SN8P275X Series

USER'S MANUAL

Version 0.9

SN8P2754
SN8P2755
SN8P2758

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 0.1	Nov. 2008	1. Preliminary Version first issue
VER 0.2	Nov. 2008	1. Modify Package information.
VER 0.3	Nov. 2008	1. Modify ICE MSP emulation.
VER 0.4	Feb. 2008	1. Modify MSP,Package information.
VER 0.5	Apr. 2009	1. Modify SOP32 Package size.
VER 0.6	Jun. 2009	1. Update code option table.
VER 0.7	Nov. 2009	1. Modify LQFP48 marking name 2. Add AVDD pin descriptment 3. Modify DAO as 7bit DAC output in 1.4 Pin Description 4. Fix typing error
VER 0.8	May.2011	1. Remove P33.
VER 0.9	Dec.2011	1. Add QFN32 Package.

Table of Content

AMENDENT HISTORY 2

1 PRODUCT OVERVIEW 9

1.1 FEATURES 9
1.2 SYSTEM BLOCK DIAGRAM 10
1.3 PIN ASSIGNMENT 11
1.4 PIN DESCRIPTIONS 14
1.5 PIN CIRCUIT DIAGRAMS 15

2 CENTRAL PROCESSOR UNIT (CPU) 16

2.1 MEMORY MAP 16
2.1.1 PROGRAM MEMORY (ROM) 16
2.1.2 RESET VECTOR (0000H) 17
2.1.3 INTERRUPT VECTOR (0008H) 18
2.1.3.1 LOOK-UP TABLE DESCRIPTION 19
2.1.4 JUMP TABLE DESCRIPTION 22
2.1.4.1 CHECKSUM CALCULATION 24
2.1.5 CODE OPTION TABLE 25
2.1.6 DATA MEMORY (RAM) 26
2.1.7 SYSTEM REGISTER 27
2.1.7.1 SYSTEM REGISTER TABLE 27
2.1.7.2 SYSTEM REGISTER DESCRIPTION 27
2.1.7.3 BIT DEFINITION of SYSTEM REGISTER 28
2.1.7.4 ACCUMULATOR 32
2.1.8 PROGRAM FLAG 33
2.1.8.1 PROGRAM COUNTER 34
2.1.9 H, L REGISTERS 36
2.1.10 Y, Z REGISTERS 38
2.1.10.1 X REGISTERS 38
2.1.11 R REGISTERS 39
2.2 ADDRESSING MODE 40
2.2.1 IMMEDIATE ADDRESSING MODE 40
2.2.2 DIRECTLY ADDRESSING MODE 40
2.2.3 INDIRECTLY ADDRESSING MODE 40
2.3 STACK OPERATION 41
2.3.1 OVERVIEW 41
2.3.2 STACK REGISTERS 42

2.3.3 STACK OPERATION EXAMPLE 43

3 RESET 44

3.1 OVERVIEW 44
3.2 POWER ON RESET 45
3.3 WATCHDOG RESET 45
3.4 BROWN OUT RESET 46
3.4.1 BROWN OUT DESCRIPTION 46
3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION 47
3.4.3 BROWN OUT RESET IMPROVEMENT 47
3.5 EXTERNAL RESET 49
3.6 EXTERNAL RESET CIRCUIT 49
3.6.1 Simply RC Reset Circuit 49
3.6.2 Diode & RC Reset Circuit 50
3.6.3 Zener Diode Reset Circuit 50
3.6.4 Voltage Bias Reset Circuit 51
3.6.5 External Reset IC 52

4 SYSTEM CLOCK 53

4.1 OVERVIEW 53
4.2 CLOCK BLOCK DIAGRAM 53
4.3 OSCM REGISTER 54
4.4 SYSTEM HIGH CLOCK 55
4.4.1 INTERNAL HIGH RC 55
4.4.2 EXTERNAL HIGH CLOCK 55
4.4.2.1 CRYSTAL/CERAMIC 56
4.4.2.2 RC 56
4.4.2.3 EXTERNAL CLOCK SIGNAL 57
4.5 SYSTEM LOW CLOCK 58
4.5.1 SYSTEM CLOCK MEASUREMENT 59

5 SYSTEM OPERATION MODE 60

5.1 OVERVIEW 60
5.2 SYSTEM MODE SWITCHING 61
5.3 WAKEUP 63
5.3.1 OVERVIEW 63
5.3.2 WAKEUP TIME 63
5.3.3 P1W WAKEUP CONTROL REGISTER 64

6 INTERRUPT 65

6.1 OVERVIEW 65

6.2 INTEN INTERRUPT ENABLE REGISTER 66
6.3 INTRQ INTERRUPT REQUEST REGISTER 67
6.4 GIE GLOBAL INTERRUPT OPERATION 68
6.5 PUSH, POP ROUTINE 68
6.6 EXTERNAL INTERRUPT OPERATION (INT0~INT2) 70
6.7 T0 INTERRUPT OPERATION 71
6.8 TC0 INTERRUPT OPERATION 72
6.9 TC1 INTERRUPT OPERATION 73
6.10 SIO INTERRUPT OPERATION 74
6.11 ADC INTERRUPT OPERATION 75
6.12 MULTI-INTERRUPT OPERATION 76

7 I/O PORT 78

7.1 I/O PORT MODE 78
7.2 I/O PULL UP REGISTER 80
7.3 I/O PORT DATA REGISTER 81
7.4 I/O OPEN-DRAIN REGISTER 82
7.5 PORT 4 ADC SHARE PIN 83

8 TIMERS 85

8.1 WATCHDOG TIMER 85
8.2 TIMER 0 (T0) 87
8.2.1 OVERVIEW 87
8.2.2 T0M MODE REGISTER 88
8.2.3 T0C COUNTING REGISTER 89
8.2.4 T0 TIMER OPERATION SEQUENCE 90
8.3 TIMER/COUNTER 0 (TC0) 91
8.3.1 OVERVIEW 91
8.3.2 TC0M MODE REGISTER 92
8.3.3 TC0C COUNTING REGISTER 93
8.3.4 TC0R AUTO-LOAD REGISTER 94
8.3.5 TC0 CLOCK FREQUENCY OUTPUT (BUZZER) 95
8.3.6 TC0 TIMER OPERATION SEQUENCE 96
8.3.7 TC0 TIMER NOTICE 97
8.4 TIMER/COUNTER 1 (TC1) 98
8.4.1 OVERVIEW 98
8.4.2 TC1M MODE REGISTER 99
8.4.3 TC1C COUNTING REGISTER 100
8.4.4 TC1R AUTO-LOAD REGISTER 101
8.4.5 TC1 CLOCK FREQUENCY OUTPUT (BUZZER) 102

8.4.6 TC1 TIMER OPERATION SEQUENCE 103
8.4.7 TC1 TIMER NOTICE 104
8.5 PWM0 MODE 105
8.5.1 OVERVIEW 105
8.5.2 TC0IRQ AND PWM DUTY 106
8.5.3 PWM PROGRAM EXAMPLE 107
8.5.4 PWM0 DUTY CHANGING NOTICE 108
8.6 PWM1 MODE 110
8.6.1 OVERVIEW 110
8.6.2 TC1IRQ AND PWM DUTY 111
8.6.3 PWM PROGRAM EXAMPLE 112
8.6.4 PWM1 DUTY CHANGING NOTICE 113

9 SERIAL INPUT/OUTPUT TRANSCEIVER (SIO) 115

9.1 OVERVIEW 115
9.2 SIO OPERATION 115
9.3 SIOM MODE REGISTER 117
9.4 SIOB DATA BUFFER 118
9.5 SIOR REGISTER DESCRIPTION 119

10 MAIN SERIAL PORT (MSP) 120

10.1 OVERVIEW 120
10.2 MSP STATUS REGISTER 120
10.3 MSP MODE REGISTER 1 121
10.4 MSP MODE REGISTER 2 122
10.5 MSP MSPBUF REGISTER 123
10.6 MSP MSPADR REGISTER 123
10.7 SLAVE MODE OPERATION 124
10.7.1 Addressing 124
10.7.2 Slave Receiving 125
10.7.3 Slave Transmission 126
10.7.4 General Call Address 127
10.7.5 Slave Wake up 128
10.8 MASTER MODE 129
10.8.1 Mater Mode Support 129
10.8.2 MSP Rate Generator 129
10.8.3 MSP Mater START Condition 130
WCOL Status Flag 130
10.8.4 MSP Master mode Repeat START Condition 131
WCOL Status Flag 131

10.8.5 Acknowledge Sequence Timing 132
WCOL Status Flag 132
10.8.6 STOP Condition Timing 133
WCOL Status Flag 133
10.8.7 Clock Arbitration 133
10.8.8 Master Mode Transmission 134
BF Status Flag 134
WCOL Flag 134
ACKSTAT Status Flag 134
10.8.9 Master Mode Receiving 135
BF Status Flag 135
MSPOV Flag 135
WCOL Flag 135

11 8 CHANNEL ANALOG TO DIGITAL CONVERTER 136

11.1 OVERVIEW 136
11.2 ADM REGISTER 137
11.3 ADR REGISTERS 138
11.4 ADB REGISTERS 139
11.5 P4CON REGISTERS 140
11.6 ADC CONVERTING TIME 140
11.7 ADC ROUTINE EXAMPLE 141
11.8 ADC CIRCUIT 142

12 DIGITAL TO ANALOG CONVERTER 143

12.1 OVERVIEW 143
12.2 143
12.3 DAM REGISTER 144
12.4 D/A CONVERTER OPERATION 144

13 INSTRUCTION TABLE 145

14 ELECTRICAL CHARACTERISTIC 146

14.1 ABSOLUTE MAXIMUM RATING 146
14.2 STANDARD ELECTRICAL CHARACTERISTIC 146

15 APPLICATION NOTICE 147

15.1 DEVELOPMENT TOOL VERSION 147
15.1.1 ICE (In circuit emulation) 147
15.1.2 OTP Writer 148
15.1.3 IDE (Integrated Development Environment) 148

15.1.4 ..... *148*
15.2 OTP PROGRAMMING PIN 149
15.2.1 The pin assignment of Easy Writer transition board socket: *149*
15.2.2 The pin assignment of Writer V3.0 and V2.5 transition board socket: *149*
15.2.3 SN8P275X Series Programming Pin Mapping: *150*
16 PACKAGE INFORMATION 151
16.1 SK-DIP28 PIN 151
16.2 SOP28 PIN 152
16.3 P-DIP 32 PIN 153
16.4 SOP 32 PIN 153
16.5 QFN 32 PIN 154
16.6 SSOP 48 PIN 155
16.7 LQFP 48 PIN 156
17 MARKING DEFINITION 157
17.1 INTRODUCTION 157
17.2 MARKING INDETIFICATION SYSTEM 157
17.3 MARKING EXAMPLE 158
17.4 DATECODE SYSTEM 158

1 PRODUCT OVERVIEW

1.1 FEATURES

- **Memory configuration**
  OTP ROM size: 4K * 16 bits.
  RAM size: 256 * 8 bits (bank 0 and bank 1).

- **8 levels stack buffer.**

- **I/O pin configuration (Total 36 pins)**
  Bi-directional: P0, P1, P2, P3, P4, P5
  Programmable open-drain: P1.0, P1.1, P5.2
  Wakeup: P0, P1 level change trigger
  External interrupt: P0
  Pull-up resisters: P0, P1, P2, P3, P4, P5
  P4 pins shared with ADC inputs.
  Programmable open-drain: P1.0, P1.1,P5.2

- **8-channel 12-bit SAR ADC.**

- **One channel 7-bit R2R DAC.**

- **SIO function.**

- **Powerful instructions**
  One clocks per instruction cycle (1T)
  Most of instructions are one cycle only
  All ROM area lookup table function (MOVC)
  Hardware multiplier (MUL)

- **Memory configuration**
  OTP ROM size: 4K * 16 bits.
  RAM size: 256 * 8 bits (bank 0 and bank 1).
  Eight levels stack buffer

- **I/O pin configuration (Total 36 pins)**
  Bi-directional: P0, P1, P2, P3, P4, P5
  Programmable open-drain: P1.0, P1.1, P5.2
  Wakeup: P0, P1 level change trigger
  External interrupt: P0
  Pull-up resisters: P0, P1, P2, P3, P4, P5
  P4 pins shared with ADC inputs.

- **Build in MSP interface with interrupt function**
  MSP sleep mode wake up function.

- **Eight interrupt sources**
  Six internal interrupts: T0, TC0, TC1, SIO, ADC,MSP.
  Three external interrupts: INT0, INT1, INT2.

- **Three 8-bit Timer/Counter**
  T0: Basic timer
  TC0: Auto-reload timer/Counter/PWM0/Buzzer output
  TC1: Auto-reload timer/Counter/PWM1/Buzzer output

- **On chip watchdog timer and clock source is internal low clock RC type (16KHz @3V, 32KHz @5V).**

- **Dual system clocks**
  External high clock: RC type up to 10 MHz
  External high clock: Crystal type 32KHZ up to 16 MHz
  Internal low clock: RC type 16KHz(3V), 32KHz(5V)
  Internal high clock: RC type 16MHz.

- **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 T0 timer

- **Package (Chip form support)**
  SN8P2758: SSOP 48 pins, LQFP 48 pins
  SN8P2755: PDIP 32 pins, SOP 32 pins
  SN8P2754: SK-DIP 28 pins, SOP 28pins

- **Three 8-bit Timer/Counter**
  T0: Basic timer
  TC0: Auto-reload timer/Counter/PWM0/Buzzer output
  TC1: Auto-reload timer/Counter/PWM1/Buzzer output

- **On chip watchdog timer and clock source is internal low clock RC type (16KHz @3V, 32KHz @5V).**

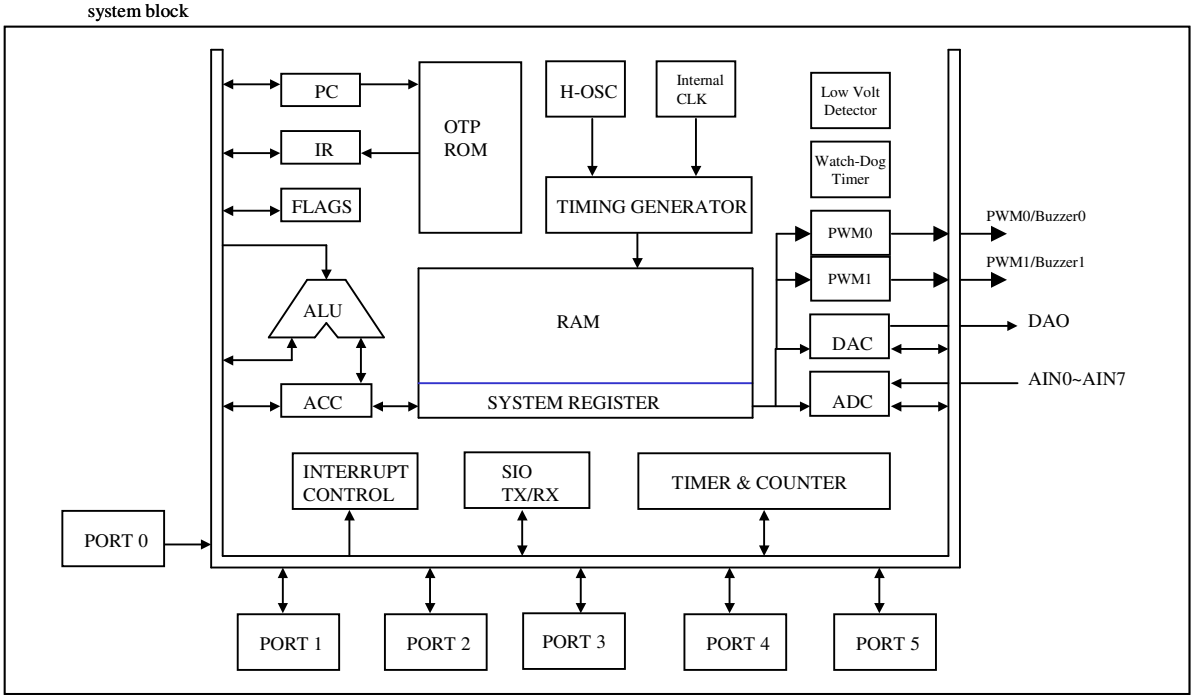
- **Dual system clocks**

☞ **Features Selection Table**

CHIP	ROM	RAM	Stack	Timer			I/O	ADC	DAC	PWM	SIO	Wakeup Pin no.	Package
				T0	TC0	TC1				Buzzer			
SN8P2758	4K*16	256	8	V	V	V	36	8ch	1ch	2	1	11	SSOP48/LQFP48
SN8P2755	4K*16	256	8	V	V	V	23	8ch	1ch	2	1	9	DIP32/SOP32/QFN32
SN8P2754	4K*16	256	8	V	V	V	18	5ch	1ch	2	1	8	SKDIP28/SOP28

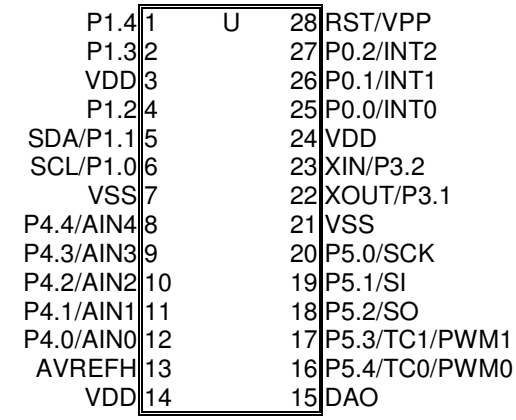
1.2 SYSTEM BLOCK DIAGRAM

system block



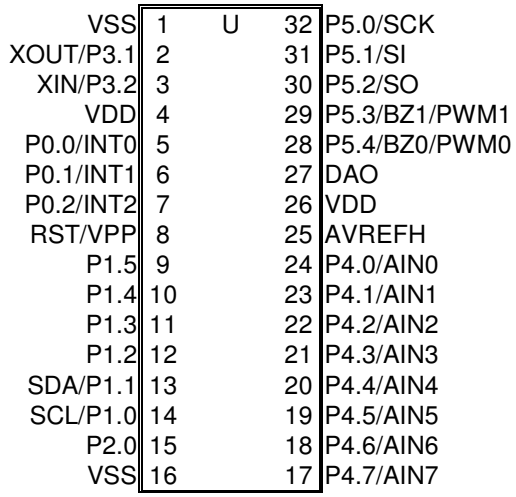
1.3 PIN ASSIGNMENT

SN8P2754K (SK-DIP28)
SN8P2754A (SOP28)



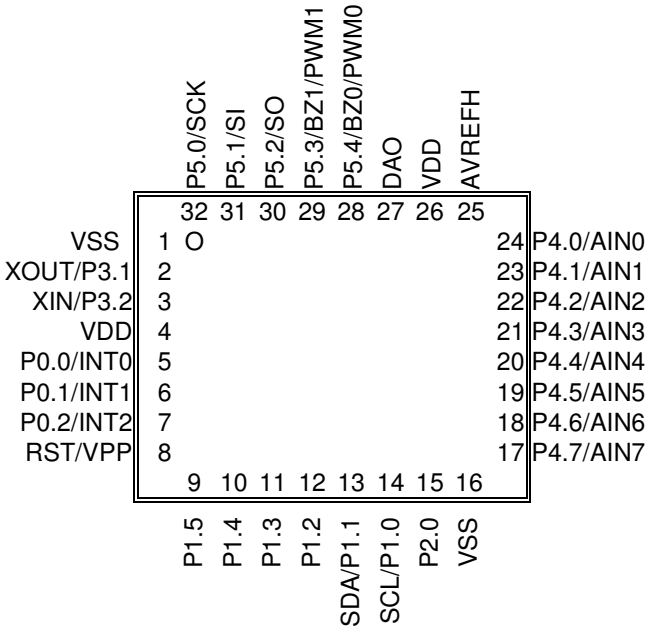
SN8P2754K
SN8P2754S

SN8P2755P (P-DIP32)
SN8P2755S (SOP32)

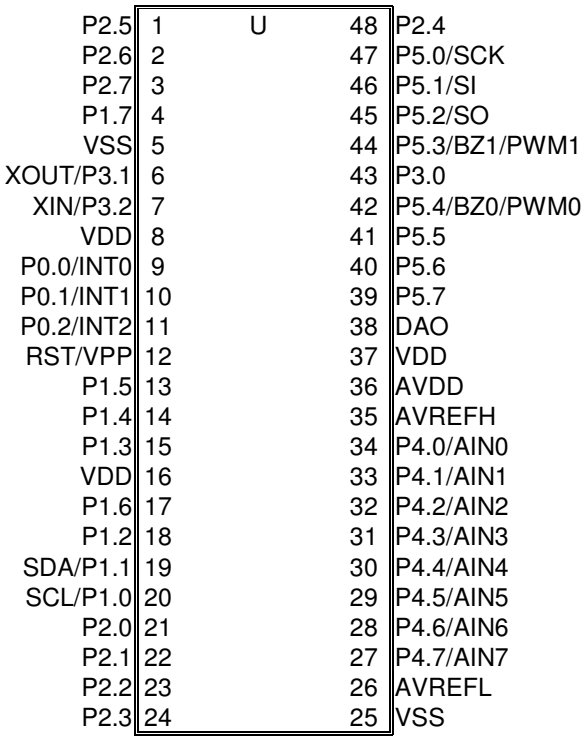


SN8P2755P
SN8P2755S

SN8P2755J (QFN32)

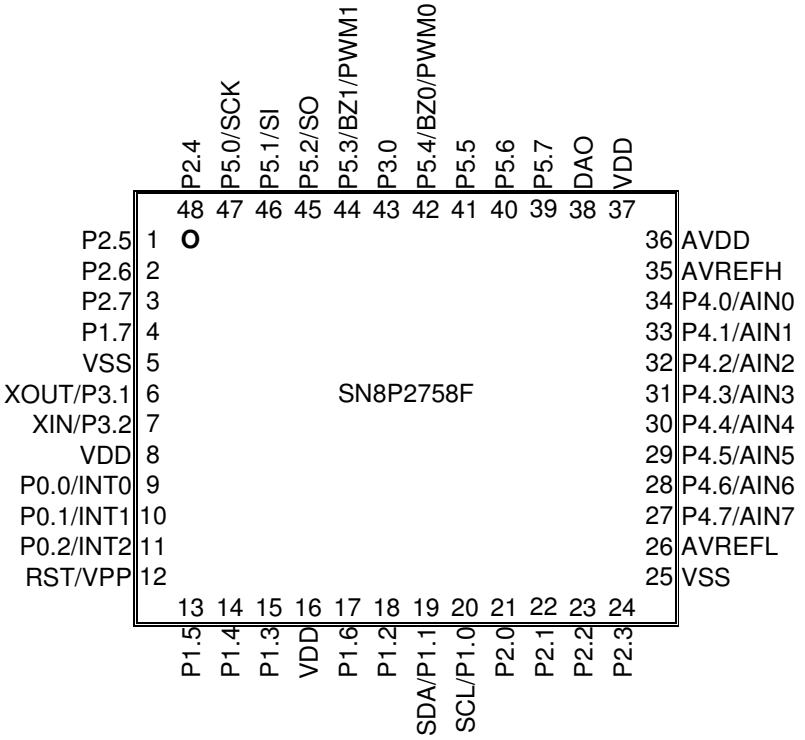


SN8P2758X (SSOP48)



SN8P2758X

SN8P2758F (LQFP48)

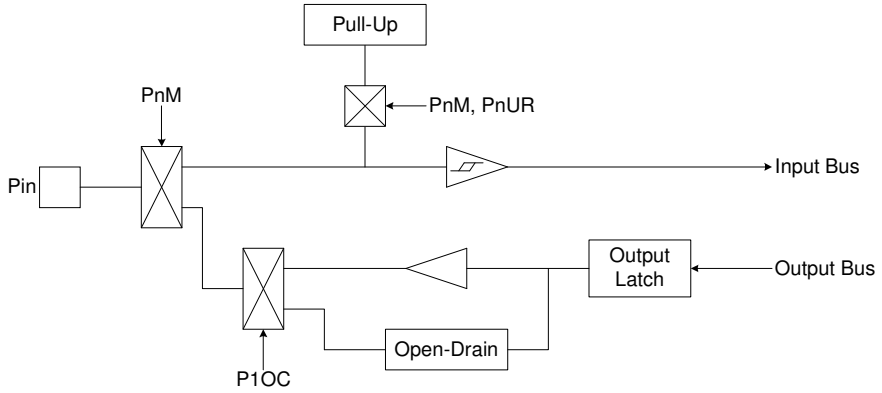


1.4 PIN DESCRIPTIONS

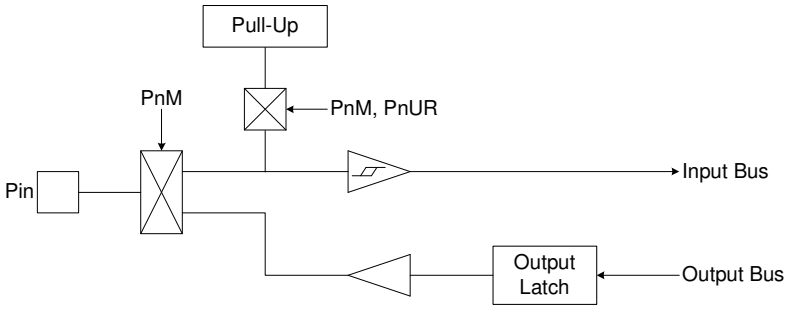
PIN NAME		DESCRIPTION
VDD, VSS	P	Power supply input pins for digital circuit.
RST/VPP	I, P	RST: System reset input pin. Schmitt trigger structure, low active, normal stay to “high”. VPP: OTP programming pin. Schmitt trigger structure as input mode. No Built-in pull-up resisters.
XIN/P3.2	I/O	Oscillator input pin while external oscillator enable (crystal and RC). Port 3.2 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
XOUT/P3.1	I/O	XOUT: Oscillator output pin while external crystal enable. Port 3.1 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
P0.0/INT0	I/O	Port 0.0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Built-in wakeup function. INT0 trigger pin (Schmitt trigger). TC0 event counter clock input pin.
P0.1/INT1	I/O	Port 0.1 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Built-in wakeup function. INT1 trigger pin (Schmitt trigger). TC1 event counter clock input pin.
P0.2/INT2	I/O	Port 0.2 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Built-in wakeup function. INT2 trigger pin (Schmitt trigger).
P1.0/SCL	I/O	Port 1.0 bi-direction pin and open-drain pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. MSP serial clock input/output pin. Programmable open-drain.
P1.1/SDA	I/O	Port P1.1 bi-direction pin and open-drain pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. MSP data I/O pin. Programmable open-drain.
P1 [7:2]	I/O	Port 1.2~P1.7 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
P2 [7:0]	I/O	Port 2 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
P3.0	I/O	Port 3.0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
P4.[7:0]/AIN[7:0]	I/O	Port 4 bi-direction pins. No Schmitt trigger structure. Built-in pull-up resisters. AIN[7:0]: ADC channel-0~7 input.
P5.0/SCK	I/O	Port 5.0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. SCK: SIO clock pin.
P5.1/SO	I/O	Port 5.1 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. SO: SIO data output pin.
P5.2/SI	I/O	Port 5.2 bi-direction pin and open-drain pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. SI: SIO data input pin.
P5.3/BZ1/PWM1	I/O	Port 5.3 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. TC1 ÷ 2 signal output pin for buzzer or PWM1 output pin.
P5.4/BZ0/PWM0	I/O	Port 5.4 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. TC0 ÷ 2 signal output pin for buzzer or PWM0 output pin.
AVDD	P	Power supply input pins for A/D circuit
AVREFH	I	ADC highest reference voltage input
AVREFL	I	ADC lowest reference voltage input
DAO	O	7bit DAC output

1.5 PIN CIRCUIT DIAGRAMS

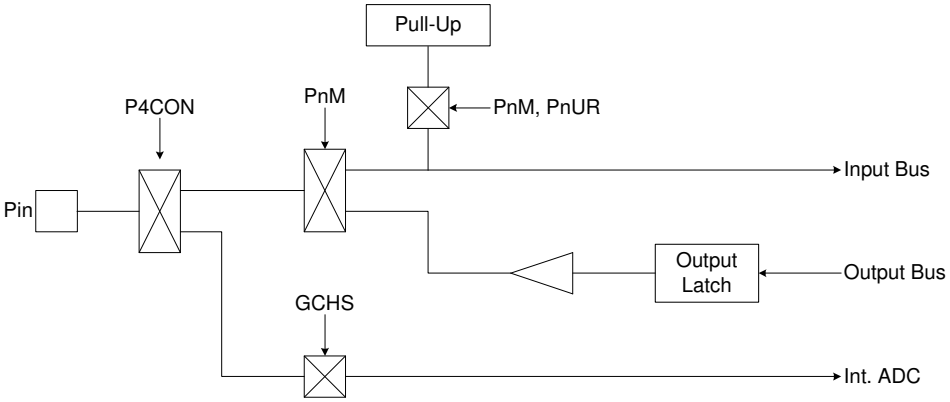
Port 1.0, P1.1, P5.2 structure:



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



Port 4 structure:

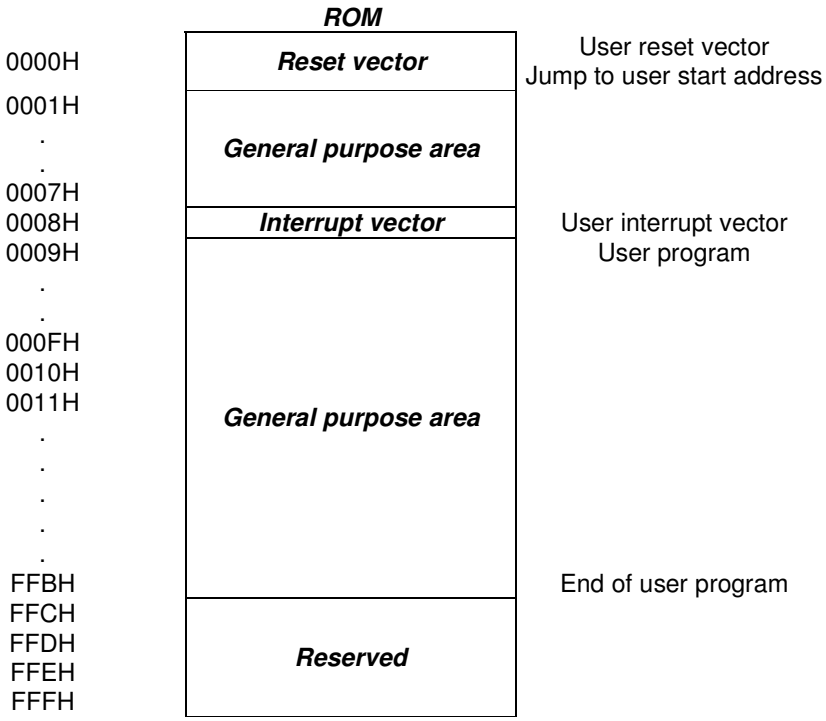


2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ ***4K words ROM***



2.1.2 RESET VECTOR (0000H)

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

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

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

➢ **Example: Defining Reset Vector**

```
                ORG     0           ; 0000H
                JMP     START       ; Jump to user program address.
                …

                ORG     10H
START:                              ; 0010H, The head of user program.
                …                   ; User program
                …


                ENDP                ; End of program
```

2.1.3 INTERRUPT VECTOR (0008H)

A 1-word vector address area is used to execute interrupt request. If any interrupt service executes, the program counter (PC) value is stored in stack buffer and jump to 0008h of program memory to execute the vectored interrupt. Users have to define the interrupt vector. The following example shows the way to define the interrupt vector in the program memory.

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

➢ **Example: Defining Interrupt Vector. The interrupt service routine is following ORG 8.**

```
.CODE
                ORG         0           ; 0000H
                JMP         START       ; Jump to user program address.
                …

                ORG         8           ; Interrupt vector.
                PUSH                    ; Save ACC and PFLAG register to buffers.
                …
                …
                POP                     ; Load ACC and PFLAG register from buffers.
                RETI                    ; End of interrupt service routine
                …

START:                                  ; The head of user program.
                …                       ; User program
                …
                JMP         START       ; End of user program
                …

                ENDP                    ; End of program
```

➢ **Example: Defining Interrupt Vector. The interrupt service routine is following user program.**

```
.CODE
                ORG     0               ; 0000H
                JMP     START           ; Jump to user program address.
                …
                ORG     8               ; Interrupt vector.
                JMP     MY_IRQ          ; 0008H, Jump to interrupt service routine address.

                ORG     10H
START:                                  ; 0010H, The head of user program.
                …                       ; User program.
                …
                …
                JMP     START           ; End of user program.
                …
MY_IRQ:                                 ;The head of interrupt service routine.
                PUSH                    ; Save ACC and PFLAG register to buffers.
                …
                …
                POP                     ; Load ACC and PFLAG register from buffers.
                RETI                    ; End of interrupt service routine.
                …

                ENDP                    ; End of program.
```

✸ ***Note: It is easy to understand the rules of SONIX program from demo programs given above. These points are as following:***

1. ***The address 0000H is a "JMP" instruction to make the program starts from the beginning.***
2. ***The address 0008H is interrupt vector.***
3. ***User's program is a loop routine for main purpose application.***

2.1.3.1 LOOK-UP TABLE DESCRIPTION

In the ROM's data lookup function, X register is pointed to high byte address (bit 16~bit 23), 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   X, #TABLE1$H    ; To set lookup table1's high address
                B0MOV   Y, #TABLE1$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 is overflow, Y=Y+1.
                JMP     @F              ; Y is not overflow.
                INCMS   X               ; Y is overflow, X=X+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 X, Y registers will not increase automatically when Y, Z registers crosses boundary from 0xFF to 0x00. Therefore, user must take care such situation to avoid loop-up table errors. If Z register is overflow, Y register must be added one. If Y register is overflow, X register must be added one. The following INC_XYZ macro shows a simple method to process X, Y and Z registers automatically.***

➢ **Example: INC_XYZ macro.**

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

                INCMS   Y               ; Y+1
                JMP     @F              ; Not overflow

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

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

```
                B0MOV     X, #TABLE1$H    ; To set lookup table1's high address
                B0MOV     Y, #TABLE1$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_XYZ                   ; Increment the index address for next address.
                                          ;
@@:             MOVC                      ; To lookup data, R = 51H, ACC = 05H.
                …                         ;
TABLE1:         DW        0035H           ; To define a word (16 bits) data.
                DW        5105H
                DW        2012H
                …
```

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

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

```
                B0MOV     X, #TABLE1$H    ; To set lookup table1's high address
                B0MOV     Y, #TABLE1$M    ; To set lookup table1'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.
                JMP       GETDATA         ; Y is not overflow.
                INCMS     X               ; Y is overflow, X=X+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
```

2.1.4.1 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
```

2.1.5 CODE OPTION TABLE

Code Option	Content	Function Description
High_Clk	RC	Low cost RC for external high clock oscillator and XOUT becomes to Fcpu frequency output pin.
	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.
	IHRC_16M	High speed internal 16MHz RC. XIN/XOUT become to P3.2/P3.1 bi-direction I/O pins.
	IHRC_RTC	High speed internal 16MHz RC with 0.5sec RTC. XIN/XOUT become to P3.2/P3.1 bit-direction I/O pins.
	32K X'tal	Low frequency, power saving crystal (e.g. 32.768KHz) for external high clock oscillator.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode. **Watchdog is running in green mode.**
	Disable	Disable Watchdog function.
Fcpu	Fhosc/1	Instruction cycle is oscillator clock. Notice: In Fosc/1, Noise Filter must be disabled.
	Fhosc/2	Instruction cycle is 2 oscillator clocks. Notice: In Fosc/1, Noise Filter must be disabled.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
	Fhosc/16	Instruction cycle is 16 oscillator clocks.
	Fhosc/32	Instruction cycle is 32 oscillator clocks.
	Fhosc/64	Instruction cycle is 64 oscillator clocks.
	Fhosc/128	Instruction cycle is 128 oscillator clocks.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
Noise_Filter	Enable	Enable Noise Filter.
	Disable	Disable Noise Filter.
External Reset Length	No	Disable External reset de-bounce time.
	128 * ILRC	Enable External reset de-bounce time.

✸ ***Note:***

1. ***In high noisy environment, enable "Noise Filter" and set Watch_Dog as "Always_On" is strongly recommended.***
2. ***Fcpu code option is only available for High Clock. Fcpu of slow mode is Flosc/4.***
3. ***In external RC mode, the Noise_Filter is enabled by assembler.***
4. ***If watchdog enable, watchdog timer is still counting in green mode.***

2.1.6 DATA MEMORY (RAM)

☞ ***256 X 8-bit RAM***

	Address	RAM location	
BANK 0	000h “ “ “ “ “ 07Fh	***General purpose area***	000h~07Fh of Bank 0 = To store general purpose data (128 bytes).
	080h “ “ “ “ “ 0FFh	***System register*** ***End of bank 0 area***	080h~0FFh of Bank 0 store system registers (128 bytes).
BANK 1	100h “ “ “ “ “ 17Fh	***General purpose area*** ***End of bank 1 area***	100h~17Fh of Bank 1 = To store general purpose data (128 bytes).

2.1.7 SYSTEM REGISTER

2.1.7.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	X	PFLAG	RBANK	-	-	-	-	-	-	-	-
9	MSPSTAT	MSPM1	MSPM2	MSPBUF	MSPADR	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P4CON	-
B	DAM	ADM	ADB	ADR	SIOM	SIOR	SIOB	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M	P3M	P4M	P5M	INTRQ_1	INTEN_1	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	P2	P3	P4	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	P2UR	P3UR	P4UR	P5UR	@HL	@YZ	-	P1OC	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.1.7.2 SYSTEM REGISTER DESCRIPTION

L, H = Working & @HL addressing register.
X = Working and ROM address register.
PFLAG = ROM page and special flag register.
DAM = DAC's mode register.
ADB = ADC's data buffer.
SIOM = SIO mode control register.
SIOB = SIO's data buffer.
PnM = Port n input/output mode register.
INTRQ = Interrupts' request register.
OSCM = Oscillator mode register.
T0M = Timer 0 mode register.
T0C = Timer 0 counting register.
TC1M = Timer/Counter 1 mode register.
TC1C = Timer/Counter 1 counting register.
STKP = Stack pointer buffer.
@HL = RAM HL indirect addressing index pointer.
P4CON= Port 4 configuration setting

R = Working register and ROM lookup data buffer.
Y, Z = Working, @YZ and ROM addressing register.
RBANK = RAM Bank Select register.
ADM = ADC's mode register.
ADR = ADC's resolution selects register.
SIOR = SIO's clock reload buffer.
P1W = Port 1 wakeup register.
Pn = Port n data buffer.
INTEN = Interrupts' enable register.
PCH, PCL = Program counter.
TC0M = Timer/Counter 0 mode register.
TC0C = Timer/Counter 0 counting register.
TC0R = Timer/Counter 0 auto-reload data buffer.
TC1R = Timer/Counter 1 auto-reload data buffer.
STK0~STK7 = Stack 0 ~ stack 7 buffer.
@YZ = RAM YZ indirect addressing index pointer.

2.1.7.3 BIT DEFINITION of SYSTEM REGISTER

Address 080H ~ 09FH

Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Register Name
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	NT0	NPD				C	DC	Z	R/W	PFLAG
087H								RBNKS0	R/W	RBANK
088H										
089H										
08AH										
08BH										
08CH										
08DH										
08EH										
08FH										
090H		CKE	D_A	P	S	RED_WRT		BF	R	MSPSTAT
091H	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK	-	MSPC	R/W	MSPM1
092H	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	R/W	MSPM2
093H	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0	R/W	MSPBUF
094H	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0	R/W	MSPADR
095H										
096H										
097H										
098H										
099H										
09AH										
09BH										
09CH										
09DH										
09EH										
09FH										

Address 0A0H ~ 0BFH

Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Register Name
0A0H										
0A1H										
0A2H										
0A3H										
0A4H										

0A5H										
0A6H										
0A7H										
0A8H										
0A9H										
0AAH										
0ABH										
0ACH										
0ADH										
0AEH	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	W	P4CON
0AFH										
0B0H	DAENB	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0	R/W	DAM
0B1H	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0	R/W	ADM
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0B3H	ADCKS2	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0B4H	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA	R/W	SIOM
0B5H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	W	SIOR
0B6H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB
0B7H										
0B8H						P02M	P01M	P00M	R/W	P0M
0B9H										
0BAH										
0BBH										
0BCH										
0BDH										
0BEH										
0BFH			P02G1	P02G0	P01G1	P01G0	P00G1	P00G0	R/W	PEDGE

Address 0C0H ~ 0DFH

Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Register Name
0C0H	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M
0C3H	-	-	-	-	-	P32M	P31M	P30M	R/W	P3M
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C5H	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C6H								MSPIRQ	R/W	INTRQ_1
0C7H								MSPIEN	R/W	INTEN_1
0C8H	ADCIRQ	TC1IRQ	TC0IRQ	T0IRQ	SIOIRQ	P02IRQ	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	TC1IEN	TC0IEN	T0IEN	SIOIEN	P02IEN	P01IEN	P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CBH										
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR

0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH					PC11	PC10	PC9	PC8	R/W	PCH
0D0H						P02	P01	P00	R/W	P0
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2
0D3H						P32	P31	P30	R/W	P3
0D4H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4
0D5H	P57	P56	P55	P54	P53	P52	P51	P50	R/W	P5
0D6H										
0D7H										
0D8H	T0ENB	T0rate2	T0rate1	T0rate0	TC1X8	TX0X8		T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP

Address 0E0H ~ 0FFH

Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Register Name
0E0H					-	P02R	P01R	P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2UR
0E3H						P32R	P31R	P30R	W	P3UR
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E8H										
0E9H						P52OC	P11OC	P10OC	W	P1OC
0EAH										
0EBH										
0ECH										
0EDH										
0EEH										
0EFH										
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H					S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H					S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L

0F5H					S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H					S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H					S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH					S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH				S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH					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.***

2.1.7.4 ACCUMULATOR

The ACC is an 8-bit data register responsible for transferring or manipulating data between ALU and data memory. If the result of operating is zero (Z) or there is carry (C or DC) occurrence, then these flags will be set to PFLAG register.

ACC is not in data memory (RAM), so ACC can't be access by "B0MOV" instruction during the instant addressing mode.

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

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

The system doesn't store ACC and PFLAG value when interrupt executed. ACC and PFLAG data must be saved to other data memories. "PUSH", "POP" save and load 0x80~0x87 system registers data into buffers. Users have to save ACC data by program.

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

```
.DATA           ACCBUF      DS   1          ; Define ACCBUF for store ACC data.
.CODE
INT_SERVICE:
                B0XCH       A, ACCBUF       ; Save ACC to buffer.
                PUSH                        ; Save PFLAG and working registers to buffer.
                …           .
                …
                POP                         ; Load PFLAG and working registers form buffers.
                B0XCH       A, ACCBUF       ; Load ACC form buffer.

                RETI                        ; Exit interrupt service vector
```

✷ ***Note: To save and re-load ACC data, users must use "B0XCH" instruction, or else the PFLAG Register might be modified by ACC operation.***

2.1.8 PROGRAM FLAG

The PFLAG register contains the arithmetic status of ALU operation, system reset status and LVD detecting status. NT0, NPD bits indicate system reset status including power on reset, LVD reset, reset by external pin active and watchdog reset. C, DC, Z bits indicate the result status of ALU operation.

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

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

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

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

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

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

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

2.1.8.1 PROGRAM COUNTER

The program counter (PC) is a 12-bit binary counter separated into the high-byte 4 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 11.

	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	-	-	-	-	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
	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 0x01 to 0x00, the PC will add 2 steps to skip next instruction.

DECS instruction:

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

DECMS instruction:

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

☞ **MULTI-ADDRESS JUMPING**

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

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

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

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

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

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

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

2.1.9 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	X	X	X	X	X	X	X	X

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	X	X	X	X	X	X	X	X

➢ **Example: If want to read a data from RAM address 20H of bank_0, 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
        …
        …
```

2.1.10 Y, Z REGISTERS

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

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

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

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

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

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

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

```
        B0MOV   Y, #0           ; Y = 0, bank 0
        B0MOV   Z, #07FH        ; Z = 7FH, the last address of the data memory area
CLR_YZ_BUF:
        CLR     @YZ             ; Clear @YZ to be zero
        DECMS   Z               ; Z – 1, if Z= 0, finish the routine
        JMP     CLR_YZ_BUF      ; Not zero
        CLR     @YZ
END_CLR:                        ; End of clear general purpose data memory area of bank 0
        …
```

2.1.10.1 X REGISTERS

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

- can be used as general working registers
- can be used as ROM data pointer with the MOVC instruction for look-up table

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	0	0	0	0	0	0	0	0

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

2.1.11 R REGISTERS

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

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

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

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

2.2 ADDRESSING MODE

2.2.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.2.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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

2.2.3 INDIRECTLY ADDRESSING MODE

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

Example: Indirectly addressing mode with @HL register

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

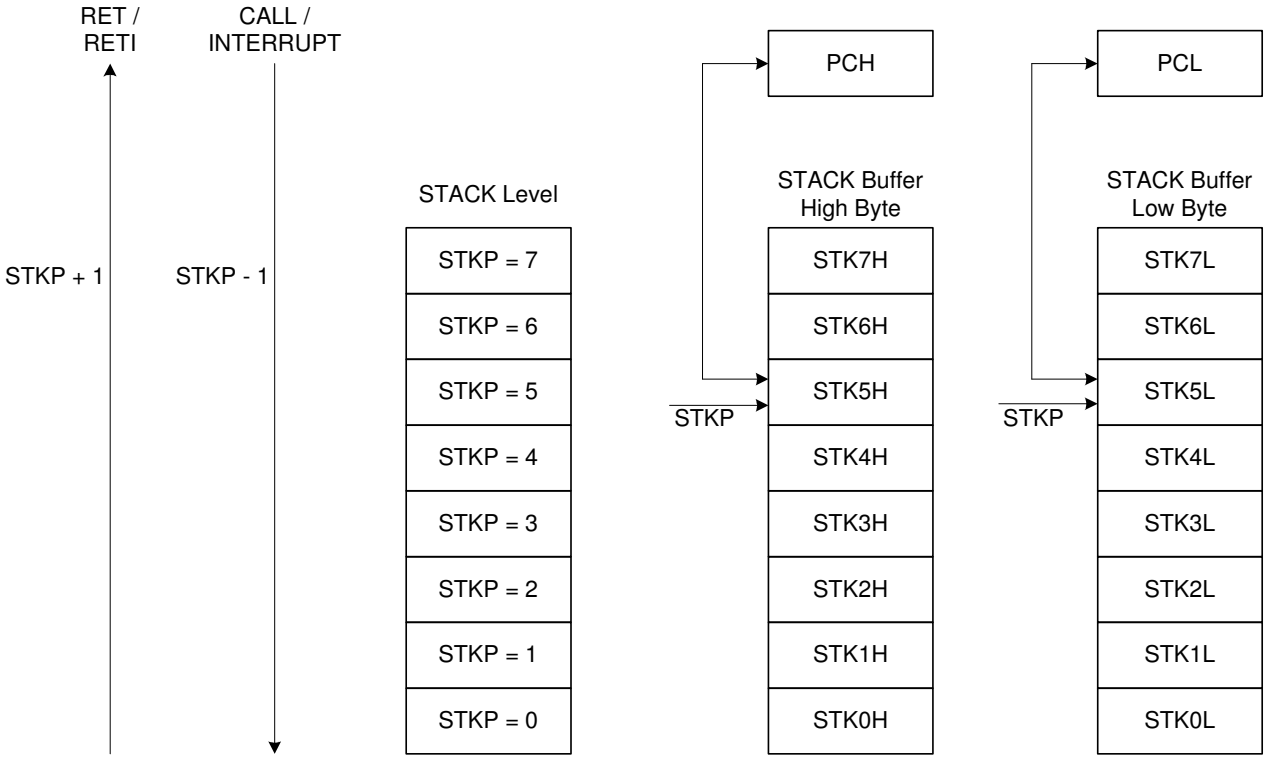
Example: Indirectly addressing mode with @YZ register

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

2.3 STACK OPERATION

2.3.1 OVERVIEW

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



2.3.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 12-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

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

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

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

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

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

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

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

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

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

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

2.3.3 STACK OPERATION EXAMPLE

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

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

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

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

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

- Power on reset
- Watchdog reset
- Brown out reset
- External reset

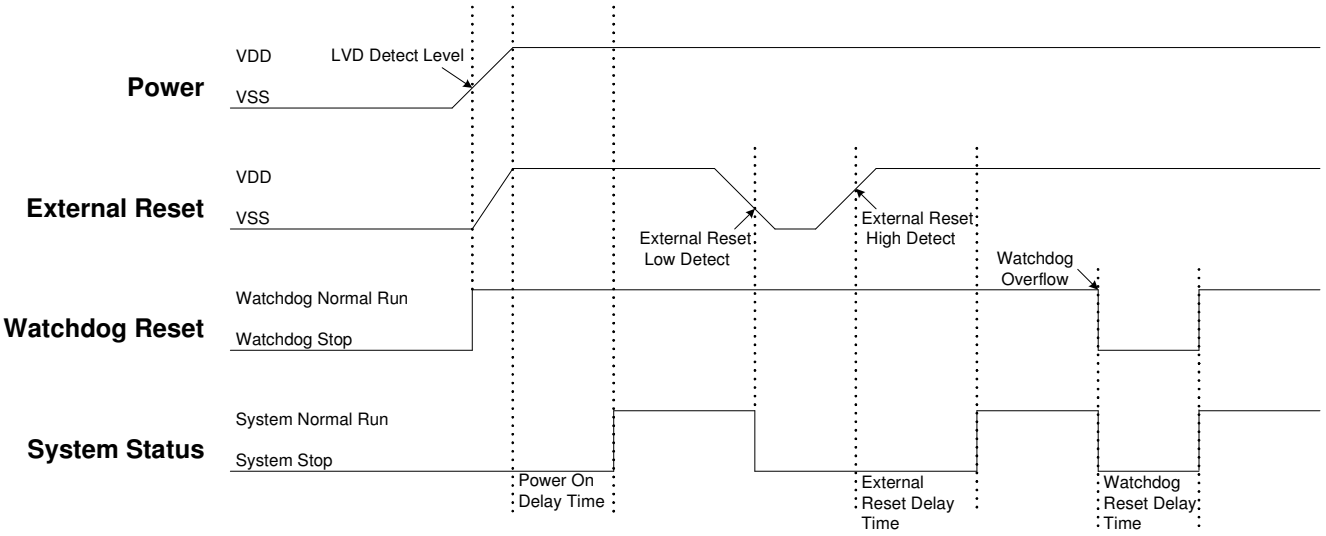
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

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

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

NT0	NPD	Condition	Description
0	0	Watchdog reset	Watchdog timer overflow.
0	1	Reserved	-
1	0	Power on reset and LVD reset.	Power voltage is lower than LVD detecting level.
1	1	External reset	External reset pin detect low level status.

Finishing any reset sequence needs some time. The system provides complete procedures to make the power on reset successful. For different oscillator types, the reset time is different. That causes the VDD rise rate and start-up time of different oscillator is not fixed. RC type oscillator's start-up time is very short, but the crystal type is longer. Under client terminal application, users have to take care the power on reset time for the master terminal requirement. The reset timing diagram is as following.



3.2 POWER ON RESET

The power on reset depend no LVD operation for most power-up situations. The power supplying to system is a rising curve and needs some time to achieve the normal voltage. Power on reset sequence is as following.

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

3.3 WATCHDOG RESET

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

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

Watchdog timer application note is as following.

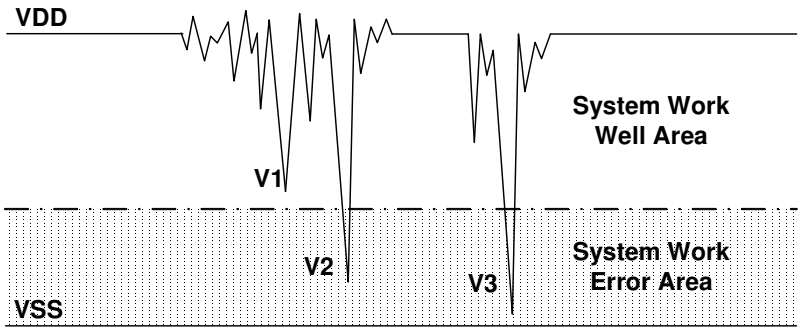
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

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

3.4 BROWN OUT RESET

3.4.1 BROWN OUT DESCRIPTION

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



Brown Out Reset Diagram

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

DC application:

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

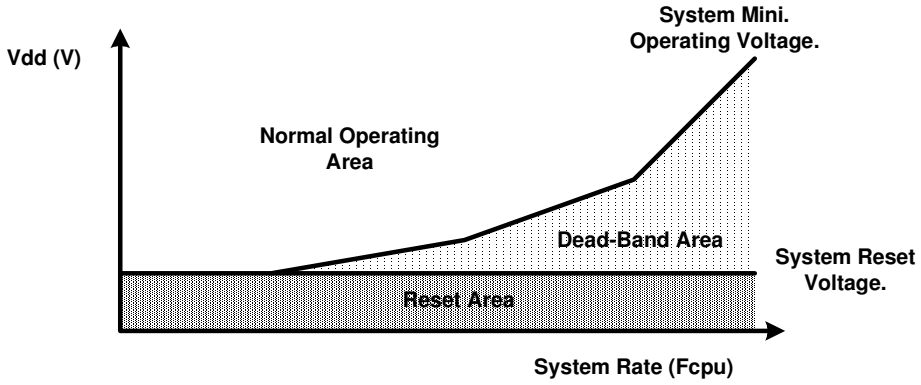
AC application:

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

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

3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION

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



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

3.4.3 BROWN OUT RESET IMPROVEMENT

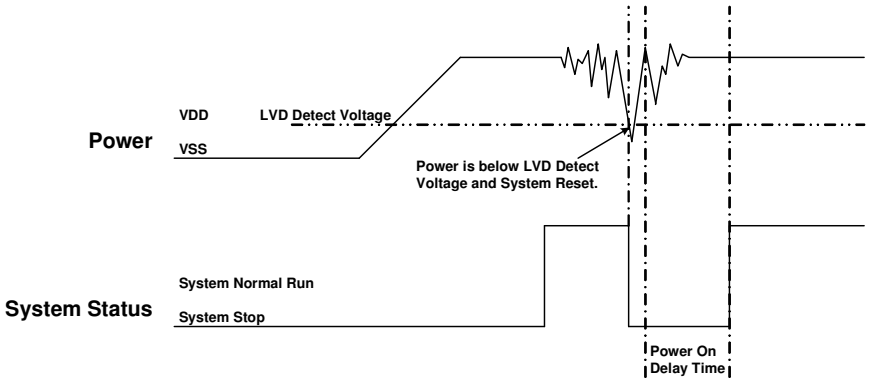
How to improve the brown reset condition? There are some methods to improve brown out reset as following.

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

✷ ***Note:***

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

LVD reset:



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

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

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

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

3.5 EXTERNAL RESET

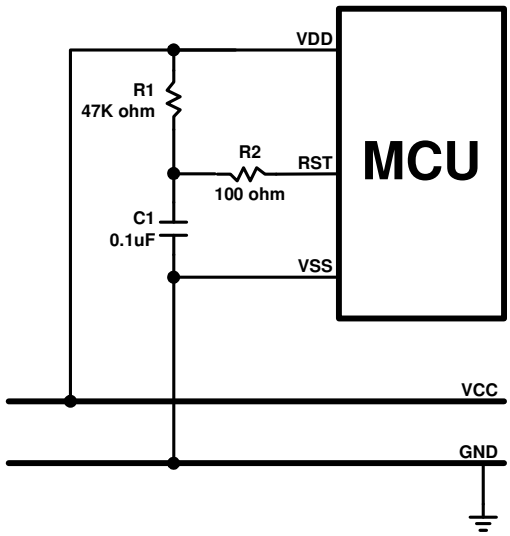
External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the low voltage and the system is reset. The external reset operation actives in power on and normal running mode. During system power-up, the external reset pin must be high level input, or the system keeps in reset status. External reset sequence is as following.

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

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

3.6 EXTERNAL RESET CIRCUIT

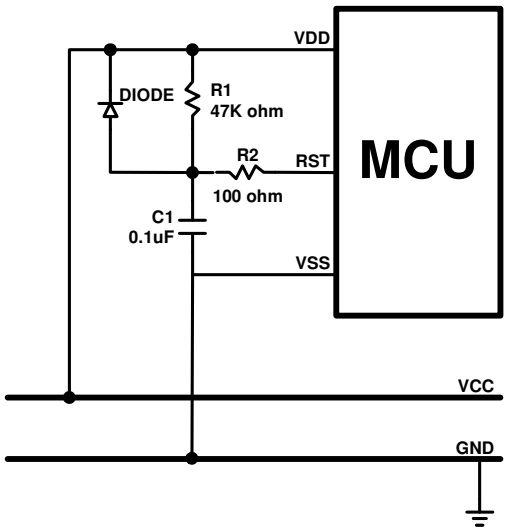
3.6.1 Simply RC Reset Circuit



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

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

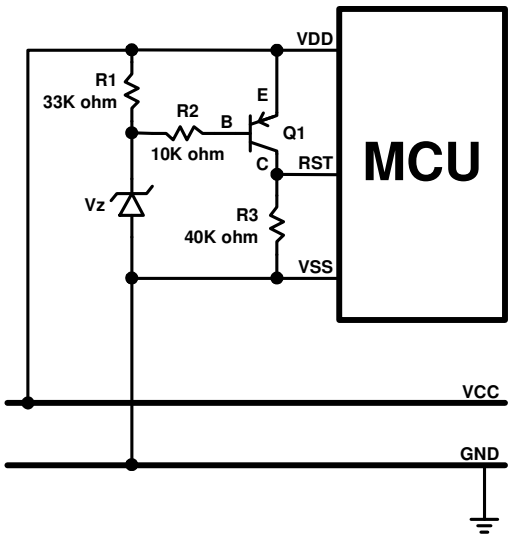
3.6.2 Diode & RC Reset Circuit



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

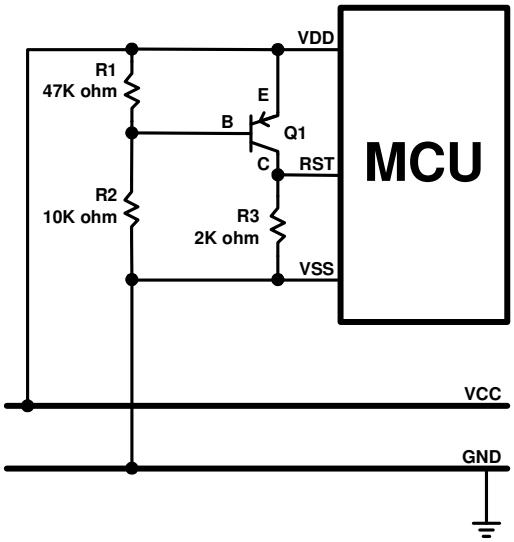
✸ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.6.3 Zener Diode Reset Circuit



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

3.6.4 Voltage Bias Reset Circuit

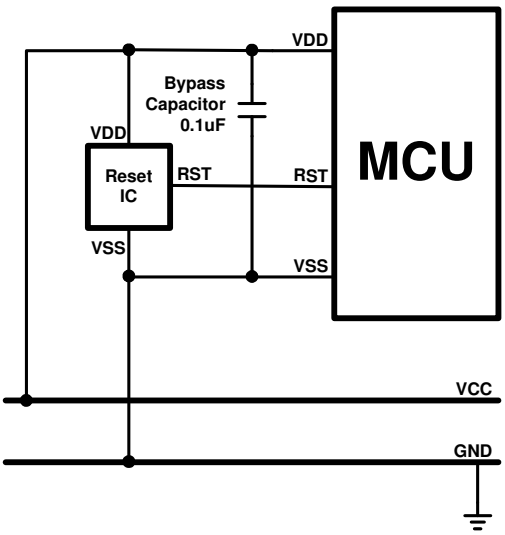


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to "0.7V x (R1 + R2) / R1",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

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

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

3.6.5 External Reset IC



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

4 SYSTEM CLOCK

4.1 OVERVIEW

The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit or on-chip 16MHz high-speed RC oscillator circuit (IHRC 16MHz). The low-speed clock is generated from on-chip low-speed RC oscillator circuit (ILRC 16KHz @3V, 32KHz @5V).

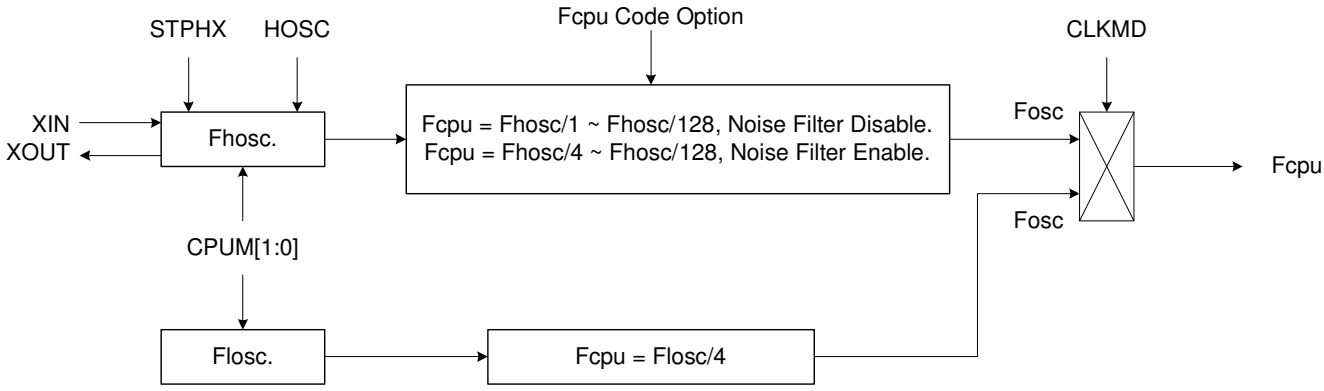
Both the high-speed clock and the low-speed clock can be system clock (Fosc). The system clock in slow mode is divided by 4 to be the instruction cycle (Fcpu).

☞ **Normal Mode (High Clock):** **Fcpu = Fhosc** / **N**, N = 1 ~ 128 Select N by Fcpu code option.

☞ **Slow Mode (Low Clock):** **Fcpu = Flosc/4.**

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

4.2 CLOCK BLOCK DIAGRAM



- HOSC: High_Clk code option.
- Fhosc: External high-speed clock / Internal high-speed RC clock.
- Flosc: Internal low-speed RC clock (about 16KHz@3V, 32KHz@5V).
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.3 OSCM REGISTER

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

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

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

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

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

- **Example: Stop high-speed oscillator**

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

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

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

4.4 SYSTEM HIGH CLOCK

The system high clock is from internal 16MHz oscillator RC type or external oscillator. The high clock type is controlled by “High_Clk” code option.

High_Clk Code Option	Description
IHRC_16M	The high clock is internal 16MHz oscillator RC type. XIN and XOUT pins are general purpose I/O pins.
IHRC_RTC	The high clock is internal 16MHz oscillator RC type. XIN and XOUT pins connect with 32768Hz crystal for RTC clock source.
RC	The high clock is external RC type oscillator. XOUT pin is general purpose I/O pin.
32K	The high clock is external 32768Hz low speed oscillator.
12M	The high clock is external high speed oscillator. The typical frequency is 12MHz.
4M	The high clock is external oscillator. The typical frequency is 4MHz.

4.4.1 INTERNAL HIGH RC

The chip is built-in RC type internal high clock (16MHz) controlled by “IHRC_16M” or “IHRC_RTC” code options. In “IHRC_16M” mode, the system clock is from internal 16MHz RC type oscillator and XIN / XOUT pins are general-purpose I/O pins. In “IHRC_RTC” mode, the system clock is from internal 16MHz RC type oscillator and XIN / XOUT pins are connected with external 32768 crystal for real time clock (RTC).

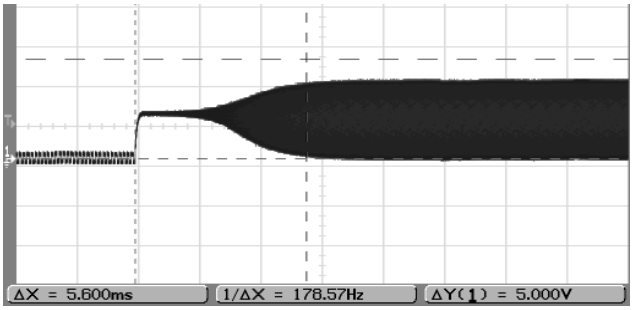
- **IHRC:** High clock is internal 16MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.
- **IHRC_RTC:** High clock is internal 16MHz oscillator RC type. XIN/XOUT pins are connected with external 32768Hz crystal/ceramic oscillator for RTC clock source.

The RTC period is controlled by OPTION register and RTC timer is T0. Please consult “T0 Timer” chapter to apply RTC function.

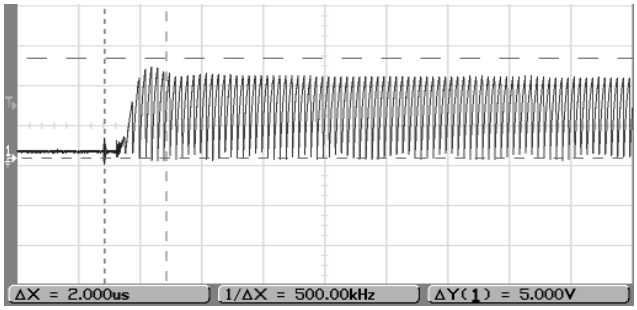
4.4.2 EXTERNAL HIGH CLOCK

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

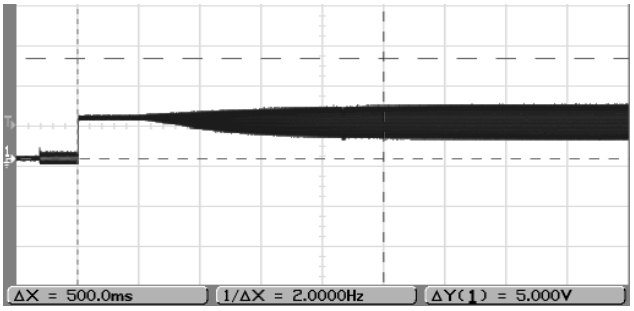
4MHz Crystal



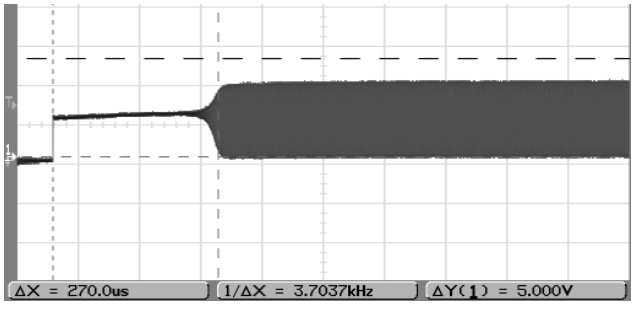
RC



32768Hz Crystal

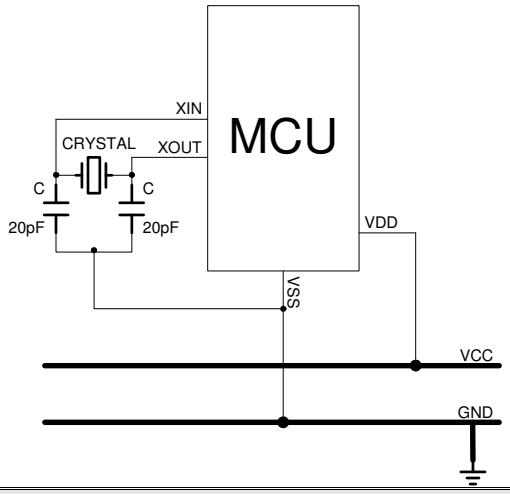


4MHz Ceramic



4.4.2.1 CRYSTAL/CERAMIC

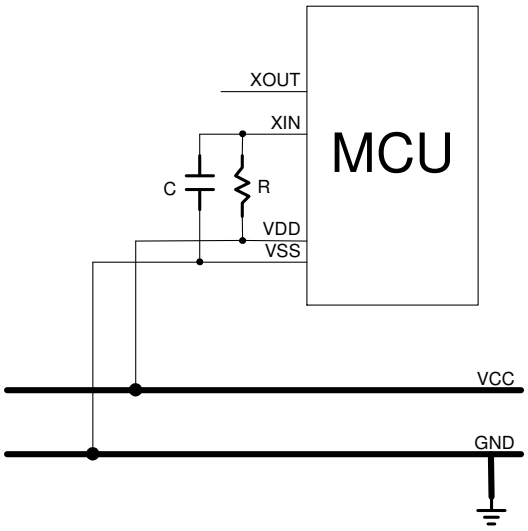
Crystal/Ceramic devices are driven by XIN, XOUT pins. For high/normal/low frequency, the driving currents are different. High_Clk code option supports different frequencies. 12M option is for high speed (ex. 12MHz). 4M option is for normal speed (ex. 4MHz). 32K option is for low speed (ex. 32768Hz).



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

4.4.2.2 RC

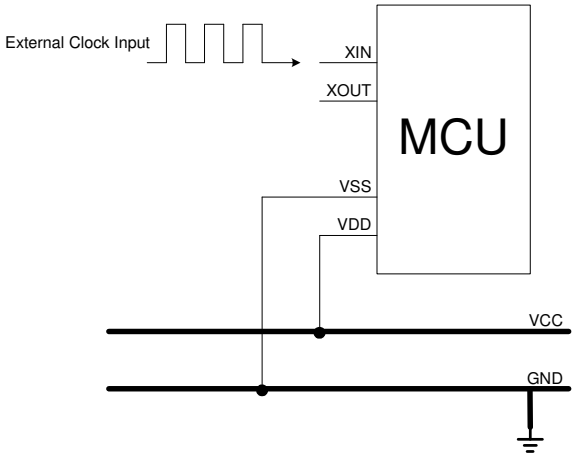
Selecting RC oscillator is by RC option of High_Clk code option. RC type oscillator's frequency is up to 10MHz. Using "R" value is to change frequency. 50P~100P is good value for "C". XOUT pin is general purpose I/O pin.



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

4.4.2.3 EXTERNAL CLOCK SIGNAL

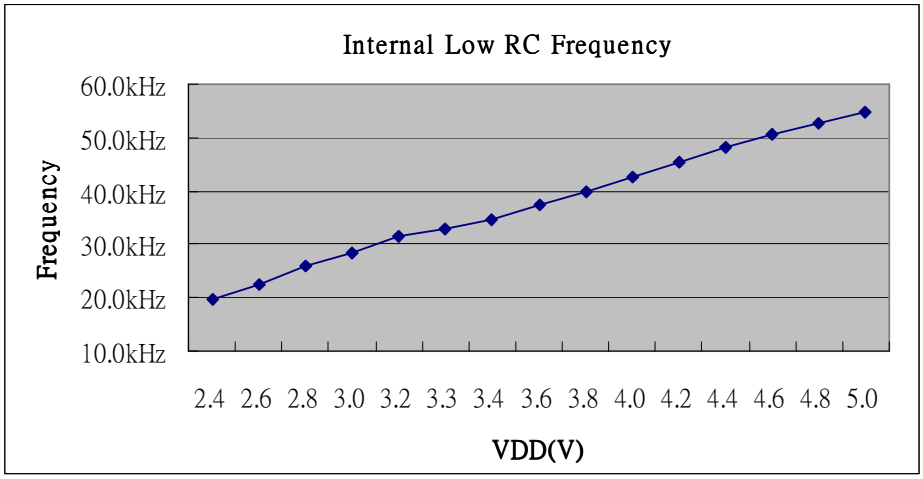
Selecting external clock signal input to be system clock is by RC option of High_Clk code option. The external clock signal is input from XIN pin. XOUT pin is general purpose I/O pin.



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

4.5 SYSTEM LOW CLOCK

The system low clock source is the internal low-speed oscillator built in the micro-controller. The low-speed oscillator uses RC type oscillator circuit. The frequency is affected by the voltage and temperature of the system. In common condition, the frequency of the RC oscillator is about 16KHz at 3V and 32KHz at 5V. The relation between the RC frequency and voltage is as the following figure.



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

☞ ***Flosc = Internal low RC oscillator (about 32 KHz @3.3V).***

☞ ***Slow mode Fcpu = Flosc / 4***

The only one condition to stop internal low RC is the system into power down mode with watchdog disable or enable. System into power down mode with watchdog set Always_On in code option, the ILRC is still running and watchdog is active well.

```
B0BSET      FCPUM0          ; To stop IHRC and ILRC Oscillator called power down mode
                            ; (sleep mode).
```

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

4.5.1 SYSTEM CLOCK MEASUREMENT

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

➢ **Example: Fcpu instruction cycle of external oscillator.**

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

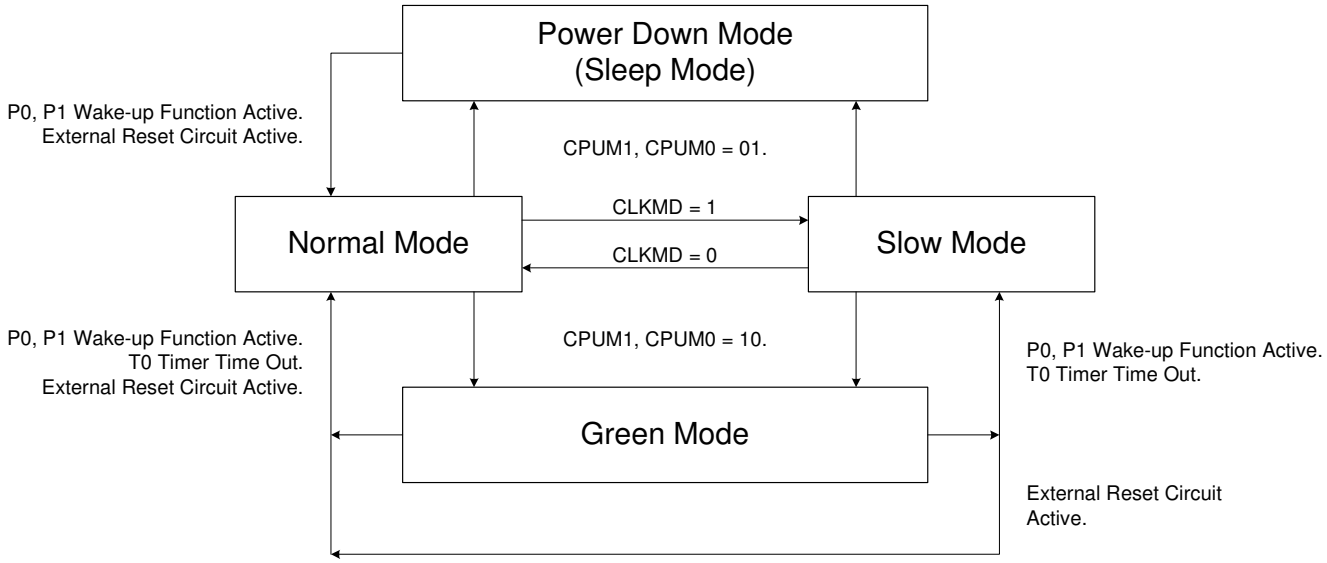
✷ ***Note: Do not measure the RC frequency directly from XIN; the probe impendence will affect the RC frequency.***

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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



System Mode Switching Diagram

Operating mode description

MODE	NORMAL	SLOW	GREEN	POWER DOWN (SLEEP)	REMARK
EHOSC	Running	By STPHX	By STPHX	Stop	
ILRC	Running	Running	Running	Stop	
CPU instruction	Executing	Executing	Stop	Stop	
T0 timer	*Active	*Active	*Active	Inactive	* Active if T0ENB=1
TC0 timer	*Active	*Active	*Active	Inactive	* Active if TC0ENB=1
TC1 timer	*Active	*Active	*Active	Inactive	* Active if TC1ENB=1
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	Refer to code option description
Internal interrupt	All active	All active	T0, TC0	All inactive	
External interrupt	All active	All active	All active	All inactive	
Wakeup source	-	-	P0, P1, T0 Reset	P0, P1, Reset	

EHOSC: External high clock
ILRC: Internal low clock (16K RC oscillator at 3V, 32K at 5V)

5.2 SYSTEM MODE SWITCHING

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

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

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

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

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

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

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

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

If external high clock stop and program want to switch back normal mode. It is necessary to delay at least 20ms for external clock stable.

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

            B0MOV       Z, #54          ; If VDD = 5V, internal RC=32KHz (typical) will delay
@@:         DECMS       Z               ; 0.125ms X 162 = 20.25ms for external clock stable
            JMP         @B

            B0BCLR      FCLKMD          ; Change the system back to the normal mode
```

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

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

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

➢ **Example: Switch normal/slow mode to Green mode and enable T0 wakeup function.**

```
; Set T0 timer wakeup function.
        B0BCLR      FT0IEN      ; To disable T0 interrupt service
        B0BCLR      FT0ENB      ; To disable T0 timer
        MOV         A,#20H      ;
        B0MOV       T0M,A       ; To set T0 clock = Fcpu / 64
        MOV         A,#74H
        B0MOV       T0C,A       ; To set T0C initial value = 74H (To set T0 interval = 10 ms)

        B0BCLR      FT0IEN      ; To disable T0 interrupt service
        B0BCLR      FT0IRQ      ; To clear T0 interrupt request
        B0BSET      FT0ENB      ; To enable T0 timer

; Go into green mode
        B0BCLR      FCPUM0      ;To set CPUMx = 10
        B0BSET      FCPUM1
```

✸ ***Note: During the green mode with T0 wake-up function, the wakeup pins, reset pin and T0 can wakeup the system back to the last mode. T0 wake-up period is controlled by program and T0ENB must be set.***

5.3 WAKEUP

5.3.1 OVERVIEW

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

- 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).

5.3.2 WAKEUP TIME

When the system is in power down mode (sleep mode), the high clock oscillator stops. When waked up from power down mode, MCU waits for 4096 external high-speed oscillator clocks as the wakeup time to stable the oscillator circuit. After the wakeup time, the system goes into the normal mode.

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

The value of the wakeup time is as the following.

The Wakeup time = 1/Fosc * 4096 (sec) + high clock start-up time

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

➢ **Example: In 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 * 4096 = 1.024 ms (Fosc = 4MHz)

The total wakeup time = 1.024ms + oscillator start-up time

5.3.3 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 Port 0 and Port 1 have wakeup function. Port 0 wakeup function always enables, but the Port 1 is controlled by the P1W register.

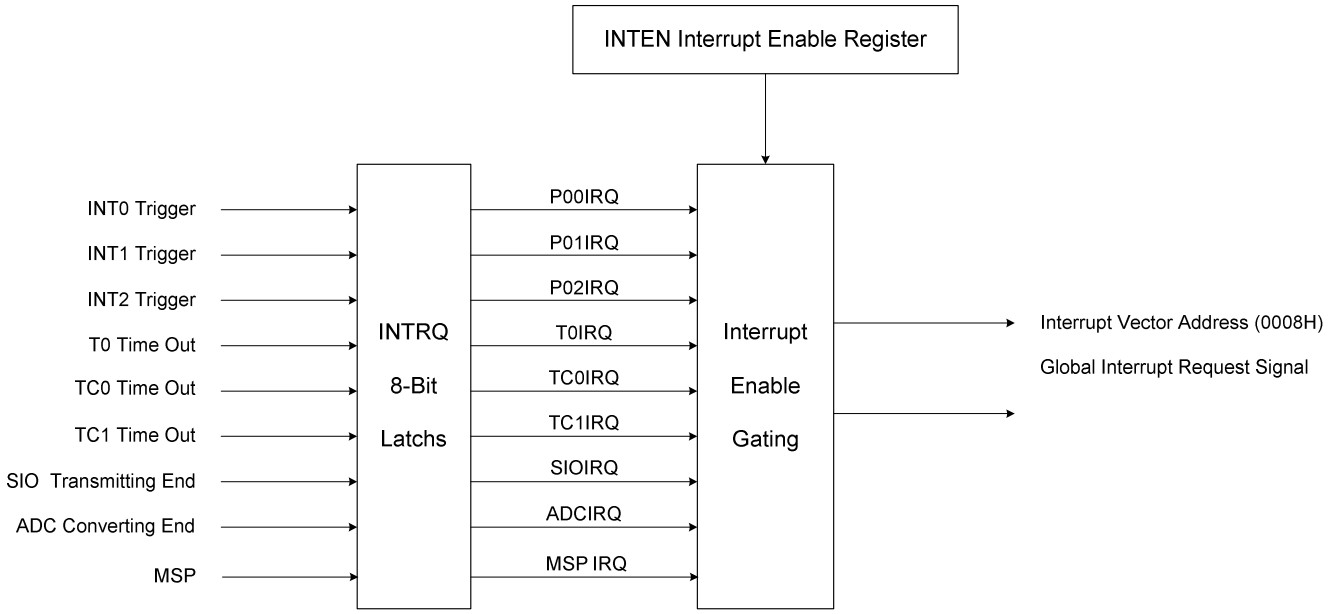
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Bit[7:0] **P10W~P17W:** Port 1 wakeup function control bits.
0 = Disable P1n wakeup function.
1 = Enable P1n wakeup function.

6 INTERRUPT

6.1 OVERVIEW

This MCU provides nine interrupt sources, including six internal interrupt (T0/TC0/TC1/SIO/ADC/MSP) and three external interrupt (INT0/INT1/INT2). 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. All of the interrupt request signals are stored in INTRQ register.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

INTEN is the interrupt request control register including three 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 to execute interrupt service routines. The program exits the interrupt service routine when the returning interrupt service routine instruction (RETI) is executed.

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	ADCIEN	TC1IEN	TC0IEN	T0IEN	SIOIEN	P02IEN	P01IEN	P00IEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 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 **P02IEN:** External P0.2 interrupt (INT2) control bit.
0 = Disable INT1 interrupt function.
1 = Enable INT1 interrupt function.

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

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

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

Bit 6 **TC1IEN:** TC1 timer interrupt control bit.
0 = Disable TC1 interrupt function.
1 = Enable TC1 interrupt function.

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

0C7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN_1	-	-	-	-	-	-	-	MSPIEN
Read/Write	-	-	-	-	-	-	-	R/W
After Reset	-	-	-	-	-	-	-	0

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

6.3 INTRQ INTERRUPT REQUEST REGISTER

INTRQ is the interrupt request flag register. The register includes all interrupt request indication flags. Each one of the interrupt requests occurs, the bit of the INTRQ register would be set "1". The INTRQ value needs to be clear by programming after detecting the flag. In the interrupt vector of program, users know the any interrupt requests occurring by the register and do the routine corresponding of the interrupt request.

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	ADCIRQ	TC1IRQ	TC0IRQ	T0IRQ	SIOIRQ	P02IRQ	P01IRQ	P00IRQ
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 0 **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 **P02IRQ:** External P0.2 interrupt (INT2) request flag.
0 = None INT1 interrupt request.
1 = INT1 interrupt request.

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

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

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

Bit 6 **TC1IRQ:** TC1 timer interrupt request flag.
0 = None TC1 interrupt request.
1 = TC1 interrupt request.

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

0C6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ_1	-	-	-	-	-	-	-	MSPIRQ
Read/Write	-	-	-	-	-	-	-	R/W
After Reset	-	-	-	-	-	-	-	0

Bit 0 **MSPIRQ:** MSP interrupt request bit.
0 = No Request.
1 = Request.

6.4 GIE GLOBAL INTERRUPT OPERATION

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

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

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

Example: Set global interrupt control bit (GIE).

```
B0BSET      FGIE        ; Enable GIE
```

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

6.5 PUSH, POP ROUTINE

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

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

➢ **Example: Store ACC and PAFLG data by PUSH, POP instructions when interrupt service routine executed.**

```
            ORG     0
            JMP     START

            ORG     8
            JMP     INT_SERVICE

            ORG     10H
START:
            …

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

            RETI                ; Exit interrupt service vector
            …
            ENDP
```

6.6 EXTERNAL INTERRUPT OPERATION (INT0~INT2)

Sonix provides 3 sets external interrupt sources in the micro-controller. INT0, INT1 and INT2 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 8) 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, P0.1 or P0.2), 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.

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	P02G1	P02G0	P01G1	P01G0	P00G1	P00G0
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit[5:4] **P02G[1:0]:** INT2 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

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, #98H
        B0MOV   PEDGE, A        ; Set INT0 interrupt trigger as bi-direction edge.

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

Example: INT0 interrupt service routine.

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

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

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

6.7 T0 INTERRUPT OPERATION

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

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

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

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

        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: T0 interrupt service routine.**

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

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

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

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


        RETI                    ; Exit interrupt vector
```

6.8 TC0 INTERRUPT OPERATION

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

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

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

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

        B0BSET      FGIE            ; Enable GIE
```

➢ **Example: TC0 interrupt service routine.**

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

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

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

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

        RETI                        ; Exit interrupt vector
```

6.9 TC1 INTERRUPT OPERATION

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

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

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

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

        B0BSET      FGIE        ; Enable GIE
```

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

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

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

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

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

        RETI                    ; Exit interrupt vector
```

6.10 SIO INTERRUPT OPERATION

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

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

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

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

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

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

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

        RETI                    ; Exit interrupt vector
```

6.11 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         8               ; 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
```

6.12 MULTI-INTERRUPT OPERATION

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

Interrupt Name	*Trigger Event Description*
P00IRQ	P0.0 trigger controlled by PEDGE
P01IRQ	P0.1 trigger controlled by PEDGE
P02IRQ	P0.2 trigger controlled by PEDGE
T0IRQ	T0C overflow
TC0IRQ	TC0C overflow
TC1IRQ	TC1C overflow
SIOIRQ	SIO transmitting end.
ADCIRQ	ADC converting end.

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

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

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

INTP00CHK:                                  ; Check INT0 interrupt request
                B0BTS1      FP00IEN         ; Check P00IEN
                JMP         INTP01CHK       ; Jump check to next interrupt
                B0BTS0      FP00IRQ         ; Check P00IRQ
                JMP         INTP00
INTP01CHK:                                  ; Check INT1 interrupt request
                B0BTS1      FP00IEN         ; Check P01IEN
                JMP         INTP02CHK       ; Jump check to next interrupt
                B0BTS0      FP01IRQ         ; Check P01IRQ
                JMP         INTP01
INTP02CHK:                                  ; Check INT2 interrupt request
                B0BTS1      FP00IEN         ; Check P02IEN
                JMP         INTT0CHK        ; Jump check to next interrupt
                B0BTS0      FP02IRQ         ; Check P02IRQ
                JMP         INTP02
INTT0CHK:                                   ; Check T0 interrupt request
                B0BTS1      FT0IEN          ; Check T0IEN
                JMP         INTTC0CHK       ; Jump check to next interrupt
                B0BTS0      FT0IRQ          ; Check T0IRQ
                JMP         INTT0           ; Jump to T0 interrupt service routine
INTTC0CHK:                                  ; Check TC0 interrupt request
                B0BTS1      FTC0IEN         ; Check TC0IEN
                JMP         INTTC1CHK       ; Jump check to next interrupt
                B0BTS0      FTC0IRQ         ; Check TC0IRQ
                JMP         INTTC0          ; Jump to TC0 interrupt service routine
INTTC1CHK:                                  ; Check T1 interrupt request
                B0BTS1      FTC1IEN         ; Check TC1IEN
                JMP         INTSIOHK        ; Jump check to next interrupt
                B0BTS0      FTC1IRQ         ; Check TC1IRQ
                JMP         INTT1           ; Jump to TC1 interrupt service routine
INTSIOCHK:                                  ; Check SIO interrupt request
                B0BTS1      FSIOIEN         ; Check SIOIEN
                JMP         INTADCHK        ; Jump check to next interrupt
                B0BTS0      FSIOIRQ         ; Check SIOIRQ
                JMP         INTSIO          ; Jump to SIO interrupt service routine
INTADCHK:                                   ; Check ADC interrupt request
                B0BTS1      FADCIEN         ; Check ADCIEN
                JMP         INT_EXIT        ; Jump to exit of IRQ
                B0BTS0      FADCIRQ         ; Check ADCIRQ
                JMP         INTADC          ; Jump to ADC interrupt service routine
INT_EXIT:
                …                           ; Pop routine to load ACC and PFLAG from buffers.

                RETI                        ; Exit interrupt vector
```

7 I/O PORT

7.1 I/O PORT MODE

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

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	-	-	-	-	-	P02M	P01M	P00M
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

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

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

0C3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3M	-	-	-	-	-	P32M	P31M	P30M
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

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

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M
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] **PnM[7:0]:** Pn mode control bits. (n = 0~5).
0 = Pn is input mode.
1 = Pn is output mode.

✸ ***Note: Users can program them by bit control instructions (B0BSET, B0BCLR).***
✸ ***Note: If not used ADC function, AVDD must be connect with VDD, otherwise P4 I/O maybe ERROR.***

➢ **Example: I/O mode selecting**

```
CLR     P0M         ; Set all ports to be input mode.
CLR     P4M
CLR     P5M

MOV     A, #0FFH    ; Set all ports to be output mode.
B0MOV   P0M, A
B0MOV   P4M,A
B0MOV   P5M, A

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

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

7.2 I/O PULL UP REGISTER

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	-	-	-	-	P02R	P01R	P00R
Read/Write	-	-	-	-	-	W	W	W
After reset	-	-	-	-	-	0	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

0E3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3UR	-	-	-	-	-	P32R	P31R	P30R
Read/Write	-	-	-	-	-	W	W	W
After reset	-	-	-	-	-	0	0	0

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

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

➢ **Example: I/O Pull up Register**

```
MOV     A, #0FFH        ; Enable Port0, 4, 5 Pull-up register,
B0MOV   P0UR, A         ;
B0MOV   P4UR,A
B0MOV   P5UR, A
```

7.3 I/O PORT DATA REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	-	-	-	-	-	P02	P01	P00
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

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

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

0D3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3	-	-	-	-	-	P32	P31	P30
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

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

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	P57	P56	P55	P54	P53	P52	P51	P50
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

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

```
B0MOV   A, P0       ; Read data from Port 0
B0MOV   A, P4       ; Read data from Port 4
B0MOV   A, P5       ; Read data from Port 5
```

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

```
MOV     A, #0FFH    ; Write data FFH to all Port.
B0MOV   P0, A
B0MOV   P4, A
B0MOV   P5, A
```

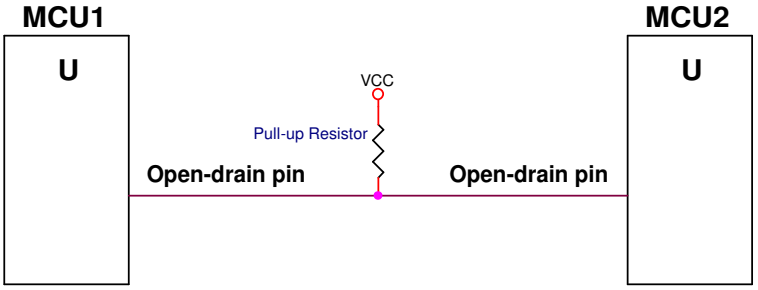
➢ **Example: Write one bit data to output port.**

```
B0BSET  P4.0        ; Set P4.0 and P5.3 to be “1”.
B0BSET  P5.3

B0BCLR  P4.0        ; Set P4.0 and P5.3 to be “0”.
B0BCLR  P5.3
```

7.4 I/O OPEN-DRAIN REGISTER

P1.0/P1.1/P5.2 is built-in open-drain function. P1.0/P1.1/P5.2 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.

0E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1OC	-	-	-	-	-	P52OC	P11OC	P10OC
Read/Write	-	-	-	-	-	W	W	W
After reset	-	-	-	-	-	0	0	0

Bit 2 **P52OC:** P5.2 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 1 **P11OC:** P1.1 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 0 **P10OC:** P1.0 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

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

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

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

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

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

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

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

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.

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
Read/Write	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[4:0] **P4CON[7:0]:** P4.n configuration control bits.
0 = P4.n can be an analog input (ADC input) or digital I/O pins.
1 = P4.n is pure analog input, can't be a digital I/O pin.

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

Port 4 ADC analog input is controlled by GCHS and CHSn bits of ADM register. If GCHS = 0, P4.n is general purpose bi-direction I/O port. If GCHS = 1, P4.n pointed by CHSn is ADC analog signal input pin. Users should set P4 ADC input pin as input mode without pull-up.

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0
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 4 **GCHS:** Global channel select bit.
0 = Disable AIN channel.
1 = Enable AIN channel.

Bit[2:0] **CHS[2:0]:** ADC input channels select bit.
000 = AIN0, 001 = AIN1, … 110 = AIN6, 111 = AIN7.

✷ ***Note: For P4.n general purpose I/O function, users should make sure of P4.n's ADC channel is disabled.***

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

```
; Check GCHS and CHS[2:0] status.
        B0BCLR      FGCHS       ;If CHS[2:0] point to P4.1 (CHS[2:0] = 001B), set GCHS=0
                                ;If CHS[2:0] don't point to P4.1 (CHS[2:0] ≠ 001B), 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[2:0] status.
        B0BCLR      FGCHS       ;If CHS[2:0] point to P4.1 (CHS[2:0] = 001B), set GCHS=0.
                                ;If CHS[2:0] don't point to P4.1 (CHS[2:0] ≠ 001B), 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 input 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, WDT overflow signal raises and resets MCU. Watchdog clock controlled by code option and the clock source is internal low-speed oscillator (16KHz @3V, 32KHz @5V).

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

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

✷ ***Note:***

1. ***If watchdog is "Always_On" mode, it keeps running event under power down mode or green mode.***
2. ***For S8KD ICE simulation, clear watchdog timer using "@RST_WDT" macro is necessary. Or the S8KD watchdog would be error.***

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

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

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

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

➢ **Example: Clear watchdog timer by @RST_WDT macro.**

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

Watchdog timer application note is as following.

- 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.
        B0BSET      FWDRST      ; Only one clearing watchdog timer of whole program.
        …
        CALL        SUB1
        CALL        SUB2
        …
        …
        …
        JMP         MAIN
```

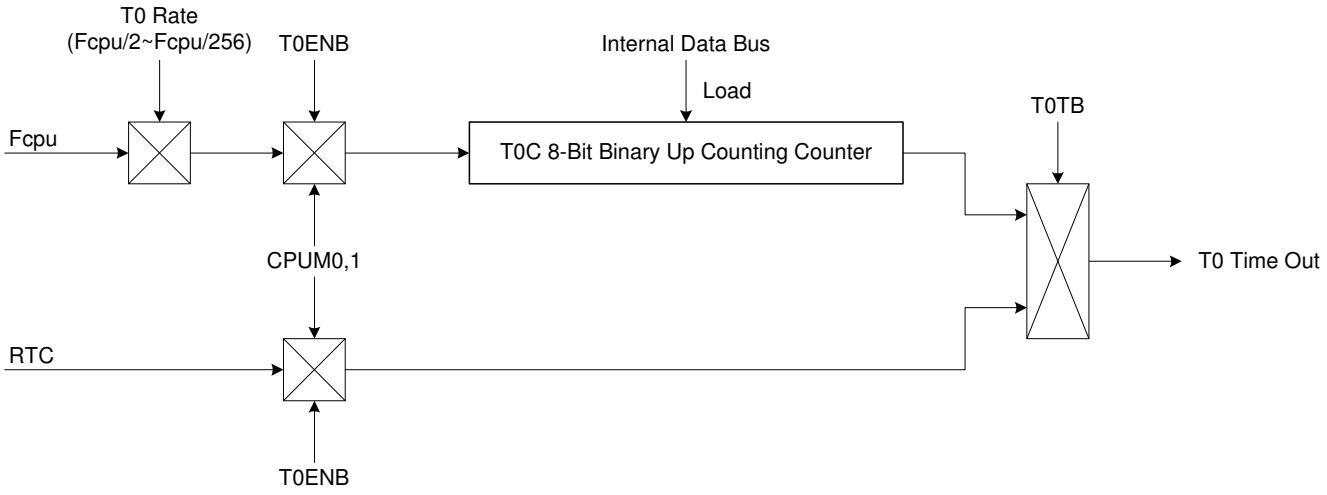
8.2 TIMER 0 (T0)

8.2.1 OVERVIEW

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

The main purposes of the T0 timer is as following.

- **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **RTC timer:** Generates interrupts at real time intervals based on the selected clock source. **RTC function is only available in T0TB=1.**
- **Green mode wakeup function:** T0 can be green mode wake-up time as T0ENB = 1. System will be wake-up by T0 time out.



> ***Note: In RTC mode, the T0 interval time is fixed at 0.5 sec and isn't controlled by T0C.***

8.2.2 T0M MODE REGISTER

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	TC1X8	TC0X8	-	T0TB
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 **T0TB:** RTC clock source control bit.
0 = Disable RTC (T0 clock source from Fcpu).
1 = Enable RTC, T0 will be 0.5 sec RTC (Low clock must be 32768 cyrstal).

Bit 2 **TC0X8:** TC0 internal clock source control bit.
0 = TC0 internal clock source is Fcpu. TC0RATE is from Fcpu/2~Fcpu/256.
1 = TC0 internal clock source is Fosc. TC0RATE is from Fosc/1~Fosc/128.

Bit 3 **TC1X8:** TC1 internal clock source control bit.
0 = TC1 internal clock source is Fcpu. TC1RATE is from Fcpu/2~Fcpu/256.
1 = TC1 internal clock source is Fosc. TC1RATE is from Fosc/1~Fosc/128.

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

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

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

8.2.3 T0C COUNTING REGISTER

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

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

The equation of T0C initial value is as following.

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

- **Example: To set 10ms interval time for T0 interrupt. High clock is external 4MHz. Fcpu=Fosc/4. Select T0RATE=010 (Fcpu/64).**

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

The basic timer table interval time of T0.

T0RATE	T0CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 1)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fcpu/256	65.536 ms	256 us	2000 ms	7812.5 us
001	Fcpu/128	32.768 ms	128 us	1000 ms	3906.25 us
010	Fcpu/64	16.384 ms	64 us	500 ms	1953.12 us
011	Fcpu/32	8.192 ms	32 us	250 ms	976.56 us
100	Fcpu/16	4.096 ms	16 us	125 ms	488.28 us
101	Fcpu/8	2.048 ms	8 us	62.5 ms	244.14 us
110	Fcpu/4	1.024 ms	4 us	31.25 ms	122.07 us
111	Fcpu/2	0.512 ms	2 us	15.625 ms	61.035 us

- ***Note: T0C is not available in RTC mode. The T0 interval time is fixed at 0.5 sec.***

8.2.4 T0 TIMER OPERATION SEQUENCE

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

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

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

☞ **Set T0 timer rate.**

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

☞ **Set T0 clock source from Fcpu or RTC.**

```
B0BCLR      FT0TB           ; Select T0 Fcpu clock source.
```

or

```
B0BSET      FT0TB           ; Select T0 RTC clock source.
```

☞ **Set T0 interrupt interval time.**

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

☞ **Set T0 timer function mode.**

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

☞ **Enable T0 timer.**

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

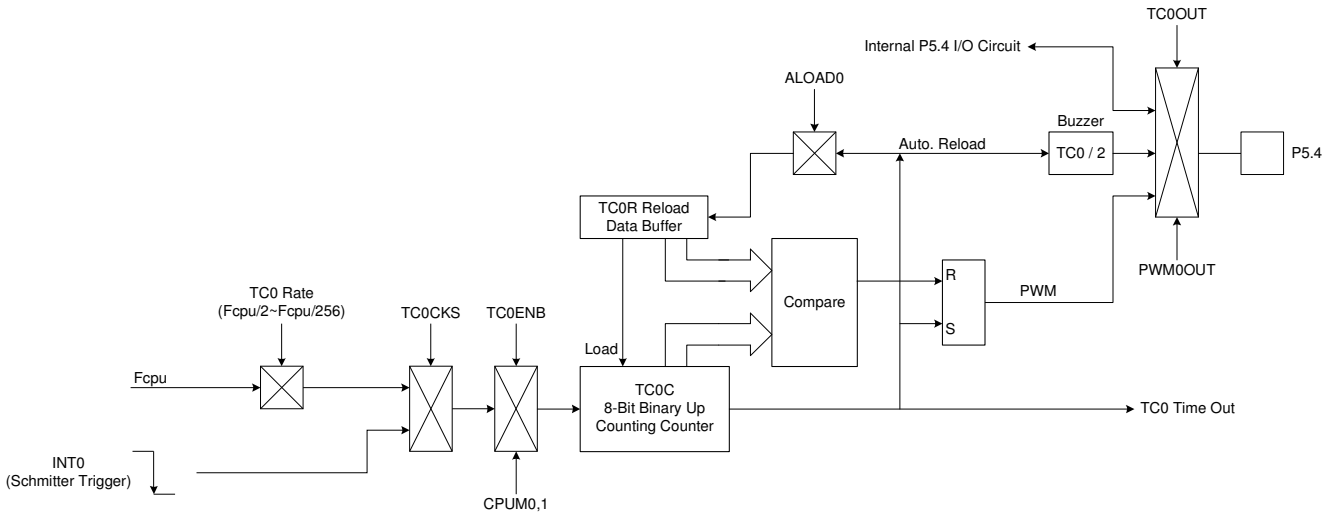
8.3 TIMER/COUNTER 0 (TC0)

8.3.1 OVERVIEW

The TC0 is an 8-bit binary up counting timer. TC0 has two clock sources including internal clock and external clock for counting a precision time. The internal clock source is from Fcpu. The external clock is INT0 from P0.0 pin (Falling edge trigger). Using TC0M register selects TC0C's clock source from internal or external. If TC0 timer occurs an overflow, it will continue counting and issue a time-out signal to trigger TC0 interrupt to request interrupt service. TC0 overflow time is 0xFF to 0X00 normally. Under PWM mode, TC0 overflow is still 256 counts.

The main purposes of the TC0 timer is as following.

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



8.3.2 TC0M MODE REGISTER

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

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

Bit 1 **TC0OUT:** TC0 time out toggle signal output control bit. **Only valid when PWM0OUT = 0.**
0 = Disable, P5.4 is I/O function.
1 = Enable, P5.4 is output TC0OUT signal.

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

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

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

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

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

8.3.3 TC0C COUNTING REGISTER

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

0DBH	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 * input clock)

➢ **Example: To set 10ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC0RATE=010 (Fcpu/64).**

*TC0C initial value = 256 - (TC0 interrupt interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 64)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 64)*
= 100
= 64H

The basic timer table interval time of TC0.

TC0RATE	TC0CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fcpu/256	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.125 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

✸ ***Note: TC0C can't be set as 0xFF when TC0 timer operating in interrupt, buzzer output modes. TC0C available range is 0x00~0xFE. The problem doesn't exist in pure PWM mode.***

8.3.4 TC0R AUTO-LOAD REGISTER

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

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

The equation of TC0R initial value is as following.

TC0R initial value = N - (TC0 interrupt interval time * input clock)

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

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

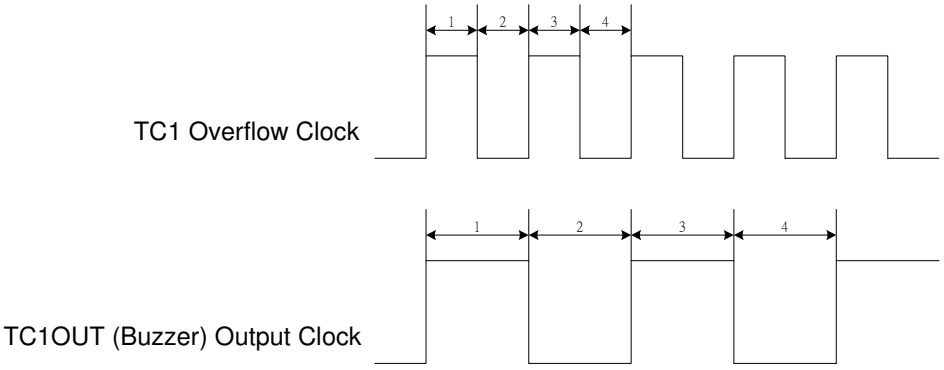
➢ **Example: To set 10ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=0) and no PWM output (PWM0=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC0RATE=010 (Fcpu/64).**

*TC0R initial value = N - (TC0 interrupt interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 64)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 64)*
= 100
= 64H

✸ ***Note: TC0R can't be set as 0xFF when TC0 timer operating in interrupt, buzzer output modes. TC0R available range is 0x00~0xFE. The problem doesn't exist in pure PWM mode.***

8.3.5 TC0 CLOCK FREQUENCY OUTPUT (BUZZER)

Buzzer output (TC0OUT) is from TC0 timer/counter frequency output function. By setting the TC0 clock frequency, the clock signal is output to P5.4 and the P5.4 general purpose I/O function is auto-disable. The TC0OUT frequency is divided by 2 from TC0 interval time. TC0OUT frequency is 1/2 TC0 frequency. The TC0 clock has many combinations and easily to make difference frequency. The TC0OUT frequency waveform is as following.



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

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

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

B0BSET    FTC0OUT         ; Enable TC0 output to P5.4 and disable P5.4 I/O function
B0BSET    FALOAD0         ; Enable TC0 auto-reload function
B0BSET    FTC0ENB         ; Enable TC0 timer
```

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

8.3.6 TC0 TIMER OPERATION SEQUENCE

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

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

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

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

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

☞ **Set TC0 timer clock source.**

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

☞ **Set TC0 timer auto-load mode.**

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

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

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

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

☞ **Set TC0 timer function mode.**

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

☞ **Enable TC0 timer.**

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

8.3.7 TC0 TIMER NOTICE

When TC0C value changes from “0xFF” to not “0xFF”, TC0IRQ is set “1” whether TC0 is operating or not. If TC0IRQ = 0 and TC0C is changed by program, TC0IRQ might be set as TC0C is from “0xFF” to not “0xFF”. The condition makes unexpected TC0 interrupt occurring.

➢ **Example: TC0C = 0xFF and TC0IRQ = 0. TC0IRQ will set as “1” when TC0C is cleared by program (TC0C = 0).**

```
        MOV         A, #0           ; Clear TC0C.
        B0MOV       TC0C, A         ; TC0IRQ changed from “0” to “1”.

        B0BSET      FTC0IEN         ; Enable TC0 interrupt function and system jumps to interrupt
                                    ; vector (ORG 8) at next cycle.
```

If TC0C changing in system operating duration is necessary, to disable TC0 interrupt function (TC0IEN = 0) before changing TC0C value. The solution can avoid unexpected TC0 interrupt occurring and example is as following.

➢ **Example: TC0C = 0xFF and TC0IRQ = 0. Clearing TC0C must be after TC0 interrupt disable.**

```
        B0BCLR      FTC0IEN         ; Disable TC0 interrupt function.

        MOV         A, #0           ; Clear TC0C.
        B0MOV       TC0C, A         ; TC0IRQ changed from “0” to “1”.

        B0BCLR      FTC0IRQ         ; Clear TC0IRQ flag.

        B0BSET      FTC0IEN         ; Enable TC0 interrupt function.
        …
        …
```

✸ ***Note: Disable TC0 interrupt function first, and load new TC0C value into TC0C buffer. This way can avoid unexpected TC0 interrupt occurring.***

✸ ***Note: TC0C and TC0R can’t be set as 0xFF when TC0 timer operating in interrupt, buzzer output modes. TC0C and TC0R available range is 0x00~0xFE. The problem doesn’t exist in pure PWM mode.***

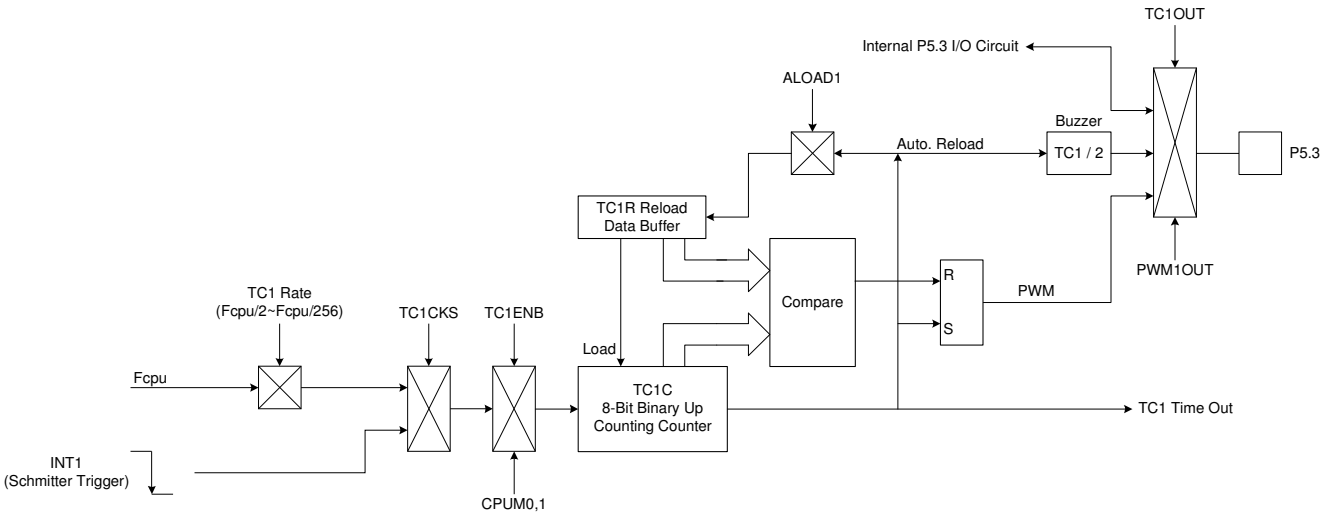
8.4 TIMER/COUNTER 1 (TC1)

8.4.1 OVERVIEW

The TC1 is an 8-bit binary up counting timer. TC1 has two clock sources including internal clock and external clock for counting a precision time. The internal clock source is from Fcpu. The external clock is INT1 from P0.1 pin (Falling edge trigger). Using TC1M register selects TC1C's clock source from internal or external. If TC1 timer occurs an overflow, it will continue counting and issue a time-out signal to trigger TC1 interrupt to request interrupt service. TC1 overflow time is 0xFF to 0X00 normally. Under PWM mode, TC1 overflow is still 256 counts.

The main purposes of the TC1 timer is as following.

- **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **External event counter:** Counts system "events" based on falling edge detection of external clock signals at the INT1 input pin.
- **Buzzer output**
- **PWM output**



8.4.2 TC1M MODE REGISTER

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

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

Bit 1 **TC1OUT:** TC1 time out toggle signal output control bit. **Only valid when PWM1OUT = 0.**
0 = Disable, P5.3 is I/O function.
1 = Enable, P5.3 is output TC1OUT signal.

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

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

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

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

✸ ***Note: When TC1CKS=1, TC1 became an external event counter and TC1RATE is useless. No more P0.1 interrupt request will be raised. (P0.1IRQ will be always 0).***

8.4.3 TC1C COUNTING REGISTER

TC1C is an 8-bit counter register for TC1 interval time control.

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

The equation of TC1C initial value is as following.

TC1C initial value = 256 - (TC1 interrupt interval time * input clock)

- **Example: To set 10ms interval time for TC1 interrupt. TC1 clock source is Fcpu (TC1KS=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC1RATE=010 (Fcpu/64).**

*TC1C initial value = 256 - (TC1 interrupt interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 64)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 64)*
= 100
= 64H

The basic timer table interval time of TC1.

TC1RATE	TC1CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fcpu/256	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.125 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

✷ ***Note: TC1C and TC1R can't be set as 0xFF when TC1 timer operating in interrupt, buzzer output modes. TC1C and TC1R available range is 0x00~0xFE. The problem doesn't exist in pure PWM mode.***

8.4.4 TC1R AUTO-LOAD REGISTER

TC1 timer is with auto-load function controlled by ALOAD1 bit of TC1M. When TC1C overflow occurring, TC1R value will load to TC1C by system. It is easy to generate an accurate time, and users don't reset TC1C during interrupt service routine.

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

The equation of TC1R initial value is as following.

TC1R initial value = N - (TC1 interrupt interval time * input clock)

N is TC1 overflow boundary number. TC1 timer overflow time has five types (TC1 timer, TC1 event counter, TC1 Fcpu clock source, PWM mode and no PWM mode). These parameters decide TC1 overflow time and valid value as follow table.

TC1CKS	PWM1	ALOAD1	TC1OUT	N	TC1R valid value	TC1R value binary type
0	0	x	x	256	0x00~0xFF	00000000b~11111111b
	1	0	0	256	0x00~0xFF	00000000b~11111111b
	1	0	1	64	0x00~0x3F	xx000000b~xx111111b
	1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
	1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
1	-	-	-	256	0x00~0xFF	00000000b~11111111b

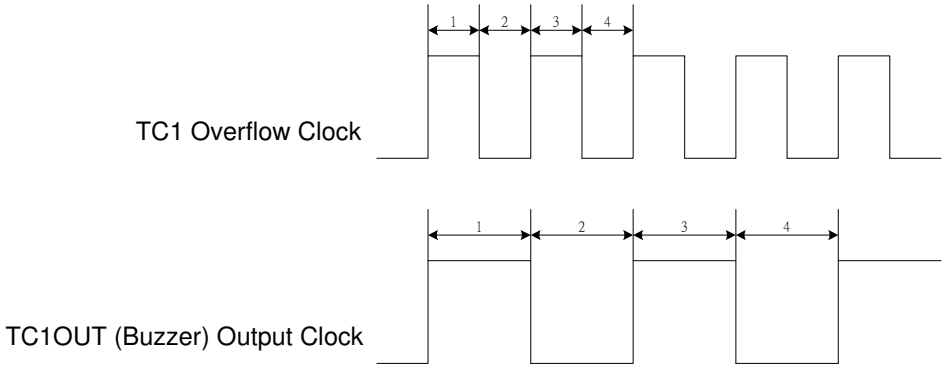
➢ **Example: To set 10ms interval time for TC1 interrupt. TC1 clock source is Fcpu (TC1KS=0) and no PWM output (PWM1=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC1RATE=010 (Fcpu/64).**

*TC1R initial value = N - (TC1 interrupt interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 64)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 64)*
= 100
= 64H

✸ ***Note: TC1R can't be set as 0xFF when TC1 timer operating in interrupt, buzzer output modes. TC1R available range is 0x00~0xFE. The problem doesn't exist in pure PWM mode.***

8.4.5 TC1 CLOCK FREQUENCY OUTPUT (BUZZER)

Buzzer output (TC1OUT) is from TC1 timer/counter frequency output function. By setting the TC1 clock frequency, the clock signal is output to P5.3 and the P5.3 general purpose I/O function is auto-disable. The TC1OUT frequency is divided by 2 from TC1 interval time. TC1OUT frequency is 1/2 TC1 frequency. The TC1 clock has many combinations and easily to make difference frequency. The TC1OUT frequency waveform is as following.



➢ **Example: Setup TC1OUT output from TC1 to TC1OUT (P5.3). The external high-speed clock is 4MHz. The TC1OUT frequency is 0.5KHz. Because the TC1OUT signal is divided by 2, set the TC1 clock to 1KHz. The TC1 clock source is from external oscillator clock. TC1 rate is Fcpu/4. The TC1RATE2~TC1RATE1 = 110. TC1C = TC1R = 131.**

```
MOV       A,#01100000B
B0MOV     TC1M,A          ; Set the TC1 rate to Fcpu/4

MOV       A,#131          ; Set the auto-reload reference value
B0MOV     TC1C,A
B0MOV     TC1R,A

B0BSET    FTC1OUT         ; Enable TC1 output to P5.3 and disable P5.3 I/O function
B0BSET    FALOAD1         ; Enable TC1 auto-reload function
B0BSET    FTC1ENB         ; Enable TC1 timer
```

✸ ***Note: Buzzer output is enabled, and "PWM1OUT" must be "0".***

8.4.6 TC1 TIMER OPERATION SEQUENCE

TC1 timer operation includes timer interrupt, event counter, TC1OUT and PWM. The sequence of setup TC1 timer is as following.

☞ **Stop TC1 timer counting, disable TC1 interrupt function and clear TC1 interrupt request flag.**

```
B0BCLR      FTC1ENB     ; TC1 timer, TC1OUT and PWM stop.
B0BCLR      FTC1IEN     ; TC1 interrupt function is disabled.
B0BCLR      FTC1IRQ     ; TC1 interrupt request flag is cleared.
```

☞ **Set TC1 timer rate. (Besides event counter mode.)**

```
MOV         A, #0xxx0000b   ;The TC1 rate control bits exist in bit4~bit6 of TC1M. The
                            ; value is from x000xxxxb~x111xxxxb.
B0MOV       TC1M,A          ; TC1 timer is disabled.
```

☞ **Set TC1 timer clock source.**

```
; Select TC1 internal / external clock source.
            B0BCLR      FTC1CKS     ; Select TC1 internal clock source.
or
            B0BSET      FTC1CKS     ; Select TC1 external clock source.
```

☞ **Set TC1 timer auto-load mode.**

```
            B0BCLR      FALOAD1     ; Enable TC1 auto reload function.
or
            B0BSET      FALOAD1     ; Disable TC1 auto reload function.
```

☞ **Set TC1 interrupt interval time, TC1OUT (Buzzer) frequency or PWM duty cycle.**

```
; Set TC1 interrupt interval time, TC1OUT (Buzzer) frequency or PWM duty.
            MOV         A,#7FH      ; TC1C and TC1R value is decided by TC1 mode.
            B0MOV       TC1C,A      ; Set TC1C value.
            B0MOV       TC1R,A      ; Set TC1R value under auto reload mode or PWM mode.

; In PWM mode, set PWM cycle.
            B0BCLR      FALOAD1     ; ALOAD1, TC1OUT = 00, PWM cycle boundary is 0~255.
            B0BCLR      FTC1OUT
or
            B0BCLR      FALOAD1     ; ALOAD1, TC1OUT = 01, PWM cycle boundary is 0~63.
            B0BSET      FTC1OUT
or
            B0BSET      FALOAD1     ; ALOAD1, TC1OUT = 10, PWM cycle boundary is 0~31.
            B0BCLR      FTC1OUT
or
            B0BSET      FALOAD1     ; ALOAD1, TC1OUT = 11, PWM cycle boundary is 0~15.
            B0BSET      FTC1OUT
```

☞ **Set TC1 timer function mode.**

```
        B0BSET      FTC1IEN         ; Enable TC1 interrupt function.
or
        B0BSET      FTC1OUT         ; Enable TC1OUT (Buzzer) function.
or
        B0BSET      FPWM1OUT        ; Enable PWM function.
```

☞ **Enable TC1 timer.**

```
        B0BSET      FTC1ENB         ; Enable TC1 timer.
```

8.4.7 TC1 TIMER NOTICE

When TC1C value changes from “0xFF” to not “0xFF”, TC1IRQ is set “1” whether TC1 is operating or not. If TC1IRQ = 0 and TC1C is changed by program, TC1IRQ might be set as TC1C is from “0xFF” to not “0xFF”. The condition makes unexpected TC1 interrupt occurring.

➢ **Example: TC1C = 0xFF and TC1IRQ = 0. TC1IRQ will set as “1” when TC1C is cleared by program (TC1C = 0).**

```
        MOV         A, #0           ; Clear TC1C.
        B0MOV       TC1C, A         ; TC1IRQ changed from “0” to “1”.

        B0BSET      FTC1IEN         ; Enable TC1 interrupt function and system jumps to interrupt
                                    ; vector (ORG 8) at next cycle.
```

If TC1C changing in system operating duration is necessary, to disable TC1 interrupt function (TC1IEN = 0) before changing TC1C value. The solution can avoid unexpected TC1 interrupt occurring and example is as following.

➢ **Example: TC1C = 0xFF and TC1IRQ = 0. Clearing TC1C must be after TC1 interrupt disable.**

```
        B0BCLR      FTC1IEN         ; Disable TC1 interrupt function.

        MOV         A, #0           ; Clear TC1C.
        B0MOV       TC1C, A         ; TC1IRQ changed from “0” to “1”.

        B0BCLR      FTC1IRQ         ; Clear TC1IRQ flag.

        B0BSET      FTC1IEN         ; Enable TC1 interrupt function.
        …
        …
```

✸ ***Note: Disable TC1 interrupt function first, and load new TC1C value into TC1C buffer. This way can avoid unexpected TC1 interrupt occurring.***

✸ ***Note: TC1C and TC1R can’t be set as 0xFF when TC1 timer operating in interrupt, buzzer output modes. TC1C and TC1R available range is 0x00~0xFE. The problem doesn’t exist in pure PWM mode.***

8.5 PWM0 MODE

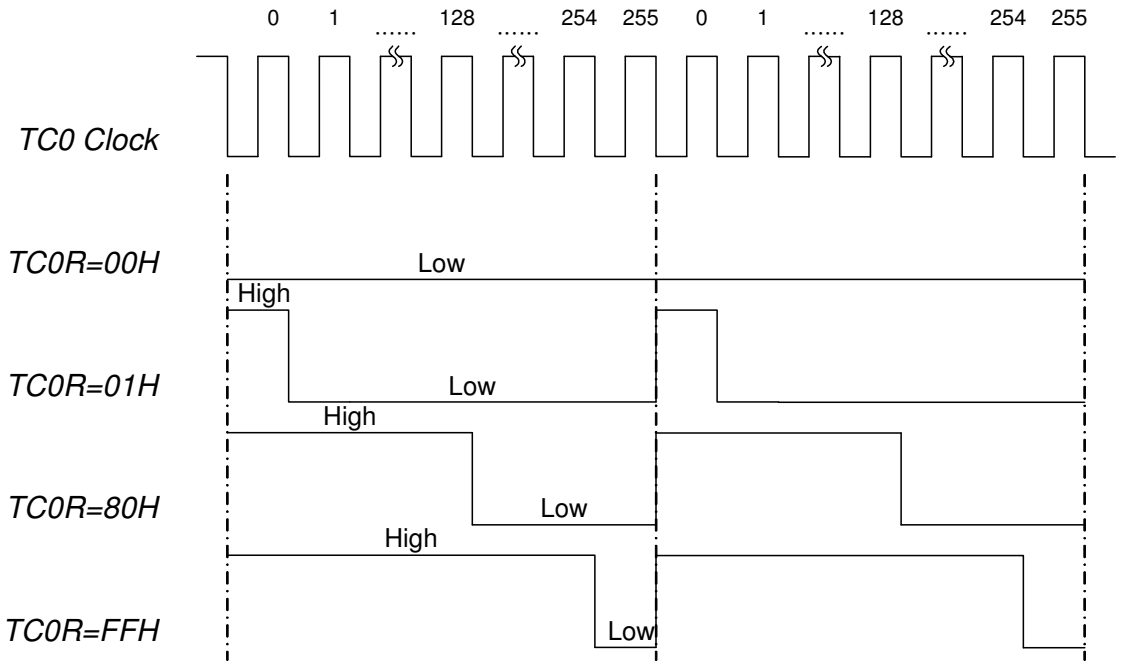
8.5.1 OVERVIEW

PWM function is generated by TC0 timer counter and output the PWM signal to PWM0OUT pin (P5.4). The 8-bit counter counts modulus 256, 64, 32, 16 controlled by ALOAD0, TC0OUT bits. The value of the 8-bit counter (TC0C) is compared to the contents of the reference register (TC0R). When the reference register value (TC0R) is equal to the counter value (TC0C), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWM0 output is TC0R/256, 64, 32, 16.

✷ ***Note: TC0C and TC0R can be 0xFF in pure PWM output. If PWM function is operating with TC0 interrupt, TC0C and TC0R can't be set as 0xFF and the available range is 0x00~0xFE.***

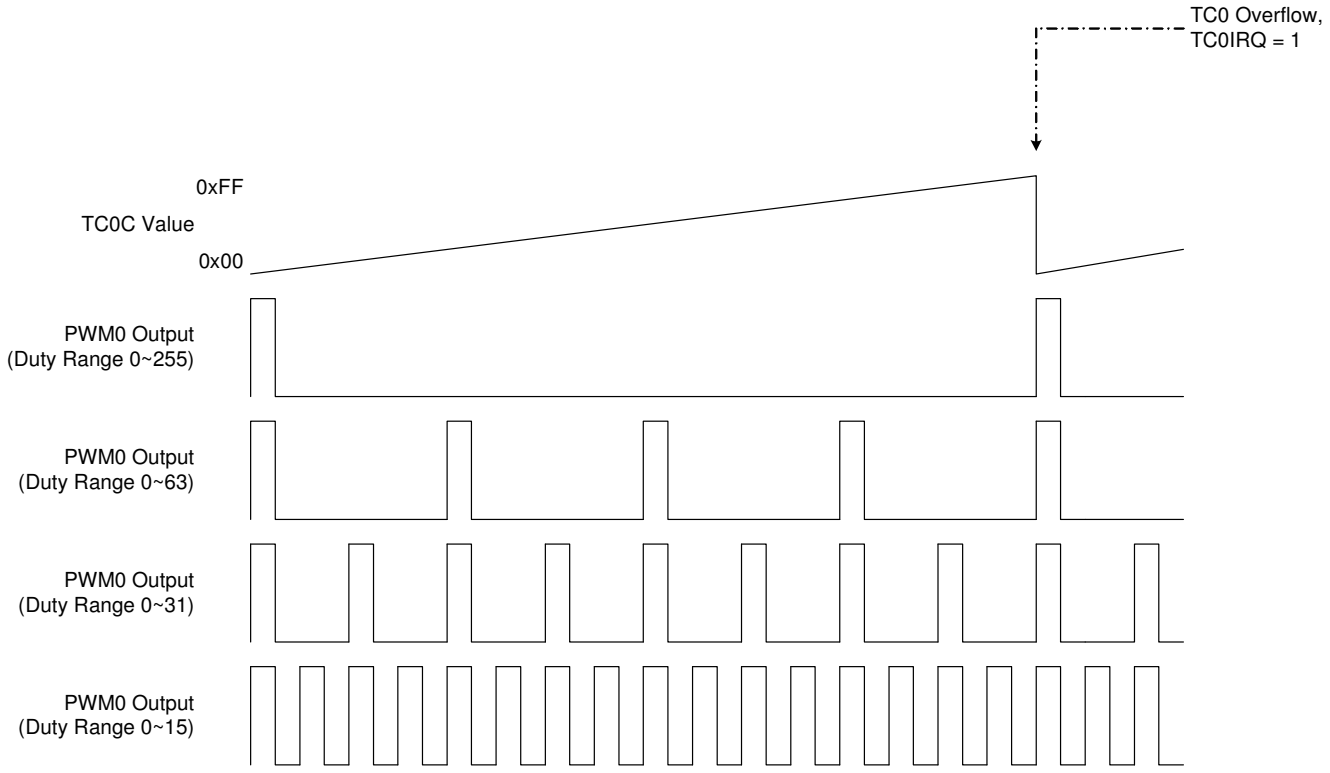
ALOAD0	TC0OUT	PWM duty range	TC0C valid value	TC0R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0	0	0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count
0	1	0/64~63/64	0x00~0x3F	0x00~0x3F	31.25K	Overflow per 64 count
1	0	0/32~31/32	0x00~0x1F	0x00~0x1F	62.5K	Overflow per 32 count
1	1	0/16~15/16	0x00~0x0F	0x00~0x0F	125K	Overflow per 16 count

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



8.5.2 TC0IRQ AND PWM DUTY

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



8.5.3 PWM PROGRAM EXAMPLE

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

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

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

B0BCLR  FTC0OUT         ; Set duty range as 0/256~255/256.
B0BCLR  FALOAD0
B0BSET  FPWM0OUT        ; Enable PWM0 output to P5.4 and disable P5.4 I/O function
B0BSET  FTC0ENB         ; Enable TC0 timer
```

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

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

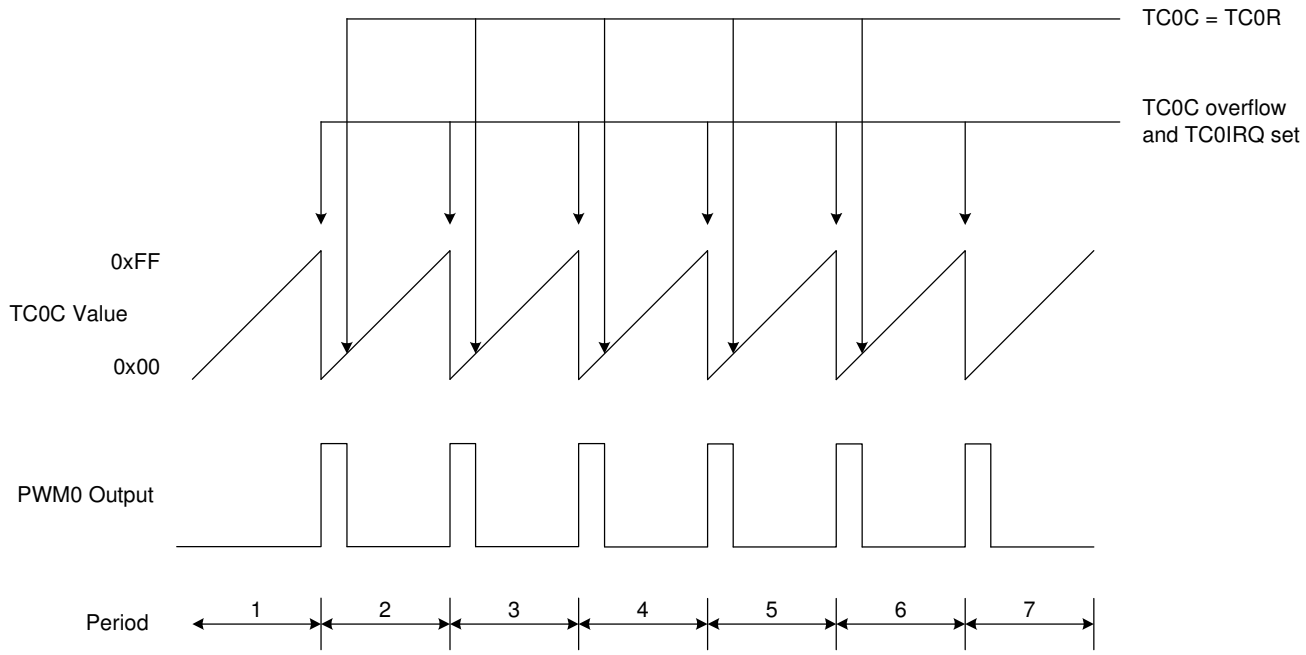
```
MOV     A, #30H         ; Input a number using B0MOV instruction.
B0MOV   TC0R, A

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

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

8.5.4 PWM0 DUTY CHANGING NOTICE

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

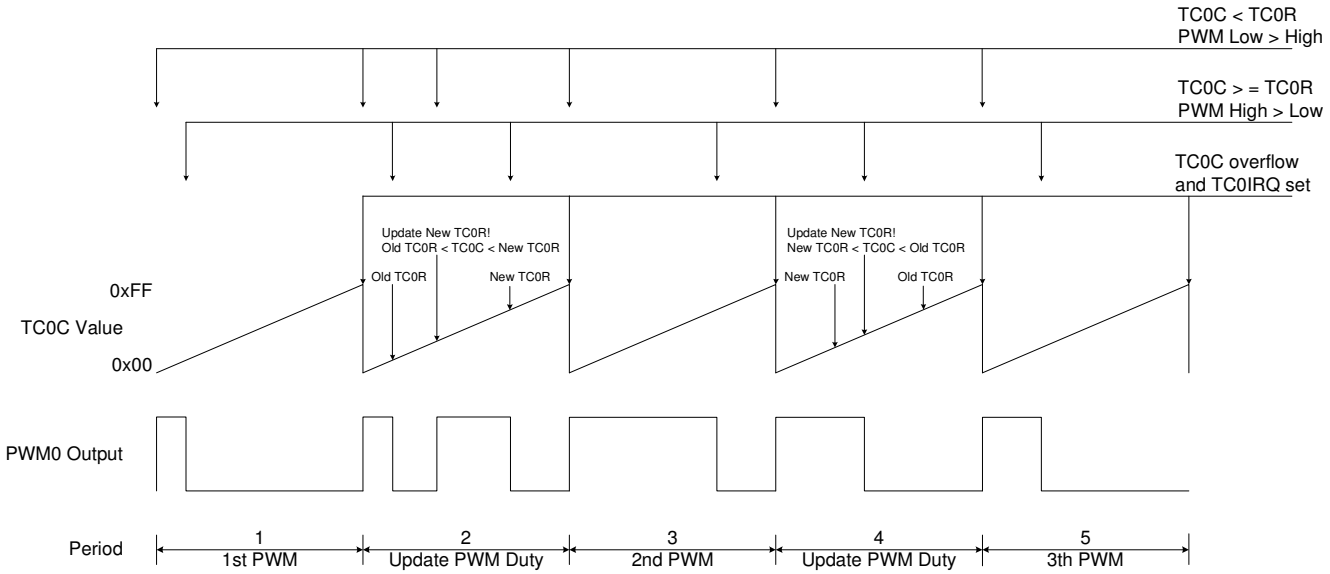


Above diagram is shown the waveform with fixed TC0R. In every TC0C overflow PWM output "High, when TC0C≧ TC0R PWM output "Low".

✸ ***Note: Setting PWM duty in program processing must be at the new cycle start.***

✸ ***Note: TC0C and TC0R can be 0xFF in pure PWM output. If PWM function is operating with TC0 interrupt, TC0C and TC0R can't be set as 0xFF and the available range is 0x00~0xFE.***

If TC0R is changing in the program processing, the PWM waveform will became as following diagram.



In period 2 and period 4, new Duty (TC0R) is set, but the PWM output waveform of period 2 and period 4 are wrong. In period 2, the new TC0R value is greater than old TC0R value. If setting new TC0R is after PWM output “low”, system is getting TC0C < TC0R result and making PWM output “high”. There are two high level periods in the cycle, and the waveform is unexpected. Until next cycle, PWM outputs correct duty. In period 4, the new TC0R value is smaller than the old TC0R value. If setting new TC0R is before PWM output “low”, system is getting TC0C≧TC0R result and making PWM output “low”. In the cycle, the high duty is shorter than last cycle and longer than correct cycle. It is an unexpected PWM output.

Though the wrong waveforms only exist in one cycle, it is still a problem for precise PWM application and might make outside loading operations error. The solution is to load new TC0R after TC0 timer overflow. Using TC0IRQ status to determine TC0 timer is overflow or not. When TC0IRQ becomes “1”, to set the new TC0R value into TC0R buffer, and the unexpected PWM output is resolved.

➢ **Example: Using TC0 interrupt function to set new TC0R value for changing PWM duty.**

```
MAIN:
        …
        B0MOV       TC0RBUF, A      ; Load new PWM duty setting value into TC0RBUF.
        …
        …

INT_SER:
        …                           ; Push routine to save ACC and PFLAG to buffers.
        …
        B0BTS1      FTC0IRQ
        JMP         INT_SER90
        B0MOV       A, TC0RBUF      ; When TC0 Interrupt occurs, update TC0R.
        B0MOV       TC0R, A
        …
        …
INT_SER90:
        …                           ; Pop routine to load ACC and PFLAG from buffers.
        RETI
```

8.6 PWM1 MODE

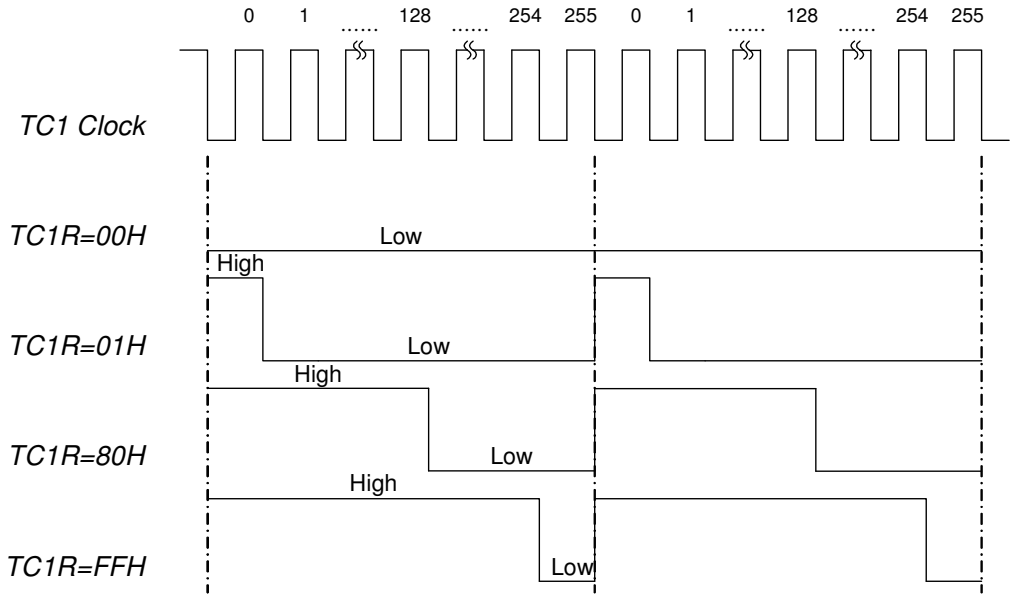
8.6.1 OVERVIEW

PWM function is generated by TC1 timer counter and output the PWM signal to PWM1OUT pin (P5.3). The 8-bit counter counts modulus 256, 64, 32, 16 controlled by ALOAD1, TC1OUT bits. The value of the 8-bit counter (TC1C) is compared to the contents of the reference register (TC1R). When the reference register value (TC1R) is equal to the counter value (TC1C), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWM1 output is TC1R/256, 64, 32, 16.

✸ ***Note: TC1C and TC1R can be 0xFF in pure PWM output. If PWM function is operating with TC1 interrupt, TC1C and TC1R can't be set as 0xFF and the available range is 0x00~0xFE.***

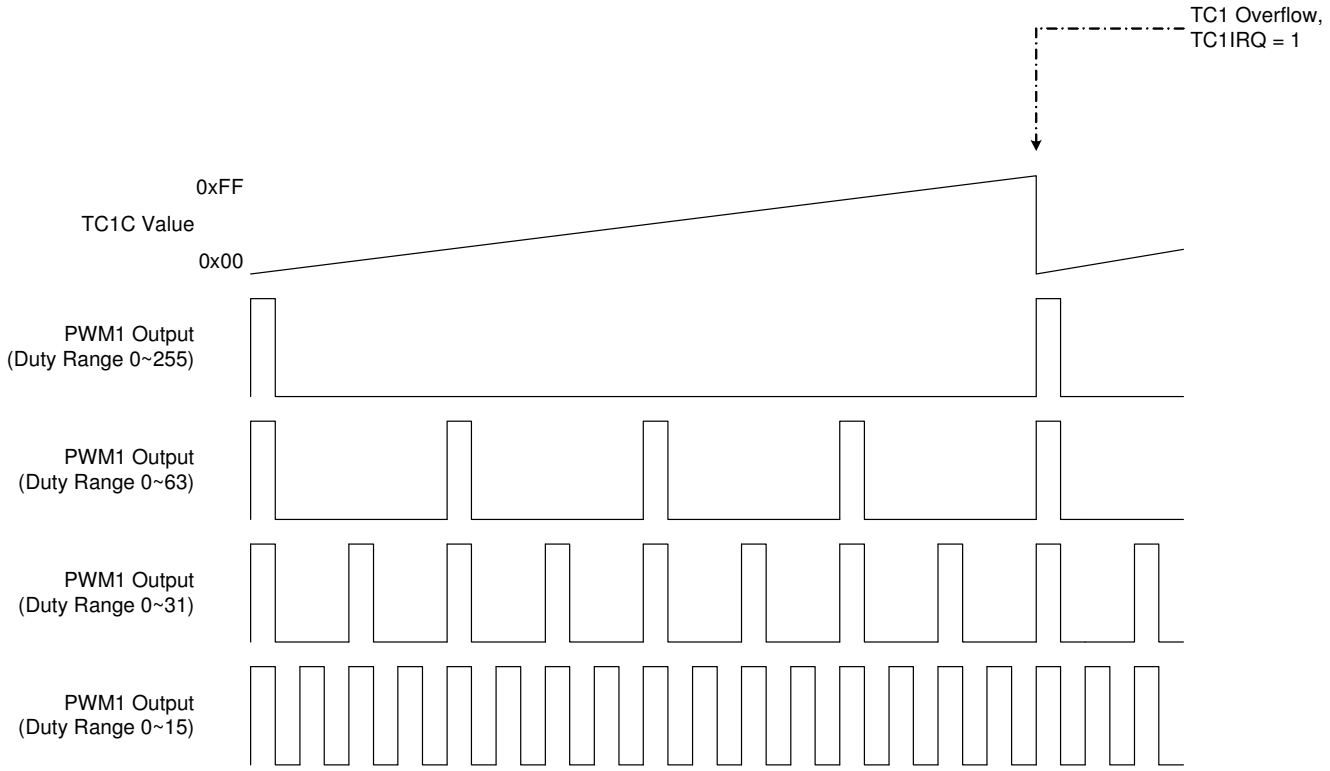
ALOAD1	TC1OUT	PWM duty range	TC1C valid value	TC1R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0	0	0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count
0	1	0/64~63/64	0x00~0x3F	0x00~0x3F	31.25K	Overflow per 64 count
1	0	0/32~31/32	0x00~0x1F	0x00~0x1F	62.5K	Overflow per 32 count
1	1	0/16~15/16	0x00~0x0F	0x00~0x0F	125K	Overflow per 16 count

The Output duty of PWM is with different TC1R. Duty range is from 0/256~255/256.



8.6.2 TC1IRQ AND PWM DUTY

In PWM mode, the frequency of TC1IRQ is depended on PWM duty range. From following diagram, the TC1IRQ frequency is related with PWM duty.



8.6.3 PWM PROGRAM EXAMPLE

➢ **Example: Setup PWM1 output from TC1 to PWM1OUT (P5.3). The external high-speed oscillator clock is 4MHz. Fcpu = Fosc/4. The duty of PWM is 30/256. The PWM frequency is about 1KHz. The PWM clock source is from external oscillator clock. TC1 rate is Fcpu/4. The TC1RATE2~TC1RATE1 = 110. TC1C = TC1R = 30.**

```
MOV     A,#01100000B
B0MOV   TC1M,A          ; Set the TC1 rate to Fcpu/4

MOV     A,#30           ; Set the PWM duty to 30/256
B0MOV   TC1C,A
B0MOV   TC1R,A

B0BCLR  FTC1OUT         ; Set duty range as 0/256~255/256.
B0BCLR  FALOAD1
B0BSET  FPWM1OUT        ; Enable PWM1 output to P5.3 and disable P5.3 I/O function
B0BSET  FTC1ENB         ; Enable TC1 timer
```

✸ ***Note: The TC1R is write-only register. Don't process them using INCMS, DECMS instructions.***

➢ **Example: Modify TC1R registers' value.**

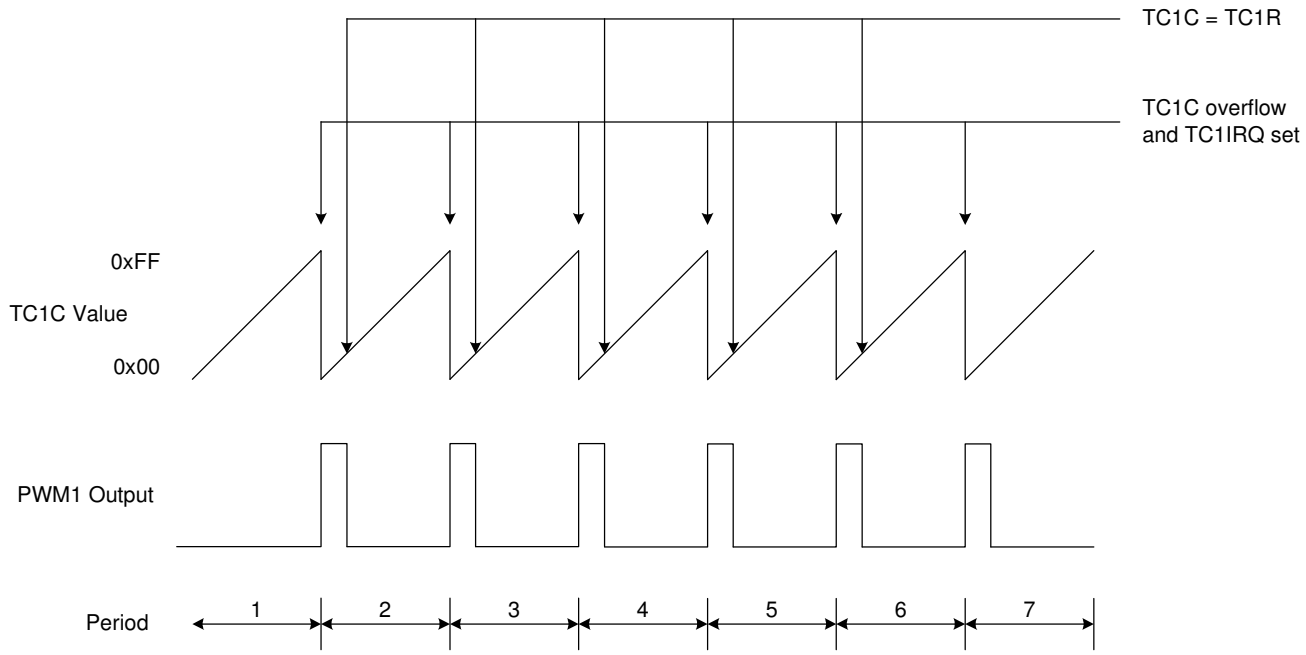
```
MOV     A, #30H         ; Input a number using B0MOV instruction.
B0MOV   TC1R, A

INCMS   BUF0            ; Get the new TC1R value from the BUF0 buffer defined by
NOP                     ; programming.
B0MOV   A, BUF0
B0MOV   TC1R, A
```

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

8.6.4 PWM1 DUTY CHANGING NOTICE

In PWM mode, the system will compare TC1C and TC1R all the time. When TC1C<TC1R, the PWM will output logic "High", when TC1C≧TC1R, the PWM will output logic "Low". If TC1C is changed in certain period, the PWM duty will change immediately. If TC1R is fixed all the time, the PWM waveform is also the same.

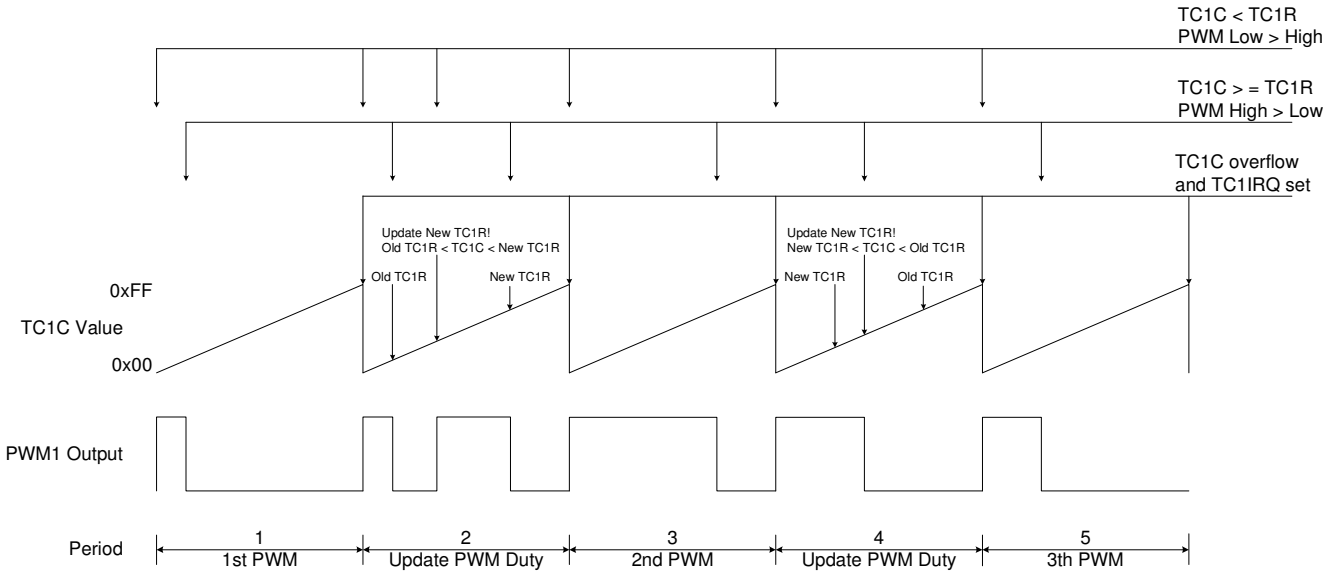


Above diagram is shown the waveform with fixed TC1R. In every TC1C overflow PWM output "High, when TC1C≧ TC1R PWM output "Low".

✸ ***Note: Setting PWM duty in program processing must be at the new cycle start.***

✸ ***Note: TC1C and TC1R can be 0xFF in pure PWM output. If PWM function is operating with TC1 interrupt, TC1C and TC1R can't be set as 0xFF and the available range is 0x00~0xFE.***

If TC1R is changing in the program processing, the PWM waveform will became as following diagram.



In period 2 and period 4, new Duty (TC1R) is set, but the PWM output waveform of period 2 and period 4 are wrong. In period 2, the new TC1R value is greater than old TC1R value. If setting new TC1R is after PWM output “low”, system is getting TC1C < TC1R result and making PWM output “high”. There are two high level periods in the cycle, and the waveform is unexpected. Until next cycle, PWM outputs correct duty. In period 4, the new TC1R value is smaller than the old TC1R value. If setting new TC1R is before PWM output “low”, system is getting TC1C≧TC1R result and making PWM output “low”. In the cycle, the high duty is shorter than last cycle and longer than correct cycle. It is an unexpected PWM output.

Though the wrong waveforms only exist in one cycle, it is still a problem for precise PWM application and might make outside loading operations error. The solution is to load new TC1R after TC1 timer overflow. Using TC1IRQ status to determine TC1 timer is overflow or not. When TC1IRQ becomes “1”, to set the new TC1R value into TC1R buffer, and the unexpected PWM output is resolved.

➢ **Example: Using TC1 interrupt function to set new TC1R value for changing PWM duty.**

```
MAIN:
        …
        B0MOV       TC1RBUF, A      ; Load new PWM duty setting value into TC1RBUF.
        …
        …

INT_SER:
        …                           ; Push routine to save ACC and PFLAG to buffers.
        …
        B0BTS1      FTC1IRQ
        JMP         INT_SER90
        B0MOV       A, TC1RBUF      ; When TC1 Interrupt occurs, update TC1R.
        B0MOV       TC1R, A
        …
        …
INT_SER90:
        …                           ; Pop routine to load ACC and PFLAG from buffers.
        RETI
```

9 SERIAL INPUT/OUTPUT TRANSCEIVER (SIO)

9.1 OVERVIEW

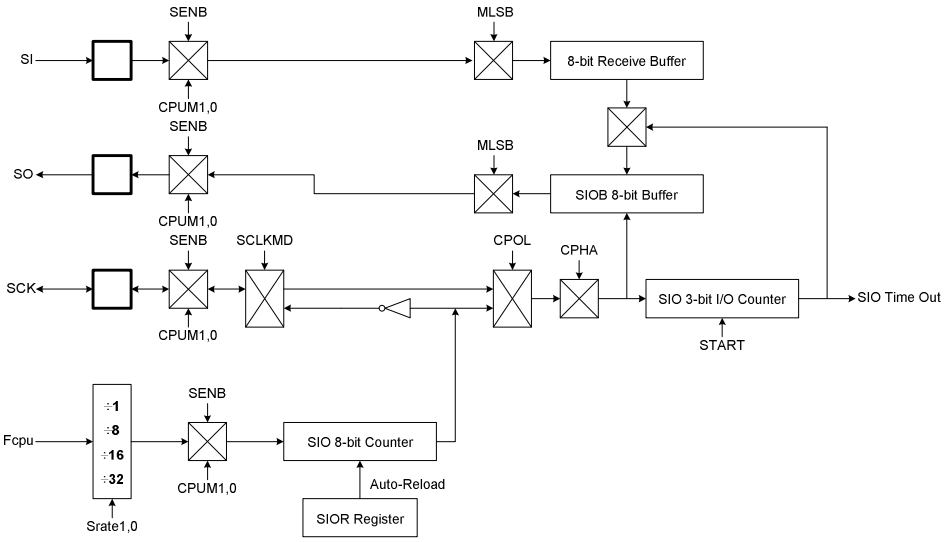
The SIO (serial input/output) transceiver is a serial communicate interface for data exchanging from one MCU to one MCU or other hardware peripherals. It is a simple 8-bit interface without a major definition of protocol, packet or control bits. The SIO transceiver includes three pins, clock (SCK), data input (SI) and data output (SO) 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, SI, SO are programmable open-drain output pin for multiple salve devices application.**
- **Two programmable bit rates (Only in master mode).**
- **End-of-Transfer interrupt.**

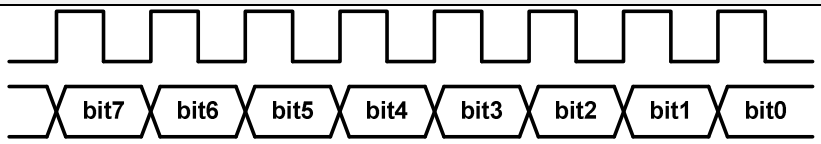
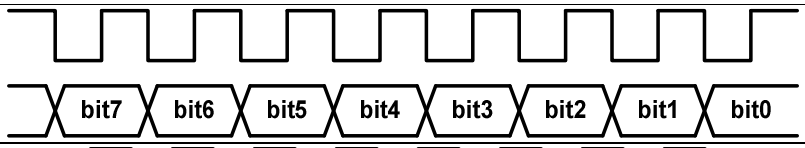
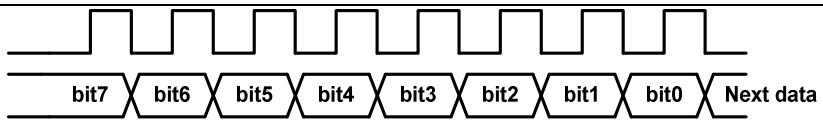
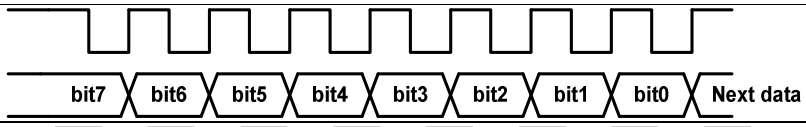
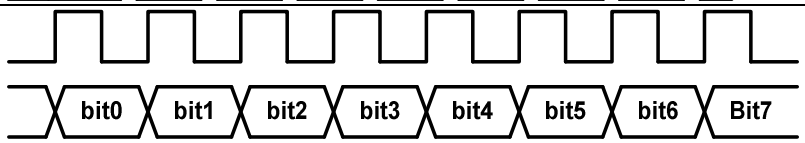
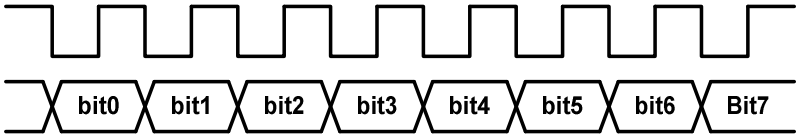
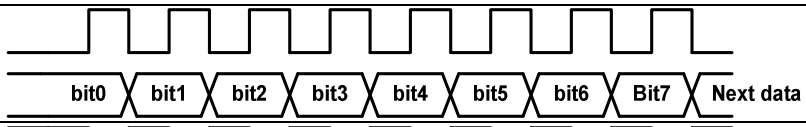
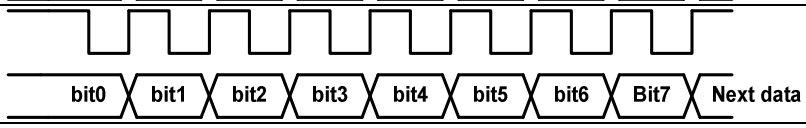
9.2 SIO OPERATION

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 Circuit 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 8) 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.***

9.3 SIOM MODE REGISTER

SIOM initial value = 0000 0000

0B4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOM	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **SENB:** SIO function control bit.
0 = Disable SIO function. P5.0~P5.2 are GPIO.
1 = Enable SIO function. P5.0~P5.2 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.

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

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

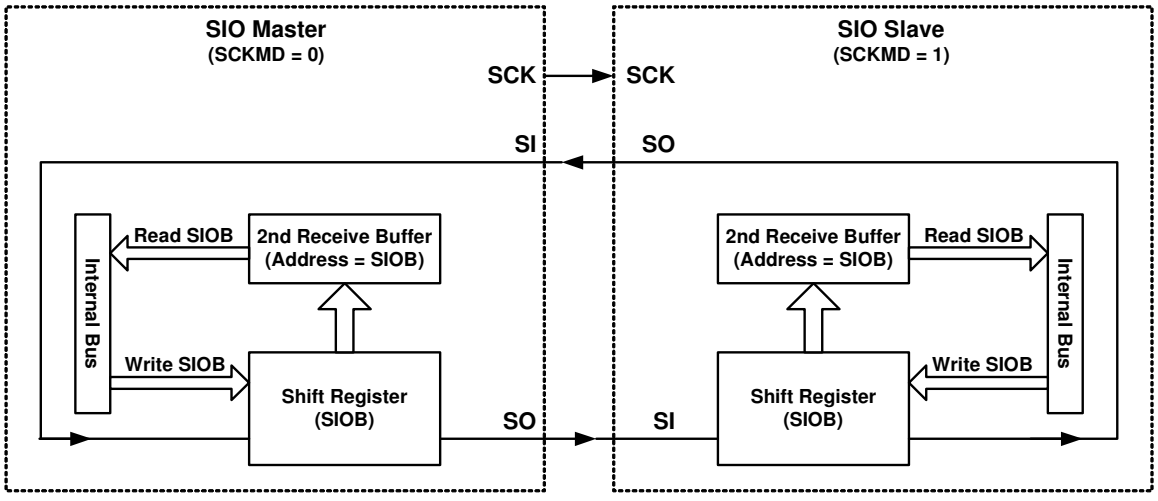
✷ ***Note:***
1. ***If SCKMD=1 for external clock, the SIO is in SLAVE mode. If SCKMD=0 for internal clock, the SIO is in MASTER mode.***
2. ***Don't set SENB and START bits in the same time. That makes the SIO function error.***
3. ***SIO pin can be push-pull structure and open-drain structure controlled by P1OC register.***

9.4 SIOB DATA BUFFER

SIOB initial value = 0000 0000

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOB	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

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

9.5 SIOR REGISTER DESCRIPTION

SIOR initial value = 0000 0000

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

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

SCK frequency = SIO rate / (256 - SIOR)

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

- **Example: Setup the SIO clock to be 5KHz. Fosc = 3.58MHz. SIO's rate = Fcpu = Fosc/4.**

SIOR = 256 – (1/(5KHz) * 3.58MHz/4)
= 256 – (0.0002*895000)
= 256 – 179
= 77

10 MAIN SERIAL PORT (MSP)

10.1 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)**
- **Support Single mast, mutiplex slave.**

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

Note. Use ICE emulation MSP function, reference Ch15.1.1

10.2 MSP STATUS REGISTER

MSPSTAT initial value = X000 00X0

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPSTAT	-	CKE	D_A	P	S	RED_WRT	-	BF
Read/Write	-	R/W	R	R	R	R	-	R
After reset	-	0	0	0	0	0	-	0

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

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

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

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

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

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

Note1. It will be cleared when STOP bit was detected.

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

This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next start bit, stop bit, or not ACK bit.
In slave mode:
0 = Write.
1 = Read.
In master mode:
0 = Transmit is not in progress.
1 = Transmit is in progress.
Or this bit with SEN, RSEN, PEN, RCEN, or ACKEN will indicate if the MSP is in IDLE mode.

Bit 0 **BF:** Buffer Full Status bit
Receive
1 = Receive complete, MSPBUF is full.
0 = Receive not complete, MSPBUF is empty.
Transmit
1 = Data Transmit in progress (does not include the ACK and stop bits), MSPBUF is full.
0 = Data Transmit complete (does not include the ACK and stop bits), MSPBUF is empty.

10.3 MSP MODE REGISTER 1

MSPM1 initial value = 0000 00X0

091H	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

Note1. After MSP Disable or Enable,must delay 2 instruction cycle.

```
Ex: b0bset FMSPENB
    NOP
    NOP
    ....
Ex:b0bclr FMSPENB
   NOP
   NOP
   ....
```

Note2.MSP status register will be clear after MSP Disable. So,user should setting MSP register again befort MSP Enable.

```
Ex:Call MSP_init_setting
    B0bset FMSPENB
    NOP
```

```
NOP
```

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

Note1. 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.

Note1.If MSP want to operated on Master mode,MSPC must be setting 1 after MSPENB.

```
Ex:b0bset MSPENB
   NOP
   NOP
   B0bset FMSPC
```

Note1.If MSP want to operated on slave mode,MSPC must be setting 1 before MSPENB.

```
Ex:b0bclr FMSPC
   b0bset MSPENB
   NOP
   NOP
   ....
```

10.4 MSP MODE REGISTER 2

MSPM2 initial value = 0000 0000

092H	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.

10.5 MSP MSPBUF REGISTER

MSPBUF initial value = 0000 0000

093H	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

10.6 MSP MSPADR REGISTER

MSPADR initial value = 0000 0000

094H	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.

10.7 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 condition 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 Sofrware.

10.7.1 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 MSPADDR 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

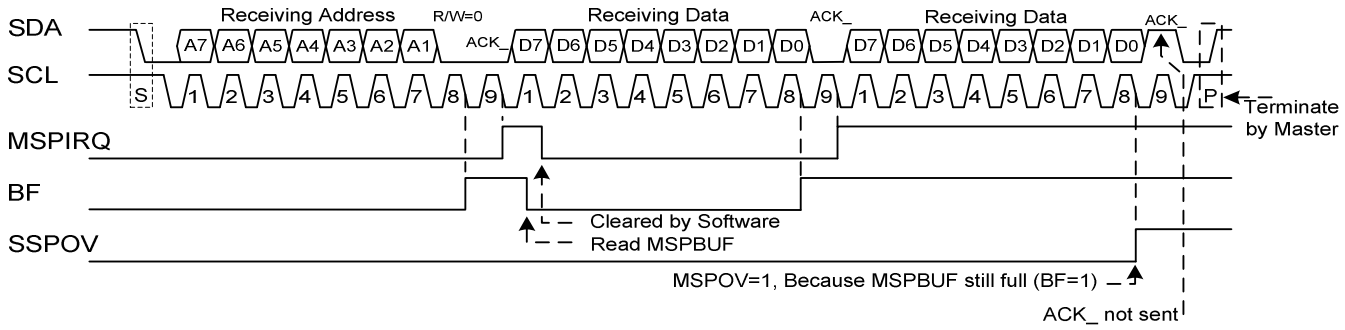
Note1. BF=0, MSPOV=1 shows the software is not set properly to clear Overflow register.

10.7.2 Slave Receiving

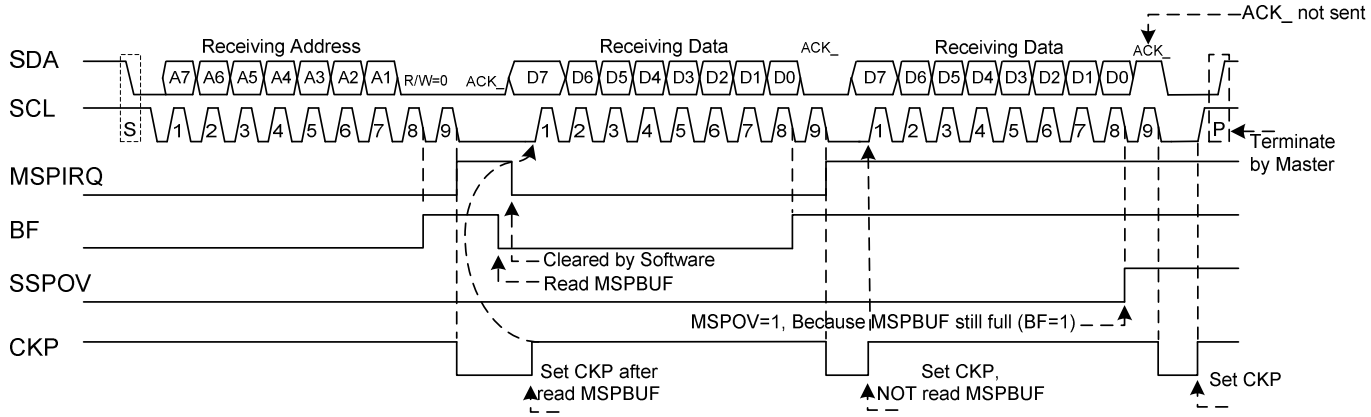
When the R/W bit of address byte =0 and address is matched, the R/W bit of MSPSTAT is cleared. The address will be load into MSPBUF. After reply an ACK_ signal, MSP will receive data every 8 clock. The CKP function enable or disable (Default) is controlled by SLRXCKP bit and data latch edge -Rising edge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

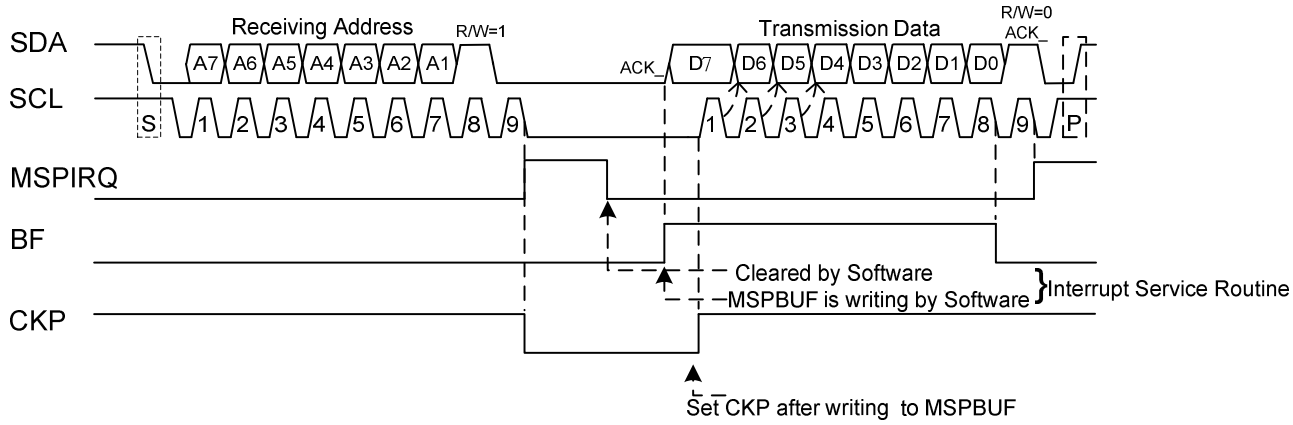


10.7.3 Slave Transmission

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.



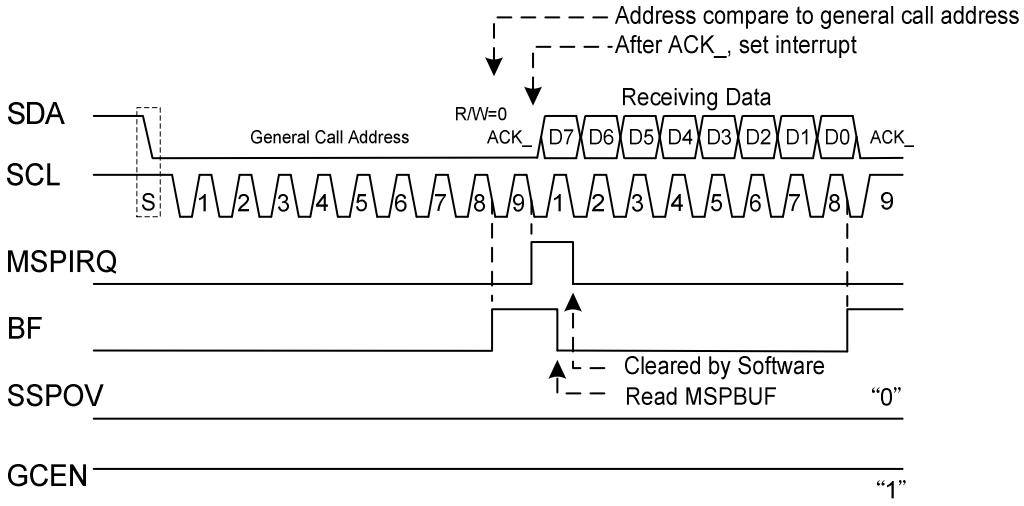
MSP Slave Transmission Timing Diagram

10.7.4 General Call Address

In MSP bus, the first 7-byte is the Slave address. Only the address match MSPADDR 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-bytes 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.

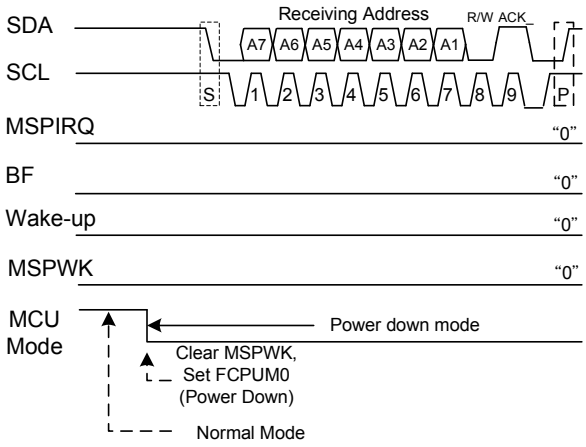
If the genera call address matches, the MSPSR data is transferred into MSPBUF, the BF flag bit is set, and in the falling edge of the ninth clock (ACK_) MSPIRQ flag set for interrupt request. In the interrupt service routine, reading MSPBUF can check if the address is the general call address or device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



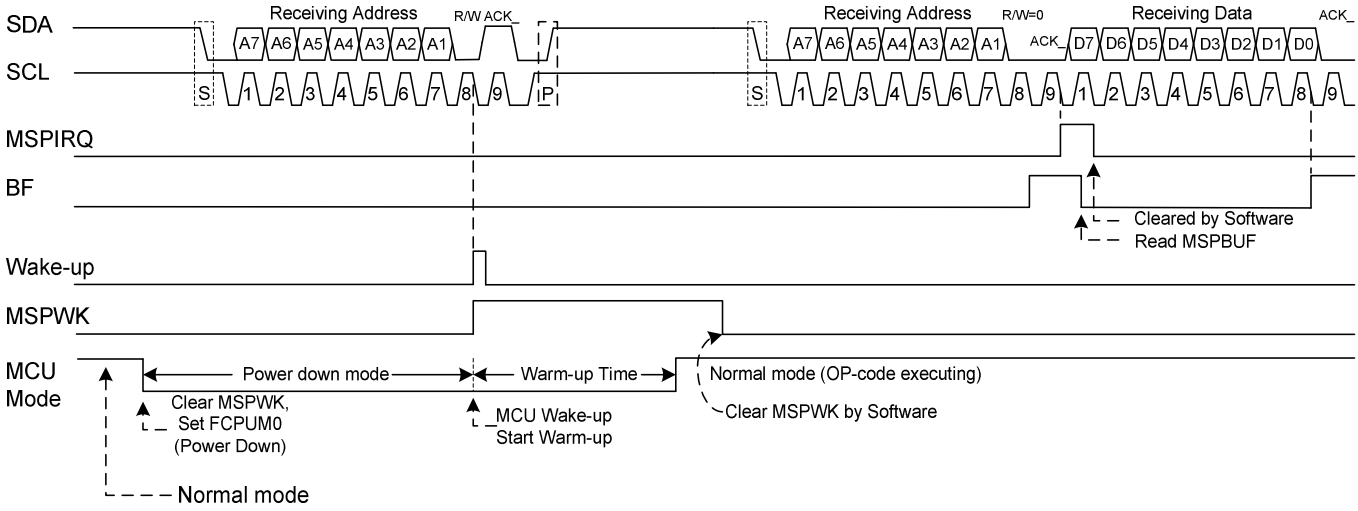
General Call Address Timing Diagram

10.7.5 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-byte 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, MSPWKset and start wake-up procedure but MSPIRQ will not set and MSPSR data will not load to MSPUBF. After MCU finish wake-up procedure, MSP will be in idle status and waiting master's START signal. Control register BF, MSPIRQ, MSPOV and MSPBUF will be the same status/data before power down.
If address not matches, a NOT acknowledge is still sent on the ninth clock of SCL, but MCU will be NOT wake-up and still keep in power down mode.



MSP Wake-up Timing Diagram: Address NOT Matched



MSP Wake-up Timing Diagram: Address Matched

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

10.8 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.

10.8.1 Mater Mode Support

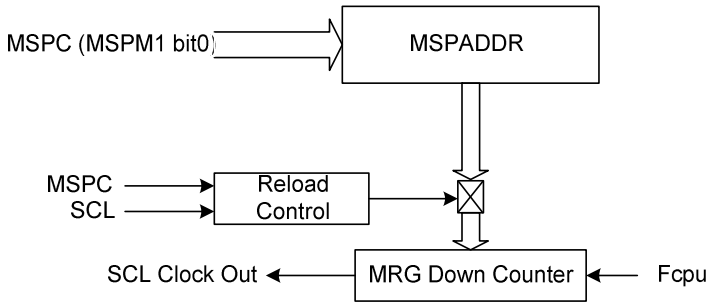
Master mode enable when MSPC and MSPENB bit set. Once the Master mode enabled, the user had following six options.

- 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.

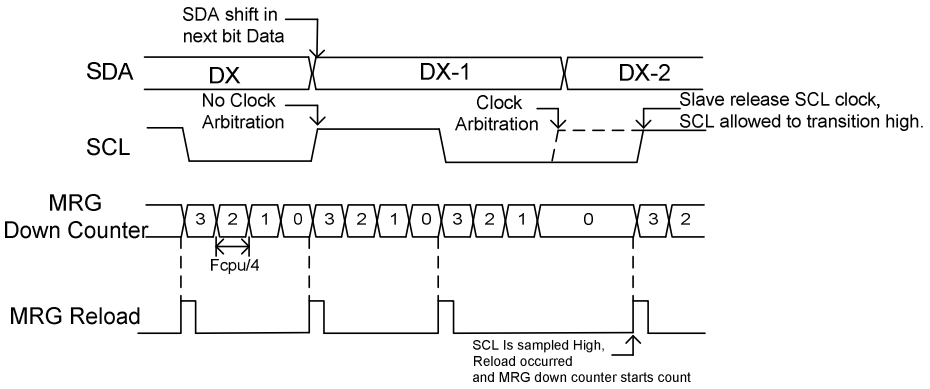
10.8.2 MSP Rate Generator

In MSP Mode, the MSP rate generator's reload value is located in the lower 7 bit of MSPADDR 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 MSPADDR 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/(MSPADDR)*2**
For example, if we want to set 400Khz in 4Mhz Fcpu, the MSPADDR have to set 0x05h.
MSPADDR=4Mhz/400Khz*2=5



MSP Rate Generator Block Diagram



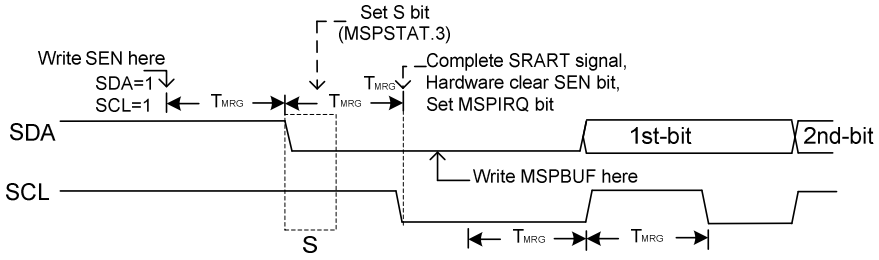
MRG Timing Diagram with and without Clock Arbitration (MSPADRR=0x03)

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

● WCOL Status Flag

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



START Condition Timing Diagram

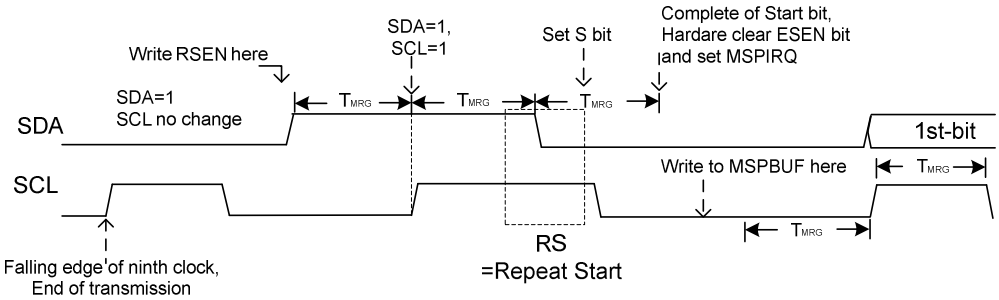
10.8.4 MSP Master mode Repeat START Condition

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

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

● WCOL Status Flag

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



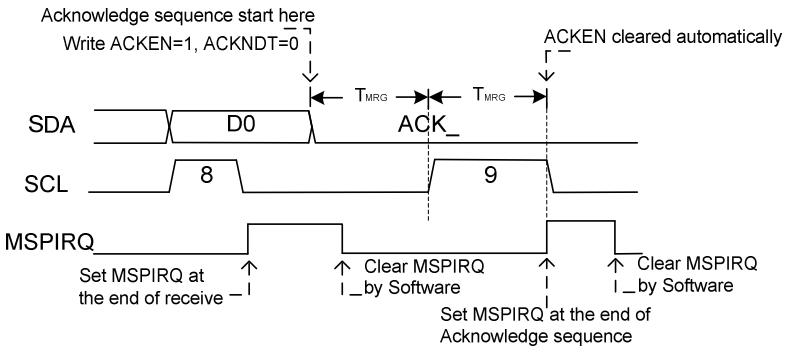
Repeat Start Condition Timing Diagram

10.8.5 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.

● WCOL Status Flag

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



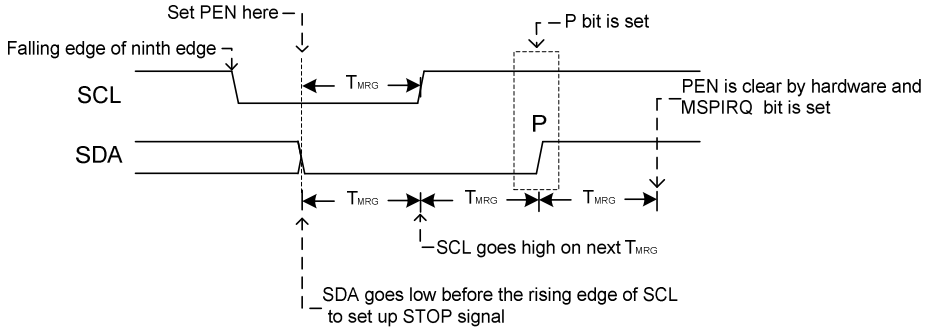
Acknowledge Sequence Timing Diagram

10.8.6 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.

● WCOL Status Flag

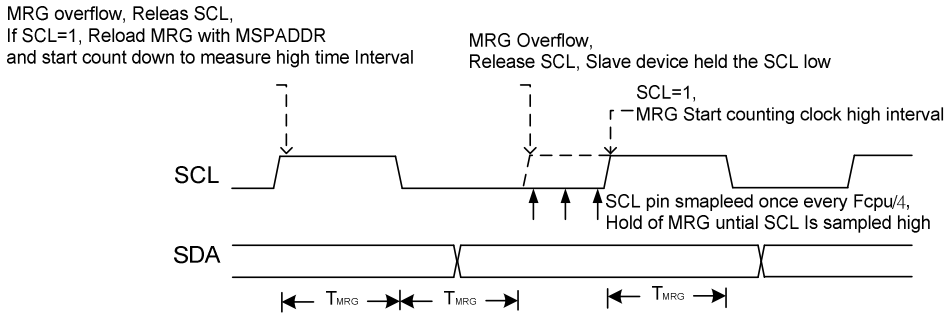
If user write to MSPBUF when a STOP condition is processing, then WCOL bit is set and the content of MSPBUF data is un-changed. (the writer doesn't occur)



STOP condition sequence Timing Diagram

10.8.7 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 MSPADDR[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

10.8.8 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.

● BF Status Flag

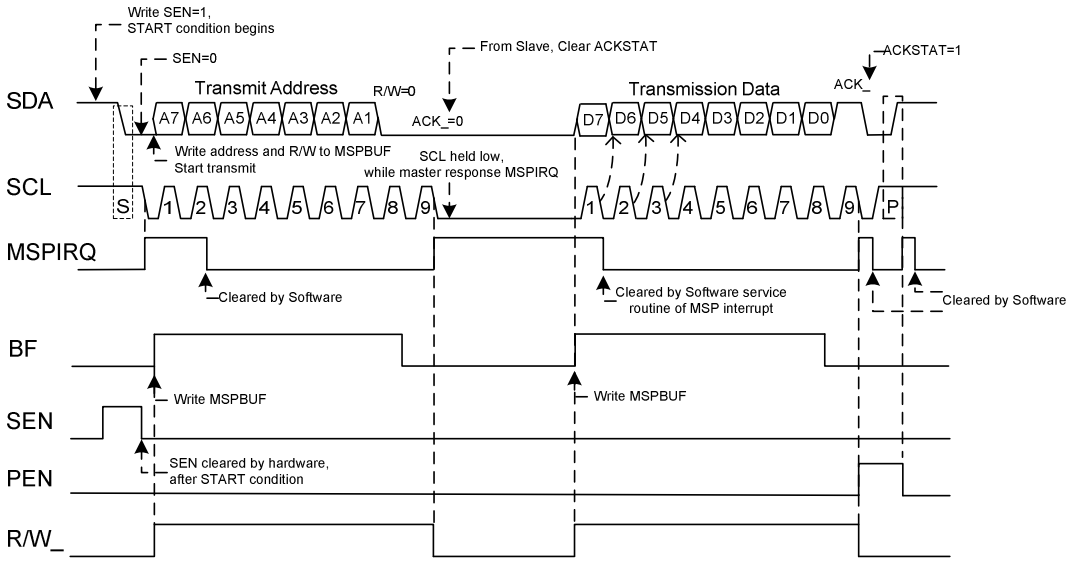
In transmission mode, the BF bit is set when user writes to MSPBUF and is cleared automatically when all 8 bit data are shift out.

● WCOL Flag

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

● ACKSTAT Status Flag

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



MSP Master Transmission Mode Timing Diagram

10.8.9 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.

● BF Status Flag

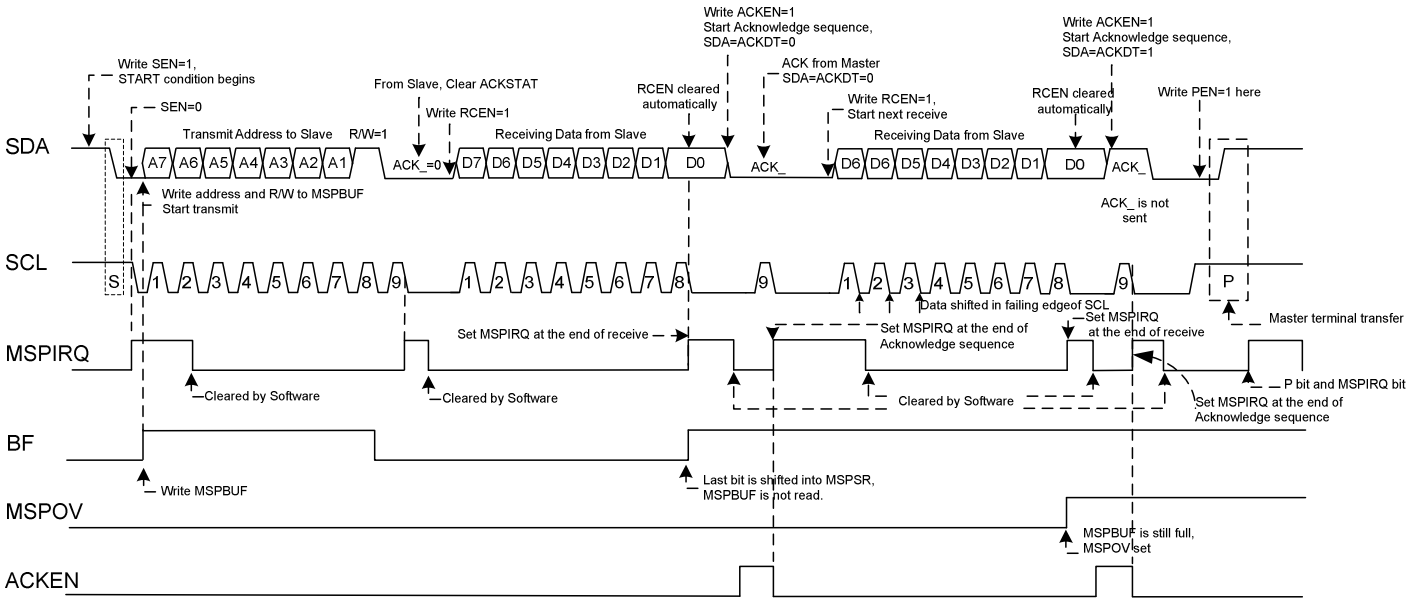
In Reception mode, the BF bit is set when an address or data byte is loaded into MSPBUF from MSPSR. It is cleared automatically when MSPBUF is read.

● MSPOV Flag

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

● WCOL Flag

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

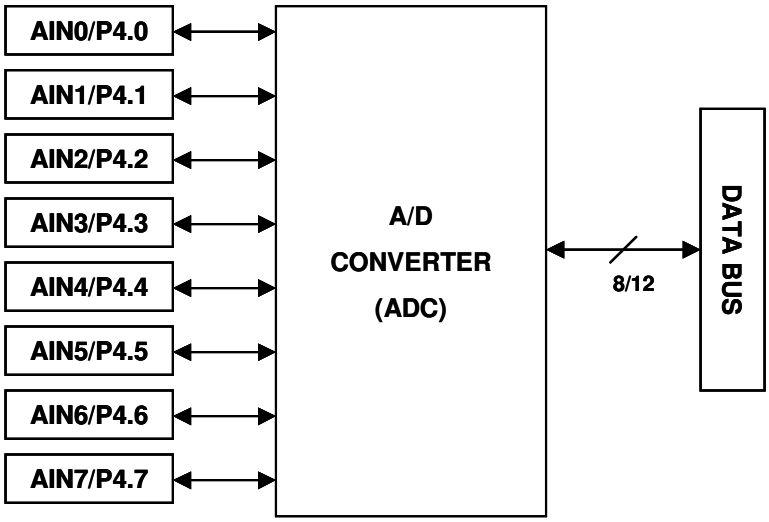


MSP Master Receiving Mode Timing Diagram

11 8 CHANNEL ANALOG TO DIGITAL CONVERTER

11.1 OVERVIEW

This analog to digital converter has 8-input sources with up to 4096-step resolution to transfer analog signal into 12-bits digital data. The sequence of ADC operation is to select input source (AIN0 ~ AIN7) at first, then set GCHS and ADS bit to "1" to start conversion. When the conversion is complete, the ADC circuit will set EOC bit to "1" and final value output in ADB register. This ADC circuit can select between 8-bit and 12-bit resolution operation by programming ADLEN bit in ADR register.



- ***Note: For 8-bit resolution the conversion time is 12 steps.***
  - ***For 12-bit resolution the conversion time is 16 steps***

- ***Note: The analog input level must be between the AVREFH and AVREFL.***
- ***Note: The AVREFL connects to VSS internally in SN8P2754, SN8P2755 and SN8P2756.***
- ***Note: The AVREFH level must be between the AVDD and AVREFL + 2.0V.***
- ***Note: The AVREFL level must be between the VSS and AVREFH - 2.0V.***
- ***Note: ADC programming notice:***
  1. ***Set ADC input pin I/O direction as input mode***
  2. ***Disable pull-up resistor of ADC input pin***
  3. ***Disable ADC before enter power down (sleep) mode to save power consumption.***
  4. ***Set related bit of P4CON register to avoid extra power consumption in power down mode.***
  5. ***Delay 100uS after enable ADC (set ADENB = "1") to wait ADC circuit ready for conversion.***
  6. ***Disable ADC (set ADENB = "0") before enter sleep mode to save power consumption.***

11.2 ADM REGISTER

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0
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 **ADENB:** ADC control bit.
0 = Disable.
1 = Enable.

Bit 6 **ADS:** ADC start bit.
0 = Stop.
1 = Starting.

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

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

Bit[2:0] **CHS[2:0]:** ADC input channels select bit.
000 = AIN0, 001 = AIN1, 010 = AIN2, 011 = AIN3
100 = AIN4, 101 = AIN5, 110 = AIN6, 111 = AIN7

✸ ***Note: If ADENB = 1, users should set P4.n/AINn as input mode without pull-up. System doesn't set automatically. If P4CON.n is set, the P4.n/AINn's digital I/O function including pull-up is isolated.***

11.3 ADR REGISTERS

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	ADCKS2	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
After reset	0	0	0	0	-	-	-	-

Bit 7,6,4 **ADCKS [2:0]:** ADC's clock source select bit.

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

Bit 5 **ADLEN:** ADC's resolution select bits.
0 = 8-bit
1 = 12-bit.

Bit [3:0] **ADB [3:0]:** ADC data buffer.
ADB11~ADB4 bits for 8-bit ADC
ADB11~ADB0 bits for 12-bit ADC

✷ ***Note: ADC buffer ADR [3:0] initial value after reset is unknown.***

11.4 ADB REGISTERS

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
Read/Write	R	R	R	R	R	R	R	R
After reset	-	-	-	-	-	-	-	-

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

ADB is ADC data buffer to store AD converter result. The ADB is only 8-bit register including bit 4~bit11 ADC data. To combine ADB register and the low-nibble of ADR will get full 12-bit ADC data buffer. The ADC buffer is a read-only register. In 8-bit ADC mode, the ADC data is stored in ADB register. In 12-bit ADC mode, the ADC data is stored in ADB and ADR registers.

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

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

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

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

✷ ***Note: ADC buffer ADB initial value after reset is unknown.***

11.5 P4CON REGISTERS

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

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
Read/Write	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[4:0] **P4CON[7:0]:** P4.n configuration control bits.
0 = P4.n can be an analog input (ADC input) or digital I/O pins.
1 = P4.n is pure analog input, can't be a digital I/O pin.

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

11.6 ADC CONVERTING TIME

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

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

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

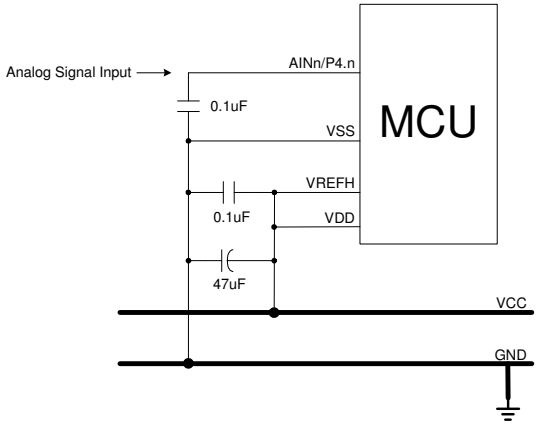
ADLEN	ADCKS2	ADCKS1	ADCKS0	ADC Clock	ADC conversion time
0 (8-bit)	0	0	0	Fcpu/16	1/(4MHz/16/4)*12 = 192 us
	0	0	1	Fcpu/8	1/(4MHz/8/4)*12 = 96 us
	0	1	0	Fcpu	1/(4MHz/4)*12 = 12 us
	0	1	1	Fcpu/2	1/(4MHz/2/4)*12 = 24 us
0 (8-bit)	1	0	0	Fcpu/64	1/(4MHz/64/4)*12 = 768 us
	1	0	1	Fcpu/32	1/(4MHz/32/4)*12 = 384 us
	1	1	0	Fcpu/4	1/(4MHz/4/4)*12 = 48 us
	1	1	1	Reserved	Reserved
1 (12-bit)	0	0	0	Fcpu/16	1/(4MHz/16/4)*16 = 256 us
	0	0	1	Fcpu/8	1/(4MHz/8/4)*16 = 128 us
	0	1	0	Fcpu	1/(4MHz/4)*16 = 16 us
	0	1	1	Fcpu/2	1/(4MHz/2/4)*16 = 32 us
1 (12-bit)	1	0	0	Fcpu/64	1/(4MHz/64/4)*16 = 1024 us
	1	0	1	Fcpu/32	1/(4MHz/32/4)*16 = 512 us
	1	1	0	Fcpu/4	1/(4MHz/4/4)*16 = 64 us
	1	1	1	Reserved	Reserved

11.7 ADC ROUTINE EXAMPLE

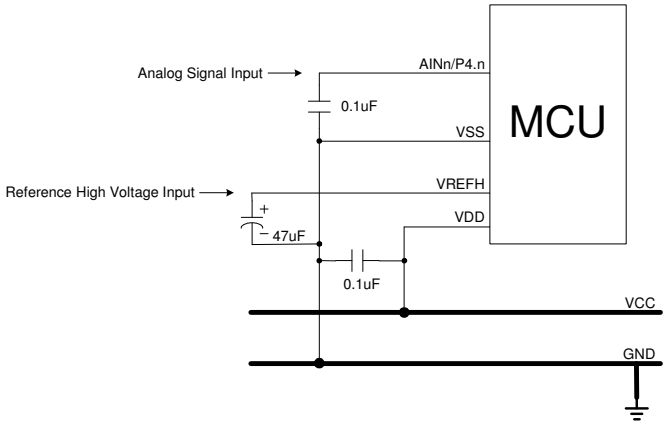
➢ **Example : Configure AIN0 as 12-bit ADC input and start ADC conversion then enter power down mode.**

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

11.8 ADC CIRCUIT



ADC reference high voltage is from VDD pin. The VERFH should be from MCU's VDD pin. Don't connect from main power.



ADC reference high voltage is from external voltage. The capacitor (47uF) between VREFH and VSS is necessary to stable VERFH voltage.

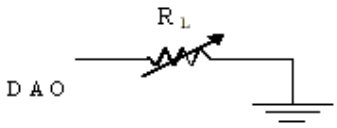
12 DIGITAL TO ANALOG CONVERTER

12.1 OVERVIEW

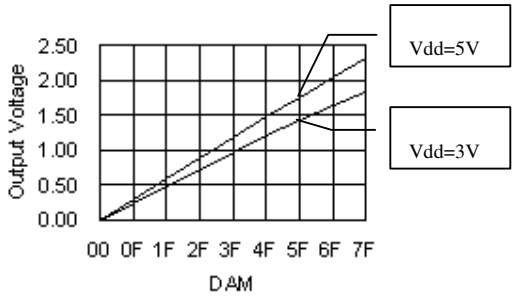
The D/A converter uses 7-bit structure to synthesize 128 steps' analog signal with current source output. After DAENB bit is set to "1", DAC circuit will turn to be enabled and the DAM register, from bit0 to bit6, will send digital signal to ladder resistors in order to generate analog signal on DAO pin.

The DA converter Block Diagram

In order to get a proper linear output, a Loading Resistor R_L is usually added between DAO and Ground. The example shows the result of Vdd = 5V, R_L =150ohm and Vdd = 3V, R_L =150ohm.



DAO Circuit with R_L



DAC Output Voltage in Vdd=5V and 3V

Note:

- ***1:The D/A converter is not designed for a precise DC voltage output and is suitable for a simple audio application e.g. Tone or Melody generation.***
- ***2:For best linearity performance, the max. Loading Resistance R_L is 150 ohm @5V, 100 ohm @3V***

12.2

12.3 DAM REGISTER

DAM initial value = 0000 0000

0B0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAM	DAENB	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0
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 **DAENB:** Digital to Analog converter control bit.
0 = disable
1 = enable

Bit [6:0] **DAB [6:0]:** Digital input data.

12.4 D/A CONVERTER OPERATION

When the DAENB = 0, the DAO pin is output floating status. After setting DAENB to "1", the DAO output value is controlled by DAB bits.

➢ **Example: Output 1/2 VDD from DAO pin.**

```
MOV     A, #00111111B
B0MOV   DAM, A          ; Set DAB to a half of the full scale.

B0BSET  FDAENB          ; Enable D/A function.
```

The DAB's data v.s. DAO's output voltage as following:

DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0	DAO
0	0	0	0	0	0	0	VSS
0	0	0	0	0	0	1	Idac
0	0	0	0	0	1	0	2 * Idac
0	0	0	0	0	1	1	3 * Idac
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	1	1	1	1	1	0	126 * Idac
1	1	1	1	1	1	1	127 * Idac

✸ ***Note: Idac = I_{FSO} / (2^7-1) (I_{FSO}: Full-scale Output Current).***

13 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
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1+N
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1+N
	CLR	M	M ← 0	-	-	-	1
	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
BRANCH	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1+ S
	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1+ S
	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1+N+S
	BTS0	M.b	If M.b = 0, then skip next instruction	-	-	-	1 + S
	BTS1	M.b	If M.b = 1, then skip next instruction	-	-	-	1 + S
	B0BTS0	M.b	If M(bank 0).b = 0, then skip next instruction	-	-	-	1 + S
	B0BTS1	M.b	If M(bank 0).b = 1, then skip next instruction	-	-	-	1 + S
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
MISC	RET		PC ← Stack	-	-	-	2
	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
	PUSH		To push ACC and PFLAG (except NT0, NPD bit) into buffers.	-	-	-	1
	POP		To pop ACC and PFLAG (except NT0, NPD bit) from buffers.	√	√	√	1
	NOP		No operation	-	-	-	1

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

14 ELECTRICAL CHARACTERISTIC

14.1 ABSOLUTE MAXIMUM RATING

(All of the voltages referenced to Vss)

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin).....Vss - 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2754K, SN8P2754S, SN8P2755P, SN8P2755S, SN8P2756P, SN8P2757Q
SN8P2758P, SN8P2758X..... 0°C ~ + 70°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

14.2 STANDARD ELECTRICAL CHARACTERISTIC

(All of voltages referenced to Vss, Vdd = 5.0V, fosc = 4 MHz,fcpu=1MHZ, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd		2.4	5.0	5.5	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	150	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O output source current	IoH	Vop = Vdd - 0.5V		8	12	-	mA
sink current	IoL	Vop = Vss + 0.5V		8	15	-	mA
INTn trigger pulse width	Tint0	INT0 ~ INT2 interrupt request pulse width		2/fcpu	-	-	Cycle
AVREFH input voltage	Varfh	Vdd = 5.0V		Varfl+2V	-	Vdd	V
AVREFL input voltage	Varfl	Vdd = 5.0V		Vss	-	Varfh–2V	V
AIN0 ~ AIN7 input voltage	Vani	Vdd = 5.0V		Varfl	-	Varfh	V
Supply Current (ADC Disable)	Idd1	normal Mode (No loading, Fcpu = Fosc/4)	Vdd= 5V 4MHz	-	3	6	mA
			Vdd= 3V 4MHz	-	1.5	3	mA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd= 5V ILRC 32KHz	-	80	160	uA
			Vdd= 3V ILRC 16KHz	-	15	30	uA
	Idd3	Sleep Mode	Vdd= 5V	-	1	2	uA
			Vdd= 3V	-	0.5	2	uA
	Idd4	Green Mode (No loading, Fcpu = Fosc/4, Watchdog Disable)	Vdd= 5V 4MHz	-	0.6	1.2	mA
			Vdd= 3V 4MHz	-	0.2	0.4	mA
			Vdd= 5V ILRC 32Khz	-	20	40	uA
			Vdd= 3V ILRC 16Khz	-	5	10	uA
LVD Detect Voltage	Vdet	Low voltage detect level		1.5	1.8	2.2	V
DAC Full-scale Output Current	I_{FSO}	Vdd=5.0V		8	14	21	mA
		Vdd=3.0V		5	11	18	mA
DAC Loading Resistance	R_L	Vdd=5.0V		-	-	150	Ω
		Vdd=3.0V		-	-	100	Ω
DAC DNL	DAC_{DNL}	DAC Differential NonLinearity		-	±1*	-	LSB
DAC INL	DAC_{INL}	DAC Integral NonLinearity		-	±3*	-	LSB
ADC current consumption	I_{ADC}	Vdd=5.0V		-	0.6*	-	mA
		Vdd=3.0V		-	0.4*	-	mA
ADC enable time	Tast	Ready to start convert after set ADENB = “1”		100	-	-	uS
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	±2	±16	LSB
Integral Nonlinearity	INL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	±2	±4	±16	LSB
No Missing Code	NMC	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	8	10	12	Bits

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

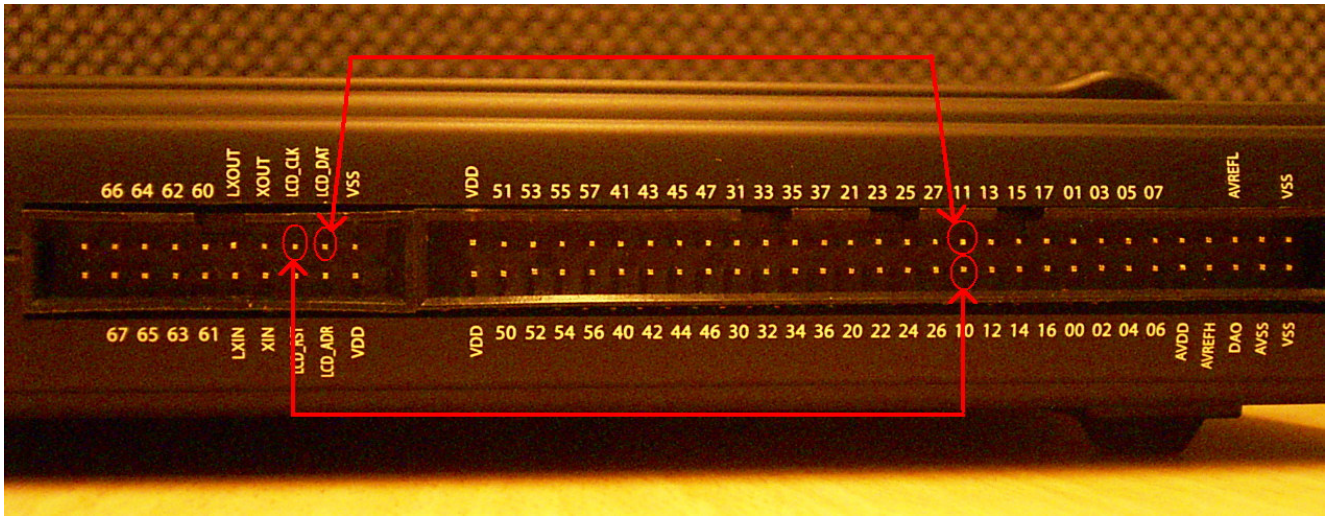
15 APPLICATION NOTICE

15.1 Development Tool Version

15.1.1 ICE (In circuit emulation)

- **SN8ICE 2K Plus:** Full function emulates SN8P275X series.
- **Use ICE emulation MSP function:**

  1. P10 must connect to LCD_CLK,P11 must connect to LCD_SDA.



2. P10/P11 is built-in open-drain function. When enable open-drain function, P10/P11 must connect pull-up resistor.

15.1.2 OTP Writer

- **MPIII Writer:** It's convenient to connect Full Speed USB 1.1 port with PC and then update the writer, connect programming chip or download programming code.

15.1.3 IDE (Integrated Development Environment)

SONiX 8-bit MCU integrated development environment include Assembler, ICE debugger and OTP writer software.

- **For SN8ICE 2K Plus:** M2IDE V1.19 or later

15.1.4

15.2 OTP Programming Pin

15.2.1 The pin assignment of Easy Writer transition board socket:

Easy Writer JP1/JP2

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

JP1 for MP transition board
JP2 for Writer V3.0 transition board

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

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

JP3 for MP transition board

15.2.2 The pin assignment of Writer V3.0 and V2.5 transition board socket:

GND	**1**	**2**	**VDD**
CE	**3**	**4**	**CLK**
OE	**5**	**6**	**PGM**
D0	**7**	**8**	**D1**
D2	**9**	**10**	**D3**
D4	**11**	**12**	**D5**
D6	**13**	**14**	**D7**
VPP	**15**	**16**	**VDD**
RST	**17**	**18**	**HLS**

Writer V2.5 JP1 Pin Assignment

GND	**2**	**1**	**VDD**
CE	**4**	**3**	**CLK**
OE	**6**	**5**	**PGM**
D0	**8**	**7**	**D1**
D2	**10**	**9**	**D3**
D4	**12**	**11**	**D5**
D6	**14**	**13**	**D7**
VPP	**16**	**15**	**VDD**
RST	**18**	**17**	**HLS**
	20	**19**	

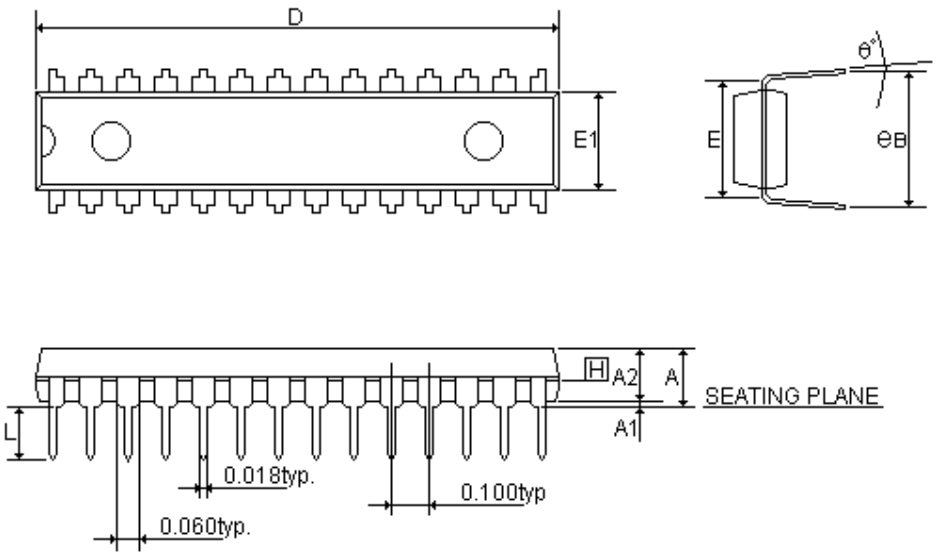
Writer V3.0 JP1 Pin Assignment

15.2.3 SN8P275X Series Programming Pin Mapping:

OTP Programming Pin of SN8P275X Series											
Chip Name		SN8P2754		SN8P2755		SN8P2758					
EZ Writer / Writer V3.0		OTP IC / JP3 Pin Assignment									
Number	Pin	Number	Pin	Number	Pin	Number	Pin				
1	VDD	3,14,24	VDD	4,26	VDD	8,16,36,37	VDD				
2	GND	7,21	VSS	1,16	VSS	5,25	VSS				
3	CLK	20	P5.0	32	P5.0	47	P5.0				
4	CE	-	-	-	-	-	-				
5	PGM	6	P1.0	14	P1.0	20	P1.0				
6	OE	19	P5.1	31	P5.1	46	P5.1				
7	D1	-	-	-	-	-	-				
8	D0	-	-	-	-	-	-				
9	D3	-	-	-	-	-	-				
10	D2	-	-	-	-	-	-				
11	D5	-	-	-	-	-	-				
12	D4	-	-	-	-	-	-				
13	D7	-	-	-	-	-	-				
14	D6	-	-	-	-	-	-				
15	VDD	3,14,24	VDD	4,26	VDD	8,16,36,37	VDD				
16	VPP	28	RST	8	RST	12	RST				
17	HLS	-	-	-	-	-	-				
18	RST	-	-	-	-	-	-				
19	-	-	-	-	-	-	-				
20	ALSB/PDB	5,22	P1.1/P3.1	13,2	P1.1/P3.1	19,6	P1.1/P3.1				

16 PACKAGE INFORMATION

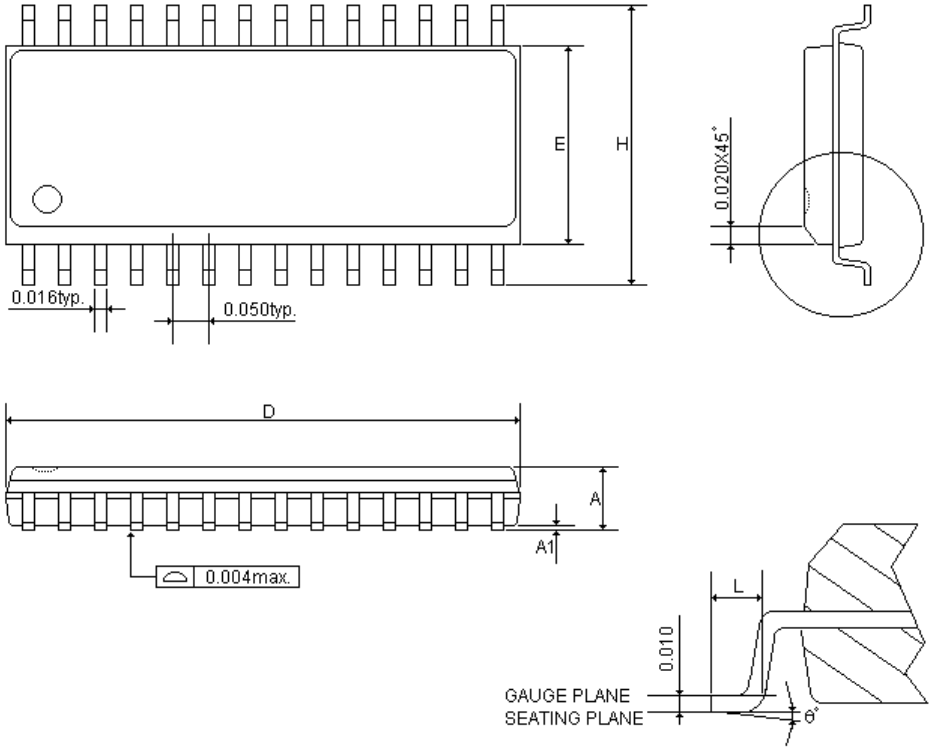
16.1 SK-DIP28 PIN



Symbols	MIN.	NOR.	MAX.
A	-	-	0.210
A1	0.015	-	-
A2	0.114	0.130	0.135
D	1.390	1.390	1.400
E		0.310BSC.	
E1	0.283	0.288	0.293
L	0.115	0.130	0.150
e_B	0.330	0.350	0.370
θ °	0	7	15

UNIT : INCH

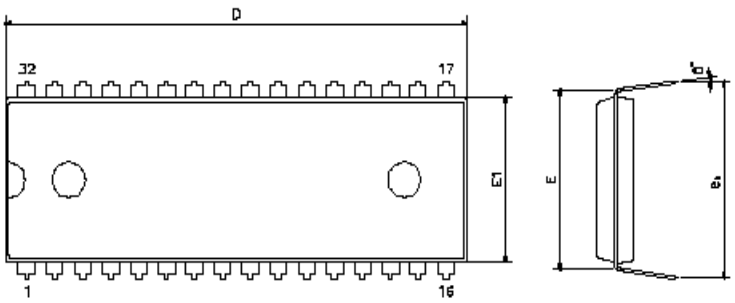
16.2 SOP28 PIN



Symbols	MIN.	MAX.
A	0.093	0.104
A1	0.004	0.012
D	0.697	0.713
E	0.291	0.299
H	0.394	0.419
L	0.016	0.050
θ °	0	8

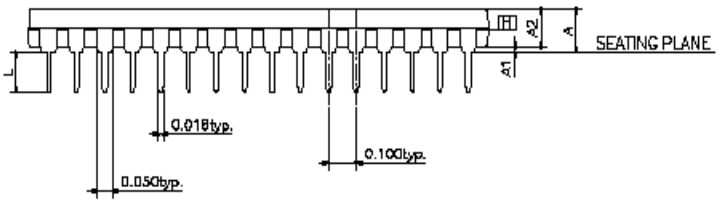
UNIT : INCH

16.3 P-DIP 32 PIN



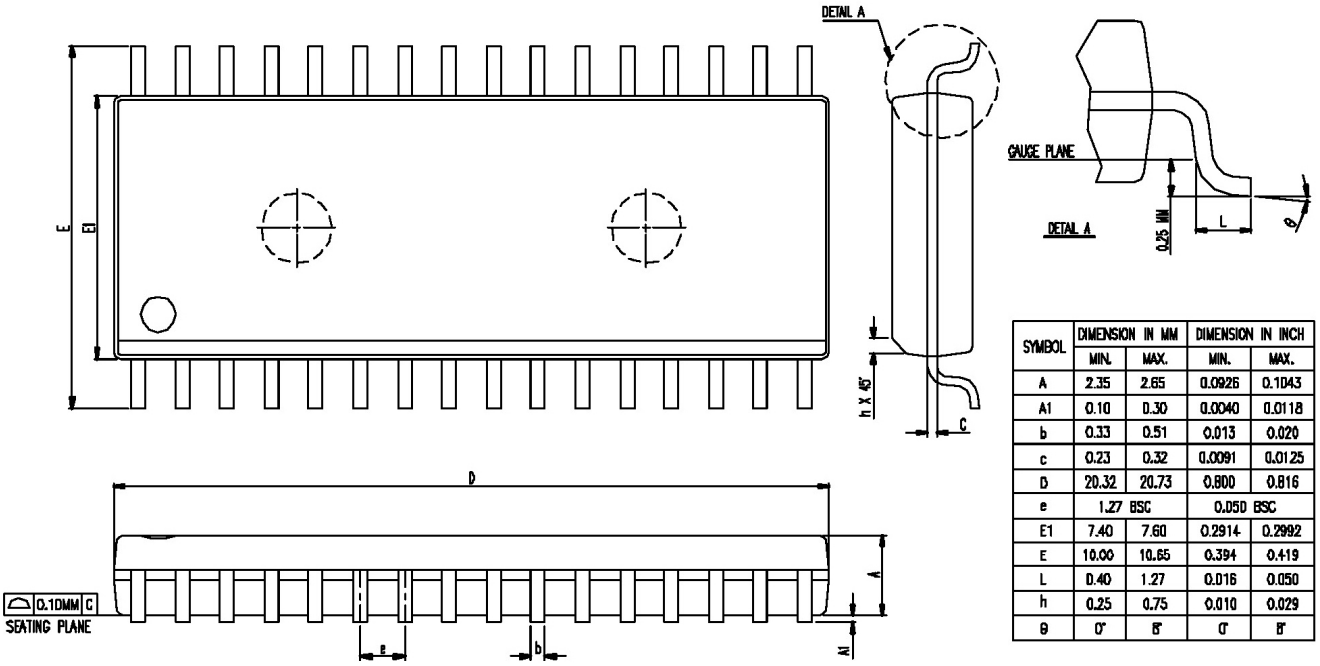
SYMBOLS	MIN.	NOR.	MAX.
A	–	–	0.220
A1	0.015	–	–
A2	0.150	0.155	0.160
D	1.645	1.650	1.660
E		0.600 BSC	
E1	0.540	0.545	0.550
L	0.115	0.130	0.150
e_B	0.630	0.650	0.670
θ°	0	7	15

UNIT : INCH



NOTE:
1.JEDEC OUTLINE : MS-011 AC

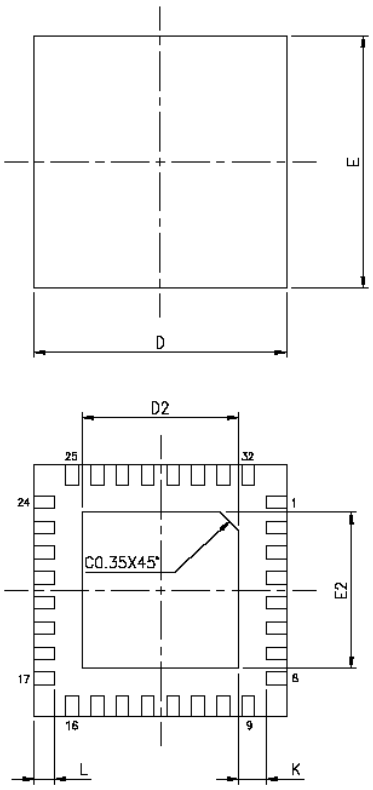
16.4 SOP 32 PIN

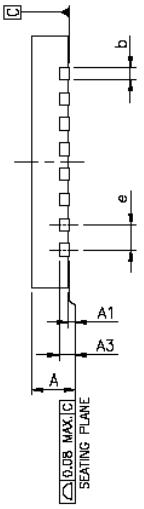


SYMBOL	DIMENSION IN MM		DIMENSION IN INCH	
	MIN.	MAX.	MIN.	MAX.
A	2.35	2.65	0.0926	0.1043
A1	0.10	0.30	0.0040	0.0118
b	0.33	0.51	0.013	0.020
c	0.23	0.32	0.0091	0.0125
D	20.32	20.73	0.800	0.816
e	1.27 BSC		0.050 BSC	
E1	7.40	7.60	0.2914	0.2992
E	10.00	10.65	0.394	0.419
L	0.40	1.27	0.016	0.050
h	0.25	0.75	0.010	0.029
θ	0°	8°	0°	8°

*NOTES : DIMENSION " D " DOES NOT INCLUDE MOLD FLASH , PROTRUSIONS OR GATE BURRS.
MOLD FLASH , PROTRUSIONS AND GATE BURRS SHALL NOT EXCEED 0.15 MM (0.006 INCH) PER SIDE.

16.5 QFN 32 PIN



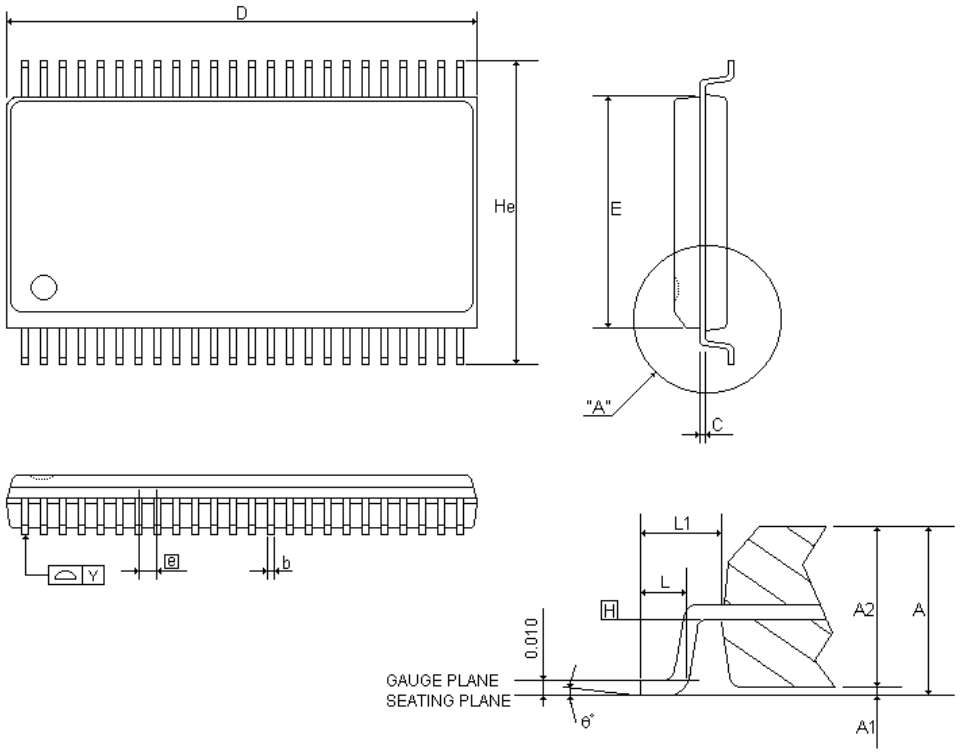


NOTES :
1. JEDEC OUTLINE : N/A.
2. DIMENSION b APPLIES TO METALLIZED TERMINAL AND IS MEASURED BETWEEN 0.15mm AND 0.30mm FROM THE TERMINAL TIP. IF THE TERMINAL HAS THE OPTIONAL RADIUS ON THE OTHER END OF THE TERMINAL, THE DIMENSION b SHOULD NOT BE MEASURED IN THAT RADIUS AREA.
3. THE MINIMUM "K" VALUE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

SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.003	0.030	0.031	0.070	0.750	0.800
A1	0.000	0.001	0.002	0.000	0.020	0.050
A3		0.008 REF.			0.203 REF.	
b	0.007	0.010	0.012	0.180	0.250	0.300
D		0.20 BSC			5.00 BSC	
E		0.20 BSC			5.00 BSC	
e		0.02 BSC			0.50 BSC	
L	0.014	0.016	0.018	0.350	0.400	0.450
K	0.008	-	-	0.20	-	-

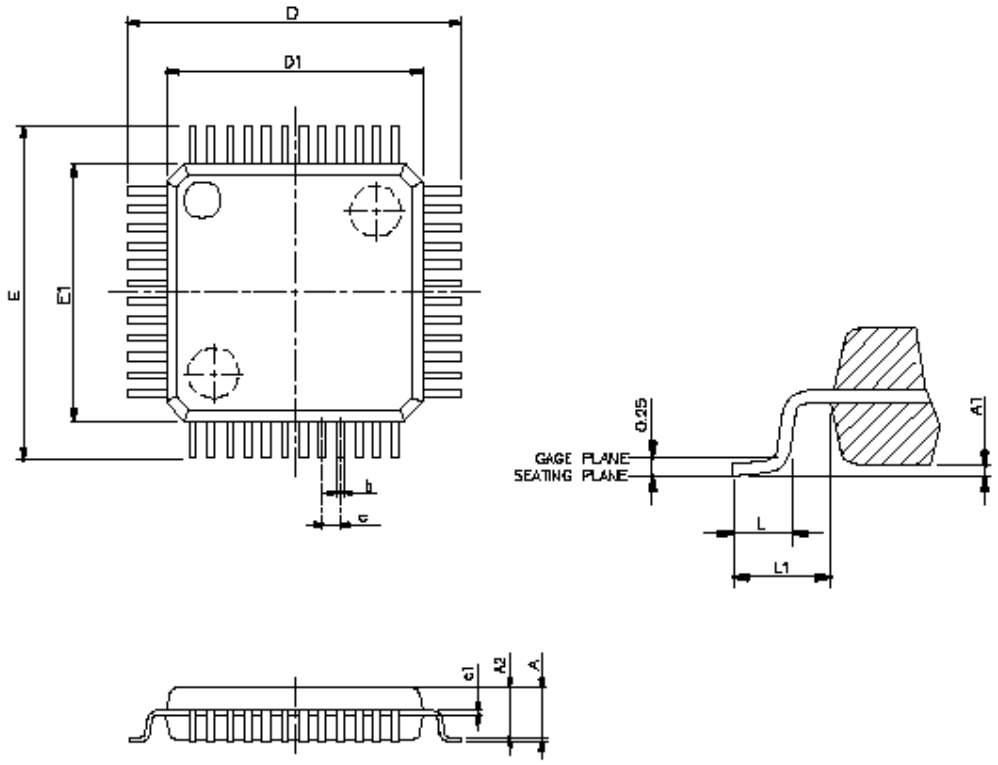
PAD SIZE	D2 (mm)			E2 (mm)		
	MIN	NOR	MAX	MIN	NOR	MAX
114x114 MIL	2.60	2.70	2.75	2.60	2.70	2.75
134x134 MIL	3.10	3.20	3.25	3.10	3.20	3.25

16.6 SSOP 48 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.095	0.102	0.110	2.413	2.591	2.794
A1	0.008	0.012	0.016	0.203	0.305	0.406
A2	0.089	0.094	0.099	2.261	2.388	2.515
b	0.008	0.010	0.030	0.203	0.254	0.762
C	-	0.008	-	-	0.203	-
D	0.620	0.625	0.630	15.748	15.875	16.002
E	0.291	0.295	0.299	7.391	7.493	7.595
[e]	-	0.025	-	-	0.635	-
He	0.396	0.406	0.416	10.058	10.312	10.566
L	0.020	0.030	0.040	0.508	0.762	1.016
L1	-	0.056	-	-	1.422	-
Y	-	-	0.003	-	-	0.076
θ°	0°	-	8°	0°	-	8°

16.7 LQFP 48 PIN



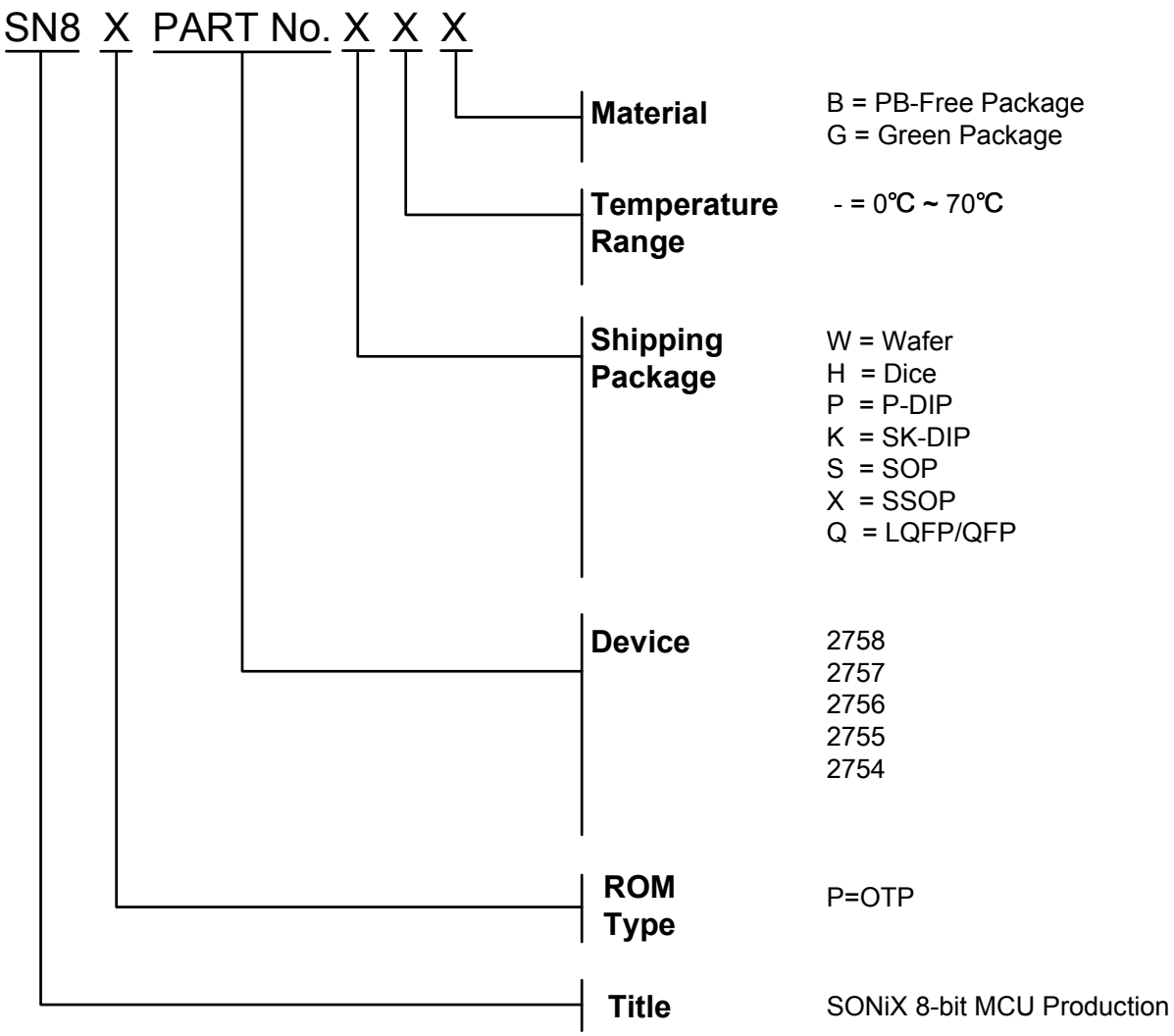
SYMBOLS	MIN	NOR	MAX
	(mm)		
A	-	-	1.6
A1	0.05	-	0.15
A2	1.35	-	1.45
c1	0.09	-	0.16
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
e	0.5 BSC		
B	0.17	-	0.27
L	0.45	-	0.75
L1	1 REF		

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 OTP MCU.

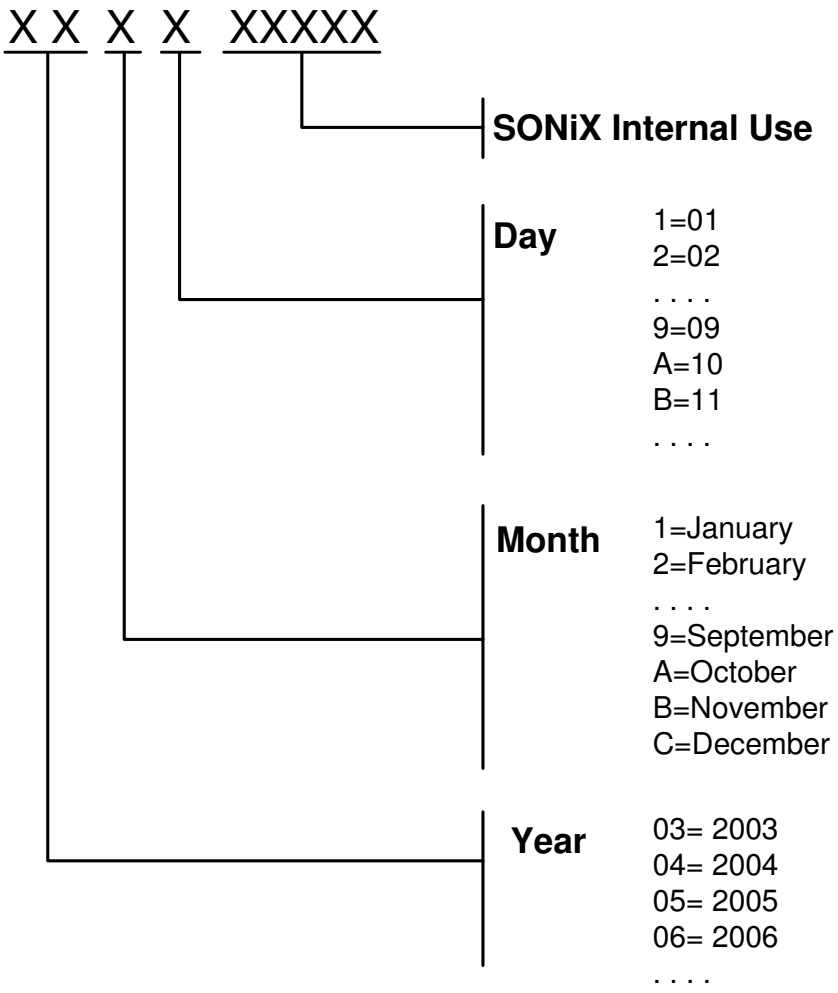
17.2 MARKING INDETIFICATION SYSTEM



17.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P2758XB	OTP	2758	SSOP	0℃~70℃	PB-Free Package

17.4 DATECODE SYSTEM



Main Office:
Address: 10F-1, NO. 36, Taiyuan Stree., Chupei City, Hsinchu, Taiwan R.O.C.
Tel: 886-3-5600 888
Fax: 886-3-5600 889

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

Hong Kong Office:
Address: Flat 3 9/F Energy Plaza 92 Granville Road, Tsimshatsui East Kowloon.
Tel: 852-2723 8086
Fax: 852-2723 9179

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