SN8F27E90 Series

USER'S MANUAL

Version 1.0

SN8F27E93
SN8F27E93L
SN8F27E94
SN8F27E94L

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 1.0	Nov. 2010	First version is released.

Table of Content

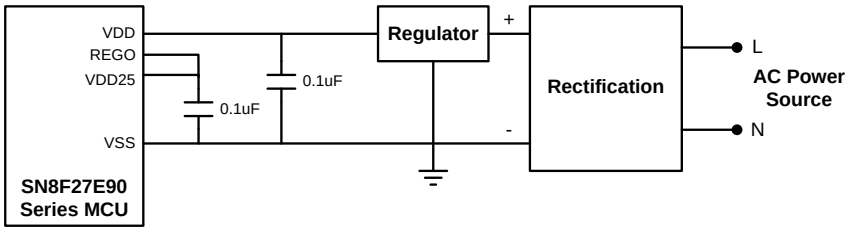
1 PRODUCT OVERVIEW

SN8F27E90 series 8-bit micro-controller is a new series production applied advanced semiconductor technology to implement flash ROM architecture. Under flash ROM platform, SN8F27E90 builds in in-system-programming (ISP) function extending to EEPROM emulation and Embedded ICE function. It offers high performance 6-ch 12-bit ADC, one programmable PWM, 3-type serial interfaces and flexible operating modes. Powerful functionality, high reliability and low power consumption can apply to AC power application and battery level application easily.

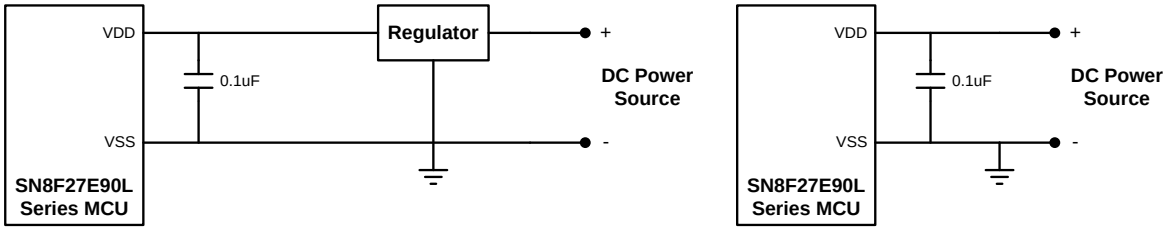
1.1 FEATURES

- **Memory configuration**
  Flash ROM size: 16K x 16 bits. Including EEROM emulation. (In system programming)
  RAM size: 1K x 8 bits.
- **16 levels stack buffer.**
- **11 interrupt sources**
  9 internal interrupts: T0, TC0, T1, ADC, SIO, MSP, UTX(UART TX), URX(UART RX), WAKE
  2 external interrupts: INT0, INT1
- **Multi-interrupt vector structure.**
  Each of interrupt sources has a unique interrupt vector.
- **I/O pin configuration**
  Bi-directional: P0, P1, P4.
  Wakeup: P0, P1 level change.
  Pull-up resisters: P0, P1, P4.
  External interrupt: P0.0, P0.1
  ADC input pin: AIN0~AIN5.
- **Fcpu (Instruction cycle)**
  Fcpu = Fhosc/1, Fhosc/2, Fhosc/4, Fhosc/8, Fhosc/16, Fhosc/32, Fhosc/64, Fhosc/128
- **On chip watchdog timer and clock source**
- **1.8V/2.3V/3.0V 3-level LVD with trim.**
- **Powerful instructions**
  Instruction's length is one word.
  Most of instructions are one cycle only.
  All ROM area JMP instruction.
  All ROM area lookup table function (MOVC).
- **Two 8-bit timer. (T0, TC0).**
  T0: Basic timer.
  TC0: Timer/counter/PWM0.
- **1 channel duty/cycle programmable PWM to Generate PWM, Buzzer and IR carrier signals. (PWM0).**
- **One 16-bit timer (T1).**
- **6- channel 12-bit SAR ADC.**
- **Serial Interface: SIO, UART, MSP**
- **Build in Embedded ICE function.**
- **Four system clocks**
  External high clock: RC type up to 10MHz
  External high clock: Crystal type up to 16MHz
  Internal high clock: RC type 16MHz
  Internal low clock: RC type 16KHz
- **Four operating modes**
  Normal mode: Both high and low clock active
  Slow mode: Low clock only
  Sleep mode: Both high and low clock stop
  Green mode: Periodical wakeup by timer
- **Package (Chip form support)**
  SK-DIP 24 pin
  SOP 24 pin
  SSOP 24 pin
  QFN 24 pin
  SK-DIP 28 pin

SN8F27E90 series micro-controller includes two types for different power types.
For AC power type (alternating current power source), the power pin has VDD and VDD25. VDD pin is connect to DC power source from DC-DC inverter or regulator and connects a 0.1uF capacitor to VSS pin (ground). VDD25 is internal power terminal, connect to DC power source from REGO pin, and connects a 0.1uF capacitor to VSS pin (ground). This pin assignment has high power noise immunity, but the static current is larger. The application field is household, motor control…



For DC power type (battery power source), the power pin is VDD. VDD pin is connect to DC power source from battery and connects a 0.1uF capacitor to VSS pin (ground). This pin assignment has low power noise immunity, but the static current is very low. The application field is portable application…



☞ **Features Selection Table**

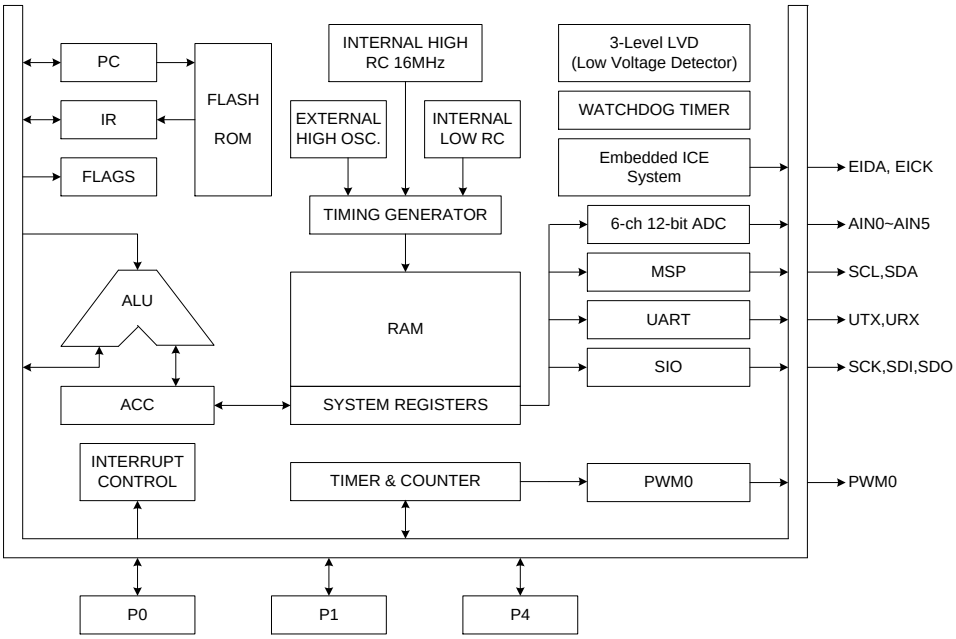
SN8F27E90 Series

CHIP	ROM	RAM	Stack	Timer	I/O	PWM	ADC	SIO	UART	MSP	Ext.INT	ISP/ Embedded ICE	Operating Voltage	Package
SN8F27E93	16K*16	1K	16	8-bit*2 16-bit*1	14	1-ch	6-ch	V	V	V	2	V	2.3V~5.5V	SK-DIP24 SOP24 SSOP24 QFN24
SN8F27E94	16K*16	1K	16	8-bit*2 16-bit*1	14	1-ch	6-ch	V	V	V	2	V	2.3V~5.5V	SK-DIP28

SN8F27E90L Series

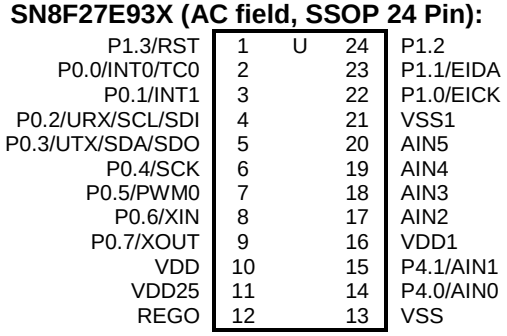
CHIP	ROM	RAM	Stack	Timer	I/O	PWM	ADC	SIO	UART	MSP	Ext.INT	ISP/ Embedded ICE	Operating Voltage	Package
SN8F27E93L	16K*16	1K	16	8-bit*2 16-bit*1	14	1-ch	6-ch	V	V	V	2	V	2.3V~3.0V	SK-DIP24 SOP24 SSOP24 QFN24
SN8F27E94L	16K*16	1K	16	8-bit*2 16-bit*1	14	1-ch	6-ch	V	V	V	2	V	2.3V~3.0V	SK-DIP28

1.2 SYSTEM BLOCK DIAGRAM

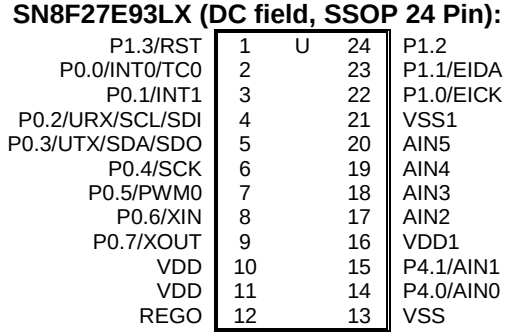


1.3 PIN ASSIGNMENT

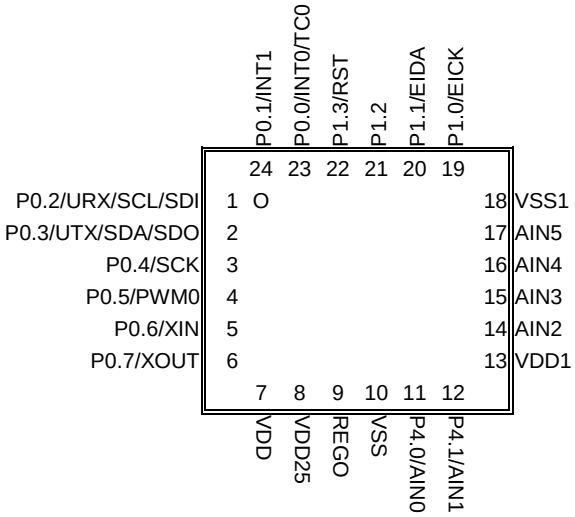
- **SN8F27E93K (AC field, SK-DIP 24 Pin):**
- **SN8F27E93S (AC field, SOP 24 Pin):**
- **SN8F27E93X (AC field, SSOP 24 Pin):**



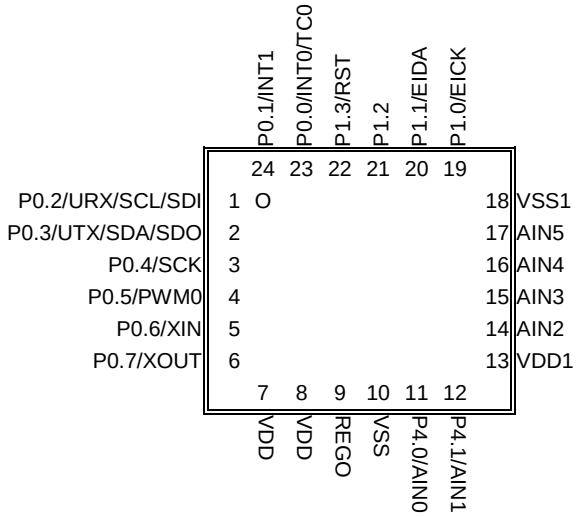
- **SN8F27E93LK (DC field, SK-DIP 24 Pin):**
- **SN8F27E93LS (DC field, SOP 24 Pin):**
- **SN8F27E93LX (DC field, SSOP 24 Pin):**



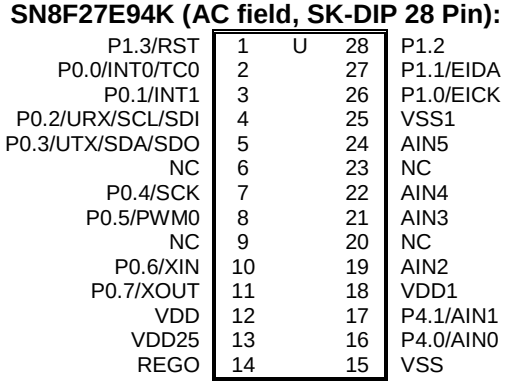
- **SN8F27E93J (AC field, QFN 4x4 24 Pin):**



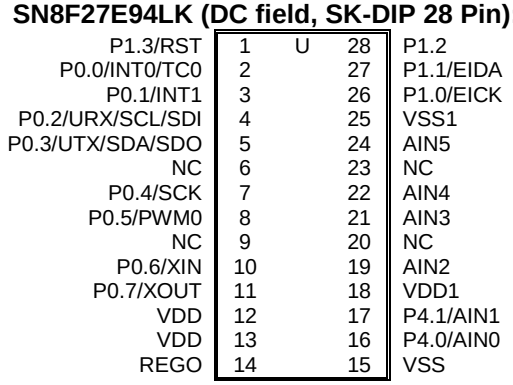
- **SN8F27E93LJ (DC field, QFN 4x4 24 Pin):**



- **SN8F27E94K (AC field, SK-DIP 28 Pin):**



- **SN8F27E94LK (DC field, SK-DIP 28 Pin):**

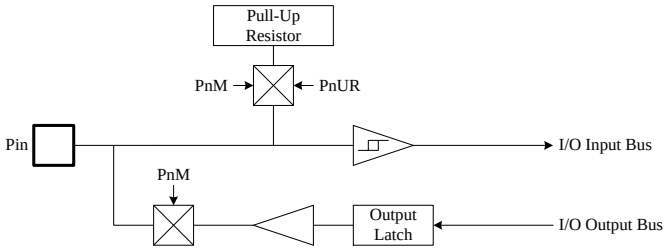


1.4 PIN DESCRIPTIONS

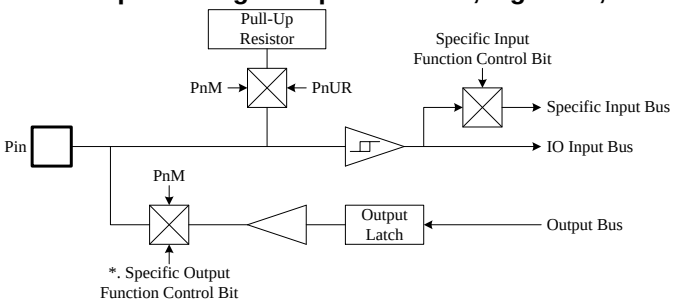
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital circuit.
VDD1, VSS1	P	Power supply input pins for analog circuit.
VDD25	P	SN8F27E90 Series (AC field): Power supply 2.5V input pin for digital circuit.
REGO	P	SN8F27E90 Series (AC field): Regulator 2.5V output pin. SN8F27E90L Series (DC field): Disconnect.
P0.0/INT0/ TC0	I/O	P0.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		INT0: External interrupt 0 input pin.
		TC0: TC0 event counter input pin.
P0.1/INT1	I/O	P0.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		INT1: External interrupt 1 input pin.
P0.2/URX/SCL/SDI	I/O	P0.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. Programmable open-drain structure.
		URX: UART receive input pin.
		SCL: MSP clock pin.
		SDI: SIO data input pin.
P0.3/UTX/SDA/SDO	I/O	P0.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. Programmable open-drain structure.
		UTX: UART transmit output pin.
		SDA: MSP data pin.
		SDO: SIO data output pin.
P0.4/SCK	I/O	P0.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. Programmable open-drain structure.
		SCK: SIO clock pin.
P0.5/ PWM0	I/O	P0.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		PWM0: PWM 0 output pin.
P0.6/ XIN	I/O	P0.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		XIN: Oscillator input pin while external oscillator enable (crystal and RC).
P0.7/ XOUT	I/O	P0.7: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		XOUT: Oscillator output pin while external crystal enable.
P1.0/EICK	I/O	P1.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		EICK: Embedded ICE clock pin.
P1.1/EIDA	I/O	P1.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		EIDA: Embedded ICE data pin.
P1.2	I/O	P1.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
P1.3/ RST	I/O	P1.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		RST: System external reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”.
P4.0/AIN0	I/O	P4.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		AIN0: ADC channel 0 input pin.
P4.1/AIN1	I/O	P4.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		AIN1: ADC channel 1 input pin.
AIN2	I	AIN2: ADC channel 2 input pin.
AIN3	I	AIN3: ADC channel 3 input pin.
AIN4	I	AIN4: ADC channel 4 input pin.
AIN5	I	AIN5: ADC channel 5 input pin.

1.5 PIN CIRCUIT DIAGRAMS

- **Normal bi-direction I/O pin.**

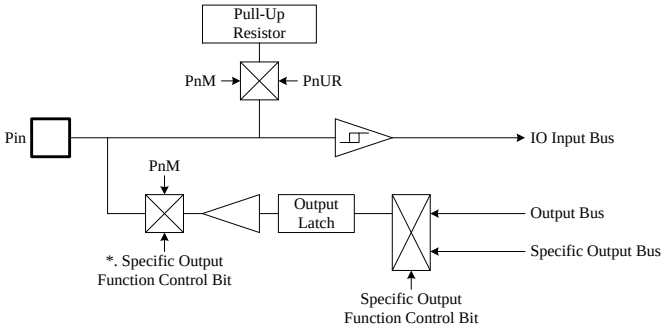


- **Bi-direction I/O pin shared with specific digital input function, e.g. INT0, event counter, SIO, MSP, UART…**



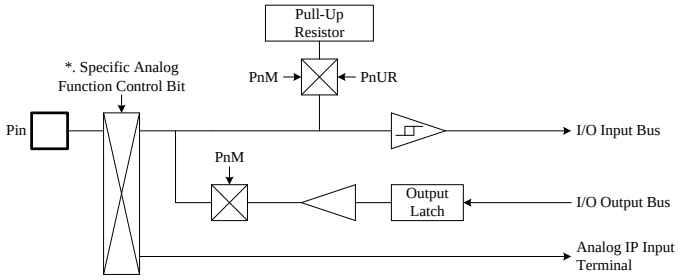
***. Some specific functions switch I/O direction directly, not through PnM register.**

- **Bi-direction I/O pin shared with specific digital output function, e.g. PWM, SIO, MSP, UART…**



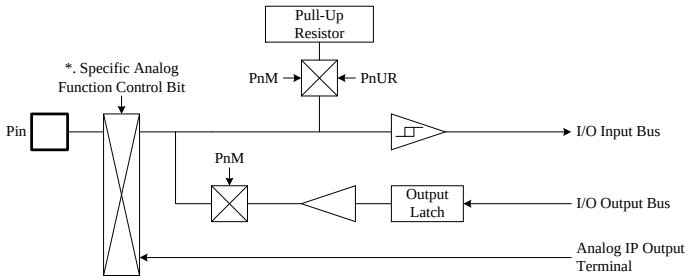
***. Some specific functions switch I/O direction directly, not through PnM register.**

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



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

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



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

2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (FLASH ROM)

☞ ***16K words FLASH ROM***

Address	***ROM***	Comment
0000H	*Reset vector*	Reset vector
0001H . 0007H	*General purpose area*	User program
0008H	***WAKE Interrupt vector***	**Interrupt vector**
0009H	***INT0 Interrupt vector***	
000AH	***INT1 Interrupt vector***	
000BH	***T0 Interrupt vector***	
000CH	***TC0 Interrupt vector***	
000DH	*Reserved*	
000EH	*Reserved*	
000FH	***T1 Interrupt vector***	
0010H	***ADC Interrupt vector***	
0011H	***SIO Interrupt vector***	
0012H	***MSP Interrupt vector***	
0013H	***UART RX Interrupt vector***	
0014H	***UART TX Interrupt vector***	
0015H	*Reserved*	
0016H	*General purpose area*	User program End of user program
3FF8H . 3FFFH	*Reserved*	

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

- **0x0000 Reset Vector: Program counter points to 0x0000 after any reset events (power on reset, reset pin reset, watchdog reset, LVD reset…).**
- **0x0001~0x0007: General purpose area to process system reset operation.**
- **0x0008~0x0015: Multi interrupt vector area. Each of interrupt events has a unique interrupt vector.**
- **0x0016~0x3F7F: General purpose area for user program and ISP (EEPROM function).**
- **0x3F80~0x3FF7: General purpose area for user program. Do not execute ISP.**
- **0x3FF8~0x3FFF: Reserved area. Do not execute ISP.**
- **ROM security rule is even address ROM data protected and outputs 0x0000.**

2.1.1 RESET VECTOR (0000H)

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

- ☞ **Power On Reset (POR=1).**
- ☞ **Watchdog Reset (WDT=1).**
- ☞ **External Reset (RST=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 POR, WDT, and RST 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     16H
START:                              ; 0016H, The head of user program.
                …                   ; User program
                …

                ENDP                ; End of program
```

✸ ***Note: The head of user program should skip interrupt vector area to avoid program execution error.***

2.1.2 INTERRUPT VECTOR (0008H~0015H)

A 14-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~0015h of program memory to execute the vectored interrupt. This interrupt is multi-vector and each of interrupts points to unique vector. Users have to define the interrupt vector. The following example shows the way to define the interrupt vector in the program memory.

✸ ***Note: The "PUSH" and "POP" operations aren't through instruction (PUSH, POP) and can executed save and load ACC and working registers (0x80~0x8F) by hardware automatically.***

	ROM	Priority
0008H	WAKE Interrupt vector	1
0009H	INT0 Interrupt vector	2
000AH	INT1 Interrupt vector	3
000BH	T0 Interrupt vector	4
000CH	TC0 Interrupt vector	5
000DH	Reserved	6
000EH	Reserved	7
000FH	T1 Interrupt vector	8
0010H	ADC Interrupt vector	9
0011H	SIO Interrupt vector	10
0012H	MSP Interrupt vector	11
0013H	UART RX Interrupt vector	12
0014H	UART TX Interrupt vector	13
0015H	Reserved	14

When one interrupt request occurs, and the program counter points to the correlative vector to execute interrupt service routine. If WAKE interrupt occurs, the program counter points to ORG 8. If INT0 interrupt occurs, the program counter points to ORG 9. In normal condition, several interrupt requests happen at the same time. So the priority of interrupt sources is very important, or the system doesn't know which interrupt is processed first. The interrupt priority is follow vector sequence. ORG 8 is priority 1. ORG 9 is priority 2. In the case, the interrupt processing priority is as following.

If WAKE, T0, TC0, T1 and SIO interrupt requests happen at the same time, the system processing interrupt sequence is WAKE, T0, TC0, T1, and then SIO. The system processes WAKE interrupt service routine first, and then processes T0 interrupt routine…Until finishing processing all interrupt requests.

➢ **Example:**

Interrupt Request Occurrence Sequence: (2~8 interrupt requests occur during WAKE interrupt service routine execution.)

1	2	3	4	5	6	7	8
WAKE	ADC	TC0	T0	SIO	INT0	T1	UART RX

Interrupt Processing Sequence:

1	2	3	4	5	6	7	8
WAKE	INT0	T0	TC0	T1	ADC	SIO	UART RX

➢ **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, 0008H.
                JMP     ISR_WAKE        ; Jump to interrupt service routine address.
                JMP     ISR_INT0
                JMP     ISR_INT1
                JMP     ISR_T0
                JMP     ISR_TC0
                NOP                     ; Reserved.
                NOP                     ; Reserved.
                JMP     ISR_T1
                JMP     ISR_ADC
                JMP     ISR_SIO
                JMP     ISR_MSP
                JMP     ISR_UART_RX
                JMP     ISR_UART_TX
                NOP                     ; Reserved.

                ORG     16H
START:                                  ; 0016H, The head of user program.
                …                       ; User program.
                …
                …
                JMP     START           ; End of user program.
                …
ISR_WAKE:                               ; The head of interrupt service routine.
                                        ; Save ACC and 0x80~0x8F register to buffers.
                …
                                        ; Load ACC and 0x80~0x8F register from buffers.
                RETI                    ; End of interrupt service routine.
ISR_INT0:                               ;
                                        ; Save ACC and 0x80~0x8F register to buffers.
                …
                                        ; Load ACC and 0x80~0x8F register from buffers.
                RETI                    ; End of interrupt service routine.
                …
                …
                …
                …
ISR_UART_TX:                            ;
                                        ; Save ACC and 0x80~0x8F register to buffers.
                …
                                        ; Load ACC and 0x80~0x8F 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~0015H is interrupt vector.***
3. ***User's program is a loop routine for main purpose application.***

2.1.3 LOOK-UP TABLE DESCRIPTION

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

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

```
                B0MOV   Y, #TABLE1$M    ; To set lookup table1's middle address
                B0MOV   Z, #TABLE1$L    ; To set lookup table1's low address.
                MOVC                    ; To lookup data, R = 00H, ACC = 35H
                                        ; Increment the index address for next address.
                INCMS   Z               ; Z+1
                JMP     @F              ; Z is not overflow.
                INCMS   Y               ; Z overflow (FFH → 00), → Y=Y+1
                NOP                     ;
                                        ;
@@:             MOVC                    ; To lookup data, R = 51H, ACC = 05H.
                …                       ;
TABLE1:         DW      0035H           ; To define a word (16 bits) data.
                DW      5105H
                DW      2012H
                …
```

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

➢ **Example: INC_YZ macro.**

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

                INCMS   Y               ; Y+1
                NOP                     ; Not overflow
@@:
                ENDM
```

➢ **Example: Modify above example by "INC_YZ" macro.**

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

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

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

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

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

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

                B0BTS1    FC              ; Check the carry flag.
                JMP       GETDATA         ; FC = 0
                INCMS     Y               ; FC = 1. Y+1.
                NOP
GETDATA:                                  ;
                MOVC                      ; To lookup data. If BUF = 0, data is 0x0035
                                          ; If BUF = 1, data is 0x5105
                                          ; If BUF = 2, data is 0x2012
                …
TABLE1:         DW        0035H           ; To define a word (16 bits) data.
                DW        5105H
                DW        2012H
                …
```

2.1.4 JUMP TABLE DESCRIPTION

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

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

➢ **Example: Jump table.**

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

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

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

➢ **Example: If "jump table" crosses over ROM boundary will cause errors.**

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

✸ ***Note: "VAL" is the number of the jump table listing number.***

➢ **Example: "@JMP_A" application in SONIX macro file called "MACRO3.H".**

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

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

➢ **Example: "@JMP_A" operation.**

; Before compiling program.

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

; After compiling program.

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

2.1.5 CHECKSUM CALCULATION

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

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

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

Y_ADD_1:
                INCMS   Y                   ; Increase Y
                NOP
                JMP     @B                  ; Jump to checksum calculate
CHECKSUM_END:
                …
                …
END_USER_CODE:                              ; Label of program end
```

2.2 DATA MEMORY (RAM)

✸ ***1K X 8-bit RAM***

Bank	Address	*RAM Location*	Comment
Bank 0	000H	***General purpose area***	RAM Bank 0
	...		
	...		
	07FH		
	080H	***System Register***	
	...		
	0FFH		End of Bank 0
Bank 1	100H	***General purpose area***	RAM Bank 1
	…		
	…		
	1FFH		End of Bank 1
Bank 2	200H	***General purpose area***	RAM Bank 2
	…		
	…		
	2FFH		End of Bank 2
Bank 3	300H	***General purpose area***	RAM Bank 3
	…		
	…		
	3FFH		End of Bank 3
Bank 4	400H	***General purpose area***	RAM Bank 4
	…		
	…		
	47FH		End of Bank 4

The 1K-byte general purpose RAM is separated into Bank0, Bank1, Bank2, Bank3 and Bank4. Accessing the five banks' RAM is controlled by "RBANK" register. When RBANK = 0, the program controls Bank 0 RAM directly. When RBANK = 1, the program controls Bank 1 RAM directly. When RBANK = 2, the program controls Bank 2 RAM directly. When RBANK = 3, the program controls Bank 3 RAM directly. When RBANK = 4, the program controls Bank 4 RAM directly. Under one bank condition and need to access the other bank RAM, setup the RBANK register is necessary. When interrupt occurs, RBANK register is saved, and RAM bank is still last condition. User can select RAM bank through setup RBANK register during processing interrupt service routine. When RETI is executed to leave interrupt operation, RBANK register is reloaded, and RAN bank returns to last condition. Sonix provides "Bank 0" type instructions (e.g. b0mov, b0add, b0bts1, b0bset…) to control Bank 0 RAM in non-zero RAM bank condition directly.

➢ **Example: Access Bank 0 RAM in Bank 1 condition. Move Bank 0 RAM (WK00) value to Bank 1 RAM (WK01).**

```
; Bank 1 (RBANK = 1)
        B0MOV   A, WK00     ; Use Bank 0 type instruction to access Bank 0 RAM.
        MOV     WK01,A
```

✸ ***Note: For multi-bank RAM program, it is not easy to control RAM Bank selection. Users have to take care the RBANK condition very carefully, especially for interrupt service routine. The system won't save the RBANK and switch RAM bank to Bank 0, so these controls must be through program. It is a good to use Bank 0 type instruction to process the situations.***

2.2.1 SYSTEM REGISTER

2.2.1.1SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	X	PFLAG	RBANK	W0	W1	W2	W3	W4	W5	W6	W7
9	@HL	@YZ	-	PCL	PCH	OSCM	WDTR	INTRQ0	INTRQ1	-	INTEN0	INTEN1	P0OC	-	P1W	PEDGE
A	P0M	P1M	-	-	P4M	-	P0	P1	-	-	P4	-	P0UR	P1UR	-	-
B	P4UR	-	T0M	T0C	TC0M	TC0C	TC0R	TC0D	-	ADM2	AIN2M	AIN3M	AIN4M	AIN5M	-	-
C	T1M	T1CL	T1CH	-	-	-	P4CON	-	ADM	ADB	ADR	-	PECMD	PE ROML	PE ROMH	PE RAML
D	PERA MCNT	SIOM	SIOR	SIOB	SIOC	URTX	URRX	URCR	UTXD	URXD	MSP STAT	MSPM1	MSPM2	MSP BUF	MSP ADR	STKP
E	STKFL	STKFH	STKEL	STKEH	STKDL	STKDH	STKCL	STKCH	STKBL	STKBH	STKAL	STKAH	STK9L	STK9H	STK8L	STK8H
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2SYSTEM REGISTER DESCRIPTION

H, L = Working, @HL addressing register.
R = Working register and ROM look-up data buffer.
X = Working and ROM address register
RBANK = RAM bank select register.
P1W = Port 1 wakeup register.
PEDGE = P0.0, P0.1 edge direction register.
URTX = UART transmit control register.
URRX = UART receive control register.
URCR = UART baud rate control register.
UTXD = UART transmit data buffer.
T1CH, L = T1 counting registers.
ADM,ADM2 = ADC mode register.
ADR = ADC resolution select register.
PEDGE = P0.0, P0.1, P0.2 edge direction register.
INTEN0,1 = Interrupt enable register.
PnM = Port n input/output mode register.
PnUR = Port n pull-up resister control register.
PCH, PCL = Program counter.
T0C = T0 counting register.
TC0C = TC0 counting register.
TC0D= TC0 duty control register.
MSPBUF= MSP buffer register.
MSPADR= MSP address register.
PECMD= ISP command register.
PEROM= ISP ROM address
@HL = RAM HL indirect addressing index pointer.
STKP = Stack pointer buffer.

Y, Z = Working, @YZ and ROM addressing register.
PFLAG = Special flag register.
W0~W7= Working register
P0OC = Open-drain control register.
SIOM = SIO mode control register.
SIOR = SIO clock rate control register.
SIOB = SIO data buffer.
SIOC = SIO control register.
T1M = T1 mode register.
URXD = UART receive data buffer.
P4CON = P4 configuration register.
ADB = ADC data buffer.
AINnM = ADC channel select register.
INTRQ0,1 = Interrupt request register.
WDTR = Watchdog timer clear register.
Pn = Port n data buffer.
OSCM = Oscillator mode register.
T0M = T0 mode register.
TC0M = TC0 mode register.
TC0R = TC0 auto-reload data buffer.
MSPSTAT= MSP status register
MSPM1= MSP mode register1
MSPM2= MSP mode register2
PERAM= ISP RAM mapping address
PERAMCNT= ISP RAM programming counter register.
@YZ = RAM YZ indirect addressing index pointer.
STK0~STKF = Stack 0 ~ stack 15 buffer.

2.2.1.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
085H	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0	R/W	X
086H	POR	WDT	RST	STKOV		C	DC	Z	R/W	PFLAG
087H						RBANKS2	RBANKS1	RBANKS0	R/W	RBANK
088H	W0BIT7	W0BIT6	W0BIT5	W0BIT4	W0BIT3	W0BIT2	W0BIT1	W0BIT0	R/W	W0
089H	W1BIT7	W1BIT6	W1BIT5	W1BIT4	W1BIT3	W1BIT2	W1BIT1	W1BIT0	R/W	W1
08AH	W2BIT7	W2BIT6	W2BIT5	W2BIT4	W2BIT3	W2BIT2	W2BIT1	W2BIT0	R/W	W2
08BH	W3BIT7	W3BIT6	W3BIT5	W3BIT4	W3BIT3	W3BIT2	W3BIT1	W3BIT0	R/W	W3
08CH	W4BIT7	W4BIT6	W4BIT5	W4BIT4	W4BIT3	W4BIT2	W4BIT1	W4BIT0	R/W	W4
08DH	W5BIT7	W5BIT6	W5BIT5	W5BIT4	W5BIT3	W5BIT2	W5BIT1	W5BIT0	R/W	W5
08EH	W6BIT7	W6BIT6	W6BIT5	W6BIT4	W6BIT3	W6BIT2	W6BIT1	W6BIT0	R/W	W6
08FH	W7BIT7	W7BIT6	W7BIT5	W7BIT4	W7BIT3	W7BIT2	W7BIT1	W7BIT0	R/W	W7
090H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
091H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
093H	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
094H			PC13	PC12	PC11	PC10	PC9	PC8	R/W	PCH
095H				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
096H	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
097H	ADCIRQ	T1IRQ			TC0IRQ	T0IRQ	P01IRQ	P00IRQ	R/W	INTRQ0
098H				MSPIRQ	UTXIRQ	URXIRQ	SIOIRQ	WAKEIRQ	R/W	INTRQ1
09AH	ADCIEN	T1IEN			TC0IEN	T0IEN	P01IEN	P00IEN	R/W	INTEN0
09BH				MSPIEN	UTXIEN	URXIEN	SIOIEN	WAKEIEN	R/W	INTEN1
09CH						P04OC	P03OC	P02OC	R/W	P0OC
09EH					P13W	P12W	P11W	P10W	R/W	P1W
09FH					P01G1	P01G0	P00G1	P00G0	R/W	PEDGE
0A0H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0A1H					P13M	P12M	P11M	P10M	R/W	P1M
0A4H							P41M	P40M	R/W	P4M
0A6H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0A7H					P13	P12	P11	P10	R/W	P1
0AAH							P41	P40	R/W	P4
0ACH	P07UR	P06UR	P05UR	P04UR	P03UR	P02UR	P01UR	P00UR	R/W	P0UR
0ADH					P13UR	P12UR	P11UR	P10UR	R/W	P1UR
0B0H							P41UR	P40UR	R/W	P4UR
0B2H	T0ENB	T0rate2	T0rate1	T0rate0				T0TB	R/W	T0M
0B3H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0B4H	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS1	TC0CKS0		PWM0OUT	R/W	TC0M
0B5H	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0B6H	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0B7H	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0	R/W	TC0D
0B9H	GCHS2								R/W	ADM2
0BAH			AIN2M5	AIN2M4	AIN2M3	AIN2M2	AIN2M1	AIN2M0	R/W	AIN2M
0BBH			AIN3M5	AIN3M4	AIN3M3	AIN3M2	AIN3M1	AIN3M0	R/W	AIN3M
0BCH			AIN4M5	AIN4M4	AIN4M3	AIN4M2	AIN4M1	AIN4M0	R/W	AIN4M
0BDH			AIN5M5	AIN5M4	AIN5M3	AIN5M2	AIN5M1	AIN5M0	R/W	AIN5M
0C0H	T1ENB	T1rate2	T1rate1	T1rate0	T1CKS				R/W	T1M
0C1H	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0	R/W	T1CL
0C2H	T1C15	T1C14	T1C13	T1C12	T1C11	T1C10	T1C9	T1C8	R/W	T1CH
0C6H							P4CON1	P4CON0	R/W	P4CON
0C8H	ADENB	ADS	EOC	GCHS			CHS1	CHS0	R/W	ADM
0C9H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0CAH		ADCKS1	ADCKS0	ADLEN	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0CCH	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0	R/W	PECMD
0CDH	PEROML7	PEROML6	PEROML5	PEROML4	PEROML3	PEROML2	PEROML1	PEROML0	R/W	PEROML
0CEH	PEROMH7	PEROMH6	PEROMH5	PEROMH4	PEROMH3	PEROMH2	PEROMH1	PEROMH0	R/W	PEROMH
0CFH	PERAML7	PERAML6	PERAML5	PERAML4	PERAML3	PERAML2	PERAML1	PERAML0	R/W	PERAML
0D0H	PERAMCNT7	PERAMCNT6	PERAMCNT5	PERAMCNT4	PERAMCNT3		PERAML9	PERAML8	R/W	PERAMCNT
0D1H	SENB	START	SRATE1	SRATE0	MLSB	SCLKMD	CPOL	CPHA	R/W	SIOM
0D2H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	W	SIOR
0D3H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB
0D4H						SIOBZ			R/W	SIOC
0D5H	UTXEN	UTXPEN	UTXPS	UTXBRK	URXBZ	UTXBZ			R/W	URTX

0D6H	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0	R/W	URRX
0D7H	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0	R/W	URCR
0D8H	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0	R/W	UTXD
0D9H	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0	R/W	URXD
0DAH	-	CKE	D_A	P	S	RED_WRT	-	BF	R	MSPSTAT
0DBH	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK	-	MSPC	R/W	MSPM1
0DCH	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	R/W	MSPM2
0DDH	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0	R/W	MSPBUF
0DEH	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0	R/W	MSPADR
0DFH	GIE	LVD23	LVD30		STKPB3	STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	SFPC7	SFPC6	SFPC5	SFPC4	SFPC3	SFPC2	SFPC1	SFPC0	R/W	STKFL
0E1H			SFPC13	SFPC12	SFPC11	SFPC10	SFPC9	SFPC8	R/W	STKFH
0E2H	SEPC7	SEPC6	SEPC5	SEPC4	SEPC3	SEPC2	SEPC1	SEPC0	R/W	STKEL
0E3H			SEPC13	SEPC12	SEPC11	SEPC10	SEPC9	SEPC8	R/W	STKEH
0E4H	SDPC7	SDPC6	SDPC5	SDPC4	SDPC3	SDPC2	SDPC1	SDPC0	R/W	STKDL
0E5H			SDPC13	SDPC12	SDPC11	SDPC10	SDPC9	SDPC8	R/W	STKDH
0E6H	SCPC7	SCPC6	SCPC5	SCPC4	SCPC3	SCPC2	SCPC1	SCPC0	R/W	STKCL
0E7H			SCPC13	SCPC12	SCPC11	SCPC10	SCPC9	SCPC8	R/W	STKCH
0E8H	SBPC7	SBPC6	SBPC5	SBPC4	SBPC3	SBPC2	SBPC1	SBPC0	R/W	STKBL
0E9H			SBPC13	SBPC12	SBPC11	SBPC10	SBPC9	SBPC8	R/W	STKBH
0EAH	SAPC7	SAPC6	SAPC5	SAPC4	SAPC3	SAPC2	SAPC1	SAPC0	R/W	STKAL
0EBH			SAPC13	SAPC12	SAPC11	SAPC10	SAPC9	SAPC8	R/W	STKAH
0ECH	S9PC7	S9PC6	S9PC5	S9PC4	S9PC3	S9PC2	S9PC1	S9PC0	R/W	STK9L
0EDH			S9PC13	S9PC12	S9PC11	S9PC10	S9PC9	S9PC8	R/W	STK9H
0EEH	S8PC7	S8PC6	S8PC5	S8PC4	S8PC3	S8PC2	S8PC1	S8PC0	R/W	STK8L
0EFH			S8PC13	S8PC12	S8PC11	S8PC10	S8PC9	S8PC8	R/W	STK8H
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, make sure to put all the "0" and "1" as it indicates in the above table.***
2. ***All of register names had been declared in SN8ASM assembler.***
3. ***One-bit name had been declared in SN8ASM assembler with "F" prefix code.***
4. ***"b0bset", "b0bclr","bset","bclr" instructions are only available to the "R/W" registers.***

2.2.2 ACCUMULATOR

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

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

```
; Read ACC data and store in BUF data memory
                MOV     BUF, A
; Write a immediate data into ACC
                MOV     A, #0FH
; Write ACC data from BUF data memory
                MOV     A, BUF
;or
                B0MOV   A, BUF
```

The system will store ACC and working registers (0x80-0x8F) by hardware automatically when interrupt executed.

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

```
.CODE
INT_SERVICE:
                PUSH                    ; Save ACC to buffer.
                                        ; Save working registers to buffers.
                …       .
                …
                POP                     ; Load working registers from buffers.
                                        ; Load ACC from buffer.

                RETI                    ; Exit interrupt service vector
```

2.2.3 PROGRAM FLAG

The PFLAG register contains the arithmetic status of ALU operation, system reset status and LVD detecting status. POR, WDT, and RST 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. LVD23, LVD30 bits indicate LVD detecting power voltage status.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	RST	STKOV	-	C	DC	Z
Read/Write	R	R	R	R	-	R/W	R/W	R/W
After Reset	-	-	-	-	-	0	0	0

Bit 7 **POR:** Power on reset and LVD brown-out reset indicator.
0 = Non-active.
1 = Reset active. LVD announces reset flag.

Bit 6 **WDT:** Watchdog reset indicator.
0 = Non-active.
1 = Reset active. Watchdog announces reset flag.

Bit 5 **RST:** External reset indicator.
0 = Non-active.
1 = Reset active. External reset announces reset flag.

Bit 4 **STKOV:** Stack overflow indicator.
0 = Non-overflow.
1 = Stack overflow.

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.

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	LVD23	LVD30	-	STKPB3	STKPB2	STKPB1	STKPB0
Read/Write	R/W	R	R	-	R/W	R/W	R/W	R/W
After Reset	0	-	-	-	1	1	1	1

Bit 6 **LVD23:** LVD23 low voltage detect indicator.
0 = Vdd > LVD23 detect level.
1 = Vdd < LVD23 detect level.

Bit 5 **LVD30:** LVD30 low voltage detect indicator.
0 = Vdd > LVD30 detect level.
1 = Vdd < LVD30 detect level.

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

2.2.4 PROGRAM COUNTER

The program counter (PC) is a 14-bit binary counter separated into the high-byte 6 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

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

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

☞ **ONE ADDRESS SKIPPING**

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

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

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

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

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

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

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

INCS instruction:

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

INCMS instruction:

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

If the destination decreased by 1, which results underflow of 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
                …
                …
```

2.2.5 H, L REGISTERS

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

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

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

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

Example: If want to read a data from RAM address 20H of bank0, it can use indirectly addressing mode to access data as following.

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

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

```
        CLR     H           ; H = 0, bank 0
        B0MOV   L, #07FH    ; L = 7FH, the last address of the data memory area
CLR_HL_BUF:
        CLR     @HL         ; Clear @HL to be zero
        DECMS   L           ; L – 1, if L = 0, finish the routine
        JMP     CLR_HL_BUF  ; Not zero

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

2.2.6 X REGISTERS

X register is an 8-bit buffer and only general working register purpose.

- Can be used as general working registers

085H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

2.2.7 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
        …
```

2.2.8 R REGISTER

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

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

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

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

2.2.9 W REGISTERS

W register includes W0~W7 8-bit buffers. There are two major functions of the register.

- Can be used as general working registers in assembly language situation.
- Can be used as program buffers in C-language situation.

088H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W0	W0BIT7	W0BIT6	W0BIT5	W0BIT4	W0BIT3	W0BIT2	W0BIT1	W0BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W1	W1BIT7	W1BIT6	W1BIT5	W1BIT4	W1BIT3	W1BIT2	W1BIT1	W1BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W2	W2BIT7	W2BIT6	W2BIT5	W2BIT4	W2BIT3	W2BIT2	W2BIT1	W2BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W3	W3BIT7	W3BIT6	W3BIT5	W3BIT4	W3BIT3	W3BIT2	W3BIT1	W3BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W4	W4BIT7	W4BIT6	W4BIT5	W4BIT4	W4BIT3	W4BIT2	W4BIT1	W4BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W5	W5BIT7	W5BIT6	W5BIT5	W5BIT4	W5BIT3	W5BIT2	W5BIT1	W5BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W6	W6BIT7	W6BIT6	W6BIT5	W6BIT4	W6BIT3	W6BIT2	W6BIT1	W6BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W7	W7BIT7	W7BIT6	W7BIT5	W7BIT4	W7BIT3	W7BIT2	W7BIT1	W7BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

✸ ***Note:***
1. ***In assembly language situation, W0~W7 can be used as general working registers.***
2. ***In C-language situation, W0~W7 are reserved for C-compiler, and recommend not to access W0~W7 by program strongly.***

2.3 ADDRESSING MODE

2.3.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

✸ ***Note: In immediate addressing mode application, the specific RAM must be 0x80~0x8F working register.***

2.3.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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

2.3.3 INDIRECTLY ADDRESSING MODE

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

Example: Indirectly addressing mode with @HL register

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

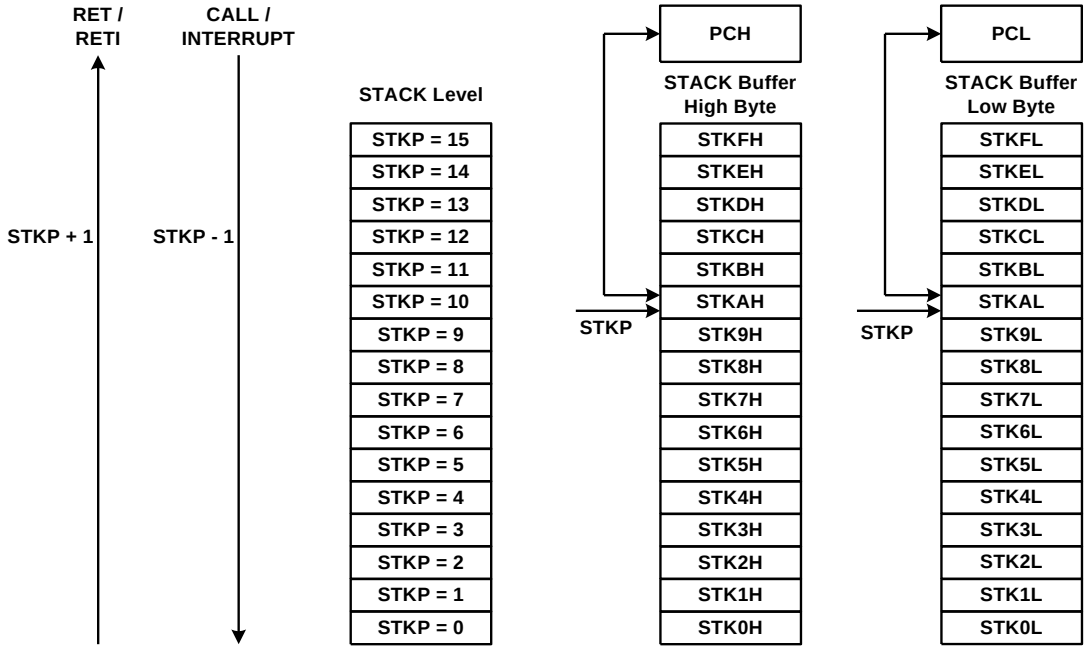
Example: Indirectly addressing mode with @YZ register

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

2.4 STACK OPERATION

2.4.1 OVERVIEW

The stack buffer has 16-level. These buffers are designed to push and pop up program counter's (PC) data when interrupt service routine and "CALL" instruction are executed. The STKP register is a pointer designed to point active level in order to push or pop up data from stack buffer. The STKnH and STKnL are the stack buffers to store program counter (PC) data.



2.4.2 STACK POINTER

The stack pointer (STKP) is a 4-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.

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	LVD23	LVD30	-	STKPB3	STKPB2	STKPB1	STKPB0
Read/Write	R/W	R	R	-	R/W	R/W	R/W	R/W
After reset	0	-	-	-	1	1	1	1

Bit[2:0] **STKPBn:** Stack pointer (n = 0 ~ 3)

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

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

```
MOV     A, #00001111B
B0MOV   STKP, A
```

2.4.3 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.

0E0H~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

0E0H~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 = F ~ 0)

2.4.4 STACK OVERFLOW INDICATOR

If stack pointer is normal and not overflow, the program execution is correct. If stack overflows, the program counter would be incorrect making program execution error. STKOV bit is stack pointer overflow indicator to monitor stack pointer status. When STKOV=0, stack pointer status is normal. If STKOV=1, stack overflow occurs, and the program execution would be error. The program can take measures to recover program execution from stack overflow situation through STKOV bit.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	RST	STKOV	-	C	DC	Z
Read/Write	R	R	R	R	-	R/W	R/W	R/W
After Reset	-	-	-	-	-	0	0	0

Bit 4 **STKOV:** Stack overflow indicator.
0 = Non-overflow.
1 = Stack overflow.

✸ ***Note: If STKOV bit is set as stack overflowing, only system reset event can clear STKOV bit, e.g. watchdog timer overflow, external reset pin low status or LVD reset.***

➢ **Example: Stack overflow protection through watchdog reset. Watchdog timer must be enabled.**

```
MAIN:
        …
StackChk:
        B0BTS1  STKOV
        JMP     MAIN    ; STKOV=0, program keeps executing.
        JMP     $       ; STKOV=1, stack overflows, and use “jump here” operation
                        ; making watchdog timer overflow to trigger system reset.
```

➢ **Example: Stack overflow protection through external reset. External reset function must be enabled, and one GPIO pin (output mode) connects to external reset pin.**

```
MAIN:
        …
StackChk:
        B0BTS1  STKOV
        JMP     MAIN    ; STKOV=0, program keeps executing.
        B0BCLR  P1.0    ; STKOV=1, stack overflows, and set P1.0 output low status to
                        ; force reset pin to low status to trigger system reset.
```

2.4.5 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		STKOV	Description
	STKPB2	STKPB2	STKPB1	STKPB0	High Byte	Low Byte		
0	1	1	1	1	Free	Free	0	-
1	1	1	1	0	STK0H	STK0L	0	-
2	1	1	0	1	STK1H	STK1L	0	-
3	1	1	0	0	STK2H	STK2L	0	-
4	1	0	1	1	STK3H	STK3L	0	-
5	1	0	1	0	STK4H	STK4L	0	-
6	1	0	0	1	STK5H	STK5L	0	-
7	1	0	0	0	STK6H	STK6L	0	-
8	0	1	1	1	STK7H	STK7L	0	-
9	0	1	1	0	STK8H	STK8L	0	-
10	0	1	0	1	STK9H	STK9L	0	-
11	0	1	0	0	STKAH	STKAL	0	-
12	0	0	1	1	STKBH	STKBL	0	-
13	0	0	1	0	STKCH	STKCL	0	-
14	0	0	0	1	STKDH	STKDL	0	-
15	0	0	0	0	STKEH	STKEL	0	-
16	1	1	1	1	STKFH	STKFL	0	-
> 16	1	1	1	0	-	-	**1**	**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		STKOV	Description
	STKPB2	STKPB2	STKPB1	STKPB0	High Byte	Low Byte		
16	1	1	1	1	STKFH	STKFL	0	-
15	0	0	0	0	STKEH	STKEL	0	-
14	0	0	0	1	STKDH	STKDL	0	-
13	0	0	1	0	STKCH	STKCL	0	-
12	0	0	1	1	STKBH	STKBL	0	-
11	0	1	0	0	STKAH	STKAL	0	-
10	0	1	0	1	STK9H	STK9L	0	-
9	0	1	1	0	STK8H	STK8L	0	-
8	0	1	1	1	STK7H	STK7L	0	-
7	1	0	0	0	STK6H	STK6L	0	-
6	1	0	0	1	STK5H	STK5L	0	-
5	1	0	1	0	STK4H	STK4L	0	-
4	1	0	1	1	STK3H	STK3L	0	-
3	1	1	0	0	STK2H	STK2L	0	-
2	1	1	0	1	STK1H	STK1L	0	-
1	1	1	1	0	STK0H	STK0L	0	-
0	1	1	1	1	Free	Free	0	-

✸ ***Note: When stack overflow occurs, the system detects the condition and set STKOV flag ("Logic 1"). STKOV flag can't be cleared by program.***

CODE OPTION TABLE

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

Code Option	Content	Function Description
High_Clk	IHRC_16M	High speed internal 16MHz RC. XIN/XOUT pins are bi-direction GPIO mode.
	IHRC_RTC	High speed internal 16MHz RC. XIN/XOUT pins are connected to external 32768Hz crystal.
	RC	Low cost RC for external high clock oscillator. XIN pin is connected to RC oscillator. XOUT pin is bi-direction GPIO mode.
	32K X'tal	Low frequency, power saving crystal (e.g. 32.768KHz) for external high clock oscillator.
	12M X'tal	High speed crystal /resonator (e.g. 12MHz) for external high clock oscillator.
	4M X'tal	Standard crystal /resonator (e.g. 4M) for external high clock oscillator.
High_Fcpu	Fhosc/1	Normal mode instruction cycle is 1 high speed oscillator clocks.
	Fhosc/2	Normal mode instruction cycle is 2 high speed oscillator clocks.
	Fhosc/4	Normal mode instruction cycle is 4 high speed oscillator clocks.
	Fhosc/8	Normal mode instruction cycle is 8 high speed oscillator clocks.
	Fhosc/16	Normal mode instruction cycle is 16 high speed oscillator clocks.
	Fhosc/32	Normal mode instruction cycle is 32 high speed oscillator clocks.
	Fhosc/64	Normal mode instruction cycle is 64 high speed oscillator clocks.
	Fhosc/128	Normal mode instruction cycle is 128 high speed oscillator clocks.
Low_Fcpu	Flosc/1	Slow mode instruction cycle is 1 low speed oscillator clocks.
	Flosc/2	Slow mode instruction cycle is 2 low speed oscillator clocks.
	Flosc/4	Slow mode instruction cycle is 4 low speed oscillator clocks.
	Flosc/8	Slow mode instruction cycle is 8 low speed oscillator clocks.
Noise_Filter	Enable	Enable Noise Filter.
	Disable	Disable Noise Filter.
WDT_CLK	Flosc/4	Watchdog timer clock source Flosc/4.
	Flosc/8	Watchdog timer clock source Flosc/8.
	Flosc/16	Watchdog timer clock source Flosc/16.
	Flosc/32	Watchdog timer clock source Flosc/32.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
Reset_Pin	Reset	Enable External reset pin.
	P13	Enable P1.3.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
LVD	LVD_L	LVD will reset chip if VDD is below 1.8V
	LVD_M	LVD will reset chip if VDD is below 1.8V Enable LVD23 bit of PFLAG register for 2.3V low voltage indicator.
	LVD_H	LVD will reset chip if VDD is below 2.3V Enable LVD30 bit of PFLAG register for 3.0V low voltage indicator.
	LVD_MAX	LVD will reset chip if VDD is below 3.0V

2.4.6 Fcpu Code Option

Fcpu means instruction cycle whose clock source includes high/low speed oscillator in different operating modes. High_Fcpu and Low_Fcpu code options select instruction cycle pre-scaler to decide instruction cycle rate. In normal mode (high speed clock), the system clock source is high speed oscillator, and Fcpu clock rate has eight options including Fhosc/1, Fhosc/2, Fhosc/4, Fhosc/8, Fhosc/16, Fhosc/32, Fhosc/64, Fhosc/128. In slow mode (low speed clock), the system clock source is internal low speed RC oscillator, and the Fcpu including Flosc/1, Flosc/2, Flosc/4, Flosc/8.

2.4.7 Reset_Pin code option

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

- **Reset:** The reset pin is external reset function. When falling edge trigger occurring, the system will be reset.
- **P13:** Set reset pin to general bi-direction pin (P1.3). The external reset function is disabled and the pin is bi-direction pin.

2.4.8 Security code option

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

2.4.9 Noise Filter code option

Noise Filter code option is a power noise filter manner to reduce noisy effect of system clock. If noise filter enable, In high noisy environment, enable noise filter, enable watchdog timer and select a good LVD level can make whole system work well and avoid error event occurrence.

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 POR, WDT and RST flags indicate system reset status. The system can depend on POR, WDT and RST status and go to different paths by program.

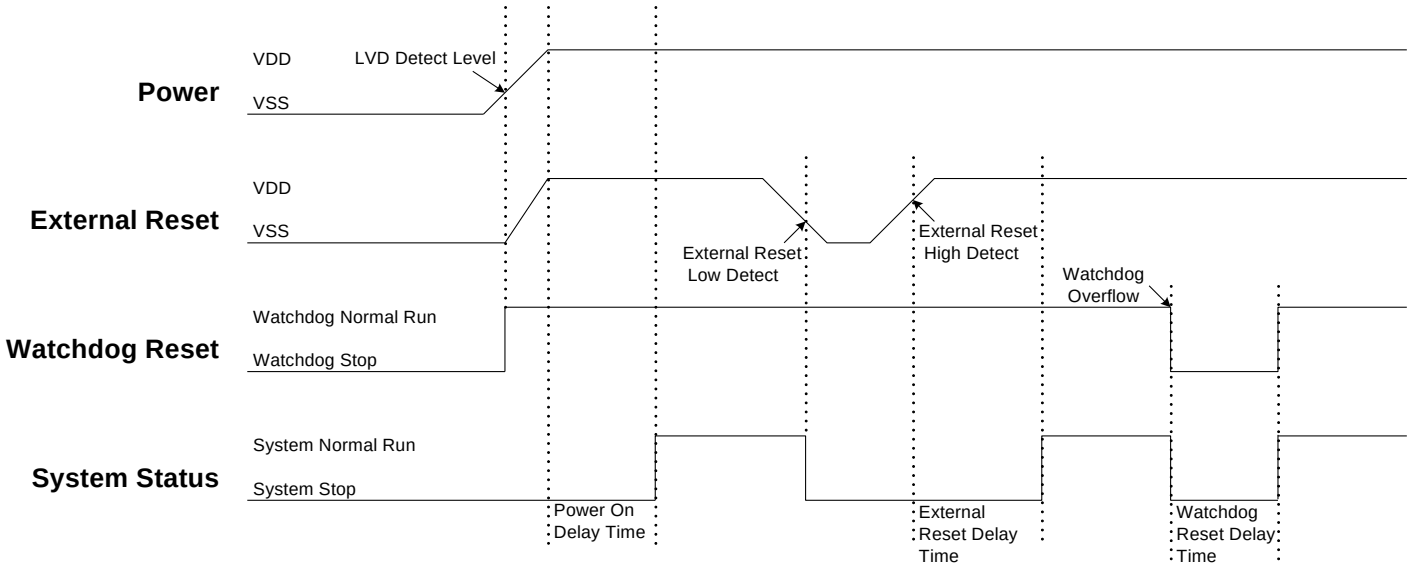
086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	RST	STKOV	-	C	DC	Z
Read/Write	R	R	R	R	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

Bit 7 **POR:** Power on reset and LVD brown-out reset indicator.
0 = Non-active.
1 = Reset active. LVD announces reset flag.

Bit 6 **WDT:** Watchdog reset indicator.
0 = Non-active.
1 = Reset active. Watchdog announces reset flag.

Bit 5 **RST:** External reset indicator.
0 = Non-active.
1 = Reset active. External reset announces reset flag.

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.

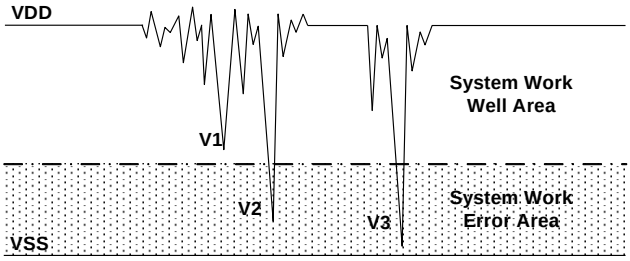
Watchdog timer application note is as following.

- 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

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



Brown Out Reset Diagram

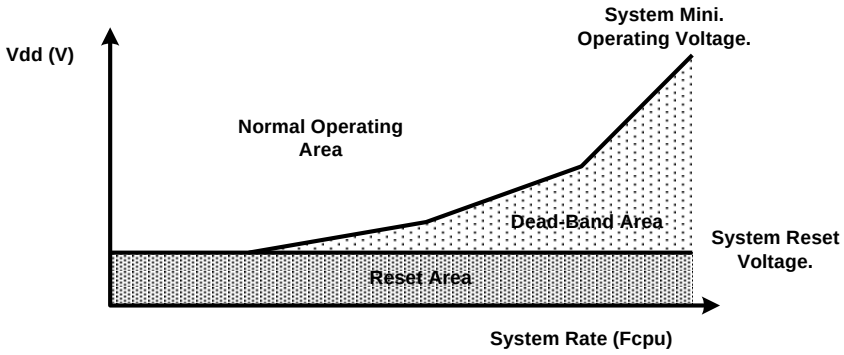
The power dropping might through the voltage range that's the system dead-band. The dead-band means the power range can't offer the system minimum operation power requirement. The above diagram is a typical brown out reset diagram. There is a serious noise under the VDD, and VDD voltage drops very deep.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

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

AC application:
In AC power application, the DC power is regulated from AC power source. This kind of power usually couples with AC noise that makes the DC power dirty. Or the external loading is very heavy, e.g. driving motor. The loading operating induces noise and overlaps with the DC power. VDD drops by the noise, and the system works under unstable power situation.
The power on duration and power down duration are longer in AC application. The system power on sequence protects the power on successful, but the power down situation is like DC low battery condition. When turn off the AC power, the VDD drops slowly and through the dead-band for a while.

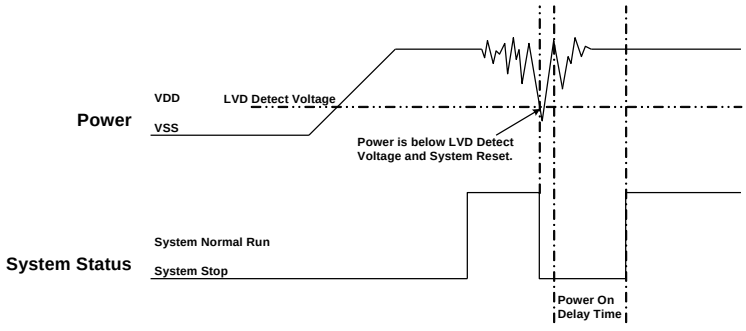
3.4.1 THE SYSTEM OPERATING VOLTAGE

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



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

3.4.2 LOW VOLTAGE DETECTOR (LVD)



The LVD (low voltage detector) is built-in Sonix 8-bit MCU to be brown out reset protection. When the VDD drops and is below LVD detect voltage, the LVD would be triggered, and the system is reset. The LVD detect level is different by each MCU. The LVD voltage level is a point of voltage and not easy to cover all dead-band range. Using LVD to improve brown out reset is depend on application requirement and environment. If the power variation is very deep, violent and trigger the LVD, the LVD can be the protection. If the power variation can touch the LVD detect level and make system work error, the LVD can't be the protection and need to other reset methods. More detail LVD information is in the electrical characteristic section.
The LVD is three levels design (1.8V/2.3V/3.0V) and controlled by LVD code option. The 1.8V LVD is always enable for power on reset and Brown Out reset. The 2.3V LVD includes LVD reset function and flag function to indicate VDD status function. The 3.0V includes flag function to indicate VDD status. LVD flag function can be an **easy low battery detector**. LVD23, LVD30 flags indicate VDD voltage level. For low battery detect application, only checking LVD23, LVD30 status to be battery status. This is a cheap and easy solution.

0EFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	LVD23	LVD30	-	STKPB3	STKPB2	STKPB1	STKPB0
Read/Write	R/W	R	R	-	R/W	R/W	R/W	R/W
After Reset	0	-	-	-	1	1	1	1

Bit 6 **LVD23:** LVD23 low voltage detect indicator.
0 = Vdd > LVD23 detect level.
1 = Vdd < LVD23 detect level.

Bit 5 **LVD30:** LVD30 low voltage detect indicator.
0 = Vdd > LVD30 detect level.
1 = Vdd < LVD30 detect level.

LVD	LVD Code Option		
	LVD_L	LVD_M	LVD_H
1.8V Reset	Available	Available	Available
2.3V Flag	-	Available	-
2.3V Reset	-	-	Available
3.0V Flag	-	-	Available

LVD_L
If VDD < 1.8V, system will be reset.
Disable LVD23 and LVD30 bit of PFLAG register.
LVD_M
If VDD < 1.8V, system will be reset.
Enable LVD23 bit of PFLAG register. If VDD > 2.3V, LVD23 is "0". If VDD <= 2.3V, LVD23 flag is "1".
Disable LVD30 bit of PFLAG register.
LVD_H
If VDD < 2.3V, system will be reset.
Enable LVD23 bit of PFLAG register. If VDD > 2.3V, LVD23 is "0". If VDD <= 2.3V, LVD23 flag is "1".
Enable LVD30 bit of PFLAG register. If VDD > 3.0V, LVD30 is "0". If VDD <= 3.0V, LVD30 flag is "1".
LVD_MAX
If VDD < 3.0V, system will be reset.

✸ ***Note:***
1. ***After any LVD reset, LVD23, LVD30 flags are cleared.***
2. ***The voltage level of LVD 2.3V or 3.0V is for design reference only. Don't use the LVD indicator as precision VDD measurement.***

3.4.3 BROWN OUT RESET IMPROVEMENT

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

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

✷ ***Note:***

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

Watchdog reset:

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

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

Reduce the system executing rate:

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

External reset circuit:

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

3.5 EXTERNAL RESET

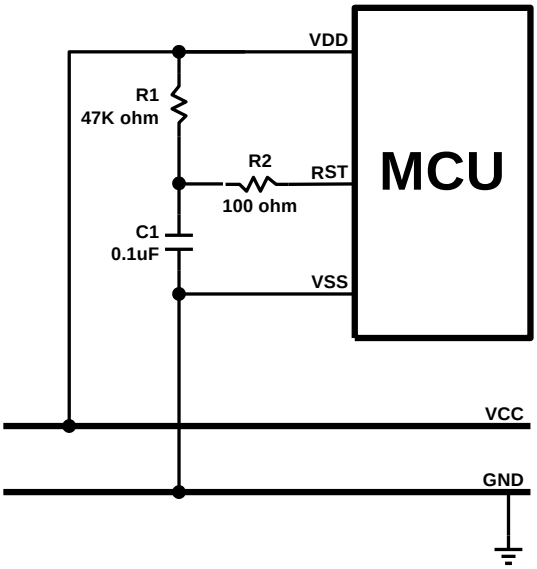
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

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

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

3.6 EXTERNAL RESET CIRCUIT

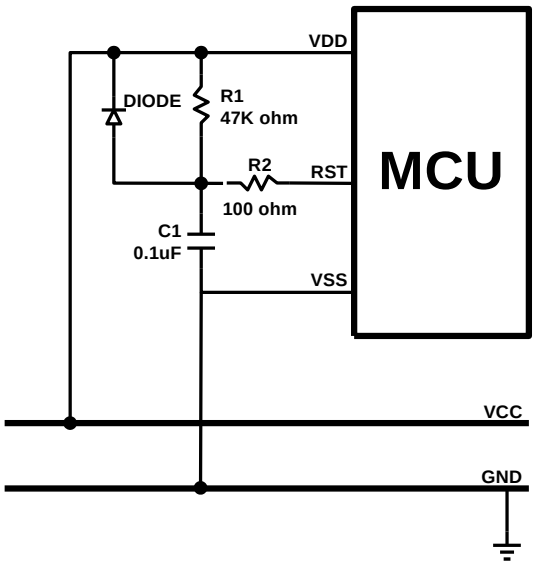
3.6.1 Simply RC Reset Circuit



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

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

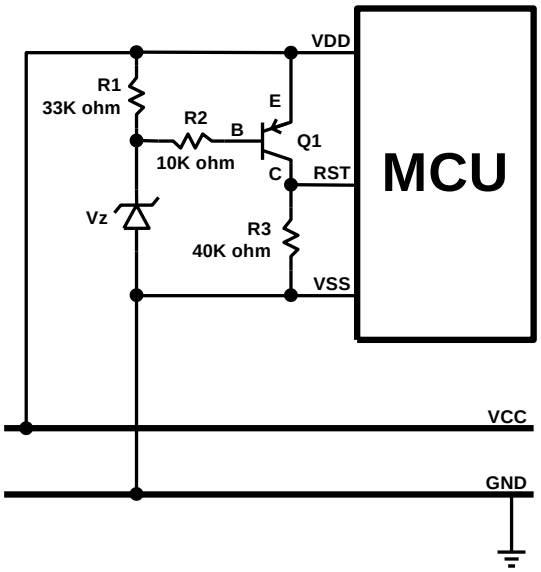
3.6.2 Diode & RC Reset Circuit



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

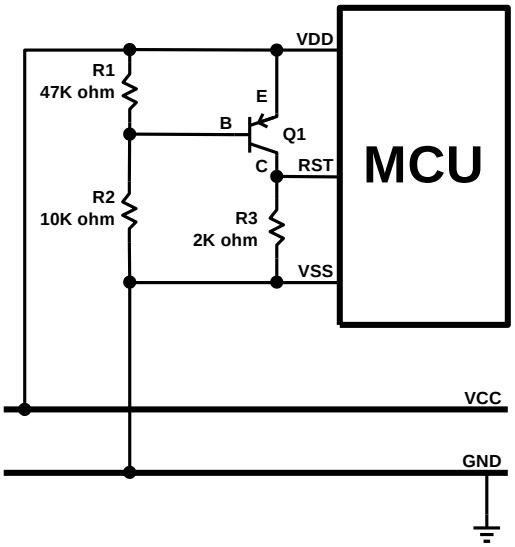
✸ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.6.3 Zener Diode Reset Circuit



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

3.6.4 Voltage Bias Reset Circuit

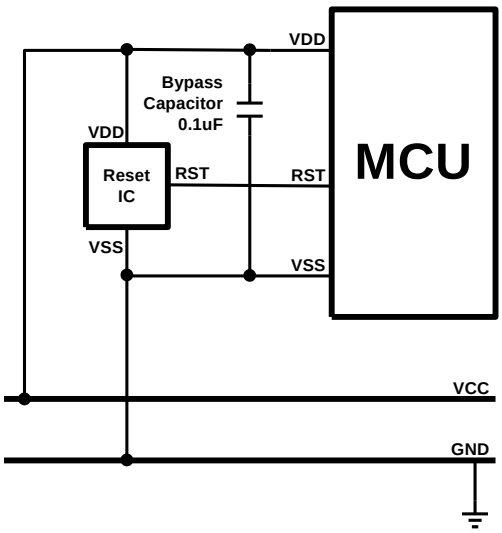


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to “0.7V x (R1 + R2) / R1”,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

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

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

3.6.5 External Reset IC



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

4 SYSTEM CLOCK

4.1 OVERVIEW

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

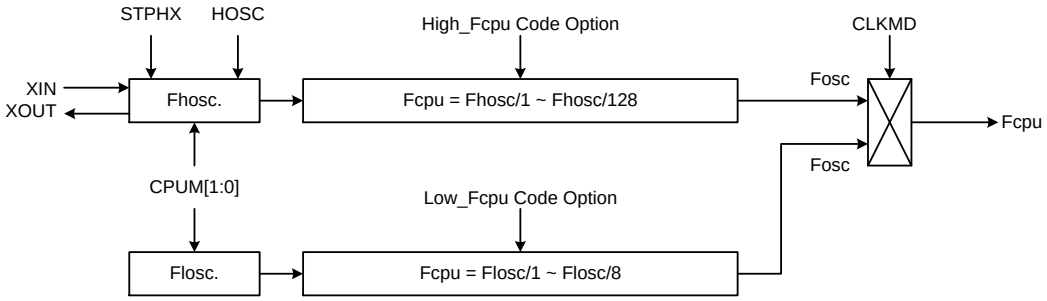
- **High-speed oscillator**

Internal high-speed oscillator is 16MHz RC type called “**IHRC**” and “**IHRC_RTC**”.
External high-speed oscillator includes crystal/ceramic (4MHz, 12MHz, 32KHz) and RC type.

- **Low-speed oscillator**

Internal low-speed oscillator is 16KHz RC type called “**ILRC**”.

- **System clock block diagram**



- HOSC: High_Clk code option.
- Fhosc: External high-speed clock / Internal high-speed RC clock.
- Flosc: Internal low-speed RC clock (about 16KHz@3V and @5V).
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.2 FCPU (INSTRUCTION CYCLE)

The system clock rate is instruction cycle called “**Fcpu**” which is divided from the system clock source and decides the system operating rate. Fcpu rate is selected by High_Fcpu code option and the range is **Fhosc/1~Fhosc/128** under system normal mode. If the system high clock source is external 4MHz crystal, and the High_Fcpu code option is Fhosc/4, the Fcpu frequency is 4MHz/4 = 1MHz. Under system slow mode, the Fcpu range is **Flosc/1~Flosc/8** controlled by Low_Fcpu code option, If Low_Fcpu code option is Flosc/4, the Fcpu frequency is 16KHz/4=4KHz.

4.3 NOISE FILTER

The Noise Filter controlled by “Noise_Filter” code option is a low pass filter and supports external oscillator including RC and crystal modes. The purpose is to filter high rate noise coupling on high clock signal from external oscillator.
In high noisy environment, enable “Noise_Filter” code option is the strongly recommendation to reduce noise effect.

4.4 SYSTEM HIGH-SPEED CLOCK

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

4.4.1 HIGH_CLK CODE OPTION

For difference clock functions, Sonix provides multi-type system high clock options controlled by “High_CLK” code option. The High_CLK code option defines the system oscillator types including IHRC_16M, IHRC_RTC, RC, 32K X’tal, 12M X’tal and 4M X’tal. These oscillator options support different bandwidth oscillator.

- **IHRC_16M:** The system high-speed clock source is internal high-speed 16MHz RC type oscillator. In the mode, XIN and XOUT pins are bi-direction GPIO mode, and not to connect any external oscillator device.
- **IHRC_RTC:** The system high-speed clock source is internal high-speed 16MHz RC type oscillator. The RTC clock source is external low-speed 32768Hz crystal. The XIN and XOUT pins are defined to drive external 32768Hz crystal and disables GPIO function.
- **RC:** The system high-speed clock source is external low cost RC type oscillator. The RC oscillator circuit only connects to XIN pin, and the XOUT pin is bi-direction GPIO mode.
- **32K X’tal:** The system high-speed clock source is external low-speed 32768Hz crystal. The option only supports 32768Hz crystal and the RTC function is workable.
- **12M X’tal:** The system high-speed clock source is external high-speed crystal/ceramic. The oscillator bandwidth is 10MHz~16MHz.
- **4M X’tal:** The system high-speed clock source is external high-speed crystal/resonator. The oscillator bandwidth is 1MHz~10MHz.

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

4.4.2 INTERNAL HIGH-SPEED OSCILLATOR RC TYPE (IHRC)

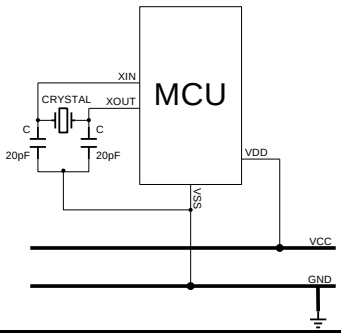
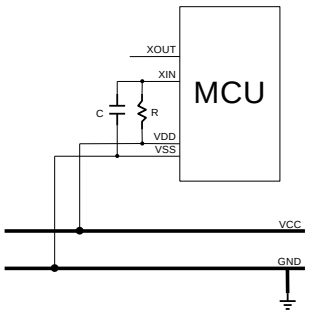
The internal high-speed oscillator is 16MHz RC type. The accuracy is ±2% under commercial condition. When the “High_CLK” code option is “IHRC_16M” or “IHRC_RTC”, the internal high-speed oscillator is enabled.

- **IHRC_16M:** The system high-speed clock is internal 16MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.
- **IHRC_RTC:** The system high-speed clock is internal 16MHz oscillator RC type, and the real time clock is external 32768Hz crystal. XIN/XOUT pins connect with external 32768Hz crystal.

4.4.3 EXTERNAL HIGH-SPEED OSCILLATOR

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

4.4.4 EXTERNAL OSCILLATOR APPLICATION CIRCUIT

CRYSTAL/CERAMIC	RC Type
	

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

4.5 SYSTEM LOW-SPEED CLOCK

The system low clock source is the internal low-speed oscillator built in the micro-controller. The low-speed oscillator uses RC type oscillator circuit. The frequency is affected by the voltage and temperature of the system. In common condition, the frequency of the RC oscillator is about 16KHz.

The internal low RC supports watchdog clock source and system slow mode controlled by “CLKMD” bit of OSCM register.

- ***Flosc = Internal low RC oscillator (about 16KHz).***
- ***Slow mode Fcpu = Flosc/ 1 ~ Flosc/8 controlled by Low_Fcpu code option.***

When watchdog timer is disabled and system is in power down mode, the internal low RC stops.

➢ **Example: Stop internal low-speed oscillator by power down mode as watchdog timer disable**

```
B0BSET    FGCHS2      ; To set AIN2~AIN5 aren't ADC input channel to reduce power
B0MOV     A, #0x18    ; consumption.
B0MOV     AIN2M, A
B0MOV     AIN3M, A
B0MOV     AIN4M, A
B0MOV     AIN5M, A

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

4.6 OSCM REGISTER

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

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-
Read/Write	-	-	-	R/W	R/W	R/W	R/W	-
After reset	-	-	-	0	0	0	0	-

Bit 1 **STPHX:** External high-speed oscillator control bit.
0 = External high-speed oscillator free run.
1 = External high-speed oscillator free run stop. Internal low-speed RC oscillator is still running.

Bit 2 **CLKMD:** System high/Low clock mode control bit.
0 = Normal (dual) mode. System clock is high clock.
1 = Slow mode. System clock is internal low clock.

Bit[4:3] **CPUM[1:0]:** CPU operating mode control bits.
00 = normal.
01 = sleep (power down) mode.
10 = green mode.
11 = reserved.

“STPHX” bit controls internal high speed RC type oscillator and external oscillator operations. When “STPHX=0”, the external oscillator or internal high speed RC type oscillator active. When “STPHX=1”, the external oscillator or internal high speed RC type oscillator are disabled. The STPHX function is depend on different high clock options to do different controls.

- **IHRC_16M: “STPHX=1” disables internal high speed RC type oscillator.**
- **IHRC_RTC: “STPHX=1” disables internal high speed RC type oscillator, and external 32768Hz crystal keeps oscillating.**
- **RC, 4M, 12M, 32K: “STPHX=1” disables external oscillator.**

4.7 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.**

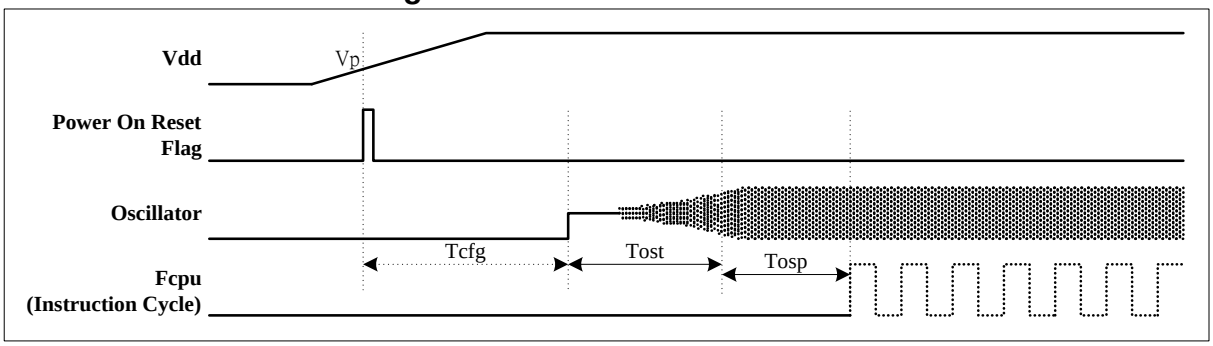
```
        B0BSET    P0M.0     ; Set P0.0 to be output mode for outputting Fcpu toggle signal.
@@:
        B0BSET    P0.0      ; Output Fcpu toggle signal in low-speed clock mode.
        B0BCLR    P0.0      ; Measure the Fcpu frequency by oscilloscope.
        JMP       @B
```

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

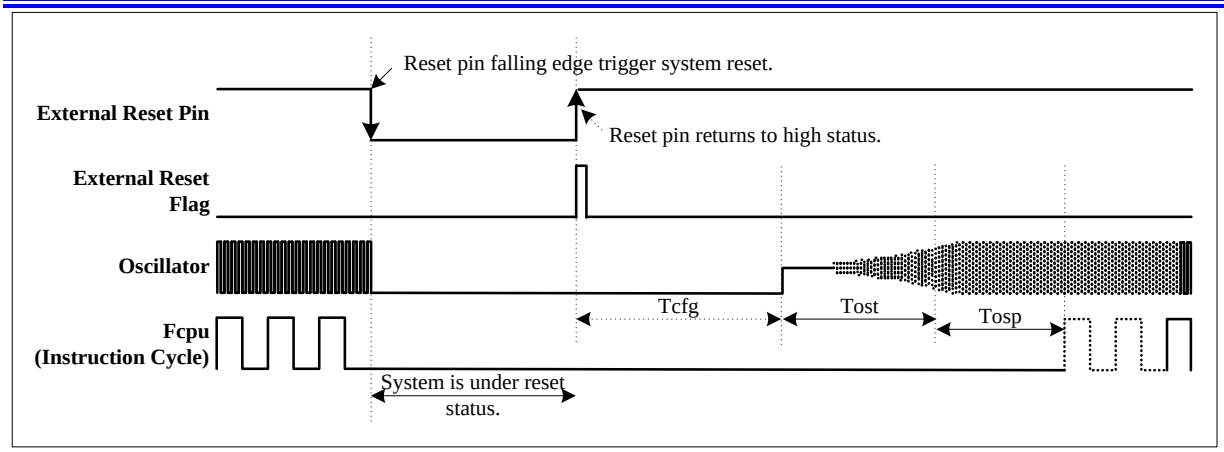
4.8 SYSTEM CLOCK TIMING

Parameter	Symbol	Description	Typical
Hardware configuration time	Tcfg	2048*F_{ILRC}	64ms @ F_{ILRC} = 32KHz 128ms @ F_{ILRC} = 16KHz
Oscillator start up time	Tost	The start-up time is depended on oscillator's material, factory and architecture. Normally, the low-speed oscillator's start-up time is lower than high-speed oscillator. The RC type oscillator's start-up time is faster than crystal type oscillator.	-
Oscillator warm-up time	Tosp	Oscillator warm-up time of reset condition. 2048*F_{hosc} (Power on reset, LVD reset, watchdog reset, external reset pin active.)	64ms @ F_{hosc} = 32KHz 512us @ F_{hosc} = 4MHz 128us @ F_{hosc} = 16MHz
		Oscillator warm-up time of power down mode wake-up condition. 2048*F_{hosc} ……Crystal/resonator type oscillator, e.g. 32768Hz crystal, 4MHz crystal, 16MHz crystal… 32*F_{hosc}……RC type oscillator, e.g. external RC type oscillator, internal high-speed RC type oscillator.	X'tal: 64ms @ F_{hosc} = 32KHz 512us @ F_{hosc} = 4MHz 128us @ F_{hosc} = 16MHz RC: 8us @ F_{hosc} = 4MHz 2us @ F_{hosc} = 16MHz

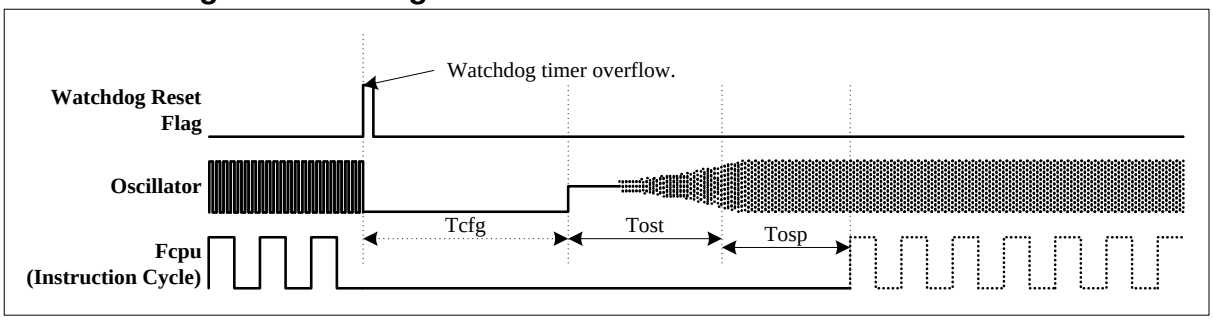
- **Power On Reset Timing**



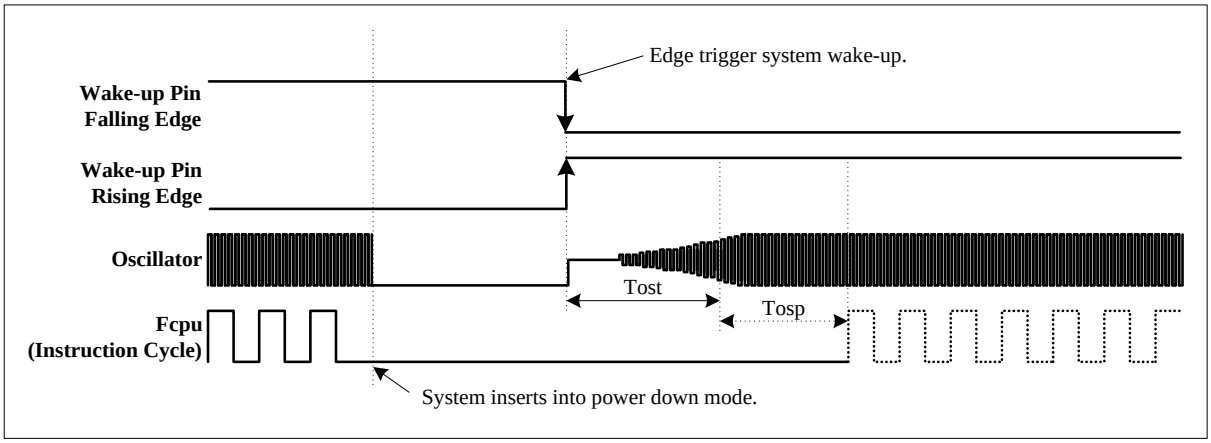
- **External Reset Pin Reset Timing**



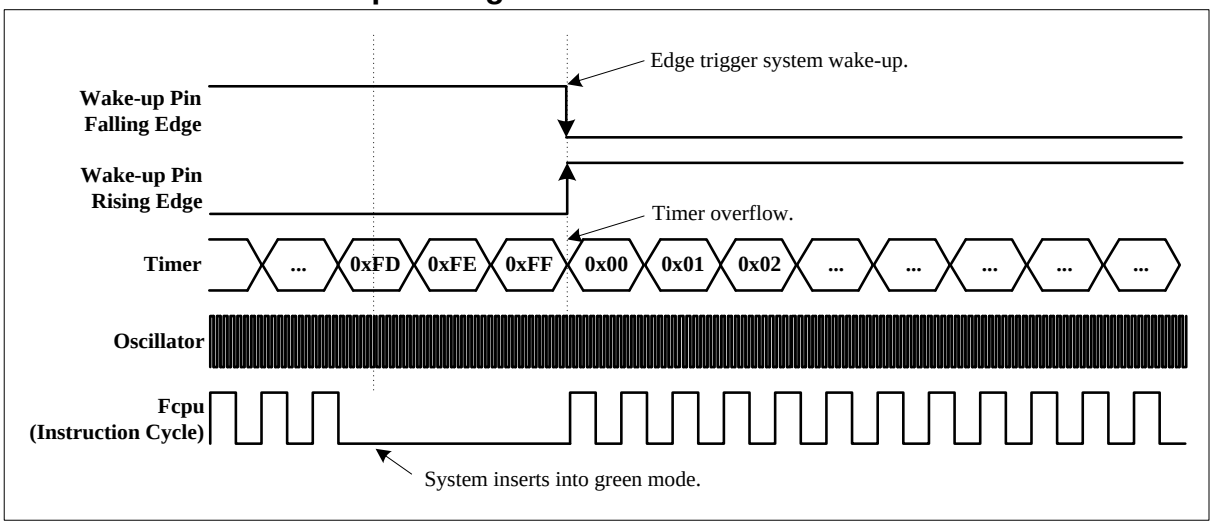
Watchdog Reset Timing



Power Down Mode Wake-up Timing

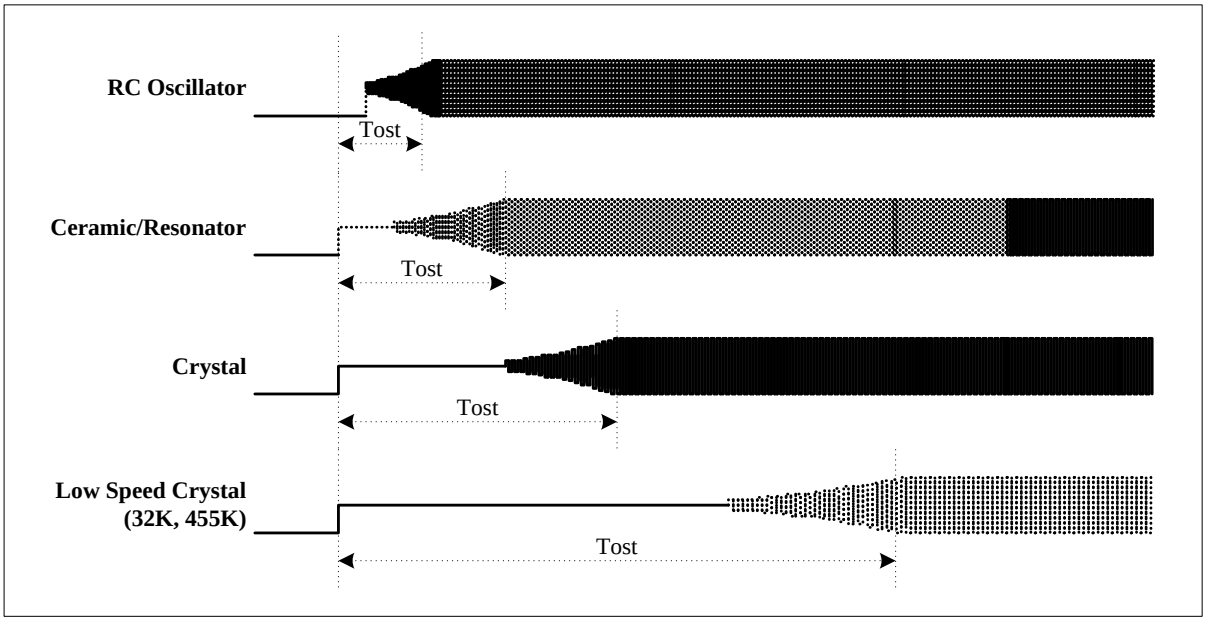


Green Mode Wake-up Timing



Oscillator Start-up Time

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



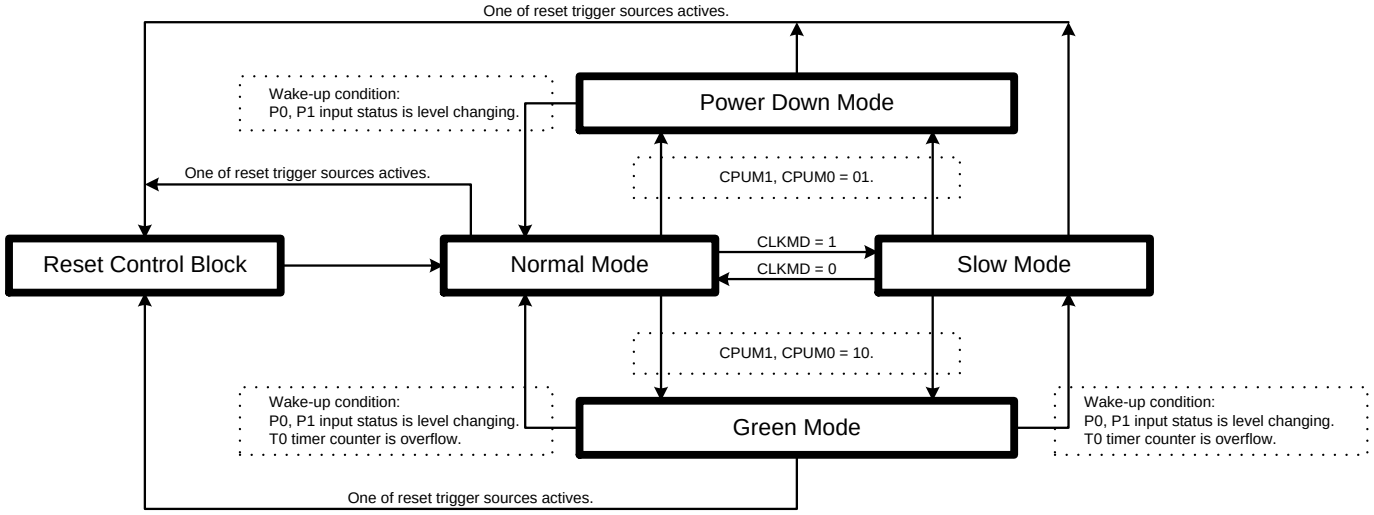
5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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

Operating Mode Control Block



Operating Mode Clock Control Table

Operating Mode	Normal Mode	Slow Mode	Green Mode	Power Down Mode
IHRC	**IHRC, IHRC_RTC:** Running **Ext. OSC:** Disable	**IHRC, IHRC_RTC:** By STPHX **Ext. OSC:** Disable	**IHRC, IHRC_RTC:** By STPHX **Ext. OSC:** Disable	Stop
ILRC	Running	Running	Running	Stop
Ext. Osc.	**IHRC:** Disable **IHRC_RTC, Ext. OSC:** Running	**IHRC:** Disable **IHRC_RTC:** Running **Ext. OSC:** By STPHX	**IHRC:** By STPHX **IHRC_RTC:** Running **Ext. OSC:** By STPHX	Stop
CPU instruction	Executing	Executing	Stop	Stop
T0 timer	Active By T0ENB	Active By T0ENB	Active By T0ENB	Inactive
TC0 timer (Timer, Event counter, PWM)	Active By TC0ENB	Active By TC0ENB	Active By TC0ENB	Inactive
T1 timer (Timer)	Active By T1ENB	Active By T1ENB	Active By T1ENB	Inactive
SIO	Active as enable	Inactive	Inactive	Inactive
MSP	Active as enable	Inactive	Inactive	Inactive
UART	Active as enable	Inactive	Inactive	Inactive
ADC	Active as enable	Active as enable	Active as enable	Inactive
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option
Internal interrupt	All active	All active	All active	All inactive
External interrupt	All active	All active	All active	All inactive
Wakeup source	-	-	P0, P1, T0, Reset	P0, P1, Reset

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

✸ ***Note:***
1. ***SIO, MSP and UART inactive in slow mode and green mode, because the clock source doesn't exist. Use firmware to disable SIO, MSP, UART function before inserting slow mode and green mode.***
2. ***In IHRC_RTC mode, STPHX only controls IHRC, not Ext. 32K. STPHX=0, IHRC actives. STPHX=1, IHRC stops.***

5.2 NORMAL MODE

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

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

5.3 SLOW MODE

The slow mode is system low clock operating mode. The system clock source is from internal low speed RC type oscillator. The slow mode is controlled by CLKMD bit of OSCM register. When CLKMD=0, the system is in normal mode. When CLKMD=1, the system inserts into slow mode. The high speed oscillator won't be disabled automatically after switching to slow mode, and must be disabled by SPTHX bit to reduce power consumption. In slow mode, the system rates are Flosc/1, Flosc/2, Flosc/4, Flosc/8 (Flosc is internal low speed RC type oscillator frequency) controlled by code option.

- The program is executed, and full functions are controllable.
- The system rate is low speed (Flosc/1, Flosc/2, Flosc/4, Flosc/8 controlled by code option).
- The internal low speed RC type oscillator actives, and the high speed oscillator is controlled by STPHX=1. In slow mode, to stop high speed oscillator is strongly recommendation.
- Slow mode can be switched to other operating modes through OSCM register.
- Power down mode from slow mode is wake-up to normal mode.
- Normal mode is switched to slow mode.
- Green mode from slow mode is wake-up to slow mode.
-

5.4 POWER DOWN MDOE

The power down mode is the system ideal status. No program execution and oscillator operation. Only internal regulator actives to keep all control gates status, register status and SRAM contents. The power down mode is waked up by P0, P1 hardware level change trigger. P0 wake-up function is always enables, and P1 wake-up function is controlled by P1W register. Any operating modes into power down mode, the system is waked up to normal mode. Inserting power down mode is controlled by CPUM0 bit of OSCM register. When CPUM0=1, the system inserts into power down mode. After system wake-up from power down mode, the CPUM0 bit is disabled (zero status) automatically, and the WAKE bit set as "1".

- The program stops executing, and full functions are disabled.
- All oscillators including external high speed oscillator, internal high speed oscillator and internal low speed oscillator stop.
- The system inserts into normal mode after wake-up from power down mode.
- The power down mode wake-up source is P0 and P1 level change trigger.
- After system wake-up from power down mode, the WAKE bit set as "1" and cleared by program.
- If wake-up source is external interrupt source, the WAKE bit won't be set, and external interrupt IRQ bit is set. The system issues external interrupt request and executes interrupt service routine.

✸ ***Note: If the system is in normal mode, to set STPHX=1 to disable the high clock oscillator. The system is under no system clock condition. This condition makes the system stay as power down mode, and can be wake-up by P0, P1 level change trigger.***

5.5 GREEN MODE

The green mode is another system ideal status not like power down mode. In power down mode, all functions and hardware devices are disabled. But in green mode, the system clock source keeps running, so the power consumption of green mode is larger than power down mode. In green mode, the program isn't executed, but the timer with wake-up function actives as enabled, and the timer clock source is the non-stop system clock. The green mode has 2 wake-up sources. One is the P0, P1 level change trigger wake-up. The other one is internal timer with wake-up function occurring overflow. That's mean users can setup one fix period to timer, and the system is waked up until the time out. Inserting green mode is controlled by CPUM1 bit of OSCM register. When CPUM1=1, the system inserts into green mode. After system wake-up from green mode, the CPUM1 bit is disabled (zero status) automatically, and the WAKE bit set as "1".

- The program stops executing, and full functions are disabled.
- Only the timer with wake-up function actives.
- The oscillator to be the system clock source keeps running, and the other oscillators operation is depend on system operation mode configuration.
- If inserting green mode from normal mode, the system insets to normal mode after wake-up.
- If inserting green mode from slow mode, the system insets to slow mode after wake-up.
- The green mode wake-up sources are P0, P1 level change trigger and unique time overflow.
- After system wake-up from power down mode, the WAKE bit set as "1" and cleared by program.
- If wake-up source is external interrupt source, the WAKE bit won't be set, and external interrupt IRQ bit is set. The system issues external interrupt request and executes interrupt service routine.
- If the function clock source is system clock, the functions are workable as enabled and under green mode, e.g. Timer, PWM, event counter…But the functions doesn't has wake-up function.

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

5.6 OPERATING MODE CONTROL MACRO

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

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

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

```
B0BSET      FGCHS2          ; To set AIN2~AIN5 aren't ADC input channel to reduce
B0MOV       A, #0x18        ; power consumption.
B0MOV       AIN2M, A
B0MOV       AIN3M, A
B0MOV       AIN4M, A
B0MOV       AIN5M, A

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

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

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

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

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

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

```
B0BSET      FGCHS2          ; To set AIN2~AIN5 aren't ADC input channel to reduce
B0MOV       A, #0x18        ; power consumption.
B0MOV       AIN2M, A
B0MOV       AIN3M, A
B0MOV       AIN4M, A
B0MOV       AIN5M, A

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

➢ **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
        B0BSET      FGCHS2      ; To set AIN2~AIN5 aren't ADC input channel to reduce
        B0MOV       A, #0x18    ; power consumption.
        B0MOV       AIN2M, A
        B0MOV       AIN3M, A
```

```
B0MOV       AIN4M, A
B0MOV       AIN5M, A

GreenMode                   ; Declare “GreenMode” macro directly.
```

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

```
CLR         T0C             ; Clear T0 counter.
B0BSET      FT0TB           ; Enable T0 RTC function.
B0BSET      FT0ENB          ; To enable T0 timer.

; Go into green mode
B0BSET      FGCHS2          ; To set AIN2~AIN5 aren’t ADC input channel to reduce
B0MOV       A, #0x18        ; power consumption.
B0MOV       AIN2M, A
B0MOV       AIN3M, A
B0MOV       AIN4M, A
B0MOV       AIN5M, A

GreenMode                   ; Declare “GreenMode” macro directly.
```

5.7 WAKEUP

5.7.1 OVERVIEW

Under power down mode (sleep mode) or green mode, program doesn't execute. The wakeup trigger can wake the system up to normal mode or slow mode. The wakeup trigger sources are external trigger (P0/P1 level change) and internal trigger (T0 timer overflow). The wakeup function builds in interrupt operation issued IRQ flag and trigger system executing interrupt service routine as system wakeup occurrence.

- Power down mode is waked up to normal mode. The wakeup trigger is only external trigger (P0/P1 level change)
- Green mode is waked up to last mode (normal mode or slow mode). The wakeup triggers are external trigger (P0/P1 level change) and internal trigger (T0 timer overflow).
- Wakeup interrupt function issues WAKEIRQ as system wakeup from power down mode or green mode. If WAKEIEN is "1" meaning enable, the wakeup event triggers program counter point to interrupt vector (ORG 8) executing interrupt service routine.

✸ ***Note: If wake-up source is external interrupt source, the WAKE bit won't be set, and external interrupt IRQ bit is set. The system issues external interrupt request and executes interrupt service routine.***

5.7.2 WAKEUP TIME

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

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

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

The Wakeup time = 1/Fosc * 2048 (sec) + high clock start-up time

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

The wakeup time = 1/Fosc * 2048 = 0.512 ms (Fosc = 4MHz)
The total wakeup time = 0.512 ms + oscillator start-up time

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

The Wakeup time = 1/Fosc * 32 (sec) + high clock start-up time

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

The wakeup time = 1/Fosc * 32 = 2 us (Fhosc = 16MHz)

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

5.7.3 P1W WAKEUP CONTROL REGISTER

Under power down mode (sleep mode) and green mode, the I/O ports with wakeup function are able to wake the system up to normal mode. The wake-up trigger edge is level changing. When wake-up pin occurs rising edge or falling edge, the system is waked up by the trigger edge. The Port 0 and Port 1 have wakeup function. Port 0 wakeup function always enables, but the Port 1 is controlled by the P1W register.

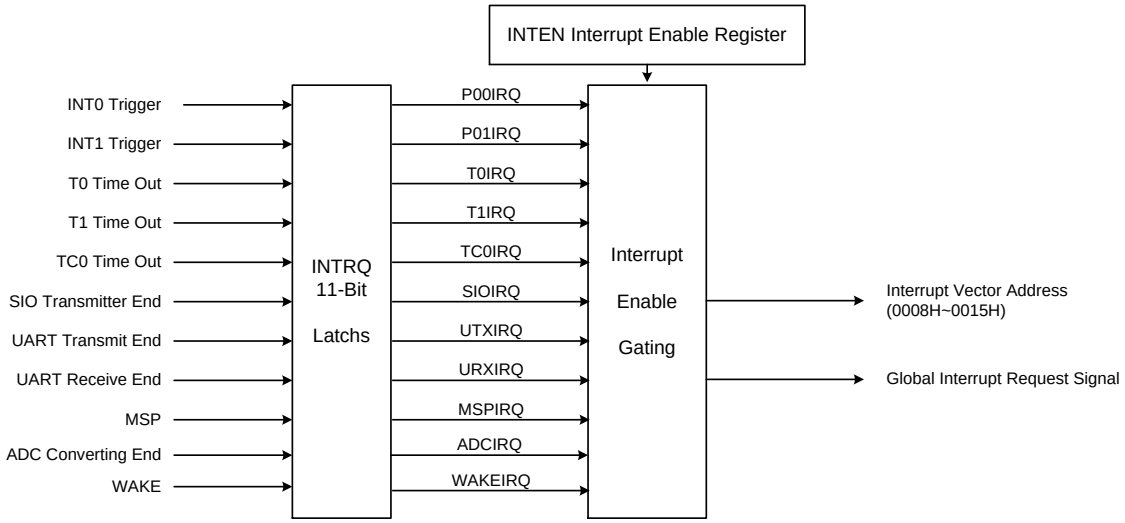
09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	-	-	-	-	P13W	P12W	P11W	P10W
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

Bit[3:0] **P10W~P13W:** Port 1 wakeup function control bits.
0 = Disable P1n wakeup function.
1 = Enable P1n wakeup function.

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 11 interrupt sources, including 2 external interrupt (INT0/INT1) and 9 internal interrupt (T0/T1/TC0/SIO/MSP/UTX/URX/WAKE/ADC). The external interrupt can wakeup the chip while the system is switched from power down mode to high-speed normal mode, and interrupt request is latched until return to 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. The interrupt request signals are stored in INTRQ register.



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

6.2 INTERRUPT OPERATION

Interrupt operation is controlled by IRQ and IEN bits. The IRQ is interrupt source event indicator, no matter what interrupt function status (enable or disable). The IEN control the system interrupt execution. If IEN = 0, the system won’t jump to interrupt vector to execute interrupt routine. If IEN = 1, the system executes interrupt operation when each of interrupt IRQ flags actives.

- **IEN = 1 and IRQ = 1, the program counter points to interrupt vector and execute interrupt service routine.**

When any interrupt requests occurs, the system provides to jump to interrupt vector and execute interrupt routine. The first procedure is “PUSH” operation. The end procedure after interrupt service routine execution is “POP” operation. The “PUSH” and “POP” operations aren’t through instruction (PUSH, POP) and executed by hardware automatically.

- **“PUSH” operation: PUSH operation saves the contents of ACC and working registers (0x80~0x8F) into hardware buffers. PUSH operation executes before program counter points to interrupt vector. The RAM bank keeps the status of main routine and doesn’t switch to bank 0 automatically. The RAM bank is selected by program.**
- **“POP” operation: POP operation reloads the contents of ACC and working registers (0x80~0x8F) from hardware buffers. POP operation executes as RETI instruction executed. The RAM bank switches to last status of main routine after reloading RBANK content.**
- **0x80~0x87 working registers include L, H, R, Z, Y, X, PFLAG, RBANK, W0~W7.**

6.3 INTEN INTERRUPT ENABLE REGISTER

INTEN is the interrupt request control register including eleven internal interrupts, two 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 8~15 to execute interrupt service routines. The program exits the interrupt service routine when the returning interrupt service routine instruction (RETI) is executed.

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN0	ADCIEN	T1IEN	-	-	TC0IEN	T0IEN	P01IEN	P00IEN
Read/Write	R/W	R/W	-	-	R/W	R/W	R/W	R/W
After reset	0	0	-	-	0	0	0	0

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

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

Bit 2 **T0IEN:** T0 timer interrupt control bit.
0 = Disable T0 interrupt function.
1 = Enable T0 interrupt function.

Bit 3 **TC0IEN:** TC0 timer interrupt control bit.
0 = Disable TC0 interrupt function.
1 = Enable TC0 interrupt function.

Bit 6 **T1IEN:** T1 timer interrupt control bit.
0 = Disable T1 interrupt function.
1 = Enable T1 interrupt function.

Bit 7 **ADCIEN:** ADC interrupt control bit.
0 = Disable ADC interrupt function.
1 = Enable ADC interrupt function.

09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN1	-	-	-	MSPIEN	UTXIEN	URXIEN	SIOIEN	WAKEIEN
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

Bit 0 **WAKEIEN:** Wakeup interrupt control bit.
0 = Disable wakeup interrupt function.
1 = Enable wakeup interrupt function.

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

Bit 2 **URXIEN:** UART receive interrupt control bit.
0 = Disable UART receive interrupt function.
1 = Enable UART receive interrupt function.

Bit 3 **UTXIEN:** UART transmit interrupt control bit.
0 = Disable UART transmit interrupt function.
1 = Enable UART transmit interrupt function.

Bit 4 **MSPIEN:** MSP interrupt control bit.
0 = Disable MSP interrupt function.
1 = Enable MSP interrupt function.

6.4 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.

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ0	ADCIRQ	T1IRQ	-	-	TC0IRQ	T0IRQ	P01IRQ	P00IRQ
Read/Write	R/W	R/W	-	-	R/W	R/W	R/W	R/W
After reset	0	0	-	-	0	0	0	0

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

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

Bit 2 **TC0IRQ:** T0 timer interrupt request flag.
0 = None T0 interrupt request.
1 = T0 interrupt request.

Bit 3 **TC0IRQ:** TC0 timer interrupt request flag.
0 = None TC0 interrupt request.
1 = TC0 interrupt request.

Bit 6 **T1IRQ:** T1 timer interrupt request flag.
0 = None T1 interrupt request.
1 = T1 interrupt request.

Bit 7 **ADCIRQ:** ADC interrupt request flag.
0 = None ADC interrupt request.
1 = ADC interrupt request.

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ1	-	-	-	MSPIRQ	UTXIRQ	URXIRQ	SIOIRQ	WAKEIRQ
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

Bit 0 **WAKEIRQ:** Wakeup interrupt request flag.
0 = None wakeup interrupt request.
1 = Wakeup interrupt request.

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

Bit 2 **URXIRQ:** UART receive interrupt request flag.
0 = None UART receive interrupt request.
1 = UART receive interrupt request.

Bit 3 **UTXIRQ:** UART transmit interrupt request flag.
0 = None UART transmit interrupt request.
1 = UART transmit interrupt request.

Bit 4 **MSPIRQ:** MSP interrupt request flag.
0 = None MSP interrupt request.
1 = MSP interrupt request.

6.5 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 8~15) 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	LVD23	LVD30	-	STKPB3	STKPB2	STKPB1	STKPB0
Read/Write	R/W	R	R	-	R/W	R/W	R/W	R/W
After reset	0	-	-	-	1	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.***

6.6 EXTERNAL INTERRUPT OPERATION (INT0~INT1)

Sonix provides 2 sets external interrupt sources in the micro-controller. INT0 and INT1 are external interrupt trigger sources and build in edge trigger configuration function. When the external edge trigger occurs, the external interrupt request flag will be set to “1” when the external interrupt control bit enabled. If the external interrupt control bit is disabled, the external interrupt request flag won’t active when external edge trigger occurrence. When external interrupt control bit is enabled and external interrupt edge trigger is occurring, the program counter will jump to the interrupt vector (ORG 0x0009, 0x000A) and execute interrupt service routine.

The external interrupt builds in wake-up latch function. That means when the system is triggered wake-up from power down mode, the wake-up source is external interrupt source (P0.0 or P0.1), and the trigger edge direction matches interrupt edge configuration, the trigger edge will be latched, and the system executes interrupt service routine fist after wake-up.

09FH	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]:** INT1 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

Bit[1:0] **P00G[1:0]:** INT0 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

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

```
        MOV     A, #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     9               ; 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         0BH         ; 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
```

✸ ***Note: In RTC mode, don’t reset T0C in interrupt service routine.***

6.8 TC0 INTERRUPT OPERATION

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

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

```
        B0BCLR      FTC0IEN         ; Disable TC0 interrupt service
        B0BCLR      FTC0ENB         ; Disable TC0 timer
        MOV         A, #10H         ;
        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         0CH             ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

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

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

        B0BCLR      FTC0IRQ         ; Reset TC0IRQ
        MOV         A, #74H
        B0MOV       TC0C, A         ; Reset TC0C.
        …                           ; TC0 interrupt service routine
        …
EXIT_INT:
        …                           ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

6.9 T1 INTERRUPT OPERATION

When the T1C (T1CH, T1CL) counter occurs overflow, the T1IRQ will be set to “1” however the T1IEN is enable or disable. If the T1IEN = 1, the trigger event will make the T1IRQ to be “1” and the system enter interrupt vector. If the T1IEN = 0, the trigger event will make the T1IRQ to be “1” but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

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

```
        B0BCLR      FT1IEN          ; Disable T1 interrupt service
        B0BCLR      FT1ENB          ; Disable T1 timer
        MOV         A, #20H         ;
        B0MOV       T1M, A          ; Set T1 clock = Fcpu / 32 and falling edge trigger.
        CLR         T1CH
        CLR         T1CL

        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         0FH             ; 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
        B0MOV       A, T1CH
        B0MOV       T1CHBUF, A
        B0MOV       A, T1CL
        B0MOV       T1CLBUF, A      ; Save pulse width.
        CLR         T1CH
        CLR         T1CL
        …                           ; T1 interrupt service routine
        …
EXIT_INT:
        …                           ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

6.10 ADC INTERRUPT OPERATION

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

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

```
        B0BCLR      FADCIEN         ; Disable ADC interrupt service

        MOV         A, #10110000B   ;
        B0MOV       ADM, A          ; Enable P4.0 ADC input and ADC function.
        MOV         A, #00000000B   ; Set ADC converting rate = Fcpu/16
        B0MOV       ADR, A

        B0BSET      FADCIEN         ; Enable ADC interrupt service
        B0BCLR      FADCIRQ         ; Clear ADC interrupt request flag
        B0BSET      FGIE            ; Enable GIE

        B0BSET      FADS            ; Start ADC transformation
```

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

```
        ORG         10H             ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

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

        B0BTS1      FADCIRQ         ; Check ADCIRQ
        JMP         EXIT_INT        ; ADCIRQ = 0, exit interrupt vector

        B0BCLR      FADCIRQ         ; Reset ADCIRQ
        …                           ; ADC interrupt service routine
        …
EXIT_INT:
        …                           ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

6.11 SIO INTERRUPT OPERATION

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

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

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

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

```
        ORG         11H         ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:
        …                       ; Push routine to save ACC and PFLAG to buffers.

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

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

        RETI                    ; Exit interrupt vector
```

6.12 UART INTERRUPT OPERATION

When the UART transmitter successfully, the URXIRQ/UTXIRQ will be set to “1” no matter the URXIEN/UTXIEN is enable or disable. If the URXIEN/UTXIEN and the trigger event URXIRQ/UTXIRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the URXIEN/UTXIEN = 0, the trigger event URXIRQ/UTXIRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the URXIEN/UTXIEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: UART receive and transmit interrupt request setup.**

```
        B0BSET      FURXIEN     ; Enable UART receive interrupt service
        B0BCLR      FURXIRQ     ; Clear UART receive interrupt request flag

        B0BSET      FUTXIEN     ; Enable UART transmit interrupt service
        B0BCLR      FUTXIRQ     ; Clear UART transmit interrupt request flag
        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: UART receive interrupt service routine.**

```
        ORG         13H         ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:
        …                       ; Push routine to save ACC and PFLAG to buffers.

        B0BTS1      FURXIRQ     ; Check RXIRQ
        JMP         EXIT_INT    ; RXIRQ = 0, exit interrupt vector

        B0BCLR      FURXIRQ     ; Reset RXIRQ
        …                       ; UART receive interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.13 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*
WAKEIRQ	Wake-up from power down or green mode
P00IRQ	P0.0 trigger controlled by PEDGE
P01IRQ	P0.1 trigger controlled by PEDGE
T0IRQ	T0C overflow
TC0IRQ	TC0C overflow
T1IRQ	T1CH, T1CL overflow
ADCIRQ	ADC converting end.
SIOIRQ	SIO transmitter successfully.
MSPIRQ	MSP transmitter successfully.
RXIRQ	UART transmit successfully.
TXIRQ	UART receive successfully.

For multi-interrupt conditions, two things need to be taking care of. One is that it is multi-vector and each of interrupts points to unique vector. Two is users have to define the interrupt vector. The following example shows the way to define the interrupt vector in the program memory.

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

```
        ORG     8               ; Interrupt vector
        JMP     ISR_WAKE
        JMP     ISR_INT0
        JMP     ISR_INT1
        JMP     ISR_T0
        JMP     ISR_TC0
        NOP
        NOP
        JMP     ISR_T1
        JMP     ISR_ADC
        JMP     ISR_SIO
        JMP     ISR_MSP
        JMP     ISR_UART_RX
        JMP     ISR_UART_TX
        NOP

ISR_WAKE:                       ; WAKE-UP interrupt service routine

        RETI                    ; Exit interrupt vector
ISR_INT0:                       ; INT0 interrupt service routine
                                ;
        RETI                    ; Exit interrupt vector
ISR_INT1:                       ; INT1 interrupt service routine

        RETI                    ; Exit interrupt vector
        …
        …

ISR_UART_TX:                    ; UART_TX interrupt service routine

        RETI                    ; Exit interrupt vector
```

7 I/O PORT

7.1 OVERVIEW

The micro-controller builds in 14 pin I/O. Most of the I/O pins are mixed with analog pins and special function pins. The I/O shared pin list is as following.

I/O Pin		Shared Pin		Shared Pin Control Condition
Name	Type	Name	Type	
P0.0	I/O	INT0	DC	P00IEN=1
		TC0	DC	TC0CKS=1, TC0ENB=1
P0.1	I/O	INT1	DC	P01IEN=1
P0.2	I/O	URX	DC	URXEN=1
		SCL	DC	MSPENB=1
		SDI	DC	SENB=1
P0.3	I/O	UTX	DC	UTXEN=1
		SDA	DC	MSPENB=1
		SDO	DC	SENB=1
P0.4	I/O	SCK	DC	SENB=1
P0.5	I/O	PWM0	DC	TC0ENB=1, PWM0OUT=1
P0.6	I/O	XIN	AC	High_CLK code option = IHRC_RTC, RC, 32K, 4M, 12M
P0.7	I/O	XOUT	AC	High_CLK code option = IHRC_RTC, 32K, 4M, 12M
P1.0	I/O	EICK	DC	Embedded ICE mode.
P1.1	I/O	EIDA	DC	Embedded ICE mode.
P1.2	I/O	-	-	
P1.3	I/O	RST	DC	Reset_Pin code option = Reset
P4.0	I/O	AIN0	AC	ADENB=1,GCHS=1,CHS[1:0]=00b
P4.1	I/O	AIN1	AC	ADENB=1,GCHS=1,CHS[1:0]=01b

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

7.2 I/O PORT MODE

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

0A0H	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

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	-	-	-	-	P13M	P12M	P11M	P10M
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	-	-	-	-	-	-	P41M	P40M
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

Bit [7:0] **PnM[7:0]:** Pn mode control bits. (n = 0~4).
0 = Pn is input mode.
1 = Pn is output mode.

✸ ***Note: Users can program them by bit control instructions (B0BSET, B0BCLR).***

➢ **Example: I/O mode selecting**

```
CLR     P0M         ; Set all ports to be input mode.
CLR     P1M
CLR     P4M

MOV     A, #0FFH    ; Set all ports to be output mode.
B0MOV   P0M, A
B0MOV   P1M, A
B0MOV   P4M,A

B0BCLR  P4M.0       ; Set P4.0 to be input mode.

B0BSET  P4M.0       ; Set P4.0 to be output mode.
```

7.3 I/O PULL UP REGISTER

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

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

0ADH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	-	-	-	-	P13R	P12R	P11R	P10R
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

0B0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	-	-	-	-	-	-	P41R	P40R
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

➢ **Example: I/O Pull up Register**

```
MOV     A, #0FFH        ; Enable Port0, 1, 4 Pull-up register,
B0MOV   P0UR, A         ;
B0MOV   P1UR,A
B0MOV   P4UR, A
```

7.4 I/O PORT DATA REGISTER

0A6H	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	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	-	-	-	-	P13	P12	P11	P10
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

0AAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	-	-	-	-	-	-	P41	P40
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

✸ ***Note: The P13 keeps "1" when external reset enable by code option.***

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

```
B0MOV   A, P0       ; Read data from Port 0
B0MOV   A, P1       ; Read data from Port 1
B0MOV   A, P4       ; Read data from Port 4
```

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

```
MOV     A, #0FFH    ; Write data FFH to all Port.
B0MOV   P0, A
B0MOV   P1, A
B0MOV   P4, A
```

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

```
B0BSET  P0.3        ; Set P0.3 and P4.0 to be "1".
B0BSET  P4.0

B0BCLR  P0.3        ; Set P0.3 and P4.0 to be "0".
B0BCLR  P4.0
```

7.5 PORT 4 ADC SHARE PIN

The Port 4 is shared with ADC input function and no Schmitt trigger structure. 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.n will configure related port 4 pin as pure analog input pin to avoid current leakage.

0C6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	-	-	-	-	-	-	P4CON1	P4CON0
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

Bit [1:0] **P4CON [1:0]:** 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 are 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.***

Port 4 ADC analog input is controlled by GCHS and CHSn bits of ADM register. If GCHS = 0, P4.n are general purpose bi-direction I/O port. If GCHS = 1, P4.n pointed by CHSn is ADC analog signal input pin.

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	-	-	CHS1	CHS0
Read/Write	R/W	R/W	R/W	R/W	-	-	R/W	R/W
After reset	0	0	0	0	-	-	0	0

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

Bit [1:0] **CHS [1:0]:** ADC input channels select bit.
00 = AIN0, 01 = AIN1, 10 = VSS, 11 = VDD.

✷ ***Note: For P4.n general purpose I/O function, users should make sure of P4.n’s ADC channel are disabled, or P4.n are automatically set as ADC analog input when GCHS = 1 and CHS[1:0] point to P4.n.***

➢ **Example: Set P4.1 to be general purpose input mode. P4CON.1 must be set as "0".**

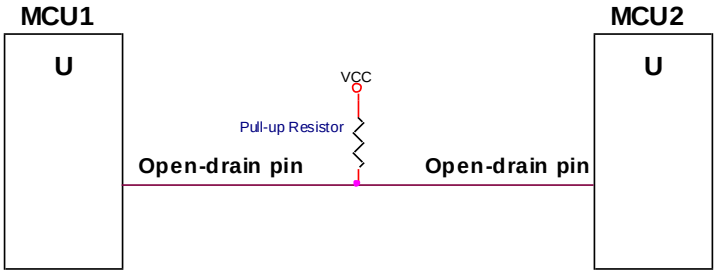
```
; Check GCHS and CHS [1:0] status.
        B0BCLR      FGCHS       ;If CHS[1:0] point to P4.1 (CHS[1:0] = 01B), set GCHS=0
                                ;If CHS[1:0] don't point to P4.1 (CHS[1:0] ≠ 01B), don't
                                care GCHS status.

; Clear P4CON.
        B0BCLR      P4CON.1     ; Enable P4.1 digital function.

; Enable P4.1 input mode.
        B0BCLR      P4M.1       ; Set P4.1 as input mode.
```

➢ **Example: Set P4.1 to be general purpose output. P4CON.1 must be set as "0".**

```
; Check GCHS and CHS [1:0] status.
        B0BCLR      FGCHS       ;If CHS [1:0] point to P4.1 (CHS [1:0] = 01B), set GCHS=0.
                                ;If CHS [1:0] don't point to P4.1 (CHS [1:0] ≠ 01B), don't
                                care GCHS status.

; Clear P4CON.
        B0BCLR      P4CON.1     ; Enable P4.1 digital function.

; Set P4.1 output buffer to avoid glitch.
        B0BSET      P4.1        ; Set P4.1 buffer as "1".
; or
        B0BCLR      P4.1        ; Set P4.1 buffer as "0".

; Enable P4.1 output mode.
        B0BSET      P4M.1       ; Set P4.1 as output mode.
```

7.6 OPEN-DRAIN REGISTER

P0.2, P0.3, P0.4 built in open-drain function. These pins must be set as output mode when enable 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.

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0OC	-	-	-	-	-	P04OC	P03OC	P02OC
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

Bit [2:0] **P02OC, P03OC, P04OC:** P0.2, P0.3, P0.4 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

➢ **Example: Enable P0.2 to open-drain mode and output high.**

```
B0BSET    P0.2     ; Set P0.2 buffer high.

B0BSET    P02M     ; Enable P0.2 output mode.
B0BSET    P02OC    ; Enable P0.2 open-drain function.
```

➢ **Example: Disable open-drain mode.**

```
B0BCLR    P02OC    ; Disable P0.2 open-drain function.
```

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

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, watchdog timer overflow signal raises and resets MCU. Watchdog timer clock source is internal low-speed oscillator 16KHz RC type and through programmable pre-scaler controlled by WDT_CLK code option.

Watchdog timer interval time = 256 * 1/ (Internal Low-Speed oscillator frequency/WDT Pre-scalar) …sec
= 256 / (16KHz/WDT Pre-scaler) …sec

Internal low-speed oscillator	WDT pre-scaler	Watchdog interval time
Flosc=16KHz	Flosc/4	256/(16000/4)=64ms
	Flosc/8	256/(16000/8)=128ms
	Flosc/16	256/(16000/16)=256ms
	Flosc/32	256/(16000/32)=512ms

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

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

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

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

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

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

➢ **Example: Clear watchdog timer by "@RST_WDT" macro of Sonix IDE.**

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

Watchdog timer application note is as following.

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

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

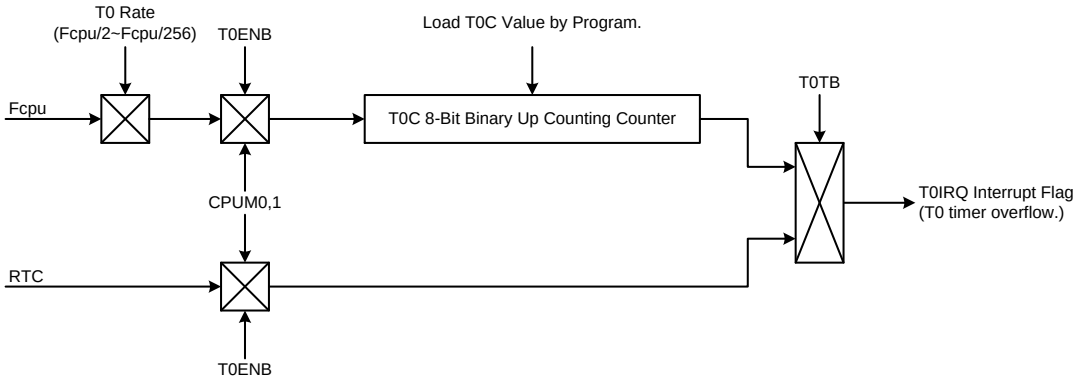
```
Main:
        …                       ; Check I/O.
        …                       ; Check RAM
Err:    JMP $                   ; I/O or RAM error. Program jump here and don't
                                ; clear watchdog. Wait watchdog timer overflow to reset IC.

Correct:                        ; 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 T0 8-BIT BASIC TIMER

8.2.1 OVERVIEW

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

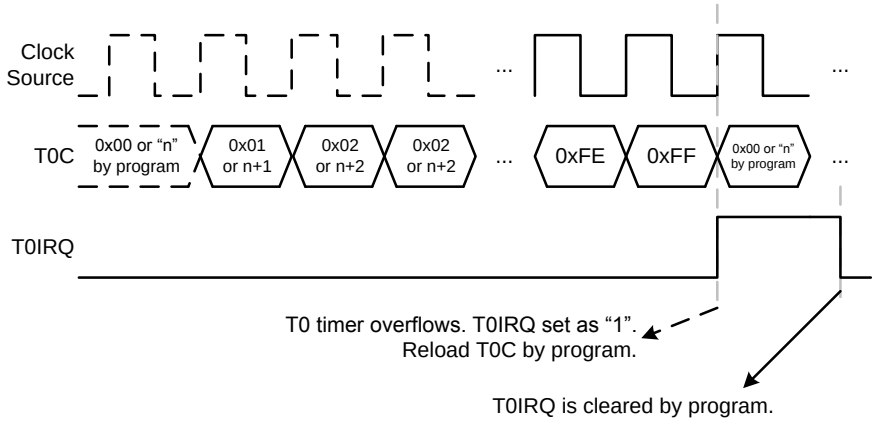
- **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** T0 timer function supports interrupt function. When T0 timer occurs overflow, the T0IRQ actives and the system points program counter to interrupt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



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

8.2.2 T0 Timer Operation

T0 timer is controlled by T0ENB bit. When T0ENB=0, T0 timer stops. When T0ENB=1, T0 timer starts to count. T0C increases “1” by timer clock source. When T0 overflow event occurs, T0IRQ flag is set as ”1” to indicate overflow and cleared by program. The overflow condition is T0C count from full scale (0xFF) to zero scale (0x00). T0 doesn’t build in double buffer, so load T0C by program when T0 timer overflows to fix the correct interval time. If T0 timer interrupt function is enabled (T0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 000BH) and executes interrupt service routine after T0 overflow occurrence. Clear T0IRQ by program is necessary in interrupt procedure. T0 timer can works in normal mode, slow mode and green mode. In green mode, T0 keeps counting, set T0IRQ and wakes up system when T0 timer overflows.



T0 clock source is Fcpu (instruction cycle) through T0rate[2:0] pre-scalar to decide Fcpu/2~Fcpu/256. T0 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

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

8.2.3 T0M MODE REGISTER

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

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB
Read/Write	R/W	R/W	R/W	R/W	-	-	-	R/W
After reset	0	0	0	0	-	-	-	0

Bit 0 **T0TB:** RTC clock source control bit.
0 = Disable RTC (T0 clock source from Fcpu).
1 = Enable RTC.

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

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

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

8.2.4 T0C COUNTING REGISTER

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

0B3H	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 * T0 clock rate)

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

*T0C initial value = 256 - (T0 interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 128)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 128)*
= B2H

✸ ***Note: In RTC mode, T0C is 256 counts and generatesT0 0.5 sec interval time. Don't change T0C value in RTC mode.***

8.2.5 T0 TIMER OPERATION EXPLAME

- **T0 TIMER CONFIGURATION:**

```
; Reset T0 timer.
        MOV     A, #0x00        ; Clear T0M register.
        B0MOV   T0M, A

; Set T0 clock source and T0 rate.
        MOV     A, #0nnn0000b
        B0MOV   T0M, A

; Set T0C register for T0 Interval time.
        MOV     A, #value
        B0MOV   T0C, A

; Clear T0IRQ.
        B0BCLR  FT0IRQ

; Enable T0 timer and interrupt function.
        B0BSET  FT0IEN          ; Enable T0 interrupt function.
        B0BSET  FT0ENB          ; Enable T0 timer.
```

- **T0 works in RTC mode:**

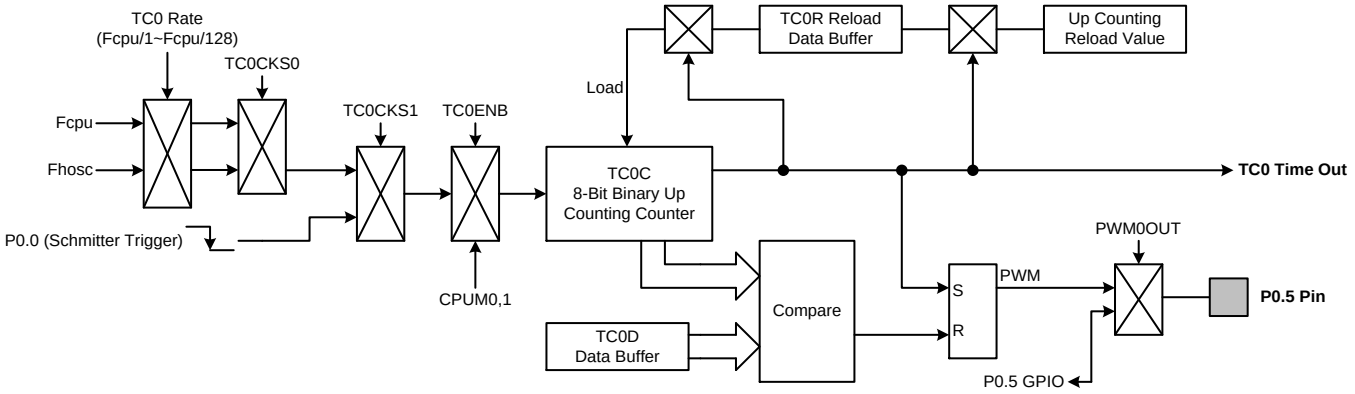
```
; Reset T0 timer.
        MOV     A, #0x00        ; Clear T0M register.
        B0MOV   T0M, A

; Set T0 RTC function.
        B0BSET  FT0TB

; Clear T0C.
        CLR     T0C

; Clear T0IRQ.
        B0BCLR  FT0IRQ

; Enable T0 timer and interrupt function.
        B0BSET  FT0IEN          ; Enable T0 interrupt function.
        B0BSET  FT0ENB          ; Enable T0 timer.
```

8.3 TC0 8-BIT TIMER/COUNTER

8.3.1 OVERVIEW

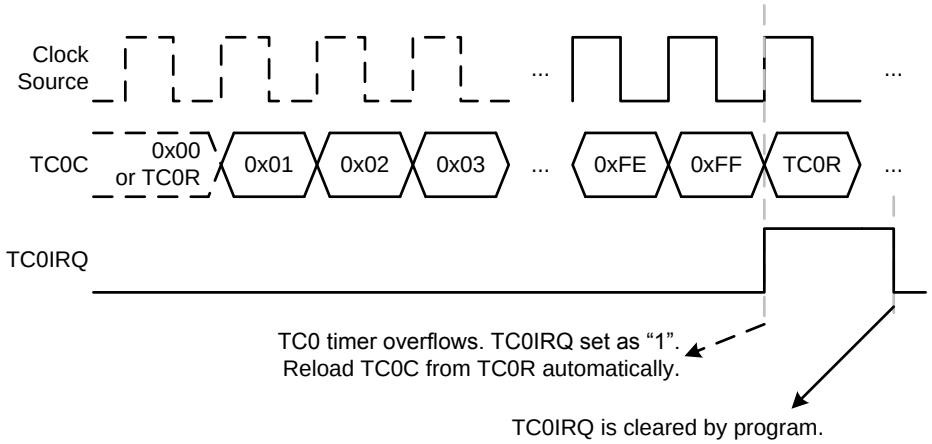
The TC0 timer is an 8-bit binary up timer with basic timer, event counter and PWM functions. The basic timer function supports flag indicator (TC0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TC0M, TC0C, TC0R registers. The event counter is changing TC0 clock source from system clock (Fcpu/Fhosc) to external clock like signal (e.g. continuous pulse, R/C type oscillating signal…). TC0 becomes a counter to count external clock number to implement measure application. TC0 also builds in duty/cycle programmable PWM. The PWM cycle and resolution are controlled by TC0 timer clock rate, TC0R and TC0D registers, so the PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster…The main purposes of the TC0 timer are as following.

- ☞ **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- ☞ **Interrupt function:** TC0 timer function supports interrupt function. When TC0 timer occurs overflow, the TC0IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- ☞ **Event Counter:** The event counter function counts the external clock counts.
- ☞ **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by TC0R and TC0D registers.
- ☞ **Green mode function:** All TC0 functions (timer, PWM, event counter, auto-reload) keep running in green mode and no wake-up function.



8.3.2 TC0 TIMER OPERATION

TC0 timer is controlled by TC0ENB bit. When TC0ENB=0, TC0 timer stops. When TC0ENB=1, TC0 timer starts to count. Before enabling TC0 timer, setup TC0 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…TC0C increases "1" by timer clock source. When TC0 overflow event occurs, TC0IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is TC0C count from full scale (0xFF) to zero scale (0x00). In difference function modes, TC0C value relates to operation. If TC0C value changing effects operation, the transition of operations would make timer function error. So TC0 builds in double buffer to avoid these situations happen. The double buffer concept is to flash TC0C during TC0 counting, to set the new value to TC0R (reload buffer), and the new value will be loaded from TC0R to TC0C after TC0 overflow occurrence automatically. In the next cycle, the TC0 timer runs under new conditions, and no any transitions occur. The auto-reload function is no any control interface and always actives as TC0 enables. If TC0 timer interrupt function is enabled (TC0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 000CH) and executes interrupt service routine after TC0 overflow occurrence. Clear TC0IRQ by program is necessary in interrupt procedure. TC0 timer can works in normal mode, slow mode and green mode. But in green mode, TC0 keep counting, set TC0IRQ and outputs PWM, but can't wake-up system.



TC0 provides different clock sources to implement different applications and configurations. TC0 clock source includes Fcpu (instruction cycle), Fhosc (high speed oscillator) and external input pin (P0.0) controlled by TC0CKS[1:0] bits. TC0CKS0 bit selects the clock source is from Fcpu or Fhosc. If TC0CKS0=0, TC0 clock source is Fcpu through TC0rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. If TC0CKS0=1, TC0 clock source is Fhosc through TC0rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. TC0CKS1 bit controls the clock source is external input pin or controlled by TC0CKS0 bit. If TC0CKS1=0, TC0 clock source is selected by TC0CKS0 bit. If TC0CKS1=1, TC0 clock source is external input pin that means to enable event counter function. TC0rate[2:0] pre-scalar is unless when TC0CKS0=1 or TC0CKS1=1 conditions. TC0 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

TC0CKS0	TC0rate[2:0]	TC0 Clock	TC0 Interval Time			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	8.192	32	32.768	128
0	001b	Fcpu/64	4.096	16	16.384	64
0	010b	Fcpu/32	2.048	8	8.192	32
0	011b	Fcpu/16	1.024	4	4.096	16
0	100b	Fcpu/8	0.512	2	2.048	8
0	101b	Fcpu/4	0.256	1	1.024	4
0	110b	Fcpu/2	0.128	0.5	0.512	2
0	111b	Fcpu/1	0.064	0.25	0.256	1
1	000b	Fhosc/128	2.048	8	8.192	32
1	001b	Fhosc/64	1.024	4	4.096	16
1	010b	Fhosc/32	0.512	2	2.048	8
1	011b	Fhosc/16	0.256	1	1.024	4
1	100b	Fhosc/8	0.128	0.5	0.512	2
1	101b	Fhosc/4	0.064	0.25	0.256	1
1	110b	Fhosc/2	0.032	0.125	0.128	0.5
1	111b	Fhosc/1	0.016	0.0625	0.064	0.25

8.3.3 TC0M MODE REGISTER

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

0B4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS1	TC0CKS0	-	PWM0OUT
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	-	R/W
After reset	0	0	0	0	0	0	-	0

Bit 0 **PWM0OUT:** PWM output control bit.
0 = Disable PWM output function, and P0.5 is GPIO mode.
1 = Enable PWM output function, and P0.5 outputs PWM signal.

Bit 2 **TC0CKS0:** TC0 clock source select bit.
0 = Fcpu.
1 = Fhosc.

Bit 3 **TC0CKS1:** TC0 clock source select bit.
0 = Internal clock (Fcpu and Fhosc controlled by TC0CKS0 bit).
1 = External input pin (P0.0/INT0) and enable event counter function. **TC0rate[2:0] bits are useless.**

Bit [6:4] **TC0RATE[2:0]:** TC0 timer clock source select bits.
TC0CKS0=0 -> 000 = Fcpu/128, 001 = Fcpu/64, 010 = Fcpu/32, 011 = Fcpu/16, 100 = Fcpu/8, 101 = Fcpu/4, 110 = Fcpu/2,111 = Fcpu/1.
TC0CKS0=1 -> 000 = Fhosc/128, 001 = Fhosc/64, 010 = Fhosc/32, 011 = Fhosc/16, 100 = Fhosc/8, 101 = Fhosc/4, 110 = Fhosc/2,111 = Fhosc/1.

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

8.3.4 TC0C COUNTING REGISTER

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

0B5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0C	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of TC0C initial value is as following.

TC0C initial value = 256 - (TC0 interrupt interval time * TC0 clock rate)

8.3.5 TC0R AUTO-RELOAD REGISTER

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

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of TC0R initial value is as following.

TC0R initial value = 256 - (TC0 interrupt interval time * TC0 clock rate)

- **Example: To calculation TC0C and TC0R value to obtain 10ms TC0 interval time. TC0 clock source is Fcpu = 16MHz/16 = 1MHz. Select TC0RATE=000 (Fcpu/128).**
  TC0 interval time = 10ms. TC0 clock rate = 16MHz/16/128

  *TC0C/TC0R initial value = 256 - (TC0 interval time * input clock)*
  *= 256 - (10ms * 16MHz / 16 / 128)*
  *= 256 - (10^{-2} * 16 * 10^{6} / 16 / 128)*
  = B2H

8.3.6 TC0D PWM DUTY REGISTER

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

0B7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0D	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0
Read/Write	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 TC0D initial value is as following.

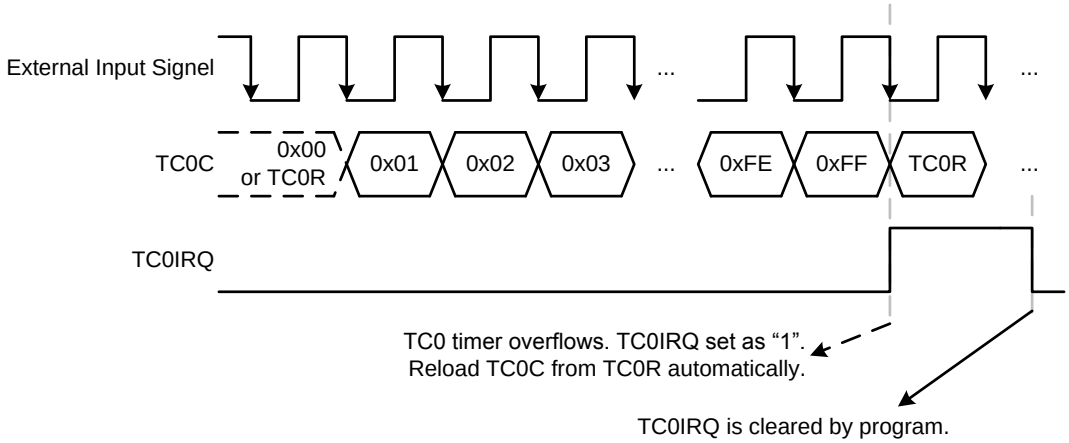
TC0D initial value = TC0R + (PWM high pulse width period / TC0 clock rate)

- **Example: To calculate TC0D value to obtain 1/3 duty PWM signal. The TC0 clock source is Fcpu = 16MHz/16= 1MHz. Select TC0RATE=000 (Fcpu/128).**
  TC0R = B2H. TC0 interval time = 10ms. So the PWM cycle is 100Hz. In 1/3 duty condition, the high pulse width is about 3.33ms.

  TC0D initial value = B2H + (PWM high pulse width period / TC0 clock rate)
  *= B2H + (3.33ms * 16MHz / 16 / 128)*
  = B2H + 1AH
  = CCH

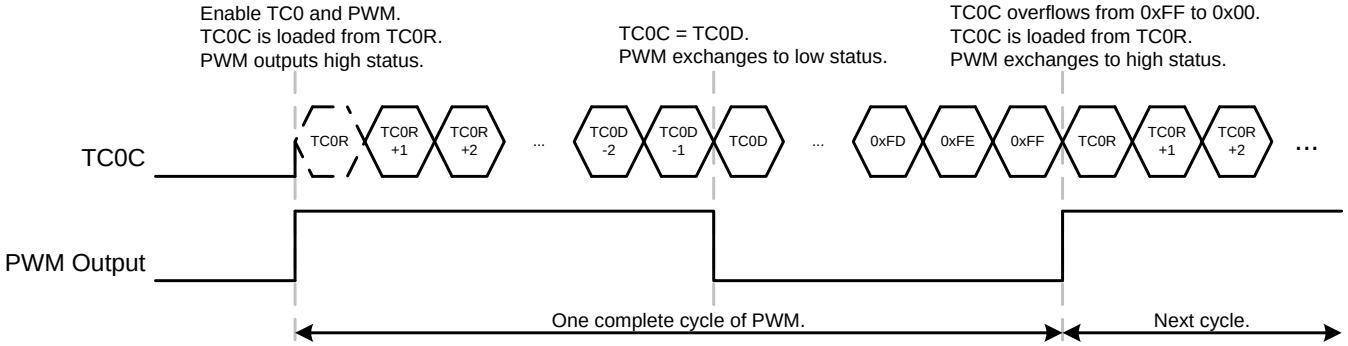
8.3.7 TC0 EVENT COUNTER

TC0 event counter is set the TC0 clock source from external input pin (P0.0). When TC0CKS1=1, TC0 clock source is switch to external input pin (P0.0). TC0 event counter trigger direction is falling edge. When one falling edge occurs, TC0C will up one count. When TC0C counts from 0xFF to 0x00, TC0 triggers overflow event. The external event counter input pin's wake-up function of GPIO mode is disabled when TC0 event counter function enabled to avoid event counter signal trigger system wake-up and not keep in power saving mode. The external event counter input pin's external interrupt function is also disabled when TC0 event counter function enabled, and the P00IRQ bit keeps "0" status. The event counter usually is used to measure external continuous signal rate, e.g. continuous pulse, R/C type oscillating signal…These signal phase don't synchronize with MCU's main clock. Use TC0 event to measure it and calculate the signal rate in program for different applications.



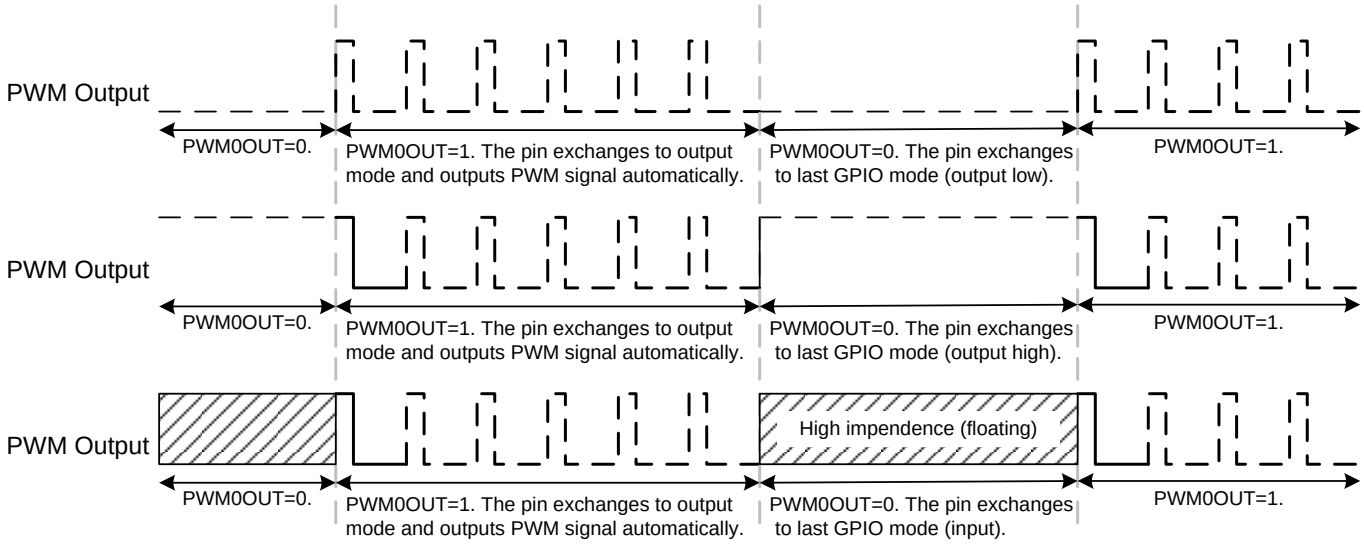
8.3.8 PULSE WIDTH MODULATION (PWM)

The PWM is duty/cycle programmable design to offer various PWM signals. When TC0 timer enables and PWM0OUT bit sets as "1" (enable PWM output), the PWM output pin (P0.5) outputs PWM signal. One cycle of PWM signal is high pulse first, and then low pulse outputs. TC0R register controls the cycle of PWM, and TC0D decides the duty (high pulse width length) of PWM. TC0C initial value is TC0R reloaded when TC0 timer enables and TC0 timer overflows. When TC0C count is equal to TC0D, the PWM high pulse finishes and exchanges to low level. When TC0 overflows (TC0C counts from 0xFF to 0x00), one complete PWM cycle finishes. The PWM exchanges to high level for next cycle. The PWM is auto-reload design to load TC0C from TC0R automatically when TC0 overflows and the end of PWM's cycle, to keeps PWM continuity. If modify the PWM cycle by program as PWM outputting, the new cycle occurs at next cycle when TC0C loaded from TC0R.



The resolution of PWM is decided by TC0R. TC0R range is from 0x00~0xFF. If TC0R = 0x00, PWM's resolution is 1/256. If TC0R = 0x80, PWM's resolution is 1/128. TC0D controls the high pulse width of PWM for PWM's duty. When TC0C = TC0D, PWM output exchanges to low status. TC0D must be greater than TC0R, or the PWM signal keeps low status. When PWM outputs, TC0IRQ still actives as TC0 overflows, and TC0 interrupt function actives as TC0IEN = 1. But strongly recommend be careful to use PWM and TC0 timer together, and make sure both functions work well.

The PWM output pin is shared with GPIO and switch to output PWM signal as PWM0OUT=1 automatically. If PWM0OUT bit is cleared to disable PWM, the output pin exchanges to last GPIO mode automatically. It easily to implement carry signal on/off operation, not to control TC0ENB bit.



8.3.9 TC0 TIMER OPERATION EXAMPLE

- **TC0 TIMER CONFIGURATION:**

```
; Reset TC0 timer.
        CLR     TC0M            ; Clear TC0M register.

; Set TC0 clock source and TC0 rate.
        MOV     A, #0nnn0n00b
        B0MOV   TC0M, A

; Set TC0C and TC0R register for TC0 Interval time.
        MOV     A, #value       ; TC0C must be equal to TC0R.
        B0MOV   TC0C, A
        B0MOV   TC0R, A

; Clear TC0IRQ
        B0BCLR  FTC0IRQ

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

- **TC0 EVENT COUNTER CONFIGURATION:**

```
; Reset TC0 timer.
        CLR     TC0M            ; Clear TC0M register.

; Enable TC0 event counter.
        B0BSET  FTC0CKS1        ; Set TC0 clock source from external input pin (P0.0).

; Set TC0C and TC0R register for TC0 Interval time.
        MOV     A, #value       ; TC0C must be equal to TC0R.
        B0MOV   TC0C, A
        B0MOV   TC0R, A

; Clear TC0IRQ
        B0BCLR  FTC0IRQ

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

- **TC0 PWM CONFIGURATION:**

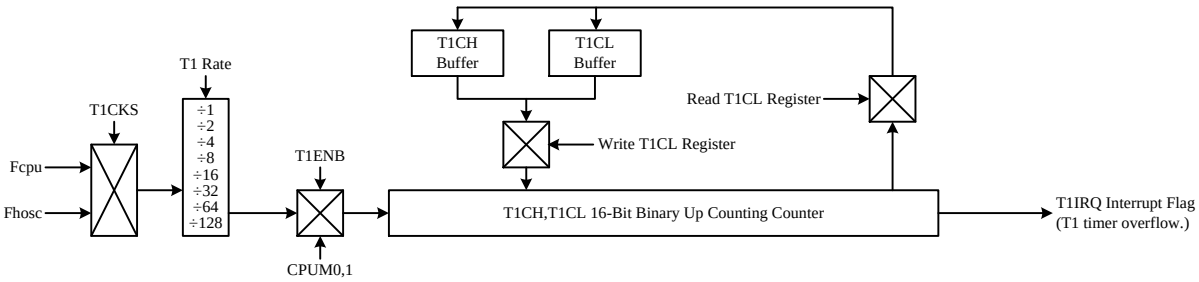
```
; Reset TC0 timer.
        CLR     TC0M            ; Clear TC0M register.

; Set TC0 clock source and TC0 rate.
        MOV     A, #0nnn0n00b
        B0MOV   TC0M, A

; Set TC0C and TC0R register for PWM cycle.
        MOV     A, #value1      ; TC0C must be equal to TC0R.
        B0MOV   TC0C, A
        B0MOV   TC0R, A

; Set TC0D register for PWM duty.
        MOV     A, #value2      ; TC0D must be greater than TC0R.
        B0MOV   TC0D, A

; Enable PWM and TC0 timer.
        B0BSET  FTC0ENB         ; Enable TC0 timer.
        B0BSET  FPWM0OUT        ; Enable PWM.
```

8.4 T1 16-BIT TIMER

8.4.1 OVERVIEW

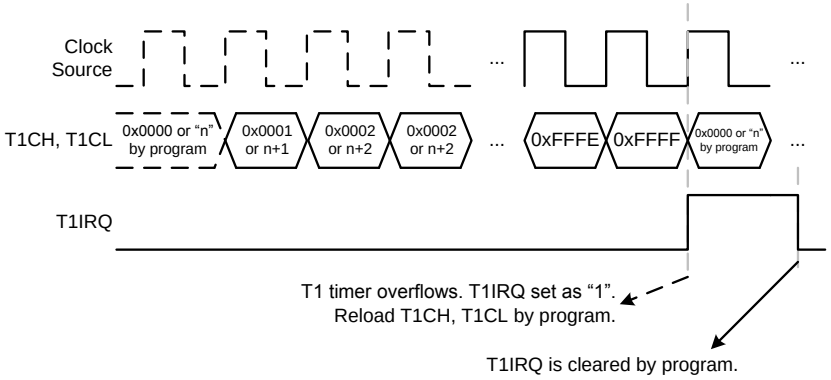
The T1 timer is a 16-bit binary up timer with basic timer. The basic timer function supports flag indicator (T1IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T1M, T1CH/T1CL 16-bit counter registers. The main purposes of the T1 timer are as following.

- ☞ **16-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- ☞ **Interrupt function:** T1 timer function support interrupt function. When T1 timer occurs overflow, the T1IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- ☞ **Green mode function:** All T1 functions (timer…) keeps running in green mode, but no wake-up function. Timer IRQ actives as any IRQ trigger occurrence, e.g. timer overflow…



8.4.2 T1 TIMER OPERATION

T1 timer is controlled by T1ENB bit. When T1ENB=0, T1 timer stops. When T1ENB=1, T1 timer starts to count. Before enabling T1 timer, setup T1 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…T1 16-bit counter (T1CH, T1CL) increases "1" by timer clock source. When T1 overflow event occurs, T1IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is T1CH, T1CL count from full scale (0xFFFF) to zero scale (0x0000). T1 doesn't build in double buffer, so load T1CH, T1CL by program when T1 timer overflows to fix the correct interval time. If T1 timer interrupt function is enabled (T1IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 000FH) and executes interrupt service routine after T1 overflow occurrence. Clear T1IRQ by program is necessary in interrupt procedure. T1 timer can works in normal mode, slow mode and green mode.



T1 provides different clock sources to implement different applications and configurations. T1 clock source includes Fcpu (instruction cycle) and Fhosc (high speed oscillator) controlled by T1CKS bit. T1CKS bit selects the clock source is from Fcpu or Fhosc. If T1CKS=0, T1 clock source is Fcpu through T1rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. If T1CKS=1, T1 clock source is Fhosc through T1rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. T1 length is 16-bit (65536 steps), and the one count period is each cycle of input clock.

T1CKS	T1rate[2:0]	T1 Clock	T1 Interval Time			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	2097.152	32	8388.608	128
0	001b	Fcpu/64	1048.576	16	4194.304	64
0	010b	Fcpu/32	524.288	8	2097.152	32
0	011b	Fcpu/16	262.144	4	1048.576	16
0	100b	Fcpu/8	131.072	2	524.288	8
0	101b	Fcpu/4	65.536	1	262.144	4
0	110b	Fcpu/2	32.768	0.5	131.072	2
0	111b	Fcpu/1	16.384	0.25	65.536	1
1	000b	Fhosc/128	524.288	8	2097.152	32
1	001b	Fhosc/64	262.144	4	1048.576	16
1	010b	Fhosc/32	131.072	2	524.288	8
1	011b	Fhosc/16	65.536	1	262.144	4
1	100b	Fhosc/8	32.768	0.5	131.072	2
1	101b	Fhosc/4	16.384	0.25	65.536	1
1	110b	Fhosc/2	8.192	0.125	32.768	0.5
1	111b	Fhosc/1	4.096	0.0625	16.384	0.25

8.4.3 T1M MODE REGISTER

T1M is T1 timer mode control register to configure T1 operating mode including T1 pre-scalar, clock source…These configurations must be setup completely before enabling T1 timer.

0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1rate2	T1rate1	T1rate0	T1CKS	-	-	-
Read/Write	R/W	R/W	R/W	R/W	R/W	-	-	-
After reset	0	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 timer clock source select bits.
T1CKS=0 -> 000 = Fcpu/128, 001 = Fcpu/64, 010 = Fcpu/32, 011 = Fcpu/16, 100 = Fcpu/8, 101 = Fcpu/4, 110 = Fcpu/2,111 = Fcpu/1.
T1CKS=1 -> 000 = Fhosc/128, 001 = Fhosc/64, 010 = Fhosc/32, 011 = Fhosc/16, 100 = Fhosc/8, 101 = Fhosc/4, 110 = Fhosc/2,111 = Fhosc/1.

Bit 3 **T1CKS:** T1 clock source control bit.
0 = Fcpu.
1 = Fhosc.

8.4.4 T1CH, T1CL 16-bit COUNTING REGISTERS

T1 counter is 16-bit counter combined with T1CH and T1CL registers. When T1 timer overflow occurs, the T1IRQ flag is set as “1” and cleared by program. The T1CH, T1CL decide T1 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T1CH and T1CL registers, and then enable T1 timer to make sure the fist cycle correct. After one T1 overflow occurs, the T1CH and T1CL registers are loaded correct values by program.

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CL	T1CL7	T1CL6	T1CL5	T1CL4	T1CL3	T1CL2	T1CL1	T1CL0
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
T1CH	T1CH7	T1CH6	T1CH5	T1CH4	T1CH3	T1CH2	T1CH1	T1CH0
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 T1 timer counter length is 16-bit and points to T1CH and T1CL registers. The timer counter is double buffer design. The core bus is 8-bit, so access 16-bit data needs a latch flag to avoid the transient status affect the 16-bit data mistake occurrence. Under write mode, the write T1CH is the latch control flag. Under read mode, the read T1CL is the latch control flag. So, write T1 16-bit counter is to write T1CH first, and then write T1CL. The 16-bit data is written to 16-bit counter buffer after executing writing T1CL. Read T1 16-bit counter is to read T1CL first, and then read T1CH. The 16-bit data is dumped to T1CH, T1CL after executing reading T1CH.

- **Read T1 counter buffer sequence is to read T1CL first, and then read T1CH.**
- **Write T1 counter buffer sequence is to write T1CH first, and then write T1CL.**

The equation of T1 16-bit counter (T1CH, T1CL) initial value is as following.

T1CH, T1CL initial value = 65536 - (T1 interrupt interval time * T1 clock rate)

- **Example: To calculation T1CH and T1CL values to obtain 500ms T1 interval time. T1 clock source is Fcpu = 16MHz/16 = 1MHz. Select T1RATE=000 (Fcpu/128).**
  T1 interval time = 500ms. T1 clock rate = 16MHz/16/128

*T1 16-bit counter initial value = 65536 - (T1 interval time * input clock)*
*= 65536 - (500ms * 16MHz / 16 / 128)*
*= 65536 - (500*10^{-3} * 16 * 10^{6} / 16 / 128)*
= F0BDH ***(T1CH = F0H, T1CL = BDH)***

8.4.5 T1 TIMER OPERATION EXAMPLE

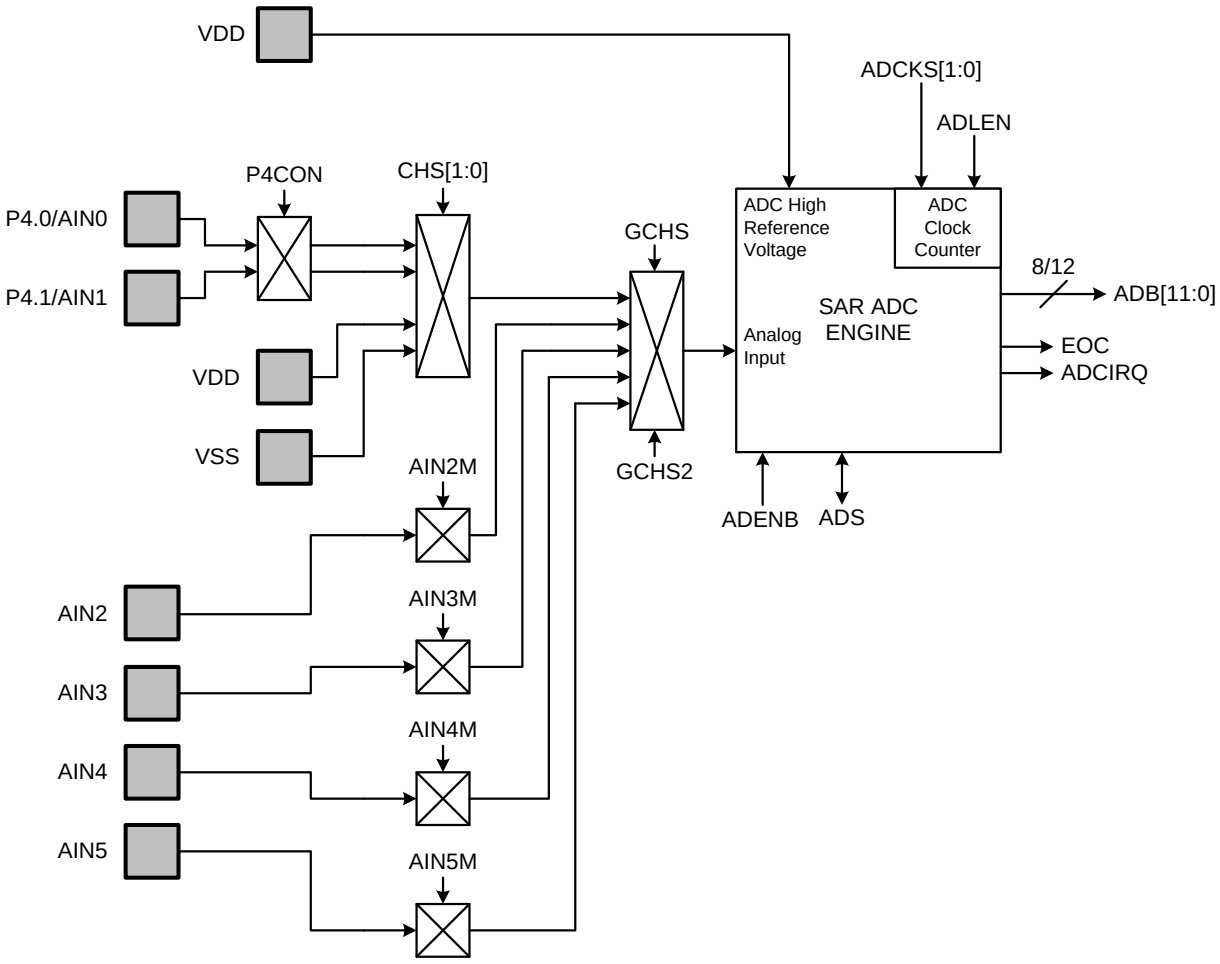
- **T1 TIMER CONFIGURATION:**

```
; Reset T1 timer.
        MOV     A, #0x00        ; Clear T1M register.
        B0MOV   T1M, A

; Set T1 clock rate.
        MOV     A, #0nnn0000b   ; T1rate[2:0] bits.
        B0MOV   T1M, A

; Set T1CH, T1CL registers for T1 Interval time.
        MOV     A, #value1      ; Set high byte first.
        B0MOV   T1CH, A
        MOV     A, #value2      ; Set low byte.
        B0MOV   T1CL, A

; Clear T1IRQ
        B0BCLR  FT1IRQ

; Enable T1 timer and interrupt function.
        B0BSET  FT1IEN          ; Enable T1 interrupt function.
        B0BSET  FT1ENB          ; Enable T1 timer.
```

9 6 CHANNEL ANALOG TO DIGITAL CONVERTER (ADC)

9.1 OVERVIEW

The analog to digital converter (ADC) is SAR structure with 6-input sources and up to 4096-step resolution to transfer analog signal into 12-bits digital buffers. The ADC builds in 6-channel input source (AIN0~AIN5) to measure 6 different analog signal sources controlled by CHS[1:0] and GCHS bits. 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 reference high voltage is VDD. The ADC builds in P4CON register to set pure analog input pin. It is necessary to set P4 as input mode without 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. If ADC interrupt function is enabled (ADCIEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 0010H) and executes interrupt service routine after finishing ADC converting. Clear ADCIRQ by program is necessary in interrupt procedure.



9.2 ADC MODE REGISTER

ADM and ADM2 are ADC mode control registers to configure ADC configurations including ADC start, ADC channel selection, and ADC processing indicator…These configurations must be setup completely before starting ADC converting.

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	-	-	CHS1	CHS0
Read/Write	R/W	R/W	R/W	R/W	-	-	R/W	R/W
After reset	0	0	0	0	-	-	0	0

Bit 7 **ADENB:** ADC control bit. **In power saving mode, disable ADC to reduce power consumption.**
0 = Disable ADC function.
1 = Enable ADC function.

Bit 6 **ADS:** ADC start control bit. **ADS bit is cleared after ADC processing automatically.**
0 = ADC converting stops.
1 = Start to execute ADC converting.

Bit 5 **EOC:** ADC status bit. **EOC bit must be cleared by program before ADC start.**
0 = ADC progressing.
1 = End of converting and reset ADS bit.

Bit 4 **GCHS:** ADC global channel select bit.
0 = Disable AIN0~AIN1 channel.
1 = Enable AIN0~AIN1 channel.

Bit [3:0] **CHS[1:0]:** ADC input channel select bit.
00 = AIN0, 01 = AIN1, 10 = VSS, 11 = VDD.

0B9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM2	GCHS2	-	-	-	-	-	-	-
Read/Write	R/W	-	-	-	-	-	-	-
After reset	0	-	-	-	-	-	-	-

Bit 7 **GCHS2:** ADC global channel select bit.
0 = Disable AIN2~AIN5 channel.
1 = Enable AIN2~AIN5 channel.

✷ ***Note: GCHS and GCHS2 can't be set "Enable" at the same time.***

AIN2M~AIN5M are ADC mode control registers to configure ADC channel2~channel5 selections. These configurations must be setup completely before starting ADC converting.

0BAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AIN2M	-	-	AIN2M5	AIN2M4	AIN2M3	AIN2M2	AIN2M1	AIN2M0
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] **AIN2M:** ADC input channel2 select bits.
0x18 = AIN2 isn't ADC analog signal input pin.
0x1A = AIN2 is ADC analog signal input pin.
Others: Reserved.

0BBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AIN3M	-	-	AIN3M5	AIN3M4	AIN3M3	AIN3M2	AIN3M1	AIN3M0
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] **AIN3M:** ADC input channel3 select bits.
0x18 = AIN3 isn't ADC analog signal input pin.
0x1A = AIN3 is ADC analog signal input pin.
Others: Reserved.

0BCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AIN4M	-	-	AIN4M5	AIN4M4	AIN4M3	AIN4M2	AIN4M1	AIN4M0
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] **AIN4M:** ADC input channel4 select bits.
0x18 = AIN4 isn't ADC analog signal input pin.
0x1A = AIN4 is ADC analog signal input pin.
Others: Reserved.

0BDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AIN5M	-	-	AIN5M5	AIN5M4	AIN5M3	AIN5M2	AIN5M1	AIN5M0
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] **AIN5M:** ADC input channel5 select bits.
0x18 = AIN5 isn't ADC analog signal input pin.
0x1A = AIN5 is ADC analog signal input pin.
Others: Reserved.

✸ ***Note:***
1. AIN2~AIN5 can't be set as ADC analog signal input pin at the same time.
2. In power saving mode, set AIN2~AIN5 aren't ADC analog signal input pin to reduce power consumption.

ADR register includes ADC mode control and ADC low-nibble data buffer. ADC configurations including ADC clock rate and ADC resolution. These configurations must be setup completely before starting ADC converting.

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

Bit [6:5] **ADCKS [1:0]:** ADC's clock rate select bit.
00 = Fcpu/16, 01 = Fcpu/8, 10 = Fcpu/1, 11 = Fcpu/2

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

9.3 ADC DATA BUFFER REGISTERS

ADC data buffer is 12-bit length to store ADC converter result. The high byte is ADB register, and the low-nibble is ADR[3:0] bits. The ADB register is only 8-bit register including bit4~bit11 ADC data. To combine ADB register and the low-nibble of ADR will get full 12-bit ADC data buffer. The ADC data 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.**

0C9H	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[11:4]:** 8-bit ADC data buffer and the high-byte data buffer of 12-bit ADC.

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

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

The AIN input voltage v.s. ADB 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 ADC 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

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

9.4 ADC OPERATION DESCRIPTION AND NOTIC

9.4.1 ADC SIGNAL FORMAT

ADC sampling voltage range is limited by high/low reference voltage. The ADC low reference voltage is Vss and not changeable. The ADC high reference voltage is VDD. ADC reference voltage range limitation is **"(ADC high reference voltage – low reference voltage) ≧ 2V"**. ADC low reference voltage is Vss = 0V. **ADC high reference voltage range is = VDD**.

- **ADC Internal Low Reference Voltage = 0V.**
- **ADC Internal High Reference Voltage = VDD.**

ADC sampled input signal voltage must be from ADC low reference voltage to ADC high reference. If the ADC input signal voltage is over the range, the ADC converting result is error (full scale or zero).

- **ADC Low Reference Voltage ≦ ADC Sampled Input Voltage ≦ ADC High Reference Voltage**

9.4.2 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 and ADC clock rate. 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. ADC clock source is Fcpu and includes Fcpu/1, Fcpu/2, Fcpu/8 and Fcpu/16 controlled by ADCKS[1:0] bits.
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 rate/4)*16 sec

ADLEN	ADCKS1, ADCKS0	ADC Clock Rate	Fcpu=4MHz		Fcpu=16MHz	
			ADC Converting time	ADC Converting Rate	ADC Converting time	ADC Converting Rate
1 (12-bit)	00	Fcpu/16	1/(4MHz/16/4)*16 = 256 us	3.906KHz	1/(16MHz/16/4)*16 = 64 us	15.625KHz
	01	Fcpu/8	1/(4MHz/8/4)*16 = 118 us	7.812KHz	1/(16MHz/8/4)*16 = 32 us	31.25KHz
	10	Fcpu	1/(4MHz/4)*16 = 16 us	62.5KHz	1/(16MHz/4)*16 = 4 us	250KHz
	11	Fcpu/2	1/(4MHz/2/4)*16 = 32 us	31.25KHz	1/(16MHz/2/4)*16 = 8 us	125KHz

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

ADLEN	ADCKS1, ADCKS0	ADC Clock Rate	Fcpu=4MHz		Fcpu=16MHz	
			ADC Converting time	ADC Converting Rate	ADC Converting time	ADC Converting Rate
0 (8-bit)	00	Fcpu/16	1/(4MHz/16/4)*12 = 192 us	5.208KHz	1/(16MHz/16/4)*12 = 48 us	20.833KHz
	01	Fcpu/8	1/(4MHz/8/4)*12 = 96 us	10.416KHz	1/(16MHz/8/4)*12 = 24 us	41.667KHz
	10	Fcpu	1/(4MHz/4)*12 = 12 us	83.333KHz	1/(16MHz/4)*12 = 3 us	333.333KHz
	11	Fcpu/2	1/(4MHz/2/4)*12 = 24 us	41.667KHz	1/(16MHz/2/4)*12 = 6 us	166.667KHz

9.4.3 ADC PIN CONFIGURATION

ADC input channels are shared with Port4. ADC channel selection is through CHS[1:0] bit. CHS[1:0] value points to the ADC input channel directly. CHS[1:0]=00 selects AIN0. CHS[1:0]=01 selects AIN1……Only one pin of Port4 can be configured as ADC input in the same time. The pins of Port4 configured as ADC input channel must be set input mode, disable internal pull-up and enable P4CON first by program. After selecting ADC input channel through CHS[1:0], set GCHS bit as “1” to enable ADC channel function.

- **The GPIO mode of ADC input channels must be set as input mode.**
- **The internal pull-up resistor of ADC input channels must be disabled.**
- **P4CON bits of ADC input channel must be set.**

ADC input pins are shared with 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 Port4 will encounter above current leakage situation. P4CON is Port4 configuration register. Write “1” into P4CON [1:0] will configure related Port4 pin as pure analog input pin to avoid current leakage.

0C6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	-	-	-	-	-	-	P4CON1	P4CON0
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

Bit[1:0] **P4CON[1:0]:** 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.***

9.4.4 ADC OPERATION EXAMPLE

- **ADC CONFIGURATION AIN0~AIN1:**

```
; Reset ADC.
        CLR     ADM             ; Clear ADM register.
        CLR     ADM2            ; Clear ADM2 register.

; Set ADC clock rate and ADC resolution.
        MOV     A, #0nnm0000b   ; nn: ADCKS[1:0] for ADC clock rate.
        B0MOV   ADR, A          ; m: ADLEN for ADC reolution.


; Set ADC input channel configuration.
        MOV     A, #value1      ; Set P4CON for ADC input channel.
        B0MOV   P4CON, A
        MOV     A, #value2      ; Set ADC input channel as input mode.
        B0MOV   P4M, A
        MOV     A, #value3      ; Disable ADC input channel's internal pull-up resistor.
        B0MOV   P4UR, A

; Enable ADC.
        B0BSET  FADENB

; Execute ADC 100us warm-up time delay loop.
        CALL    100usDLY        ; 100us delay loop.

; Select ADC input channel.
        MOV     A, #value       ; Set CHS[1:0] for ADC input channel selection.
        OR      ADM, A

; Enable ADC input channel.
        B0BSET  FGCHS

; Enable ADC interrupt function.
        B0BCLR  FADCIRQ         ; Clear ADC interrupt flag.
        B0BSET  FADCIEN         ; Enable ADC interrupt function.

; Start to execute ADC converting.
        B0BSET  FADS
```

✸ ***Note:***

1. ***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.***
2. ***In power saving situation like power down mode and green mode, and not using ADC function, to disable ADC by program is necessary to reduce power consumption.***

- **ADC CONFIGURATION AIN2~AIN5:**

```
; Reset ADC.
        CLR     ADM             ; Clear ADM register.
        CLR     ADM2            ; Clear ADM2 register.
        MOV     A, #0x18        ; Disable ADC input channel.
        B0MOV   AIN2M, A
        B0MOV   AIN3M, A
        B0MOV   AIN4M, A
        B0MOV   AIN5M, A

; Set ADC clock rate and ADC resolution.
        MOV     A, #0nnm0000b   ; nn: ADCKS[1:0] for ADC clock rate.
        B0MOV   ADR, A          ; m: ADLEN for ADC reolution.

; Enable ADC.
        B0BSET  FADENB

; Execute ADC 100us warm-up time delay loop.
        CALL    100usDLY        ; 100us delay loop.

; Enable ADC input channel.
        B0BSET  FGCHS2

; Select ADC input channel.
        MOV     A, #0x1A        ; Set AIN2 for ADC input channel selection.
        B0MOV   AIN2M, A

; Enable ADC interrupt function.
        B0BCLR  FADCIRQ         ; Clear ADC interrupt flag.
        B0BSET  FADCIEN         ; Enable ADC interrupt function.

; Start to execute ADC converting.
        B0BSET  FADS
```

✸ ***Note: In power saving situation like power down mode and green mode, and not using ADC function, to set AIN2~AIN5 aren't ADC input channel by program is necessary to reduce power consumption.***

```
Ex:   B0BSET FGCHS2
      MOV A, #0x18
      B0MOV AIN2M, A
      B0MOV AIN3M, A
      B0MOV AIN4M, A
      B0MOV AIN5M, A
```

- **ADC CONVERTING OPERATION:**

; ADC Interrupt disable mode.

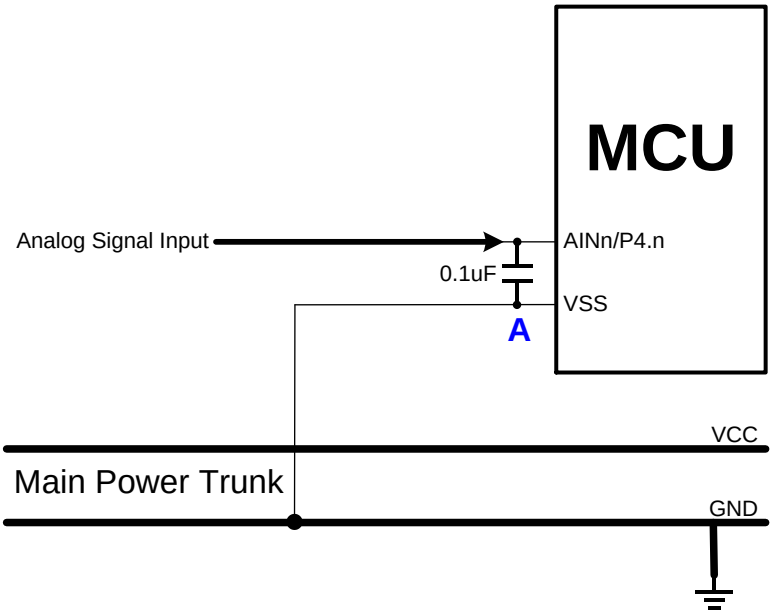
```
@@:
        B0BTS1      FEOC        ; Check ADC processing flag.
        JMP         @B          ; EOC=0: ADC is processing.
        B0MOV       A, ADB      ; EOC=1: End of ADC processing. Process ADC result.
        B0MOV       BUF1,A
        MOV         A, #00001111b
        AND         A, ADR
        B0MOV       BUF2,A
        …                       ; End of processing ADC result.
        CLR         FEOC        ; Clear ADC processing flag for next ADC converting.
```

; ADC Interrupt enable mode.

```
        ORG 10                  ; Interrupt vector.
        JMP         INT_SR
        …
INT_SR:                         ; Interrupt service routine.
        PUSH
        B0BTS1      FADCIRQ     ; Check ADC interrupt flag.
        JMP         EXIT_INT    ; ADCIRQ=0: Not ADC interrupt request.
        B0MOV       A, ADB      ; ADCIRQ=1: End of ADC processing. Process ADC result.
        B0MOV       BUF1,A
        MOV         A, #00001111b
        AND         A, ADR
        B0MOV       BUF2,A
        …                       ; End of processing ADC result.
        CLR         FEOC        ; Clear ADC processing flag for next ADC converting.
        JMP         INT_EXIT

INT_EXIT:
        POP
        RETI                    ; Exit interrupt service routine.
```

✸ ***Note: 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.***

9.5 ADC APPLICATION 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.

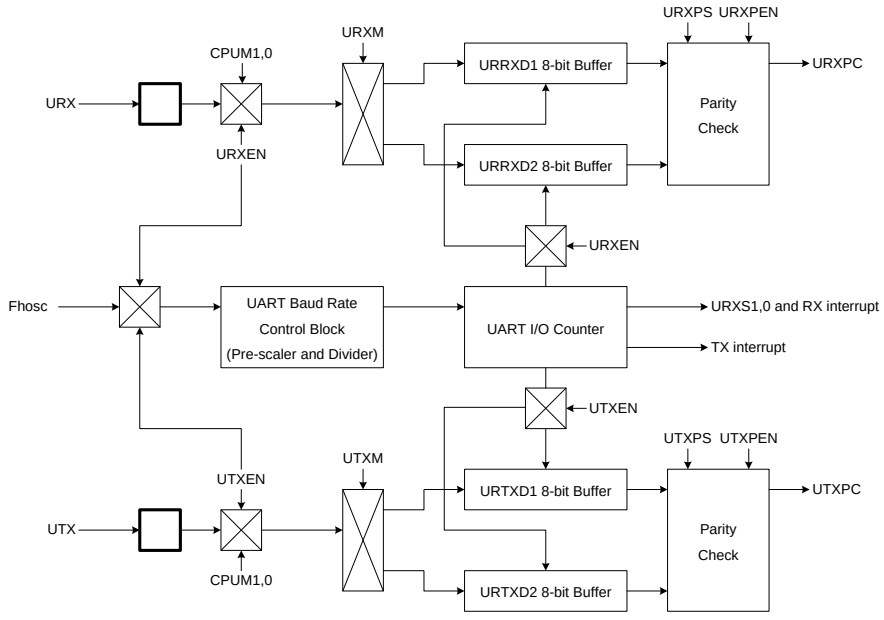
10 Universal Asynchronous Receiver/Transmitter (UART)

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 byte data length. The transfer format has start bit, 8-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 data length.**
- **Odd and even parity bit.**
- **End-of-Transfer interrupt.**
- **Support DMX512 protocol.**
- **Support break pocket function.**
- **Support wide range baud rate.**



UART Interface Structure Diagram

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.

The UART supports interrupt function. URXIEN/UTXIEN are UART transfer interrupt function control bit. URXIEN=0, disable UART receiver interrupt function. UTXIEN=0, disable UART transmitter interrupt function. URXIEN=1, enable UART receiver interrupt function. UTXIEN=1, enable UART transmitter interrupt function. When UART interrupt function enable, the program counter points to interrupt vector (ORG 0013H/0014H) to do UART interrupt service routine after UART operating. URXIRQ/UTXIRQ is UART interrupt request flag, and also to be the UART operating status indicator when URXIEN=0 or UTXIEN=0, but cleared by program. When UART operation finished, the URXIRQ/UTXIRQ would be set to “1”.

The UART also builds in “Busy Bit” to indicate UART bus status. URXBZ bit is UART RX operation indicator. UTXBZ bit is UART TX operation indicator. If bus is transmitting, the busy bit is “1” status. If bus is finishing operation or in idle status, the busy bit is “0” status.

UART TX operation is controlled by loading UTXD data buffer. After UART TX configuration, load transmitted data into UTXD 8-bit buffer, and then UART starts to transmit the pocket following UART TX configuration.

UART RX operation is controlled by receiving the start bit from master terminal. After UART RX configuration, URX pin detects the falling edge of start bit, and then UART starts to receive the pocket from master terminal.

UART provides URXPC bit and UFMER bit to check received pocket. URXPC bit is received parity bit checker. If received parity is error, URXPC sets as “1”. If URXPC bit is zero after receiving pocket, the parity is correct. UFMER bit is received stream frame checker. The stream frame error definition includes “Start bit error”, “Stop bit error”, “Stream length error”, “UART baud rate error”... Each of frame error conditions makes UFMER bit sets as “1” after receiving pocket.

10.3 UART BAUD RATE

UART clock is 2-stage structure including a pre-scaler and an 8-bit buffer. UART clock source is generated from system oscillator called Fhosc. Fhosc passes through UART pre-scaler to get UART main clock called Fuart. UART pre-scaler has 8 selections (Fhosc/1, Fhosc/2, Fhosc/4, Fhosc/8, Fhosc/16, Fhosc/32, Fhosc/64, Fhosc/128) and 3-bit control bits (URS[2:0]). UART main clock (Fuart) purposes are the front-end clock and through UART 8-bit buffer (URCR) to obtain UART processing clock and decide UART baud rate.

UART Pre-scaler Selection, URS[2:0]	UART Main Clock Rate	Fuart (Fhosc=16MHz)
000b	Fhosc/1	16MHz
001b	Fhosc/2	8MHz
010b	Fhosc/4	4MHz
011b	Fhosc/8	2MHz
100b	Fhosc/16	1MHz
101b	Fhosc/32	0.5MHz
110b	Fhosc/64	0.25MHz
111b	Fhosc/128	0.125MHz

0D7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URCR	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0
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 UART baud rate clock source is Fhosc and divided by pre-scalar. The equation is as following.

UART Baud Rate = 1/2 *(Fuart * 1/(256 - URCR))…bps

Fhosc = 16MHz

Baud Rate	UART Pre-scaler	URS[2:0]	URCR (Hex)	Accuracy (%)
1200	Fhosc/32	101b	30	-0.16%
2400	Fhosc/32	101b	98	-0.16%
4800	Fhosc/32	101b	CC	-0.16%
9600	Fhosc/32	101b	E6	-0.16%
19200	Fhosc/32	101b	F3	-0.16%
38400	Fhosc/1	000b	30	-0.16%
51200	Fhosc/1	000b	64	-0.16%
57600	Fhosc/1	000b	75	0.08%
102400	Fhosc/1	000b	B2	-0.16%
115200	Fhosc/1	000b	BB	-0.64%
128000	Fhosc/1	000b	C1	0.80%
250000	Fhosc/1	000b	E0	0.00%

✸ ***Note:***
1. ***We strongly recommend not to set URCR = 0xFF, or UART operation would be error.***
2. ***If Noise_Filter code option is "Enable", we strongly recommend to set Fcpu as Fhosc/2~Fhosc/16, or UART operation would be error. If Noise_Filter code option is "Disable", the limitation doesn't exist.***

10.4 UART TRANSFER FORMAT

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.

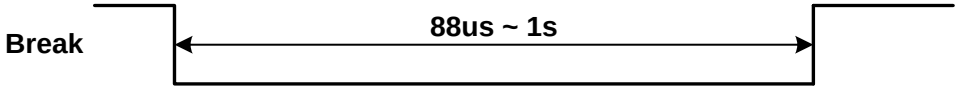
8-bit Data: The data format is 8-bit length, and LSB transfers first following start bit. The one bit data duration is the unit of UART baud rate controlled by register.

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.

10.5 BREAK POCKET

The break pocket is an empty stream to reset UART bus. Break pocket is like a long time zero pocket, and the period is 88us~1s.



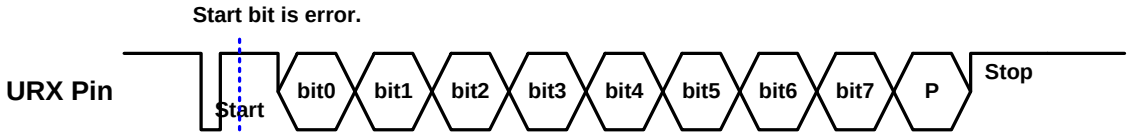
TX Break Pocket: UART builds in a UTXBRK bit to transmit Break pocket. When UTXEN = 1 (enable UART TX function), set UTXBRK bit to transmit Break pocket. When Break pocket finishes transmitting, UTXIRQ is set as “1”, and UTXBRK is cleared automatically. The period of transmitted break pocket is 25 UART baud rate clocks. If UART baud rate is 250000bps, the break pocket period is 100us.

UART TX Break Pocket Period = 25/UART Baud Rate…sec

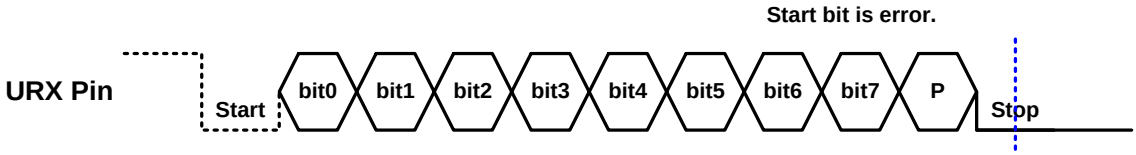
RX Break Pocket:
UART receives break pocket will get a frame error signal because the data period is longer than typical UART duration. UART can’t receive a complete data pocket. After receiving a UART pocket, the break pocket is still output low. UART issues frame error flag (UFMER = 1) and URXIRQ. Maybe the parity bit is error in parity mode. UART changes to initial status until detecting next start bit.

10.6 ABNORMAL POCKET

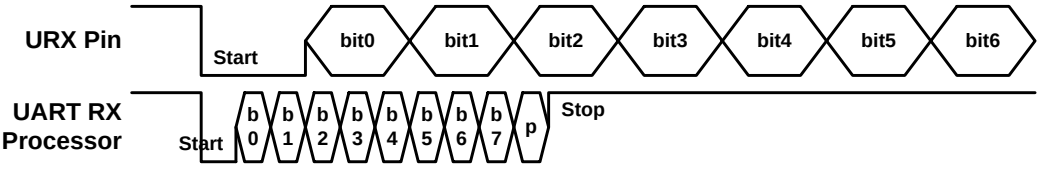
The abnormal pocket occurs in UART RX mode. Break pocket is one abnormal pocket of the UART architecture. The abnormal pocket includes Stream period error, start bit error, stop bit error…When UART receives abnormal pocket, the UFMER bit will be set “1”, and UART issues URXIRQ. The system finds the abnormal pocket through firmware. UART changes to initial status until detecting next start bit.



UART check the start bit is error and issue UFMER flag, but the UART still finishes receiving the pocket.



UART check the stop bit is error and issue UFMER flag, but the UART still finishes receiving the pocket.



If the host’s UART baud rate isn’t match to receiver terminal, the received pocket is error. But it is not easy to differentiate the pocket is correct or not, because the received error pocket maybe match UART rule, but the data is error. Use checking UFMER bit and URXPC bit status to decide the stream. If the two conditions seem like correct, but the pocket is abnormal, UART will accept the pocket as correct one.

10.7 UART RECEIVER CONTROL REGISTER

0D6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRX	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0
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 **URXEN**: UART RX control bit.
0 = Disable UART RX. URX pin is GPIO mode or returns to GPIO status.
1 = Enable UART RX. URX pin exchanges from GPIO mode to UART RX mode.

Bit 6 **URXPEN**: UART RX parity bit control bit.
0 = Disable UART RX parity bit function. The data stream doesn’t include parity bit.
1 = Enable UART RX parity bit function. The data stream includes parity bit.

Bit 5 **UTXPS**: 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 4 **URXPC**: UART RX parity bit checking flag.
0 = Parity bit is correct or no parity function.
1 = Parity bit is error.

Bit 3 **UFMER**: UART RX stream frame error flag bit.
0 = Collect UART frame.
1 = UART frame is error including start/stop bit, stream length.

Bit [2:0] **URS[2:0]**: UART per-scalar select bit.
000 = Fhosc/1, 001 = Fhosc/2, 010 = Fhosc/4, 011 = Fhosc/8, 100 = Fhosc/16, 101 = Fhosc/32, 110 = Fhosc/64, 111 = Fhosc/128.

10.8 UART TRANSMITTER CONTROL REGISTER

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTX	UTXEN	UTXPEN	UTXPS	UTXBRK	URXBZ	UTXBZ	-	-
Read/Write	R/W	R/W	R/W	R/W	R	R	-	-
After reset	0	0	0	0	0	0	-	-

Bit 7 **UTXEN**: UART TX control bit.
0 = Disable UART TX. UTX pin is GPIO mode or returns to GPIO status.
1 = Enable UART TX. UTX pin exchanges from GPIO mode to UART TX mode and idle high status.

Bit 6 **UTXPEN**: UART TX parity bit control bit.
0 = Disable UART TX parity bit function. The data stream doesn't include parity bit.
1 = Enable UART TX parity bit function. The data stream includes parity bit.

Bit 5 **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 4 **UTXBRK**: UART TX BREAK pocket control bit.
0 = End of transmitting UART BREAK pocket.
1 = Start to transmit UART BREAK pocket.

Bit 3 **URXBZ**: UART RX operating status flag.
0 = UART RX is idle or the end of processing.
1 = UART RX is busy and processing.

Bit 2 **UTXBZ**: UART TX operating status flag.
0 = UART TX is idle or the end of processing.
1 = UART TX is busy and processing.

✸ ***Note: URXBZ and UTXBZ bits are UART operating indicators. After setting UART RX/TX operations, set a "NOP" instruction is necessary, and then check UART status through URXBZ and UTXBZ bits.***

10.9 UART DATA BUFFER

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UTXD	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0
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] **UTXD:** UART transmitted data buffer.

0D9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URXD	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0
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] **URXD:** UART received data buffer.

10.10 UART OPERATION EXAMPLE

- **UART TX Configuration:**

```
; Select parity bit function.
        B0BCLR      FUTXPEN     ; Disable UART TX parity bit function.
;or
        B0BSET      FUTXPEN     ; Enable UART TX parity bit function.

; Select parity bit format.
        B0BCLR      FUTXPS      ; UART TX parity bit format is even parity.
;or
        B0BSET      FUTXPS      ; UART TX parity bit format is odd parity.

; Set UART baud rate.
        MOV         A, #value1  ; Set UART pre-scaler URS[2:0].
        B0MOV       URRX, A
        MOV         A, #value2  ; Set UART baud rate 8-bit buffer.
        B0MOV       URCR, A

; Enable UART TX pin.
        B0BSET      FUTXEN      ; Enable UART TX function and UART TX pin.

; Enable UART TX interrupt function.
        B0BCLR      FUTXIRQ     ; Clear UART TX interrupt flag.
        B0BSET      FUTXIEN     ; Enable UART TX interrupt function.

; Load TX data buffer and execute TX transmitter.
        MOV         A, #value3  ; Load 8-bit data to UTXD data buffer.
        B0MOV       UTXD, A
                                ;After loading UTXD, UART TX starts to transmit.
        NOP                     ; One instruction delay for UTXBZ flag.

; Check TX operation.
        B0BTS0      FUTXBZ      ; Check UTXBZ bit.
        JMP         CHKTX       ; UTXBZ=1, TX is operating.
        JMP         ENDTX       ; UTXBZ=0, the end of TX.
```

✷ ***Note: UART TX operation is started through loading UTXD data buffer.***

- **Transmit Break Pocket:**

```
; Select parity bit function.
        B0BCLR      FUTXPEN     ; Disable UART TX parity bit function.
;or
        B0BSET      FUTXPEN     ; Enable UART TX parity bit function.

; Select parity bit format.
        B0BCLR      FUTXPS      ; UART TX parity bit format is even parity.
;or
        B0BSET      FUTXPS      ; UART TX parity bit format is odd parity.

; Set UART baud rate.
        MOV         A, #value1  ; Set UART pre-scaler URS[2:0].
        B0MOV       URRX, A
        MOV         A, #value2  ; Set UART baud rate 8-bit buffer.
        B0MOV       URCR, A

; Enable UART TX pin.
        B0BSET      FUTXEN      ; Enable UART TX function and UART TX pin.

; Enable UART TX interrupt function.
        B0BCLR      FUTXIRQ     ; Clear UART TX interrupt flag.
        B0BSET      FUTXIEN     ; Enable UART TX interrupt function.

; Start UART break pocket.
        B0BSET      FUTXBRK     ; Transmit UART break pocket.
        NOP                     ; One instruction delay for UTXBZ flag.

; Check TX operation.
        B0BTS0      FUTXBZ      ; Check UTXBZ bit.
        JMP         CHKTX       ; UTXBZ=1, TX is operating.
        JMP         ENDTX       ; UTXBZ=0, the end of TX.
```

✸ ***Note: UART TX break pocket is controlled by UTXBRK bit and needn't load UTXD buffer.***

- **UART RX Configuration:**

```
; Select parity bit function.
        B0BCLR      FURXPEN     ; Disable UART RX parity bit function.
;or
        B0BSET      FURXPEN     ; Enable UART RX parity bit function.

; Select parity bit format.
        B0BCLR      FURXPS      ; UART RX parity bit format is even parity.
;or
        B0BSET      FURXPS      ; UART RX parity bit format is odd parity.

; Set UART baud rate.
        MOV         A, #value1  ; Set UART pre-scaler URS[2:0].
        B0MOV       URRX, A
        MOV         A, #value2  ; Set UART baud rate 8-bit buffer.
        B0MOV       URCR, A

; Enable UART RX pin.
        B0BSET      FURXEN      ; Enable UART RX function and UART RX pin.

; Enable UART RX interrupt function.
        B0BCLR      FURXIRQ     ; Clear UART RX interrupt flag.
        B0BSET      FURXIEN     ; Enable UART RX interrupt function.
        NOP                     ; One instruction delay for URXBZ flag.


; Check RX operation.
        B0BTS0      FURXBZ      ; Check URXBZ bit.
        JMP         CHKRX       ; URXBZ=1, RX is operating.
        JMP         ENDRX       ; URXBZ=0, the end of RX.
```

✸ ***Note: UART RX operation is started as start bit transmitted from master terminal.***

11 SERIAL INPUT/OUTPUT TRANSCEIVER (SIO)

11.1 OVERVIEW

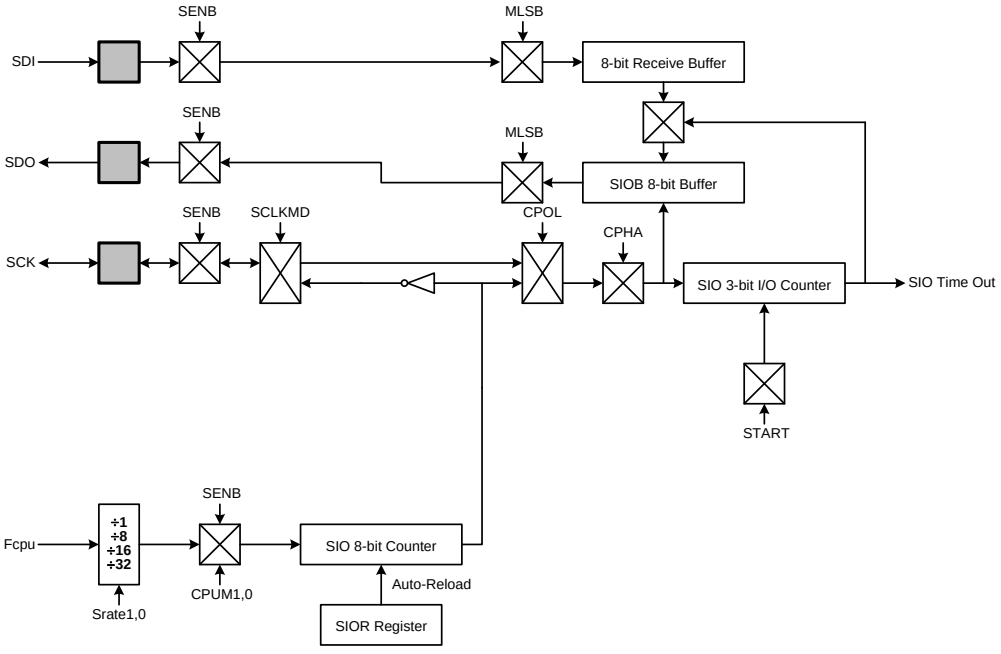
The SIO (serial input/output) transceiver is a serial communicate interface for data exchanging from one MCU to one MCU or other hardware peripherals. It is a simple 8-bit interface without a major definition of protocol, packet or control bits. The SIO transceiver includes three pins, clock (SCK), data input (SDI) and data output (SDO) to send data between master and slaver terminals. The SIO interface builds in 8-mode which are the clock idle status, the clock phases and data fist bit direction. The 8-bit mode supports most of SIO/SPI communicate format.

The SIO features include the following:

- **Full-duplex, 3-wire synchronous data transfer.**
- **Master (SCK is clock output) or Slave (SCK is clock input) operation.**
- **MSB/LSB first data transfer.**
- **The start phase of data sampling location selection is 1st-phase or 2nd-phase controlled register.**
- **SCK, SDI, SDO are programmable open-drain output pin for multiple salve devices application.**
- **Two programmable bit rates (Only in master mode).**
- **End-of-Transfer interrupt.**

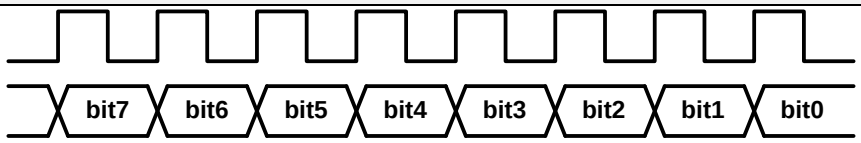
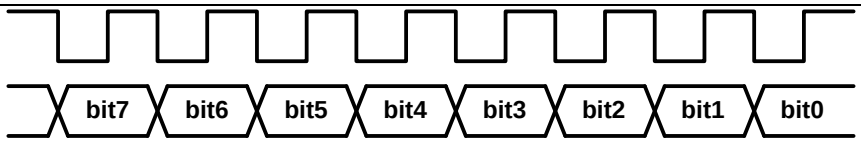
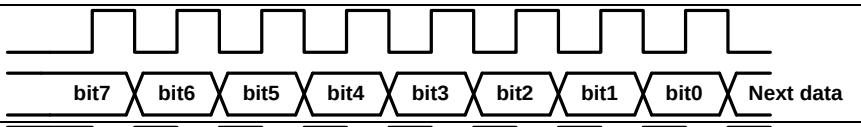
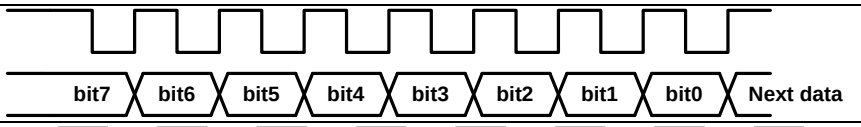
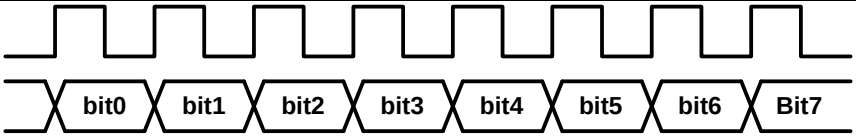
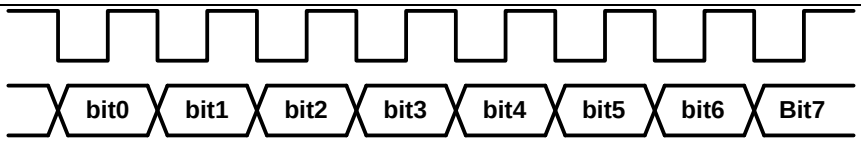
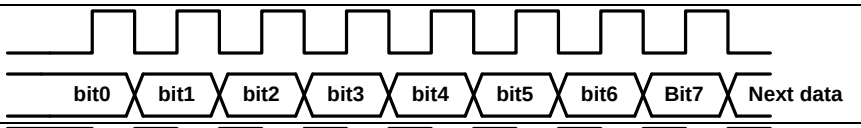
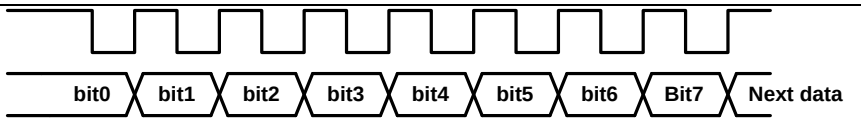
11.2 SIO OPERATION

The SIOM register can control SIO operating function, such as: transmit/receive, clock rate, data transfer direction, SIO clock idle status and clock control phase 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 SIO data buffer is double buffer design. When the SIO operating, the SIOB register stores transfer data and one internal buffer stores receive data. When SIO operation is successfully, the internal buffer reloads into SIOB register automatically. The SIO 8-bit counter and SIOR register 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. CPOL bit is designed to control SIO clock idle status. CPHA bit is designed to control the clock edge direction of data receive. CPOL and CPHA bits decide the SIO format. The SIO data transfer direction is controlled by MLSB bit to decide MSB first or LSB first.



SIO Interface Structure Diagram

The SIO supports 8-mode format controlled by MLSB, CPOL and CPHA bits. The edge direction is “Data Transfer Edge”. When setting rising edge that means to receive and transmit one bit data at SCK rising edge, and data transition is at SCK falling edge. When setting falling edge, that means to receive and transmit one bit data at SCK falling edge, and data transition is at SCK rising edge.
“CPHA” is the clock phase bit controls the phase of the clock on which data is sampled. When CPHA=1, the SCK first edge is for data transition, and receive and transmit data is at SCK 2nd edge. When CPHA=0, the 1st bit is fixed already, and the SCK first edge is to receive and transmit data. The SIO data transfer timing as following figure:

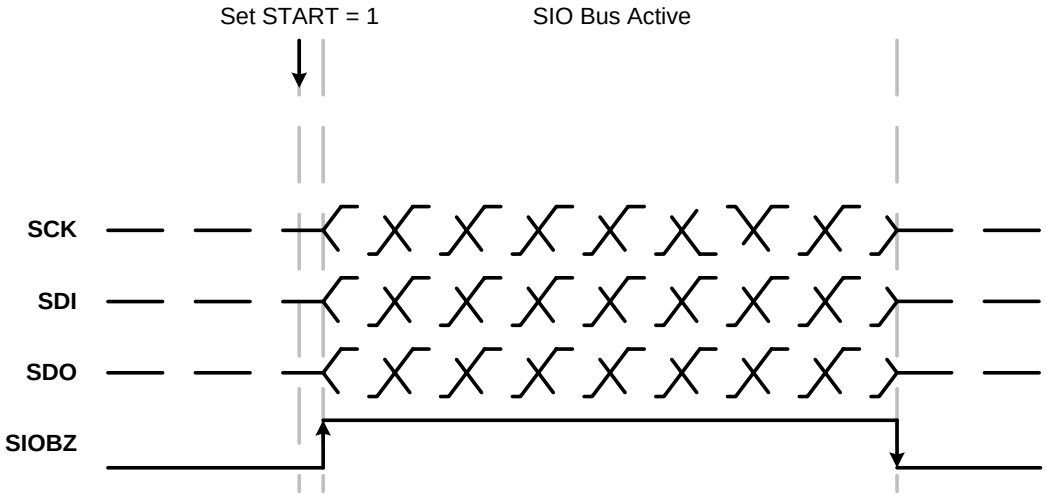
MLSB	CPOL	CPHA	Diagrams	Description
0	0	1		SCK idle status = Low. The transfer first bit = MSB. SCK data transfer edge = Falling edge.
0	1	1		SCK idle status = High. The transfer first bit = MSB. SCK data transfer edge = Rising edge.
0	0	0		SCK idle status = Low. The transfer first bit = MSB. SCK data transfer edge = Rising edge.
0	1	0		SCK idle status = High. The transfer first bit = MSB. SCK data transfer edge = Falling edge.
1	0	1		SCK idle status = Low. The transfer first bit = LSB. SCK data transfer edge = Falling edge.
1	1	1		SCK idle status = High. The transfer first bit = LSB. SCK data transfer edge = Rising edge.
1	0	0		SCK idle status = Low. The transfer first bit = LSB. SCK data transfer edge = Rising edge.
1	1	0		SCK idle status = High. The transfer first bit = LSB. SCK data transfer edge = Falling edge.

SIO Data Transfer Timing

The SIO supports interrupt function. SIOIEN is SIO interrupt function control bit. SIOIEN=0, disable SIO interrupt function. SIOIEN=1, enable SIO interrupt function. When SIO interrupt function enable, the program counter points to interrupt vector (ORG 0011H) to do SIO interrupt service routine after SIO operating. SIOIRQ is SIO interrupt request flag, and also to be the SIO operating status indicator when SIOIEN = 0, but cleared by program. When SIO operation finished, the SIOIRQ would be set to "1", and the operation is the inverse status of SIO "START" control bit.
The SIOIRQ and SIO START bit indicating the end status of SIO operation is after one 8-bit data transferring. The duration from SIO transfer end to SIOIRQ/START active is about **"1/2*SIO clock"**, means the SIO end indicator doesn't active immediately.

✸ ***Note: The first step of SIO operation is to setup the SIO pins' mode. Enable SENB, select CPOL and CPHA bits. These bits control SIO pins' mode.***

SIO builds in SIOBZ bit to indicate SIO processing status. SIOBZ=1 means SIO is processing. SIOBZ=0 means SIO is in idle status or the end of SIO processing. When SIO bus starts to execute, the SIOBZ bit changes to logic high status. When SIO bus finishes transmitting, the SIOBZ bit changes to logic low status. SIOBZ operation of different modes is as below diagram.



11.3 SIOM MODE REGISTER

0D1H	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 SIO function. SIO pins are GPIO.
1 = Enable SIO function. GPIO pins are SIO pins.
SIO pin structure can be push-pull structure and open-drain structure controlled by P1OC register.

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

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

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

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

Bit 1 **CPOL:** SCK idle status control 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 first clock phase.
1 = Data receive at the second clock phase.

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOC	-	-	-	-	-	SIOBZ	-	-
Read/Write	-	-	-	-	-	R	-	-
After reset	-	-	-	-	-	0	-	-

Bit 2 **SIOBZ**: SIO operating status flag.
0 = SIO is idle or end of processing.
1 = SIO is busy and processing.

Because SIO function is shared with GPIO. The following table shows the SIO pin mode mode behavior and setting when SIO function enable and disable.

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

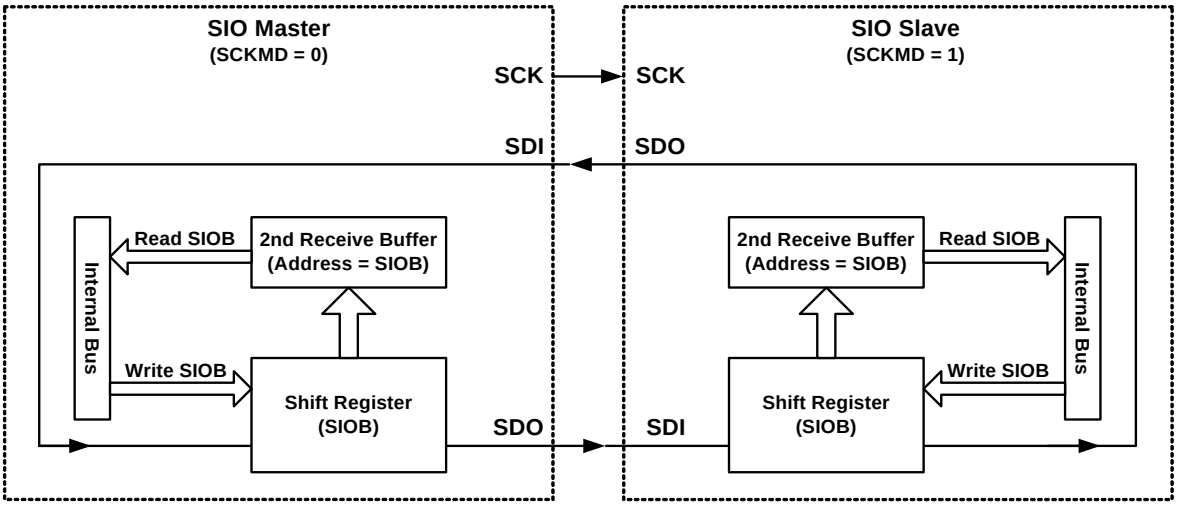
✸ ***Note:***

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.***
3. ***SIO pin can be push-pull structure and open-drain structure controlled by P0OC register.***

11.4 SIOB DATA BUFFER

0D3H	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. 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 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

11.5 SIOR REGISTER DESCRIPTION

0D2H	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-scalar 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))/2

SIOR = 256 - (1 / (2 * SCK frequency) * SIO rate)

- **Example: Setup the SIO clock to be 5KHz. Fhosc = 4MHz. SIO's rate = Fcpu = Fhosc/4.**

SIOR = 256 – (1/(2*5KHz) * 4MHz/4)
= 256 – (0.0001*1000000)
= 256 – 100
= 156

12 MAIN SERIAL PORT (MSP)

12.1 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:

- **Master Tx,Rx Mode**
- **Slave Tx,Rx mode (with general address call) for multiplex slave in single master situation.**

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 multiplex salve devices application.**
- **Support 400K clock rate @ Fcpu=4MIPs.**
- **End-of-Transfer/Receiver interrupt.**

12.2 MSP STATUS REGISTER

0DAH	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.***
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.

✸ ***Note: 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

✸ ***Note: 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.

12.3 MSP MODE REGISTER 1

0DBH	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 SSPBUF register was attempted while the MSP conditions were not valid for a transmission to be started
Slave Mode:
0 = No collision
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)

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

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

✸ ***Note: MSP status register will be clear after MSP Disable. So, user should setting MSP register again before MSP Enable.***

```
Ex:  B0bclr   FMSPENB
     Call     MSP_init_setting
     B0bset   FMSPENB
```

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 CPK 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 Slave 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.

12.4 MSP MODE REGISTER 2

0DCH	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 AKDT data bit. Automatically cleared by hardware.

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

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.

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.

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.

12.5 MSP MSPBUF REGISTER

0DDH	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[7:0]: MSP Buffer.

12.6 MSP MSPADR REGISTER

0DEH	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 [7:1] 7-bit Address.

Bit 0 Tx/Rx mode control bit.
0=Tx mode.
1=Rx mode.

12.7 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.

12.7.1 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 SSPOV 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

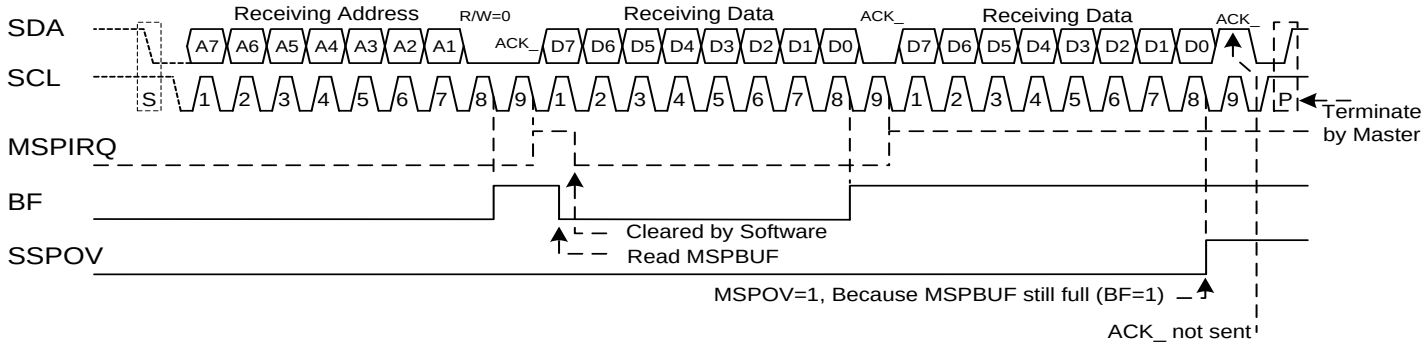
✸ ***Note: BF=0, MSPOV=1 shows the software is not set properly to clear Overflow register.***

12.7.2 Slave Receiving

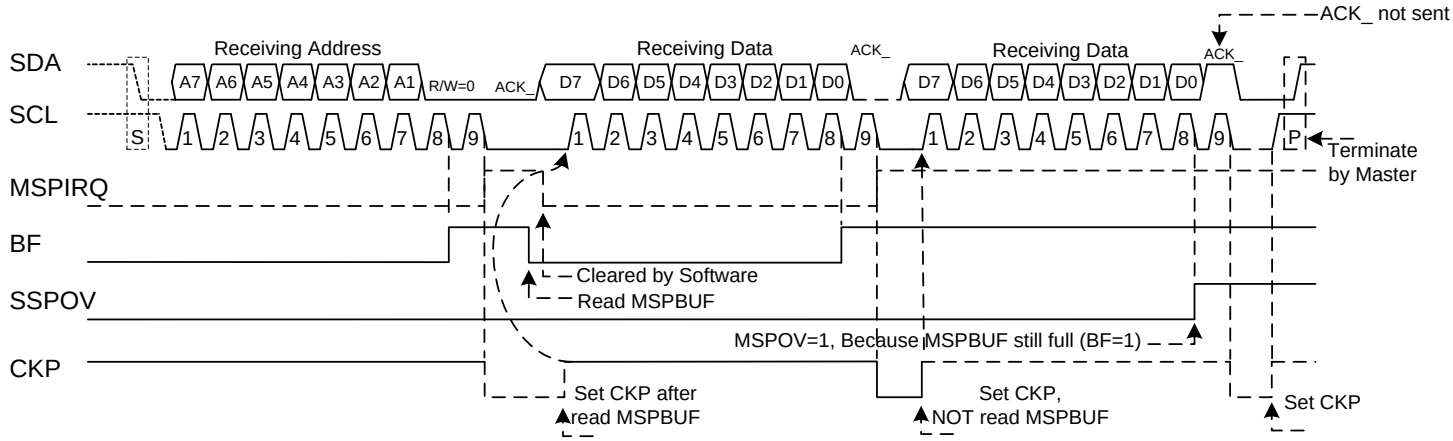
When the R/W bit of address byte =0 and address is matched, the R/W bit of MSPSTAT is cleared. The address will be load into MSPBUF. After reply an ACK_ signal, MSP will receive data every 8 clock. The CKP function enable or disable (Default) is controlled by SLRXCKP bit and data latch edge -Rising edge (Default) or Falling edge is controlled by CPE 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

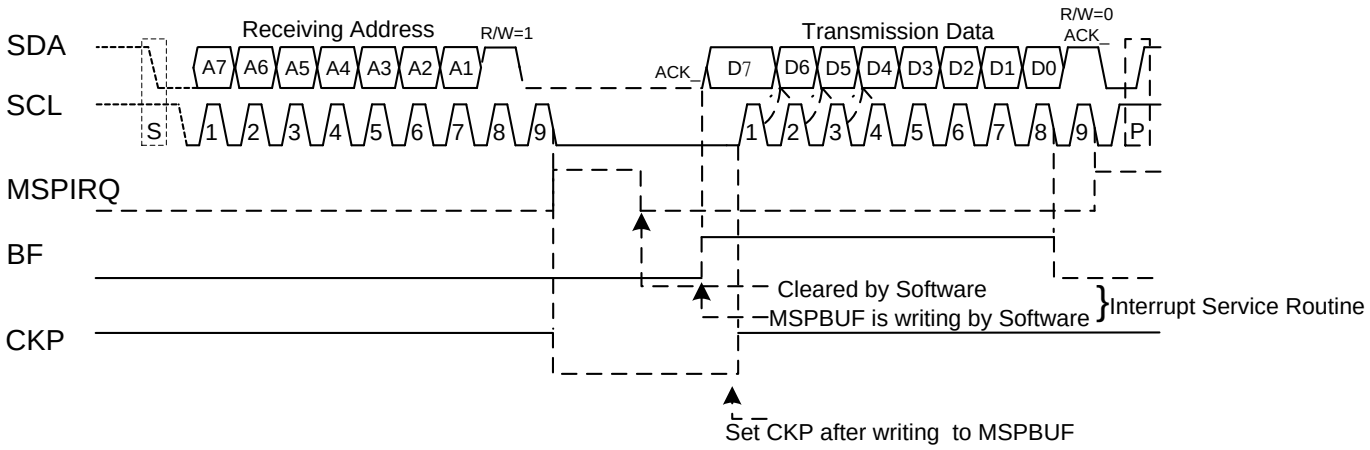


SLRXCKP=1



12.7.3 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.



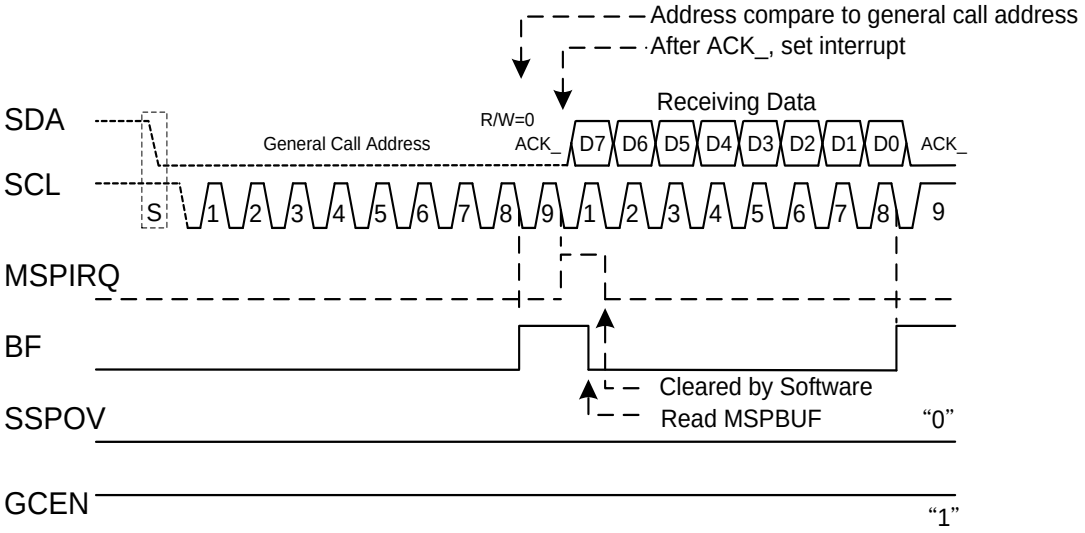
MSP Slave Transmission Timing Diagram

12.7.4 General Call Address

In MSP bus, the first 7-bit 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 GECN=1, following a START signal, 8-bit will shift into MSPSR and the address is compared with MSPADD and also the general call address which fixed by hardware.

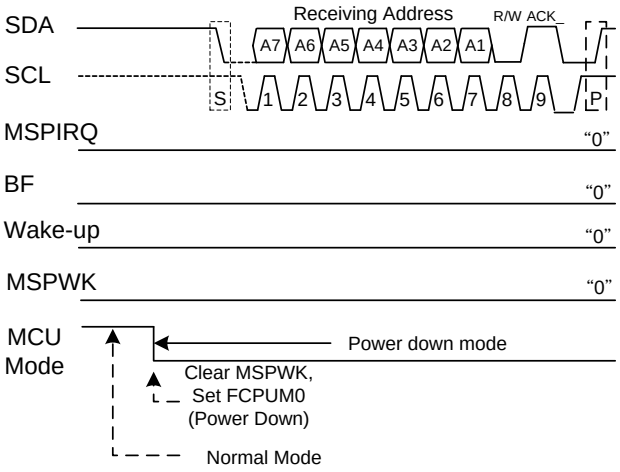
If the genera call address matches, the MSPSR data is transferred into MSPBUF, the BF flag bit is set, and in the falling edge of the ninth clock (ACK_) MSPIRQ flag set for interrupt request. In the interrupt service routine, reading MSPBUF can check if the address is the general call address or device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



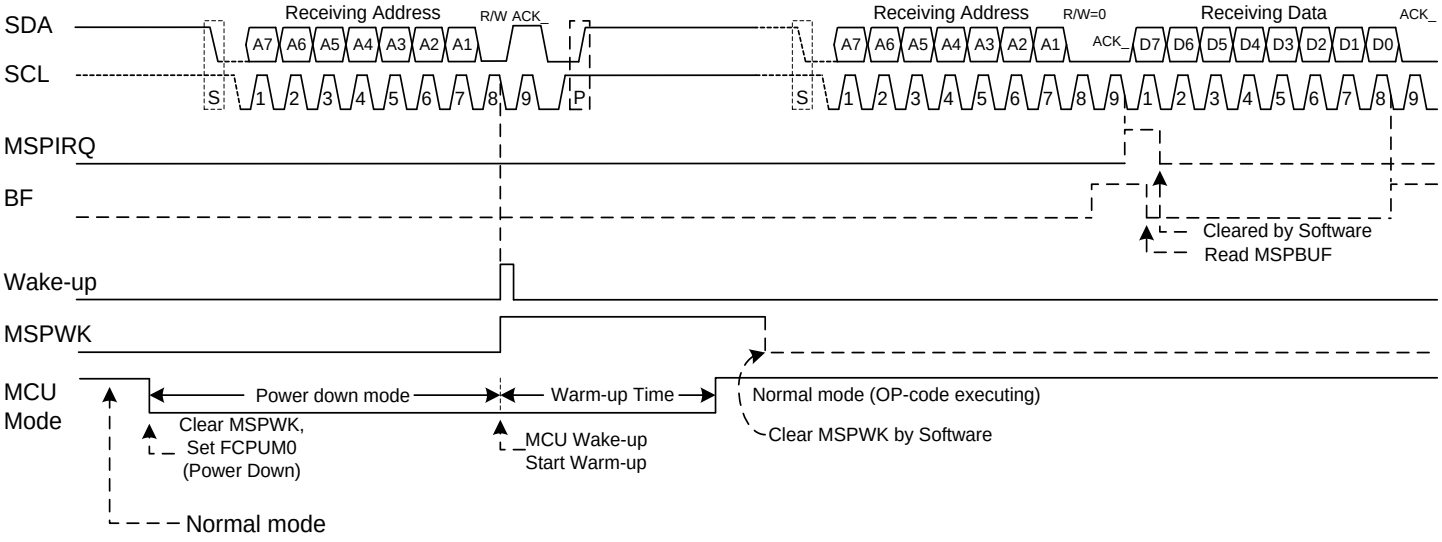
General Call Address Timing Diagram

12.7.5 Slave Wake up

When MCU enter Power down mode, if MSBENB bit is still set, MCU can wake-up by matched device address. The address of MSP bus following START bit, 8-bit 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 matches, MCU will keep in power down mode.
Note 3. Clear MSPWK before enter power down mode by Software for wake-up indication.

12.8 MASTER MODE

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.

12.8.1 Mater Mode Support

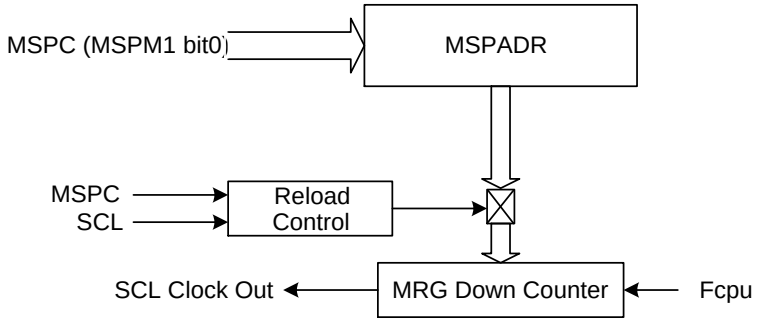
Master mode enable when MSPC and MSPENB bit set. Once the Master mode enabled, the user had following six options.

- 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.

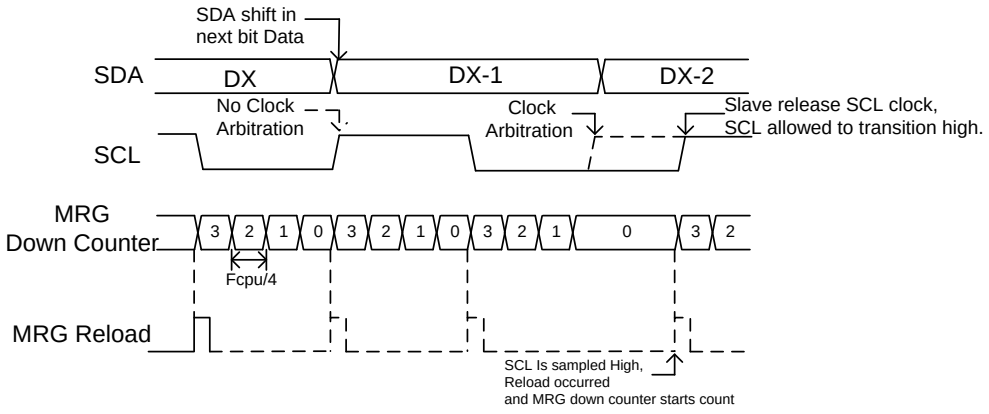
12.8.2 MSP Rate Generator

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



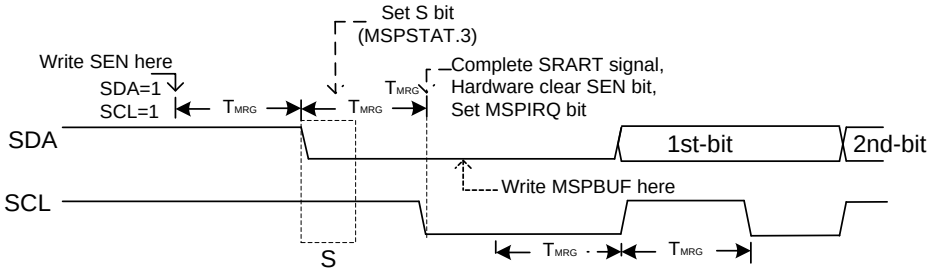
MRG Timing Diagram with and without Clock Arbitration (MSPADR=0x03)

12.8.3 MSP Mater 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.

12.8.3.1WCOL 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

12.8.4 MSP Master mode Repeat START Condition

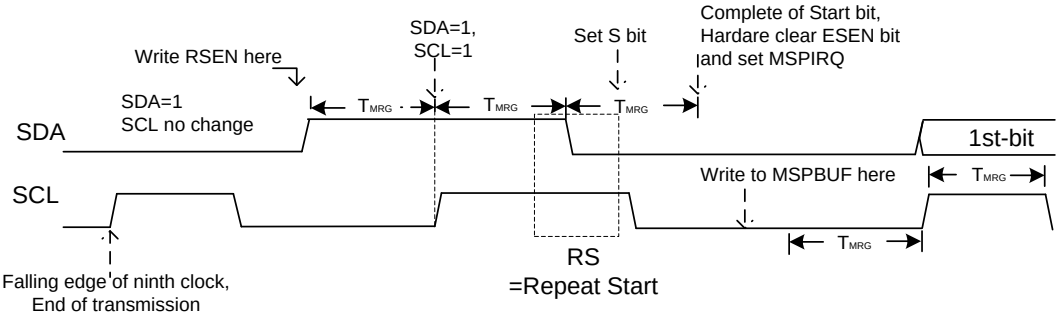
When MSP logic module is idle and RSEN set to 1, Repeat Start progress occurs. RSEN set and SCL pin is sampled low, MSPADR[6:0] data reload to MSP rate generator and start down counter. The SDA pin is release to high in one MSP rate generate counter (T_{MRG}). When the MRG is overflow, if SDA is sampled 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.***
2. ***A bus collision during the Repeat Start condition occurs:* SDA is sampled low when SCL *goes from low to high.***

12.8.4.1WCOL 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)



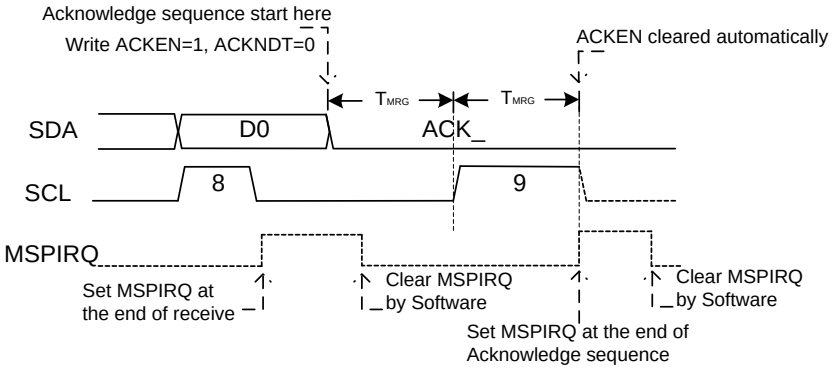
Repeat Start Condition Timing Diagram

12.8.5 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 whished to reply a 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.

12.8.5.1WCOL 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)



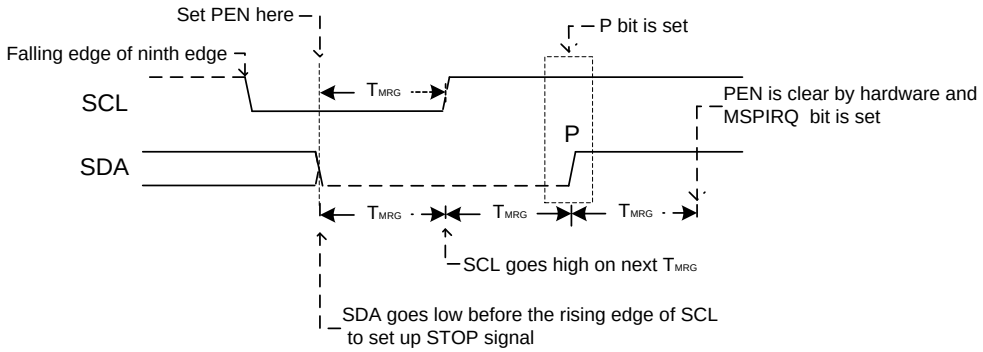
Acknowledge Sequence Timing Diagram

12.8.6 STOP Condition Timing

At the end of received/transmitted, a STOP signal present on SDA pin by setting the STOP bit register, PEN (MSPM2.1). 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.

12.8.6.1WCOL Status Flag

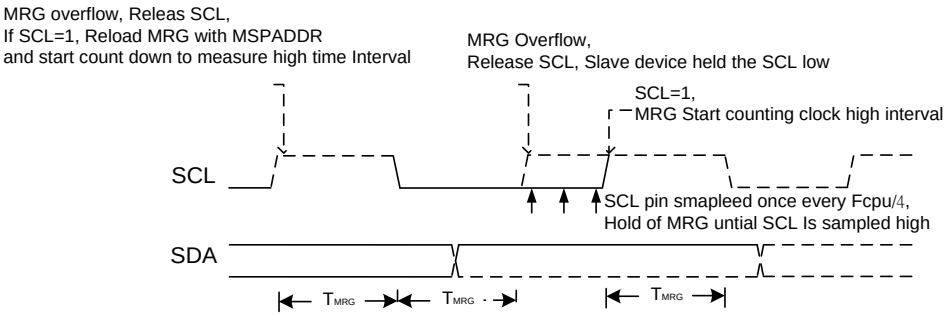
If user write to MSPBUF when a STOP condition is processing, then WCOL bit is set and the content of MSPBUF data is un-changed. (the writer doesn't occur)



STOP condition sequence Timing Diagram

12.8.7 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 master 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

12.8.8 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 failing 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.

12.8.8.1BF Status Flag

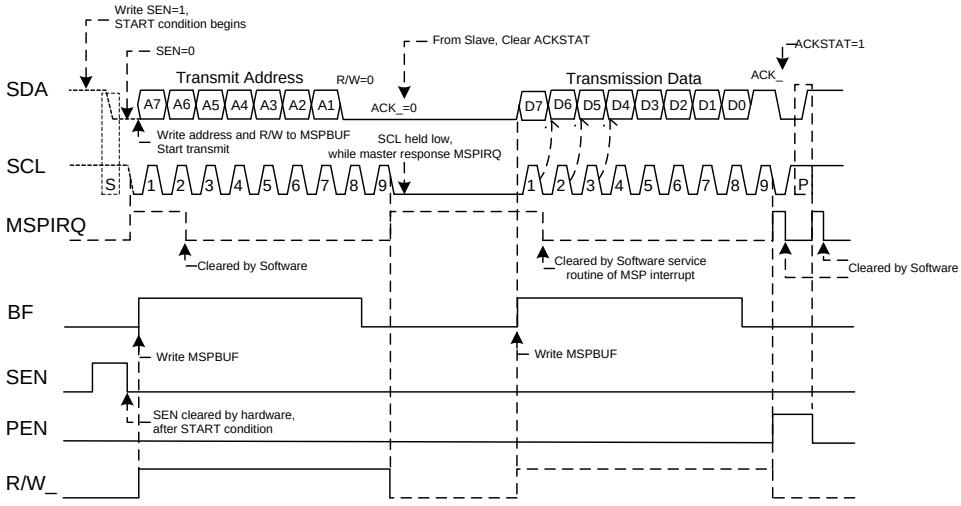
In transmission mode, the BF bit is set when user writes to MSPBUF and is cleared automatically when all 8 bit data are shift out.

12.8.8.2WCOL Flag

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

12.8.8.3ACKSTAT 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

12.8.9 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 fro, 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 cleat automatically. By setting ACKEN bit, user can send an acknowledge bit at the end of receiving.

12.8.9.1BF Status Flag

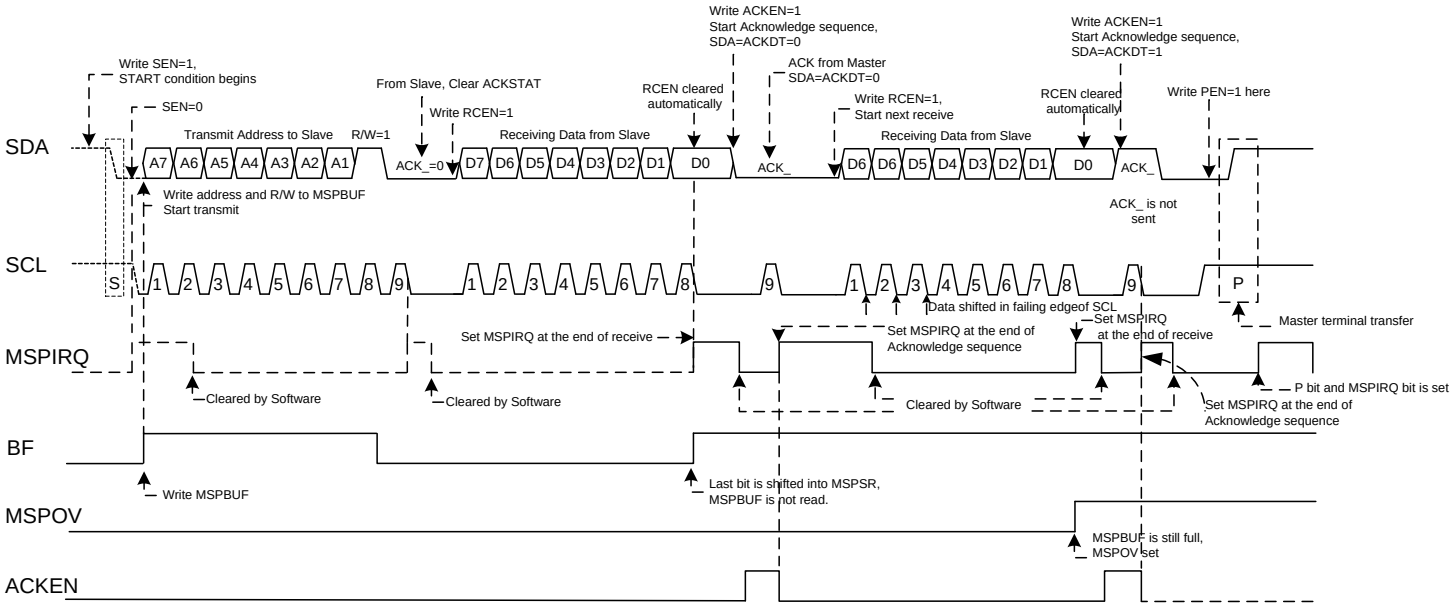
In Reception mode, the BF bit is set when an address or data byte is loaded into MSPBUF from MSPSR. It is cleared automatically when MSPBUF is read.

12.8.9.2MSPOV 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

12.8.9.3WCOL 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

13 IN SYSTEM PROGRAM FLASH ROM

13.1 OVERVIEW

The SN8F27E90 series 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. The SN8F27E90 provides security options at the disposal of the designer to prevent unauthorized access to information stored in FLASH memory. ISP Flash ROM provided user an easy way to storage data into Flash ROM. The ISP concept is memory mapping idea that is to move RAM buffer to flash ROM. Choice ROM/RAM address and executing ROM programming command – PECMD, after programming words which controlled by PERAMCNT, PERAML/PERAMCNT data will be programmed into address PEROML/PEROMH.

	RAM (byte)			Flash ROM (word)	
RAM Address	bit7 ~ bit0		ROM Address	bit15 ~ bit8	bit7 ~ bit0
X	DATA0		Y	DATA1	DATA0
X+1	DATA1		Y+1	DATA3	DATA2
X+2	DATA2	=>	Y+2		
X+3	DATA3		Y+3		
…			…		
…			…		
X+N	DATAN		Y+M	DATAN	DATAN-1

During Flash program or erase operation, the MCU is stalled, although peripherals (Timers, WDT, I/O, PWM, etc.) remain active. When PECMD register is set to execute ISP program and erase operations, the program counter stops, op-code can't be dumped from flash ROM, instruction stops operating, and program execution is hold not to active. At this time hardware depends on ISP operation configuration to do flash ROM erasing and flash ROM programming automatically. After ISP operation is finished, hardware releases system clock to make program counter running, system returns to last operating mode, and the next instruction is executed. Recommend to add two "NOP" instructions after ISP operations.

- **ISP flash ROM erase time = 25ms……1-page, 128-word.**
- **ISP flash ROM program time = 28us……1-word.**
  ISP flash ROM program time = 56us……2-word.
  …
  ISP flash ROM program time = 448us……16-word.
  …
  ISP flash ROM program time = 896us……32-word.

✸ ***Note:***

1. ***Watch dog timer should be clear before the Flash write (program) or erase operation, or watchdog timer would overflow and reset system during ISP operating.***
2. ***Besides program execution, all functions keep operating during ISP operating, e.g. timer, ADC, SIO, UART, MSP... All interrupt events still active and latch interrupt flags automatically. If any interrupt request occurs during ISP operating, the interrupt request will be process by program after ISP finishing.***

13.2 ISP FLASH ROM ERASE OPERATION

ISP flash ROM erase operation is to clear flash ROM contents to blank status “1”. Erasing ROM length is 128-word and has ROM page limitation. ISP flash ROM erase ROM map is as following:

ISP ROM MAP		ROM address bit0~bit6 (hex)														
		0000	0001	0002	…	0010	0011	…	0050	0051	…	0070	0071	…	007E	007F
ROM address bit7~bit15 (hex)	0000	**This page includes reset vector and interrupt sector. We strongly recommend to reserve the area not to do ISP erase.**														
	0080	One ISP Erase Page														
	0100	One ISP Erase Page														
	0180	One ISP Erase Page														
	0200	One ISP Erase Page														
	0280	One ISP Erase Page														
	…	One ISP Erase Page														
	0F00	One ISP Erase Page														
	0F80	One ISP Erase Page														
	1000	One ISP Erase Page														
	1080	One ISP Erase Page														
	1100	One ISP Erase Page														
	1180	One ISP Erase Page														
	…	One ISP Erase Page														
	3E00	One ISP Erase Page														
	3E80	One ISP Erase Page														
	3F00	One ISP Erase Page														
	3F80	**This page includes ROM reserved area. We strongly recommend to reserve the area not to do ISP erase.**														

ISP flash ROM erase density is 128-word which limits erase page boundary. The first 128-word of flash ROM (0x0000~0x007F) includes reset vector and interrupt vectors related essential program operation, and the last page 128-word of flash ROM (0x3F80~0x3FFF) includes system reserved ROM area, we strongly recommend do not execute ISP flash ROM erase operation in the two pages. Flash ROM area 0x0080~0x3F7F includes 126-page for ISP flash ROM erase operation.

The first step to do ISP flash ROM erase is to address ROM-page location. The address must be the head location of a page area, e.g. 0x0080, 0x0100, 0x0180…0x3E00, 0x3E80 and 0x3F00. PEROML [7:0] and PEROMH [7:0] define the target starting address [15:0] of flash ROM. Write the start address into PEROML and PEROMH registers, set PECMD register to “0xC3”, and the system start to execute ISP flash ROM erase operation.

➢ **Example : Use ISP flash ROM erase to clear 0x0080~0x00FF contents of flash ROM.**

```
; Set erased start address 0x0080.
        MOV     A, #0x80
        B0MOV   PEROML, A           ; Move low byte address 0x80 to PEROML.
        MOV     A, #0x00
        B0MOV   PEROMH, A           ;Move high byte address 0x00 to PEROMH

; Clear watchdog timer.
        MOV     A,#0X5A
        B0MOV   WDTR,A

; Start to execute ISP flash ROM erase operation.
        MOV     A,#0XC3             ; Start to page erase.
        B0MOV   PECMD, A
        NOP                         ; NOP Delay
        NOP
                                    ; The end of ISP flash ROM erase operation.
```

The two “NOP” instructions make a short delay to let system stable after ISP flash ROM erase operation.

✸ ***Note: Don’t execute ISP flash ROM erase operation for the first page and the last page, or affect program operation.***

13.3 ISP FLASH ROM PROGRAM OPERATION

ISP flash ROM program operation is to write data into flash ROM by program. Program ROM doesn't limit written ROM address and length, but limits 32-word density of one page. The number of ISP flash ROM program operation can be 1-word ~ 32-word at one time, but these words must be in the same page. ISP flash ROM program ROM map is as following:

ISP ROM MAP		ROM address bit0~bit4 (hex)								
		0000	0001	0002	…	000F	0010	…	001E	001F
ROM address bit5~bit15 (hex)	0000	**This page includes reset vector and interrupt sector. We strongly recommend to reserve the area not to do ISP erase.**								
	0020	One ISP Program Page								
	0040	One ISP Program Page								
	0060	One ISP Program Page								
	0080	One ISP Program Page								
	00A0	One ISP Program Page								
	00C0	One ISP Program Page								
	00E0	One ISP Program Page								
	0100	One ISP Program Page								
	0120	One ISP Program Page								
	…	One ISP Program Page								
	1000	One ISP Program Page								
	1020	One ISP Program Page								
	…	One ISP Program Page								
	3F00	One ISP Program Page								
	3F20	One ISP Program Page								
	…	One ISP Program Page								
	3F80	**This page includes ROM reserved area. We strongly recommend to reserve the area not to do ISP erase.**								

ISP flash ROM program page density is 32-word which limits program page boundary. The first 32-word of flash ROM (0x0000~0x001F) includes reset vector and interrupt vectors related essential program operation, and the last page 32-word of flash ROM (0x3F80~0x3FFF) includes system reserved ROM area, we strongly recommend do not execute ISP flash ROM program operation in the two pages. Flash ROM area 0x0020~0x3F7F includes 507-page for ISP flash ROM program operation.

ISP flash ROM program operation is a simple memory mapping operation. The first step is to plan a RAM area to store programmed data and keeps the RAM address for IS RAM addressing. The second step is to plan a ROM area will be programmed from RAM area in ISP flash ROM program operation. The RAM addressing is through PERAML[9:0] 10-bit buffer to configure the start RAM address. The RAM data storage sequence is down-up structure. The first RAM data is the low byte data of the first word of flash ROM. The second RAM data is the high byte data of the first word of ROM, and so on.

ISP programming length is 1-word~32-word. ISP flash ROM programming length is controlled by PERAMCNT[7:3] bits which is 5-bit format. Before ISP ROM programming execution, set the length by program. PEROML [7:0] and PEROMH [7:0] define the target starting address [15:0] of flash ROM. Write the start address into PEROML and PEROMH registers, set PECMD register to "0x5A", and the system start to execute ISP flash ROM program operation. If the programming length is over ISP flash ROM program page boundary, the hardware immediately stops programming flash ROM after finishing programming the last word of the ROM page. So it is very important to plan right ROM address and programming length.

- **Case 1:** 32-word ISP program. RAM buffer length is 64-byte and RAM address is X ~ X+63. PERAMCNT[7:3] =11111b meets a complete one page 32-word of flash ROM. The page address of flash ROM is Y ~ Y+31. The Y is the start address and set to PEROML, PEROMH registers.

RAM (byte)				Flash ROM (word)			
RAM Address 64-byte	bit7 ~ bit0			ROM Address 32-word	bit15 ~ bit8	bit7 ~ bit0	
X	DATA0			Y	DATA1	DATA0	The head of the page. The start address of ISP.
X+1	DATA1			Y+1	DATA3	DATA2	
X+2	DATA2	=>		Y+2	…	…	
X+3	DATA3			Y+3	…	…	
…	…			…	…	…	
X+62	DATA62			…	…	…	
X+63	DATA63			Y+31	DATA63	DATA62	The end of the page. The end address of ISP.

- **Case 2:** 16-word ISP program: RAM buffer length is 32-byte. PERAMCNT [7:3] =01111b meets 16-word of flash ROM. The page address of flash ROM is Y ~ Y+31, but the start address isn't the head of the page. Define the start address is Y+10 and set to PEROML, PEROMH registers. The programmed flash ROM area is Y+10~Y+25 addresses.

RAM (byte)				Flash ROM (word)			
RAM Address 32-byte	bit7 ~ bit0			ROM Address 32-word	bit15 ~ bit8	bit7 ~ bit0	
X	DATA0			Y			The head of the page.
X+1	DATA1			Y+1			
X+2	DATA2	=>		…			
X+3	DATA3			Y+10	DATA1	DATA0	The start address of ISP.
…	…			Y+11	DATA3	DATA2	
X+30	DATA30			…	…	…	
X+31	DATA31			Y+25	DATA31	DATA30	The end address of ISP.
				…			
				Y+30			
				Y+31			The end of the page.

- **Case 3:** Follow above case and change the ROM start address to Y+20. The programmed flash ROM area is Y+20~Y+35 addresses. The ROM range is out of the page boundary. After ISP flash ROM operation, the last 4-word data can't be written into flash ROM successfully. The programming length is over ISP flash ROM program page boundary, the hardware immediately stops programming flash ROM after finishing programming the last word (Y+31) of the ROM page.

RAM (byte)				Flash ROM (word)			
RAM Address 32-byte	bit7 ~ bit0			ROM Address 32-word	bit15 ~ bit8	bit7 ~ bit0	
X	DATA0			Y			The head of the page.
X+1	DATA1			Y+1			
X+2	DATA2	=>		…			
X+3	DATA3			Y+20	DATA1	DATA0	The start address of ISP.
…	…			Y+21	DATA3	DATA2	
X+30	DATA30			…	…	…	
X+31	DATA31			Y+30	DATA21	DATA20	
				Y+31	DATA23	DATA22	The end of the page. The end address of ISP.

➢ **Example : Use ISP flash ROM program to program 32-word data to flash ROM as case 1. Set RAM buffer start address is 0x010. Set flash ROM programmed start address is 0x0020.**

```
; Load data into 64-byte RAM buffer.
        …
        …

; Set RAM start address of 64-byte buffer.
        MOV         A, #0x10
        B0MOV       PERAML, A           ; Set PERAML[7:0] to 0x20.
        MOV         A, #0x00
        B0MOV       PERAMCNT, A         ; Set PERAML[9:8] to 00b.

; Set ISP program length to 32-word.
        MOV         A, #11111000b
        B0MOV       PERAMCNT, A         ; Set PERAMCNT[7:3] to 11111b.

; Set programmed start address of flash ROM to 0x0020..
        MOV         A, #0x20
        B0MOV       PEROML, A           ; Move low byte address 0x20 to PEROML.
        MOV         A, #0x00
        B0MOV       PEROMH, A           ;Move high byte address 0x00 to PEROMH

; Clear watchdog timer.
        MOV         A,#0X5A
        B0MOV       WDTR,A

; Start to execute ISP flash ROM program operation.
        MOV         A,#0X5A             ; Start to program flash ROM.
        B0MOV       PECMD, A
        NOP                             ; NOP Delay
        NOP
                                        ; The end of ISP flash ROM program operation.
```

The two “NOP” instructions make a short delay to let system stable after ISP flash ROM program operation.

✸ ***Note: Don’t execute ISP flash ROM program operation for the first page and the last page, or affect program operation.***

13.4 ISP PROGRAM/ERASE CONTROL REGISTER

0CCH	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	-	-	-	-	-	-	-	-

Bit [7:0] **PECMD [7:0]:** ISP operation control register.
0x5A: Page Program (32 words / page).
0xC3: Page Erase (128 words / page).
Others: Reserved.

✸ ***Note: Before executing ISP program and erase operations, clear PECMD register is necessary. After ISP configuration, set ISP operation code in "MOV A,I" and "B0MOV M,A" instructions to start ISP operations.***

13.5 ISP ROM ADDRESS REGISTER

ISP ROM address length is 16-bit and separated into PEROML and PEROMH registers. Before ISP execution, set the head address of ISP ROM by program.

0CDH	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]:** The low byte buffer of ISP ROM address.

0CEH	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]:** The high byte buffer of ISP ROM address.

13.6 ISP RAM ADDRESS REGISTER

ISP RAM address length is 10-bit and separated into PERAML register and PERAMCNT[1:0] bits. Before ISP execution, set the head address of ISP RAM by program.

0CFH	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

Bit [7:0] **PEROML[7:0]:** ISP RAM address [7:0].

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAMCNT	PERAMCNT7	PERAMCNT6	PERAMCNT5	PERAMCNT4	PERAMCNT3	-	PERAML9	PERAML8
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 [1:0] **PEROMCNT[1:0]:** ISP RAM address [9:8].

13.7 ISP ROM PROGRAMMING LENGTH REGISTER

ISP programming length is 1-word ~ 32-word. ISP ROM programming length is controlled by PEROMCNT[7:3] bits which is 5-bit format. Before ISP ROM programming execution, set the length by program.

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAMCNT	PERAMCNT7	PERAMCNT6	PERAMCNT5	PERAMCNT4	PERAMCNT3	-	PERAML9	PERAML8
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 [7:3] **PEROMCNT[7:3]:** ISP ROM programming length control register.

ISP programming length = PEROMCNT[7:3] + 1

PEROMCNT[7:3]=0: ISP programming length is 1-word.
PEROMCNT[7:3]=1: ISP programming length is 2-word.
…
…
PEROMCNT[7:3]=30: ISP programming length is 31-word.
PEROMCNT[7:3]=31: ISP programming length is 32-word.

✸ ***Note: 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 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
	DAA		To adjust ACC’s data format from HEX to DEC.	√	-	-	1
	MUL	A,M	R, A ← A * M, The LB of product stored in Acc and HB stored in R register. ZF affected by Acc.	-	-	√	2
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
	COM	M	A ← M (1’s complement).	-	-	√	1
	COMM	M	M ← M (1’s complement).	-	-	√	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
	INC	M	A ← M + 1.	-	-	√	1
	INCM	M	M ← M + 1.	-	-	√	1+N
	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
	DEC	M	A ← M – 1.	-	-	√	1
	DECM	M	M ← M – 1.	-	-	√	1+N
	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
	TS0M	M	If M = 0, Z = 1. Else Z = 0.	-	-	√	1
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALLHL		Stack ← PC15~PC0, PC15~PC8 ← H register, PC7~PC0 ← L register	-	-	-	2

	CALLYZ	Stack ← PC15~PC0, PC15~PC8 ← Y register, PC7~PC0 ← Z register	-	-	-	2
M	RET	PC ← Stack	-	-	-	2
I	RETI	PC ← Stack, and to enable global interrupt	-	-	-	2
S	RETLW I	PC ← Stack, and load I to ACC.	-	-	√	2
C	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".

15 ELECTRICAL CHARACTERISTIC

15.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)
SN8F27E93L, SN8F27E94L..... - 0.3V ~ 3.0V
SN8F27E93, SN8F27E94..... - 0.3V ~ 5.5V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8F27E93L, SN8F27E94L..... 0°C ~ + 70°C
SN8F27E93, SN8F27E94..... 0°C ~ + 70°C
Storage ambient temperature (Tstor) ..... –30°C ~ + 125°C

15.2 ELECTRICAL CHARACTERISTIC

- **DC CHARACTERISTIC**

(All of voltages refer to Vss, Vdd = 2.5V, Fosc = 16MHz, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage (SN8F27E93L, SN8F27E94L)	Vdd1	*Normal mode.*		2.3	2.5	3.0	V
Operating voltage (SN8F27E93, SN8F27E94)	Vdd1	*Normal mode.*		2.3	-	5.5	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset.		0.05	-	-	V/ms
Input Low Voltage	ViL	All input ports, Reset pin, XIN/XOUT pins.		Vss	-	0.3*Vdd	V
Input High Voltage	ViH	All input ports, Reset pin, XIN/XOUT pins.		0.7*Vdd	-	Vdd	V
Input Voltage	Vin1	I/O port's input voltage range.		-0.5	-	Vdd+0.2	V
Output Voltage	Voh1	I/O output ports.		0	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd.		-	-	2	uA
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd.		-	-	2	uA
I/O port pull-up resistor	Rup1	Vin = Vss, Vdd = 2.5V.		150	250	350	KΩ
I/O output source current	IoH1	Vop = Vdd – 0.1*Vdd.		3	6	-	mA
I/O sink current	IoL1	Vop = Vss + 0.1*Vdd.		5	10	-	
Supply Current (SN8F27E93L, SN8F27E94L)	Idd1	Normal Mode (No loading, Fcpu = Fosc/1)	Vdd= 2.5V, Fcpu = 16MHz	-	10	15*	mA
	Idd3	Sleep Mode	Vdd= 2.5V	-	10	15	uA
Supply Current (SN8F27E93, SN8F27E94)	Idd1	Normal Mode (No loading, Fcpu = Fosc/1)	Vdd= 2.5V, Fcpu = 16MHz	-	10	15*	mA
	Idd3	Sleep Mode	Vdd= 2.5V	-	100	150	uA

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

- **ADC CHARACTERISTIC**

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
AIN0 ~ AIN5 input voltage	Vani	Vdd = 5.0V	0	-	Vdd	V
ADC reference Voltage	Vref		-	Vdd	-	V
*ADC enable time	Tast	Ready to start convert after set ADENB = "1"	100	-	-	us
*ADC current consumption	I_{ADC}	Vdd=5.0V	-	0.6	-	mA
		Vdd=3.0V	-	0.4	-	mA
ADC Clock Frequency	F_{ADCLK}	VDD=5.0V	-	-	8M	Hz
		VDD=3.0V	-	-	5M	Hz
ADC Conversion Cycle Time	F_{ADCYL}	VDD=2.4V~5.5V	64	-	-	$1/F_{ADCLK}$
ADC Sampling Rate (Set FADS=1 Frequency)	F_{ADSMP}	VDD=5.0V	-	-	125	K/sec
		VDD=3.0V	-	-	80	K/sec
Differential Nonlinearity	DNL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	-1	-	+1	LSB
Integral Nonlinearity	INL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	-1	-	+1	LSB
No Missing Code	NMC	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	9	-	11	Bits
ADC offset Voltage	$V_{ADCoffset}$	Non-trimmed	-10	0	+10	mV
		Trimmed	-2	0	+2	mV

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

- **FLASH MEMORY CHARACTERISTIC**

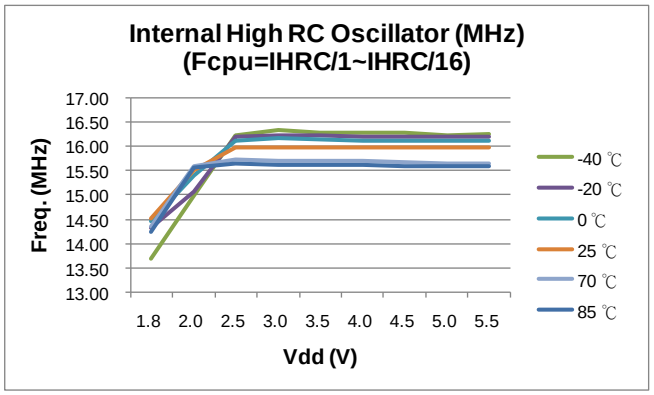
(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

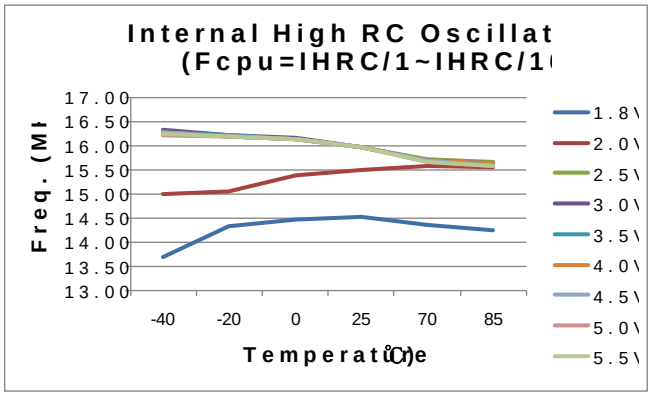
PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Supply Voltage	Vdd1	Read mode	1.8		Vdd	V
		Erase/Program	2.5		Vdd	V
Endurance time	Ten	Erase + Program, *0℃~70℃*	20K	-	-	Cycle
Page erase current	Ier	Vdd1=2.5V	-	2.5	5	mA
Program current	Ipg	Vdd1=2.5V	-	3.5	7	mA
Page erase time	Ter	Vdd = 2.5V, 1-page (128-word).	-	-	30	ms
Program time	Tpg1	Vdd = 2.5V, ISP setup time.	-	-	380	us
	Tpg2	Vdd = 2.5V, 1-word program.	-	-	30	us

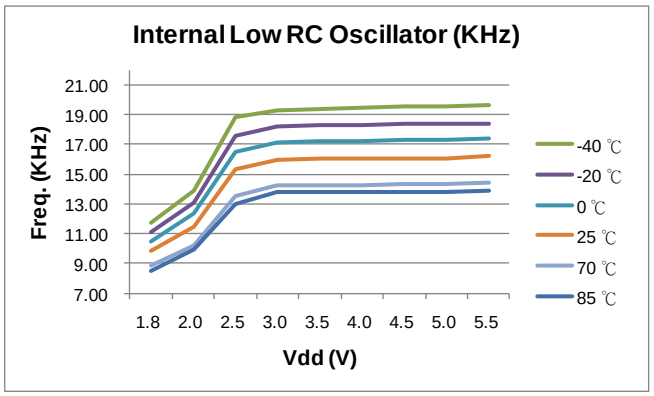
" * " These parameters are for design reference, not tested.

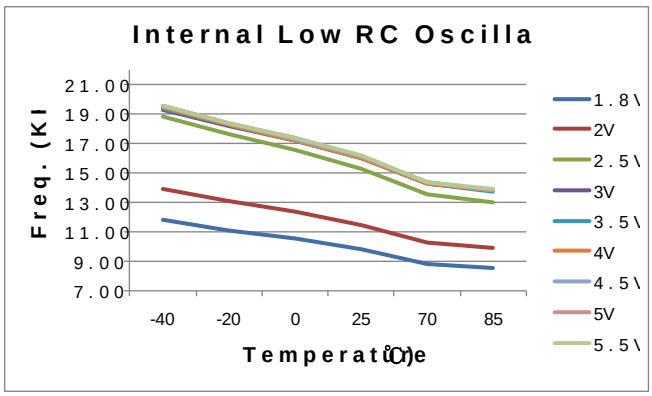
15.3 CHARACTERISTIC GRAPHS

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





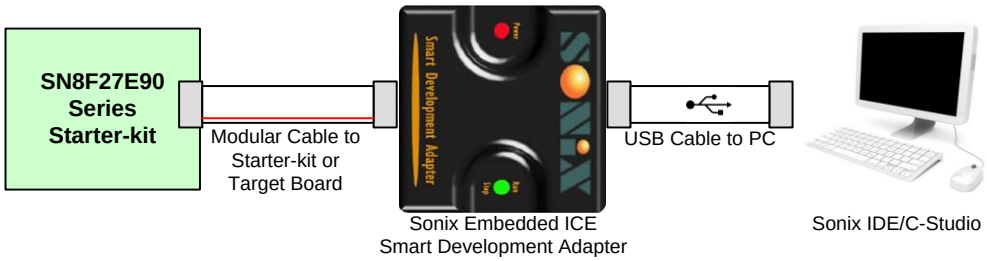




16 DEVELOPMENT TOOL

SONIX provides an Embedded ICE emulator system to offer SN8F27E90 series firmware development. The platform is a in-circuit debugger and controlled by SONIX M2IDE software on Microsoft Windows platform. The platform includes Smart Development Adapter, SN8F27E90 series Starter-kit and M2IDE software to build a high-speed, low cost, powerful and multi-task development environment including emulator, debugger and programmer. To execute emulation is like run real chip because the emulator circuit integrated in SN8F27E90 series to offer a real development environment.

SN8F27E90 Series Embedded ICE Emulator System:



SN8F27E90 Series Embedded ICE Emulator includes:

- Smart Development Adapter.
- USB cable to provide communications between the Smart Development Adapter and a PC.
- SN8F27E90 Series Starter-Kit.
- Modular cable to connect the Smart Development Adapter and SN8F27E90 Series Starter-Kit or target board.
- CD-ROM with M2IDE software (M2IDE V124 or greater).

SN8F27E90 Series Embedded ICE Emulator Feature:

- Target's Operating Voltage: 2.3V~5.5V.
- Up to 6 hardware break points.
- System clock rate up to 16MHz (Fcpu=16mips).
- Oscillator supports internal high speed RC, internal low speed RC, external crystal/resonator and external RC.

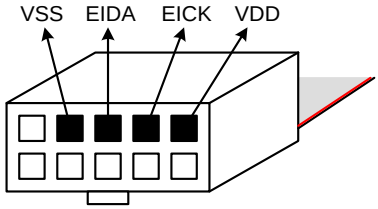
SN8F27E90 Series Embedded ICE Emulator Limitation:

- EIDA and EICK pins are shared with GPIO pins. In embedded ICE mode, the shared GPIO function can't work. We strongly recommend planning the two pins as simple function which can be verified without debugger platform.

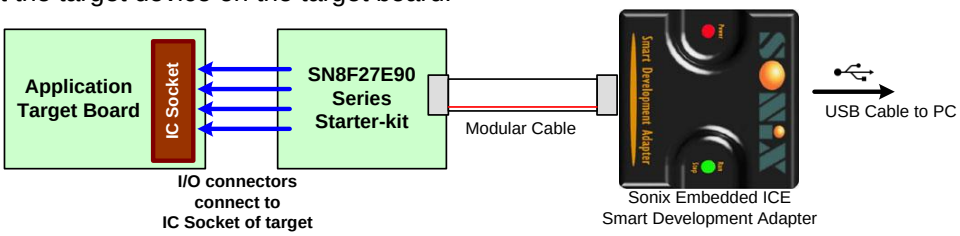
16.1 SMART DEVELOPMENT ADAPTER

Smart Development Adapter is a high speed emulator for Sonix Embedded ICE type flash MCU. It debugs and programs Sonix flash MCU and transfers MCU's system status, RAM data and system register between M2IDE and Sonix flash MCU through USB interface. The other terminal connected to SN8F27E90 Series Starter-kit or Target board is a 4-wire serial interface. In addition to debugger functions, the Smart Starter-Kit system also may be used as a programmer to load firmware from PC to MCU for engineering production, even mass production.

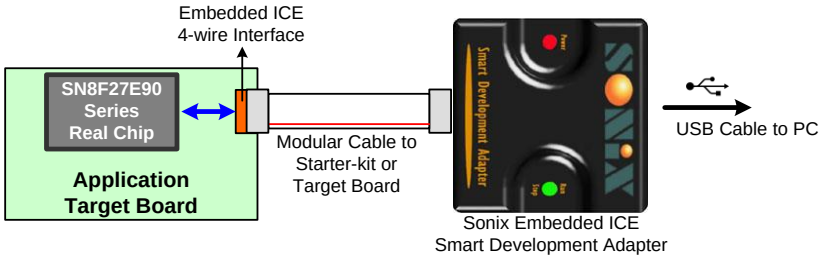
Smart Development Adapter communication with SN8F27E90 series flash MCU is through a 4-wire bus. The pin definition of the Modular cable is as following:



The modular cable can be inserted into SN8F27E90 Series Starter-Kit plugged into the target board or inserted into a matching socket at the target device on the target board.



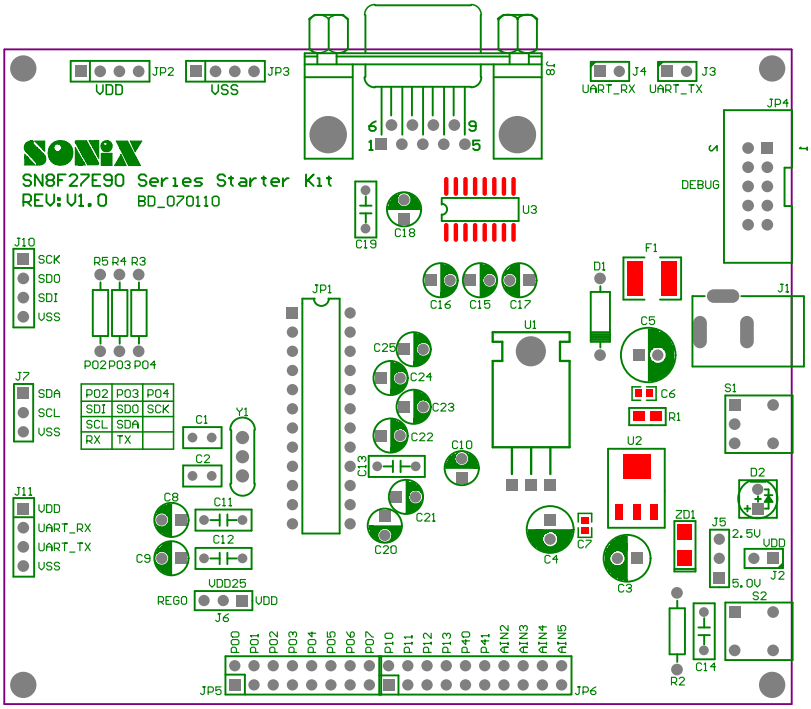
If the target board of application is designed and ready, the modular cable can be inserted into the target directly to replace SN8F27E90 Series Starter-Kit. Design the 4-wire interface connected with SN8F27E90 Series IC to build a real application environment. In the mode, set SN8F27E90 Series IC on the target is necessary, or the emulation would be error without MCU.



EIDA and EICK share with P1.0/P1.1 GPIO. In emulation mode, EIDA and EICK are Embedded ICE interface and not execute GPIO functions. The P1.0/P1.1 GPIO status still display on M2IDE window to simulate P1.0/P1.1 program execution.

16.2 SN8F27E90 SERIES STARTER-KIT

SN8F27E90 Series Starter-kit is an easy-development platform. It includes SN8F27E90 series real chip and I/O connectors to input signal or drive extra device of user's application. It is a simple platform to develop application as target board not ready. The starter-kit can be replaced by target board, because SN8F27E90 series integrates embedded ICE in-circuit debugger circuitry. SN8F27E90 Series Starter-Kit Outline is as below:

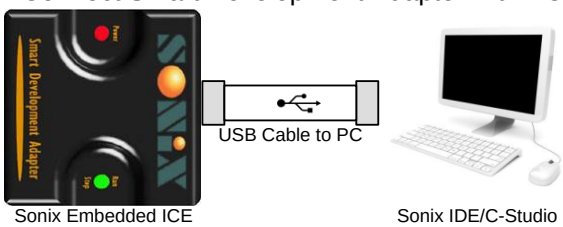


- S1: Target power switch.
- VDD/VSS connector: Target power connector. SN8F27E93: 2.3V~5.5V. SN8F27E93L: 2.3V~3.0V.
- J2: VDD power from External Power source or Internal Regulator Power.

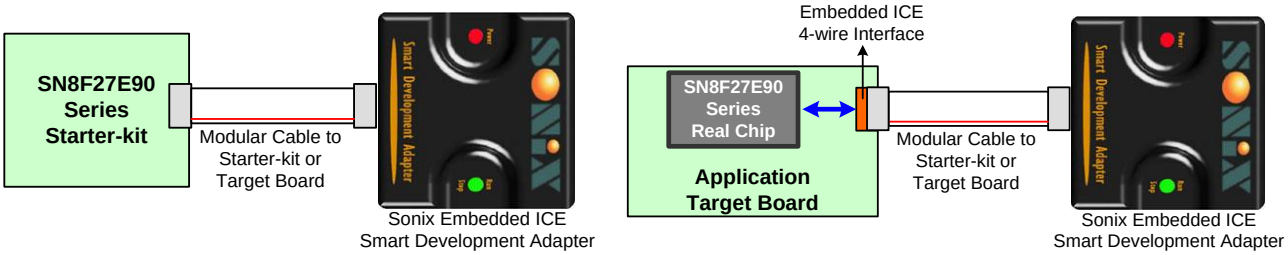
- J5: Internal Regulator Power from 2.5V regulator or 5.0V regulator.
- JP1: SN8F27E90 series real chip (Sonix standard option).
- VDD25 of J6: Switch it to REGO position for SN8F27E93 real chip. If the JP1 is SN8F27E93L, not SN8F27E93, please switch VDD25 to VDD position.
- Reset key (S2): External reset trigger source.
- I/O connector (JP5, JP6): SN8F27E90 series I/O pins connected to target board.
- J7: MSP connector.
- J10: SIO connector.
- J11: UART connector.
- J8: COM port and connected to SN8F27E90 series UART pins.
- J3: SN8F27E90 series UART_Tx pin connected to COM port.
- J4: SN8F27E90 series UART_Rx pin connected to COM port.
- JP4: SN8F27E90 series Embedded IC pins and connected to Smart Development Adapter.
- R3: P04/SCK pull-up resister.
- R4: P03/SDO/SDA/UART_Tx pull-up resister.
- R5: P02/SDI/SCL/UART_Rx pull-up resister.
- Y1, C1, C2: External crystal/resonator oscillator components.
- C20: P4.0 ADC capacitor.
- C21: P4.1 ADC capacitor.
- C22: AIN2 ADC capacitor.
- C23: AIN3 ADC capacitor.
- C24: AIN4 ADC capacitor.
- C25: AIN5 ADC capacitor.

16.3 EMULATOR/DEBUGGER INSTALLATION

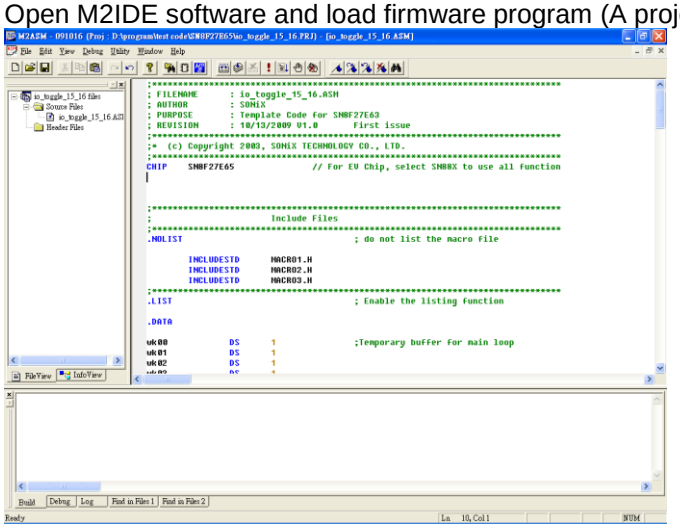
- Install the M2IDE Software (V124 or greater).
- Connect Smart Development Adapter with PC plugging in USB cable.



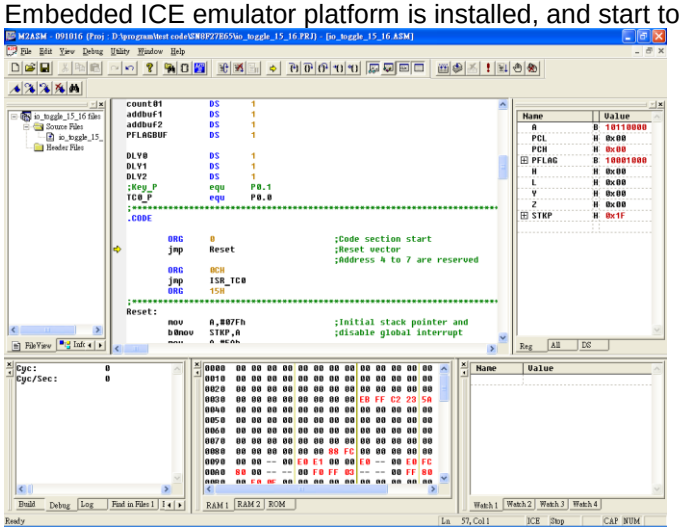
- Attach the modular cable between Smart Development Adapter and SN8F27E90 Series Starter-kit or target.



- Connect the power supplier to SN8F27E90 Series Starter-kit or target, and turn off the power.
- Open M2IDE software and load firmware program (A project or a ".ASM" file).

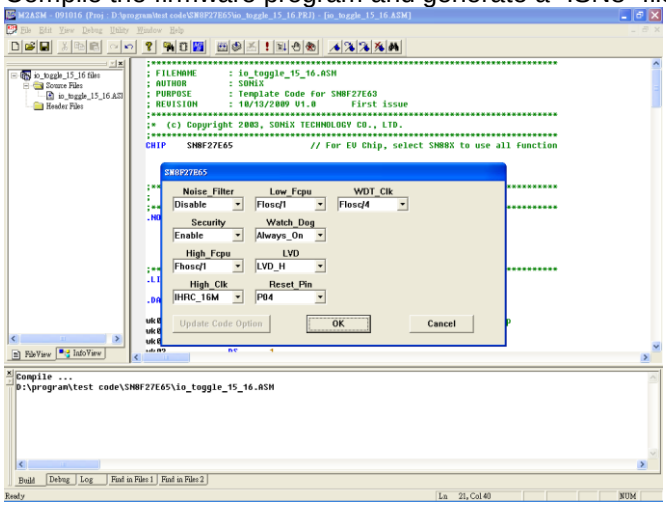


- Turn on the power switch of SN8F27E90 Series Starter-kit or target.
- Embedded ICE emulator platform is installed, and start to execute debugger.

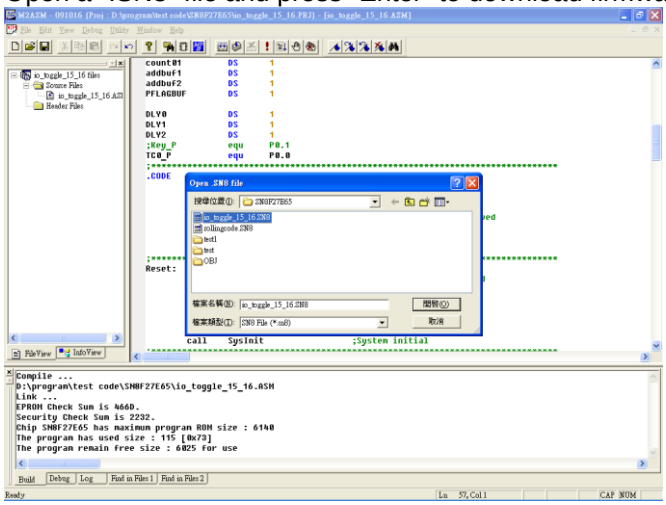


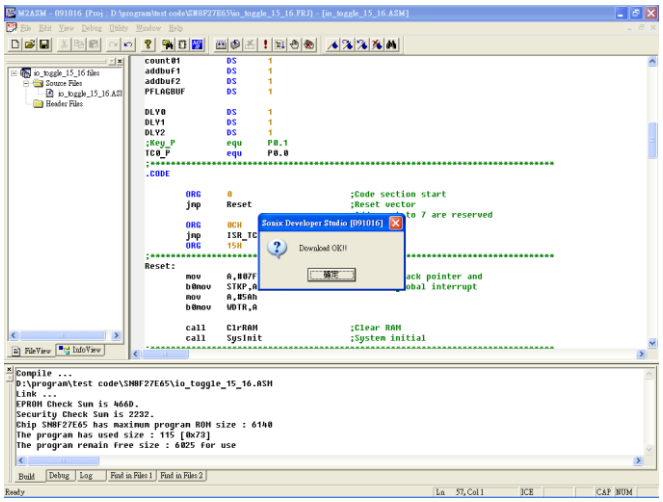
16.4 PROGRAMMER INSTALLATION

- Setup emulator/debugger environment first.
- Compile the firmware program and generate a ".SN8" file.



- Execute download (F8) function of M2IDE.
- Open a ".SN8" file and press "Enter" to download firmware to SN8F27E90 Series Starter-kit or target.





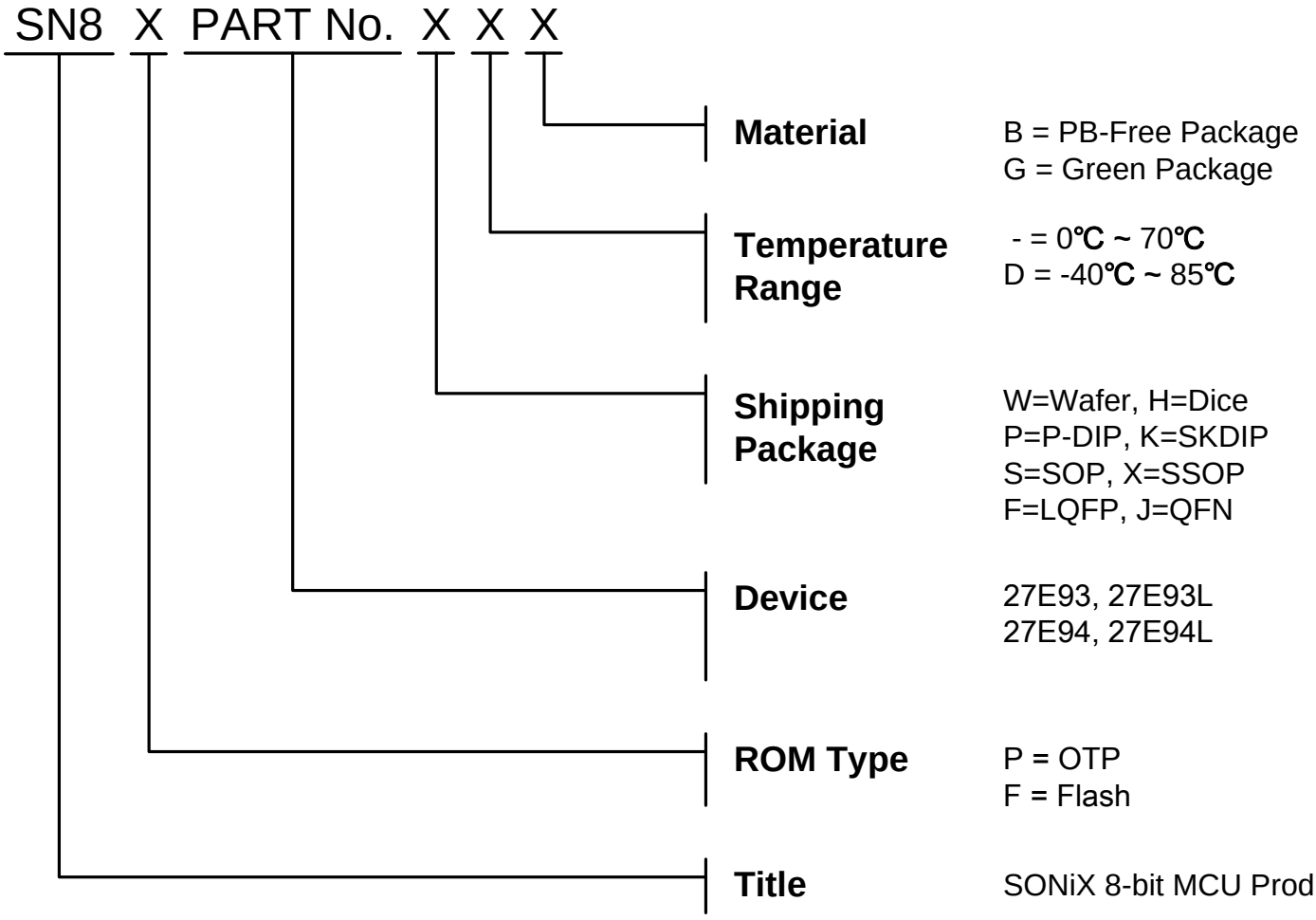
- Turn off the power of SN8F27E90 Series Starter-kit or target.
- Disconnect SN8F27E90 Series Starter-kit or target from Smart Development Adapter.
- Turn on the power of SN8F27E90 Series Starter-kit or target, and MCU works independently.

17 Marking Definition

17.1 INTRODUCTION

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

17.2 MARKING INDETIFICATION SYSTEM



17.3 MARKING EXAMPLE

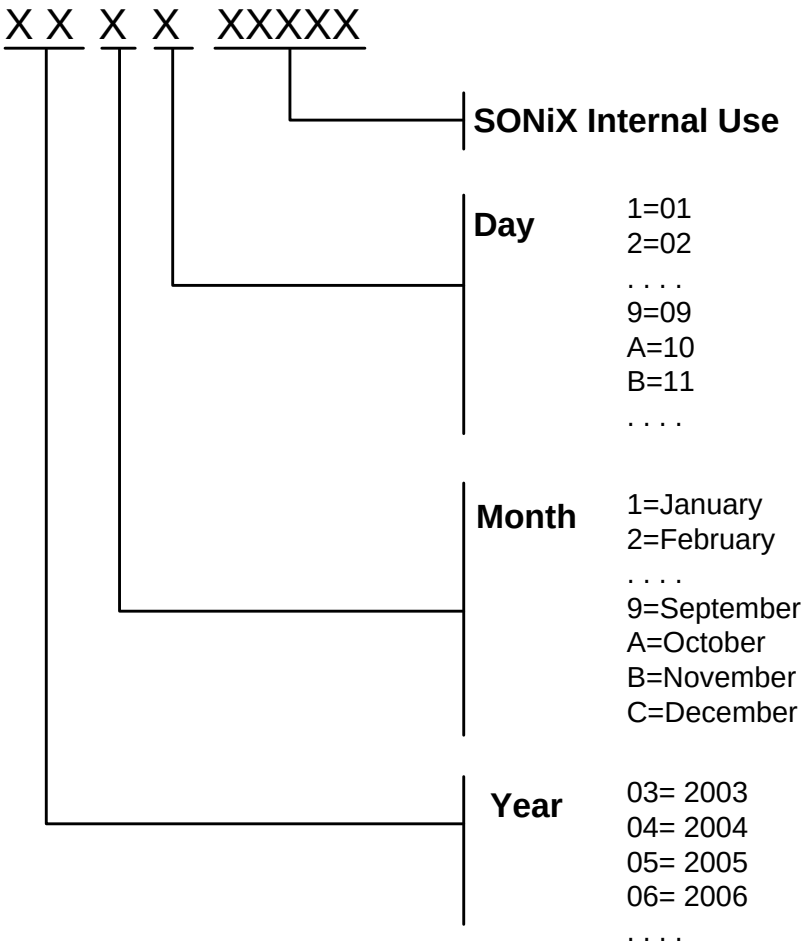
- **Wafer, Dice:**

Name	ROM Type	Device	Package	Temperature	Material
SN8F27E93W	FLASH	27E93	Wafer	0℃~70℃	-
SN8F27E93H	FLASH	27E93	Dice	0℃~70℃	-
SN8F27E94W	FLASH	27E93	Wafer	0℃~70℃	-
SN8F27E94H	FLASH	27E93	Dice	0℃~70℃	-

- **Green Package:**

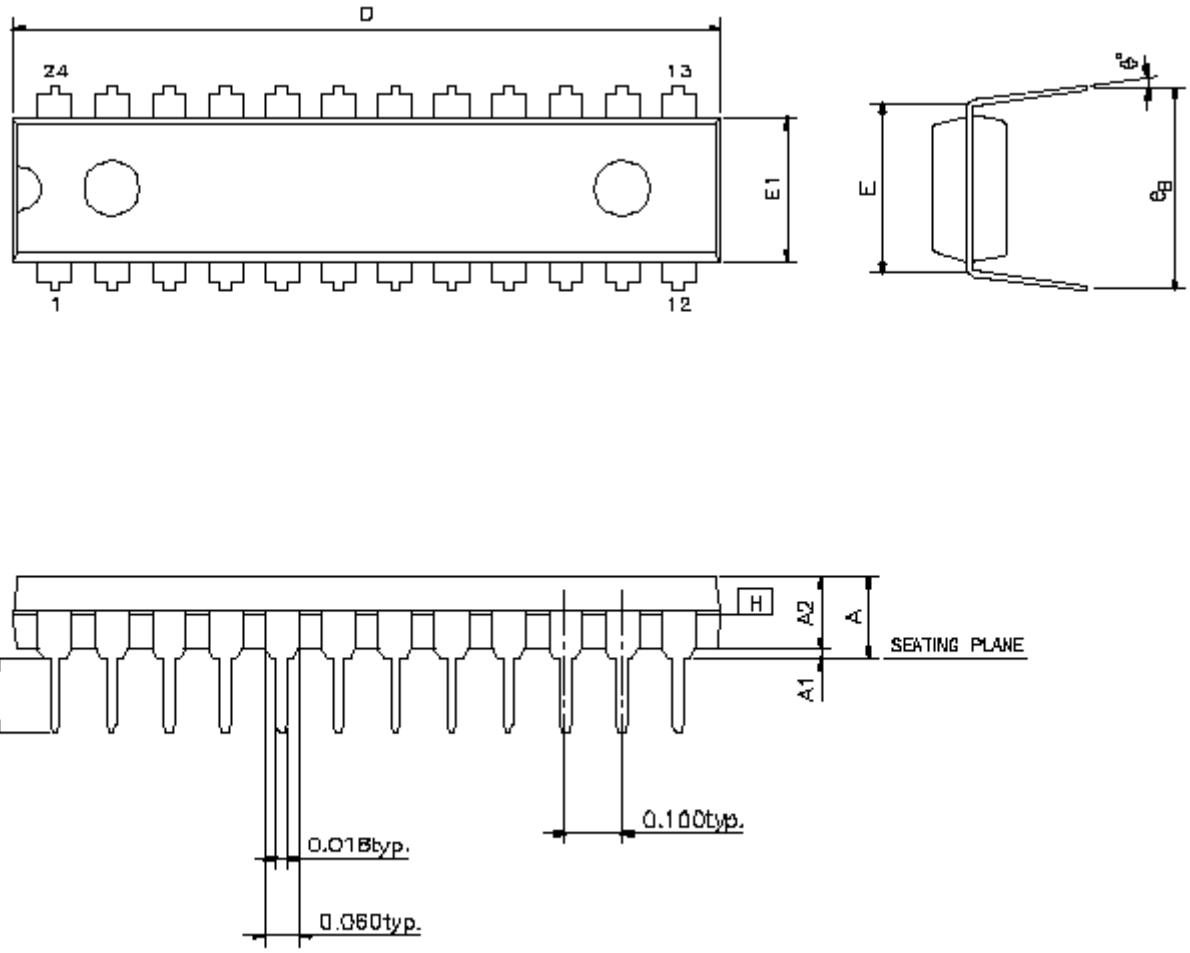
Name	ROM Type	Device	Package	Temperature	Material
8F27E93J	FLASH	27E93	QFN	0℃~70℃	Green Package
SN8F27E93KG	FLASH	27E93	SK-DIP	0℃~70℃	Green Package
SN8F27E93SG	FLASH	27E93	SOP	0℃~70℃	Green Package
SN8F27E93XG	FLASH	27E93	SSOP	0℃~70℃	Green Package
SN8F27E94KG	FLASH	27E93	SK-DIP	0℃~70℃	Green Package
8F27E93LJ	FLASH	27E93	QFN	0℃~70℃	Green Package
SN8F27E93LKG	FLASH	27E93	SK-DIP	0℃~70℃	Green Package
SN8F27E93LSG	FLASH	27E93	SOP	0℃~70℃	Green Package
SN8F27E93LXG	FLASH	27E93	SSOP	0℃~70℃	Green Package
SN8F27E94LKG	FLASH	27E93	SK-DIP	0℃~70℃	Green Package

17.4 DATECODE SYSTEM



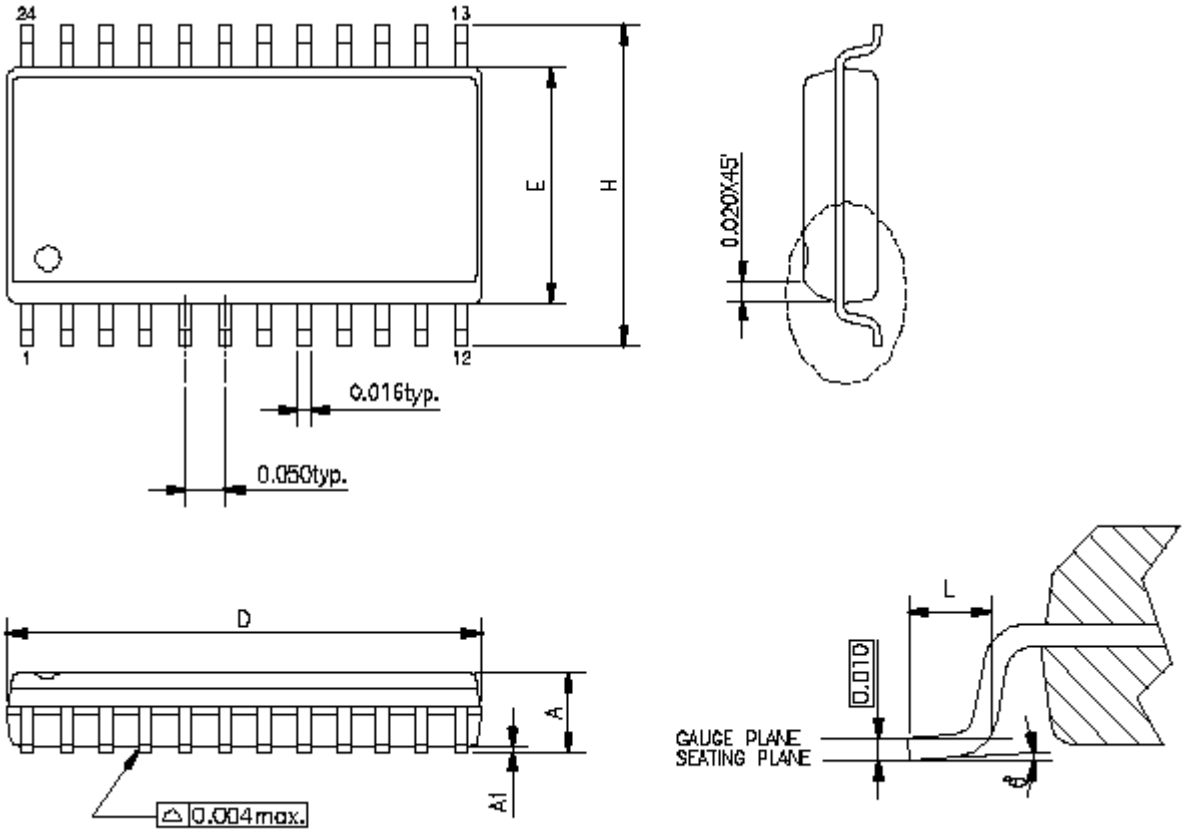
18 PACKAGE INFORMATION

18.1 SK-DIP 24 PIN



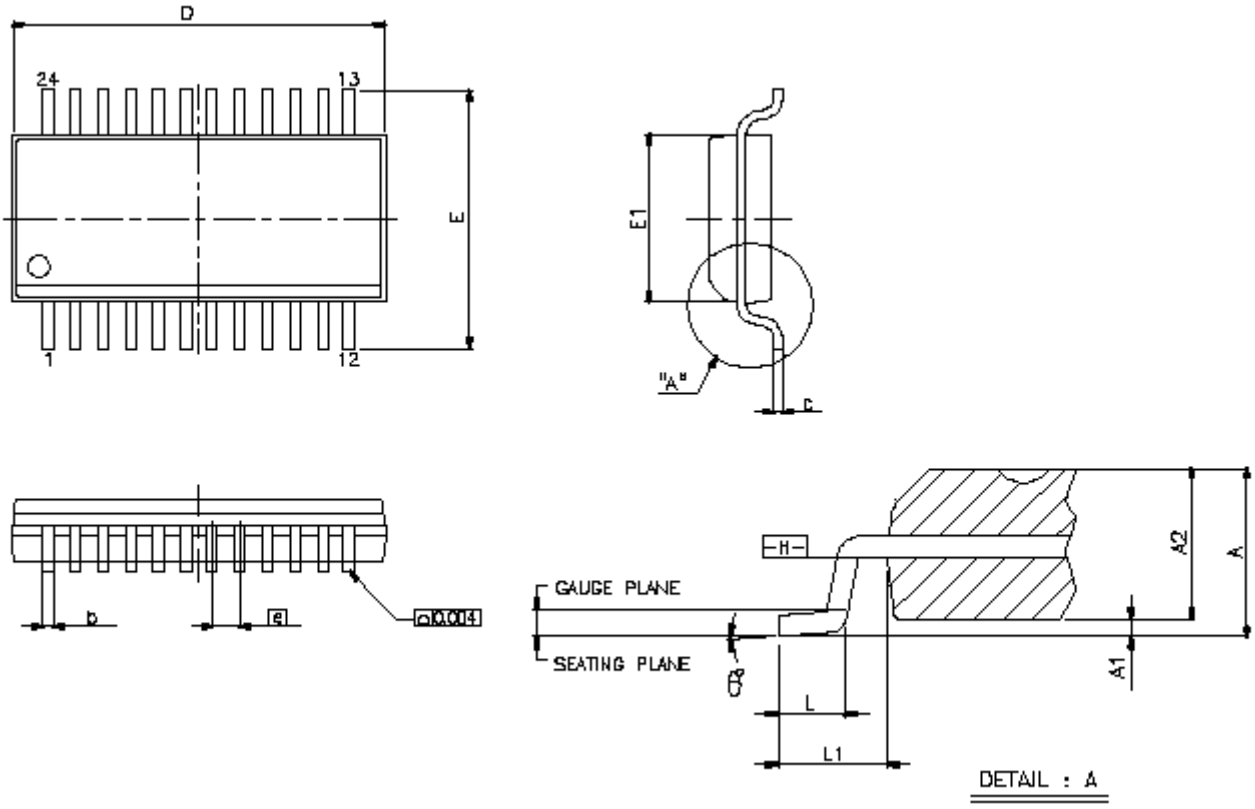
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	1.230	1.250	1.280	31.242	31.75	32.51
E	0.300 BSC			7.62 BSC		
E1	0.252	0.258	0.263	6.4	6.553	5.994
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017	9,525
θ°	0°	7°	15°	0°	7°	15°

18.2 SOP 24 PIN



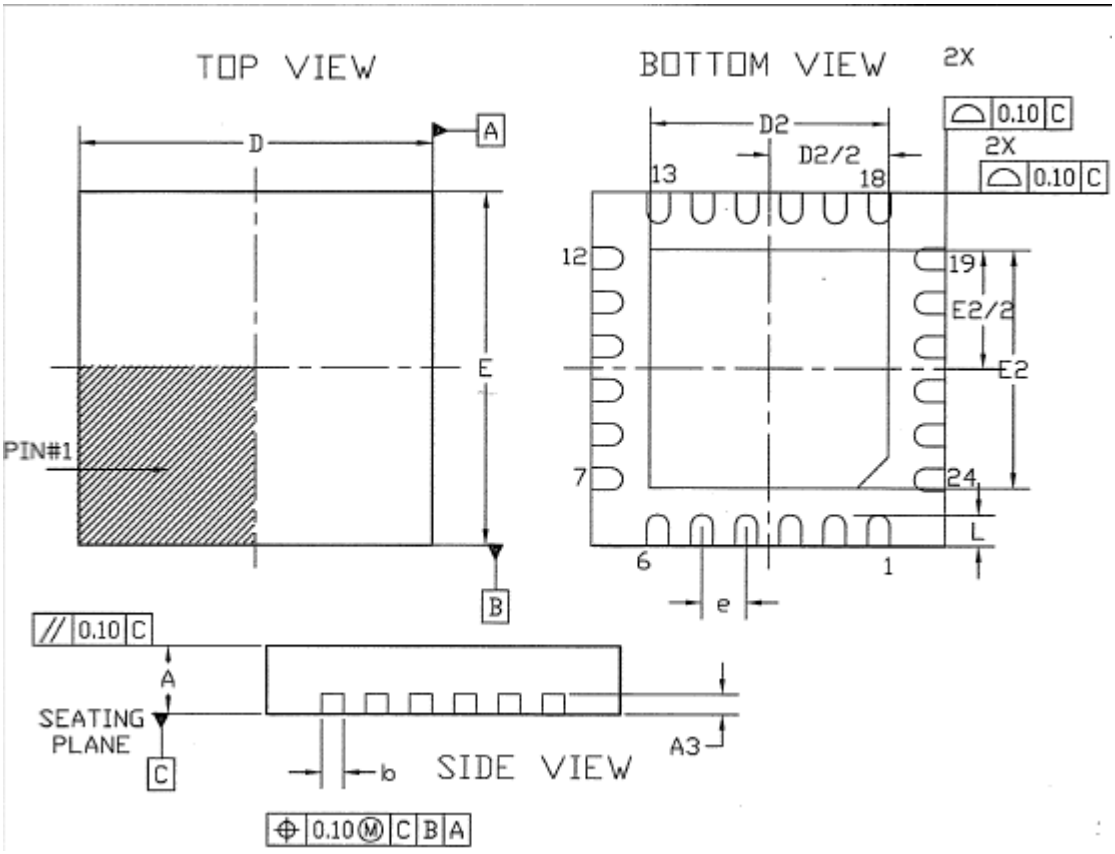
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.104	-	-	2.642
A1	0.004	-	-	0.102	-	-
D	0.599	0.600	0.624	15.214	15.24	15.84
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.337	10.643
L	0.016	0.035	0.050	0.406	0.889	1.270
θ°	0°	4°	8°	0°	4°	8°

18.3 SSOP 24 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.053	0.064	0.069	1.346	1.625	1.752
A1	0.004	0.006	0.010	0.101	0.152	0.254
A2	-	-	0.059	-	-	1.499
D	0.337	0.341	0.344	8.559	8.661	8.737
E	0.228	0.236	0.244	5.791	5.994	6.197
E1	0.150	0.154	0.157	3.81	3.911	3.987
b	0.008	-	0.012	0.203	-	0.304
C	0.007	-	0.010	0.177	-	0.254
[e]	0.025 BASIC			0.635 BASIC		
L	0.016	0.025	0.050	0.406	0.635	1.27
L1	0.041 BASIC			1.041 BASIC		
θ°	0°	-	8°	0°	-	8°

18.4 QFN 4X4 24 PIN



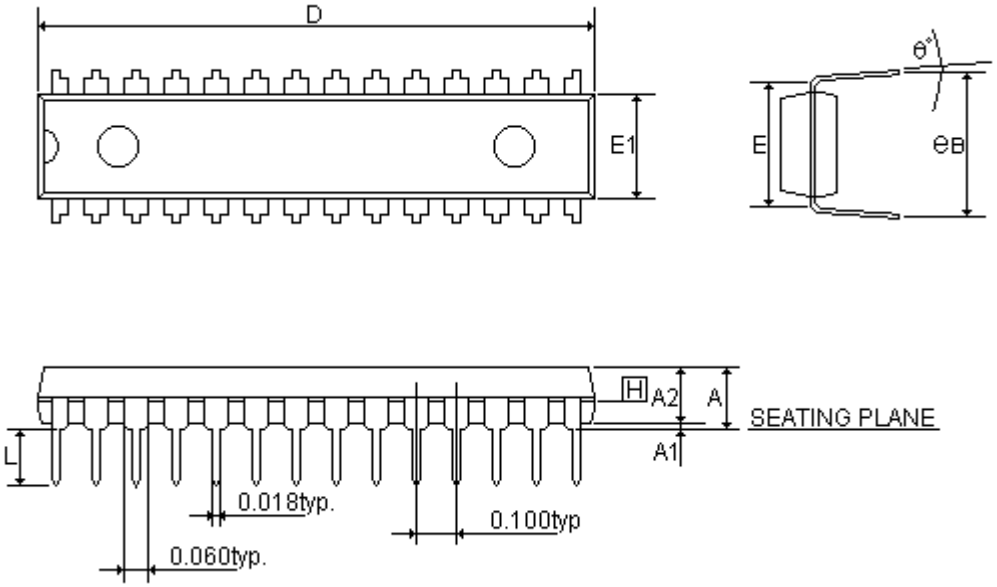
SYMBOL	COMMON					
	DIMENSIONS MILLIMETER			DIMENSIONS INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	SEE VARIATION					
A3	0.195	0.203	0.211	0.0077	0.008	0.0083
b	0.180	0.230	0.300	0.007	0.009	0.012
D	3.925	4.0	4.075	0.154	0.157	0.160
E	3.925	4.0	4.075	0.154	0.157	0.160
e	0.50 BSC			0.020 BSC		
L	SEE VARIATION					

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

18.5 SK-DIP 28 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.114	0.130	0.135	2.896	3.302	3.429
D	1.390	1.390	1.400	35.306	35.306	35.560
E		0.310			7.874	
E1	0.283	0.288	0.293	7.188	7.315	7.442
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.330	0.350	0.370	8.382	8.890	9.398
θ°	0°	7°	15°	0°	7°	15°

Main Office:
Address: 10F-1, NO.36, Taiyuan Street, Chupei City, Hsinchu, Taiwan R.O.C.
Tel: 886-3-560 0888
Fax: 886-3-560 0889

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 1519, Chevalier Commercial Centre, NO.8 Wang Hoi Road, Kowloon Bay, Hong Kong.
Tel: 852-2723-8086
Fax: 852-2723-9179

Technical Support by Email:
Sn8fae@sonix.com.tw