 2 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode.
P4[7:0]/AIN[7:0]	I/O	P4: Port 4 bi-direction pin. Built-in pull-up resisters as input mode. AIN[7:0]: ADC channel – 0~7 input.
P5[2:0]	I/O	P5: Port 5 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode.
P5[5:3]/PWM2~PWM0	I/O	P5: Port 5 bi-direction pin. Schmitt trigger structure and built-in pull-up resisters as input mode. PWM0, PWM1, PWM2: PWM function

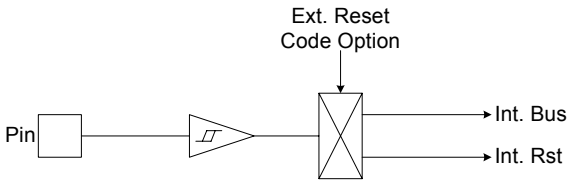
XOUT	I/O	XOUT: Oscillator output pin while external crystal enable.
XIN	I/O	XIN: Oscillator input pin while external oscillator enable (crystal and RC).
VREG25	P	2.5V power pin. Please connect 1uF capacitor to GND.
VREG33	P	3.3V power pin. Please connect XuF capacitor to GND. X=1~10.
D+, D-	I/O	USB differential data line.

1.5 PIN CIRCUIT DIAGRAMS

Port 0, 1, 2, 4, 5 structures:



Port 0.7 structure:

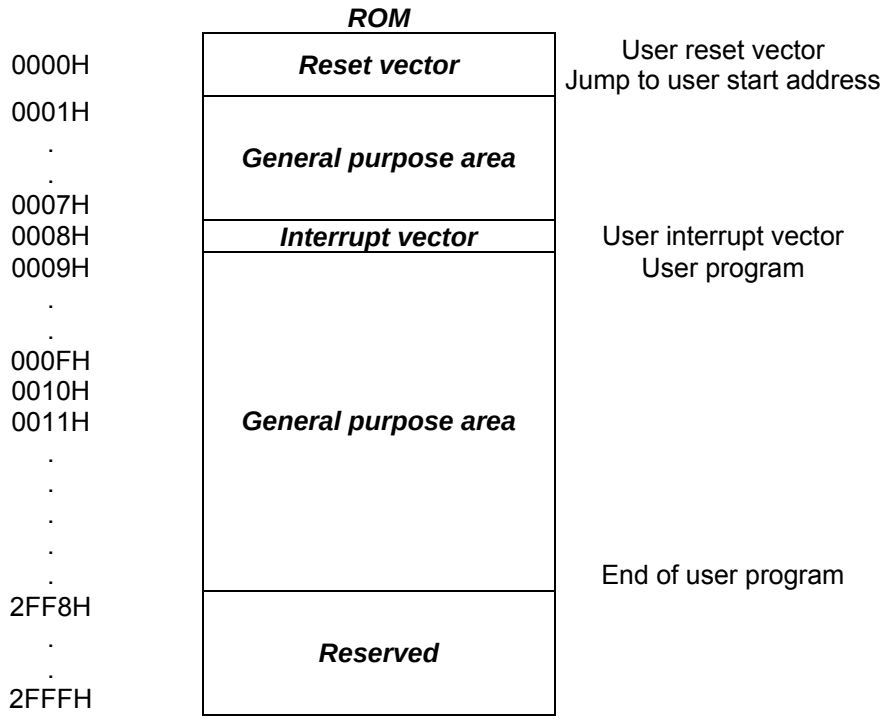


2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ ***12K 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 ROM 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
                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
High_CLK	6M_X'tal	6MHz crystal /resonator for external oscillator.
	12M_X'tal	12MHz crystal /resonator for external oscillator.
	16M_X'tal	16MHz crystal /resonator for external oscillator.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
Fcpu	Fhosc/1	Instruction cycle is **12 MHz clock.**
	Fhosc/2	Instruction cycle is **6 MHz clock.**
	Fhosc/4	Instruction cycle is **3 MHz clock.**
	Fhosc/8	Instruction cycle is **1.5 MHz clock.**
Reset_Pin	Reset	Enable External reset pin.
	P07	Enable P0.7 I/O function.
Fslow	Flosc/2	Slow mode clock = Flosc/2.
	Flosc/4	Slow mode clock = Flosc/4.
Rst_Length	No	No external reset de-bounce time.
	128*ILRC	External reset de-bounce time = 128*ILRC.
LVD	LVD_M	Low Voltage Detect 2.4V.
	LVD_H	Low Voltage Detect 3.6V.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.

2.1.3 DATA MEMORY (RAM)

☞ ***512 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 “ “ “ “ 1FFh	***General purpose area***	BANK1
BANK2	200h “ “ “ “ 27Fh	***General purpose area***	BANK2

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

- *Endpoint 0 support only Control pipe – 8 byte.*
- *Endpoint 1 to Endpoint 4, support Interrupt data transfer, Bulk data transfer – Configurable FIFO depth by setting the USB FIFO control register.*

136 x 8 RAM (FIFO)

00h ~ 07h	***Endpoint 0 RAM (8 byte)***
08h	***Endpoint 1 RAM (W byte)*** ***Interrupt IN/OUT***
	Endpoint 2 RAM (X byte) ***Interrupt IN/OUT***
	Endpoint 3 RAM (Y byte) ***Interrupt IN/OUT*** ***BULK IN/OUT***
	Endpoint 4 RAM (Z byte) ***Interrupt IN/OUT*** ***BULK IN/OUT***

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	TC1M	TC1C	TC1R	TC2M	TC2C
9	TC2R	UDA	USTATUS	EP0OUT_CNT	USB_INT_EN	EP_ACK	EP_NAK	UE0R	UE1R	UE1R_C	UE2R	UE2R_C	UE3R	UE3R_C	UE4R	UE4R_C
A	EP2FIFO_ADDR	EP3FIFO_ADDR	EP4FIFO_ADDR	UDP0	-	UDR0_R	UDR0_W	UPID	UToggle	URTX	URRX	URBRC	URTXD1	URTXD2	URRXD1	URRXD2
B	SIOM	SIOR	SIOB	-	-	P0M	ADM	ADB	ADR	P4CON	PECMD	PEROML	PEROMH	PERAML	PERAMCNT	PEDGE
C	P1W	P1M	P2M	-	P4M	P5M	INTRQ1	INTEN1	INTRQ	INTEN	OSCM	-	WDTR	-	PCL	PCH
D	P0	P1	P2	-	P4	P5	-	-	T0M	T0C	T1M	T1CL	T1CH	-	-	STKP
E	P0UR	P1UR	P2UR	-	P4UR	P5UR	-	@YZ	-	P1OC	MSPSTAT	MSPM1	MSPM2	MSPBUF	MSPADR	-
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.
UEXR_C = EPX byte counter 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.
EP0OUT_CNT = USB endpoint 0 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 = Timer counting register. n = 0, 1, C0, C1, C2
PnUR = Port n pull-up resister control register.
P1W = Port 1 wakeup control register.
PEROM = ISP ROM address.
PERAMCNT = ISP RAM programming counter register.
UTRX = UART RX control register
URXXDX = UART data buffer.
ADB, ADR = A/D converting data buffer.
MSPMX = MSP mode register.

Y, Z = Working, @YZ and ROM addressing register.
RBANK = RAM bank selection register.
UEXR = EPX control registers.
EPXFIFO_ADDR = EPX FIFO start address of USB FIFO.
UDR0_R = USB FIFO read data buffer by UDP0 point to.
EP_NAK = Endpoint NAK flag register.
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, 1, C0, C1, C2
TnR = Tn register. n = C0, C1, C2
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.
URTX = UART TX control regitster
URBRC = UART Baud rate register
ADM = A/D converter mode control register
MSPSTAT = MSP status register
MSPBUF = MSP buffer.
MSPADR = MSP 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							RBNKS1	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
08BH	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT	R/W	TC1M
08CH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
08DH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	R/W	TC1R
08EH	TC2ENB	TC2rate2	TC2rate1	TC2rate0	TC2CKS	ALOAD2	TC2OUT	PWM2OUT	R/W	TC2M
08FH	TC2C7	TC2C6	TC2C5	TC2C4	TC2C3	TC2C2	TC2C1	TC2C0	R/W	TC2C
090H	TC2R7	TC2R6	TC2R5	TC2R4	TC2R3	TC2R2	TC2R1	TC2R0	R/W	TC2R
091H	UDE	UDA6	UDA5	UDA4	UDA3	UDA2	UDA1	UDA0	R/W	UDA
092H	CRCERR	PKTERR	SOF	BUS_RST	SUSPEND	EP0SETUP	EP0IN	EP0OUT	R/W	USTATUS
093H				UEP0OC4	UEP0OC3	UEP0OC2	UEP0OC1	UEP0OC0	R/W	EP0OUT_CNT
094H	REG_EN	DP_PU_EN	SOF_INT _EN		EP4NAK _INT_EN	EP3NAK _INT_EN	EP2NAK _INT_EN	EP1NAK _INT_EN	R/W	USB_INT_EN
095H					EP4_ACK	EP3_ACK	EP2_ACK	EP1_ACK	R/W	EP_ACK
096H					EP4_NAK	EP3_NAK	EP2_NAK	EP1_NAK	R/W	EP_NAK
097H		UE0M1	UE0M0		UE0C3	UE0C2	UE0C1	UE0C0	R/W	UE0R
098H	UE1E	UE1M1	UE1M0						R/W	UE1R
099H		UE1C6	UE1C5	UE1C4	UE1C3	UE1C2	UE1C1	UE1C0	RW	UE1R_C
09AH	UE2E	UE2M1	UE2M0						R/W	UE2R
09BH		UE2C6	UE2C5	UE2C4	UE2C3	UE2C2	UE2C1	UE2C0	RW	UE2R_C
09CH	UE3E	UE3M1	UE3M0						R/W	UE3R
09DH		UE3C6	UE3C5	UE3C4	UE3C3	UE3C2	UE3C1	UE3C0	RW	UE3R_C
09EH	UE4E	UE4M1	UE4M0						R/W	UE4R
09FH		UE4C6	UE4C5	UE4C4	UE4C3	UE4C2	UE4C1	UE4C0	RW	UE4R_C
0A0H	EP2FIFO7	EP2FIFO6	EP2FIFO5	EP2FIFO4	EP2FIFO3	EP2FIFO2	EP2FIFO1	EP2FIFO0	R/W	EP2FIFO_ADDR
0A1H	EP3FIFO7	EP3FIFO6	EP3FIFO5	EP3FIFO4	EP3FIFO3	EP3FIFO2	EP3FIFO1	EP3FIFO0	R/W	EP3FIFO_ADDR
0A2H	EP4FIFO7	EP4FIFO6	EP4FIFO5	EP4FIFO4	EP4FIFO3	EP4FIFO2	EP4FIFO1	EP4FIFO0	R/W	EP4FIFO_ADDR
0A3H	UDP07	UDP06	UDP05	UDP04	UDP03	UDP02	UDP01	UDP00	R/W	UDP0
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						UBDE	DDP	DDN	R/W	UPID
0A8H					EP4_DATA 01	EP3_DATA 01	EP2_DATA 01	EP1_DATA 01	R/W	Utoggle
0A9H	UCLKS			UTXEN	UTXPEN	UTXPS	UTXM		R/W	URTX
0AAH	URXEN	URXS1	URXS0	URXPEN	URXPS	URXPC	URXM		R/W	URRX
0ABH	UDIV4	UDIV3	UDIV2	UDIV1	UDIV0	UPCS2	UPCS1	UPCS0	R/W	URBRC
0ACH	UTXD17	UTXD16	UTXD15	UTXD14	UTXD13	UTXD12	UTXD11	UTXD10	R/W	URTXD1
0ADH	UTXD27	UTXD26	UTXD25	UTXD24	UTXD23	UTXD22	UTXD21	UTXD20	R/W	URTXD2
0AEH	URXD17	URXD16	URXD15	URXD14	URXD13	URXD12	URXD11	URXD10	R	URRXD1
0AFH	URXD27	URXD26	URXD25	URXD24	URXD23	URXD22	URXD21	URXD20	R	URRXD2
0B0H	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA	R/W	SIOM
0B1H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	W	SIOR
0B2H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB
0B5H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0B6H	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0	R/W	ADM
0B7H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R/W	ADB
0B8H	ADCKS2	ADCKS1	ADCKS0	ADLEN	ADB3	ADB2	ADB1	ADB0	RW	ADR
0B9H	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	R/W	P4CON
0BAH	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0	W	PECMD
0BBH	PEROML7	PEROML6	PEROML5	PEROML4	PEROML3	PEROML2	PEROML1	PEROML0	R/W	PEROML
0BCH	PEROMH7	PEROMH6	PEROMH5	PEROMH4	PEROMH3	PEROMH2	PEROMH1	PEROMH0	R/W	PEROMH
0BDH	PERAML7	PERAML6	PERAML5	PERAML4	PERAML3	PERAML2	PERAML1	PERAML0	R/W	PERAML
0BEH	PERAMCNT 4	PERAMCNT 3	PERAMCNT 2	PERAMCNT 1	PERAMCNT 0		PERAML9	PERAML8	R/W	PERAMCNT
0BFH					P01G1	P01G0	P00G1	P00G0	R/W	PEDGE
0C0H	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	R/W	P1W
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C5H			P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C6H	P1IRQ	P0IRQ	MSPIRQ	UTRXIRQ	UTTXIRQ	T2CIRQ	TC1IRQ	TC0IRQ	R/W	INTRQ1
0C7H	P1IEN	P0IEN	MSPIEN	UTRXIEN	UTTXIEN	TC2IEN	TC1IEN	TC0IEN	R/W	INTEN1
0C8H	ADCIRQ	USBIRQ	T1IRQ	T0IRQ	SIOIRQ	WAKEIRQ	P01IRQ	P00IRQ	R/W	INTRQ

0C9H	ADCIEN	USBIEN	T1IEN	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			PC13	PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D2H	P27	P26	P25	P24	P23	P22	P21	P10	R/W	P2
0D4H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4
0D5H			P55	P54	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
0DAH	T1ENB	T1rate2	T1rate1	T1rate0					R/W	T1M
0DBH	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0	R/W	T1CL
0DCH	T1C15	T1C14	T1C13	T1C12	T1C11	T1C10	T1C9	T1C8	R/W	T1CH
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2UR
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H			P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E9H					P06OC	P05OC	P11OC	P10OC	R/W	P1OC
0EAH		CKE	D_A	P	S	RED_WRT		BF	R	MSPSTAT
0EBH	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK		MSPC	R/W	MSPM1
0ECH	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	R/W	MSPM2
0EDH	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0	R/W	MSPBUF
0EEH	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0	R/W	MSPADR
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H			S7PC13	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H			S6PC13	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H			S5PC13	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H			S4PC13	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H			S3PC13	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH			S2PC13	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH			S1PC13	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH			S0PC13	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 14-bit binary counter separated into the high-byte 6 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

Besides, it can be replaced with specific address by executing CALL or JMP instruction. When JMP or CALL instruction is executed, the destination address will be inserted to bit 0 ~ bit 13.

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

☞ **ONE ADDRESS SKIPPING**

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

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

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

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

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

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

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

INCS instruction:

```
        INCS    BUF0
        JMP     C0STEP      ; Jump to C0STEP if ACC is not zero.
        …
        …
C0STEP: NOP
```

INCMS instruction:

```
        INCMS   BUF0
        JMP     C0STEP      ; Jump to C0STEP if BUF0 is not zero.
        …
        …
C0STEP: NOP
```

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

DECS instruction:

```
        DECS    BUF0
        JMP     C0STEP      ; Jump to C0STEP if ACC is not zero.
        …
        …
C0STEP: NOP
```

DECMS instruction:

```
        DECMS   BUF0
        JMP     C0STEP      ; Jump to C0STEP if BUF0 is not zero.
        …
        …
C0STEP: NOP
```

☞ MULTI-ADDRESS JUMPING

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

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

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

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

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

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

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

2.1.4.7 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).

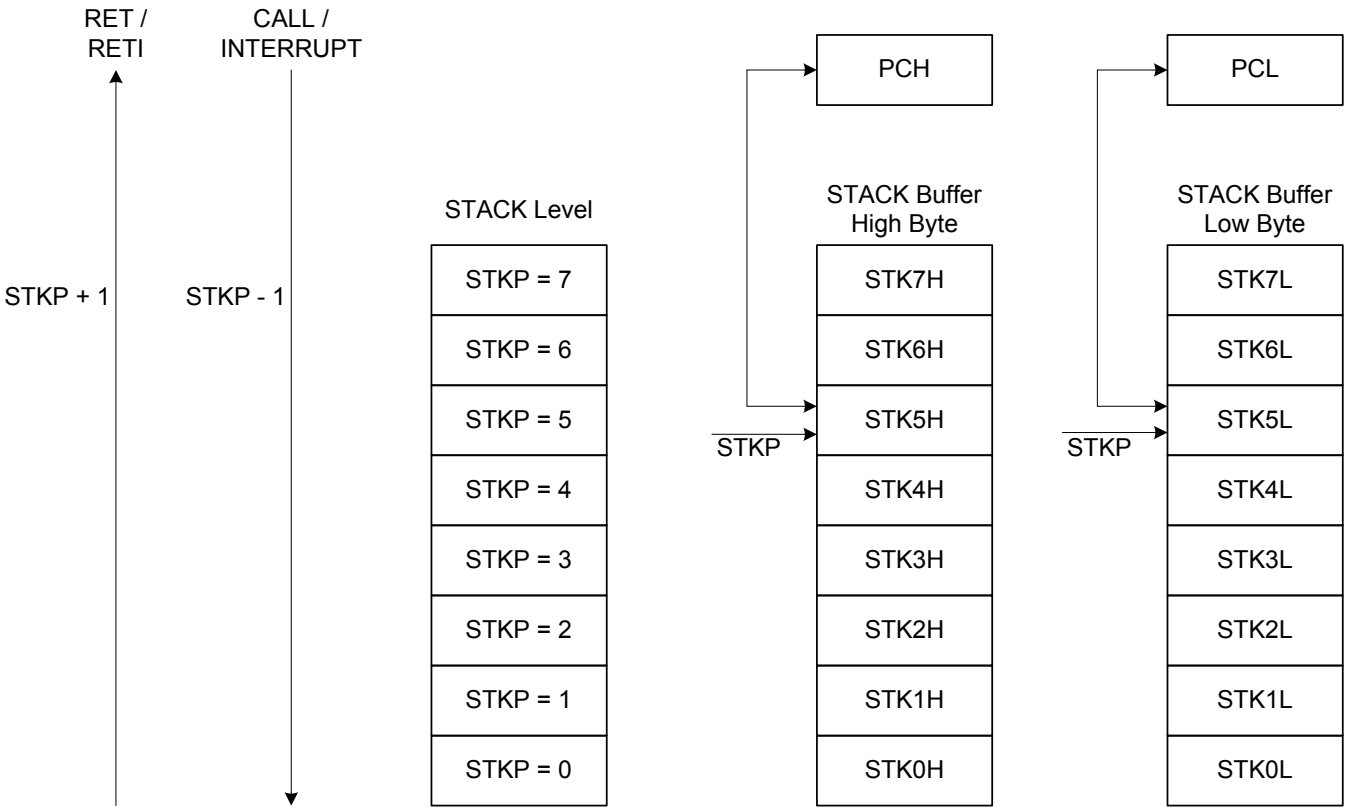
- **Example: Indirectly addressing mode with @YZ register.**

```
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, 14-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

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

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

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

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

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

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

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

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

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

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

2.3.3 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)

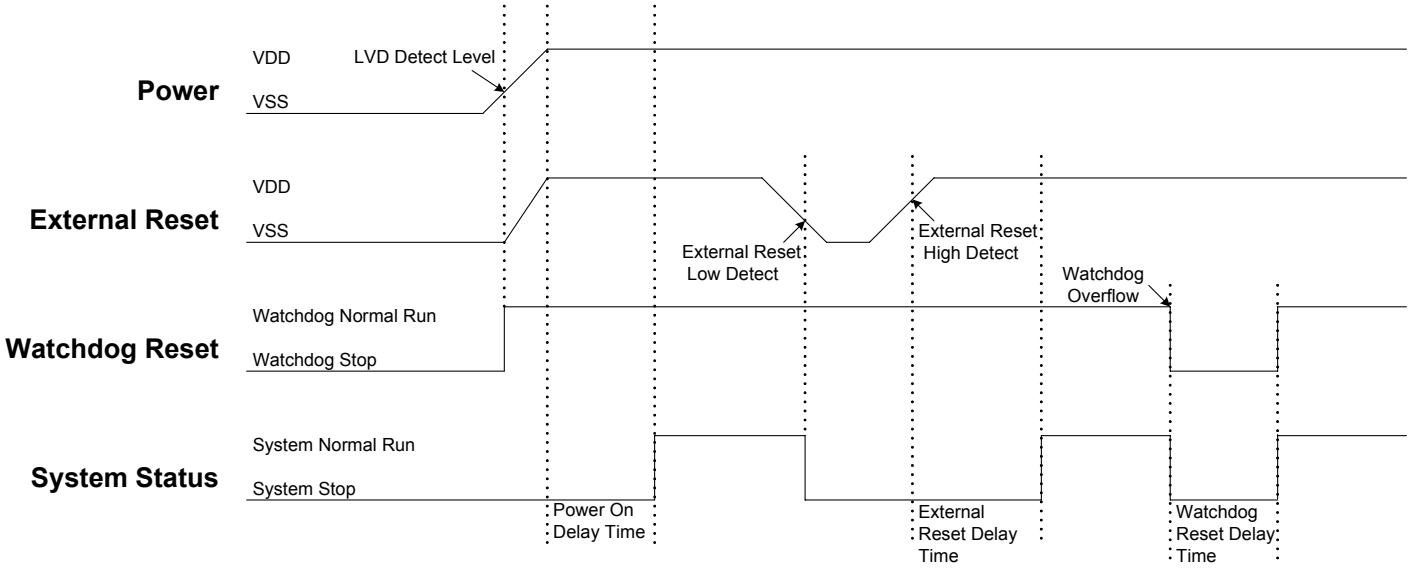
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

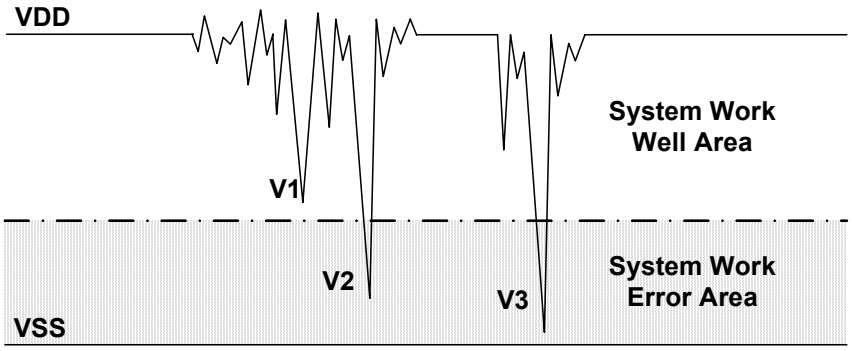
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

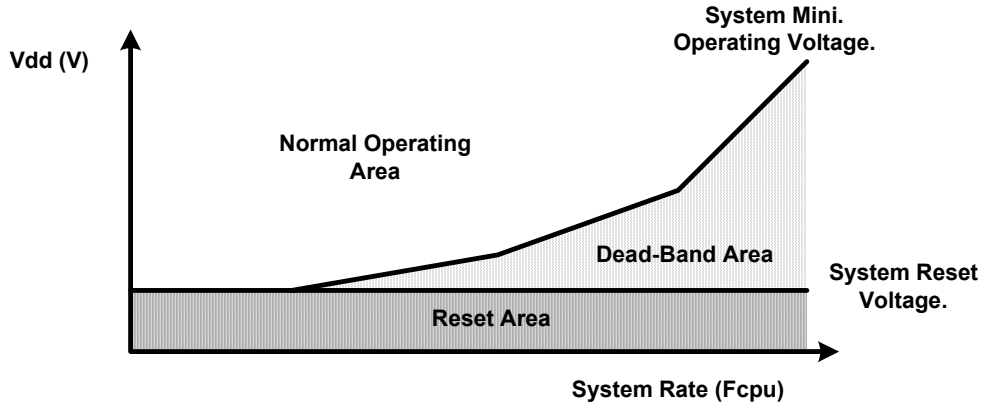
AC application:

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

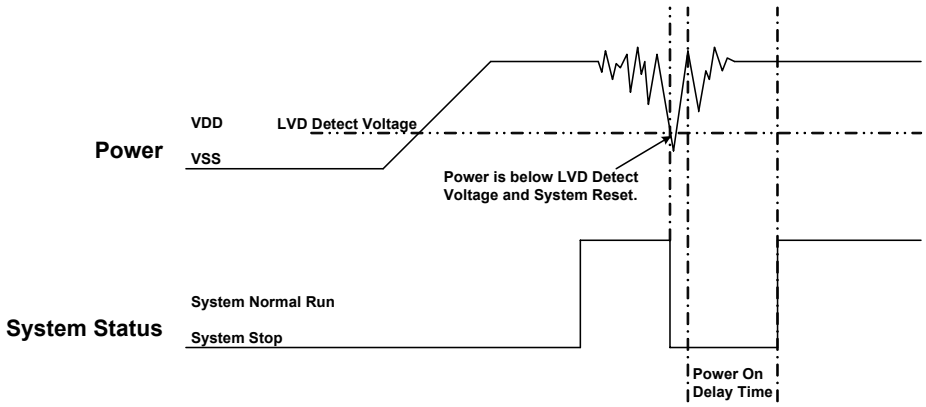
How to improve the brown reset condition? There are some methods to improve brown out reset as following.

- **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

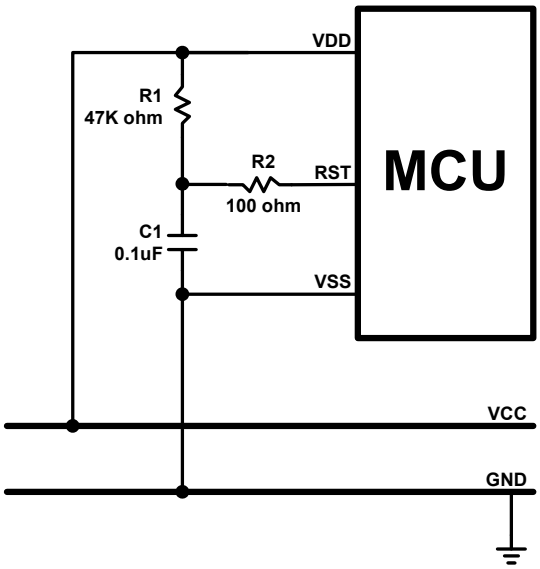
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

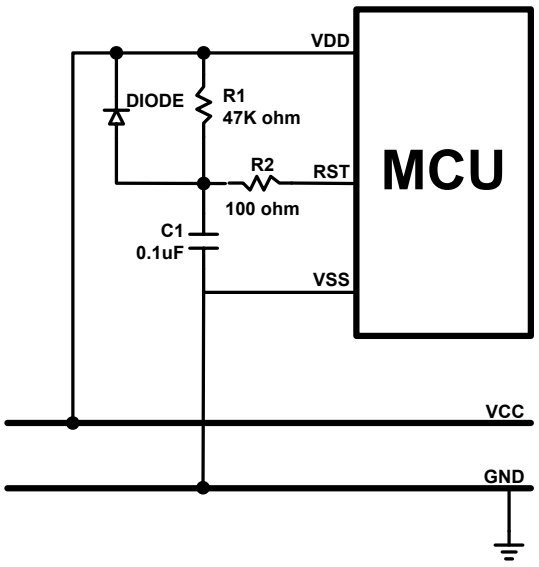
3.6.1 Simply RC Reset Circuit



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.***

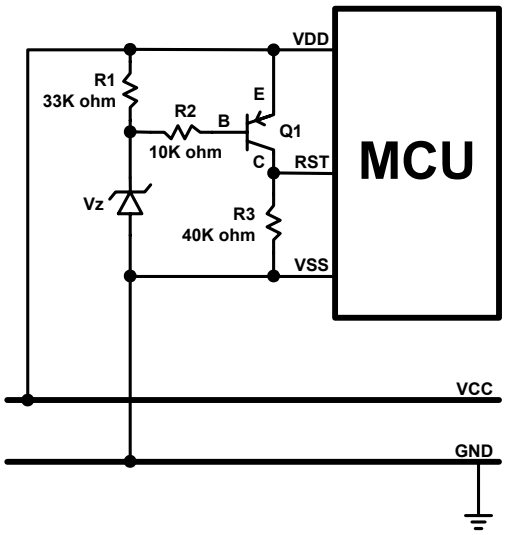
3.6.2 Diode & RC Reset Circuit



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.

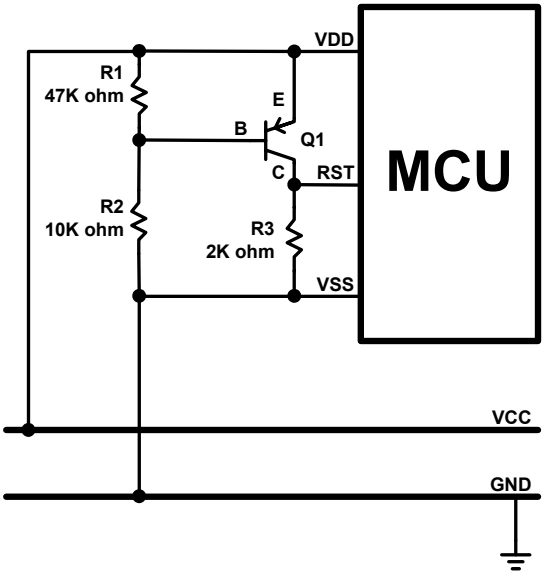
✸ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

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

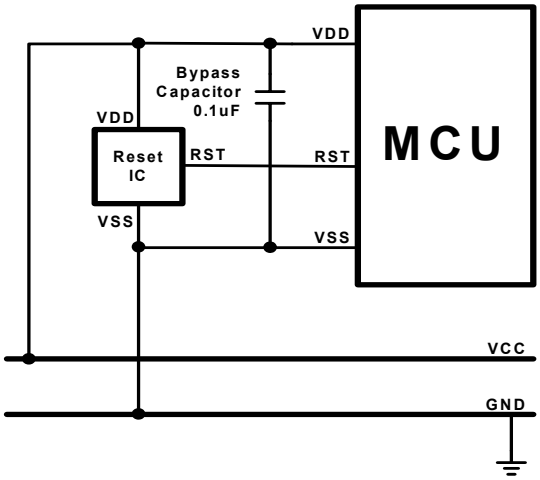


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to "0.7V x (R1 + R2) / R1", 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 reset 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 12 KHz).

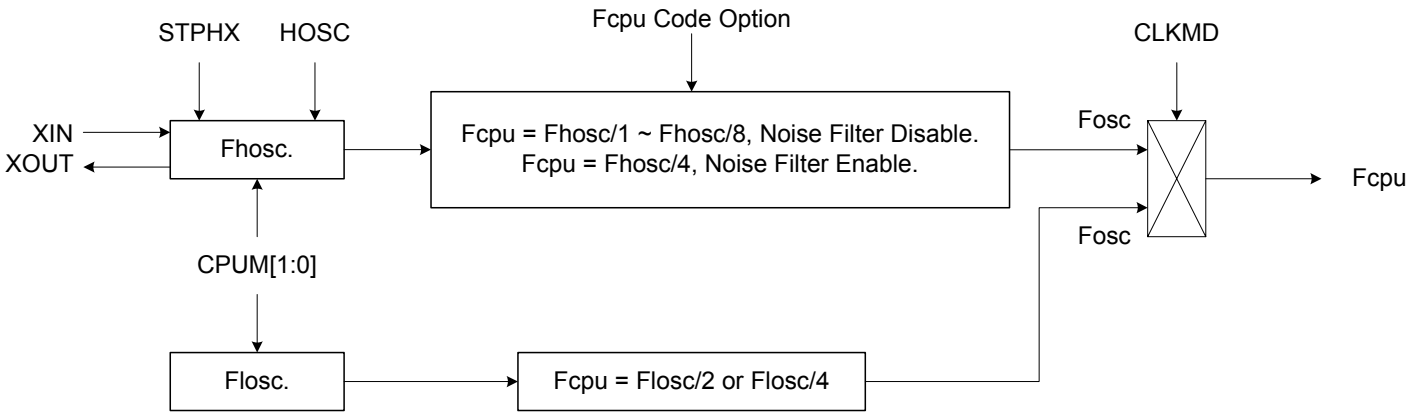
Both the high-speed clock and the low-speed clock can be system clock (Fosc). The system clock in slow mode is divided by 2 or 4 to be the instruction cycle (Fcpu).

☞ **Normal Mode (High Clock):** **Fcpu = Fhosc / N**, N = 1 ~ 8, Select N by Fcpu code option.

☞ **Slow Mode (Low Clock):** **Fcpu = Flosc/N,** N = 2 or 4, Select N by code option.

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: External high-speed clock.
- Flosc: Internal low-speed RC clock (Typical 12 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[4:3] **CPUM[1:0]:** CPU operating mode control bits.
00 = normal.
01 = sleep (power down) mode.
10 = green mode.
11 = reserved.

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 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.

➢ **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).
```

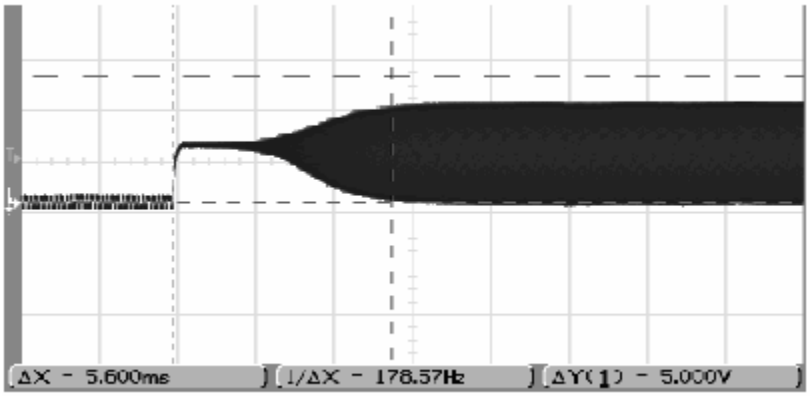
4.4 SYSTEM HIGH CLOCK

The system high clock is from the in circuit PLL. User must select the external oscillator 6MHz X'tal, 12MHz X'tal or 16MHz X'tal by the code option "Ext_OSC", and all the three clock source will input to the on-chip PLL circuit. PLL will output 12MHz to system clock (Fosc).

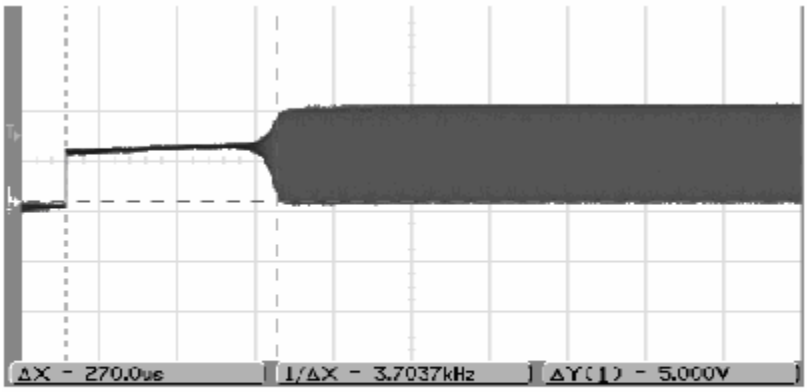
4.4.1 EXTERNAL HIGH CLOCK

External high clock includes three modules (Crystal/Ceramic and external clock signal). The start up time of crystal and ceramic oscillator is different. The oscillator start-up time decides reset time length.

4MHz Crystal

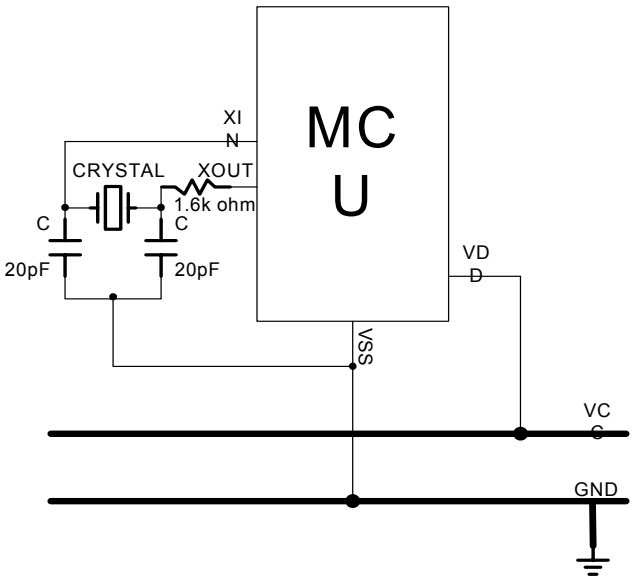


4MHz Ceramic



4.4.1.1 CRYSTAL/CERAMIC

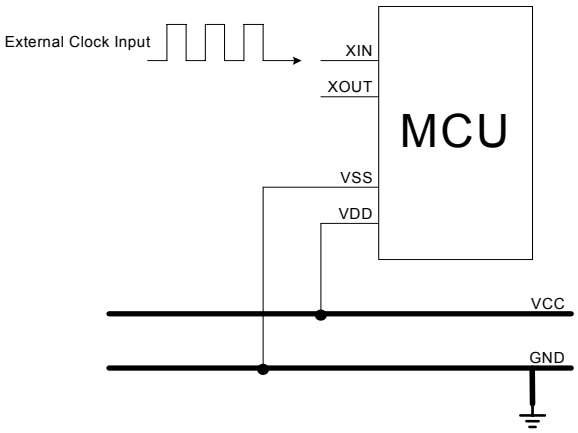
Crystal/Ceramic devices are driven by XIN, XOUT pins.



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

4.4.1.2 EXTERNAL CLOCK SIGNAL

Selecting external clock signal input to be the input clock source is by the “Ext_OSC” code option. The external clock signal is input from XIN pin. XOUT pin is general purpose I/O pin.



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

4.5 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 12KHz.

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

☞ ***Flosc = Internal low RC oscillator (12KHz).***

☞ ***Slow mode Fcpu = Flosc / N. N = 2 or 4 set by code option.***

There are two conditions to stop internal low RC. One is power down mode, and the other is green mode of 12K mode and watchdog disable. If system is in 12K mode and watchdog disable, only 12K 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 (12K, 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
```

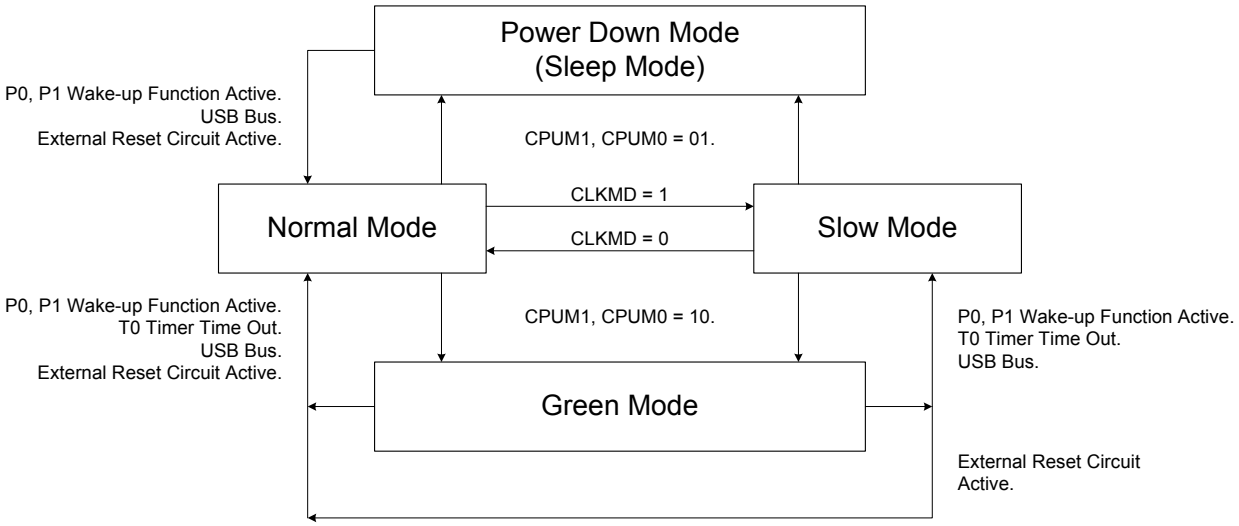
✸ ***Note: Do not measure the RC frequency directly from XIN; the probe impendence will affect the RC frequency.***

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
HOSC	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
T1 timer	*Active	*Active	Inactive	Inactive	* Active if T1ENB=1
TC0 timer	*Active	*Active	Inactive	Inactive	* Active if TC0ENB=1
TC1 timer	*Active	*Active	Inactive	Inactive	* Active if TC1ENB=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	

- **HOSC:** high clock (Fosc = 12MHz)
- **ILRC:** Internal low clock (12KHz 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, #10          ; internal RC=12KHz (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 16384 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.

"16M_X'tal/12M_X'tal/6M_X'tal" mode:

The Wakeup time = 1/Fosc * 16384 (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 16M_X'tal/12M_X'tal/6M_X'tal 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.

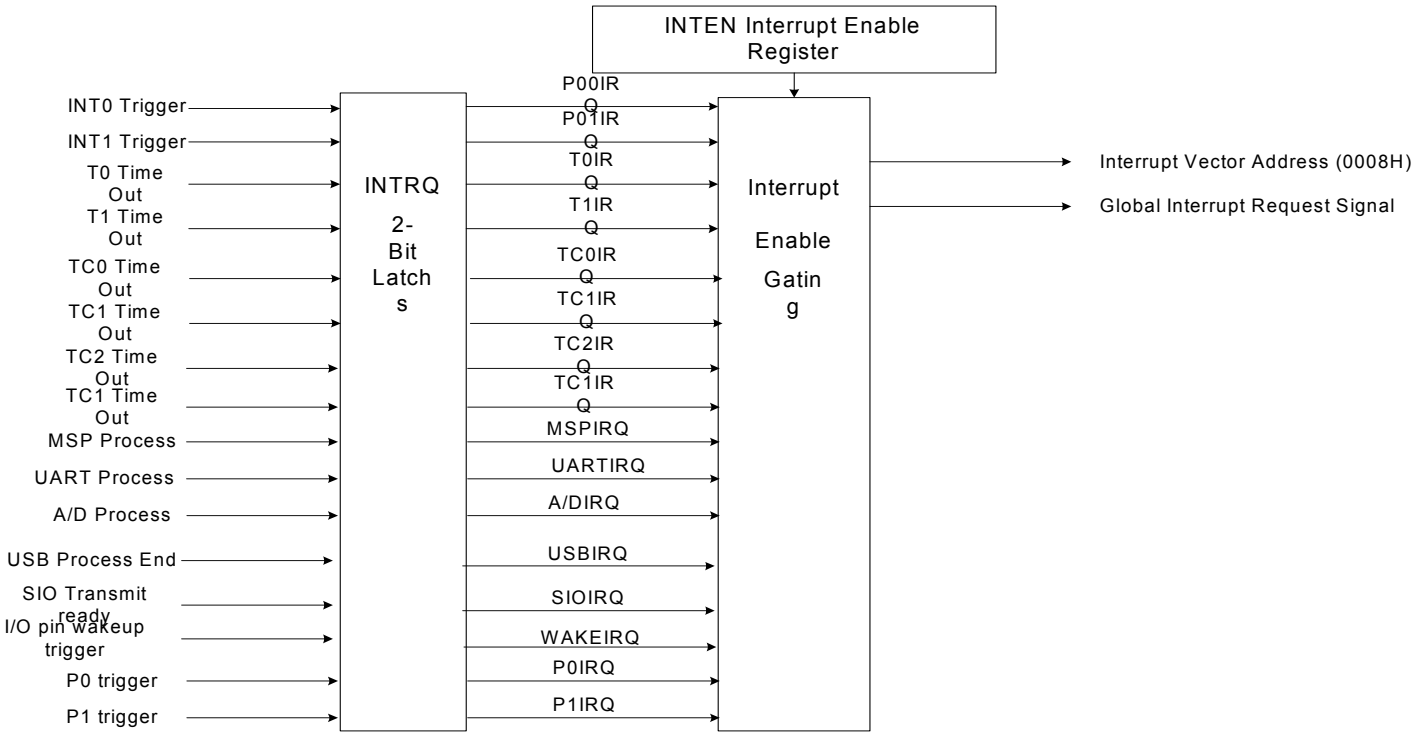
The wakeup time = 1/6MHz * 16384 = 2.72 ms

The total wakeup time = 2.72 ms + oscillator start-up time

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 15 interrupt sources, including 11 internal interrupt (T0/T1/TC0/TC1/TC2/USB/SIO/MSP/UART/AD/IO wakeup) and 4 external interrupt (INT0/INT1/P0/P1). 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.

0C7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN1	P1IEN	P0IEN	MSPIEN	UTRXIEN	UTTXIEN	TC2IEN	TC1IEN	TC0IEN
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 **P1IEN:** P1 I/O level change interrupt control bit.
0 = Disable P1 I/O level change interrupt function.
1 = Enable P1 I/O level change interrupt function.

Bit 6 **P0IEN:** P0 I/O level change interrupt control bit.
0 = Disable P0 I/O level change interrupt function.
1 = Enable P0 I/O level change interrupt function.

Bit 5 **MSPIEN**: MSP function interrupt control bit.
0 = Disable MSP interrupt function.
1 = Enable MSP interrupt function.

Bit 4 **UTRXIEN:** UART RX function interrupt control bit.
0 = Disable UART RX interrupt function.
1 = Enable UART RX interrupt function.

Bit 3 **UTTXIEN:** UART TX function interrupt control bit.
0 = Disable UART TX interrupt function.
1 = Enable UART TX interrupt function.

Bit 2 **TC2IEN:** TC2 timer function interrupt control bit.
0 = Disable TC2 interrupt function.
1 = Enable TC2 interrupt function.

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

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	ADCIEN	USBIEN	T1IEN	T0IEN	SIOIEN	WAKEIEN	P01IEN	P00IEN
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 **ADCIEN:** ADC interrupt control bit.
0 = Disable ADC interrupt function.
1 = Enable ADC interrupt function.

Bit 6 **USBIEN:** USB interrupt control bit.
0 = Disable USB interrupt function.
1 = Enable USB interrupt function.

Bit 5 **T1IEN:** T1 timer interrupt control bit.
0 = Disable T1 interrupt function.
1 = Enable T1 interrupt function.

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

Bit 3 **SIOIEN:** SIO interrupt control bit.
0 = Disable SIO interrupt function.
1 = Enable SIO 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 1 **P01IEN:** External P0.1 interrupt (INT1) control bit.
0 = Disable INT1 interrupt function.
1 = Enable INT1 interrupt function.

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

0C6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ1	P1IRQ	P0IRQ	MSPIRQ	UTRXIRQ	UTTXIRQ	TC2IRQ	TC1IRQ	TC0IRQ
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 **P1IRQ:** P1 I/O level change interrupt request flag.
0 = None P1 I/O level change interrupt request.
1 = P1 I/O level change interrupt request.

Bit 6 **P0IRQ:** P0 I/O level change interrupt request flag.
0 = None P0 I/O level change interrupt request.
1 = P0 I/O level change interrupt request.

Bit 5 **MSPIRQ**: MSP function interrupt request flag.
0 = None MSP interrupt request.
1 = MSP interrupt request.

Bit 4 **UTRXIRQ:** UART RX function interrupt request flag.
0 = None UART RX interrupt request.
1 = UART RX interrupt request.

Bit 3 **UTTXIRQ:** UART TX function interrupt request flag.
0 = None UART TX interrupt request.
1 = UART TX interrupt request.

Bit 2 **TC2IRQ:** TC2 timer function interrupt request flag.
0 = None TC2 interrupt request.
1 = T0 interrupt request.

Bit 1 **TC1IRQ:** TC1 timer interrupt request flag.
0 = None TC1 interrupt request.
1 = T0 interrupt request.

Bit 0 **TC0IRQ:** TC0 timer interrupt request flag.
0 = None TC0 interrupt request.
1 = T0 interrupt request.

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	ADCIRQ	USBIRQ	T1IRQ	T0IRQ	SIOIRQ	WAKEIRQ	P01IRQ	P00IRQ
Read/Write	RW	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 **ADCIRQ:** ADC interrupt request flag.
0 = None ADC interrupt request.
1 = ADC interrupt request.

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

Bit 5 **T1IRQ:** T1 timer interrupt request flag.
0 = None T1 interrupt request.
1 = T1 interrupt request.

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

Bit 3 **SIOIRQ:** SIO interrupt request flag.
0 = None SIO interrupt request.
1 = SIO interrupt request.

Bit 2 **WAKEIRQ:** I/O PORT0 & PORT1 WAKEUP interrupt request flag.
0 = None WAKEUP interrupt request.
1 = WAKEUP interrupt request.

Bit 1 **P01IRQ:** External P0.1 interrupt (INT1) request flag.
0 = None INT0 interrupt request.
1 = INT0 interrupt request.

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

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

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

```
MOV     A, #03H
B0MOV   PEDGE, A        ; Set INT0 interrupt trigger as bi-direction edge.

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

Example: INT0 interrupt service routine.

```
        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 T1 INTERRUPT OPERATION

When the T1C counter overflows, the T1IRQ will be set to “1” no matter the T1IEN is enable or disable. If the T1IEN and the trigger event T1IRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the T1IEN = 0, the trigger event T1IRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the T1IEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: T1 interrupt request setup.**

```
        B0BCLR      FT1IEN      ; Disable T1 interrupt service
        B0BCLR      FT1ENB      ; Disable T1 timer
        MOV         A, #00H     ;
        B0MOV       T1M, A      ; Set T1 clock = Fcpu / 256
        MOV         A, #0E5H    ; Set T1CL initial value = E5H
        B0MOV       T1CL, A
        MOV         A, #48H     ; Set T1CH initial value = 48H
        B0MOV       T1CH, A     ; Set T1 interval = 1s

        B0BSET      FT1IEN      ; Enable T1 interrupt service
        B0BCLR      FT1IRQ      ; Clear T1 interrupt request flag
        B0BSET      FT1ENB      ; Enable T1 timer

        B0BSET      FGIE        ; Enable GIE
```

Example: T1 interrupt service routine.

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

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

        B0BTS1      FT1IRQ      ; Check T1IRQ
        JMP         EXIT_INT    ; T1IRQ = 0, exit interrupt vector

        B0BCLR      FT1IRQ      ; Reset T1IRQ
        MOV         A, #0E5H
        B0MOV       T1CL, A     ; Reset T1CL.
        MOV         A, #48H
        B0MOV       T1CH, A     ; Reset T1CH.
        …                       ; T1 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.9 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
```

6.10 TC1 INTERRUPT OPERATION

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

➢ **Example: TC1 interrupt request setup.**

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

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

        B0BSET      FGIE            ; Enable GIE
```

➢ **Example: TC1 interrupt service routine.**

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

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

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

        B0BCLR      FTC1IRQ         ; Reset TC1IRQ
        MOV         A, #74H
        B0MOV       TC1C, A         ; Reset TC1C.
        …                           ; TC1 interrupt service routine
        …
EXIT_INT:
        …                           ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

6.11 TC2 INTERRUPT OPERATION

When the TC2C counter overflows, the TC2IRQ will be set to "1" no matter the TC2IEN is enable or disable. If the TC2IEN and the trigger event TC2IRQ is set to be "1". As the result, the system will execute the interrupt vector. If the TC2IEN = 0, the trigger event TC2IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the TC2IEN is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: TC1 interrupt request setup.**

```
        B0BCLR      FTC2IEN     ; Disable TC2 interrupt service
        B0BCLR      FTC2ENB     ; Disable TC2 timer
        MOV         A, #20H     ;
        B0MOV       TC2M, A     ; Set TC2 clock = Fcpu / 64
        MOV         A, #74H     ; Set TC2C initial value = 74H
        B0MOV       TC2C, A     ; Set TC2 interval = 10 ms

        B0BSET      FTC2IEN     ; Enable TC2 interrupt service
        B0BCLR      FTC2IRQ     ; Clear TC2 interrupt request flag
        B0BSET      FTC2ENB     ; Enable TC2 timer

        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: TC1 interrupt service routine.**

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

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

        B0BTS1      FTC2IRQ     ; Check TC2IRQ
        JMP         EXIT_INT    ; TC1IRQ = 0, exit interrupt vector

        B0BCLR      FTC2IRQ     ; Reset TC2IRQ
        MOV         A, #74H
        B0MOV       TC1C, A     ; Reset TC2C.
        …                       ; TC2 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.12 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
```

6.13 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
```

6.14 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
```

6.15 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
T1IRQ	T1C 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         INTT1CHK        ; Jump check to next interrupt
                B0BTS0      FT0IRQ          ; Check T0IRQ
                JMP         INTT0           ; Jump to T0 interrupt service routine
INTT1CHK:                                   ; Check T1 interrupt request
                B0BTS1      FT1IEN          ; Check T1IEN
                JMP         INTTC1CHK       ; Jump check to next interrupt
                B0BTS0      FT1IRQ          ; Check T1IRQ
                JMP         INTT1           ; Jump to T1 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	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

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

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

0C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	P55M	P54M	P53M	P52M	P51M	P50M
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit[7:0] **PnM[7:0]:** Pn mode control bits. (n = 0~3).
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	P07R	P06R	P05R	P00R	P03R	P02R	P01R	P00R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

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

0E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	-	-	P55R	P54R	P53R	P52R	P51R	P50R
Read/Write	-	-	W	W	W	W	W	W
After reset	-	-	0	0	0	0	0	0

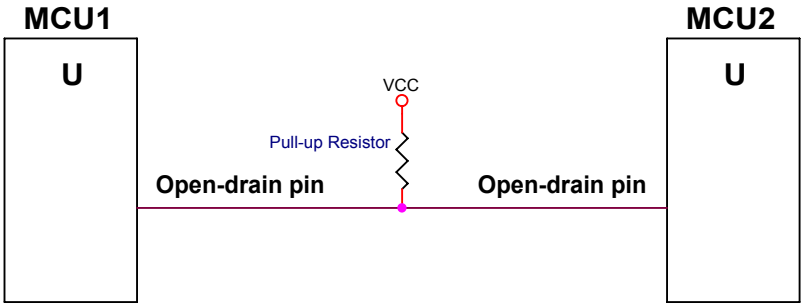
✸ ***Note: P0.4 is input only pin with pull-up resister.***

➢ **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 OPEN-DRAIN REGISTER

P1.0/P1.1/P0.6/P0.5 is built-in open-drain function. P1.0/P1.1/P0.6/P0.5 must be set as output mode when enable P1.0/P1.1/P0.6/P0.5 open-drain function. Open-drain external circuit is as following.



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

0E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1OC	-	-	-	-	P06OC	P05OC	P11OC	P10OC
Read/Write	-	-	-	-	W	W	W	W
After reset	-	-	-	-	0	0	0	0

Bit [3:2] **P0nOC:** Port 0 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit [1:0] **P1nOC:** Port 1 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

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

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

B0BSET      P10M        ; Enable P1.0 output mode.
MOV         A, #01H     ; Enable P1.0 open-drain function.
B0MOV       P1OC, A
```

✸ ***Note: P1OC is write only register. Setting P10OC must be used "MOV" instructions.***

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

```
MOV         A, #0       ; Disable P1.0 open-drain function.
B0MOV       P1OC, A
```

✸ ***Note: After disable P1.0 open-drain function, P1.0 mode returns to last I/O mode.***

7.4 I/O PORT DATA REGISTER

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

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

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

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	P47	P46	P45	P44	P43	P42	P41	P40
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	P55	P54	P53	P52	P51	P50
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

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

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

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

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

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

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

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

7.5 I/O PORT1 WAKEUP CONTROL REGISTER

0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
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] **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 (12KHz).

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

VDD	*Internal Low RC Freq.*	*Code option – slow mode setting*	*Watchdog Overflow Time*
5V	12KHz	Flosc/2	341ms
5V	12KHz	Flosc/4	682ms

✸ ***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              ; Clear the watchdog timer.
                B0MOV       WDTR,A
                …
                CALL        SUB1
                CALL        SUB2
                …
                …
                …
                JMP         MAIN
```

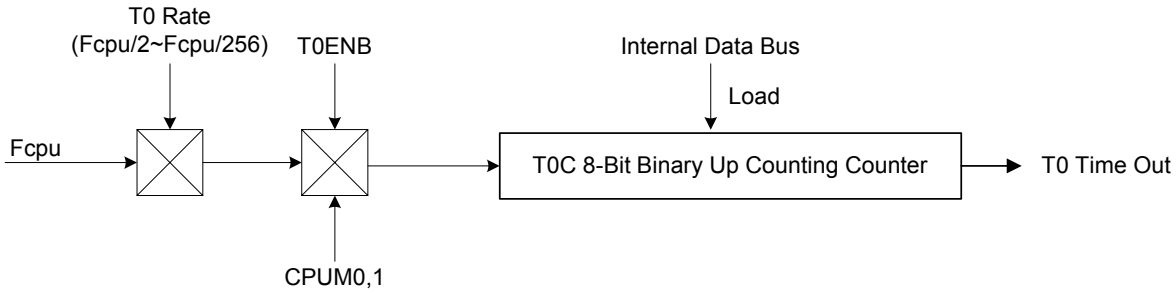
8.2 TIMER 0 (T0)

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

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

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 12MHz. Fcpu=Fosc/2. 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 = 12MHz / 2)	
		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.
```

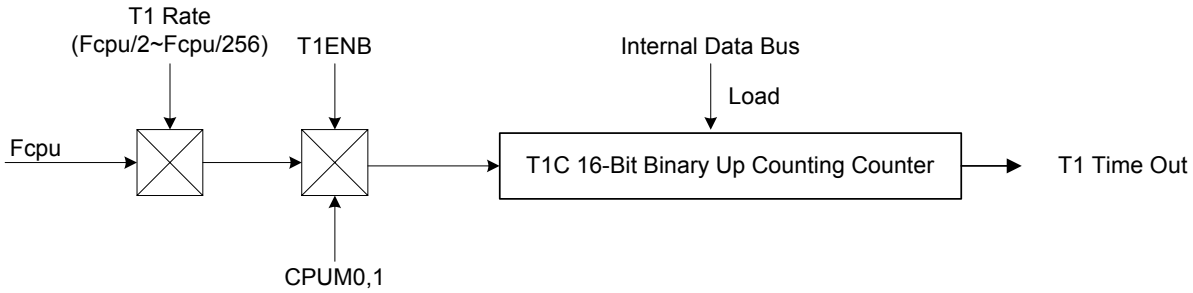
8.3 TIMER T1 (T1)

8.3.1 OVERVIEW

The T1 is a 16-bit binary up timer and event counter. If T1 timer occurs an overflow (from FFFFH to 0000H), it will continue counting and issue a time-out signal to trigger T1 interrupt to request interrupt service.

The main purpose of the T1 timer is as following.

- ☞ **16-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- ☞ **Green mode wakeup function:** T1 can be green mode wake-up time as T1ENB = 1. System will be wake-up by T1 time out.



8.3.2 T1M MODE REGISTER

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1rate2	T1rate1	T1rate0	-	-	-	
Read/Write	R/W	R/W	R/W	R/W	-	-	-	
After reset	0	0	0	0	-	-	-	

Bit 7 **T1ENB:** T1 counter control bit.
0 = Disable T1 timer.
1 = Enable T1 timer.

Bit [6:4] **T1RATE[2:0]:** T1 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
…
110 = fcpu/4.
111 = fcpu/2.

8.3.3 T1C COUNTING REGISTER

T1CL with T1CH is an 16-bit counter register for T1 interval time control.

0DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CL	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0
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

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CH	T1C15	T1C14	T1C13	T1C12	T1C11	T1C10	T1C9	T1C8
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 T1C initial value is as following.

T1C initial value = 65536 - (T1 interrupt interval time * input clock)

Example: To set 1ms interval time for T1 interrupt. High clock is 12MHz. Fcpu=Fosc/2. Select T1RATE=001 (Fcpu/128).

*T1C initial value = 65536 - (T1 interrupt interval time * input clock)*
*= 65536 - (1s * 12MHz / 2 / 128)*
*= 65536 - (10 * 6 * 10^6 / 1 / 128)*
= 18661
= 48E5H

The basic timer table interval time of T1.

T1RATE	T1CLOCK	High speed mode (Fcpu = 12MHz / 2)	
		Max overflow interval	One step = max/256
000	Fcpu/256	2.796 s	42.67 us
001	Fcpu/128	1.398 s	21.33 us
010	Fcpu/64	699.051 ms	10.67 us
011	Fcpu/32	349.525 ms	5.33 us
100	Fcpu/16	174.763 ms	2.67 us
101	Fcpu/8	87.381 ms	1.33 us
110	Fcpu/4	43.691 ms	0.67 us
111	Fcpu/2	21.845 ms	0.33 us

8.3.4 T1 TIMER OPERATION SEQUENCE

T1 timer operation sequence of setup T1 timer is as following.

☞ **Stop T1 timer counting, disable T1 interrupt function and clear T1 interrupt request flag.**

```
B0BCLR      FT1ENB      ; T1 timer.
B0BCLR      FT1IEN      ; T1 interrupt function is disabled.
B0BCLR      FT1IRQ      ; T1 interrupt request flag is cleared.
```

☞ **Set T1 timer rate.**

```
MOV         A, #0xxx0000b   ;The T1 rate control bits exist in bit4~bit6 of T1M. The
                            ; value is from x000xxxxb~x111xxxxb.
B0MOV       T1M,A           ; T1 timer is disabled.
```

☞ **Set T1 interrupt interval time.**

```
MOV         A,#0E5H
B0MOV       T1CL,A      ; Set T1CL value.
MOV         A,#48H
B0MOV       T1CH,A      ; Set T1CH value.
```

☞ **Set T1 timer function mode.**

```
B0BSET      FT1IEN      ; Enable T1 interrupt function.
```

☞ **Enable T1 timer.**

```
B0BSET      FT1ENB      ; Enable T1 timer.
```

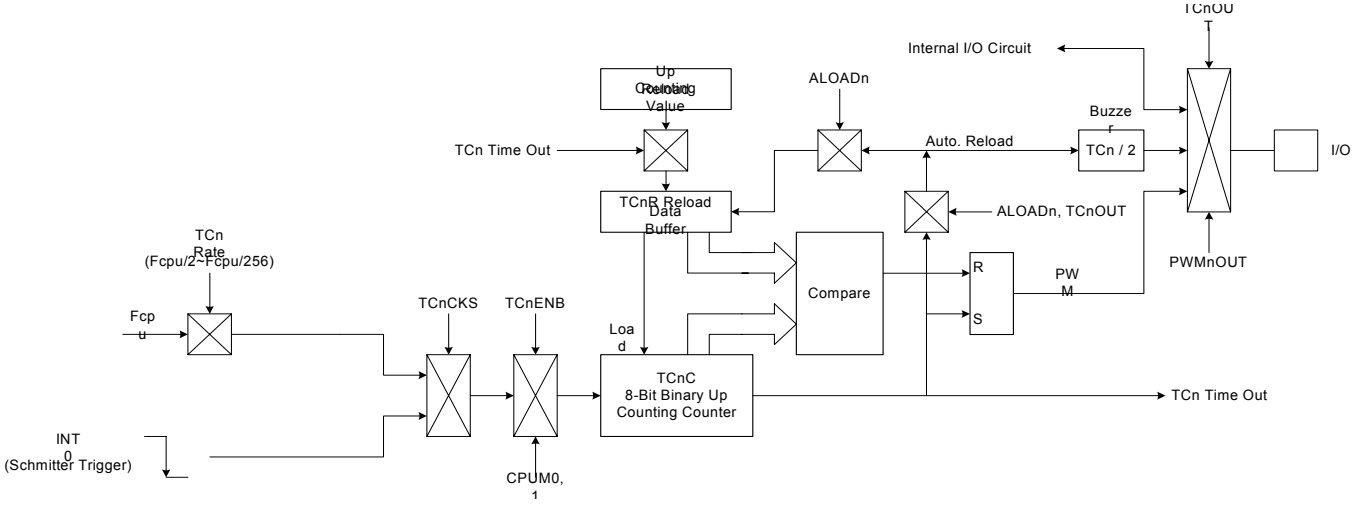
8.4 TIMER/COUNTER 0 (TC0~TC2)

8.4.1 OVERVIEW

The TCn (n=0, 1, 2) is an 8-bit binary up counting timer with double buffers. TCn 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 TCnM register selects TCnC's clock source from internal or external. If TCn timer occurs an overflow, it will continue counting and issue a time-out signal to trigger TCn interrupt to request interrupt service. TCn overflow time is 0xFF to 0X00 normally. Under PWM mode, TCn overflow is decided by PWM cycle controlled by ALOADn (n=0, 1, 2) and TCnOUT bits.

The main purposes of the TCn timer are 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**



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

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

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

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

✸ ***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).***

08BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1M	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **TC1ENB:** TC1 counter control bit.
0 = Disable TC1 timer.
1 = Enable TC1 timer.

Bit [6:4] **TC1RATE[2:0]:** TC1 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
…
110 = fcpu/4.
111 = fcpu/2.

Bit 3 **TC1CKS:** TC1 clock source select bit.
0 = Internal clock (Fcpu or Fosc).
1 = External clock from P0.1/INT1 pin.

Bit 2 **ALOAD1:** Auto-reload control bit. **Only valid when PWM1OUT = 0.**
0 = Disable TC1 auto-reload function.
1 = Enable TC1 auto-reload function.

Bit 1 **TC1OUT:** TC1 time out toggle signal output control bit. **Only valid when PWM1OUT = 0.**
0 = Disable, P5.4 is I/O function.
1 = Enable, P5.4 is output TC0OUT signal.

Bit 0 **PWM1OUT:** PWM output control bit.
0 = Disable PWM output.
1 = Enable PWM output. PWM duty controlled by TC1OUT, ALOAD1 bits.

✷ ***Note: When TC1CKS=1, TC0 became an external event counter and TC1RATE is useless. No more P0.1 interrupt request will be raised. (P0.1IRQ will be always 0).***

08EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2M	TC2ENB	TC2rate2	TC2rate1	TC2rate0	TC2CKS	ALOAD2	TC2OUT	PWM2OUT
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 **TC2ENB:** TC2 counter control bit.
0 = Disable TC2 timer.
1 = Enable TC2 timer.

Bit [6:4] **TC2RATE[2:0]:** TC2 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
…
110 = fcpu/4.
111 = fcpu/2.

Bit 3 **TC2CKS:** TC2 clock source select bit.
0 = Internal clock (Fcpu or Fosc).
1 = External clock from P0.2 pin.

Bit 2 **ALOAD2:** Auto-reload control bit. **Only valid when PWM2OUT = 0.**
0 = Disable TC2 auto-reload function.
1 = Enable TC2 auto-reload function.

Bit 1 **TC2OUT:** TC2 time out toggle signal output control bit. **Only valid when PWM2OUT = 0.**
0 = Disable, P5.5 is I/O function.
1 = Enable, P5.5 is output TC2OUT signal.

Bit 0 **PWM2OUT:** PWM output control bit.
0 = Disable PWM output.
1 = Enable PWM output. PWM duty controlled by TC2OUT, ALOAD2 bits.

8.4.3 TCnC COUNTING REGISTER

TCnC (n = 0, 1, 2) is an 8-bit counter register for TCn 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

08CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1C	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

08FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2C	TC2C7	TC2C6	TC2C5	TC2C4	TC2C3	TC2C2	TC2C1	TC2C0
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 TCnC initial value is as following.

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

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

TCnCKS	PWMn	ALOADn	TCnOUT	N	TCnC valid value	TCnC 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 TCn interrupt. TCn clock source is Fcpu (TCnKS=0) and no PWM output (PWMn=0). High clock is internal 6MHz. Fcpu=Fosc/2. Select TCnRATE=010 (Fcpu/64).**

*TCnC initial value = N - (TCn 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 TCn.

TCnRATE	TCnCLOCK	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

8.4.4 TCnR AUTO-LOAD REGISTER

TCn (n = 0, 1, 2) timer is with auto-load function controlled by ALOADn (n = 0, 1, 2) bit of TCnM (n = 0, 1, 2). When TCnC (n = 0, 1, 2) overflow occurring, TCnR (n = 0, 1, 2) value will load to TCnC by system. It is easy to generate an accurate time, and users don't reset TCnC during interrupt service routine.

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

✸ ***Note: Under PWM mode, auto-load is enabled automatically. The ALOADn 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

08DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1R	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2R	TC2R7	TC2R6	TC2R5	TC2R4	TC2R3	TC2R2	TC2R1	TC2R0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of TCnR initial value is as following.

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

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

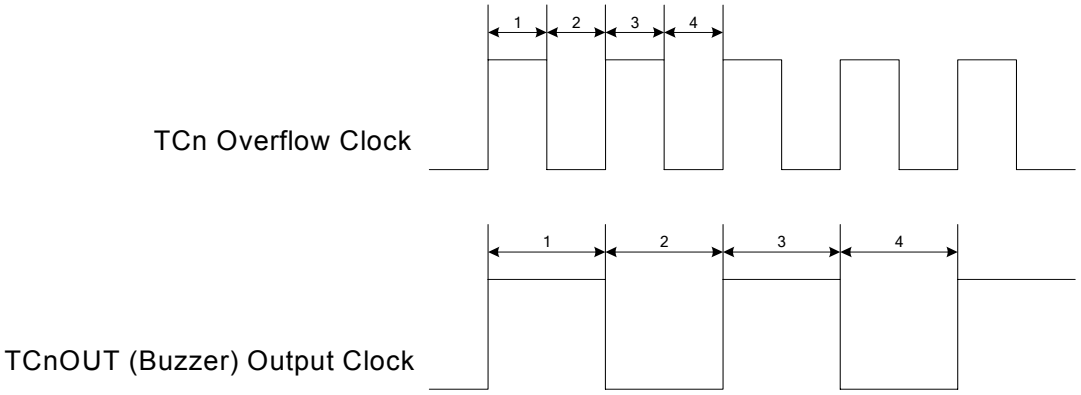
TCnCKS	PWMn	ALOADn	TCnOUT	N	TCnR valid value	TCnR value binary type
0	0	x	x	256	0x00~0xFF	00000000b~11111111b
	1	0	0	256	0x00~0xFF	00000000b~11111111b
	1	0	1	64	0x00~0x3F	xx000000b~xx111111b
	1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
	1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
1	-	-	-	256	0x00~0xFF	00000000b~11111111b

➢ **Example: To set 1ms interval time for TCn interrupt. TCn clock source is Fcpu (TCnKS=0) and no PWM output (PWMn=0). High clock is internal 6MHz. Fcpu=Fosc/2. Select TCnRATE=010 (Fcpu/64).**

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

8.4.5 TCn CLOCK FREQUENCY OUTPUT (BUZZER)

Buzzer output (TCnOUT) is from TCn timer/counter frequency output function. If 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 TCnOUT frequency is divided by 2 from TCn interval time. TCnOUT frequency is 1/2 TCn frequency. The TCn clock has many combinations and easily to make difference frequency. The TCnOUT 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 = TCnR = 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  FALOAD0         ; Enable TC0 auto-reload function
B0BSET  FTC0ENB         ; Enable TC0 timer
```

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

8.4.6 TCn TIMER OPERATION SEQUENCE

TCn timer operation includes timer interrupt, event counter, TCnOUT 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.
```

8.5 PWMn MODE

8.5.1 OVERVIEW

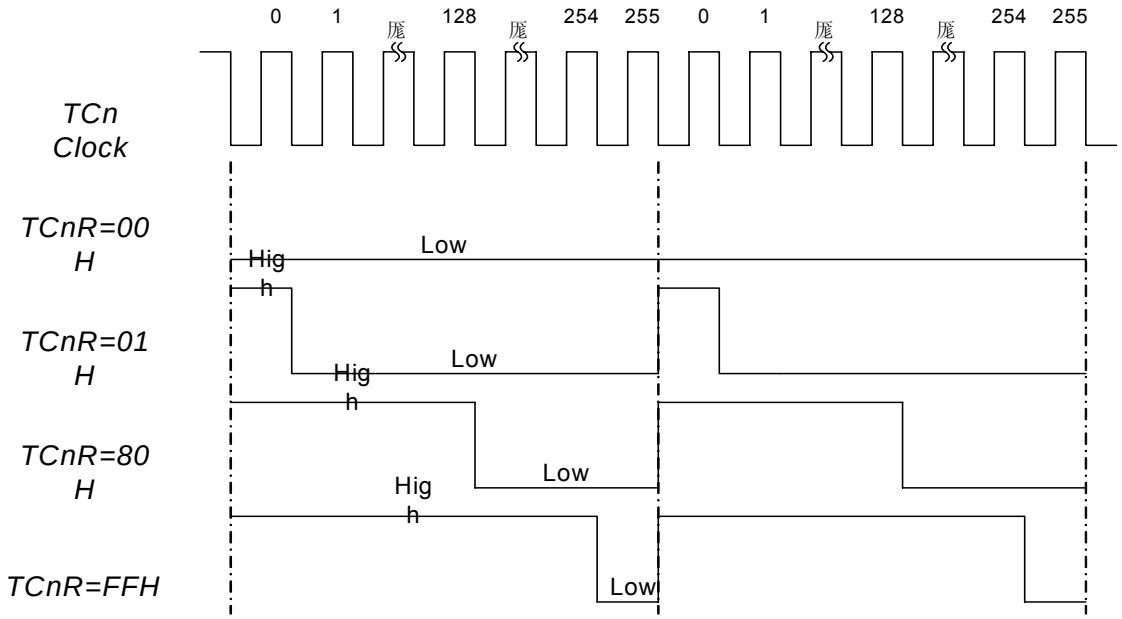
PWM function is generated by TCn timer counter and output the PWM signal to PWM0OUT pin (P5.3), PWM1OUT pin (P5.4) PWM2OUT pin (P5.5). The 8-bit counter counts modulus 256, 64, 32, 16 controlled by ALOADn, TCnOUT bits. The value of the 8-bit counter (TCnC) is compared to the contents of the reference register (TCnR). When the reference register value (TCnR) is equal to the counter value (TCnC), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWMn output is TCnR/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 TCnR.

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

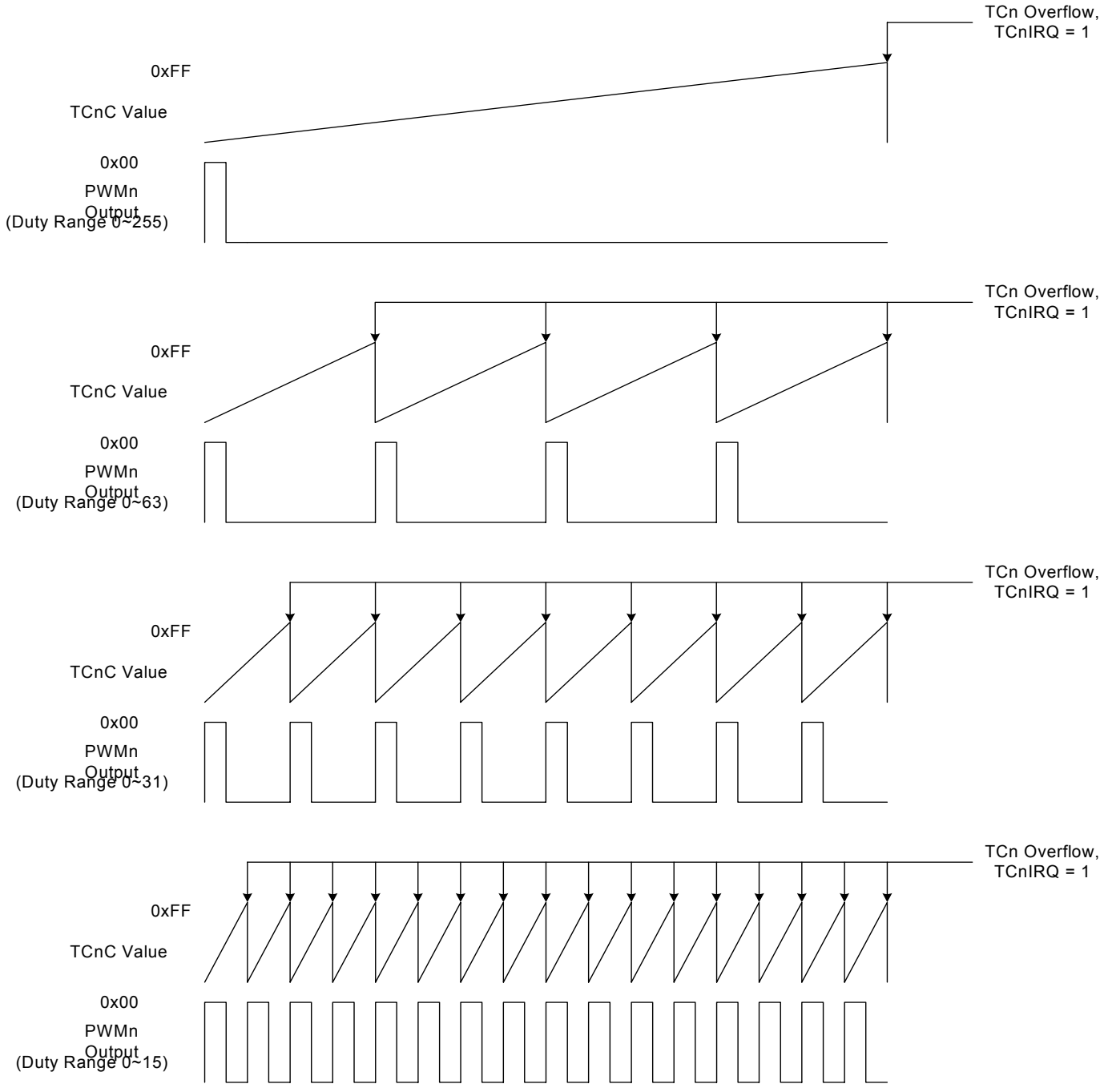
ALOADn	TCnOUT	PWM duty range	TCnC valid value	TCnR valid bits value	MAX. PWM Frequency (Fcpu = 6MHz)	Remark
0	0	0/256~255/256	0x00~0xFF	0x00~0xFF	11.719K	Overflow per 256 count
0	1	0/64~63/64	0x00~0x3F	0x00~0x3F	46.875K	Overflow per 64 count
1	0	0/32~31/32	0x00~0x1F	0x00~0x1F	93.75K	Overflow per 32 count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

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



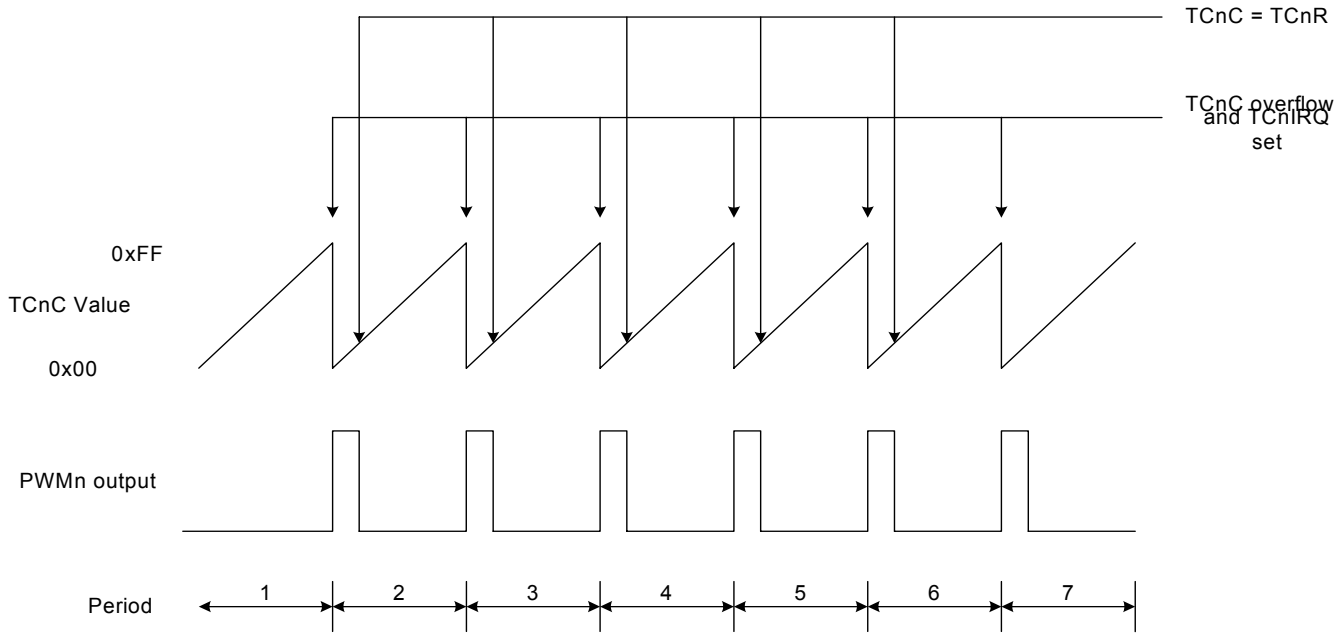
8.5.2 TCnIRQ and PWM Duty

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

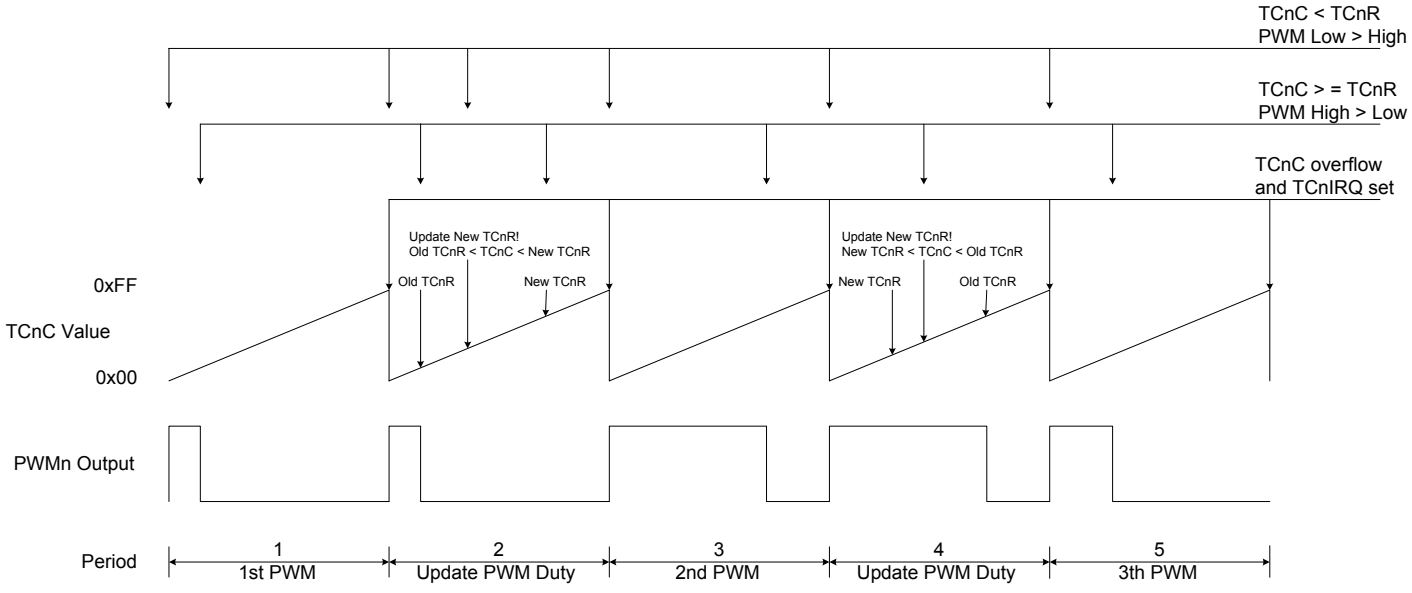


8.5.3 PWM Duty with TCnR Changing

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



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



In period 2 and period 4, new Duty (TCnR) is set. TCn 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.

8.5.4 PWM PROGRAM EXAMPLE

➢ **Example: Setup PWM0 output from TC0 to PWM0OUT (P5.3). The clock source is internal 12MHz. Fcpu = Fosc/2 = 6MHz. 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 P5.4 and disable P5.4 I/O function
B0BSET    FTC0ENB         ; Enable TC0 timer
```

✸ ***Note: The TCnR 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, keyboard, joystick, and game pad.

USB Specification Compliance
— Conforms to USB specifications, Version 2.0.
— Supports 1 Full-speed USB device address.
— Supports 1 control endpoint, 2 interrupt endpoints and 2 interrupt/bulk 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 sends 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 SOF packet received if the SOF interrupt enable.
• 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 0~3 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.

091H	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 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.

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.

9.5.2 USB STATUS REGISTER

The USB status register indicates the status of USB.

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USTATUS	CRCERR	PKTERR	SOF	BUS_RST	SUSPEND	EP0SETUP	EP0IN	EP0OUT
Read/Write	R/W	R/W	R/W	R	R	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **CRCERR:** USB data CRC check error.
0 = Non-USB data CRC check error, cleared by firmware.
1 = Set to 1 by hardware when USB data CRC check error occur.

Bit 6 **PKTERR:** USB packet error.
0 = Non-USB packet error, cleared by firmware.
1 = Set to 1 by hardware when USB packet error occur.

Bit 5 **SOF:** Indicate the USB SIE's SOF packet is received
0 = Non USB SIE's SOF packet received.
1 = If SOF_INT_EN = 1 then this bit will be set to 1 by hardware when the SOF packet is received. Otherwise the bit will always be 0.

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

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 2 **EP0SETUP:** 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 1 **EP0IN:** 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 0 **EP0OUT:** 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.

9.5.3 USB DATA COUNT REGISTER

The USB EP0 OUT token data byte counter.

093H	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.

094H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USB_INT_EN	REG_EN	DP_UP_EN	SOF_INT_EN		EP4NAK _INT_EN	EP3NAK _INT_EN	EP2NAK _INT_EN	EP1NAK _INT_EN
Read/Write	R/W	R/W	R/W		R/W	R/W	R/W	R/W
After reset	1	0	0		0	0	0	0

Bit 7 **REG_EN:** 3.3volts Regulator control bit.
0 = Disable regulator output 3.3volts.
1 = Enable regulator output 3.3volts.

Bit 6 **DP_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 5 **SOF_INT_EN:** USB SIE's SOF packet receive interrupt enable.
0 = Disable USB SIE's SOF interrupt request.
1 = Enable USB SIE's SOF interrupt request.

Bit [3:0] **EPnNAK_INT_EN [3:0]:** EP1~EP4 NAK transaction interrupts enable control bits. n = 1, 2, 3, 4.
0 = Disable NAK transaction interrupt request.
1 = Enable NAK transaction interrupt request.

9.5.5 USB endpoint's ACK handshaking flag REGISTER

The status of endpoint's ACK transaction.

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP_ACK					EP4_ACK	EP3_ACK	EP2_ACK	EP1_ACK
Read/Write					R/W	R/W	R/W	R/W
After reset					0	0	0	0

Bit [3:0] **EPn_ACK [3:0]:** EP1~EP4 ACK transaction. n= 1, 2, 3, 4. 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.

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP_NAK					EP4_NAK	EP3_NAK	EP2_NAK	EP1_NAK
Read/Write					R/W	R/W	R/W	R/W
After reset					0	0	0	0

Bit [3:0] **EPn_NAK [3:0]:** EP1~EP4 NAK transaction. n = 1, 2, 3, 4. 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.

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE0R	-	UE0M1	UE0M0	-	UE0C3	UE0C2	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 [6:5] **UE0M [1:0]:** The endpoint 0 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 0. For example, if the endpoint 0's mode bit is set to 00 that is NAK IN/OUT mode as shown in *Table,* The USB SIE will send NAK handshakes in response to any IN/OUT token set to the endpoint 0.The bit 5 UE0M0 will auto reset to zero when the ACK transaction complete.

USB endpoint 0's mode table

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

Bit [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.

9.5.8 USB ENDPOINT 1 ENABLE REGISTER

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

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE1R	UE1E	UE1M1	UE1M0					
Read/Write	R/W	R/W	R/W					
After reset	0	0	0					

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

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

099H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE1R_C			UE1C5	UE1C4	UE1C3	UE1C	UE1C1	UE1C0
Read/Write			R/W	R/W	R/W	R/W	R/W	R/W
After reset			0	0	0	0	0	0

Bit [5:0] **UE1C [5:0]:** 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.

9.5.9 USB ENDPOINT 2 ENABLE REGISTER

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

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE2R	UE2E	UE2M1	UE2M0					
Read/Write	R/W	R/W	R/W					
After reset	0	0	0					

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

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

09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE2R_C			UE2C5	UE2C4	UE2C3	UE2C	UE2C1	UE2C0
Read/Write			R/W	R/W	R/W	R/W	R/W	R/W
After reset			0	0	0	0	0	0

Bit [5:0] **UE2C [5:0]:** 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.

9.5.10 USB ENDPOINT 3 ENABLE REGISTER

The communication with the USB host using endpoint 3, endpoint 3's FIFO is implemented as Y bytes of dedicated RAM. The endpoint 3 is an interrupt and bulk endpoint.

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE3R	UE3E	UE3M1	UE3M0					
Read/Write	R/W	R/W	R/W					
After reset	0	0	0					

Bit 7 **UE3E:** USB endpoint 3 function enable bit.
0 = disable USB endpoint 3 function.
1 = enable USB endpoint 3 function.

Bit [6:5] **UE3M [1:0]:** The endpoint 3 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 3. For example, if the endpoint 3'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 3. The bit 5 UE3M0 will auto reset to zero when the ACK transaction complete.

USB endpoint 3's mode table

UE3M1	UE3M0	IN/OUT Token Handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE3R_C			UE3C5	UE3C4	UE3C3	UE3C	UE3C1	UE3C0
Read/Write			R/W	R/W	R/W	R/W	R/W	R/W
After reset			0	0	0	0	0	0

Bit [5:0] **UE3C [5:0]:** 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 3 FIFO.

9.5.11 USB ENDPOINT 4 ENABLE REGISTER

The communication with the USB host using endpoint 4, endpoint 4's FIFO is implemented as Z bytes of dedicated RAM. The endpoint 4 is an interrupt and bulk endpoint.

09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE4R	UE4E	UE4M1	UE4M0					
Read/Write	R/W	R/W	R/W					
After reset	0	0	0					

Bit 7 **UE4E:** USB endpoint 4 function enable bit.
0 = disable USB endpoint 4 function.
1 = enable USB endpoint 4 function.

Bit [6:5] **UE4M [1:0]:** The endpoint 4 modes determine how the SIE responds to USB traffic that the host sends to the endpoint 4. For example, if the endpoint 4'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 4. The bit 5 UE4M0 will auto reset to zero when the ACK transaction complete.

USB endpoint 4's mode table

UE4M1	UE4M0	IN/OUT Token Handshake
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

09FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE4R_C			UE4C5	UE4C4	UE4C3	UE4C	UE4C1	UE4C0
Read/Write			R/W	R/W	R/W	R/W	R/W	R/W
After reset			0	0	0	0	0	0

Bit [5:0] **UE4C [5:0]:** 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 4 FIFO.

9.5.12 USB ENDPOINT FIFO ADDRESS SETTING REGISTER

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP2FIFO_ADDR	EP2FIFO7	EP2FIFO6	EP2FIFO5	EP2FIFO4	EP2FIFO3	EP2FIFO2	EP2FIFO1	EP2FIFO0
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] **EP2FIFO_ADDR [7:0]:** EP2 FIFO start address.

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP3FIFO_ADDR	EP3FIFO7	EP3FIFO6	EP3FIFO5	EP3FIFO4	EP3FIFO3	EP3FIFO2	EP3FIFO1	EP3FIFO0
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] **EP3FIFO_ADDR [7:0]:** EP3 FIFO start address.

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP4FIFO_ADDR	EP4FIFO7	EP4FIFO6	EP4FIFO5	EP4FIFO4	EP4FIFO3	EP4FIFO2	EP4FIFO1	EP4FIFO0
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] **EP4FIFO_ADDR [7:0]:** EP4 FIFO start address.

9.5.13 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

USB Data Pointer: Access the USB data by the data pointer.

9.5.14 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.

9.5.15 UPID REGISTER

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

0A7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UPID		-	-	-	-	UBDE	DDP	DDN
Read/Write		-	-	-	-	R/W	R/W	R/W
After reset		-	-	-	-	0	0	0

Bit 2 **UBDE:** Enable to direct drive USB bus.
0 = disable.
1 = enable.

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

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

9.5.16 ENDPOINT TOGGLE BIT CONTROL REGISTER

0ACH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UTOGGLE	-	-	-	-	EP4 _DATA01	EP3 _DATA01	EP2 _DATA01	EP1 _DATA01
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	1	1	1	1

Bit [3:0] Endpoint 1~4's DATA0/1 toggle bit control.
0 = Clear the endpoint 1~4's toggle bit to DATA0
1 = hardware set toggle bit automatically.

10 Universal Asynchronous Receiver/Transmitter (UART)

10.1 OVERVIEW

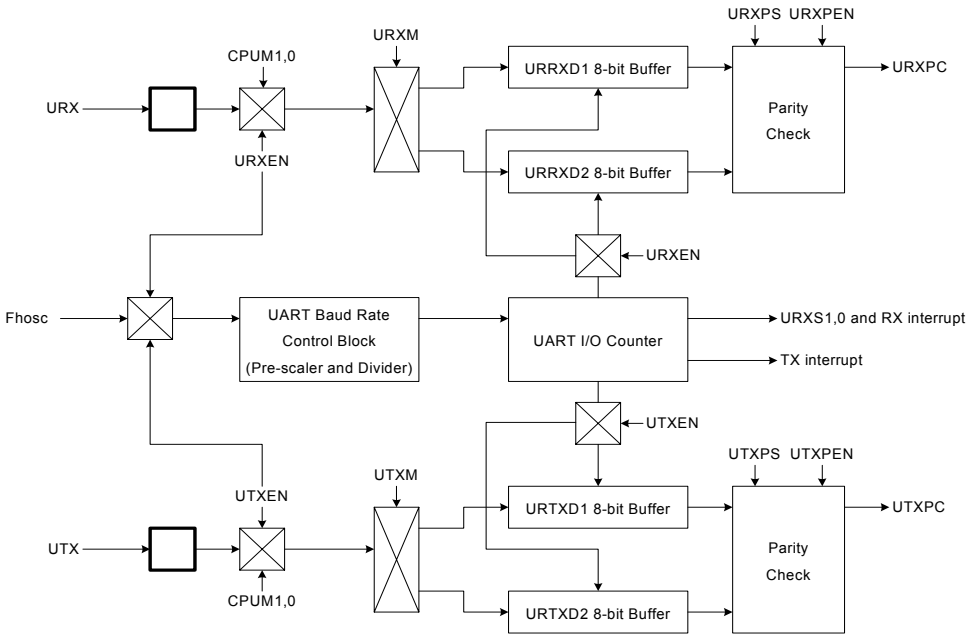
The UART interface is an universal asynchronous receiver/transmitter method. The serial interface is applied to low speed data transfer and communicate with low speed peripheral devices. The UART transceiver of Sonix 8-bit MCU allows RS232 standard and supports one and two bytes data length. The transfer format has start bit, 8/16-bit data, parity bit and stop bit. Programmable baud rate supports different speed peripheral devices. UART I/O pins support push-pull and open-drain structures controlled by register.

The UART features include the following:

- **Full-duplex, 2-wire asynchronous data transfer.**
- **Programmable baud rate.**
- **8-bit and 16-bit data length.**
- **Odd and even parity bit.**
- **End-of-Transfer interrupt.**

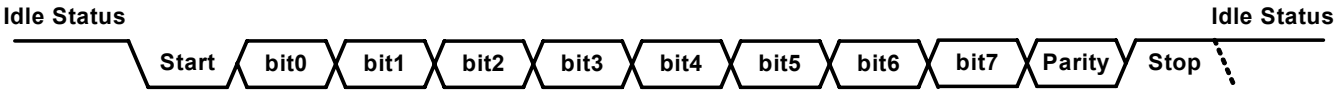
10.2 UART OPERATION

The UART RX and TX pins are shared with GPIO. When UART enables (URXEN=1, UTXEN=1), the UART shared pins transfers to UART purpose and disable GPIO function automatically. When UART disables, the UART pins returns to GPIO last status. The UART data buffer length supports 1-byte and 2-byte. After UART RX operation finished, the UTRXIRQ sets as “1”. After UART TX operation finished, the UTTXIRQ sets as “1”. The UART IRQ bits are cleared by program. If the UTRXIEN or UTTXIEN set to enable, the UTRXIRQ and UTTXIRQ triggers the interrupt request and program counter jumps to interrupt vector to execute interrupt service routine.

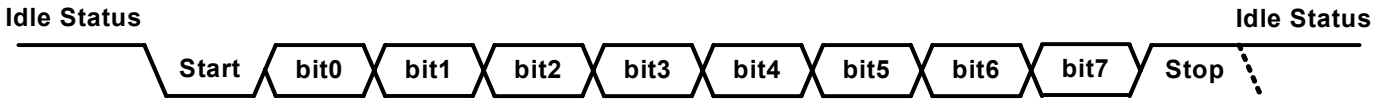


UART Interface Circuit Diagram

The UART transfer format includes “Bus idle status”, “Start bit”, “8-bit Data”, “Parity bit” and “Stop bit” as following.



UART Transfer Format with Parity Bit



UART Transfer Format without Parity Bit

Bus Idle Status

The bus idle status is the bus non-operating status. The UART receiver bus idle status of MCU is floating status and tied high by the transmitter device terminal. The UART transmitter bus idle status of MCU is high status. The UART bus will be set when URXEN and UTXEN are enabled.

Start Bit

UART is a asynchronous type of communication and need a attention bit to offer receiver the transfer starting. The start bit is a simple format which is high to low edge change and the duration is one bit period. The start bit is easily recognized by the receiver.

8-bit Data

The data format is 8-bit length, and LSB transfers first following start bit. The one bit data duration is the unit of UART baud rate controlled by register.

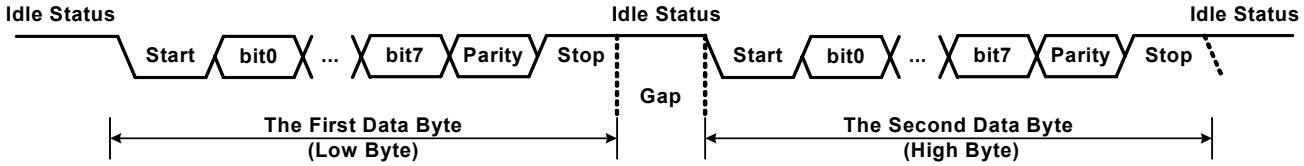
Parity Bit

The parity bit purpose is to detect data error condition. It is an extra bit following the data stream. The parity bit includes odd and even check methods controlled by URXPS/UTXPS bits. After receiving data and parity bit, the parity check executes automatically. The URXPC bit indicates the parity check result. The parity bit function is controlled by URXPEN/UTXPEN bits. If the parity bit function is disabled, the UART transfer contents remove the parity bit and the stop bit follows the data stream directly.

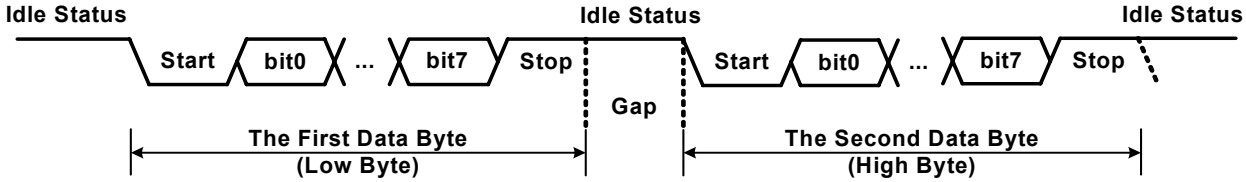
Stop Bit

The stop bit is like start bit using a simple format to indicate the end of UART transfer. The stop bit format is low to high edge change and the duration is one bit period.

The UART communication supports 2-byte data length. The function is for continuous data streams and immediate data request. The 2-byte data format is a continuously byte data form. The gap between the 2-byte data is unit baud rate. The first byte data stores in URRXD1 (receiver) and URTXD1 (transmitter). The second byte data stores in URRXD2 (receiver) and URTXD2 (transmitter).The 2-byte data format is as following.



2-Byte Transfer Format with Parity Bit



2-Byte Transfer Format without Parity Bit

The UART supports interrupt function. UTRXIEN/UTTXIEN are UART transfer interrupt function control bit. UTRXIEN=0, disable UART receiver interrupt function. UTTXIEN=0, disable UART transmitter interrupt function. UTRXIEN=1, enable UART receiver interrupt function. UTTXIEN=1, enable UART transmitter interrupt function. When UART interrupt function enable, the program counter points to interrupt vector (ORG 8) to do UART interrupt service routine after UART operating. UTRXIRQ and UTTXIRQ are UART interrupt request flags, and also to be the UART operating status indicator when UTRXIEN=0 or UTTXIEN=0, but cleared by program. When UART operation finished, the UTRXIRQ/UTTXIRQ would be set to "1".

Note: The first step of UART operation is to setup the UART pins' mode. Enable URXEN/UTXEN to control UART pins' mode.

10.3 UART TRANSMITTER CONTROL REGISTER

URTX initial value = 0xx0000x

0A9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTX	UCLKS			UTXEN	UTXPEN	UTXPS	UTXM	
Read/Write	R/W			R/W	R/W	R/W	R/W	
After Reset	0			0	0	0	0	

Bit 7 **UCLKS:** UART clock source select bit.
0 = UART clock source is 16MHz.
1 = UART clock source is 24MHz.

Bit 4 **UTXEN:** UART TX control bit.
0 = Disable UART TX. UTX pin keeps and returns to GPIO function.
1 = Enable UART TX. UTX pin transmits UART data.

Bit 3 **UTXPEN:** UART TX parity bit check function control bit.
0 = Disable UART TX parity bit check function. The data stream doesn't include parity bit.
1 = Enable UART TX parity bit check function. The data stream includes parity bit.

Bit 2 **UTXPS:** UART TX parity bit format control bit.
0 = UART TX parity bit format is even parity.
1= UART TX parity bit format is odd parity.

Bit 1 **UTXM:** UART TX data buffer length control bit.
0 = 1-byte.
1 = 2-byte.

10.4 UART RECEIVER CONTROL REGISTER

URRX initial value = 0000000x

0AAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRX	URXEN	URXS1	URXS0	URXPEN	URXPS	URXPC	URXM	
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 **URXEN:** UART RX control bit.
0 = Disable UART RX. URX pin keeps and returns to GPIO function.
1 = Enable UART RX. URX pin receives UART data.

Bit[6:5] **URXS1, URXS0:** UART RX status indicator.
00 = No data received.
01 = Data received, but parity checking error occurrence.
10, 11 = Data received successfully.

Bit 4 **URXPEN:** UART RX parity bit check function control bit.
0 = Disable UART RX parity bit check function. The data stream doesn't include parity bit.
1 = Enable UART RX parity bit check function. The data stream includes parity bit.

Bit 3 **URXPS:** UART RX parity bit format control bit.
0 = UART RX parity bit format is even parity.
1= UART RX parity bit format is odd parity.

Bit 2 **URXPC:** UART RX parity bit checking status bit.
0 = UART RX parity bit checking is error.
1 = UART RX parity bit checking is correct.

Bit 1 **URXM:** UART RX data buffer length control bit.
0 = 1-byte.
1 = 2-byte.

10.5 UART BAUD RATE CONTROL REGISTER

URBRC initial value =11010101

0ABH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URBRC	UDIV4	UDIV3	UDIV2	UDIV1	UDIV0	UPCS2	UPCS1	UPCS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	1	1	0	1	0	1	0	1

Bit[7:3] **UDIV[4:0]:** UART baud rate divider.

Bit[2:0] **UPCS[2:0]:** UART baud rate pre-scalar.

The UART baud rate clock source is Fhosc and divided by pre-scalar and divider. The equation is as following.

$$UART\ Baud\ Rate = (Fhosc / 2^{PreScalar} / (Divider+1)) / 16 \quad \in Divider \neq 0$$

Baud Rate	UART Clock Source = 16MHz (UCLKS = 0)		UART Clock Source = 24MHz (UCLKS = 1)	
	UPCS[2:0]	UDIV[4:0]	UPCS[2:0]	UDIV[4:0]
1200	101	11010	-	-
2400	100	11010	-	-
4800	011	11010	-	-
9600	010	11010	-	-
19200	001	11001	-	-
38400	000	11001	-	-
51200	000	10011	-	-
57600	000	10000	-	-
102400	000	01001	-	-
115200	-	-	000	01100

10.6 UART DATA BUFFER

URTXD1 initial value = 00000000

0ACH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTXD1	UTXD17	UTXD16	UTXD15	UTXD14	UTXD13	UTXD12	UTXD11	UTXD10
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] **URTXD1:** UART transmitted data buffer byte 1.

URTXD2 initial value = 00000000

0ADH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTXD2	UTXD27	UTXD26	UTXD25	UTXD24	UTXD23	UTXD22	UTXD21	UTXD20
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] **URTXD2:** UART transmitted data buffer byte 2.

URRXD1 initial value = 00000000

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRXD1								
Read/Write	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

Bit[7:0] **URRXD1:** UART received data buffer byte 1.

URRXD2 initial value = 00000000

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRXD2								
Read/Write	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

Bit[7:0] **URRXD2:** UART received data buffer byte 2.

UART Data Mode	URTXD2	URTXD1	URRXD2	URRXD1
1-byte	0x00	1-byte data	0x00	1-byte data
2-byte	High-byte data	Low-byte data	High-byte data	Low-byte data

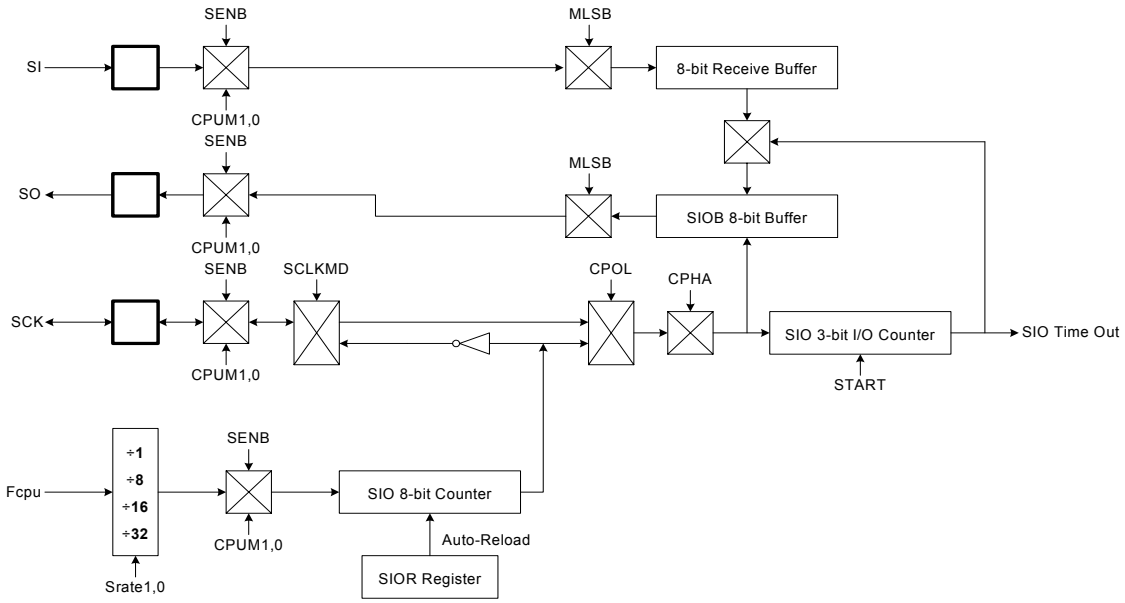
11 SERIAL INPUT/OUTPUT TRANSCEIVER

11.1 OVERVIEW

The SIO (serial input/output) transceiver allows high-speed synchronous data transfer between the SN8F2280 series MCU and peripheral devices or between several SN8F2280 devices. These peripheral devices may be Serial EEPROMs, shift registers, display drivers, etc. The SN8F2280 SIO features include the following:

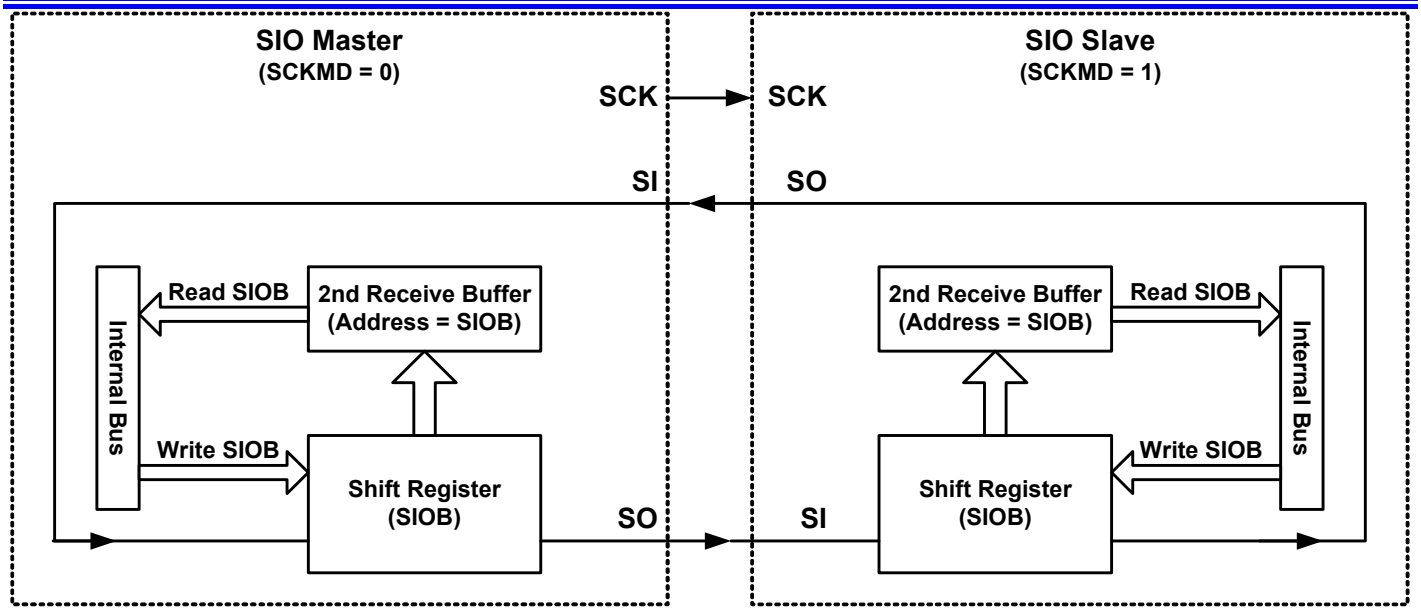
- **Full-duplex, 3-wire synchronous data transfer**
- **TX/RX or TX Only mode**
- **Master (SCK is clock output) or Slave (SCK is clock input) operation**
- **MSB/LSB first data transfer**
- **SDO (P2.1) is programmable open-drain output pin for multiple salve devices application**
- **Two programmable 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 SN8F2280 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	bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0
0	1	1	High	bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0
0	0	0	Low	bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0 Next data
0	1	0	High	bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0 Next data
1	0	1	Low	bit0 bit1 bit2 bit3 bit4 bit5 bit6 Bit7
1	1	1	High	bit0 bit1 bit2 bit3 bit4 bit5 bit6 Bit7
1	0	0	Low	bit0 bit1 bit2 bit3 bit4 bit5 bit6 Bit7 Next data
1	1	0	High	bit0 bit1 bit2 bit3 bit4 bit5 bit6 Bit7 Next data

SIO Data Transfer Timing

11.2 SIOM MODE REGISTER

0B0H	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 (P2.0~P2.2 is general purpose I/O port).
1 = Enable (P2.0~P2.2 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/2.
01 = Fcpu/64
10 = Fcpu/32
11 = Fcpu/16.

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 Port2 for P2.0 as SCK, P2.1 as SDO and P2.2 as SDI.
The following table shown the Port2[2:0] I/O mode behavior and setting when SIO function enable and disable.

SENB=1 (SIO Function Enable)		
P2.0/SCK	(SCKMD=1) SIO source = External clock	P2.0 will change to Input mode automatically, no matter what P2M setting
	(SCKMD=0) SIO source = Internal clock	P2.0 will change to Output mode automatically, no matter what P2M setting
P2.1/SDO	SIO = Transmitter/Receiver	P2.1 will change to Output mode automatically, no matter what P2M setting
P2.2/SDI	P2.2 must be set as Input mode in P2M ,or the SIO function will be abnormal	
SENB=0 (SIO Function Disable)		
P2.0/P2.1/P2.2	Port2[2:0] I/O mode are fully controlled by P2M when SIO function is disable	

11.3 SIOB DATA BUFFER

0B2H	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.

11.4 SIOR REGISTER DESCRIPTION

0B1H	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. 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 2MHz. Fosc = 12MHz. SIO's rate = Fcpu/2. Fcpu = Fosc/1 = 12MHz.

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

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

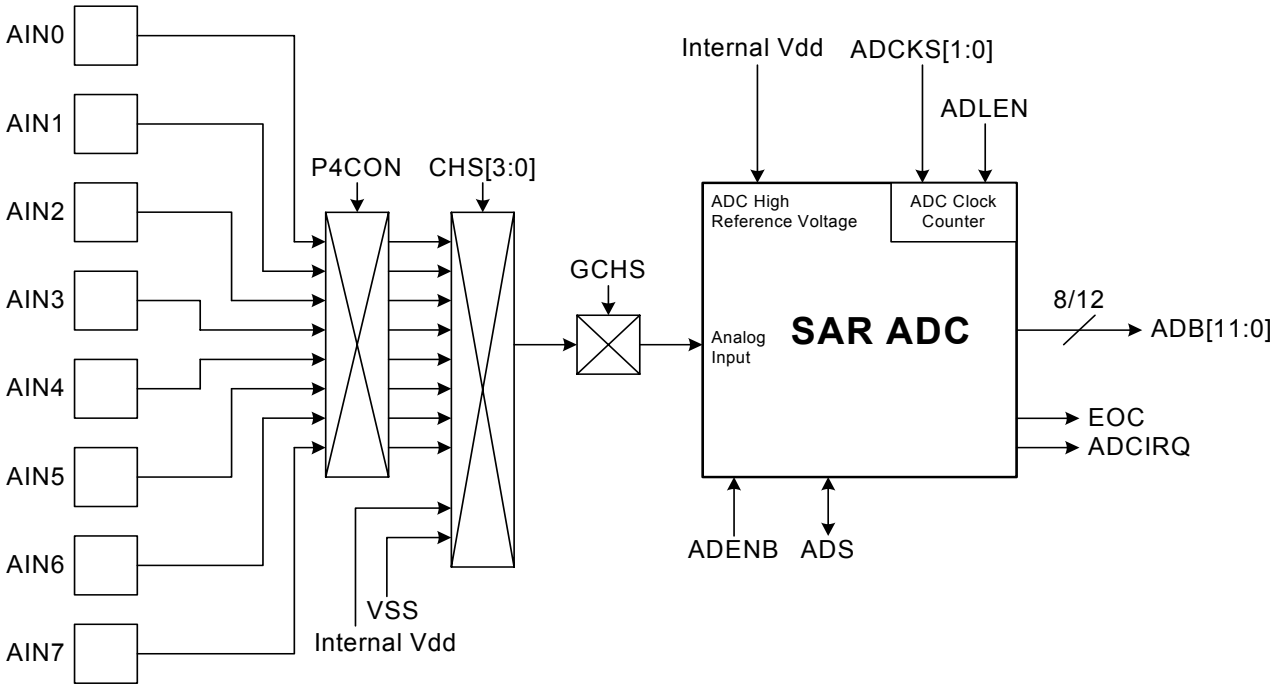
```
        MOV     A,TXDATA        ; Load transmitted data into SIOB register.
        B0MOV   SIOB,A
        MOV     A,#0FEH         ; Set SIO clock
        B0MOV   SIOR,A
        MOV     A,#10000000B    ; Setup SIOM and enable SIO function.
        B0MOV   SIOM,A
        B0BSET  FSTART          ; 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
```

12 8 CHANNEL ANALOG TO DIGITAL CONVERTER

12.1 OVERVIEW



The analog to digital converter (ADC) is SAR structure with 8-input sources and up to 4096-step resolution to transfer analog signal into 12-bits digital data. Use CHS[2:0] bits to select analog signal input pin (AIN pin), and GCHS bit enables global ADC channel, the analog signal inputs to the SAR ADC. The ADC resolution can be selected 8-bit and 12-bit resolutions through ADLEN bit. The ADC converting rate can be selected by ADCKS[1:0] bits to decide ADC converting time. The ADC also builds in P4CON register to set pure analog input pin. It is necessary to set P4 as input mode with pull-up resistor by program. After setup ADENB and ADS bits, the ADC starts to convert analog signal to digital data. When the conversion is complete, the ADC circuit will set EOC and ADCIRQ bits to “1” and the digital data outputs in ADB and ADR registers. If the ADCIEN = 1, the ADC interrupt request occurs and executes interrupt service routine when ADCIRQ = 1 after ADC converting.

✷ ***Note:***
1. Set ADC input pin I/O direction as input mode without pull-up resistor.
2. Disable ADC (set ADENB = “0”) before enter power down (sleep) mode to save power consumption.
3. Set related bit of P4CON register to avoid extra power consumption in power down mode.
4. Delay 100uS after enable ADC (set ADENB = “1”) to wait ADC circuit ready for conversion.

12.2 ADM REGISTER

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0
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 **ADENB:** ADC control bit.
0 = Disable.
1 = Enable.

Bit 6 **ADS:** ADC start control bit .
0 = ADC converting stops.
1 = Start to execute ADC converting.

Bit 5 **EOC:** ADC status bit.
0 = ADC progressing.
1 = End of converting and reset ADS bit.

Bit 4 **GCHS:** ADC global channel select bit.
0 = Disable AIN channel.
1 = Enable AIN channel.

Bit [3:0] **CHS[3:0]:** ADC input channel select bit.
0000 = AIN0. 0001 = AIN1.
0010 = AIN2. 0011 = AIN3.
0100 = AIN4. 0101 = AIN5.
0110 = AIN6. 0111 = AIN7.
1000 = VSS. 1001 = VDD.

12.3 ADR REGISTERS

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	ADCKS2	ADCKS1	ADCKS0	ADLEN	ADB3	ADB2	ADB1	ADB0
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
After reset	0	0	0	0	-	-	-	-

Bit [7:5] **ADCKS [2:0]:** ADC's clock source select bit.

ADCKS2	ADCKS1	ADCKS0	ADC Clock Source
0	0	0	Fcpu/16
0	0	1	Fcpu/8
0	1	0	Fcpu/1
0	1	1	Fcpu/2
1	0	0	Fcpu/64
1	0	1	Fcpu/32
1	1	0	Fcpu/4
1	1	1	Reserved

Bit 4 **ADLEN:** ADC's resolution select bits.
0 = 8-bit.
1 = 12-bit.

Bit [3:0] **ADB [3:0]:** 12-bit low-nibble ADC data buffer.

12.4 ADB REGISTERS

0B7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
Read/Write	R	R	R	R	R	R	R	R
After reset	-	-	-	-	-	-	-	-

Bit[7:0] **ADB[7:0]:** 8-bit ADC data buffer and the high-byte data buffer of 12-bit ADC.

ADB is ADC data buffer to store ADC converter result. The ADB register is only 8-bit register including bit 4~bit11 ADC data. To combine ADB register and the low-nibble of ADR will get full 12-bit ADC data buffer. The ADC buffer is a read-only register and the initial status is unknown after system reset.

- **ADB[11:4]: In 8-bit ADC mode, the ADC data is stored in ADB register.**
- **ADB[11:0]: In 12-bit ADC mode, the ADC data is stored in ADB and ADR registers.**

✷ ***Note: The initial status of ADC data buffer including ADB register and ADR low-nibble after the system reset is unknown.***

The AIN's input voltage v.s. ADB's output data

AIN n	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
0/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	0
1/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	1
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
4094/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	0
4095/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	1

For different applications, users maybe need more than 8-bit resolution but less than 12-bit. To process the ADB and ADR data can make the job well. First, the AD resolution must be set 12-bit mode and then to execute ADC converter routine. Then delete the LSB of ADC data and get the new resolution result. The table is as following.

ADC Resolution	ADB								ADR			
	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
8-bit	O	O	O	O	O	O	O	O	x	x	x	x
9-bit	O	O	O	O	O	O	O	O	O	x	x	x
10-bit	O	O	O	O	O	O	O	O	O	O	x	x
11-bit	O	O	O	O	O	O	O	O	O	O	O	x
12-bit	O	O	O	O	O	O	O	O	O	O	O	O
O = Selected, x = Delete												

12.5 P4CON REGISTERS

The Port 4 is shared with ADC input function. Only one pin of port 4 can be configured as ADC input in the same time by ADM register. The other pins of port 4 are digital I/O pins. Connect an analog signal to COMS digital input pin, especially, the analog signal level is about 1/2 VDD will cause extra current leakage. In the power down mode, the above leakage current will be a big problem. Unfortunately, if users connect more than one analog input signal to port 4 will encounter above current leakage situation. P4CON is Port4 Configuration register. Write "1" into P4CON [7:0] will configure related port 4 pin as pure analog input pin to avoid current leakage.

0B9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Bit[4:0] **P4CON[7:0]:** P4.n configuration control bits.
0 = P4.n can be an analog input (ADC input) or digital I/O pins.
1 = P4.n is pure analog input, can't be a digital I/O pin.

✷ ***Note: When Port 4.n is general I/O port not ADC channel, P4CON.n must set to "0" or the Port 4.n digital I/O signal would be isolated.***

12.6 ADC CONVERTING TIME

The ADC converting time is from ADS=1 (Start to ADC convert) to EOC=1 (End of ADC convert). The converting time duration is depend on ADC resolution. 12-bit ADC's converting time is 1/(ADC clock /4)*16 sec, and the 8-bit ADC converting time is 1/(ADC clock /4)*12 sec. ***The ADC converting time affects ADC performance. If input high rate analog signal, it is necessary to select a high ADC converting rate. If the ADC converting time is slower than analog signal variation rate, the ADC result would be error. So to select a correct ADC clock rate and ADC resolution to decide a right ADC converting rate is very important.***

12-bit ADC conversion time = 1/(ADC clock /4)*16 sec

8-bit ADC conversion time = 1/(ADC clock /4)*12 sec

Fcpu = 4MHz (High clock oscillator frequency (Fosc) is 16MHz and Fcpu = Fosc/4)

ADLEN	ADCKS1	ADCKS0	ADC Clock	ADC conversion time
0 (8-bit)	0	0	Fcpu/16	1/(4MHz/16/4)*12 = 192 us
	0	1	Fcpu/8	1/(4MHz/8/4)*12 = 96 us
	1	0	Fcpu	1/(4MHz/4)*12 = 12 us
	1	1	Fcpu/2	1/(4MHz/2/4)*12 = 24 us
1 (12-bit)	0	0	Fcpu/16	1/(4MHz/16/4)*16 = 256 us
	0	1	Fcpu/8	1/(4MHz/8/4)*16 = 128 us
	1	0	Fcpu	1/(4MHz/4)*16 = 16 us
	1	1	Fcpu/2	1/(4MHz/2/4)*16 = 32 us

12.7 ADC CONTORL NOTICE

12.7.1 ADC SIGNAL

The ADC high reference voltage is internal Vdd. The ADC low reference voltage is ground.
The ADC input signal voltage range must be from high reference voltage to low reference voltage.

12.7.2 ADC PROGRAM

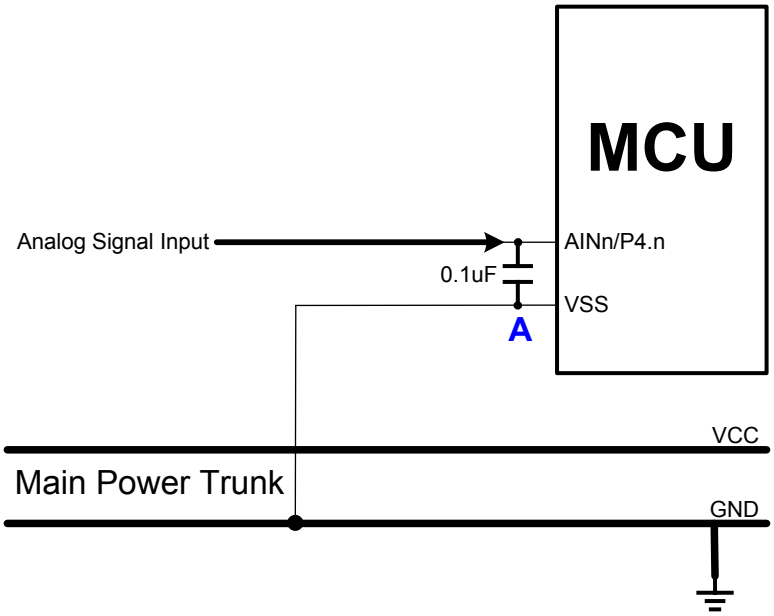
The first step of ADC execution is to setup ADC configuration. The ADC program setup sequence and notices are as following.

- **Step 1:** Enable ADC. ADENB is ADC control bit to control. ADENB = 1 is to enable ADC. ADENB = 0 is to disable ADC. ***When ADENB is enabled, the system must be delay 100us to be the ADC warm-up time by program, and then set ADS to do ADC converting. The 100us delay time is necessary after ADENB setting (not ADS setting), or the ADC converting result would be error.*** Normally, the ADENB is set one time when the system under normal run condition, and do the delay time only one time.
- **Step 2:** Select the ADC input pin by CHS[2:0], enable P4CON's related bit for the ADC input pin, and enable ADC global input. **When one AIN pin is selected to be analog signal input pin, it is necessary to setup the pin as input mode and disable the pull-up resistor by program.** ***Also to set the P4CON, and the digital I/O function including pull-up is isolated.***
- **Step 3:** Start to execute ADC conversion by setting ADS = 1.
- **Step 4:** Wait the end of ADC converting through checking EOC = 1 or ADCIRQ = 1. If enable ADC interrupt function, the program executes ADC interrupt service when ADC interrupt occurrence. ***ADS is cleared when the end of ADC converting automatically. EOC bit indicates ADC processing status immediately and is cleared when ADS = 1. Users needn't to clear it by program.***

➢ **Example : Configure AIN0 as 12-bit ADC input and start ADC conversion then enter power down mode.**

```
ADC0:
            B0BSET      FADENB          ; Enable ADC circuit
            CALL        Delay100uS      ; Delay 100uS to wait ADC circuit ready for conversion
            MOV         A, #0FEh
            B0MOV       P4UR, A         ; Disable P4.0 pull-up resistor
            B0BCLR      FP40M           ; Set P4.0 as input pin
            MOV         A, #01h
            B0MOV       P4CON, A        ; Set P4.0 as pure analog input
            MOV         A, #60H
            B0MOV       ADR, A          ; To set 12-bit ADC and ADC clock = Fosc.
            MOV         A,#90H
            B0MOV       ADM,A           ; To enable ADC and set AIN0 input
            B0BSET      FADS            ; To start conversion
WADC0:
            B0BTS1      FEOC            ; To skip, if end of converting =1
            JMP         WADC0           ; else, jump to WADC0
            B0MOV       A,ADB           ; To get AIN0 input data bit11 ~ bit4
            B0MOV       Adc_Buf_Hi, A
            B0MOV       A,ADR           ; To get AIN0 input data bit3 ~ bit0
            AND         A, 0Fh
            B0MOV       Adc_Buf_Low, A
Power_Down  .           .
            B0BCLR      FADENB          ; Disable ADC circuit
            B0BCLR      FCPUM1
            B0BSET      FCPUM0          ; Enter sleep mode
```

12.8 ADC CIRCUIT



The analog signal is inputted to ADC input pin “AINn/P4.n”. The ADC input signal must be through a 0.1uF capacitor “A”. The 0.1uF capacitor is set between ADC input pin and VSS pin, and must be on the side of the ADC input pin as possible. Don’t connect the capacitor’s ground pin to ground plain directly, and must be through VSS pin. The capacitor can reduce the power noise effective coupled with the analog signal.

13 Main Series Port (MSP)

13.1 OVERVIEW

The MSP (Main Serial Port) is a serial communication interface for data exchanging from one MCU to one MCU or other hardware peripherals. These peripheral devices may be serial EEPROM, A/D converters, Display device, etc.

The MSP module can operate in one of two modes:

- **Full Master Mode**
- **Slave mode (with general address call)**

The MSP features include the following:

- **2-wire synchronous data transfer / receiver.**
- **Master (SCL is clock output) or Slave (SCL is clock input) operation.**
- **SCL, SDA are programmable open-drain output pin for multiple salve devices application.**
- **Support 400K clock rate @ Fcpu=4MIPs.**
- **End-of-Transfer/Receiver interrupt.**

13.2 MSP STATUS REGISTER

MSPSTAT initial value =x00000x0

0EAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPSTAT		CKE	D_A	P	S	RED_WRT		BF
Read/Write		R/W	R	R	R	R		R
After Reset		0	0	0	0	0		0

Bit 6 **CKE:** Slave Clock Edge Control bit.
In Slave Mode: Receive Address or Data byte.
0 = Latch Data on SCL Rising Edge. (Default)
1 = Latch Data on SCL Falling Edge.

✷ ***Note 1. In Slave Transmit mode, Address Received depended on CKE setting. Data Transfer on SCL Falling Edge.***
✷ ***Note 2. In Slave Receiver mode, Address and Data Received depended on CKE setting.***

Bit 5 **D_A:** Data/Address bit.
0 = Indicates the last byte received or transmitted was address.
1 = Indicates the last byte received or transmitted was data.

Bit 4 **P:** Stop bit.
0 = Stop bit was not detected.
1 = Indicates that a stop bit has been detected last.

✷ ***Note1. It will be cleared when Start bit was detected.***

Bit 3 **S:** Start bit.
0 = Start bit was not detected.
1 = Indicates that a start bit has been detected last.

✷ ***Note1. It will be cleared when Stop bit was detected.***

Bit 2 **RED_WRT:** Read/Write bit information.

This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next start bit, stop bit, or not ACK bit.

In slave mode:
0 = Write.
1 = Read.
In master mode:
0 = Transmit is not in progress.
1 = Transmit is in progress.

Or this bit with SEN, RSEN, PEN, RCEN, or ACKEN will indicate if the MSP is in IDLE mode.

Bit 0 **BF:** Buffer Full Status bit.

Receive:
1 = Receive complete, MSPBUF is full.
0 = Receive not complete, MSPBUF is empty.
Transmit:
1 = Data Transmit in progress (does not include the ACK and stop bits), MSPBUF is full.
0 = Data Transmit complete (does not include the ACK and stop bits), MSPBUF is empty.

13.3 MSP MODE REGISTER 1

MSPM1 initial value =000000x0

0EBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPM1	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK		MSPC
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 **WCOL:** Write Collision Detect bit.

Master Mode:
0 = No collision.
1 = A write to the MSPBUF register was attempted while the MSP conditions were not valid for a transmission to be started.

Slave Mode:
0 = No collision.
1 = The MSPBUF register is written while it is still transmitting the previous word. (must be cleared in software)

Bit 6 **MSPOV:** Receive Overflow Indicator bit.
0 = No overflow.
1 = A byte is received while the MSPBUF register is still holding the previous byte. MSPOV is a “don’t care” in transmit mode. MSPOV must be cleared in software in either mode. (must be cleared in software)

Bit 5 **MSPENB:** Main Serial Port Communication Enable.
0 = Disables Main Serial Port and configures these pins as I/O port pins.
1 = Enables Main Serial Port and configures SCL and SDA as the source of the serial port pins.

Bit 4 **CKP:** SCL Clock Priority Control bit.

In MSP Slave mode:
0 = Hold SCL keeping Low. (Ensure data setup time and Slave device ready)
1 = Release SCL Clock.
(Slave Transistor mode CKP function always enables, Slave Receiver CKP function control by SLRXCKP)

In MSP Master mode:
Unused.

Bit 3 **SLRXCKP:** Slave Receiver mode SCL Clock Priority Control bit.

In MSP Slave Receiver mode:
0 = Disable CKP function.
1 = Enable CKP function.

In MSP Master and Slave Transistor mode:
Unused.

Bit 2 **MSPWK:** MSP Wake-up indication bit.
0 = MCU NOT wake-up by MSP.
1 = MCU wake-up by MSP.

✸ ***Note: Clear MSPWK before entering Power down mode for indication the wake-up source from MSP or not.***

Bit 0 **MSPC:** MSP mode Control register.
0 = MSP operated on Slave mode, 7-bit address.
1 = MSP operated on Master mode.

13.4 MSP MODE REGISTER 2

MSPM2 initial value =00000000

0ECH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPM2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
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 **GCEN:** General Call Enable bit. (In Slave mode only)
0 = General call address disabled.
1 = Enable interrupt when a general call address (0000h) is received.

Bit 6 **ACKSTAT:** Acknowledge Status bit. (In master mode only)
In master transmit mode:
0 = Acknowledge was received from slave.
1 = Acknowledge was not received from slave.

Bit 5 **ACKDT:** Acknowledge Data bit. (In master mode only)
In master receive mode:
Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
0 = Acknowledge.
1 = Not Acknowledge.

Bit 4 **ACKEN:** Acknowledge Sequence Enable bit. (In MSP master mode only)
In master receive mode:
0 = Acknowledge sequence idle.
1 = Initiate Acknowledge sequence on SDA and SCL pins, and transmit ACKDT data bit. Automatically cleared by hardware.

✷ ***Note: If the MSP module is not in the idle mode, this bit may not be set (no spooling), and the MSPBUF may not be written (or writes to the MSPBUF are disabled).***

Bit 3 **RCEN:** Receive Enable bit. (In master mode only)
0 = Receive idle.
1 = Enables Receive mode for MSP.

✷ ***Note: If the MSP module is not in the idle mode, this bit may not be set (no spooling), and the MSPBUF may not be written (or writes to the MSPBUF are disabled).***

Bit 2 **PEN:** Stop Condition Enable bit. (In master mode only)
0 = Stop condition idle.
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.

✷ ***Note: If the MSP module is not in the idle mode, this bit may not be set (no spooling), and the MSPBUF may not be written (or writes to the MSPBUF are disabled).***

Bit 1 **RSEN:** Repeated Start Condition Enabled bit. (In master mode only)
0 = Repeated Start condition idle.
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.

✷ ***Note: If the MSP module is not in the idle mode, this bit may not be set (no spooling), and the MSPBUF may not be written (or writes to the MSPBUF are disabled).***

Bit 0 **SEN:** Start Condition Enabled bit. (In master mode only)
0 = Start condition idle.
1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.

✷ ***Note: If the MSP module is not in the idle mode, this bit may not be set (no spooling), and the MSPBUF may not be written (or writes to the MSPBUF are disabled).***

13.5 MSP BUFFER REGISTER

MSPBUF initial value =00000000

0EDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPBUF	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0
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] **MSPBUF:** MSP Buffer.

13.6 MSP ADDRESS REGISTER

MSPADR initial value =00000000

0EEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPADR	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0
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] **MSPADR:** MSP Address.

13.7 Slave Mode Operation

When an address is matched or data transfer after and address match is received, the hardware automatically will generate the acknowledge (ACK_) signal, and load MSPBUF (MSP buffer register) with the received data from MSPSR.

There are some conditions that will cause MSP function will not reply ACK_ signal:

- **Data Buffer already full: BF=1 (MSPSTAT bit 0), when another transfer was received.**
- **Data Overflow: MSPOV=1 (MSPM1 bit 6), when another transfer was received.**

When BF=1, means MSPBUF data is still not read by MCU, so MSPSR will not load data into MSPBUF, but MSPIRQ and MSPOV bit will still set to 1. BF bit will be clear automatically when reading MSPBUF register. MSPOV bit must be clear through by Software.

13.7.1 Addressing

When MSP Slave function has been enabled, it will wait a START signal occur. Following the START signal, 8-bit address will shift into the MSPSR register. The data of MSPSR[7:1] is compare with MSPADR register on the falling edge of eight SCL pulse, If the address are the same, the BF and MSPOV bit are both clear, the following event occur:

1. MSPSR register is loaded into MSPBUF on the falling edge of eight SCL pulse.
2. Buffer full bit (BF) is set to 1, on the falling edge of eight SCL pulse.
3. An ACK_ signal is generated.
4. MSP interrupt request MSPIRQ is set on the falling edge of ninth SCL pulse.

Status when Data is Received		MSPSP→ MSPBUF	Reply an ACK_ signal	Set MSPIRQ
BF	MSPOV			
0	0	Yes	Yes	Yes
*0	*1	Yes	No	Yes
1	0	No	No	Yes
1	1	No	No	Yes

Data Received Action Table

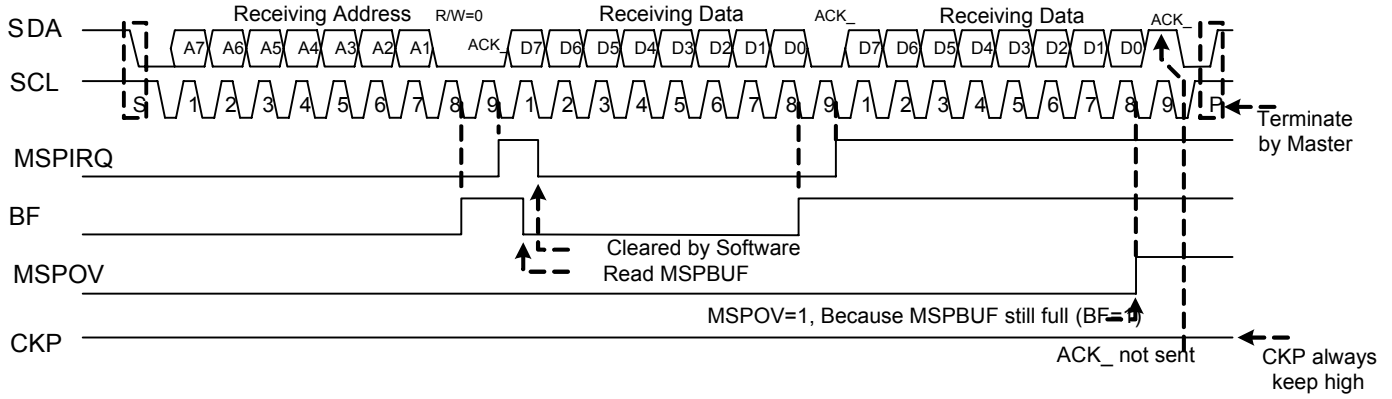
Note: BF=0, MSPOV=1 shows the software is not set properly to clear Overflow register.

13.7.2 Slave Receiving

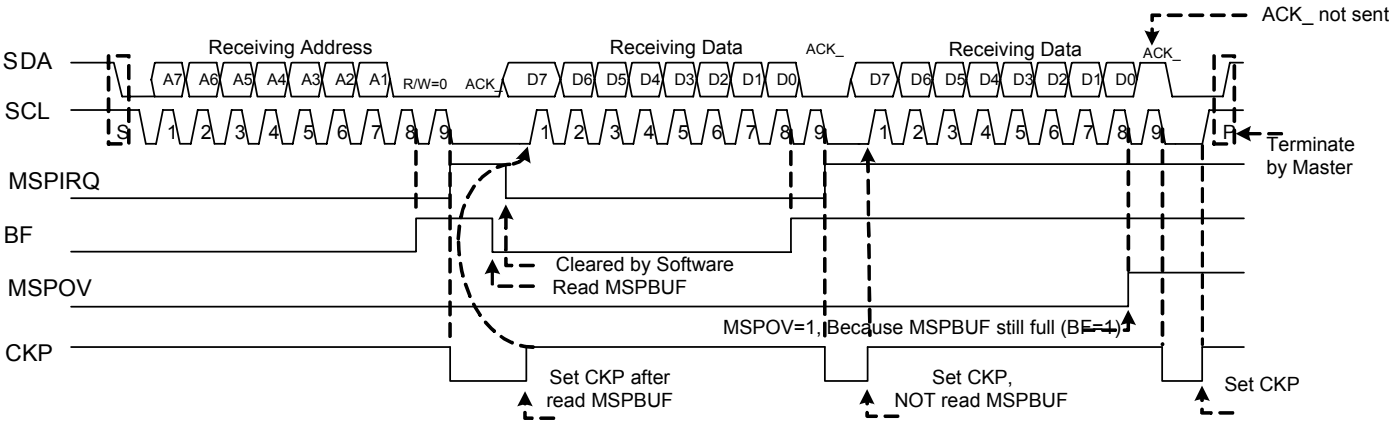
When the R/W bit of address byte =0 and address is matched, the R/W bit of MSPSTAT is cleared. The address will be load into MSPBUF. After reply an ACK_ signal, MSP will receive data every 8 clock. The CKP function enable or disable (Default) is controlled by SLRXCKP bit and data latch edge -Rising edge (Default) or Falling edge is controlled by CKE bit.
When overflow occur, no acknowledge signal replied which either BF=1 or MSPOV=1.
MSP interrupt is generated in every data transfer. The MSPIRQ bit must be clear by software.

Following is the Slave Receiving Diagram:

SLRXCKP=0

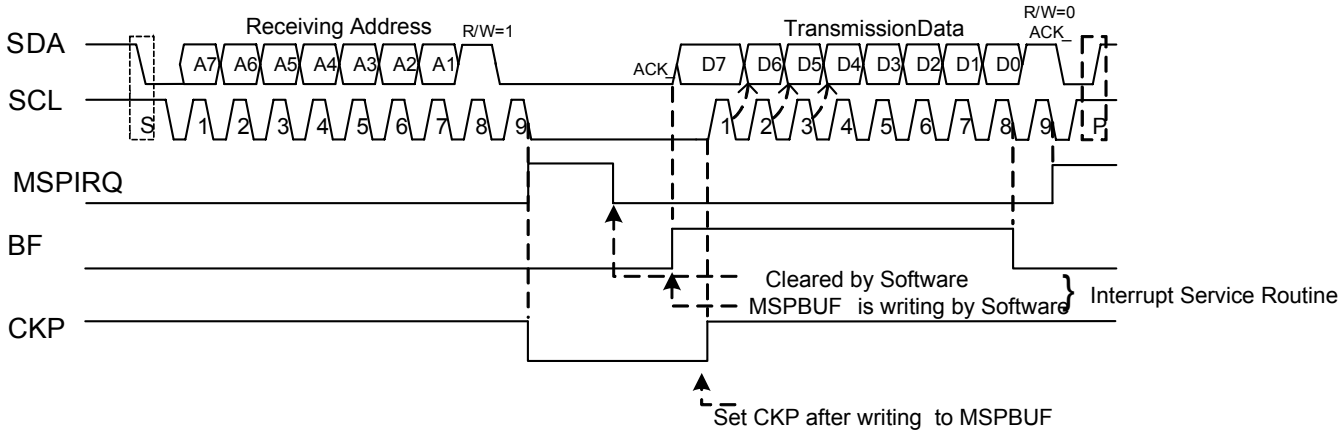


SLRXCKP=1



13.7.3 Slave Transmission

After address match, the following R/W bit is set, MSPSTAT bit 2 R/W will be set. The received address will be load to MSPBUF and ACK_ will be sent at ninth clock then SCL will be hold low. Transmission data will be load into MSPBUF which also load to MSPSR register. The Master should monitor SCL pin signal. The slave device may hold on the master by keep CKP low. When set. After load MSPBUF, set CKP bit, MSPBUF data will shift out on the falling edge on SCL signal. This will ensure the SDA signal is valid on the SCL high duty.
An MSP interrupt is generated on every byte transmission. The MSPIRQ will be set on the ninth clock of SCL. Clear MSPIRQ by software. MSPSTAT register can monitor the status of data transmission.
In Slave transmission mode, an ACK_ signal from master-receiver is latched on rising edge of ninth clock of SCL. If ACK_ = high, transmission is complete. Slave device will reset logic and waiting another START signal. If ACK_= low, slave must load MSPBUF which also MSPSR, and set CKP=1 to start data transmission again.



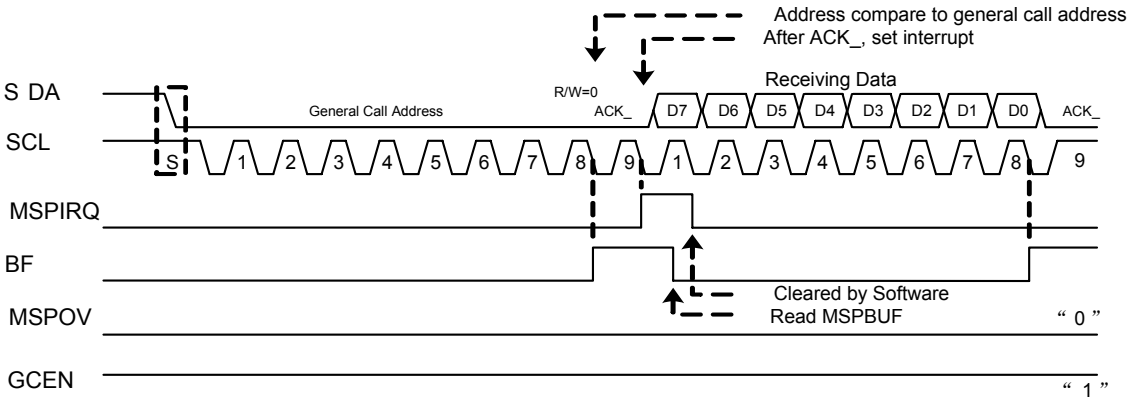
MSP Slave Transmission Timing Diagram

13.7.4 General Call Address

In MSP bus, the first 7-byte is the Slave address. Only the address match MSPADR the Slave will response an ACK_. The exception is the general call address which can address all slave devices. When this address occur, all devices should response an acknowledge.

The general call address is a special address which is reserved as all “0” of 7-bits address. The general call address function is control by GCEN bit. Set this bit will enable general call address and clear it will disable. When GCEN=1, following a START signal, 8-bit will shift into MSPSR and the address is compared with MSPADR and also the general call address which fixed by hardware.

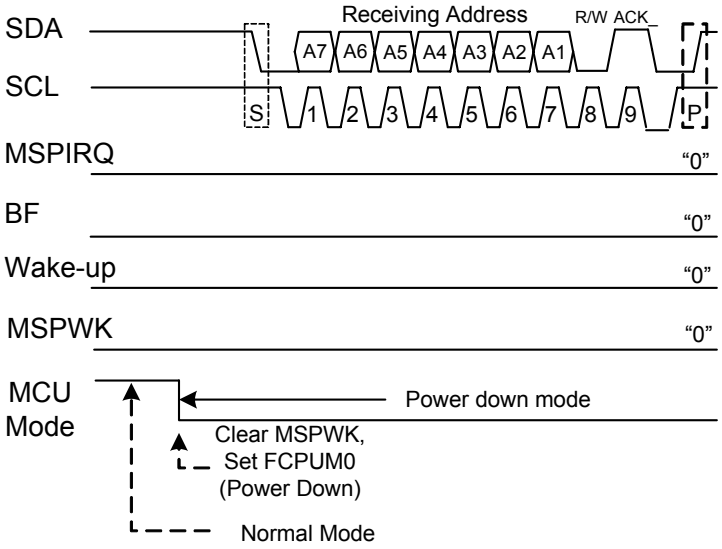
If the general call address matches, the MSPSR data is transferred into MSPBUF, the BF flag bit is set, and in the falling edge of the ninth clock (ACK_) MSPIRQ flag set for interrupt request. In the interrupt service routine, reading MSPBUF can check if the address is the general call address or device specific.



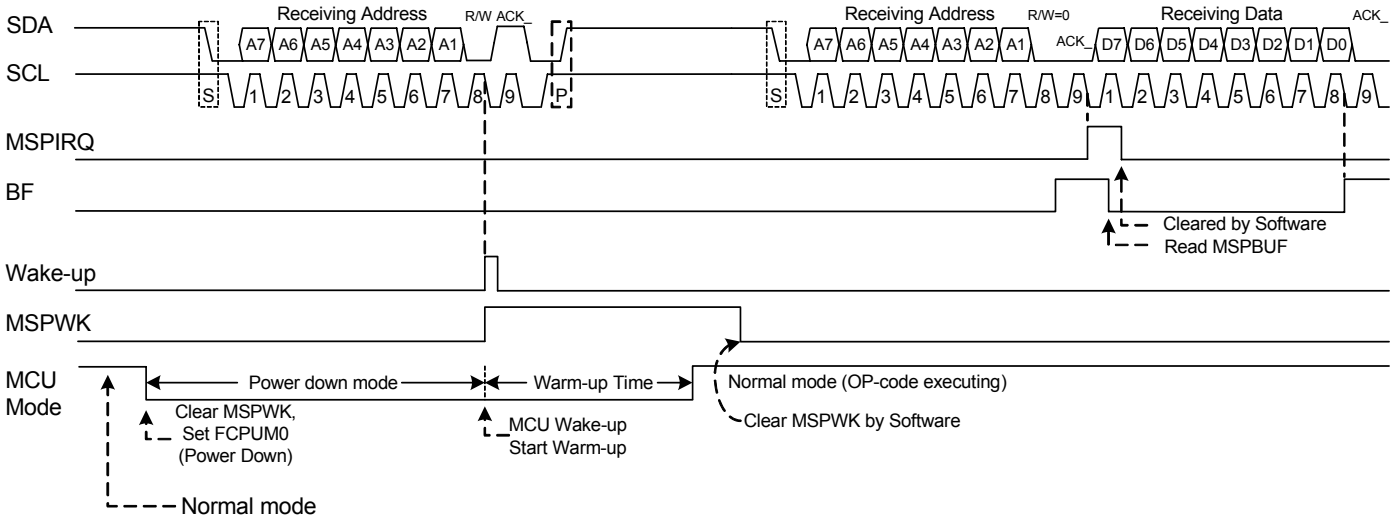
General Call Address Timing Diagram

13.7.5 Slave Wake up

When MCU enter Power down mode, if MSPENB bit is still set, MCU can wake-up by matched device address.
The address of MSP bus following START bit, 8-bits address will shift into MSPSR, if address matched, an NOT Acknowledge will response on the ninth clock of SCL and MCU will be wake-up, MSPWK set and start wake-up procedure but MSPIRQ will not set and MSPSR data will not load to MSPUBF. After MCU finish wake-up procedure, MSP will be in idle status and waiting master's START signal. Control register BF, MSPIRQ, MSPOV and MSPBUF will be the same status/data before power down.
If address not matches, a NOT acknowledge is still sent on the ninth clock of SCL, but MCU will be NOT wake-up and still keep in power down mode.



MSP Wake-up Timing Diagram: Address NOT Matched



MSP Wake-up Timing Diagram: Address Matched

- ***Note: 1. MSP function only can work on Normal mode, when wake-up from power down mode, MCU must operate in Normal mode before Master sent START signal.***
- ***Note:2. In MSP wake-up, if the address not match, MCU will keep in power down mode.***
- ***Note 3. Clear MSPWK before enter power down mode by Software for wake-up indication.***

13.8 Master Mode Operation

Master mode of MSP operation from a START signal and end by STOP signal.
The START (S) and STOP (P) bit are clear when reset or MSP function disabled.
In Master mode the SCL and SDA line are controlled by MSP hardware.

Following events will set MSP interrupt request (MSPIRQ), if MSPIEN set, interrupt occurs.

- **START condition.**
- **STOP condition.**
- **Data byte transmitted or received.**
- **Acknowledge Transmit.**
- **Repeat START.**

13.8.1 Master Mode Support

Master mode enable when MSPC and MSPENB bit set. Once the Master mode enabled, the user had following six options.

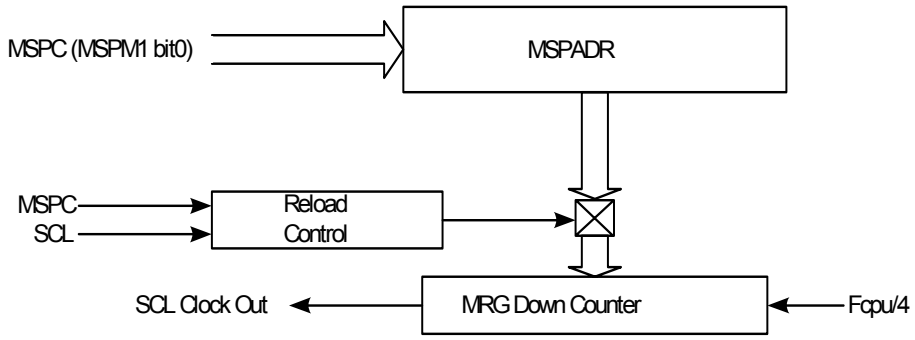
- **Send a START signal on SCL and SDA line.**
- **Send a Repeat START signal on SCL and SDA line.**
- **Write to MSPBUF register for Data or Address byte transmission.**
- **Send a STOP signal on SCL and SDA line.**
- **Configuration MSP port for receive data.**
- **Send an Acknowledge at the end of a received byte of data.**

13.8.2 MSP Rate Generator (MRG)

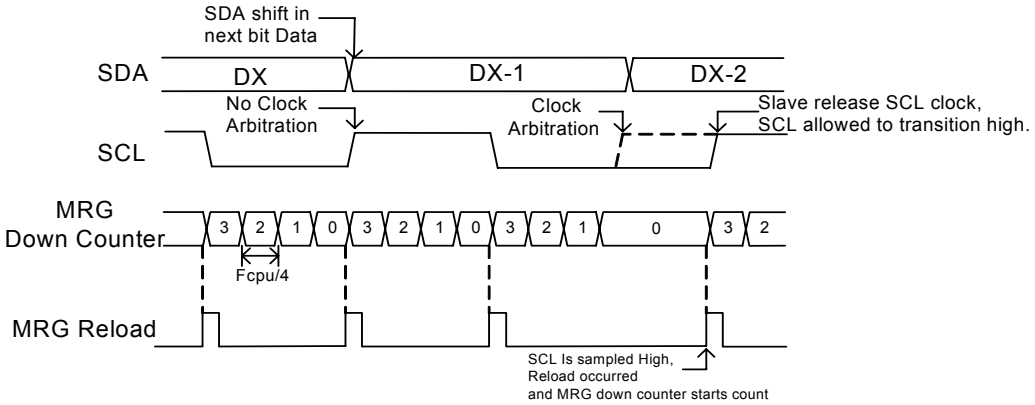
In MSP Mode, the MSP rate generator's reload value is located in the lower 7 bit of MSPADR register. When MRG is loaded with the register, the MRG count down to 0 and stop until another reload has taken place. In MSP mater mode MRG reload from MSPADR automatically. If Clock Arbitration occur for instance (SCL pin keep low by Slave device), the MRG will reload when SCL pin is detected High.

SCL clock rate = Fcpu/(MSPADR)*2

For example, if we want to set 400Khz in 4Mhz Fcpu, the MSPADR have to set 0x05h.
MSPADR=4Mhz/400Khz*2=5



MSP Rate Generator Block Diagram



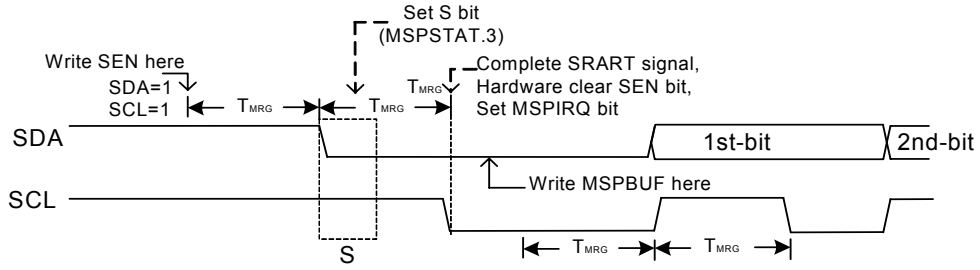
MRG Timing Diagram with and without Clock Arbitration (MSPADR=0x03)

13.8.3 MSP Master Mode START Condition

To generate a START signal, user sets SEN bit (MSPM2.0). When SDA and SCL pin are both sampled High, MSP rate generator reload MSPADR[6:0], and starts down counter. When SDA and SCL are both sampled high and MRG overflow, SDA pin is drive low. When SCL sampled high, and SDA transmitted from High to Low is the START signal and will set S bit (MSPSTAT.3). MRG reload again and start down counter. SEN bit will be clear automatically when MRG overflow, the MRG is suspend leaving SDA line held low, and START condition is complete.

WCOL Status Flag

If user write to MSPBUF when START condition processing, then WCOL is set and the content of MSPBUF data is un-changed. (the writer doesn't occur)



START Condition Timing Diagram

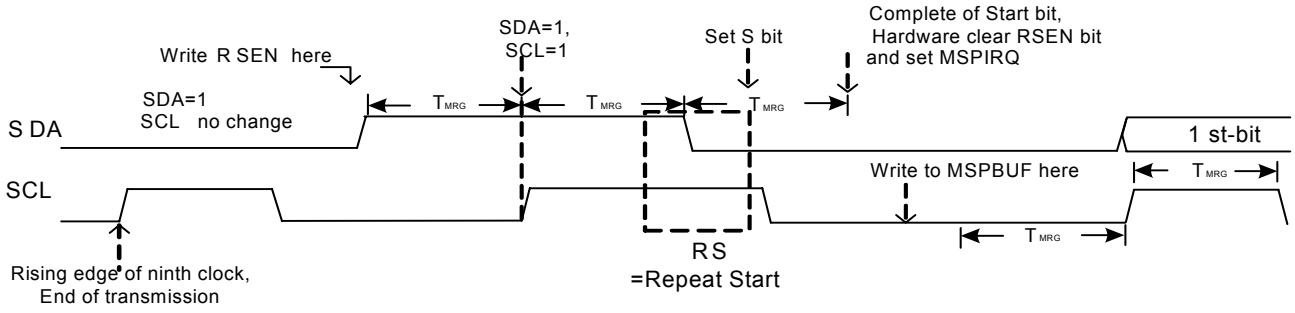
13.8.4 MSP Master Mode Repeat START Condition

When MSP logic module is idle and RSEN set to 1, Repeat Start progress occurs. RSEN set and SCL pin is sampled low, MSPADR[6:0] data reload to MSP rate generator and start down counter. The SDA pin is release to high in one MSP rate generate counter (T_{MRG}). When the MRG is overflow, if SDA is sampled high. SCL will be brought high. When SCL is sampled high, MSPADR reload to MRG and start down counter. SDA and SCL must keep high in one T_{MRG} period. In the next T_{MRG} period, SDA will be brought low when SCL is sampled high, then RSEN will clear automatically by hardware and MRG will not reload, leaving SDA pin held low. Once detect SDA and SCL occur START condition, the S bit will be set (MSPSTAT.3). MSPIRQ will not set until MRG overflow.

- ***Note: 1. While any other event is progress, Set RSEN will take no effect.***
- ***Note:2. A bus collision during the Repeat Start condition occur:***
  SDA is sampled low when SCL goes from low to high.

WCOL Status Flag

If user write to MSPBUF when Repeat START condition processing, then WCOL is set and the content of MSPBUF data is un-changed. (the writer doesn't occur)



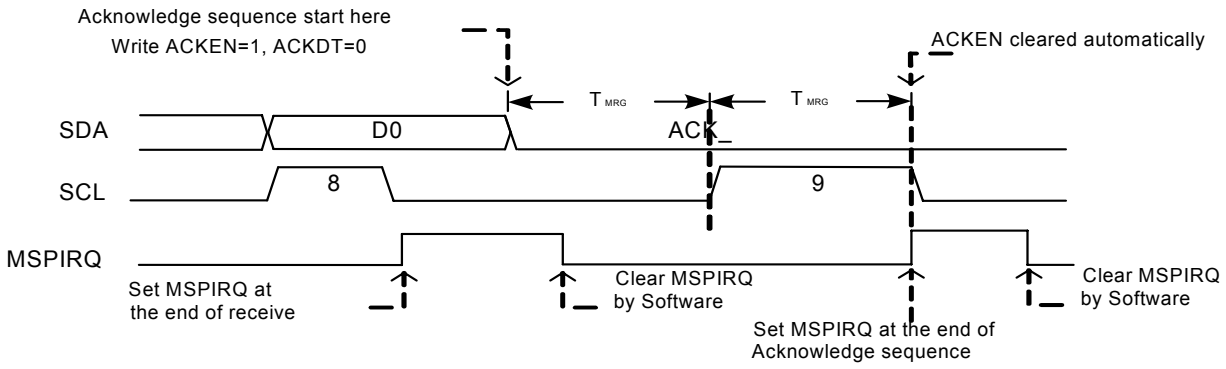
Repeat Start Condition Timing Diagram

13.8.5 Acknowledge Sequence Timing

An acknowledge sequence is enabled when set ACKEN (MSPM2.4). SCL is pulled low when set ACKEN and the content of the acknowledge data bit is present on SDA pin. If user wished to reply an acknowledge, ACKDT bit should be cleared. If not, set ACKDT bit before starting a acknowledge sequence. SCL pin will be release (brought high) when MSP rate generator overflow. MSP rate generator start a T_{MRG} period down counter, when SCL is sampled high. After this period, SCL is pulled low, and ACKEN bit is clear automatically by hardware. When next MRG overflow again, MSP goes into idle mode.

WCOL Status Flag

If user write to MSPBUF when Acknowledge sequence processing, then WCOL bit is set and the content of MSPBUF data is un-changed. (the writer doesn't occur)



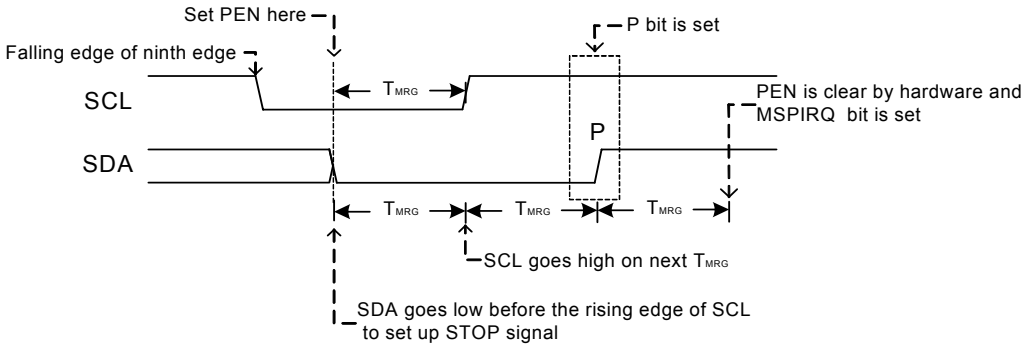
Acknowledge Sequence Timing Diagram

13.8.6 MSP Master Mode STOP Condition Timing

At the end of received/transmitted, a STOP signal present on SDA pin by setting the STOP bit register, PEN (MSPM2.2). At the end of receive/transmit, SCL goes low on the failing edge of ninth clock. Master will set SDA go low, when set PEN bit. When SDA is sampled low, MSP rate generator is reloaded and start count down to 0. When MRG overflow, SCL pin is pull high. After one T_{MRG} period, SDA goes High. When SDA is sampled high while SCL is high, bit P is set. PEN bit is clear after next one T_{MRG} period, and MSPIRQ is set.

WCOL Status Flag

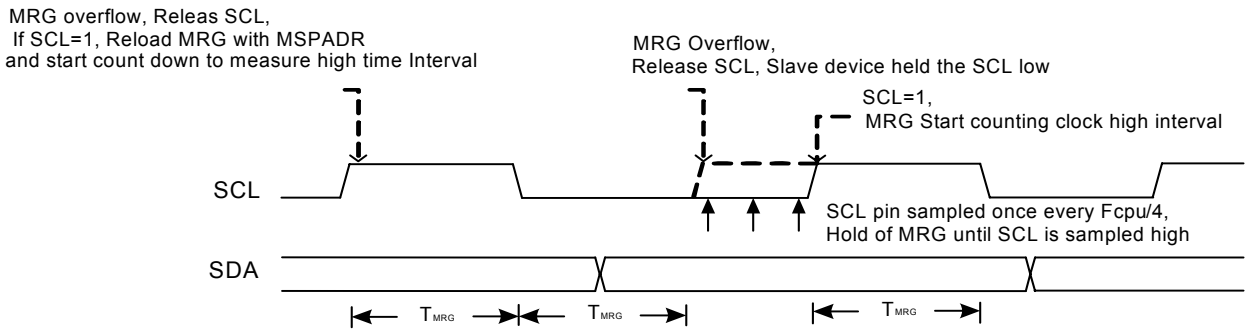
If user write to MSPBUF when a STOP condition is processing, then WCOL bit is set and the content of MSPBUF data is un-changed. (the writer doesn't occur)



STOP condition sequence Timing Diagram

13.8.7 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeat START, STOP condition that SCL pin allowed to float high. When SCL pin is allowed float high, the MSP rate generator (MRG) suspended from counting until the SCL pin is actually sampled high. When SCL is sampled high, the MRG is reloaded with the content of MSPADR[6:0], and start down counter. This ensure that SCL high time will always be at least one MRG overflow time in the event that the clock is held low by an external device.



Clock Arbitration sequence Timing Diagram

13.8.8 Master Mode Transmission

Transmission a data byte, 7-bit address or the eight bit data is accomplished by simply write to MSPBUF register. This operation will set the Buffer Full flag BF and allow MSP rate generator start counting.

After write to MSPBUF, each bit of address will be shifted out on the falling edge of SCL until 7-bit address and R/W bit are complete. On the falling edge of eighth clock, the master will pull low SDA fort slave device respond with an acknowledge. On the ninth clock falling edge, SDA is sampled to indicate the address already accept by slave device. The status of the ACK bit is load into ACKSTAT status bit. Then MSPIRQ bit is set, the BF bit is clear and the MRG is hold off until another write to the MSPBUF occurs, holding SCL low and allow SDA floating.

BF Status Flag

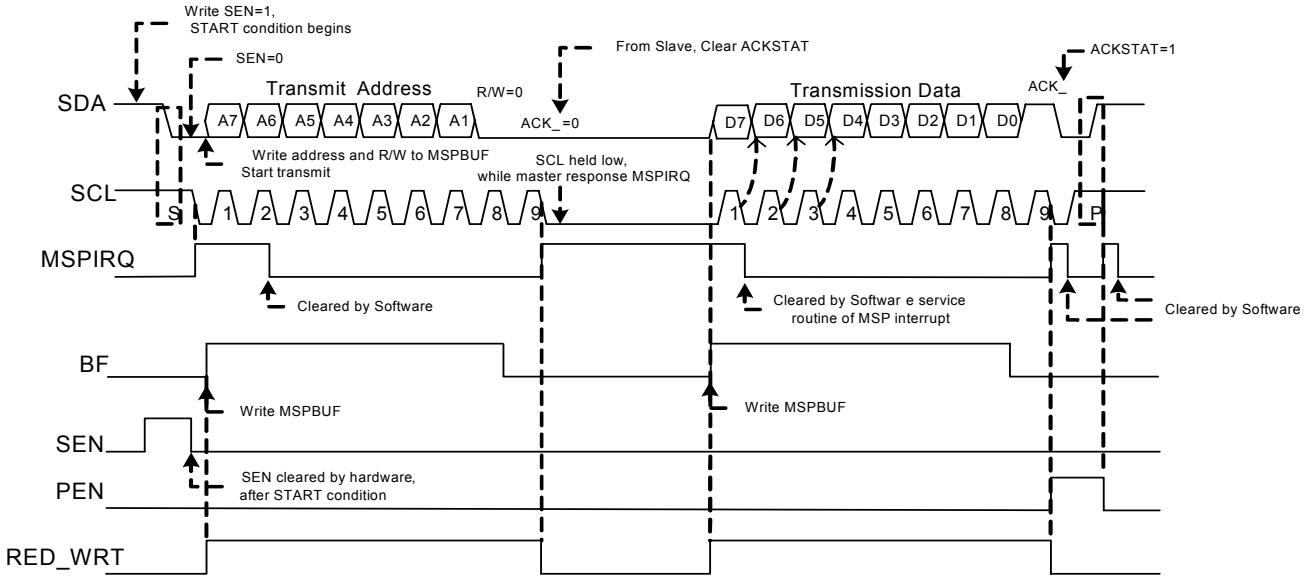
In transmission mode, the BF bit is set when user writes to MSPBUF and is cleared automatically when all 8 bit data are shift out.

WCOL Flag

If user write to MSPBUF during Transmission sequence in progress, the WCOL bit is set and the content of MSPBUF data will unchanged.

ACKSTAT Status Flag

In transmission mode, the ACKSTAT bit is cleared when the slave has sent an acknowledge (ACK_=0), and is set when slave does not acknowledge (ACK_=1). A slave send an acknowledge when it has recognized its address (including general call), or when the slave has properly received the data.



MSP Master Transmission Mode Timing Diagram

13.8.9 Master Mode Receiving

Master receiving mode is enable by set RCEN bit.

The MRG start counting and when SCL change state from low to high, the data is shifted into MSPSR. After the falling edge of eighth clock, the receive enable bit (RCEN) is clear automatically, the contents of MSP are load into MSPBUF, the BF flag is set, the MSPIRQ flag is set and MRG counter is suspended from counting, holding SCL low. The MSP is now in IDLE mode and awaiting the next operation command. When the MSPBUF data is read by Software, the BF flag is clear automatically. By setting ACKEN bit, user can send an acknowledge bit at the end of receiving.

BF Status Flag

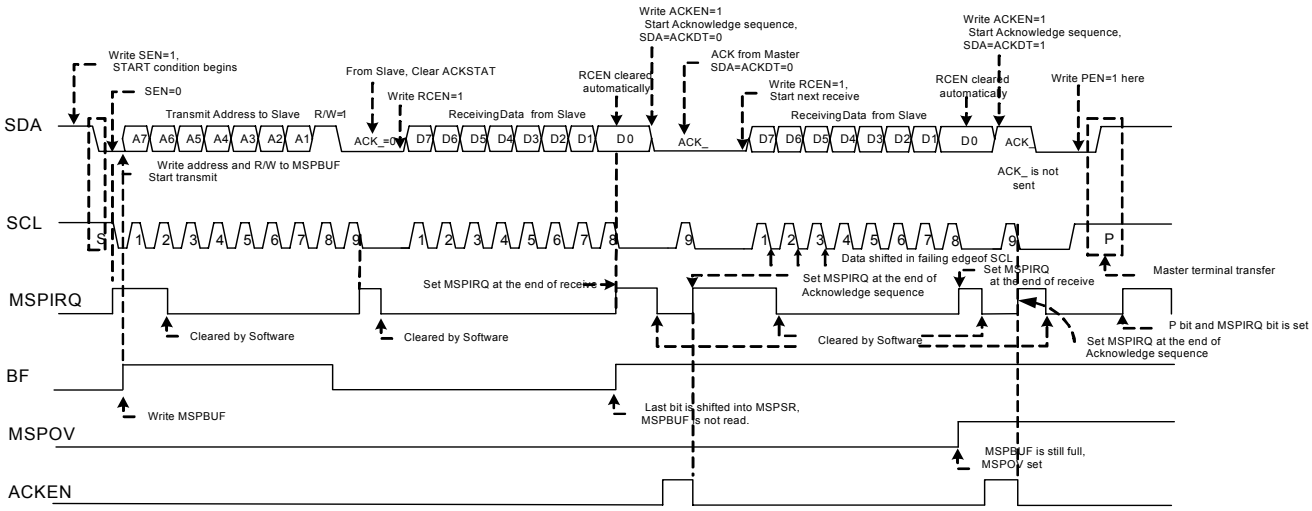
In Reception mode, the BF bit is set when an address or data byte is loaded into MSPBUF from MSPSR. It is cleared automatically when MSPBUF is read.

MSPOV Flag

In receive operation, the MSPOV bit is set when another 8-bit are received into MSPSR, and the BF bit is already set from previous reception.

WCOL Flag

If user write to MSPBUF when a receive is already progress, the WCOL bit is set and the content of MSPBUF data will unchanged.



MSP Master Receiving Mode Timing Diagram

14 Flash

14.1 OVERVIEW

The SN8F2280 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 SN8F2280 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 code option (ROM address 0x2F80 ~ 0x2FFF) cannot be erased from application code when the code option's security 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).

14.2 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)**

14.3 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

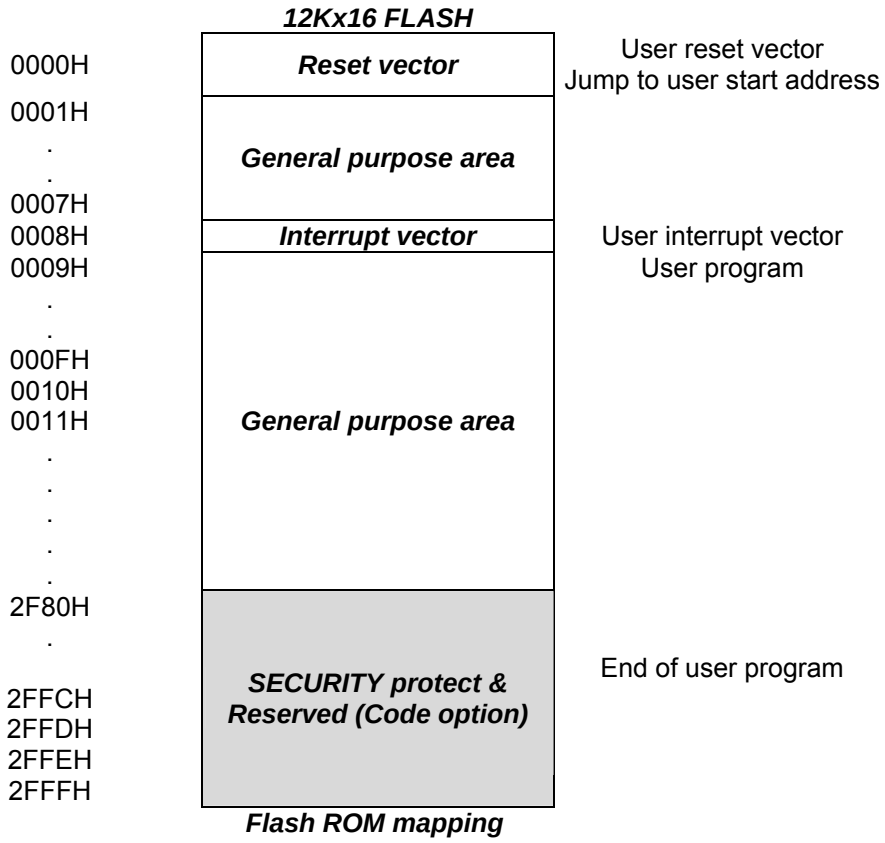
Bit [7:0] **PEROML[7:0]:** Define the target starting low byte address [7:0] of Flash memory (12K x 16) which is going to be programmed or erased.

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 (12K 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 … 0x2F80.** The page erase function is used to erase a page of 128 contiguous words in Flash ROM.

The valid **PAGE PROGRAM** starting addresses are **0x0, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, 0xE0 … 0x2FE0.** 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 SECURITY = 1, the FLASH ROM ADDRESS = 0x2F80 ~ 0x2FFF will not allow to do the "page erase and page program".

14.4 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	-	PERAML9	PERAML8
Read/Write	R/W	R/W	R/W	R/W	R/W	-	-	R/W
After reset	0	0	0	0	0	-	-	0

{PERAMCNT [1:0], PERAML [7:0]}**:** Define the starting RAM address [9:0], which stores the data wanted to be programmed. **The valid RAM addresses are 00H ~ 07FH and 0100H ~ 027FH.**

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.

14.4.1 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**

15 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
ARITHMETIC	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

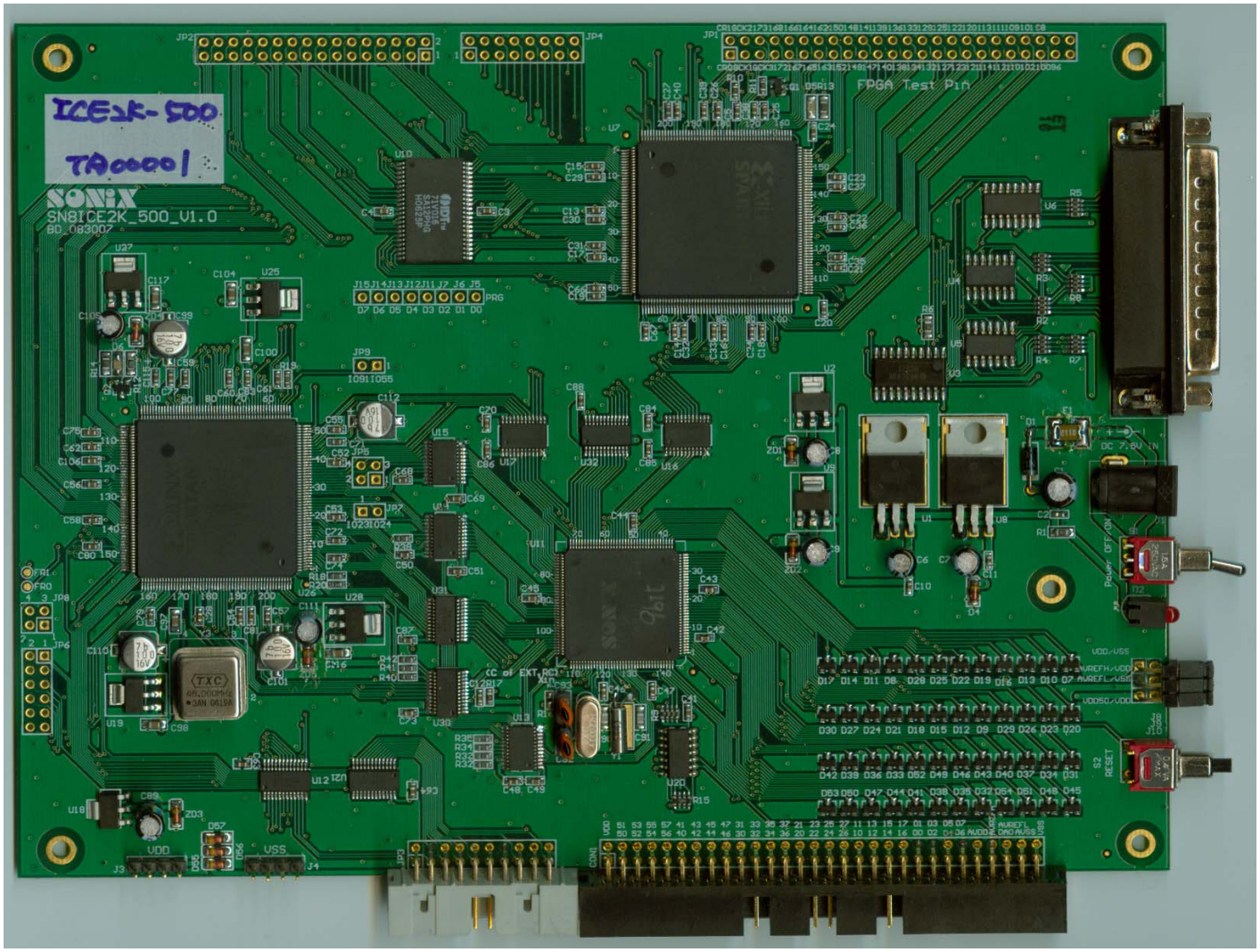
Note: 1. "M" is system register or RAM. If "M" is system registers then "N" = 0, otherwise "N" = 1.
2. If branch condition is true then "S = 1", otherwise "S = 0".

16 DEVELOPMENT TOOL

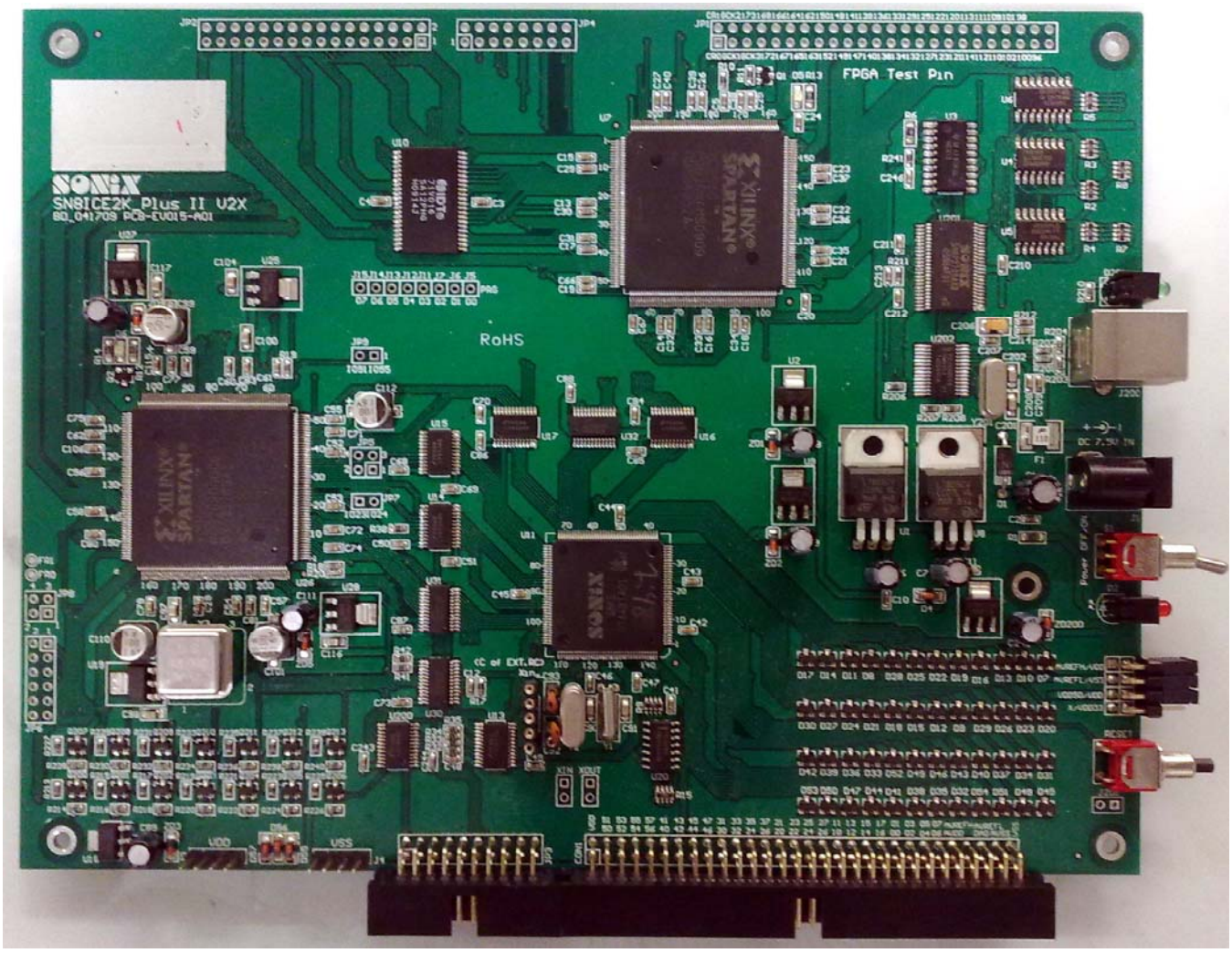
SONIX provides ICE (in circuit emulation), IDE (Integrated Development Environment), EV-kit and firmware library for USB application development. ICE and EV-kit are external hardware device and IDE is a friendly user interface for firmware development and emulation.

16.1 ICE (In Circuit Emulation)

The ICE called “SN8ICE2K Plus”.



The ICE called “SN8ICE2K Plus II”.



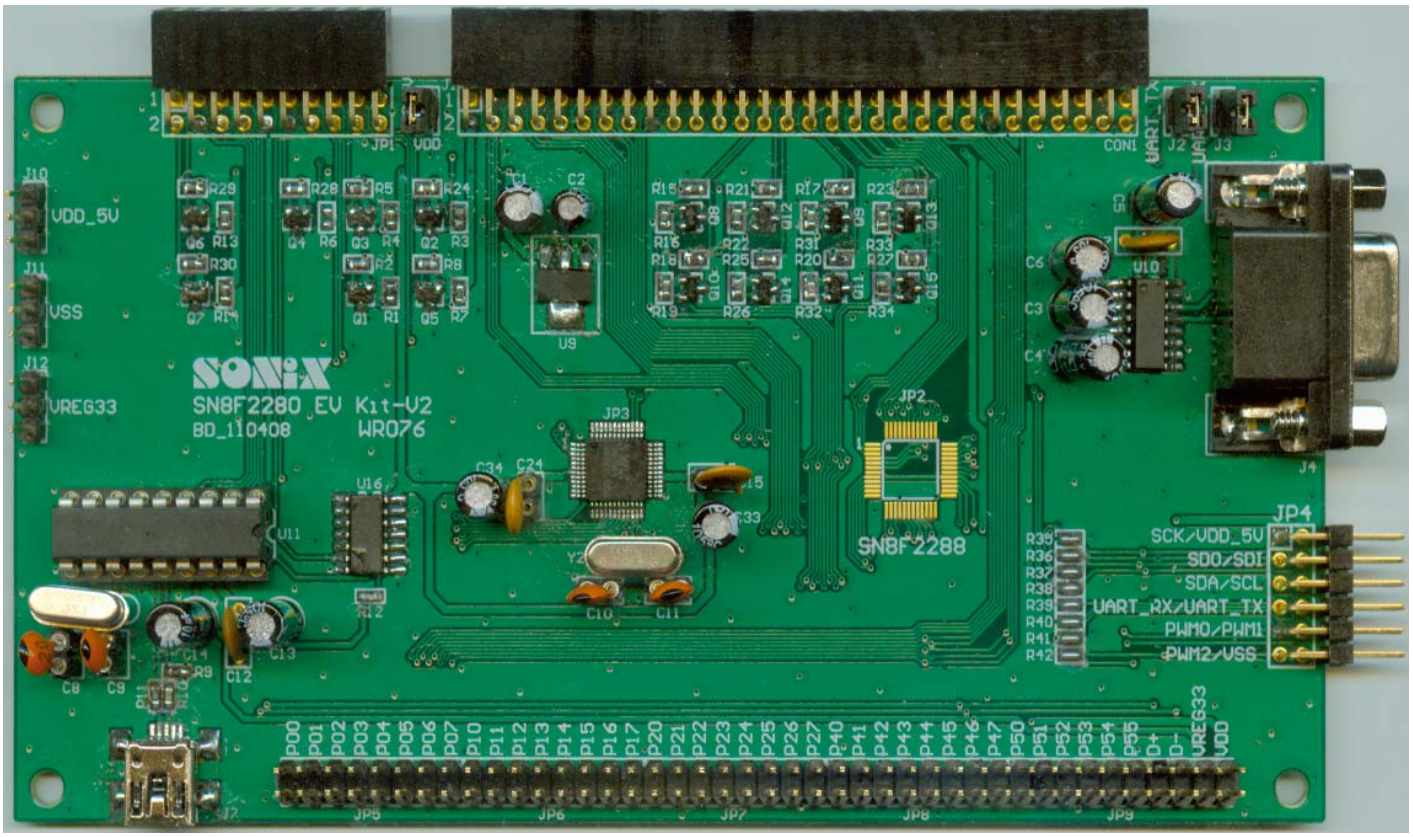
- ***Note 1: “SN8ICE2K Plus” shall be used with SN8F2280 EV-Kit V2 for FW development and emulation.***
- ***Note 2: “SN8ICE2K Plus II” shall be used with SN8F2280 EV-Kit V3 for FW development and emulation.***

16.2 SN8F2280 EV-Kit

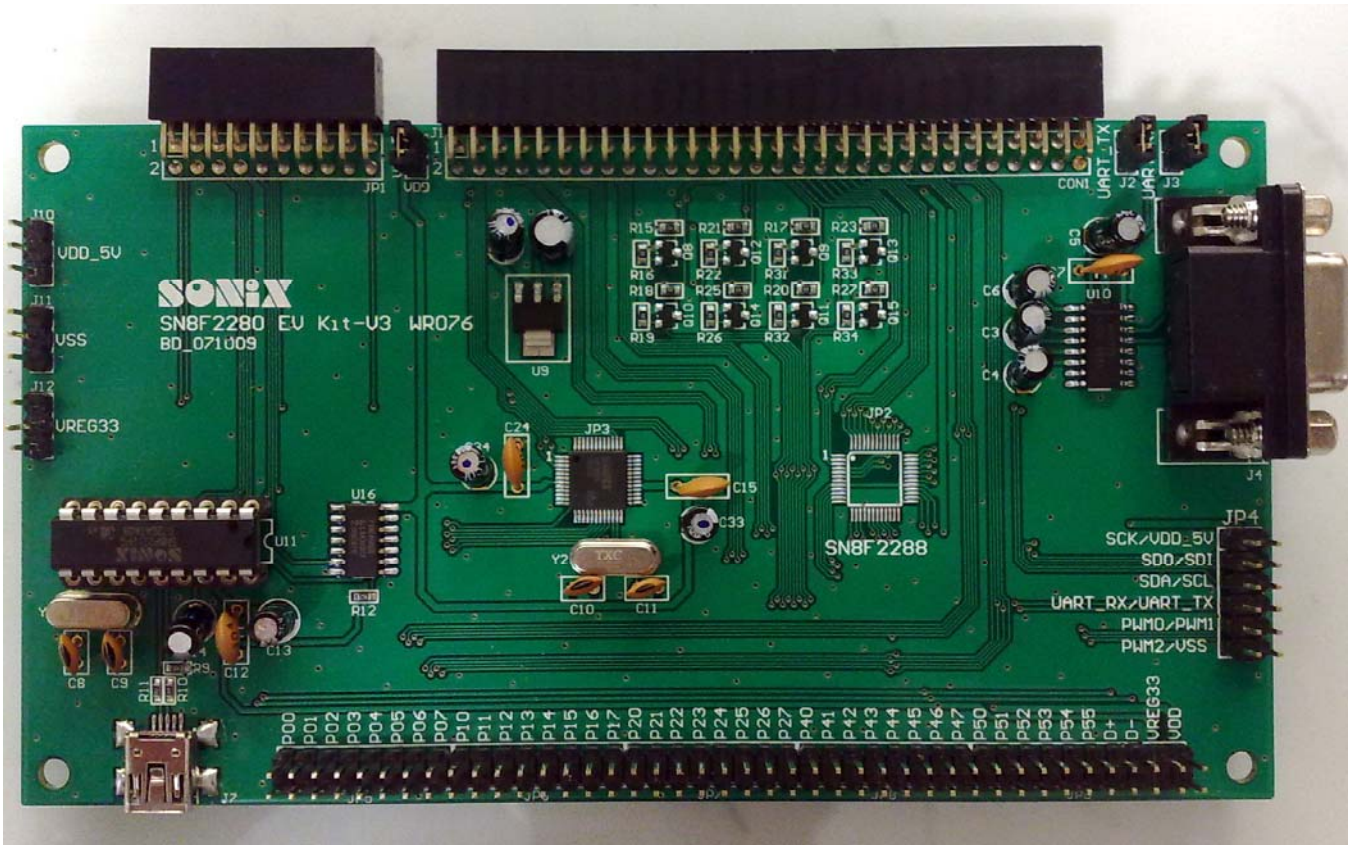
SN8F2280 EV-kit includes ICE interface, GPIO interface, USB interface, UART interface, SIO interface, MSP interface, PWM interface, and VREG 3.3V power supply.

- ICE Interface: Interface connected to SN8ICE2K Plus.
- GPIO Interface: SN8F2288 package form connector.
- USB Interface: USB Mini-B connector.
- UART Interface: Interface connected to Universal Asynchronous Receiver/Transmitter (UART) connector.
- SIO interface: Interface connected to Serial Input/Output Transceiver (SIO) connector.
- MSP interface: Interface connected to Main Series Port (MSP) connector.
- PWM interface: Output the PWM signal to PWM connector.
- VREG 3.3V Power Supply: Use SN8P2212's VREG to supply 3.3V power for VREG33 pin.

The outline of SN8F2280 EV-kit V2 is as following.



The outline of SN8F2280 EV-kit V3 is as following.

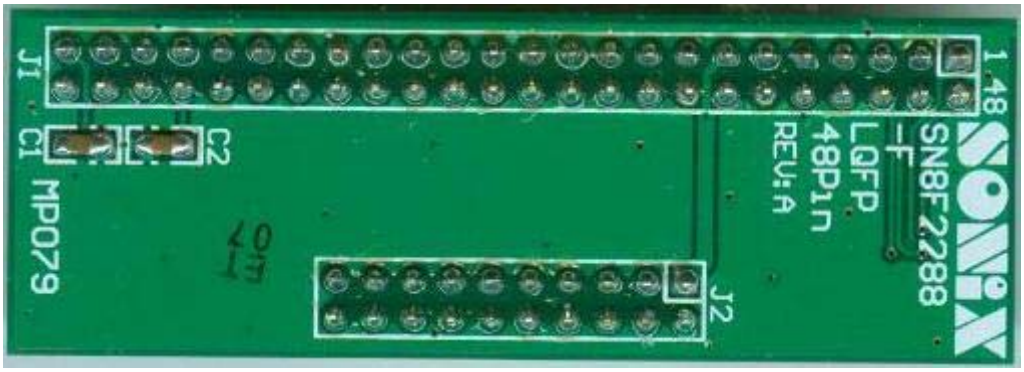


- CON1: ICE Interface connected to SN8ICE2K Plus.
- J1: Jumper to connect between the 5V VDD from SN8ICE2K Plus and VDD on SN8F2288 package from socket.
- J7: USB Mini-B connector.
- U11: SN8P2212 to supply 3.3V power for VREG33 pin and USB PHY.
- JP2: SN8F2288 connector for user's target board.
- JP3: SN8F2288 to supply P0 and P1 I/O level change interrupt function.
- J2-J4: UART interface connected to RS-232 connector.
- JP4: SIO interface connector, MSP interface connector, UART interface connector, and PWM interface connector.

- ***Note 1: In the EV-Kit, UART and MSP are not built-in open-drain function, such as P0.5/URX, P0.6/UTX, P1.0/SCL, P1.1/SDA. These are different from IC.***
- ***Note 2: In the EV-Kit, Port 2, P0.5/URX, P0.6/UTX, P1.0/SCL, P1.1/SDA, and P5.5/PWM2 are built-in 1K ohm external pull up resisters.***
- ***Note 3: In the EV-Kit, Port 2 is NOT Schmitt Trigger structure. This is different from IC.***
- ***Note 4: "SN8ICE2K Plus" shall be used with SN8F2280 EV-Kit V2 for FW development and emulation.***
- ***Note 5: "SN8ICE2K Plus II" shall be used with SN8F2280 EV-Kit V3 for FW development and emulation.***

16.3 SN8F2280 Transition Board

SN8F2280 Transition Boards is designated to IC LQFP Package. The following shows the transition board outline for SN8F2288. **Among the board, both C1 and C2 MUST be welded by 1uF capacitor.**



17 ELECTRICAL CHARACTERISTIC

17.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 5.5V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8F2288F ..... 0°C ~ + 70°C
Storage ambient temperature (Tstor) ..... –30°C ~ + 125°C

17.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, fosc = 12MHz, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd1	Normal mode except USB transmitter		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	P0, P1, P4, P5 input ports		Vss	-	0.2Vdd	V
	ViL2	P2 input ports		Vss	-	0.2 VREG33	V
Input High Voltage	ViH1	P0, P1, P4, P5 input ports		0.8Vdd	-	Vdd	V
	ViH2	P2 input ports		0.8 VREG33	-	VREG33	V
Input Voltage	Vin1	P0, P1, P4, P5 I/O port's input voltage range		-0.5	-	Vdd+0.5	V
	Vin2	P2 I/O port's input voltage range		-0.3	-	VREG33 +0.3	V
Output Voltage	Voh1	P0, P1, P4, P5 output ports		0	-	Vdd	V
	Voh2	P2 output ports		0	-	VREG33	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port pull-up resistor Rup1	Rup1	P0, P1, P4, P5 's Vin = Vss, Vdd = 5V		25	40*	70	KΩ
I/O port pull-up resistor Rup2	Rup2	P2's Vin = Vss, Vdd = 5V		30	55*	100	KΩ
D+ pull-up resistor	Rd+	Vdd = 5V, VREG = 3.3V		1	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	IoH1	P0, P1, P4, P5 I/O port's output source current, Vop1 = Vdd – 1V		15	20*		mA
	IoH2	P2 I/O port's output source current, Vop2 = VREG33 – 1V		1	2*		
sink current	IoL1	P0, P1, P4, P5 I/O port's sink current, Vop1 = Vss + 0.4V			15*	20	
	IoL2	P2 I/O port's sink current, Vop2 = 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*	50	ms
Page program (32 words)	Tpg	Flash ROM page program time (program 32 words)		-	1*	2	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, 12Mhz	-	10	15	mA
	Idd2	Slow Mode (Internal low RC)	Vdd= 5V, 12Khz	-	190	250	uA
	Idd3	Sleep Mode	Vdd= 5V	-	190	250	uA

	Idd4	Green Mode (No loading, Fcpu = Fosc/4 Watchdog Disable)	Vdd= 5V, 12Mhz	-	5	10	mA
			Vdd=5V, ILRC 12Khz	-	190	250	uA
LVD Voltage	Vdet1	Low voltage reset level middle.		1.8	2.4	2.9	V
	Vdet2	Low voltage reset level high.		2.8	3.6	4.25	V

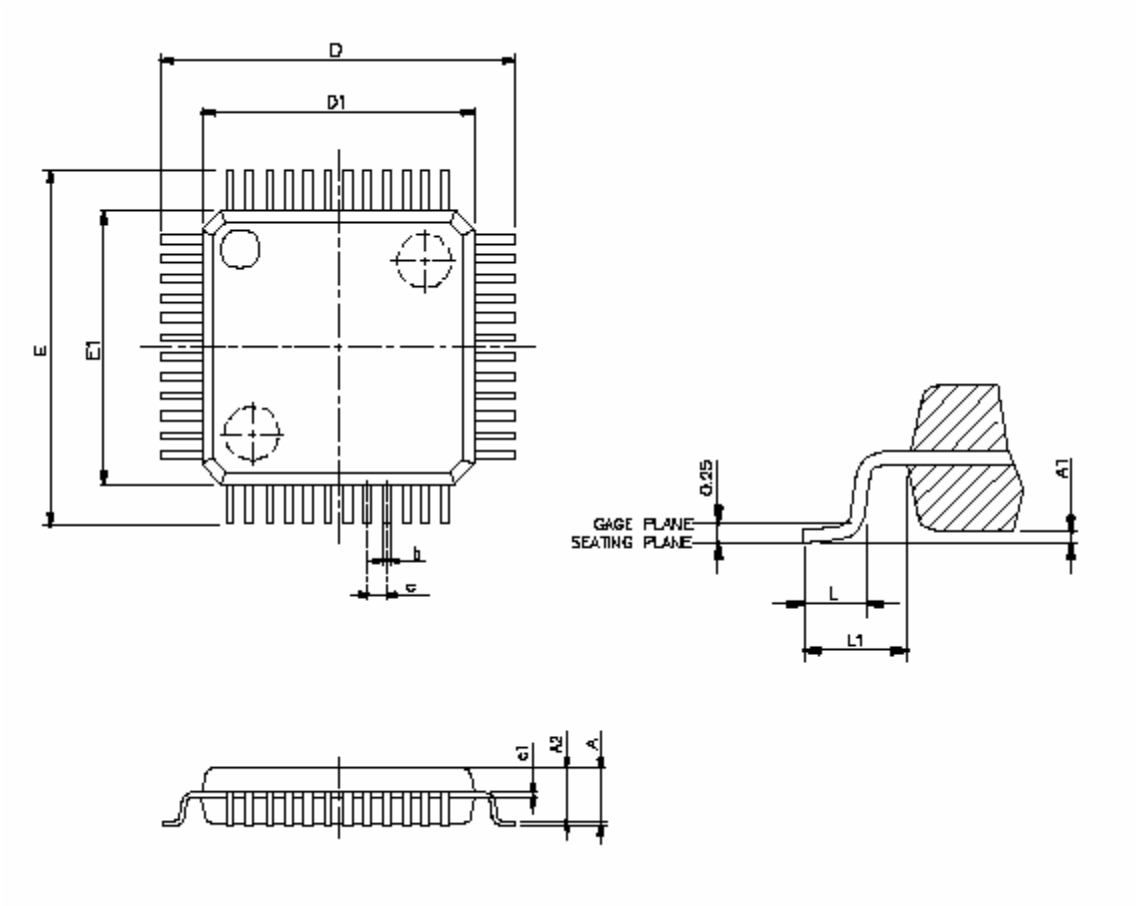
*** These parameters are for design reference, not tested.**

18 FLASH ROM PROGRAMMING PIN

Programming Information of SN8F2280 Series											
Chip Name		SN8F2288F/J		SN8F2283J							
EZ Writer / MP Writer Connector		Flash IC / JP3 Pin Assigment									
Number	Name	Number	Pin	Number	Pin	Number	Pin	Number	Pin	Number	Pin
1	VDD	7 29	VDD	11 23	VDD						
2	GND	16 25	VSS	2 12 13 19	VSS						
3	CLK	47	P1.4	5	P1.4						
4	CE										
5	PGM	1	P1.2	7	P1.2						
6	OE	46	P1.5	4	P1.5						
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	48	P1.3	6	P1.3						

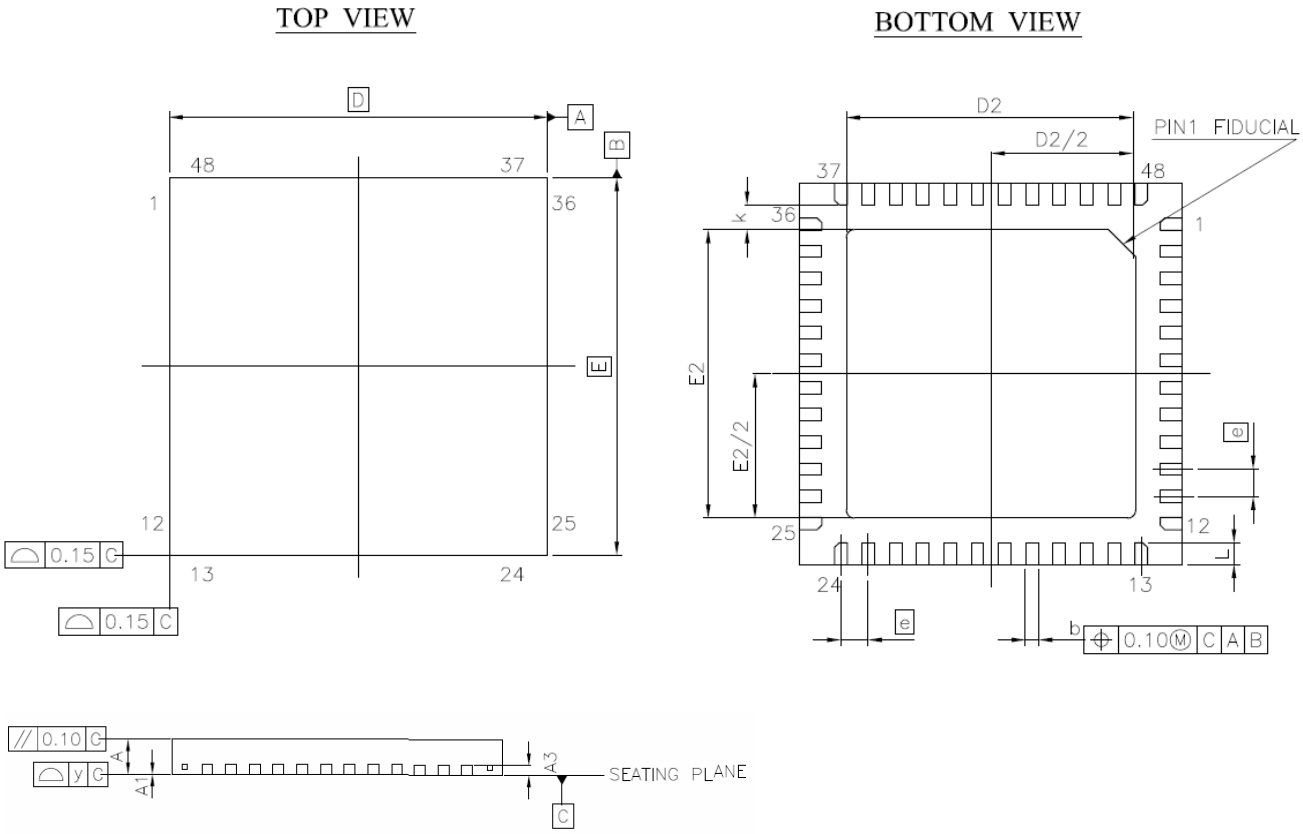
19 PACKAGE INFORMATION

19.1 LQFP 48 PIN



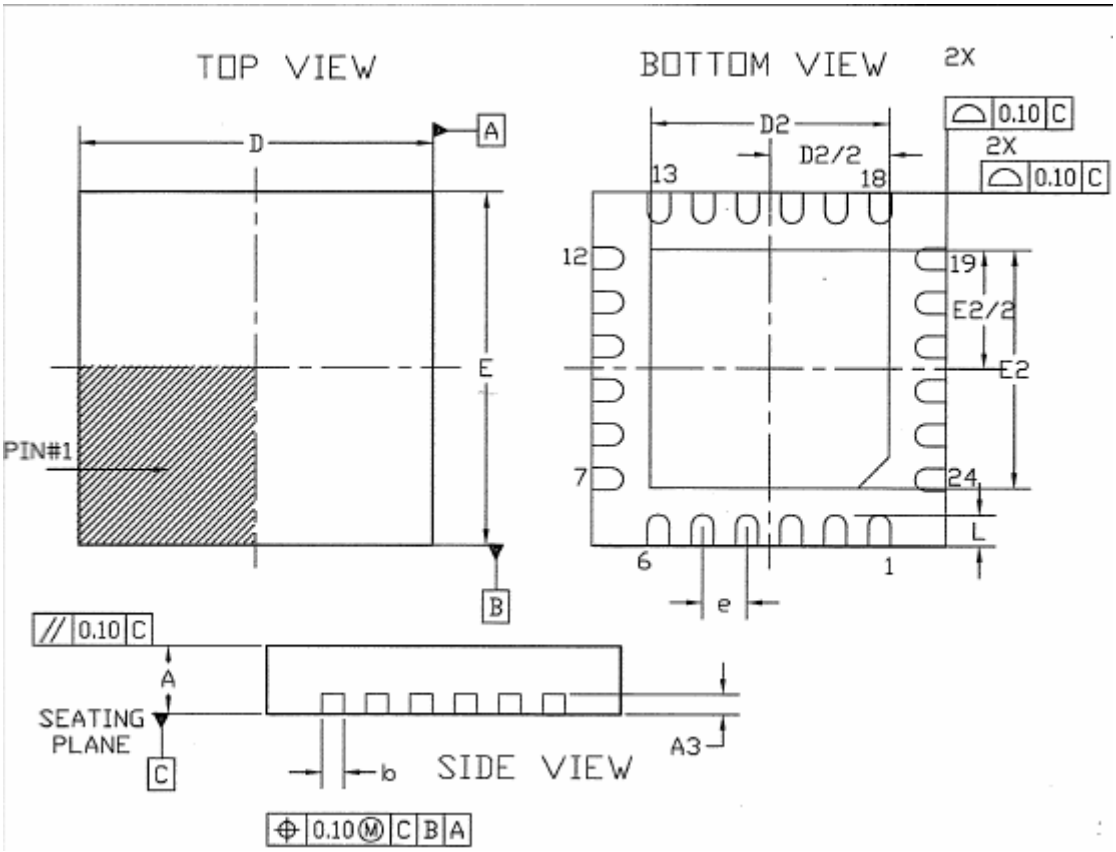
SYMBOLS	MIN	NOR	MAX
	(mm)		
A	-	-	1.6
A1	0.05	-	0.15
A2	1.35	-	1.45
c1	0.09	-	0.16
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
e	0.5 BSC		
B	0.17	-	0.27
L	0.45	-	0.75
L1	1 REF		

19.2 QFN 48 PIN



SYMBOL	DIMENSION (MM)			DIMENSION (MIL)		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	0.80	0.85	0.90	31.5	33.5	35.4
A1	0	0.02	0.05	0	0.79	1.97
A3	0.203 REF			8 REF		
b	0.18	0.25	0.30	7.1	9.8	11.8
D	7.00 BSC			276 BSC		
D2	5.10	5.20	5.60	201	205	221
E	7.00 BSC			276 BSC		
E2	5.10	5.20	5.30	201	205	209
e	0.50 BSC			19.7 BSC		
K	0.2			7.9		
L	0.30	0.40	0.50	11.8	15.7	19.7
y	0.08			3.15		

19.3 QFN 24 PIN



SYMBOL	COMMON					
	DIMENSIONS MILLIMETER			DIMENSIONS INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	SEE VARIATION					
A3	0.195	0.203	0.211	0.0077	0.008	0.0083
b	0.180	0.230	0.300	0.007	0.009	0.012
D	3.925	4.0	4.075	0.154	0.157	0.160
E	3.925	4.0	4.075	0.154	0.157	0.160
e	0.50 BSC			0.020 BSC		
L	SEE VARIATION					

SYMBOL	VARIATION " A "						REF
	DIMENSIONS MILLIMETER			DIMENSIONS INCH			
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	
TQFN	0.70	0.75	0.80	0.027	0.029	0.031	W VERY VERY THIN
QFN	0.85	0.90	0.95	0.033	0.035	0.037	V VERY THIN

PAD SIZE	VARIATION " L "						REF
	DIMENSIONS MILLIMETER			DIMENSIONS INCH			
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	
118x118	0.30	0.35	0.40	0.012	0.014	0.016	CUSTOMS A,B

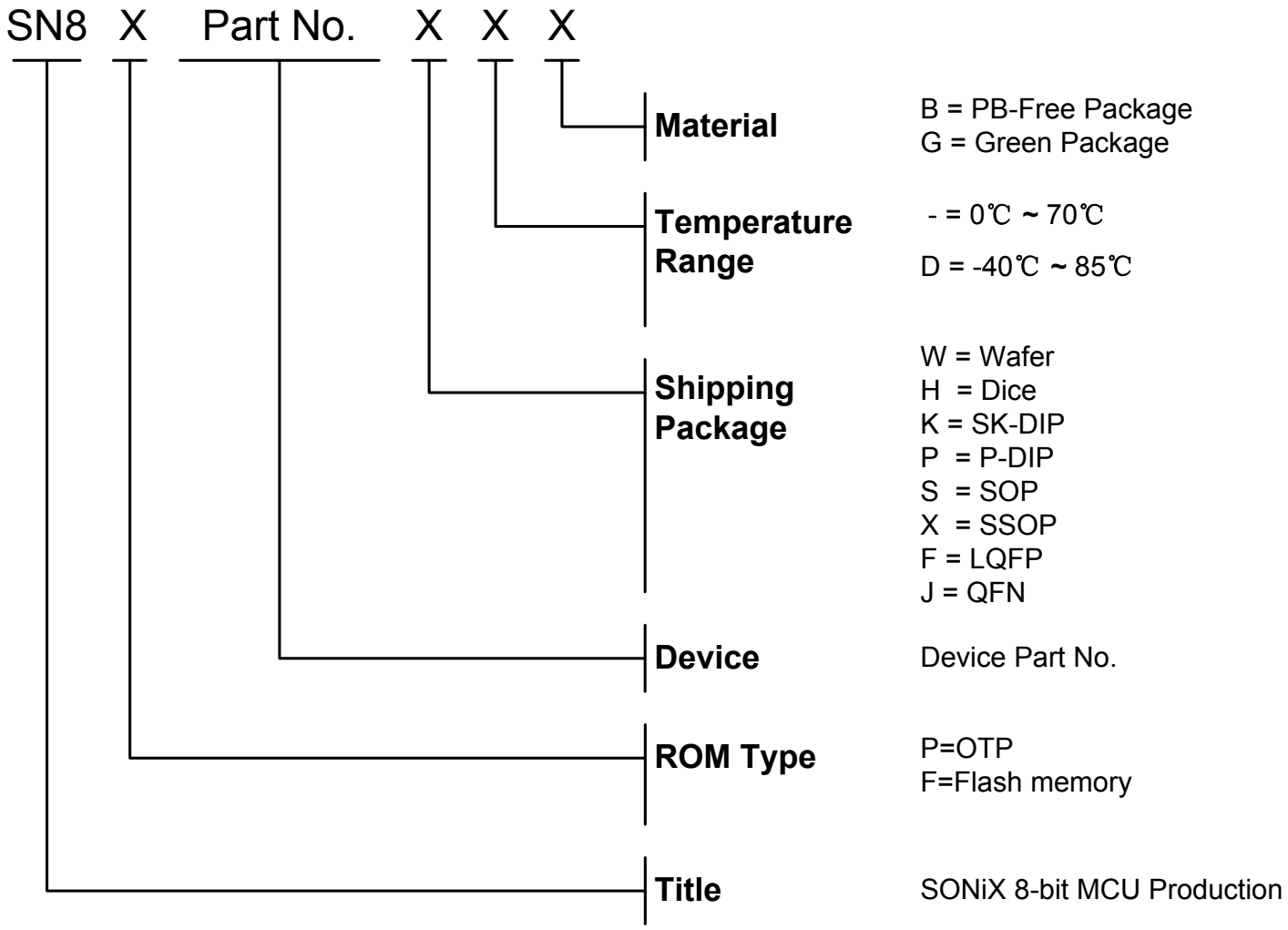
PAD SIZE	D2/E2			D2/E2			REF
	DIMENSIONS MILLIMETER			DIMENSIONS INCH			
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	
118x118	2.50/2.50	2.65/2.65	2.80/2.80	0.098/0.098	0.104/0.104	0.110/0.110	VGGD-6, WGGD-6

20 Marking Definition

20.1 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.

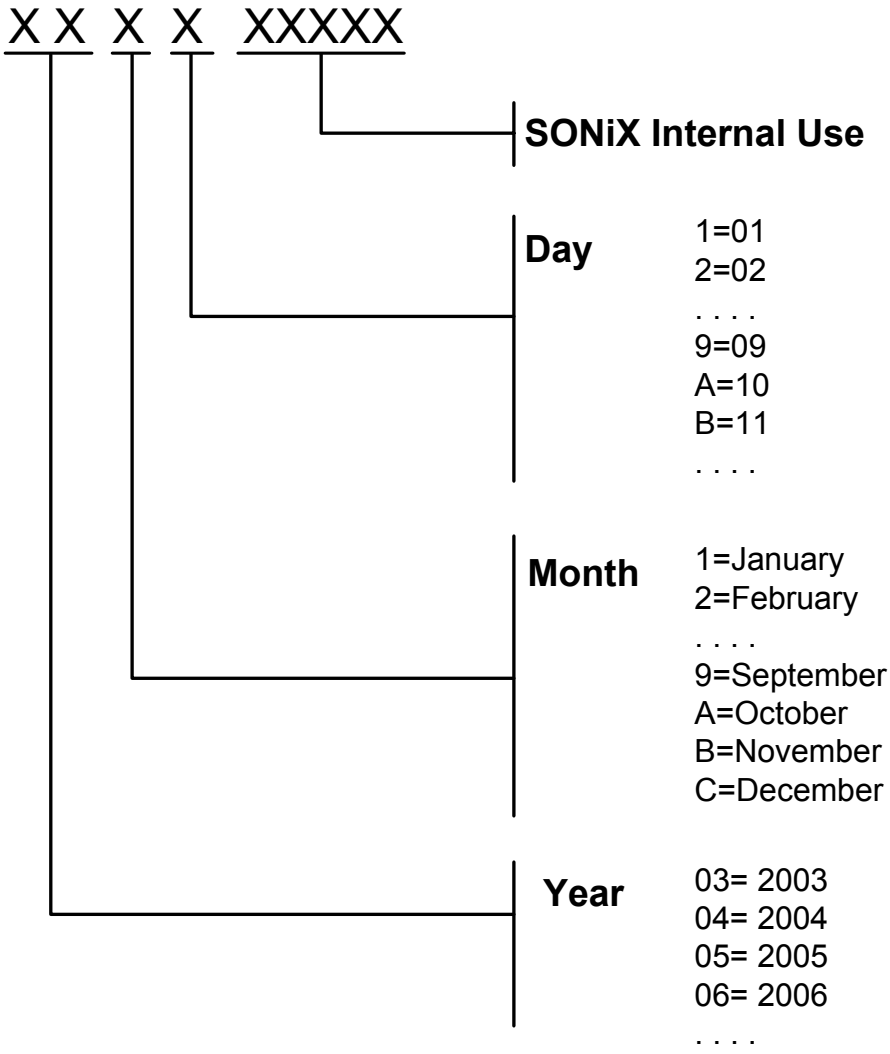
20.2 MARKING INDETIFICATION SYSTEM



20.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8F2288FG	Flash memory	2288	LQFP	0℃~70℃	Green Package
SN8F2288W	Flash memory	2288	Wafer	0℃~70℃	-
SN8F2288H	Flash memory	2288	Dice	0℃~70℃	-
SN8F2283JG	Flash memory	2288	QFN	0℃~70℃	Green Package

20.4 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