SN8P2808

USER'S MANUAL

Preliminary Specification Version 0.7

SONiX 8-Bit Micro-Controller

AMENDMENT HISTORY

Version	Date	Description
VER 0.1	Sep. 2006	First issue.
VER 0.2	Dec. 2006	1. Modify “the basic timer table interval time of T0” for Low speed mode. 2. Modify “the basic timer table interval time of TC0” for Low speed mode. 3. Modify “the basic timer table interval time of TC1” for Low speed mode. 4. Add “Bit5 ADLEN” to ADR REGISTERS.
VER 0.3	Jan. 2007	1. Modify “PIN ASSIGNMENT” for LQFP48, DIP48(SSOP48). 2. Modify “LCD REGISTER” for LCD bias control bit description 3. Delete “ADC CONVERTING TIME” about Fosc/16 sheet. ***4.*** Modify “***AVREFH pin will out VHS[1:0] setting voltage eg, VDD, 4V, 3V, 2V when REFS=1” into “AVREFH pin will out VHS[1:0] setting voltage eg, VDD, 4V, 3V, 2V when REFS=GCHS=1”*** 5. Modify “PROGRMMING PIN MAPPING” for P4.7 and P4.6 are CLK, OE.
VER 0.4	Feb. 2007	1. *Re-modify “**AVREFH pin will out VHS[1:0] setting voltage eg, VDD, 4V, 3V, 2V when REFS=1” into “AVREFH pin will out VHS[1:0] setting voltage eg, VDD, 4V, 3V, 2V when REFS=ADENB=1**”* 2. *Modify "ADC 8+1 Channel” into 8 channel.* 3. *Add “EV-KIT User manual” section*
VER 0.5	Nov. 2007	*1. To modify SN8P2808 Programming Pin Mapping table*
VER 0.6	Mar. 2009	*1. Modify TC0 Green mode wake up function.*
VER 0.7	Jun. 2009	*1 Add SN8P2807 Programming Pin table.*

Table of Content

1 PRODUCT OVERVIEW

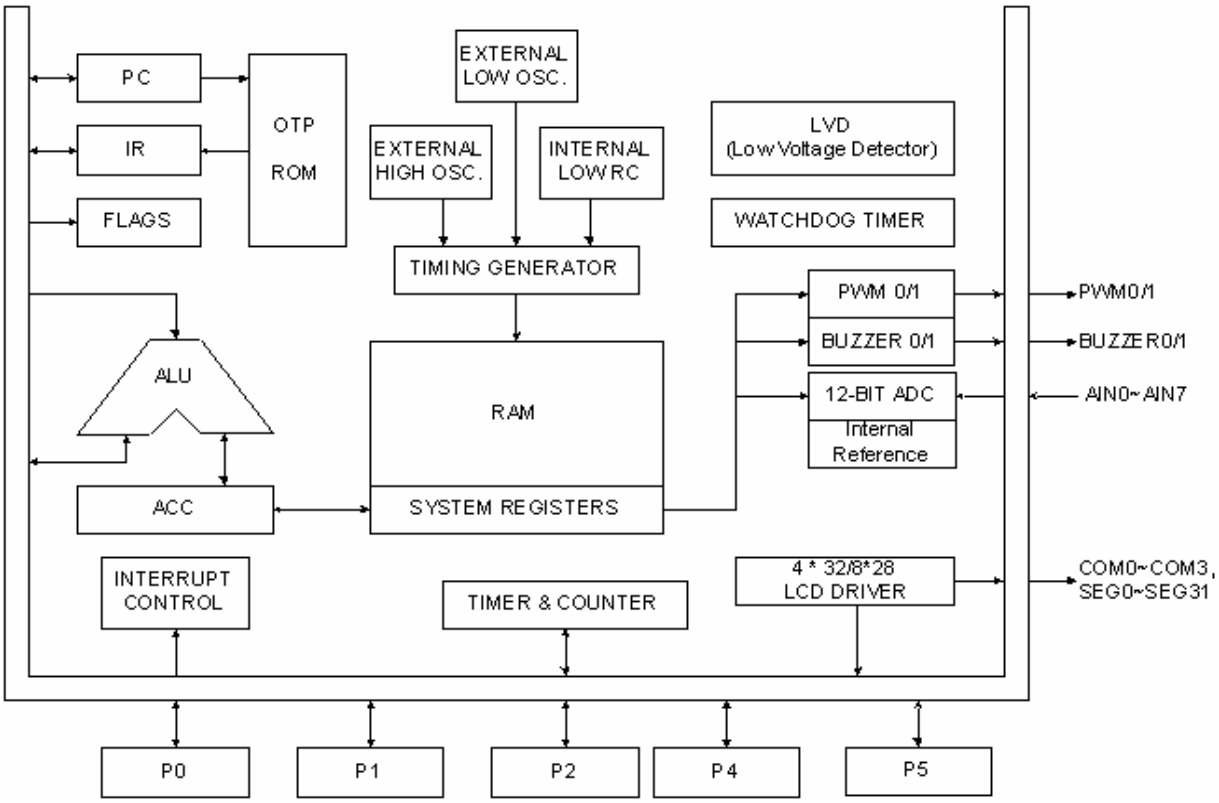
1.1 FEATURES

- **Memory configuration**
  OTP ROM size: 6K * 16 bits.
  RAM size: 256 * 8 bits.
- **4*32 / 8*28 LCD Driver**
  R type 1/3bias or 1/4 bias.
- **8 levels stack buffer.**
- **I/O pin configuration**
  Bi-directional: P0, P1, P2, P4, P5.
  Bi-directional and shared with SEG pins: P2.
  Input only: P0.3 shared with RST/VPP pin.
  Wakeup: P0, P1 level change.
  Pull-up resisters: P0, P1, P2, P4, P5
  External interrupt trigger edge:
  P0.0 controlled by PEDGE.
  Event counter input:
  P0.0 is TC0 event counter input.
  P0.1 is TC1 event counter input.
- **8-ch 12-bit ADC with internal reference voltage 2v/3v/4v/VDD.**
- **3-Level LVD**
  For system reset or brief VDD indicator.
- **Powerful instructions**
  Instruction cycle controlled by code option.
  Instruction's length is one word.
  Most of instructions are one cycle only.
  Maximum instruction cycle is two.
  All ROM area JMP and CALL instruction.
  All ROM area CALL address instruction.
  All ROM area lookup table function (MOVC)
- **Six interrupt sources**
  Four internal interrupts: T0, TC0, TC1, ADC
  Two external interrupts: INT0., INT1
- **Three 8-bit Timer/Counter**
  T0: Basic timer.
  TC0: Auto-reload timer/counter/PWM0/Buzzer output.
  TC1: Auto-reload timer/counter/PWM0/Buzzer output.
- **RTC: Real Time Clock from External Low clock**
  RTC timer of 0.5 sec.
- **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 10MHz.
  External high clock: Crystal type up to 16MHz.
  External low clock: 32KHz crystal/RC type
- **Four operation modes**
  Normal mode: Both high and low clock active.
  Slow mode: 32KHz low clock active.
  Sleep mode: Both high and low clock stop.
  Green mode: Periodical wake-up by T0 timer.
- **Package (Chip form support)**
  LQFP 48 pins.
  SSOP 48 pins
  P-DIP 48 pins
  LQFP 64 pins

☞ **Features Selection Table**

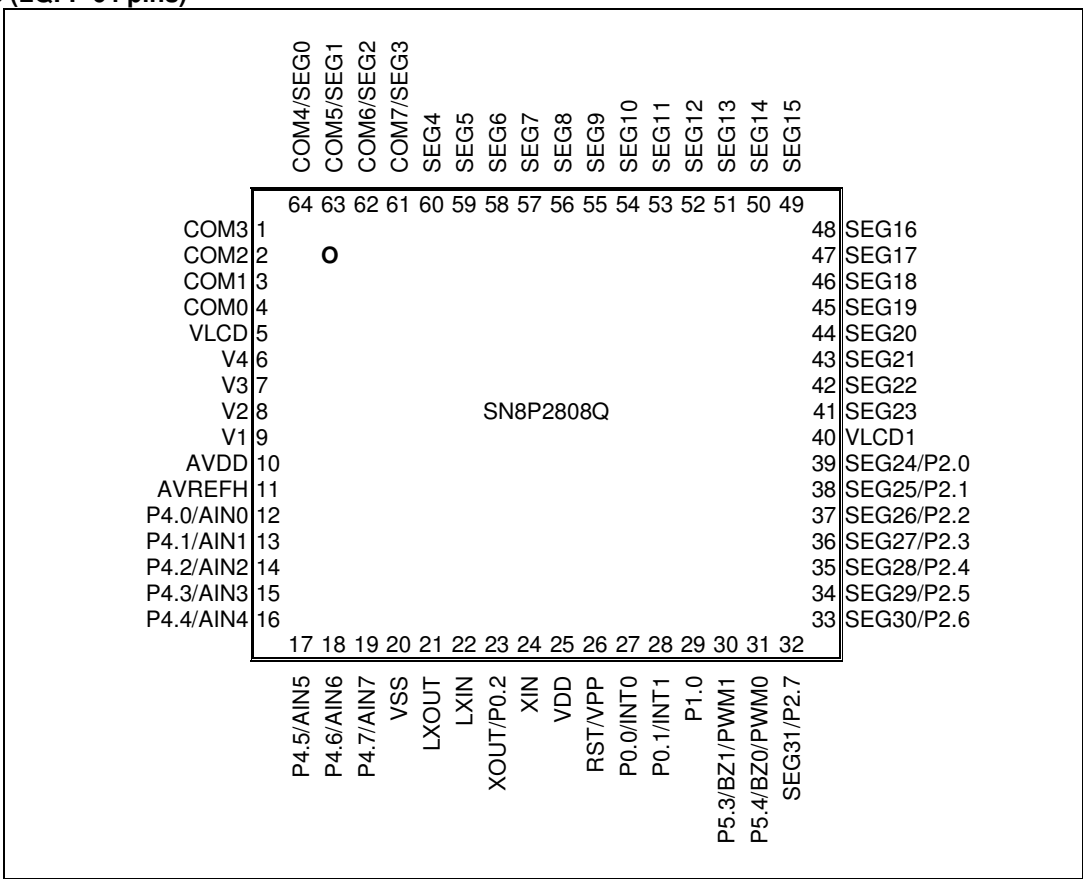
CHIP	ROM	RAM	Stack	Timer			I/O	ADC ch	LCD	PWM Buzzer	Wakeup Pin no.	Package
				T0	TC0	TC1						
SN8P2807	6K*16	256	8	Y	Y	Y	21	8	4*16/8*12	2	4	LQFP48
SN8P2808	6K*16	256	8	Y	Y	Y	21	8	4*32/8*28	2	4	LQFP64

1.2 SYSTEM BLOCK DIAGRAM

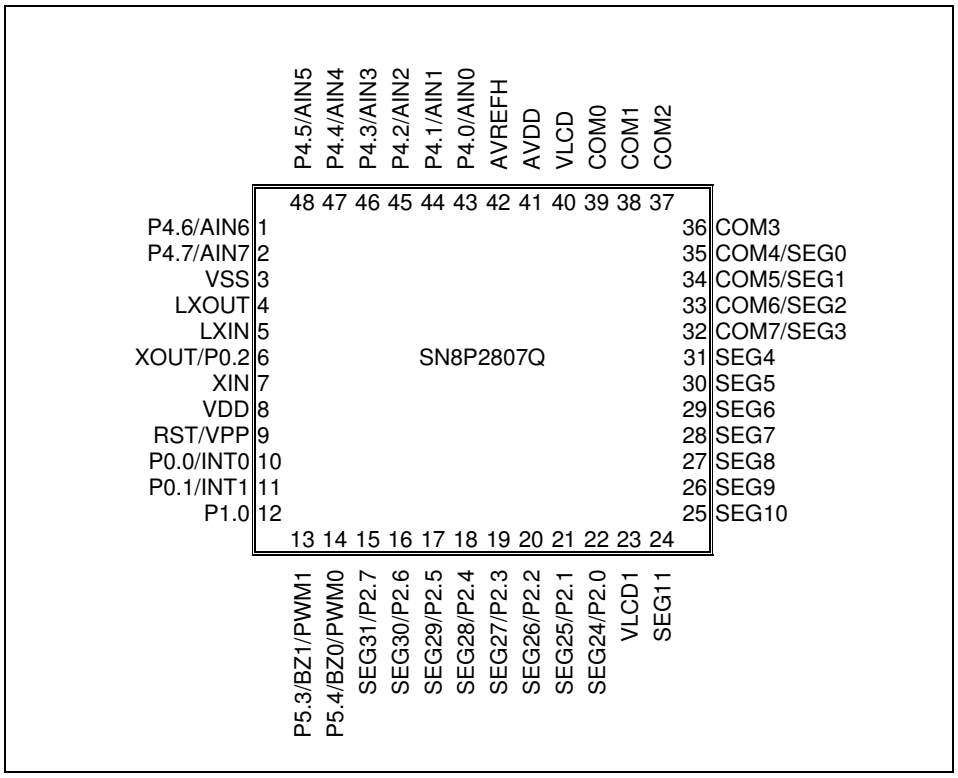


1.3 PIN ASSIGNMENT

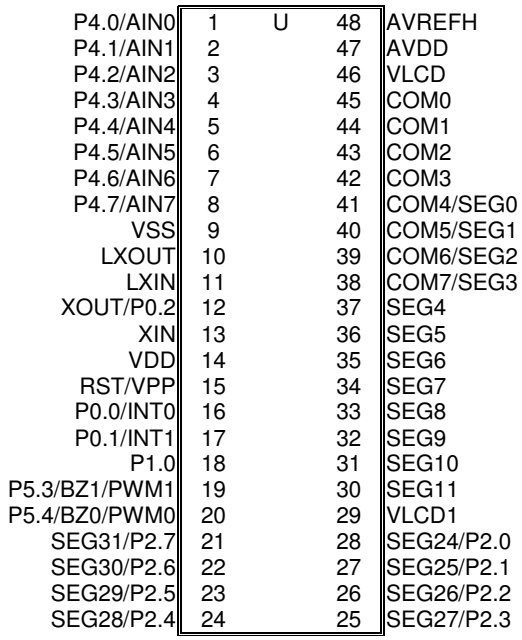
SN8P2808 (LQFP 64 pins)



SN8P2807 (LQFP48)



SN8P2807(SSOP 48/DIP 48)



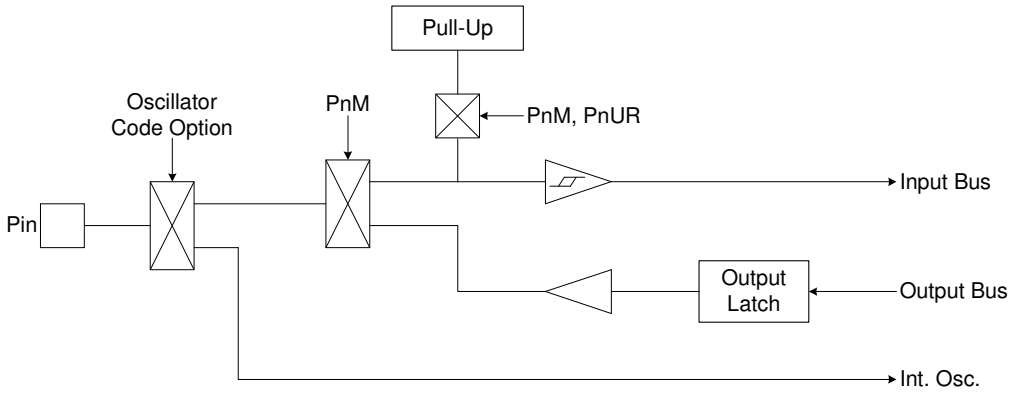
SN8P2807X
SN8P2807P

1.4 PIN DESCRIPTIONS

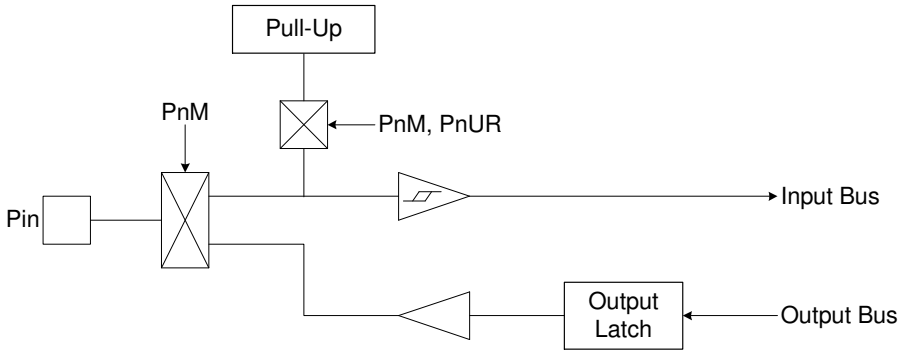
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital circuit.
VLCD	P	LCD power supply
VLCD1	P	The power of P28~35, (SEG24/P2.0~SEG31/P2.7) Connect VLCD1 to VLCD when P28~35 as LCD Segment Connect VLCD1 to VDD when P28~35 as I/O port.
V1, V2, V3, V4	P	VLVD bias voltage. Connect the external resistors to adjust VLCD voltage.
RST/VPP/P0.3	I, P	RST: System reset input pin. Schmitt trigger structure, low active, normal operation must stay to “high”. VPP: OTP programming pin.
XIN	I	Oscillator input pin while external oscillator enable (crystal and RC).
XOUT/P0.2	I/O	Port 0.2 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor. Built- in wake-up function. XOUT: Oscillator output pin while external crystal enable.
LXIN	I	External low oscillator input pin (32768Hz crystal or RC).
LXOUT	O	External low oscillator output pin.
P0.0/INT0	I/O	Port 0.0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Built- in wake-up 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 wake-up function. INT1 trigger pin (Schmitt trigger). TC1 event counter clock input pin.
P1.0	I/O	Built-in pull-up resisters. Built- in wake-up function.
P2[7:0]/SEG24~SEG32	I/O	Segment pin SEG24~32 Share with Port2. The power came from VLCD1.
P4[7:0]	I/O	Bi-direction pins/pull-up/ADC input/without Schmitt trigger input
P5[3:4]	I/O	Bi-direction pins/pull-up/Schmitt trigger input P5.4: PWM0/BZ0, P5.3: PWM1/BZ1,
COM0~COM3	O	LCD driver’s COM pin
COM4/SEG0~COM7/SEG3	O	LCD driver’s COM4~7 share with SEG0~3.
SEG4~SEG23	O	LCD driver’s SEG pin

1.5 PIN CIRCUIT DIAGRAMS

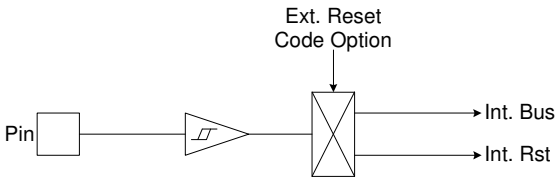
Port 0.2 structure:



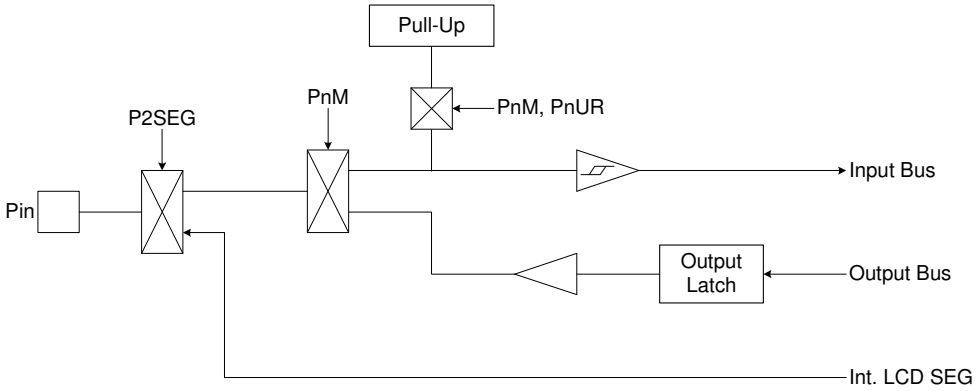
Port 0, 1 structure:



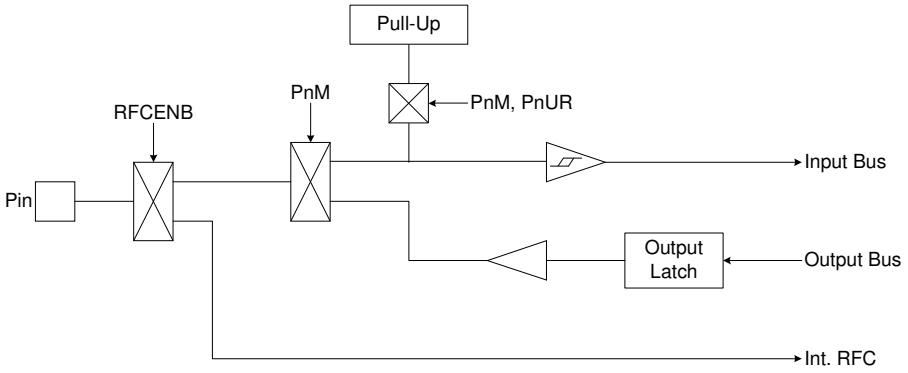
Port 0.3 structure:



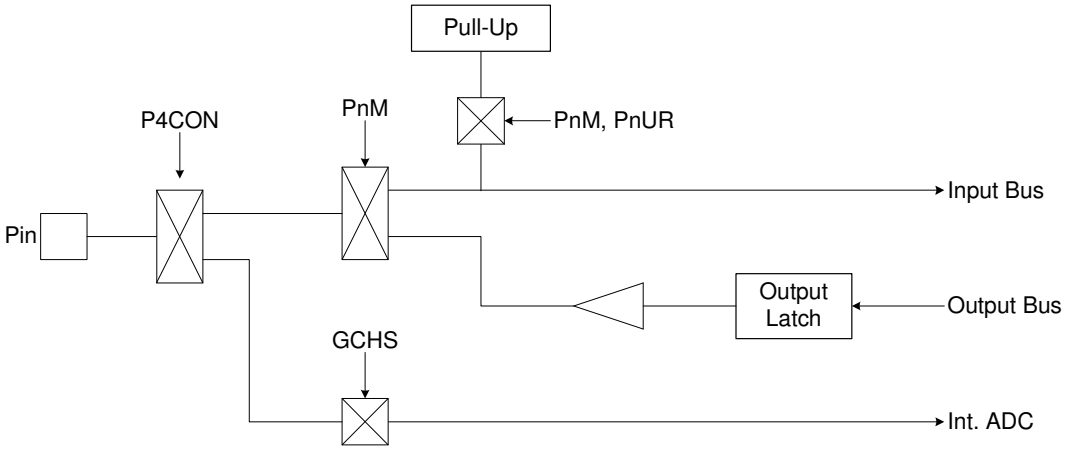
Port 2 structure:



Port 5.3/5.4 structure:



Port 4 structure:

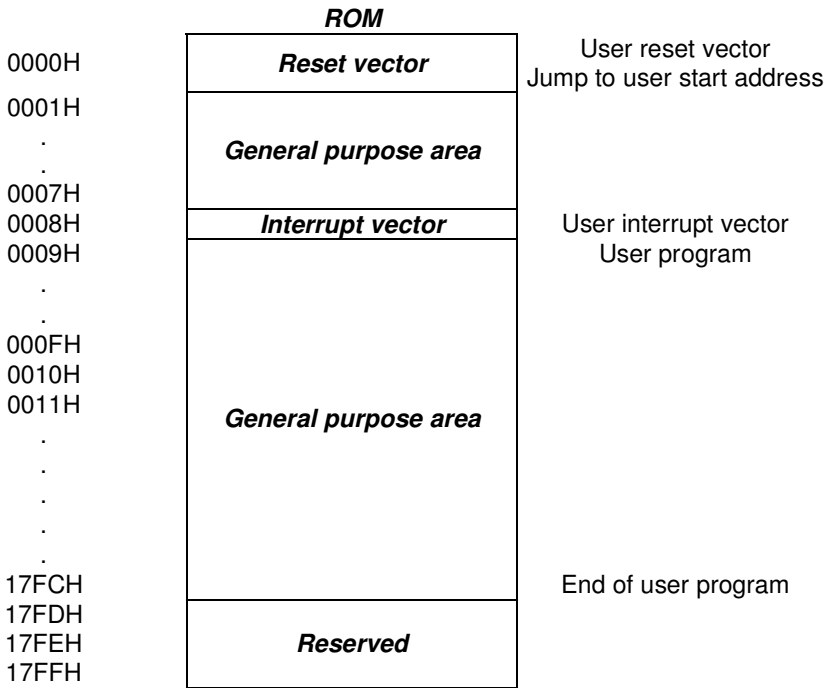


2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ ***6K 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.***

2.1.4 LOOK-UP TABLE DESCRIPTION

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

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

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

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

➢ **Example: INC_YZ macro.**

```
INC_YZ          MACRO
                INCMS   Z               ; Z+1
                JMP     @F              ; Not overflow
                INCMS   Y               ; Y+1
                NOP                     ; Not overflow
@@:
                ENDM
```

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

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

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

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

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

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

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

                B0BTS1    FC              ; Check the carry flag.
                JMP       GETDATA         ; FC = 0
                INCMS     Y               ; FC = 1. Y+1.
                NOP
GETDATA:                                  ;
                MOVC                      ; To lookup data. If BUF = 0, data is 0x0035
                                          ; If BUF = 1, data is 0x5105
                                          ; If BUF = 2, data is 0x2012
                …

TABLE1:         DW        0035H           ; To define a word (16 bits) data.
                DW        5105H
                DW        2012H
                …
```

2.1.5 JUMP TABLE DESCRIPTION

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

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

➢ **Example: Jump table.**

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

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

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

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

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

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

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

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

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

➢ **Example: "@JMP_A" operation.**

; Before compiling program.

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

; After compiling program.

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

2.1.5.1 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
                B0BCLR  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
```

2.1.6 CODE OPTION TABLE

Code Option	Content	Function Description
High_Clk	RC	Low cost RC for external high clock oscillator and XOUT becomes to P0.2 bit-direction I/O pin.
	32K X'tal	Low frequency, power saving crystal (e.g. 32.768K) for external high clock oscillator
	12M X'tal	High speed crystal /resonator (e.g. 12MHz) for external high clock oscillator.
	4M X'tal	Standard crystal /resonator (e.g. 4M) for external high clock oscillator.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
Fcpu	Fhosc/1	Instruction cycle is oscillator clock. Notice: In Fosc/1, Noise Filter must be disabled.
	Fhosc/2	Instruction cycle is 2 oscillator clocks. Notice: In Fosc/2, Noise Filter must be disabled.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
Reset_Pin	Reset	Enable External reset pin.
	P03	Enable P0.3 input only without pull-up resister.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
Noise_Filter	Enable	Enable Noise Filter and the Fcpu is Fosc/4~Fosc/8.
	Disable	Disable Noise Filter and the Fcpu is Fosc/1~Fosc/8.
LVD	LVD_L	LVD will reset chip if VDD is below 2.0V
	LVD_M	LVD will reset chip if VDD is below 2.0V Enable LVD24 bit of PFLAG register for 2.4V low voltage indicator.
	LVD_H	LVD will reset chip if VDD is below 2.4V Enable LVD36 bit of PFLAG register for 3.6V low voltage indicator.

✷ ***Note:***

1. ***In high noisy environment, enable "Noise Filter" and set Watch_Dog as "Always_On" is strongly recommended.***
2. ***Enable "Noise_Filter" will limit the Fcpu = Fosc/4, Fosc/8.***
3. ***Fcpu code option is only available for High Clock. Fcpu of slow mode is Flosc/1.***

2.1.7 DATA MEMORY (RAM)

☞ ***256 X 8-bit RAM***

	Address	RAM location	
BANK 0	000h “ “ “ “ “ 07Fh	***General purpose area***	
	080h “ “ “ “ “ 0FFh	***System register***	080h~0FFh of Bank 0 store system registers (128 bytes).
		End of bank 0 area	
BANK 1	100h “ “ “ 17Fh	***General purpose area***	
BANK 15	F00h “ “ “ F1Fh	***LCD RAM area***	0x00~0x1F of Bank 15 (32 byte) is LCD RAM registers.

2.1.8 SYSTEM REGISTER

2.1.8.1 SYSTEM REGISTER TABLE

Bank 0:

	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	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-		-	-	-	-	-	-	-	-	-	-	P4CON	VREFH
B	-	ADM	ADB	ADR	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M	-	P4M	P5M	-	-	INTRQ	INTEN	OSCM	LCDM	WDTR	TC0R	PCL	PCH
D	P0	P1	P2	-	P4	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	P2UR	-	P4UR	P5UR	@HL	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.1.8.2 SYSTEM REGISTER DESCRIPTION

R = Working register and ROM look-up data buffer.
PFLAG = ROM page and special flag register.
RBANK = RAM bank control register.
TC1M = TC1 mode register.
PnM = Port n input/output mode register.
INTRQ = Interrupt request register.
OSCM = Oscillator mode register.
TC0R = TC0 auto-reload data buffer.
Pn = Port n data buffer.
TC0M = TC0 mode register.
T0C = T0 counting register.
PnUR = Port n pull-up resister control register.
P4CON= Port 4 Configuration Register
ADB = ADC's data buffer.
STK0~STK7 = Stack 0 ~ stack 7 buffer.
TC1C = TC1 counting register.

Y, Z = Working, @YZ and ROM addressing register.
H, L = Working, @HL addressing register.
TC1R = TC1 auto-reload data buffer.
PEDGE = P0.0 edge direction register.
P1W = P1 wakeup control register.
INTEN = Interrupt enable register.
LCDM = LCD mode register.
WDTR = Watchdog timer clear register.
PCH, PCL = Program counter.
T0M = TC0/TC1 speed-up and TC0 wake-up function register.
TC0C = TC0 counting register.
STKP = Stack pointer buffer.
ADM = ADC's mode register.
ADR = ADC's resolution selects register.
@HL = RAM HL indirect addressing index pointer.
@YZ = RAM YZ indirect addressing index pointer.

2.1.8.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
085H	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0	R/W	X
086H	NT0	NPD	LVD36	LVD24		C	DC	Z	R/W	PFLAG
087H					RBNKS3	RBNKS2	RBNKS1	RBNKS0	R/W	RBANK
0AEH	P4CON7	P4CON6	P4CON 5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	W	P4CON
0AFH	EVHENB						VHS1	VHS0	R/W	VREFH
0B1H	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0	R/W	ADM mode register
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB data buffer
0B3H	-	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR register
0B8H						P02M	P01M	P00M	R/W	P0M
0BFH				P00G1	P00G0				R/W	PEDGE
0C0H								P10W	W	P1W
0C1H								P10M	R/W	P1M
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C5H				P54M	P53M				R/W	P5M
0C8H	ADCIRQ	TC1IRQ	TC0IRQ	T0IRQ			P01IRQ	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	TC1IEN	TC0IEN	T0IEN			P01IEN	P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CBH				COMSEL	RCLK	P2SEG	BIAS	LCDENB	R/W	LCDM
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				PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H					P03	P02	P01	P00	R/W	P0
0D1H								P10	R/W	P1
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2
0D4H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4
0D5H				P54	P53				R/W	P5
0D8H	T0ENB	T0RATE2	T0RATE1	T0RATE0	TC1X8	TC0X8	TC0GN	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
0E0H						P02R	P01R	P00R	W	P0UR
0E1H								P10R	W	P1UR
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2UR
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H				P54R	P53R				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
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H				S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H				S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H				S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H				S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H				S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH				S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH				S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH				S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ *Note:*

1. *To avoid system error, 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.*

2.1.9 LCD RAM

LCD RAM is mapping to every dots of COM & SEG combination in Bank 15. When 4 COM setting, only low nibble of one LCD RAM register is useful for COM0~COM3. When 8 COM setting, both high low nibble of one LCD RAM register is useful for COM0~COM8.

2.1.9.1 LCD RAM TABLE

Bank 15:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	LCD RAM AREA															
1																
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

2.1.9.2 BIT DEFINITION of LCD RAM

RAM bank 15 address vs. Common/Segment location.

RAM	Bit	0	1	2	3	4	5	6	7
Address	**LCD**	COM0	COM1	COM2	COM3	COM4	COM5	COM6	COM7
00h	SEG0	00h.0	00h.1	00h.2	00h.3	00h.4	00h.5	00h.6	00h.7
01h	SEG1	01h.0	01h.1	01h.2	01h.3	01h.4	01h.5	01h.6	01h.7
02h	SEG2	02h.0	02h.1	02h.2	02h.3	02h.4	02h.5	02h.6	02h.7
03h	SEG3	03h.0	03h.1	03h.2	03h.3	03h.4	03h.5	03h.6	03h.7
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
0Ch	SEG12	0Ch.0	0Ch.1	0Ch.2	0Ch.3	0Ch.4	0Ch.5	0Ch.6	0Ch.7
0Dh	SEG13	0Dh.0	0Dh.1	0Dh.2	0Dh.3	0Dh.4	0Dh.5	0Dh.6	0Dh.7
0Eh	SEG14	0Eh.0	0Eh.1	0Eh.2	0Eh.3	0Eh.4	0Eh.5	0Eh.6	0Eh.7
0Fh	SEG15	0Fh.0	0Fh.1	0Fh.2	0Fh.3	0Fh.4	0Fh.5	0Fh.6	0Fh.7
10h	SEG16	10h.0	10h.1	10h.2	10h.3	10h.4	10h.5	10h.6	10h.7
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
1Bh	SEG27	1Bh.0	1Bh.1	1Bh.2	1Bh.3	1Bh.4	1Bh.5	1Bh.6	1Bh.7
1Ch	SEG28	1Ch.0	1Ch.1	1Ch.2	1Ch.3	1Ch.4	1Ch.5	1Ch.6	1Ch.7
1Dh	SEG29	1Dh.0	1Dh.1	1Dh.2	1Dh.3	1Dh.4	1Dh.5	1Dh.6	1Dh.7
1Eh	SEG30	1Eh.0	1Eh.1	1Eh.2	1Eh.3	1Eh.4	1Eh.5	1Eh.6	1Eh.7
1Fh	SEG31	1Fh.0	1Fh.1	1Fh.2	1Fh.3	1Fh.4	1Fh.5	1Fh.6	1Fh.7

2.1.10 ACCUMULATOR

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

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

```
; Read ACC data and store in BUF data memory.
                MOV     BUF, A

; Write a immediate data into ACC.
                MOV     A, #0FH

; Write ACC data from BUF data memory.
                MOV     A, BUF
; or
                B0MOV   A, BUF
```

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

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

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

                RETI                    ; Exit interrupt service vector
```

2.1.11 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. LVD24, LVD36 bits indicate LVD detecting power voltage status.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

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

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

Bit 5 **LVD36:** LVD 3.6V operating flag and only support LVD code option is LVD_H.
0 = Inactive (VDD > 3.6V).
1 = Active (VDD <= 3.6V).

Bit 4 **LVD24:** LVD 2.4V operating flag and only support LVD code option is LVD_M.
0 = Inactive (VDD > 2.4V).
1 = Active (VDD <= 2.4V).

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.***

2.1.12 PROGRAM COUNTER

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

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

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

☞ **ONE ADDRESS SKIPPING**

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

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

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

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

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

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

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

INCS instruction:

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

INCMS instruction:

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

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

DECS instruction:

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

DECMS instruction:

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

☞ **MULTI-ADDRESS JUMPING**

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

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

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

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

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

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

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

2.1.13 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
        …
        …
```

2.1.13.1 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.***

2.1.13.2 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
        …
```

2.1.13.3 R REGISTERS

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

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

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

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

2.2 ADDRESSING MODE

2.2.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.2.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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

2.2.3 INDIRECTLY ADDRESSING MODE

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

Example: Indirectly addressing mode with @HL register

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

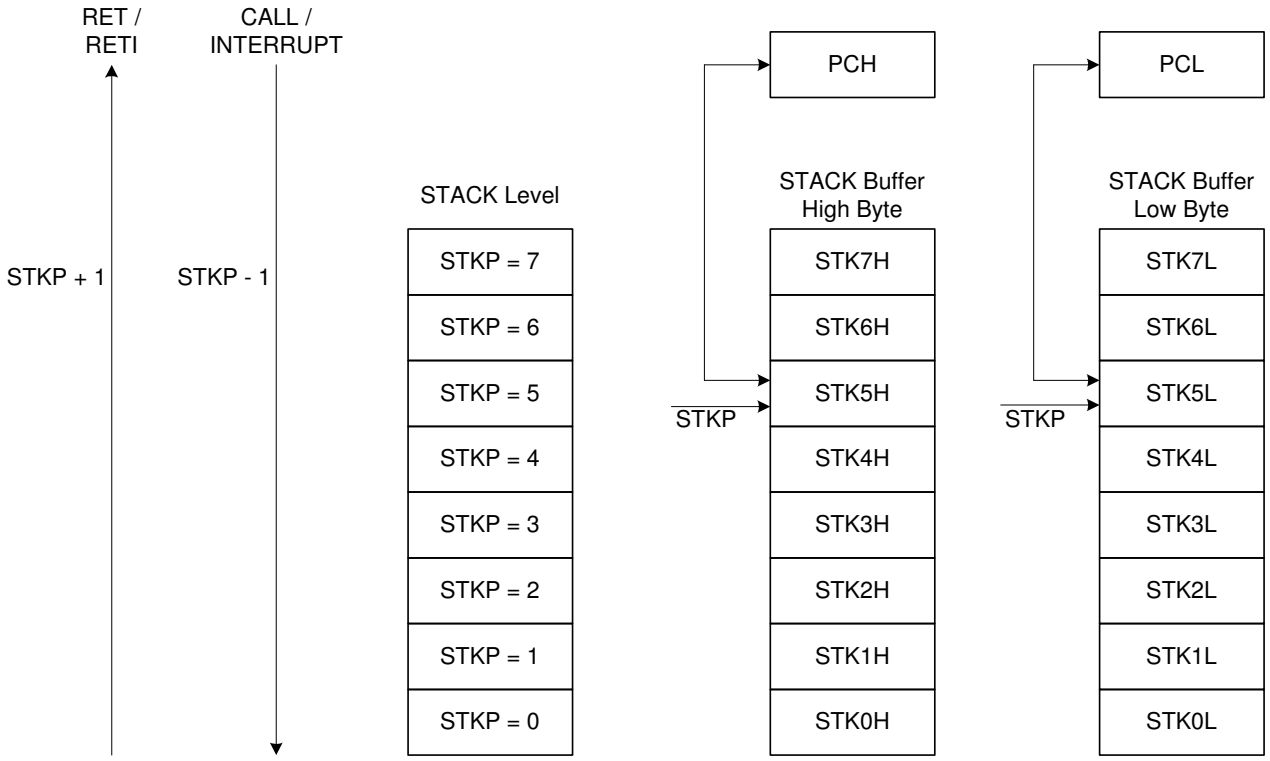
Example: Indirectly addressing mode with @YZ register

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

2.3 STACK OPERATION

2.3.1 OVERVIEW

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



2.3.2 STACK REGISTERS

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

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

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

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

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

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

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

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

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

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

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

2.3.3 STACK OPERATION EXAMPLE

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

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

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

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

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

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

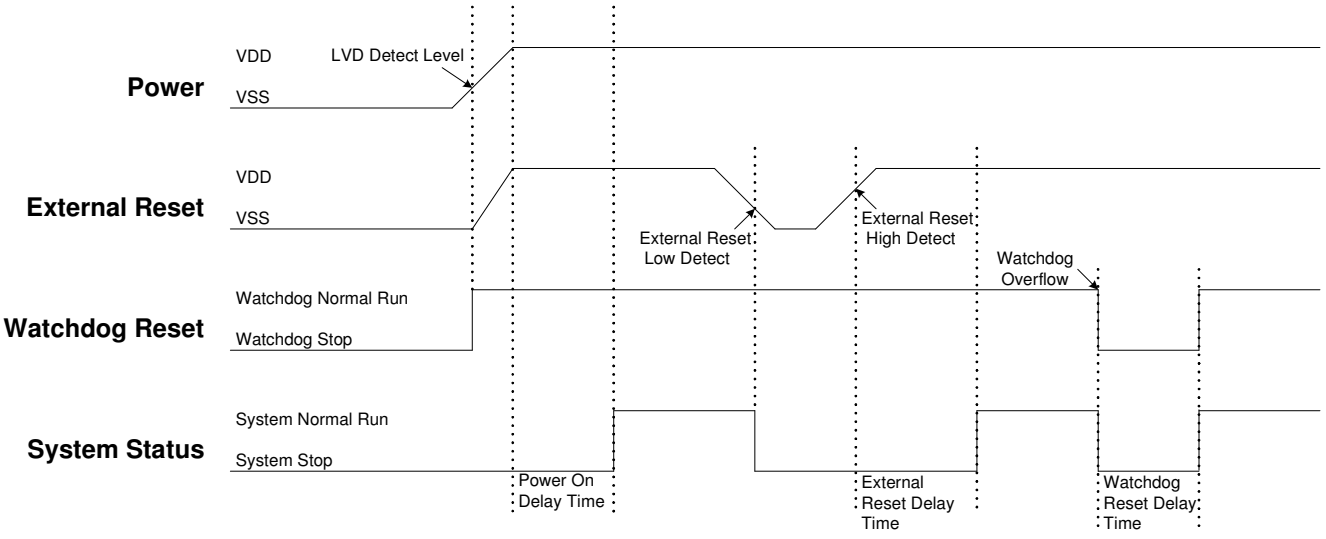
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

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

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

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



3.2 POWER ON RESET

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

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

3.3 WATCHDOG RESET

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

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

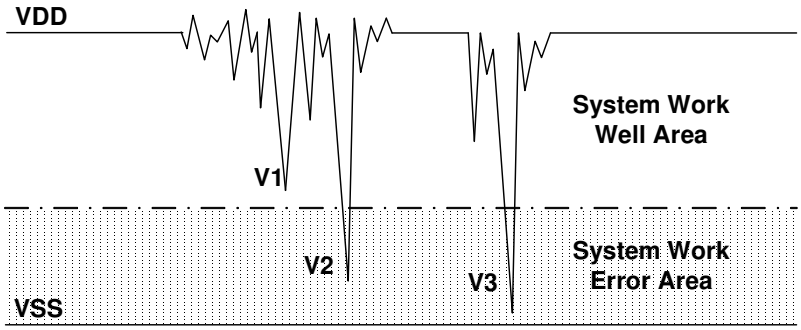
Watchdog timer application note is as following.

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

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

3.4 BROWN OUT RESET

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



Brown Out Reset Diagram

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

DC application:

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

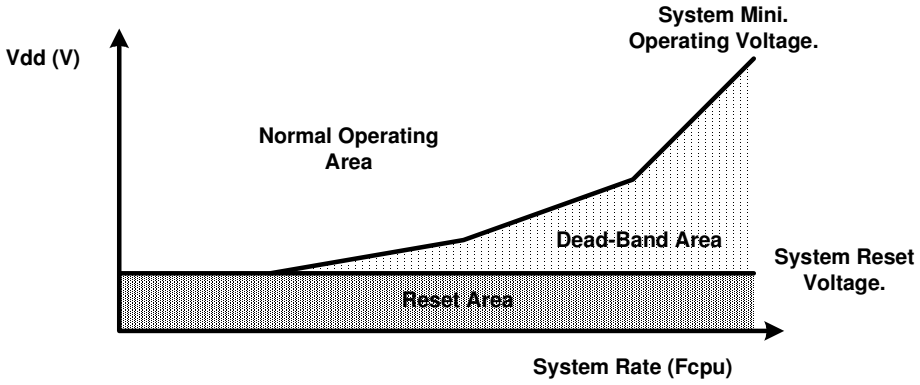
AC application:

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

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

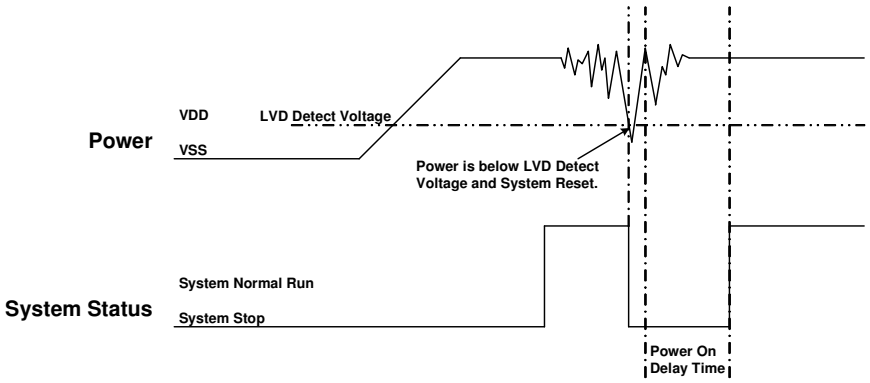
3.4.1 THE SYSTEM OPERATING VOLTAGE

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



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

3.4.2 LOW VOLTAGE DETECTOR (LVD)



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

The LVD is three levels design (2.0V/2.4V/3.6V) and controlled by LVD code option. The 2.0V LVD is always enable for power on reset and Brown Out reset. The 2.4V LVD includes LVD reset function and flag function to indicate VDD status function. The 3.6V includes flag function to indicate VDD status. LVD flag function can be an **easy low battery detector**. LVD24, LVD36 flags indicate VDD voltage level. For low battery detect application, only checking LVD24, LVD36 status to be battery status. This is a cheap and easy solution.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

Bit 5 **LVD36:** LVD 3.6V operating flag and only support LVD code option is LVD_H.
0 = Inactive (VDD > 3.6V).
1 = Active (VDD <= 3.6V).

Bit 4 **LVD24:** LVD 2.4V operating flag and only support LVD code option is LVD_M.
0 = Inactive (VDD > 2.4V).
1 = Active (VDD <= 2.4V).

LVD	LVD Code Option		
	LVD_L	LVD_M	LVD_H
2.0V Reset	Available	Available	Available
2.4V Flag	-	Available	-
2.4V Reset	-	-	Available
3.6V Flag	-	-	Available

LVD_L
If VDD < 2.0V, system will be reset.
Disable LVD24 and LVD36 bit of PFLAG register.
LVD_M
If VDD < 2.0V, system will be reset.
Enable LVD24 bit of PFLAG register. If VDD > 2.4V, LVD24 is "0". If VDD <= 2.4V, LVD24 flag is "1".
Disable LVD36 bit of PFLAG register.
LVD2_H
If VDD < 2.4V, system will be reset.
Enable LVD24 bit of PFLAG register. If VDD > 2.4V, LVD24 is "0". If VDD <= 2.4V, LVD24 flag is "1".
Enable LVD36 bit of PFLAG register. If VDD > 3.6V, LVD36 is "0". If VDD <= 3.6V, LVD36 flag is "1".

✷ ***Note:***
1. ***After any LVD reset, LVD24, LVD36 flags are cleared.***
2. ***The voltage level of LVD 2.4V or 3.6V is for design reference only. Don't use the LVD indicator as precision VDD measurement.***

3.4.3 BROWN OUT RESET IMPROVEMENT

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

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

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

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

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

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

3.5 EXTERNAL RESET

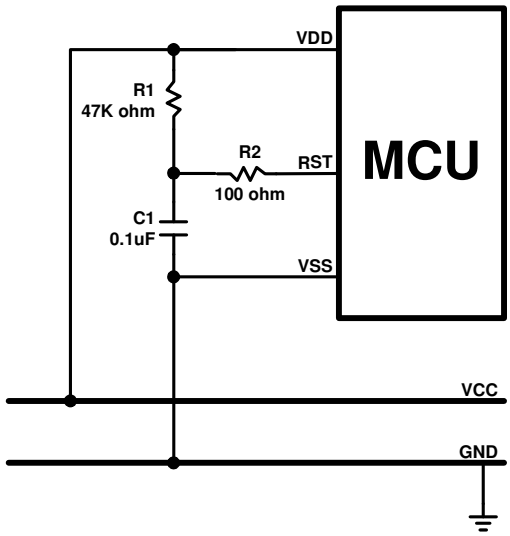
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the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

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

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

3.6 EXTERNAL RESET CIRCUIT

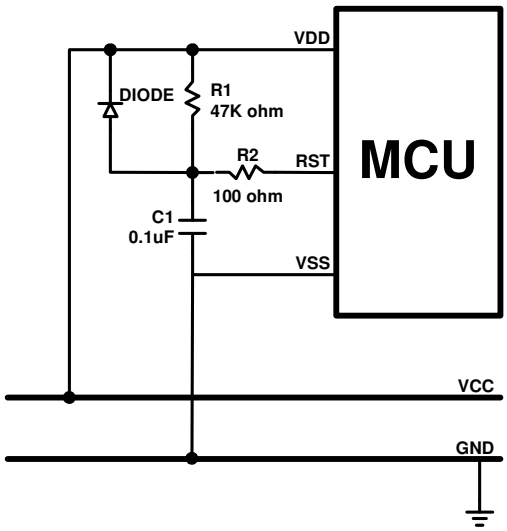
3.6.1 Simply RC Reset Circuit



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

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

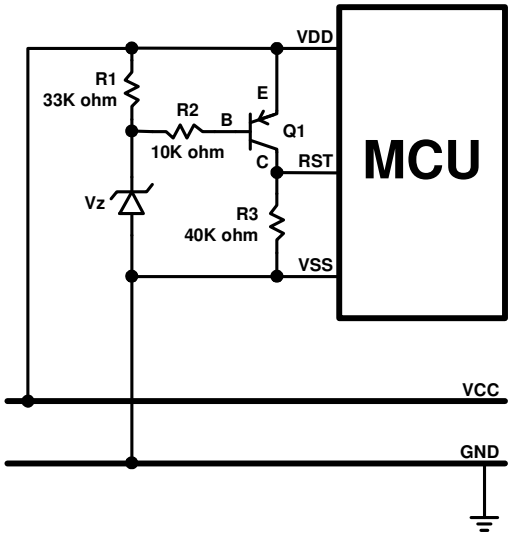
3.6.2 Diode & RC Reset Circuit



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

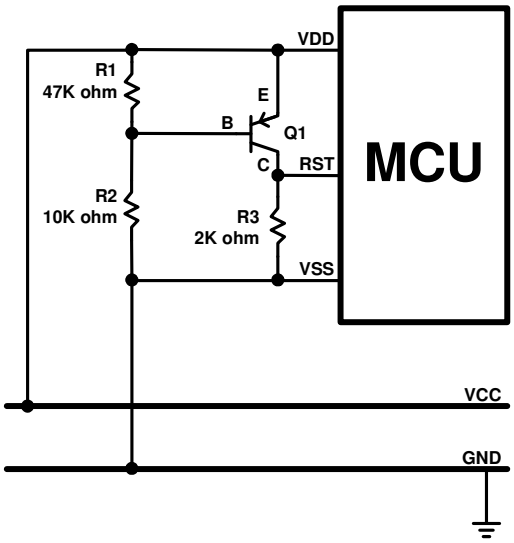
✸ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.6.3 Zener Diode Reset Circuit



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

3.6.4 Voltage Bias Reset Circuit

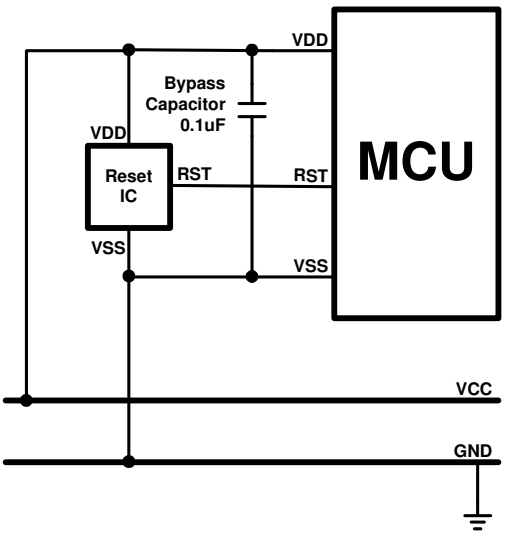


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to "0.7V x (R1 + R2) / R1",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

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

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

3.6.5 External Reset IC



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

4 SYSTEM CLOCK

4.1 OVERVIEW

The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit. The low-speed clock is generated from external low-speed 32768Hz oscillator circuit.

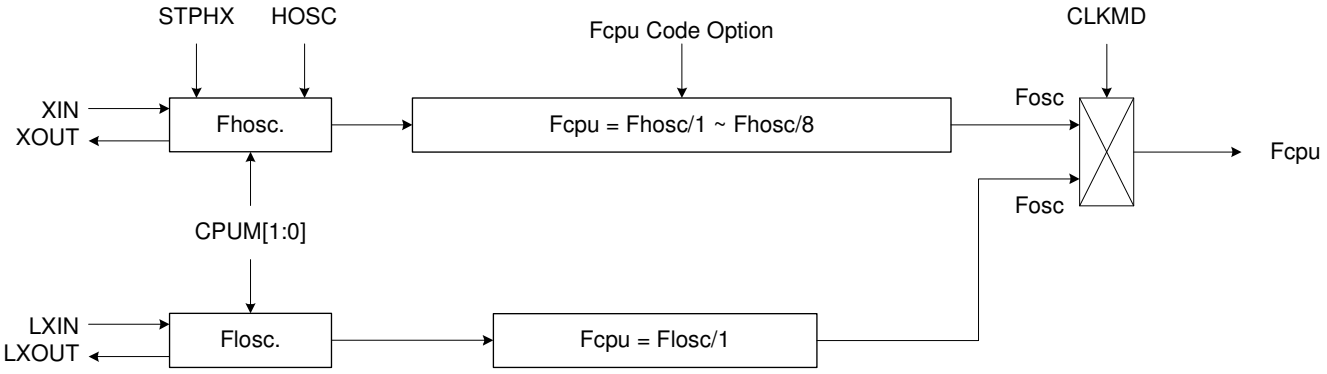
Both the high-speed clock and the low-speed clock can be system clock (Fosc). The system clock in slow mode is fixed Flosc (32768Hz) to be the instruction cycle (Fcpu).

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

☞ **Slow Mode (Low Clock):** **Fcpu = Flosc/1.**

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.

4.2 CLOCK BLOCK DIAGRAM



- HOSC: High_Clk code option.
- Fhosc: External high-speed clock.
- Flosc: External 32768Hz low-speed clock.
- 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	0	0	0	CPUM1	CPUM0	CLKMD	STPHX	0
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

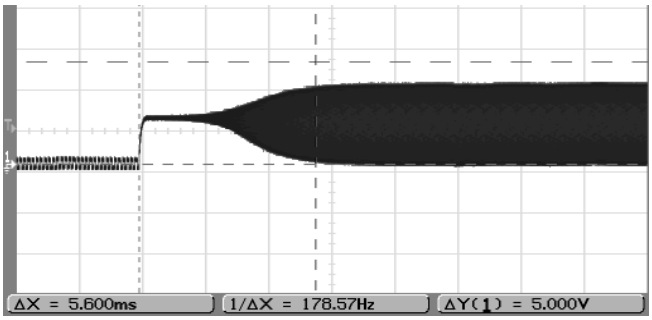
The system high clock is from external oscillator. The high clock type is controlled by “High_Clk” code option.

High_Clk Code Option	Description
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.

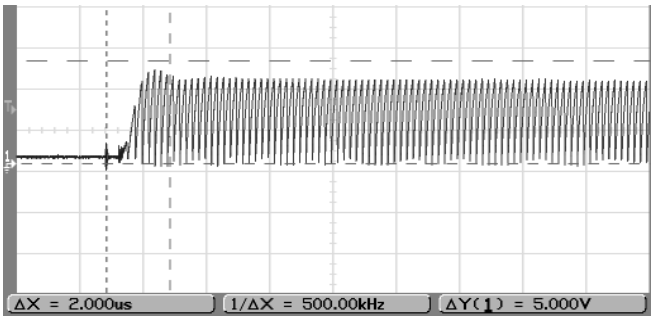
4.4.1 EXTERNAL HIGH CLOCK

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.

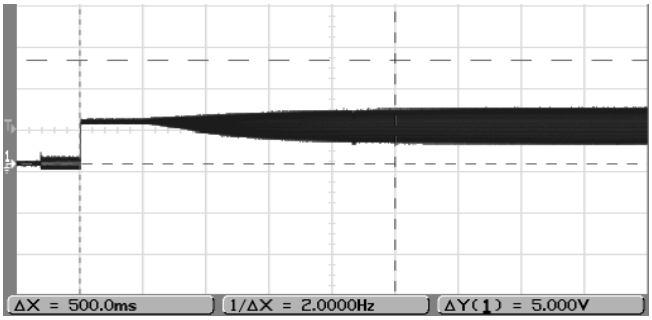
4MHz Crystal



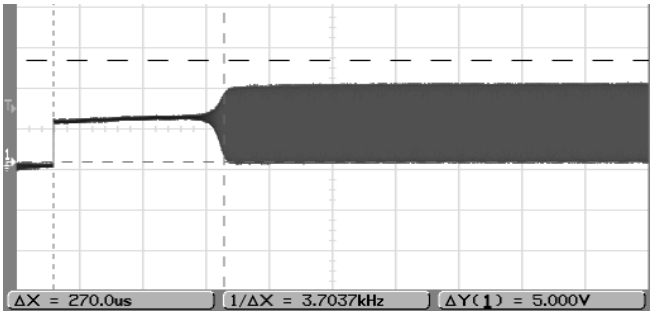
RC



32768Hz Crystal

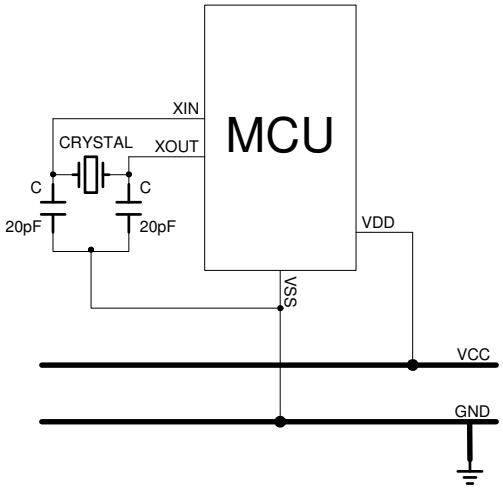


4MHz Ceramic



4.4.1.1 CRYSTAL/CERAMIC

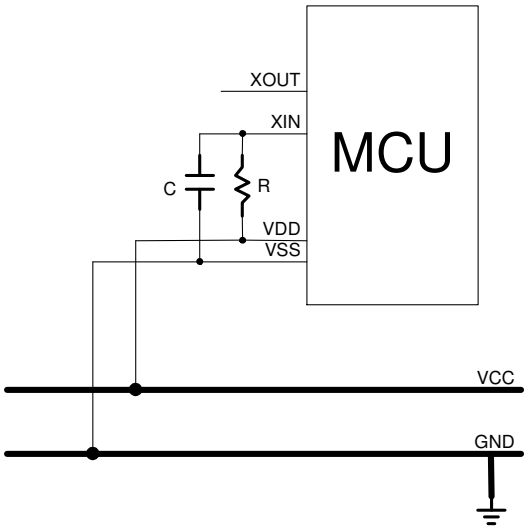
Crystal/Ceramic devices are driven by XIN, XOUT pins. For high/normal/low frequency, the driving currents are different. High_Clk code option supports different frequencies. 12M option is for high speed (ex. 12MHz). 4M option is for normal speed (ex. 4MHz). 32K option is for low speed (ex. 32768Hz).



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

4.4.1.2 RC

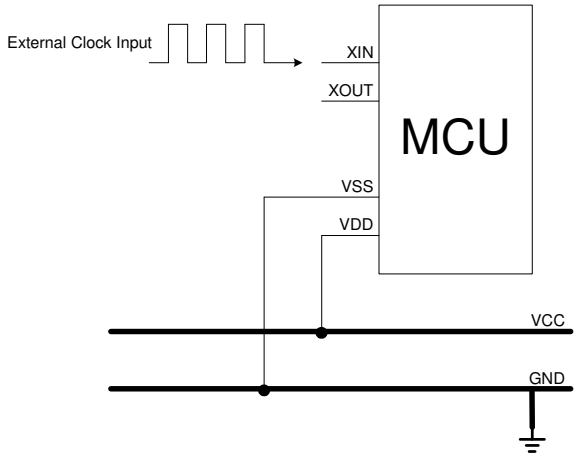
Selecting RC oscillator is by RC option of High_Clk code option. RC type oscillator's frequency is up to 10MHz. Using "R" value is to change frequency. 50P~100P is good value for "C". XOUT pin is general purpose I/O pin.



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

4.4.1.3 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.



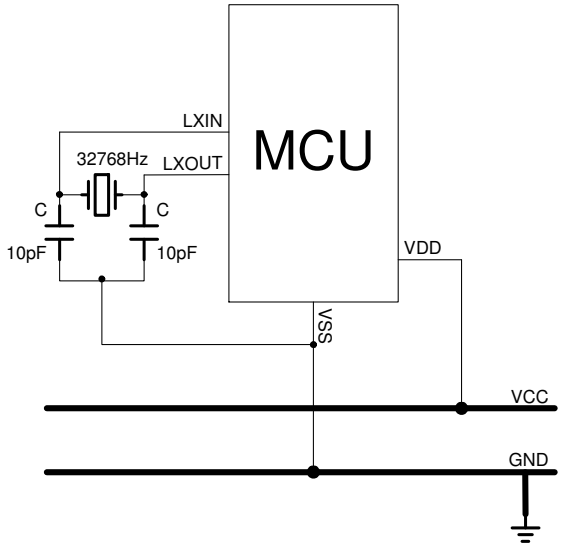
➢ ***Note: The GND of external oscillator circuit must be as near as possible to VSS pin of micro-controller.***

4.5 SYSTEM LOW CLOCK

System low clock is from external low-speed oscillator. External low clock includes three modules (Crystal/Ceramic, RC and external clock signal). The low clock oscillator module is controlled by RCLK bit of LCDM register. The external low clock supports system low clock source and LCD scanning clock source.

4.5.1 CRYSTAL/CERAMIC

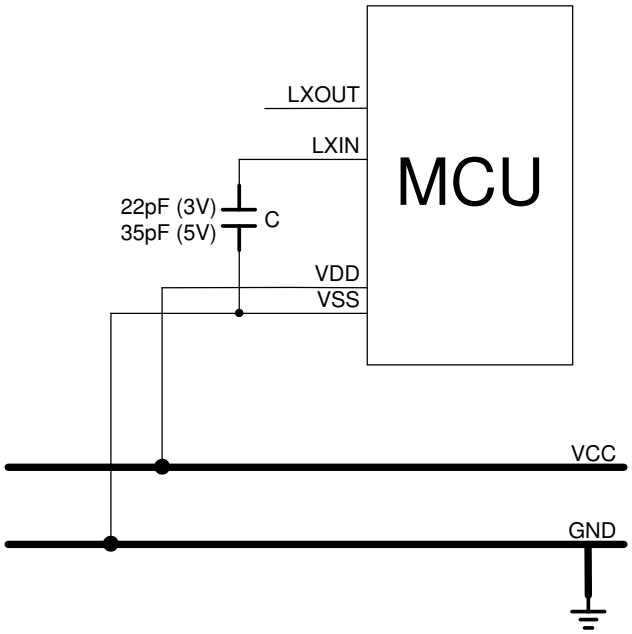
Crystal/Ceramic devices are driven by LXIN, LXOUT pins and only supports 32768Hz oscillator.



- ***Note: Connect the Crystal/Ceramic and C as near as possible to the LXIN/LXOUT/VSS pins of micro-controller. The capacitor between LXIN/LXOUT and VSS must be 10pF.***

4.5.2 RC

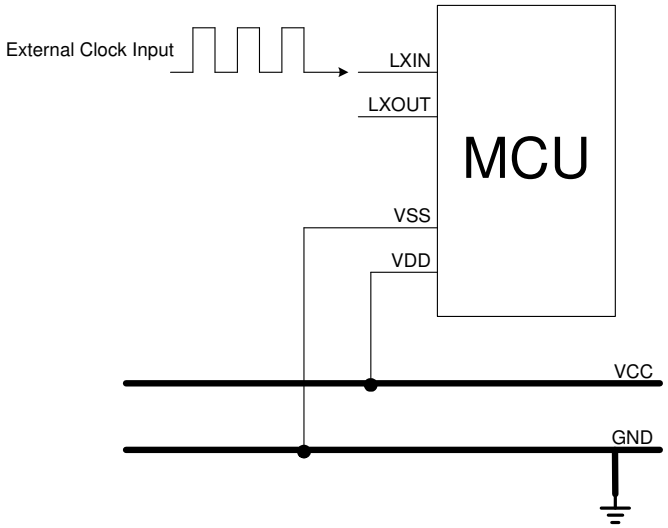
Selecting RC oscillator is by RCLK bit of LCDM register. RC type oscillator's frequency is 32768Hz. The external 32K RC type circuit only needs a capacitor, and don't connect a resistor between LXIN and VDD. **22pF @3V** and **35pF @5V** is a good value for "C" connected from LXIN to VSS. LXOUT pin is useless in external low RC mode.



➢ ***Note: Connect the C as near as possible to the VSS pin of micro-controller. The frequency of external low RC is decided by the capacitor value. Adjust capacitor value to about 32KHz frequency.***

4.5.3 EXTERNAL CLOCK SIGNAL

Selecting external clock signal input to be system clock is by RCLK bit of LCDM register. The external clock signal is input from LXIN pin. LXOUT pin is useless.



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

4.5.4 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
```

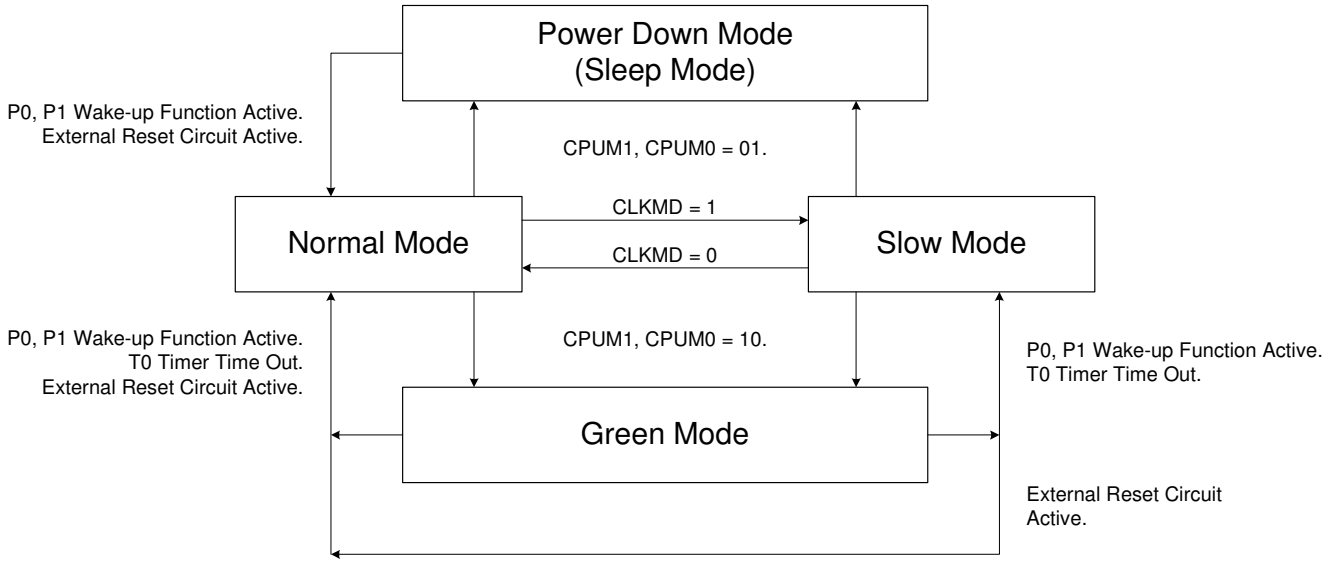
➢ ***Note: Do not measure the RC frequency directly from XIN; the probe impendence will affect the RC frequency.***

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/TC0 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/TC0 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 2048 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 * 2048 (sec) + high clock start-up time

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

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

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

✸ ***Note: The wakeup function of P0 can't be disabled. Make sure enable the pull-up resistor of P0 or use external pull-up resistors before enter sleep mode.***

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	-	-	-	-	-	-	-	P10W
Read/Write	-	-	-	-	-	-	-	W
After reset	-	-	-	-	-	-	-	0

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

6 I/O PORT

6.1 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	-	-	-	-	-	-	-	P10M
Read/Write	-	-	-	-	-	-	-	R/W
After reset	-	-	-	-	-	-	-	0

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

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

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	-	P54M	P53M	-	-	-
Read/Write	-	-	-	R/W	R/W	-	-	-
After reset	-	-	-	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:***
1. ***Users can program them by bit control instructions (B0BSET, B0BCLR).***
2. ***P0.3 input only pin, and the P0M.3 keeps "1".***
3. ***P2 is shared with SEG24~SEG31 controlled by P2SEG bit of LCDM register.***
4. ***After power on or reset, P3 default state as input LOW.***

➢ **Example: I/O mode selecting**

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

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

B0BCLR  P1M.0       ; Set P1.0 to be input mode.

B0BSET  P1M.0       ; Set P1.0 to be output mode.
```

➢ **Example: P2 I/O mode selecting**

```
B0BSET  FP2SEG      ; Enable P2 I/O function.

CLR     P2M         ; Set P2 port to be input mode.

MOV     A, #0FFH    ; Set P2 port to be output mode.
B0MOV   P2M, A

B0BCLR  P2M.0       ; Set P2.0 to be input mode.

B0BSET  P2M.0       ; Set P2.0 to be output mode.
```

6.2 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	-	-	-	-	-	-	-	P10R
Read/Write	-	-	-	-	-	-	-	W
After reset	-	-	-	-	-	-	-	0

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

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

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

✸ ***Note: P0.3 is input only pin and without pull-up resistor. The P0UR.3 keeps "1".***

➢ **Example: I/O Pull up Register**

```
MOV     A, #0FFH        ; Enable Port0, 1, 2, 5 Pull-up register,
B0MOV   P0UR, A         ;
B0MOV   P1UR, A
B0MOV   P2UR, A
B0MOV   P5UR, A
```

6.3 I/O PORT DATA REGISTER

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

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	-	-	-	-	-	-	-	P10
Read/Write	-	-	-	-	-	-	-	R/W
After reset	-	-	-	-	-	-	-	0

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

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

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

✸ ***Note: P0.3 keeps "1" when external reset enable by code option.***

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

```
B0MOV   A, P0   ; Read data from Port 0
B0MOV   A, P1   ; Read data from Port 1
B0MOV   A, P2   ; Read data from Port 2
B0MOV   A, P5   ; Read data from Port 5
```

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

```
MOV     A, #0FFH    ; Write data FFH to all Port.
B0MOV   P0, A
B0MOV   P1, A
B0MOV   P2, A
B0MOV   P5, A
```

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

```
B0BSET  P1.0    ; Set P1.0 and P5.3 to be "1".
B0BSET  P5.3

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

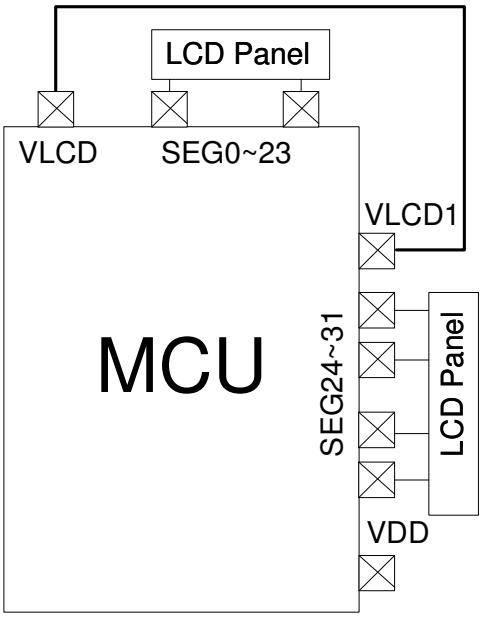
6.4 PORT2/LCD REGISTER

0CBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM	-	-	-	COMSEL	RCLK	P2SEG	BIAS	LCDENB
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

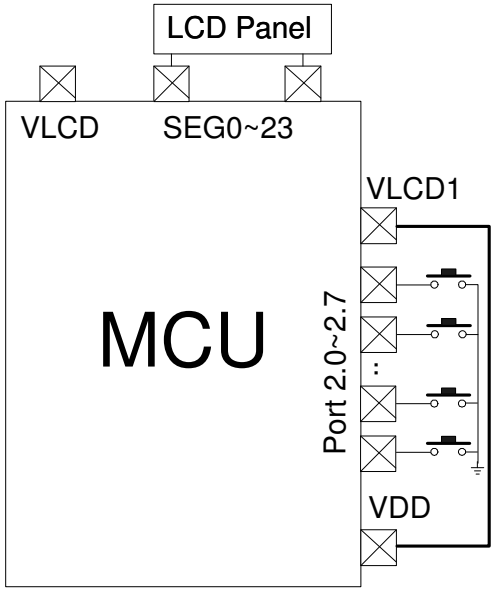
Bit 2 **P2SEG:** Seg24~Seg31 shared with P2.0~P2.7 control bit.
0 = Enable Seg24~Seg 31 pins.
1 = Enable P2.0~P2.7 pins.

✷ ***Note: Segment 24~Segment 31 pins are shared with P2.0~P2.7. When these pins are used as Port2 general purpose I/O mode, the P2SEG bit of LCDM register must be set as "1".***

Register P2SEG can select SEG24~SEG31 be as LCD segment or as Port 2 for IO application. When set P2SEG=0, these pins are for LCD application, connect VLCD1 to VLCD. When set P2SEG=1, these pins are for IO port application, connect VLCD1 to VDD.



P2SEG=0

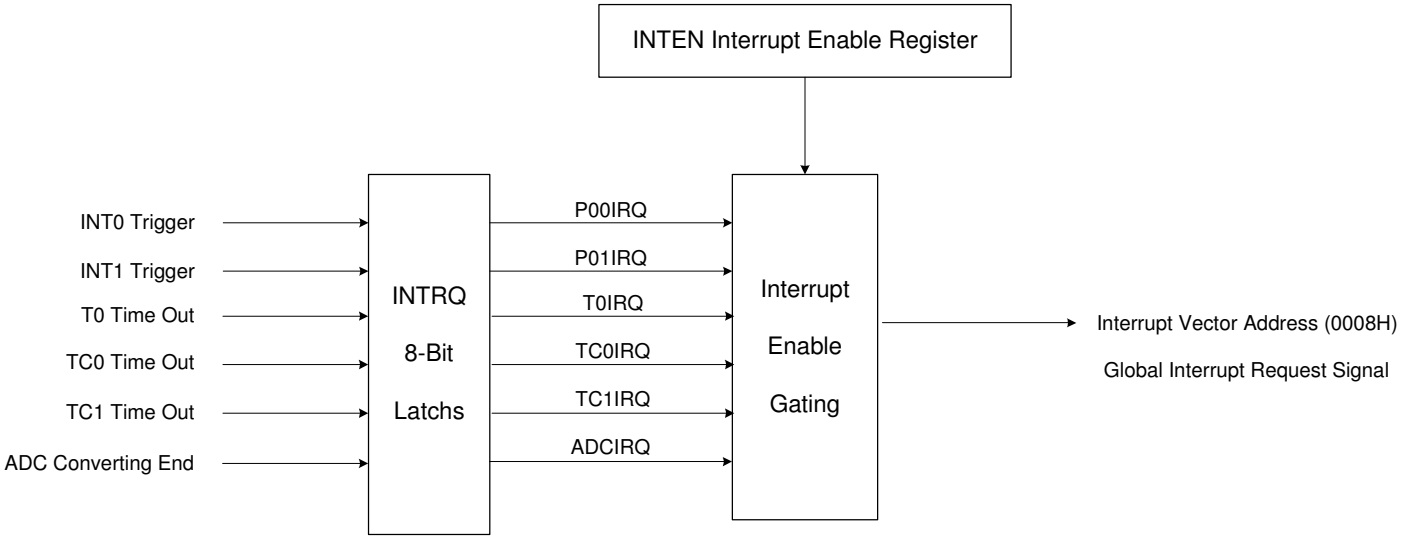


P2SEG=1

7 INTERRUPT

7.1 OVERVIEW

This MCU provides eight interrupt sources, including four internal interrupt (T0/TC0/TC1/ADC) and two external interrupt (INT0/INT1). The external interrupt can wakeup the chip while the system is switched from power down mode to high-speed normal mode, 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.***

7.2 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	-	-	P01IEN	P00IEN
Read/Write	R/W	R/W	R/W	R/W	-	-	R/W	R/W
After reset	0	0	0	0	-	-	0	0

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

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

Bit 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.

7.3 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	-	-	P01IRQ	P00IRQ
Read/Write	R/W	R/W	R/W	R/W	-	-	R/W	R/W
After reset	0	0	0	0	-	-	0	0

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

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

Bit 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.

7.4 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.***

7.5 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

7.6 INT0 (P0.0) INTERRUPT OPERATION

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

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

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

Bit[4:3] **P00G[1:0]:** P0.0 interrupt trigger edge control bits.
00 = reserved.
01 = rising edge.
10 = falling edge.
11 = rising/falling bi-direction (Level change trigger).

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

```
        MOV     A, #18H
        B0MOV   PEDGE, A        ; Set INT0 interrupt trigger as bi-direction edge.

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

➢ **Example: INT0 interrupt service routine.**

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

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

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

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

        RETI                    ; Exit interrupt vector
```

7.7 INT1 (P0.1) INTERRUPT OPERATION

When the INT1 trigger occurs, the P01IRQ will be set to "1" no matter the P01IEN is enable or disable. If the P01IEN = 1 and the trigger event P01IRQ is also set to be "1". As the result, the system will execute the interrupt vector (ORG 8). If the P01IEN = 0 and the trigger event P01IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the P01IRQ is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

If the interrupt trigger direction is identical with wake-up trigger direction, the INT1 interrupt request flag (INT1IRQ) is latched while system wake-up from power down mode or green mode by P0.1 wake-up trigger. System inserts to interrupt vector (ORG 8) after wake-up immediately.

✸ ***Note: INT1 interrupt request can be latched by P0.1 wake-up trigger.***

✸ ***Note: The interrupt trigger direction of P0.1 is falling edge.***

➢ **Example: INT1 interrupt request setup.**

```
        B0BSET      FP01IEN     ; Enable INT1 interrupt service
        B0BCLR      FP01IRQ     ; Clear INT1 interrupt request flag
        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: INT1 interrupt service routine.**

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

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

        B0BTS1      FP01IRQ     ; Check P01IRQ
        JMP         EXIT_INT    ; P01IRQ = 0, exit interrupt vector

        B0BCLR      FP01IRQ     ; Reset P01IRQ
        …                       ; INT1 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

7.8 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 as no RTC function.

```
        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
```

✸ ***Note: 1. In RTC mode, clear T0IRQ must be after 1/2 RTC clock source (32768Hz), or the RTC interval time is error. The delay is about 16us and use T0 interrupt service routine executing time to be the 16us delay time.***
2. In RTC mode, don't reset T0C in interrupt service routine.

Example: T0 interrupt service routine with RTC function.

```
                ORG         8               ; Interrupt vector
                JMP         INT_SERVICE
INT_SERVICE:
                …                           ; Push routine to save ACC and PFLAG to buffers.
> 16us
                B0BTS1      FT0IRQ          ; Check T0IRQ
                JMP         EXIT_INT        ; T0IRQ = 0, exit interrupt vector
                …                           ; T0 interrupt service routine
                …                           ; The time must be longer than 16us.
                B0BCLR      FT0IRQ          ; Reset T0IRQ
                …
EXIT_INT:
                …                           ; Pop routine to load ACC and PFLAG from buffers.

                RETI                        ; Exit interrupt vector
```

7.9 TC0 INTERRUPT OPERATION

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

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

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

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

        B0BSET      FGIE            ; Enable GIE
```

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

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

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

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

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

        RETI                        ; Exit interrupt vector
```

7.10 TC1 INTERRUPT OPERATION

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

Example: TC1 interrupt request setup.

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

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

        B0BSET      FGIE            ; Enable GIE
```

Example: TC1 interrupt service routine.

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

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

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

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

        RETI                        ; Exit interrupt vector
```

7.11 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
```

7.12 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
T0IRQ	T0C overflow
TC0IRQ	TC0C overflow
TC1IRQ	TC1C overflow
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         INTT0CHK        ; Jump check to next interrupt
                B0BTS0      FP01IRQ         ; Check P01IRQ
                JMP         INTP01
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         INTADCHK        ; Jump check to next interrupt
                B0BTS0      FTC1IRQ         ; Check TC1IRQ
                JMP         INTT1           ; Jump to TC1 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
```

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

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

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

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

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

Watchdog timer application note is as following.

- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.

- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the best structure to enhance the watchdog timer function.

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

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

Correct:                                        ; I/O and RAM are correct. Clear watchdog timer and
                                                ; execute program.
                B0BSET          FWDRST          ; Only one clearing watchdog timer of whole program.
                …
                CALL            SUB1
                CALL            SUB2
                …
                …
                …
                JMP             MAIN
```

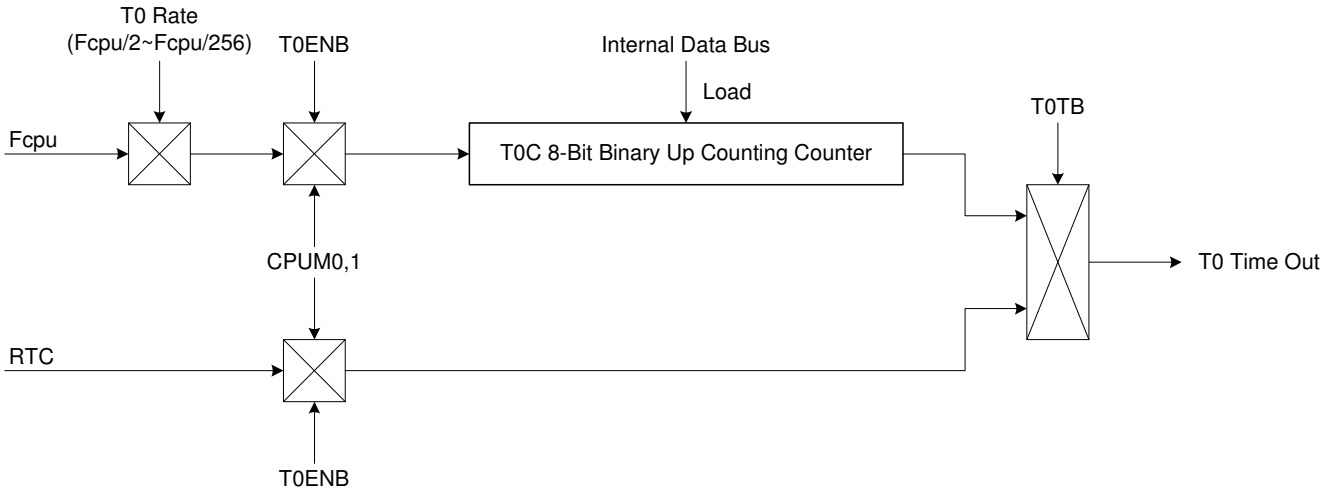
8.2 TIMER 0 (T0)

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	TC0GN	T0TB
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 **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 1 **TC0GN:** Enable TC0 Green mode wake up function
0 = Disable.
1 = Enable.

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.
```

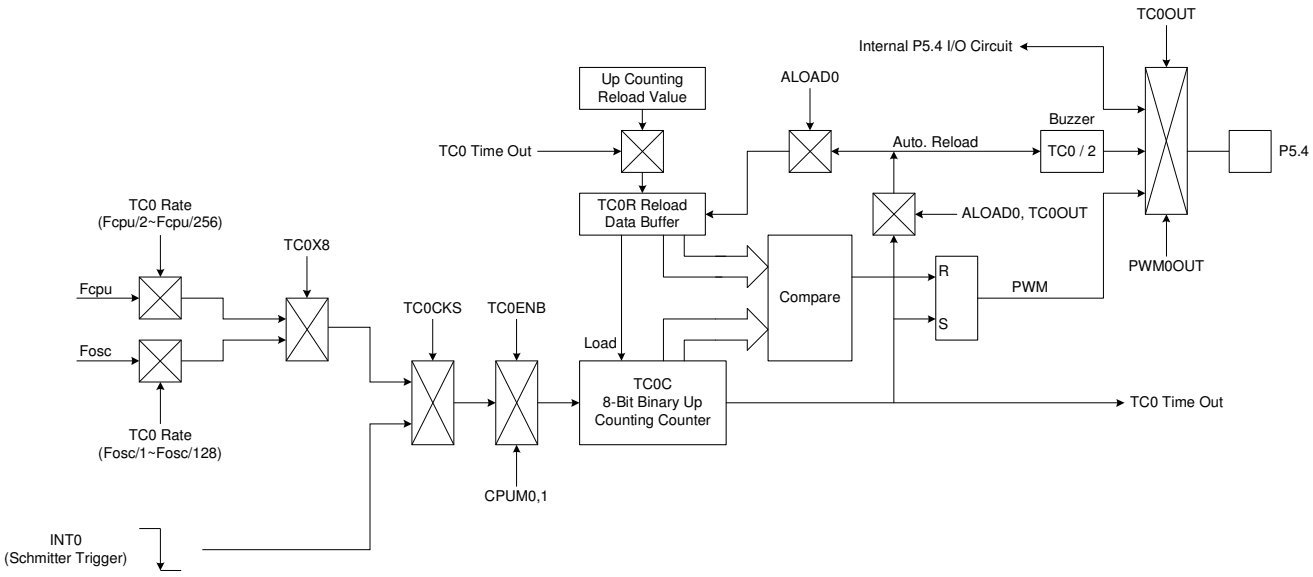
8.3 TIMER/COUNTER 0 (TC0)

8.3.1 OVERVIEW

The TC0 is an 8-bit binary up counting timer with double buffers. TC0 has two clock sources including internal clock and external clock for counting a precision time. The internal clock source is from Fcpu or Fosc controlled by TC0X8 flag to get faster clock source (Fosc). The external clock is INT0 from P0.0 pin (Falling edge trigger). Using TC0M register selects TC0C's clock source from internal or external. If TC0 timer occurs an overflow, it will continue counting and issue a time-out signal to trigger TC0 interrupt to request interrupt service. TC0 overflow time is 0xFF to 0X00 normally. Under PWM mode, TC0 overflow is decided by PWM cycle controlled by ALOAD0 and TC0OUT bits.

The main purposes of the TC0 timer is as following.

- **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **External event counter:** Counts system "events" based on falling edge detection of external clock signals at the INT0 input pin.
- **Green mode wake-up function:** TC0 can be green mode wake-up timer. System will be wake-up by TC0 time out.
- **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 or Fosc).
1 = External clock from P0.0/INT0 pin.

Bit [6:4] **TC0RATE[2:0]:** TC0 internal clock select bits.

TC0RATE [2:0]	TC0X8 = 0	TC0X8 = 1
000	Fcpu / 256	Fosc / 128
001	Fcpu / 128	Fosc / 64
010	Fcpu / 64	Fosc / 32
011	Fcpu / 32	Fosc / 16
100	Fcpu / 16	Fosc / 8
101	Fcpu / 8	Fosc / 4
110	Fcpu / 4	Fosc / 2
111	Fcpu / 2	Fosc / 1

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 TC1X8, TC0X8, TC0GN FLAGS

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	-	-	-	-	TC1X8	TC0X8	TC0GN	-
Read/Write	-	-	-	-	R/W	R/W	R/W	-
After reset	-	-	-	-	0	0	0	-

Bit 1 **TC0GN:** TC0 green mode wake-up function control bit.
0 = Disable TC0 green mode wake-up function.
1 = Enable TC0 green mode wake-up function.

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.

✸ ***Note: Under TC0 event counter mode (TC0CKS=1), TC0X8 bit and TC0RATE are useless.***
✸ ***Note: After TC0 green mode wake up,TC0 auto load register TC0R will not load to TC0C by hardware. Therefore,if use TC0 auto load function, after TC0 green mode wake up,user should load TC0R value to TC0C by software.***

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

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

TC0CKS	TC0X8	PWM0	ALOAD0	TC0OUT	N	TC0C valid value	TC0C value binary type	Remark
0	0 (Fcpu/2~Fcpu/256)	0	x	x	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	0	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b	Overflow per 64 count
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b	Overflow per 32 count
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b	Overflow per 16 count
	1 (Fosc/1~Fosc/128)	0	x	x	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	0	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b	Overflow per 64 count
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b	Overflow per 32 count
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b	Overflow per 16 count
1	-	-	-	-	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count

➢ **Example: To set 10ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=0, TC0X8=0) and no PWM output (PWM0=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC0RATE=010 (Fcpu/64).**

*TC0C initial value = N - (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, TC0X8 = 0.

TC0RATE	TC0CLOCK	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	7812.5us
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

The basic timer table interval time of TC0, TC0X8 = 1.

TC0RATE	TC0CLOCK	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	Fosc/128	8.192 ms	32 us	1000 ms	3906.25 us
001	Fosc/64	4.096 ms	16 us	500 ms	1953.12 us
010	Fosc/32	2.048 ms	8 us	250 ms	976.56 us
011	Fosc/16	1.024 ms	4 us	125 ms	488.28 us
100	Fosc/8	0.512 ms	2 us	62.5 ms	244.14 us
101	Fosc/4	0.256 ms	1 us	31.25 ms	122.07 us
110	Fosc/2	0.128 ms	0.5 us	15.625 ms	61.035 us
111	Fosc/1	0.064 ms	0.25 us	7.813 ms	30.52 us

8.3.5 TC0R AUTO-LOAD REGISTER

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

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

- ***Note: Under PWM mode, auto-load is enabled automatically. The ALOAD0 bit is selecting overflow boundary.***
- ***Note: After TC0 green mode wake up,TC0 auto load register TC0R will not load to TC0C by hardware. Therefore,if use TC0 auto load function, after TC0 green mode wake up,user should load TC0R value to TC0C by software.***

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

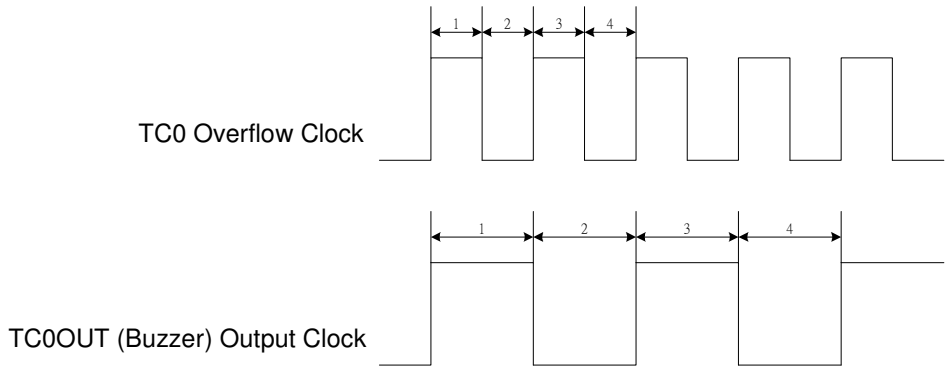
TC0CKS	TC0X8	PWM0	ALOAD0	TC0OUT	N	TC0R valid value	TC0R value binary type
0	0 (Fcpu/2~ Fcpu/256)	0	x	x	256	0x00~0xFF	00000000b~11111111b
		1	0	0	256	0x00~0xFF	00000000b~11111111b
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
	1 (Fosc/1~ Fosc/128)	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, TC0X8=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

8.3.6 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. T0C rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 131.**

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

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

B0BSET    FTC0OUT         ; Enable TC0 output to P5.4 and disable P5.4 I/O function
B0BSET    FALOAD1         ; Enable TC0 auto-reload function
B0BSET    FTC0ENB         ; Enable TC0 timer
```

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

8.3.7 TC0 TIMER OPERATION SEQUENCE

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

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

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

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

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

☞ **Set TC0 timer clock source.**

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

; Select TC0 Fcpu / Fosc internal clock source .
            B0BCLR      FTC0X8      ; Select TC0 Fcpu internal clock source.
or
            B0BSET      FTC0X8      ; Select TC0 Fosc internal clock source.
```

✸ ***Note: TC0X8 is useless in TC0 external clock source mode.***

☞ **Set TC0 timer auto-load mode.**

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

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

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

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

☞ **Set TC0 timer function mode.**

```
        B0BSET      FTC0IEN     ; Enable TC0 interrupt function.
or
        B0BSET      FTC0OUT     ; Enable TC0OUT (Buzzer) function.
or
        B0BSET      FPWM0OUT    ; Enable PWM function.
or
        B0BSET      FTC0GN      ; Enable TC0 green mode wake-up function.
```

☞ **Enable TC0 timer.**

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

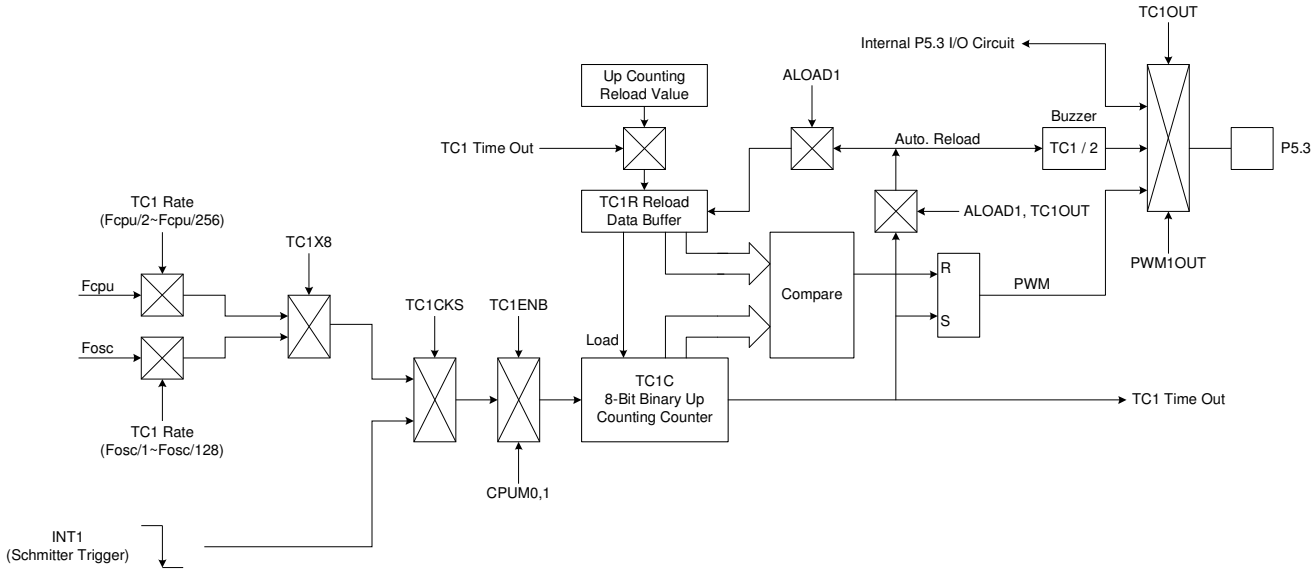
8.4 TIMER/COUNTER 1 (TC1)

8.4.1 OVERVIEW

The TC1 is an 8-bit binary up counting timer with double buffers. TC1 has two clock sources including internal clock and external clock for counting a precision time. The internal clock source is from Fcpu or Fosc controlled by TC1X8 flag to get faster clock source (Fosc). 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 decided by PWM cycle controlled by ALOAD1 and TC1OUT bits.

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 or Fosc).
1 = External clock from P0.1/INT1 pin.

Bit [6:4] **TC1RATE[2:0]:** TC1 internal clock select bits.

TC1RATE [2:0]	TC1X8 = 0	TC1X8 = 1
000	Fcpu / 256	Fosc / 128
001	Fcpu / 128	Fosc / 64
010	Fcpu / 64	Fosc / 32
011	Fcpu / 32	Fosc / 16
100	Fcpu / 16	Fosc / 8
101	Fcpu / 8	Fosc / 4
110	Fcpu / 4	Fosc / 2
111	Fcpu / 2	Fosc / 1

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 TC1X8 FLAG

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	-	-	-	-	TC1X8	-	-	-
Read/Write	-	-	-	-	R/W	-	-	-
After reset	-	-	-	-	0	-	-	-

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.

✸ ***Note: Under TC1 event counter mode (TC1CKS=1), TC1X8 bit and TC1RATE are useless.***

8.4.4 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 = N - (TC1 interrupt interval time * input clock)

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

TC1CKS	TC1X8	PWM1	ALOAD1	TC1OUT	N	TC1C valid value	TC1C value binary type	Remark
0	0 (Fcpu/2~ Fcpu/256)	0	x	x	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	0	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b	Overflow per 64 count
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b	Overflow per 32 count
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b	Overflow per 16 count
	1 (Fosc/1~ Fosc/128)	0	x	x	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	0	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b	Overflow per 64 count
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b	Overflow per 32 count
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b	Overflow per 16 count
1	-	-	-	-	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count

➢ **Example: To set 10ms interval time for TC1 interrupt. TC1 clock source is Fcpu (TC1KS=0, TC1X8=0) and no PWM output (PWM1=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC1RATE=010 (Fcpu/64).**

*TC1C initial value = N - (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, TC1X8 = 0.

TC1RATE	TC1CLOCK	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

The basic timer table interval time of TC1, TC1X8 = 1.

TC1RATE	TC1CLOCK	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	Fosc/128	8.192 ms	32 us	1000 ms	3906.25 us
001	Fosc/64	4.096 ms	16 us	500 ms	1953.12 us
010	Fosc/32	2.048 ms	8 us	250 ms	976.56 us
011	Fosc/16	1.024 ms	4 us	125 ms	488.28 us
100	Fosc/8	0.512 ms	2 us	62.5 ms	244.14 us
101	Fosc/4	0.256 ms	1 us	31.25 ms	122.07 us
110	Fosc/2	0.128 ms	0.5 us	15.625 ms	61.035 us
111	Fosc/1	0.064 ms	0.25 us	7.813 ms	30.52 us

8.4.5 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.

TC1 is double buffer design. If new TC1R value is set by program, the new value is stored in 1st buffer. Until TC1 overflow occurs, the new value moves to real TC1R buffer. This way can avoid TC1 interval time error and glitch in PWM and Buzzer output.

✷ ***Note: Under PWM mode, auto-load is enabled automatically. The ALOAD1 bit is selecting overflow boundary.***

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 six types (TC1 timer, TC1 event counter, TC1 Fcpu clock source, TC1 Fosc clock source, PWM mode and no PWM mode). These parameters decide TC1 overflow time and valid value as follow table.

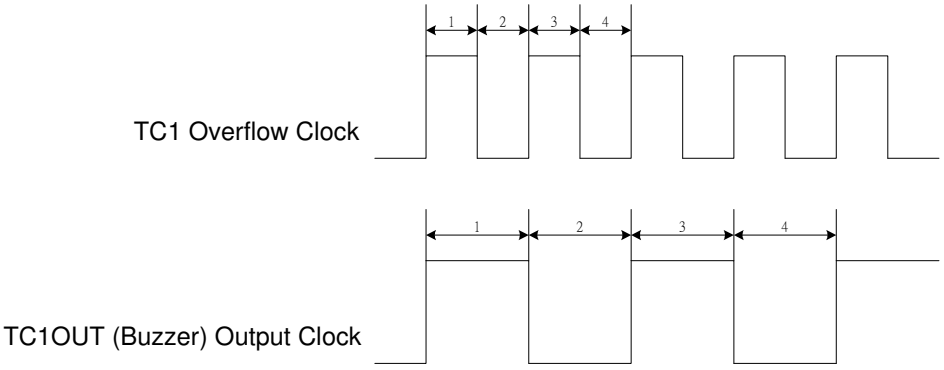
TC1CKS	TC1X8	PWM1	ALOAD1	TC1OUT	N	TC1R valid value	TC1R value binary type
0	0 (Fcpu/2~ Fcpu/256)	0	x	x	256	0x00~0xFF	00000000b~11111111b
		1	0	0	256	0x00~0xFF	00000000b~11111111b
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
	1 (Fosc/1~ Fosc/128)	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, TC1X8=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

8.4.6 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 enable, and "PWM1OUT" must be "0".***

8.4.7 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.

; Select TC1 Fcpu / Fosc internal clock source .
            B0BCLR      FTC1X8      ; Select TC1 Fcpu internal clock source.
or
            B0BSET      FTC1X8      ; Select TC1 Fosc internal clock source.
```

✸ ***Note: TC1X8 is useless in TC1 external clock source mode.***

☞ **Set TC1 timer auto-load mode.**

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

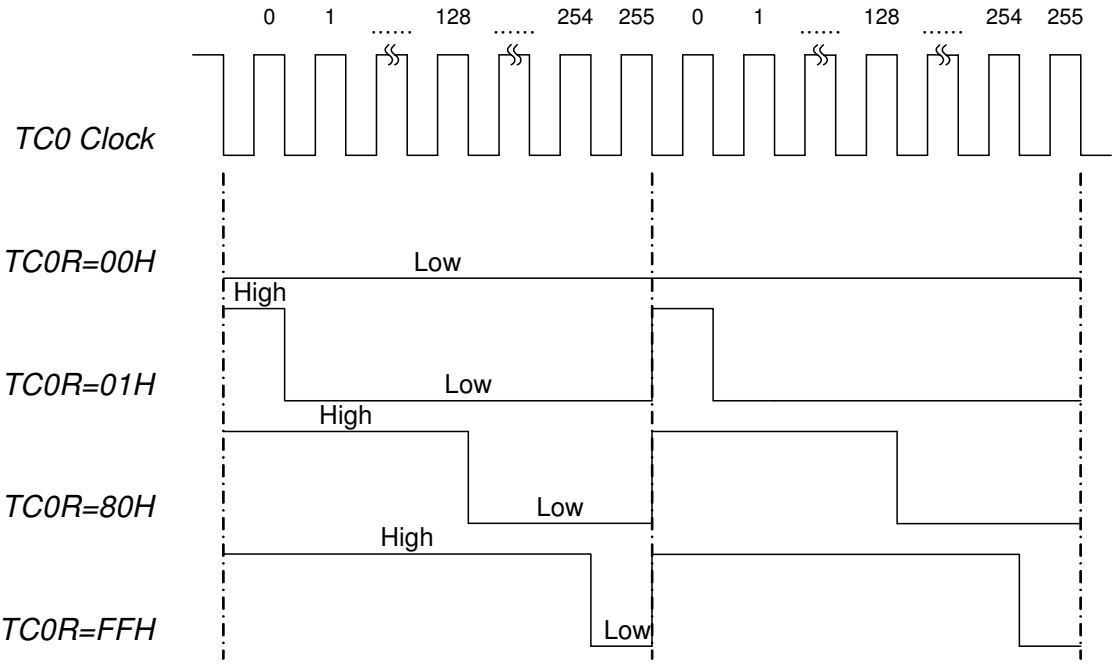
8.5 PWM0 MODE

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 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 ratio (duty) of the PWM0 output is TC0R/256.

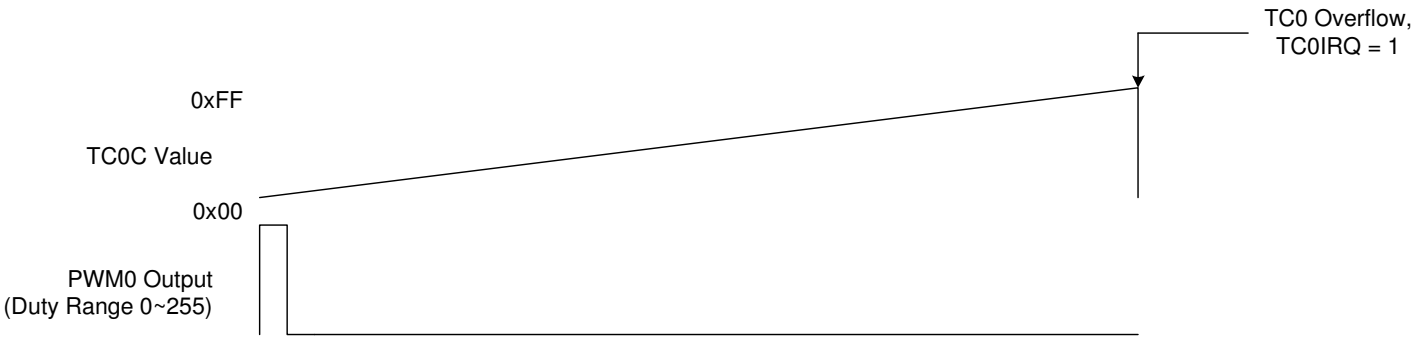
PWM duty range	TC0C valid value	TC0R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count

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

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.***

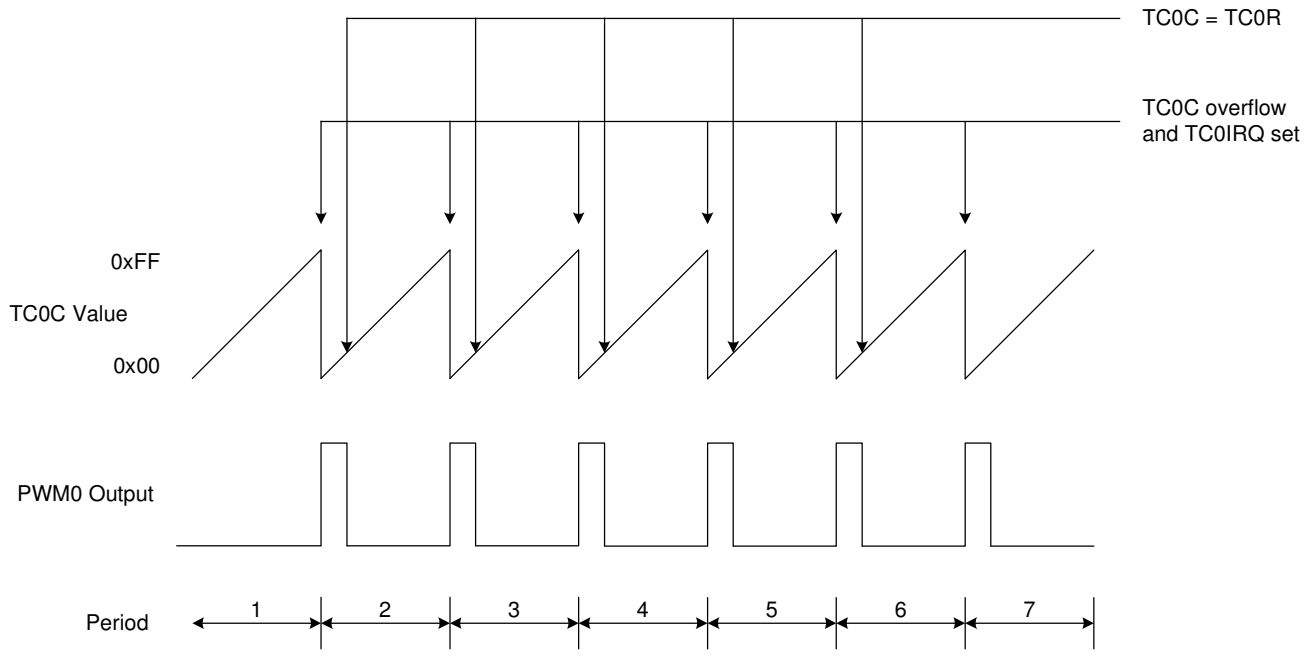
- **Example: Modify TC0R registers' value.**

```
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.***

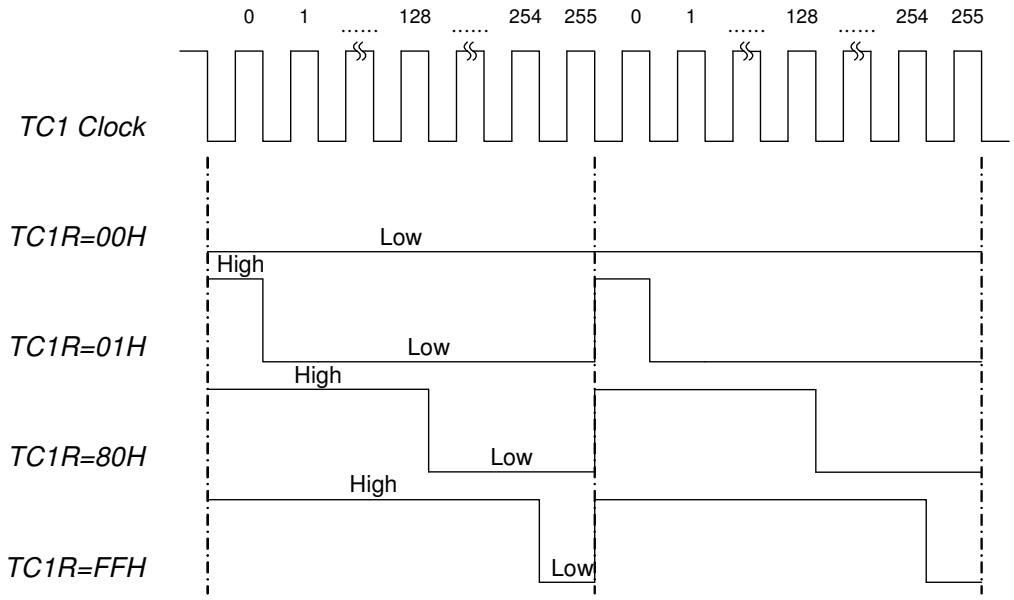
8.6 PWM1 MODE

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 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 ratio (duty) of the PWM1 output is TC1R/256,

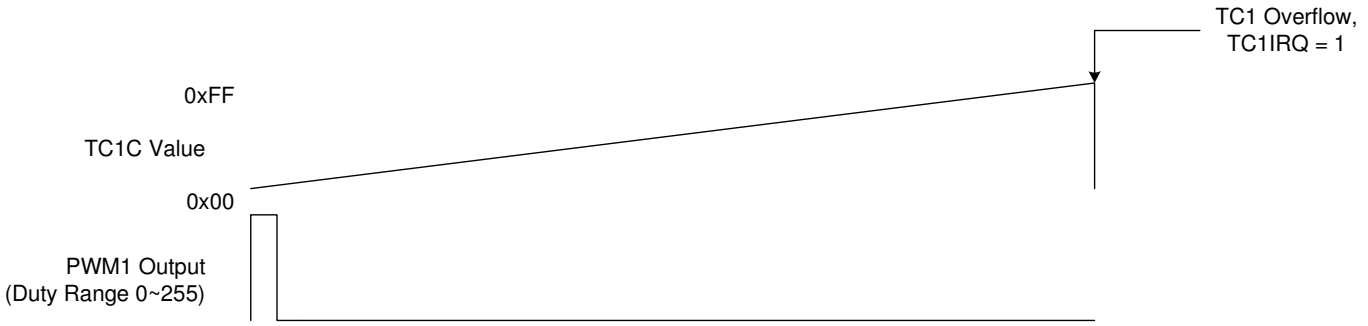
PWM duty range	TC1C valid value	TC1R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count

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

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.***

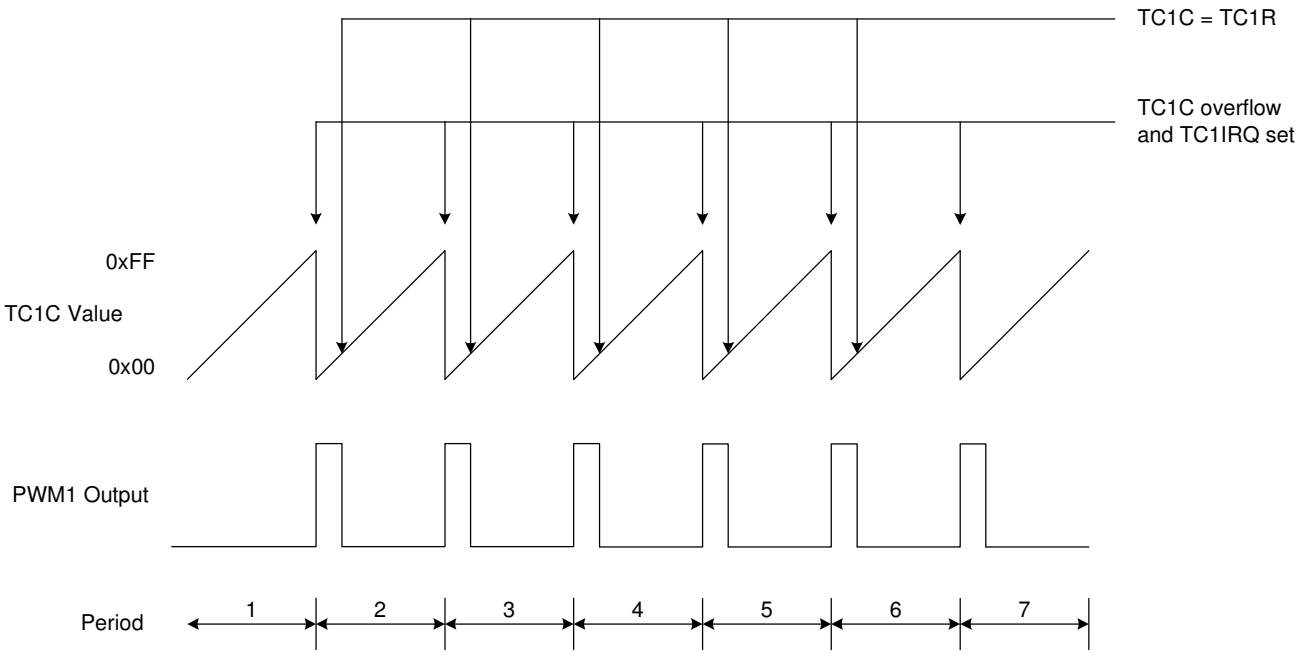
- **Example: Modify TC1R registers' value.**

```
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

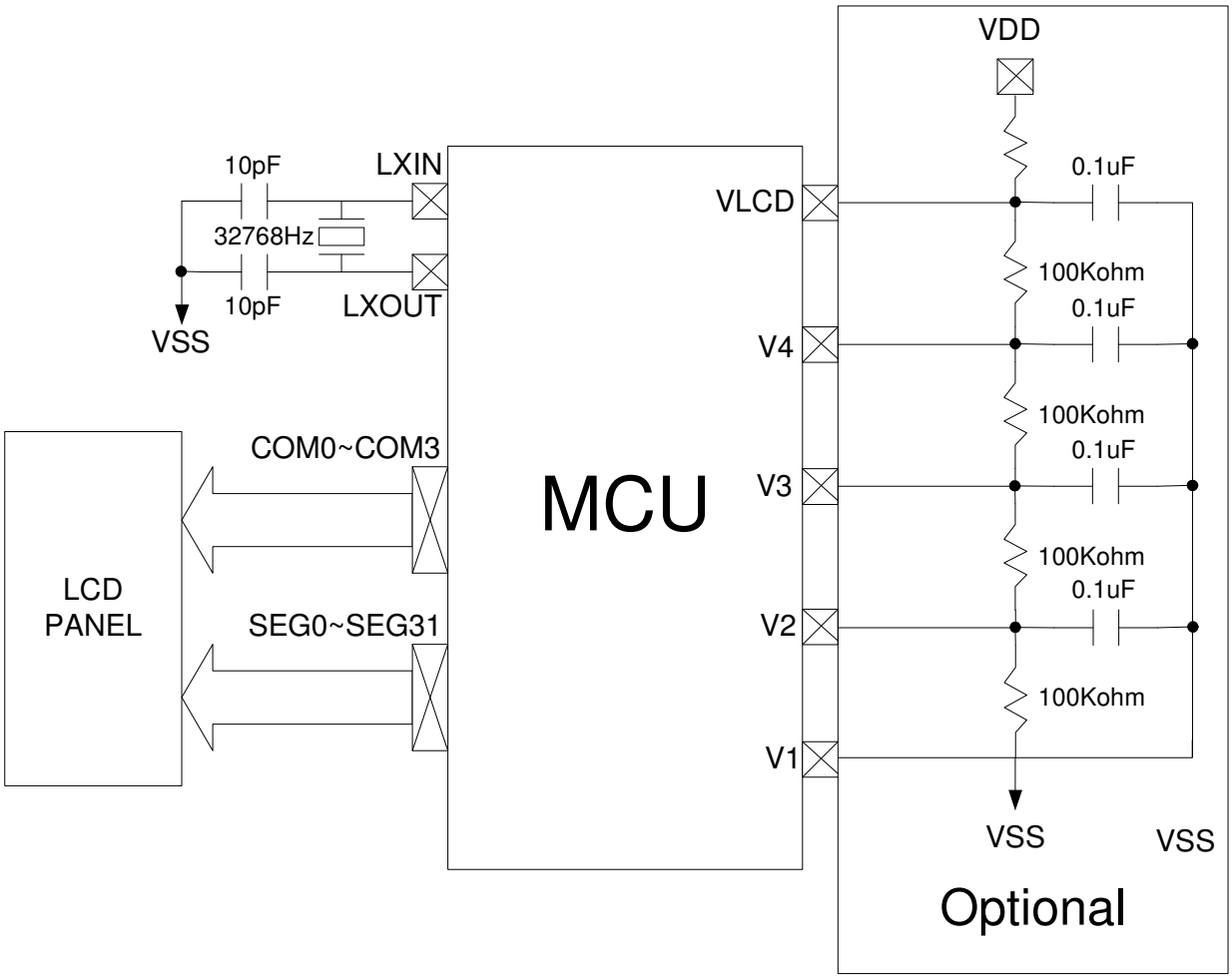
In PWM mode, the system will compare TC1C and TC1R all the time. When TC1C<TC1R, the PWM will output logic "High", when TC1C≧TC1R, the PWM will output logic "Low". If TC1C is changed in certain period, the PWM duty will change immediately. If TC1R is fixed all the time, the PWM waveform is also the same.



9 4x32 LCD DRIVER

9.1 OVERVIEW

The MCU builds in two LCD driving type, one is 4x32 (4 commons and 32 segments, 128 dots) LCD driver which supports 1/4 duty, 1/3 bias LCD panel. Another is 8x28 (8 commons and 28 segments, 224 dots) LCD driver which supports 1/8 duty, 1/4 bias LCD panel. The LCD frame rate is 64Hz (32768Hz/512 = 64Hz) and clock source is external 32768Hz oscillator crystal or RC type. User can add resistance between VLCD/V2/V1 for more driving current.



Basic LCD Circuit

9.2 LCD REGISTERS

0CBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM	-	-	-	COMSEL	RCLK	P2SEG	BIAS	LCDENB
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

Bit 4 **COMSEL: .**LCD COM Selection bit
1 = LCD as 4 COM * 32 SEG setting
0 = LCD as 8 COM * 28 SEG setting

Bit 3 **RCLK:** External low speed clock type selection bit.
0 = 32768Hz crystal/ceramic type connected to LXIN, LXOUT pins.
1 = RC type.

Bit 2 **P2SEG:** Seg24~Seg31 shared with P2.0~P2.7 control bit.
0 = Enable Seg24~Seg 31 pins.
1 = Enable P2.0~P2.7 pins.

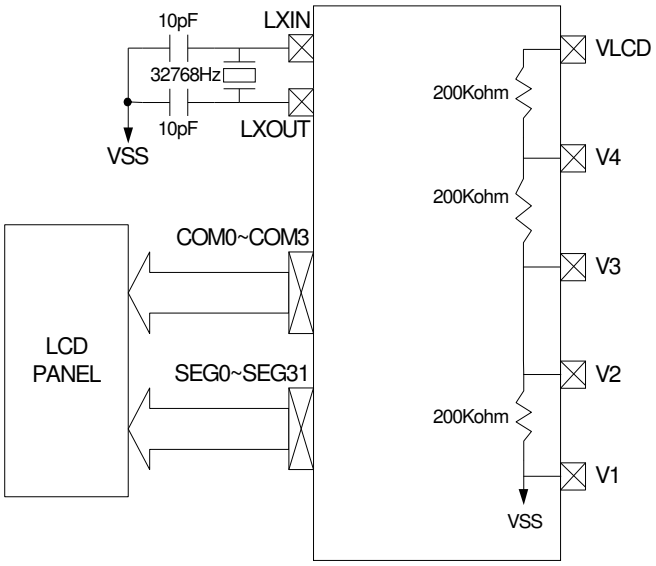
Bit 1 **BIAS:** LCD bias control bit.
0 = 1/4 bias
1 = 1/3 bias.

Bit 0 **LCDENB:** LCD control bit.
0 = Disable.
1 = Enable.

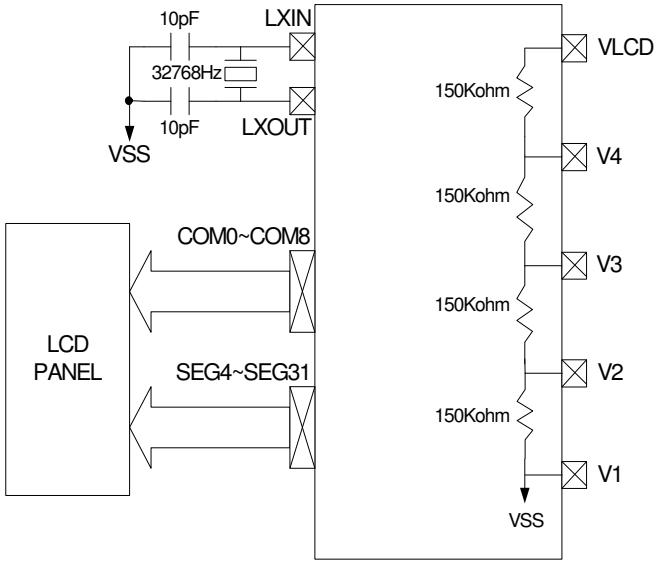
✸ ***Note: Segment 24~Segment 31 pins are shared with P2.0~P2.7. When these pins are used as Port2 general purpose I/O mode, the P3SEG bit of LCDM register must be set as "1".***

9.3 LCD Setting

In 4*32 LCD mode, COM0~COM3 and SEG0~SEG31 are available, and in 8*24 LCD mode, COM0~COM7 and SEG4~SEG31 are available. User can select the LCD dot by different LCD panel.
In LCD Bias Setting, the MCU provide 1/3 bias and 1/4 bias setting. Both of these setting have internal resistance circuit, in general application, V1/V2/V3 can only connect a 0.1 uF capacitor without any resistance.

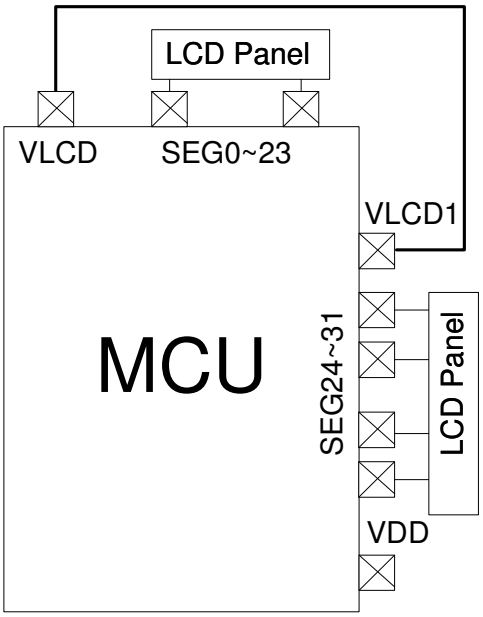


1/3 bias, 1/4 duty, LCD Circuit.

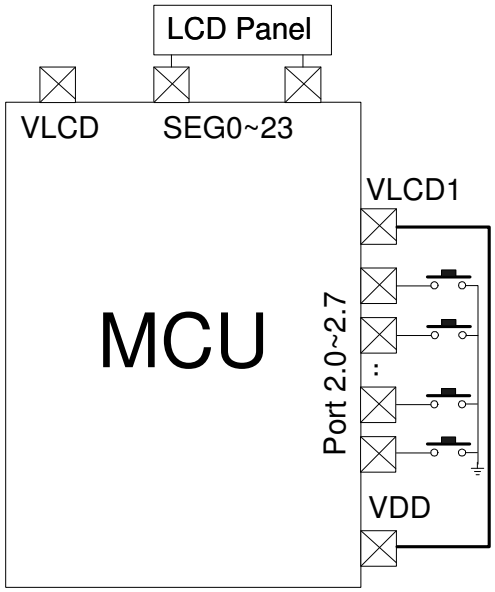


1/4 bias, 1/8 duty, LCD Circuit.

Register P2SEG can select SEG24~SEG31 be as LCD segment or as Port 2 for IO application. When set P2SEG=0, these pins are for LCD application, connect VLCD1 to VLCD. When set P2SEG=1, these pins are for IO port application, connect VLCD1 to VDD.



P2SEG=0



P2SEG=1

✸ ***Note:***

3. 1/3 bias, V3 connects to V2.

4. The 0.1uF capacitors of VLCD/V1/V2/V3/V4 pins are necessary for power stable.

➢ **Example : Set LCD mode.**

```
                                        ; R-type mode.
; Set LCD 32768Hz clock source type.
        B0BCLR      FRCLK       ; Crystal/ceramic type.
or
        B0BSET      FRCLK       ; RC type.
; Set LCD Bias.
        B0BCLR      FBIAS       ; 1/3 bias.
or
        B0BSET      FBIAS       ; 1/4 bias.
; Set LCD COM.
        B0BCLR      FCOMSEL     ; Set 8 COM * 28 SEG
or
        B0BSET      FCOMSEL     ; Set 4 COM * 32 SEG
; Set LCD P2SEG.
        B0BCLR      FP2SEG      ; Enable Seg24~Seg 31 pins
or
        B0BSET      FP2SEG      ;. Enable P2.0~P2.7 pins
; Enable LCD.
        B0BSET      FLCDENB     ; Enable LCD driver..
```

9.4 LCD RAM MAP

LCD dots are controlled by LCD RAM in Bank15. Program the LCD RAM data is using index pointer in bank 0 or directly addressing in bank 15. The LCD RAM placement is by LCD segments. One segment address includes four common bits data. COM0~COM7 of one LCD RAM (bit0~bit7). The LCD RAM map is as following.

RAM bank 15 address vs. Common/Segment location.

RAM	Bit	0	1	2	3	4	5	6	7
Address	LCD	COM0	COM1	COM2	COM3	COM4	COM5	COM	COM
00h	SEG0	00h.0	00h.1	00h.2	00h.3	.	.	.	.
01h	SEG1	01h.0	01h.1	01h.2	01h.3	.	.	.	.
02h	SEG2	02h.0	02h.1	02h.2	02h.3	.	.	.	.
03h	SEG3	03h.0	03h.1	03h.2	03h.3	.	.	.	.
04h	SEG4	04h.0	04h.1	04h.2	04h.3	04h.4	04h.5	04h.6	04h.7
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
0Ch	SEG12	0Ch.0	0Ch.1	0Ch.2	0Ch.3	0Ch.4	0Ch.5	0Ch.6	0Ch.7
0Dh	SEG13	0Dh.0	0Dh.1	0Dh.2	0Dh.3	0Dh.4	0Dh.5	0Dh.6	0Dh.7
0Eh	SEG14	0Eh.0	0Eh.1	0Eh.2	0Eh.3	0Eh.4	0Eh.5	0Eh.6	0Eh.7
0Fh	SEG15	0Fh.0	0Fh.1	0Fh.2	0Fh.3	0Fh.4	0Fh.5	0Fh.6	0Fh.7
10h	SEG16	10h.0	10h.1	10h.2	10h.3	10h.4	10h.5	10h.6	10h.7
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
1Bh	SEG27	1Bh.0	1Bh.1	1Bh.2	1Bh.3	1Bh.4	1Bh.5	1Bh.6	1Bh.7
1Ch	SEG28	1Ch.0	1Ch.1	1Ch.2	1Ch.3	1Ch.4	1Ch.5	1Ch.6	1Ch.7
1Dh	SEG29	1Dh.0	1Dh.1	1Dh.2	1Dh.3	1Dh.4	1Dh.5	1Dh.6	1Dh.7
1Eh	SEG30	1Eh.0	1Eh.1	1Eh.2	1Eh.3	1Eh.4	1Eh.5	1Eh.6	1Eh.7
1Fh	SEG31	1Fh.0	1Fh.1	1Fh.2	1Fh.3	1Fh.4	1Fh.5	1Fh6.	1Fh.7

➢ **Example : Set LCD RAM data by index pointer (@YZ) in bank 0.**

```
B0MOV   Y, #0FH         ; Set @YZ point to LCD RAM address 0x1500.
CLR     Z

MOV     A, #00001010B   ; Set COM0=0, COM1=1, OCM2=0,COM3=1 of SEG 0.
B0MOV   @YZ, A

INCMS   Z               ; Point to next segment address.
…
…
```

➢ **Example : Set LCD RAM data by directly addressing in bank 15.**

```
MOV     A, #15              ; Switch to RAM bank 15.
B0MOV   RBANK, A

MOV     A, #00001010B       ; Set COM0=0, COM1=1, OCM2=0,COM3=1 of SEG 0.
MOV     00H, A

BCLR    01H.0               ; Clear COM0=0 of SEG 1.
BSET    01H.1               ; Clear COM1=1 of SEG 1.
…
…

MOV     A, #0               ; Switch to RAM bank 0.
B0MOV   RBANK, A
```

✸ ***Note: Access RAM data of bank 0 (system registers and user define RAM 0x0000~0x007F) is using "B0xxx" instructions.***

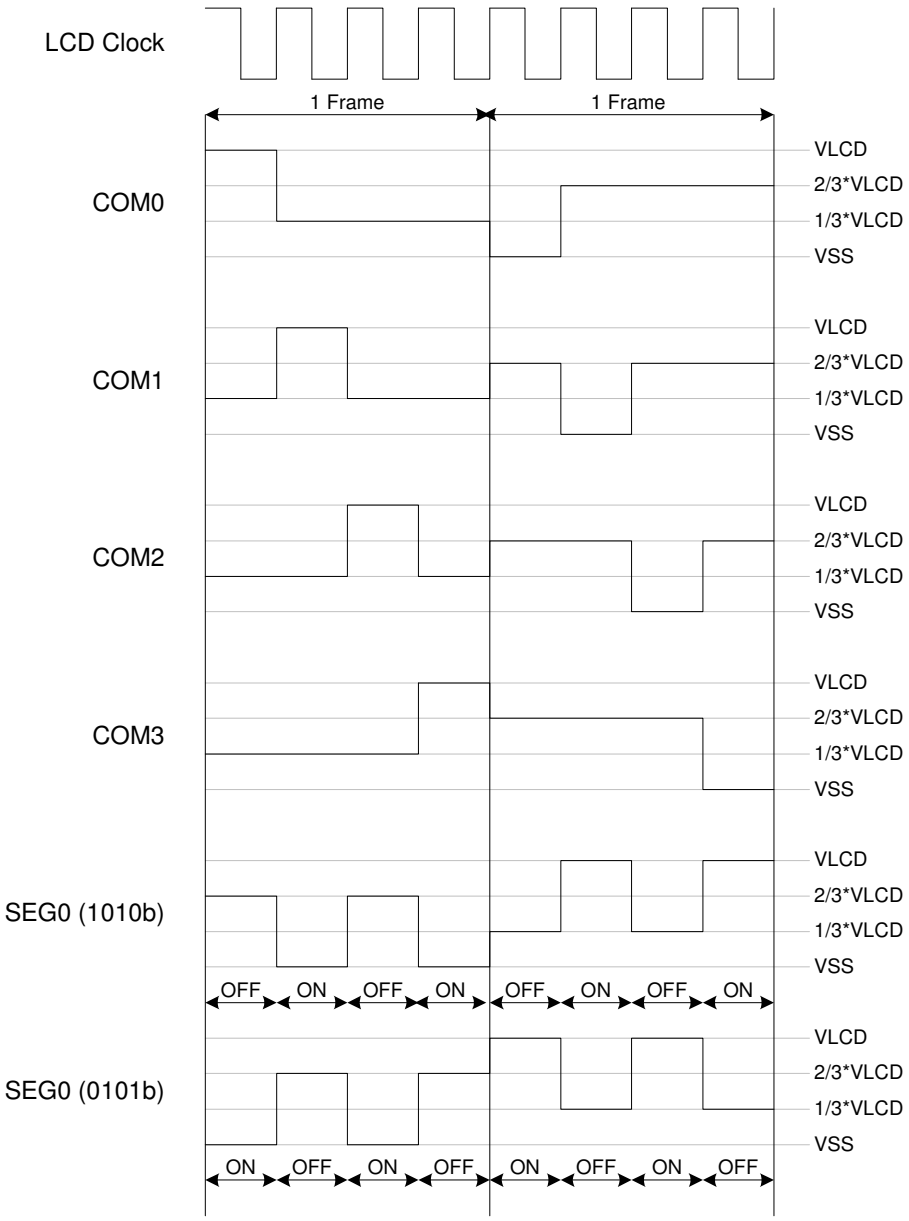
9.5 LCD TIMING AND WAVEFORM

F-frame = External Low clock / 512

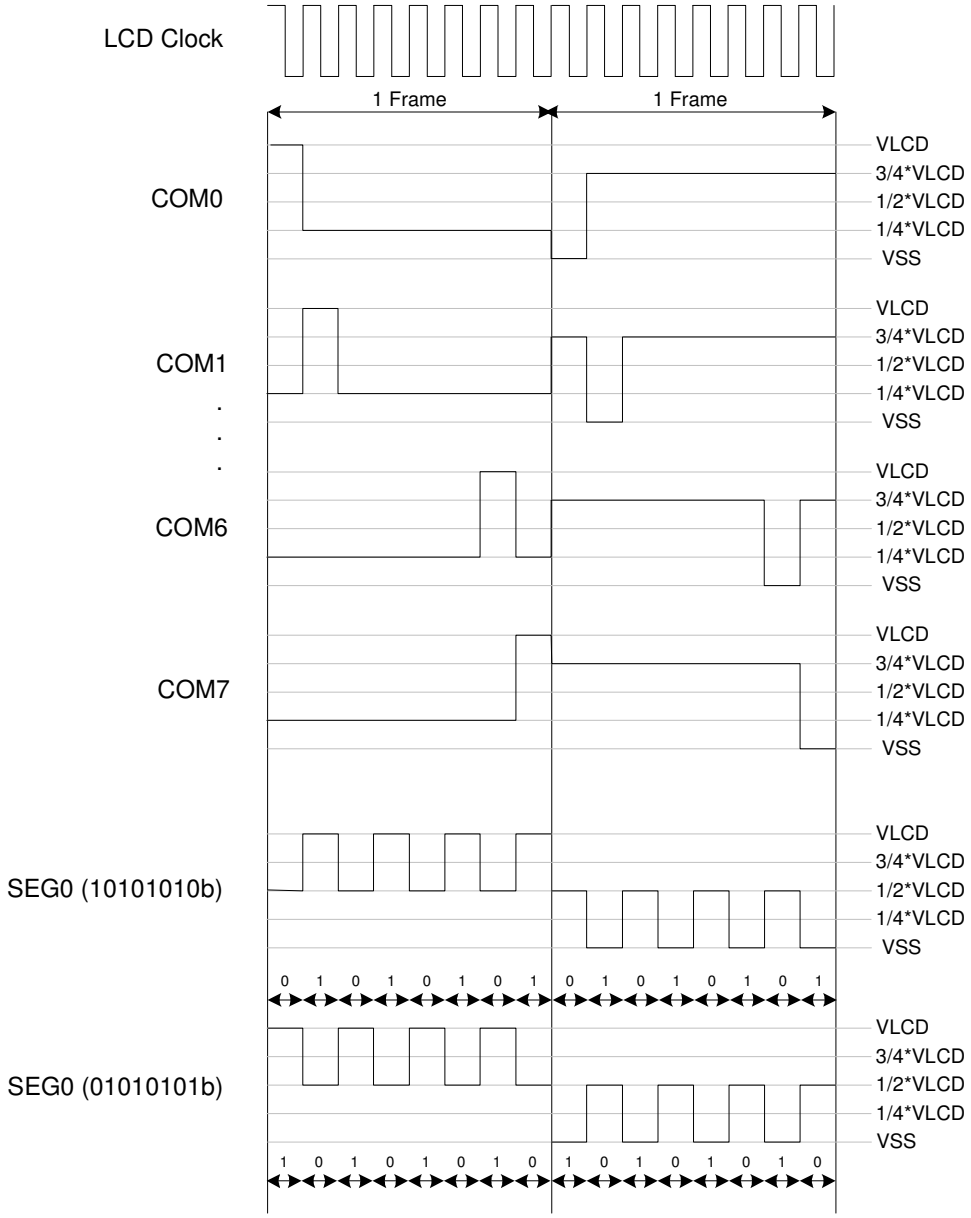
Ex. External low clock is 32768Hz. The F-frame is 32768Hz/512 = **64Hz**.

Note: The clock source of LCD driver is external low clock.

1/3 bias, 1/4 duty.



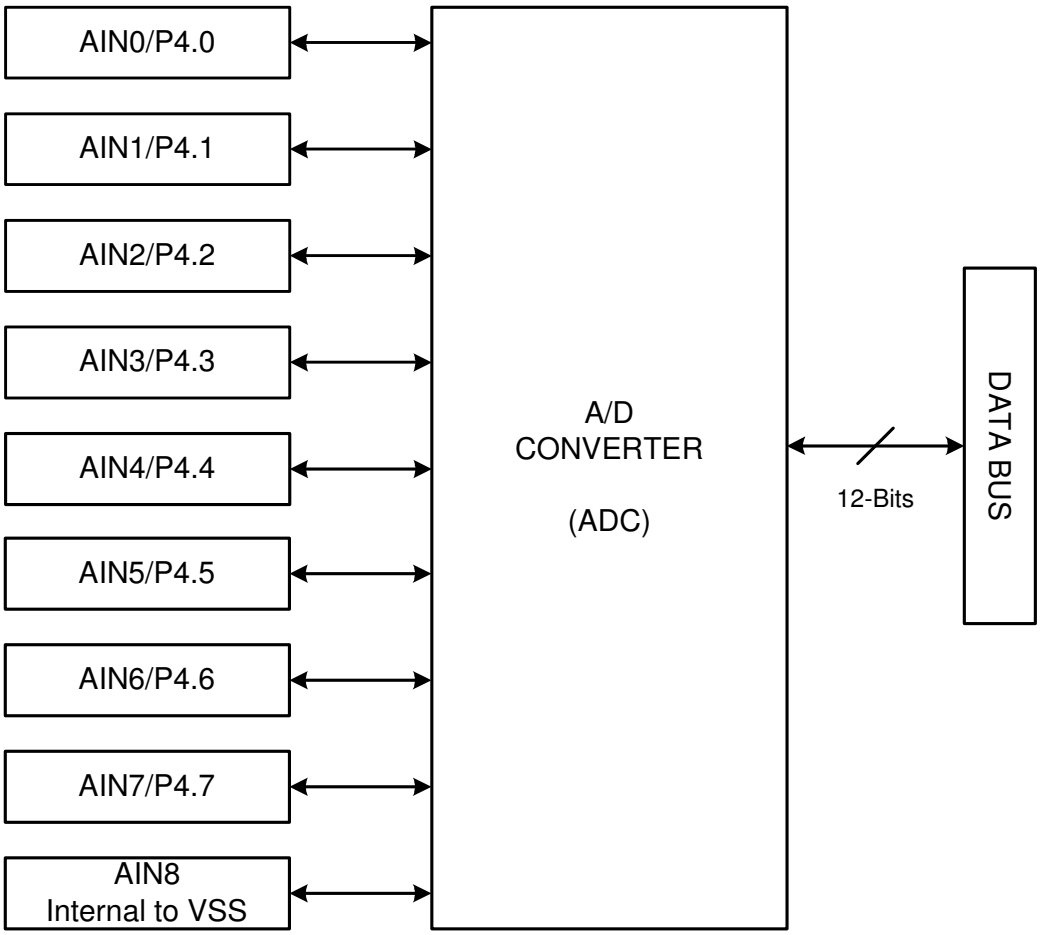
1/4 bias, 1/8 duty.



10 8+1 CHANNEL ANALOG TO DIGITAL CONVERTER

10.1 OVERVIEW

This analog to digital converter has 8 external sources (AIN0~AIN7) and one internal channel to VSS for ADC offset check, with 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 12-bits value output in ADB and ADR low-nibble registers.



10.2 ADM REGISTER

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **ADENB:** ADC control bit.
0 = Disable.
1 = Enable.

Bit 6 **ADS:** ADC start 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[3:0]:** ADC input channels select bit.
0000 = AIN0, 0001 = AIN1, 0010 = AIN2, 0011 = AIN3
0100 = AIN4 ,0101 = AIN5, 0110 = AIN6, 0111 = AIN7
1000 = To AIN8 Internal VSS for ADC offset Test

The AIN8 is internal VSS input channel. There is no any input pin from outside. AIN8 can be a ADC offset check channel without any external circuit.

✸ ***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.***

10.3 ADR REGISTERS

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

Bit[6,4] **ADCKS1, ADCKS0:** ADC clock source selection.

ADCKS1	ADCKS0	ADC Clock Source
0	0	Fcpu/16
0	1	Fcpu/8
1	0	Fcpu
1	1	Fcpu/2

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

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

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

10.4 ADB REGISTERS

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB15	ADB14	ADB13	ADB12	ADB11	ADB10	ADB9	ADB8
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.***

10.5 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[4:0]:** 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.***

10.6 VREFH REGISTERS

The MCU have external pin for ADC reference voltage (AVREFH). User can input ADC reference into this pin or use the internal reference voltage. When used the internal reference voltage, AVREFH will be the voltage of VHS[1:0] set, eg, VDD, 4V, 3V, 2V.

✸ ***Note:***
a. ADC resolution is 12-bit if use Internal VDD reference voltage or use external reference voltage.
b. No matter in which REFH setting, connect capacitor 0.1uF to AVREFH.
c. AVREFH pin will out VHS[1:0] setting voltage eg, VDD, 4V, 3V, 2V when REFS= GCHS=1

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VREFH	REFS	-	-	-	-	-	VHS1	VHS0
Read/Write	R/W	-	-	-	-	-	R/W	R/W
After reset	0	-	-	-	-	-	0	0

Bit[1:0] **VHS[1:0]:** ADC internal reference high voltage select bits.

VHS1	VHS0	Internal VREFH Voltage
1	1	VDD
1	0	4.0V
0	1	3.0V
0	0	2.0V

✸ ***Note: If internal VREFH level selected by VHS[1:0] is higher than VDD, the internal VREFH is VDD. For instance, VHS[1:0] is 10 (internal VREFH = 4.0V) and VDD is 3.0V, the actual internal VREFH is equal to VDD (3.0V).***

Bit[7] **REFS :** ADC internal reference high voltage control bit.
1 = Enable ADC internal VREFH function, AVREFH pin will out VHS[1:0] setting voltage
0 = Disable ADC internal VREFH function, Connect AVREFH to external voltage.

10.7 ADC CONVERTING TIME

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

High Clock (Fosc) = 4MHz

Fcpu		ADCKS1	ADCKS0	ADC Clock	ADC Converting Time											
Fosc/	1	0	0	Fcpu/16	1/(	(	4MHz	/	1	)	/	16	/4	)	x16=	256	us
		0	1	Fcpu/8	1/(	(	4MHz	/	1	)	/	8	/4	)	x16=	128	us
		1	0	Fcpu	1/(	(	4MHz	/	1	)	/	1	/4	)	x16=	16	us
		1	1	Fcpu/2	1/(	(	4MHz	/	1	)	/	2	/4	)	x16=	32	us
Fosc/	2	0	0	Fcpu/16	1/(	(	4MHz	/	2	)	/	16	/4	)	x16=	512	us
		0	1	Fcpu/8	1/(	(	4MHz	/	2	)	/	8	/4	)	x16=	256	us
		1	0	Fcpu	1/(	(	4MHz	/	2	)	/	1	/4	)	x16=	32	us
		1	1	Fcpu/2	1/(	(	4MHz	/	2	)	/	2	/4	)	x16=	64	us
Fosc/	4	0	0	Fcpu/16	1/(	(	4MHz	/	4	)	/	16	/4	)	x16=	1024	us
		0	1	Fcpu/8	1/(	(	4MHz	/	4	)	/	8	/4	)	x16=	512	us
		1	0	Fcpu	1/(	(	4MHz	/	4	)	/	1	/4	)	x16=	64	us
		1	1	Fcpu/2	1/(	(	4MHz	/	4	)	/	2	/4	)	x16=	128	us
Fosc/	8	0	0	Fcpu/16	1/(	(	4MHz	/	8	)	/	16	/4	)	x16=	2048	us
		0	1	Fcpu/8	1/(	(	4MHz	/	8	)	/	8	/4	)	x16=	1024	us
		1	0	Fcpu	1/(	(	4MHz	/	8	)	/	1	/4	)	x16=	128	us
		1	1	Fcpu/2	1/(	(	4MHz	/	8	)	/	2	/4	)	x16=	256	us

10.8 ADC ROUTINE EXAMPLE

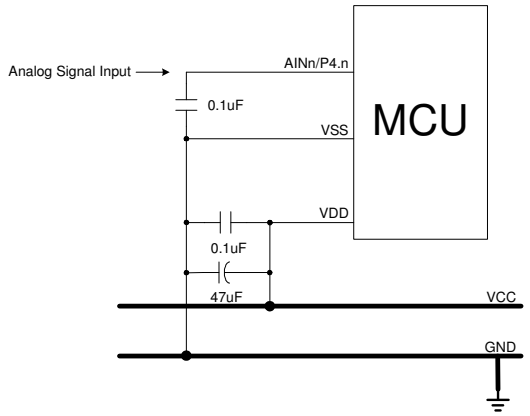
➢ **Example : To set AIN0 for ADC input and executing 12-bit ADC. VREFH is internal 3.0V. ADC clock source is Fcpu.**

```
; Enable ADC function and delay 100us for conversion.
ADC0:
                B0BSET      FADENB          ; Enable ADC circuit
                CALL        Delay100uS      ; Delay 100uS to wait ADC circuit ready for conversion.

; Set Port 4 I/O mode.
                MOV         A, #0FEH
                B0MOV       P4UR, A         ; Disable P4.0 pull-up resistor.
                B0BCLR      FP40M           ; Set P4.0 as input pin.
; or
                MOV         A, #01H
                B0MOV       P4CON, A        ; Set P4.0 as pure analog input.

; Set VREFH is internal 3.0V.
                MOV         A, #081H
                B0MOV       VREFH, A        ; Set internal 3.0V VREFH.

; Set ADC clock source = Fcpu.
                MOV         A, #40H
                B0MOV       ADR, A          ; To set ADC clock = Fcpu.

; Enable AIN0 (P4.0).
                MOV         A, #90H
                B0MOV       ADM, A          ; To enable ADC and set AIN0 input

; Start AD conversion.
                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
End_ADC:                    .
                B0BCLR      FADENB          ; Disable ADC circuit
```

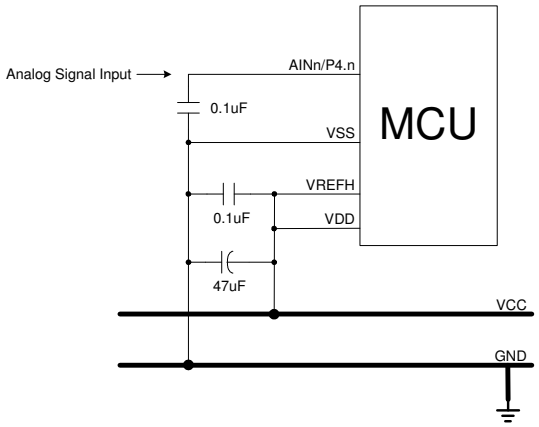
➢ **Example : To set AIN1 for ADC input and executing 12-bit ADC. VREFH is external input voltage from AVREFH pin . ADC clock source is Fcpu. Using ADC interrupt.**

```
; Enable ADC function and delay 100us for conversion.
ADC0:
                B0BSET      FADENB          ; Enable ADC circuit
                CALL        Delay100uS      ; Delay 100uS to wait ADC circuit ready for conversion.

; Set Port 4 I/O mode.
                MOV         A, #0FDH
                B0MOV       P4UR, A         ; Disable P4.1 pull-up resistor.
                B0BCLR      FP41M           ; Set P4.1 as input pin.
; or
                MOV         A, #02H
                B0MOV       P4CON, A        ; Set P4.1 as pure analog input.

; Set VREFH is external input voltage.
                B0BCLR      FREFS           ; Enable external VREFH input.

; Set ADC clock source = Fcpu.
                MOV         A, #40H
                B0MOV       ADR, A          ; To set ADC clock = Fcpu.

; Enable AIN0 (P4.1).
                MOV         A, #91H
                B0MOV       ADM,A           ; To enable ADC and set AIN1 input

; Set ADC interrupt.
                B0BCLR      FADCIRQ         ; Clear ADC interrupt request flag.
                B0BSET      FADCIEN         ; Enable ADC interrupt function.
                B0BSET      FGIE            ; Enable Global interrupt function.

; Start AD conversion.
                B0BSET      FADS            ; To start conversion
                …
                …
                …

ADC_INT_SR:
                PUSH

                B0BTS1      FADCIRQ         ; Check ADC interrupt flag.
                JMP         ADC_INT_EXIT
                B0BCLR      FADCIRQ         ; Clear ADC interrupt request flag.
                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
                B0BCLR      FADENB          ; Disable ADC circuit

ADC_INT_EXIT:               .
                POP
                RETI
```

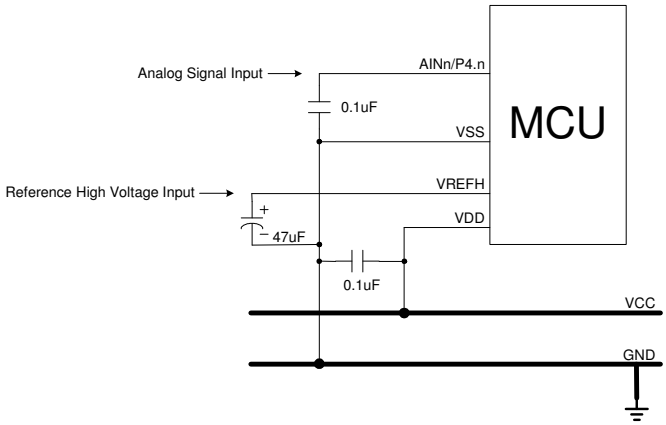
10.9 ADC CIRCUIT



ADC reference high voltage is internal reference voltage and VREFH pin is AIN0/P4.0. The capacitor (0.1uF) between AINn/P4.n and VSS is necessary to stable analog signal.



ADC reference high voltage is from VDD pin and AIN0/P4.0 is VERFH input. The VERFH should be from MCU's VDD pin. Don't connect from main power.



ADC reference high voltage is from external voltage and AIN0/P4.0 is VERFH input. The capacitor (47uF) between VREFH and VSS is necessary to stable VERFH voltage.

11 INSTRUCTION TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
	MOV	A,M	A ← M	-	-	√	1
M	MOV	M,A	M ← A	-	-	-	1
O	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
V	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
E	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
	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
A	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1+N
R	ADD	A,M	A ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
I	ADD	M,A	M ← A + M, if occur carry, then C=1, else C=0	√	√	√	1+N
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
H	ADD	A,I	A ← A + I, if occur carry, then C=1, else C=0	√	√	√	1
M	SBC	A,M	A ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
E	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
T	SUB	A,M	A ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
I	SUB	M,A	M ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1+N
C	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	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
	AND	A,M	A ← A and M	-	-	√	1
L	AND	M,A	M ← A and M	-	-	√	1+N
O	AND	A,I	A ← A and I	-	-	√	1
G	OR	A,M	A ← A or M	-	-	√	1
I	OR	M,A	M ← A or M	-	-	√	1+N
C	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
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
R	RRC	M	A ← RRC M	√	-	-	1
O	RRCM	M	M ← RRC M	√	-	-	1+N
C	RLC	M	A ← RLC M	√	-	-	1
E	RLCM	M	M ← RLC M	√	-	-	1+N
S	CLR	M	M ← 0	-	-	-	1
S	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
	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
B	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
R	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1+ S
A	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
N	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1+ S
C	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1+N+S
H	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
M	RET		PC ← Stack	-	-	-	2
I	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
S	PUSH		To push ACC and PFLAG (except NT0, NPD bit) into buffers.	-	-	-	1
C	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".

12 ELECTRICAL CHARACTERISTIC

12.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2808Q,SN8P2807Q,SN8P2807X,SN8P2807P ..... 0°C ~ + 70°C
SN8P2808QD,SN8P2807QD,SN8P2807XD,SN8P2807PD..... –40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

12.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, fosc = 4MHz, ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd		2.4	5.0	5.5	V
		Programming mode, Vpp = 12.5V			5.0		
RAM Data Retention voltage	Vdr			1.5	-	-	V
Internal POR	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	180	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
Port0, Port1, Port 4, Port 5 output source current	IoH	Vop = Vdd - 0.5V		9	-	-	mA
sink current	IoL	Vop = Vss + 0.5V		10	-	-	mA
INTn trigger pulse width	Tint0	INT0 ~ INT1 interrupt request pulse width		2/fcpu	-	-	Cycle
AVREFH input voltage	Varfh	Vdd = 5.0V		2V	-	Vdd	V
AIN0 ~ AIN7 input voltage	Vani	Vdd = 5.0V		0	-	Varfh	V
ADC Clock Frequency	F_{ADCLK}	VDD=5.0V		32K		8M	Hz
		VDD=3.0V		32K		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
ADC enable time	Tast	Ready to start convert after set ADENB = "1"		100	-	-	uS
ADC current consumption	I_{ADC}	Vdd=5.0V		-	0.6*	-	mA
		Vdd=3.0V		-	0.4*	-	mA
Supply Current (Disable ADC)	Idd1	normal Mode Fcpu = Fosc/4	Vdd= 5V 4MHz	-	2.5	5	mA
			Vdd= 3V 4MHz	-	1.5	3	mA
	Idd2	Slow Mode (External 32K,LCD OFF Stop high clock)	Vdd= 5V ~ 32768hz	-	20	30	uA
			Vdd= 3V ~ 32768Khz	-	8	20	uA
	Idd3	Slow Mode (External 32K,LCD ON Stop high clock)	Vdd= 5V ~ 32768hz	-	20	50	uA
			Vdd= 3V ~ 32768Khz	-	15	30	uA
	Idd4	Green Mode (External 32K,LCD OFF Stop high clock)	Vdd= 5V ~ 32768hz	-	10	20	uA
			Vdd= 3V ~ 32768Khz	-	5	10	uA
	Idd5	Green Mode (External 32K,LCD ON Stop high clock)	Vdd= 5V ~ 32768hz	-	20	40	uA
			Vdd= 3V ~ 32768Khz	-	8	20	uA
	Idd6	Sleep mode (LVD = LVD_L)	Vdd= 5V	-	1	2	uA
			Vdd= 3V	-	0.6	1	uA
LVD Voltage	Vdet0	Low voltage reset level		1.7	2.0	2.3	V
	Vdet1	Low voltage reset/indicator level Fcpu=1MHz		2.0	2.4	3	V
	Vdet2	Low voltage indicator level Fcpu=1MHz		2.7	3.6	4.5	V

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

13 DEVELOPMENT TOOL VERSION

13.1 ICE (In Circuit Emulation)

- **SN8ICE 2K ICE:** Full function emulates SN8P2808.

✸ ***SN8ICE 2K ICE emulation notice:***
a. Operation voltage of ICE: 3.0V ~ 5.0V.
b. Recommend maximum emulation speed at 5V: 8 MIPS (e.g. 16Mhz Crystal and Fcpu = Fhosc/2).
c. Use SN8P2808 EV-KIT to emulate LVD, LCD, Internal Reference Voltage.

✸ ***Note: S8KD-2 ICE doesn't support SN8P2808 emulation.***

13.2 OTP WRITER

- **Easy Writer V1.0:** OTP programming is controlled by ICE without firmware upgrade suffers. Please refer easy writer user manual for detailed information.
- **MP-Easy Writer V1.0:** Stand-alone operation to support SN8P2808 mass production.

13.3 IDE

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

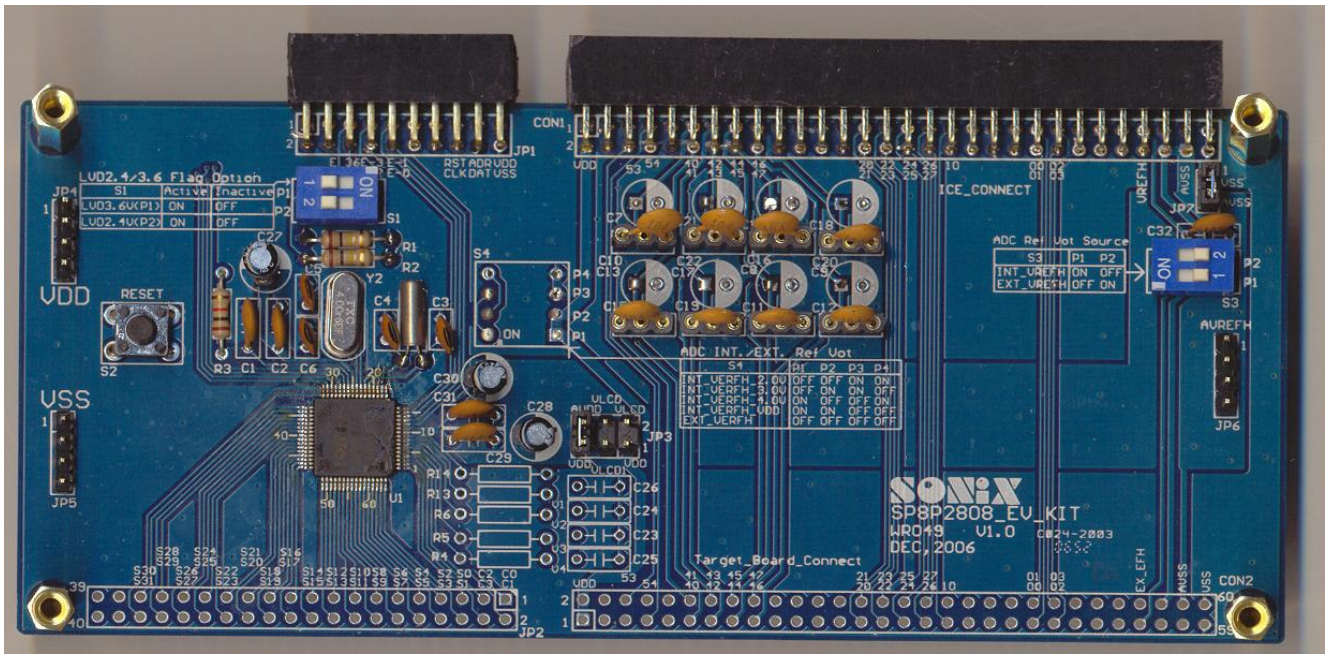
- M2IDE_V110 or later.
  - **Support SN8ICE_2K ICE**
  - **Support Assembly and ICE debug function**
  - **Support Easy Writer and MP-Easy Writer**

➢ ***Note: SN8IDE_1.99X don't support SN8P2808 emulation.***

13.4 SN8P2808 EV KIT

13.4.1 PCB DESCRIPTION

SONIX provides SN8P2808 EV Kit for function emulation. For SN8P2808 ICE emulation, the EV kit provides LCD, ADC internal reference voltage and LVD 2.4V/3.6V selection circuits.



CON1: I/O port. Connect to SN8ICE 2K CON1.

CON2: I/O port. Connect to target board.

JP1: LVD 2.4V, 3.6V input pins, LCD and Internal reference voltage control pins. Connect to SN8ICE 2K JP6.

JP2: LCD COM0~COM3, SEG0~SEG31 output pins. Connect to LCD panel.

JP3:. AVDD, VLCD, VLCD1, and VDD “SHORT” pin. About VLCD and VLCD1, please refer LCD section description.

JP6: Output internal reference voltage from 2808 IC(U1) on 2808 EV-kit board if ADENB and REFS are enable.

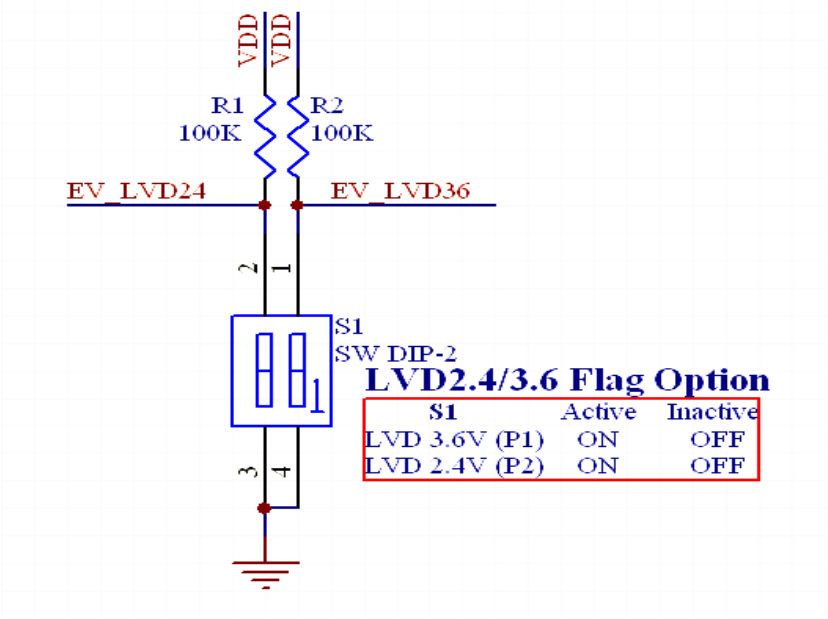
JP7: AVSS and VSS “SHORT” pin.

C1, R3: SN8P2808 EV chip reset circuit. C1=0,1uF, R3=10K ohm.

C7/C8/C9/C13/C14/C17/C18/C21, C10/C11/C12/C15/C16/C19/C20/C22: Connect these capacitors to P4 if P4 been as ADC input pin.

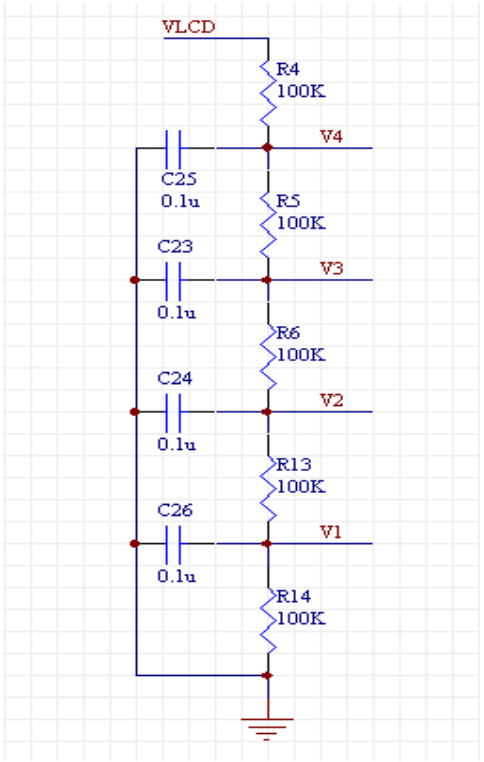
S1: LVD 2.4V/3.6V control switch. To emulate LVD 2.4V flag/reset function and LVD 3.6V flag function.

S1.	Active	Inactive
LVD 3.6V(P1)	ON	OFF
LVD 2.4V(P2)	ON	OFF



C23~C26, R4/R5/R6/R13/R14: LCD circuit devices. Please refer LCD section to adjust circuit for more driving current.

Note: If LCD segment 24 ~ segment 31 is disabled (P2SEG = "1")， set port 2 as I/O mode and VLCD1 connect to VDD.



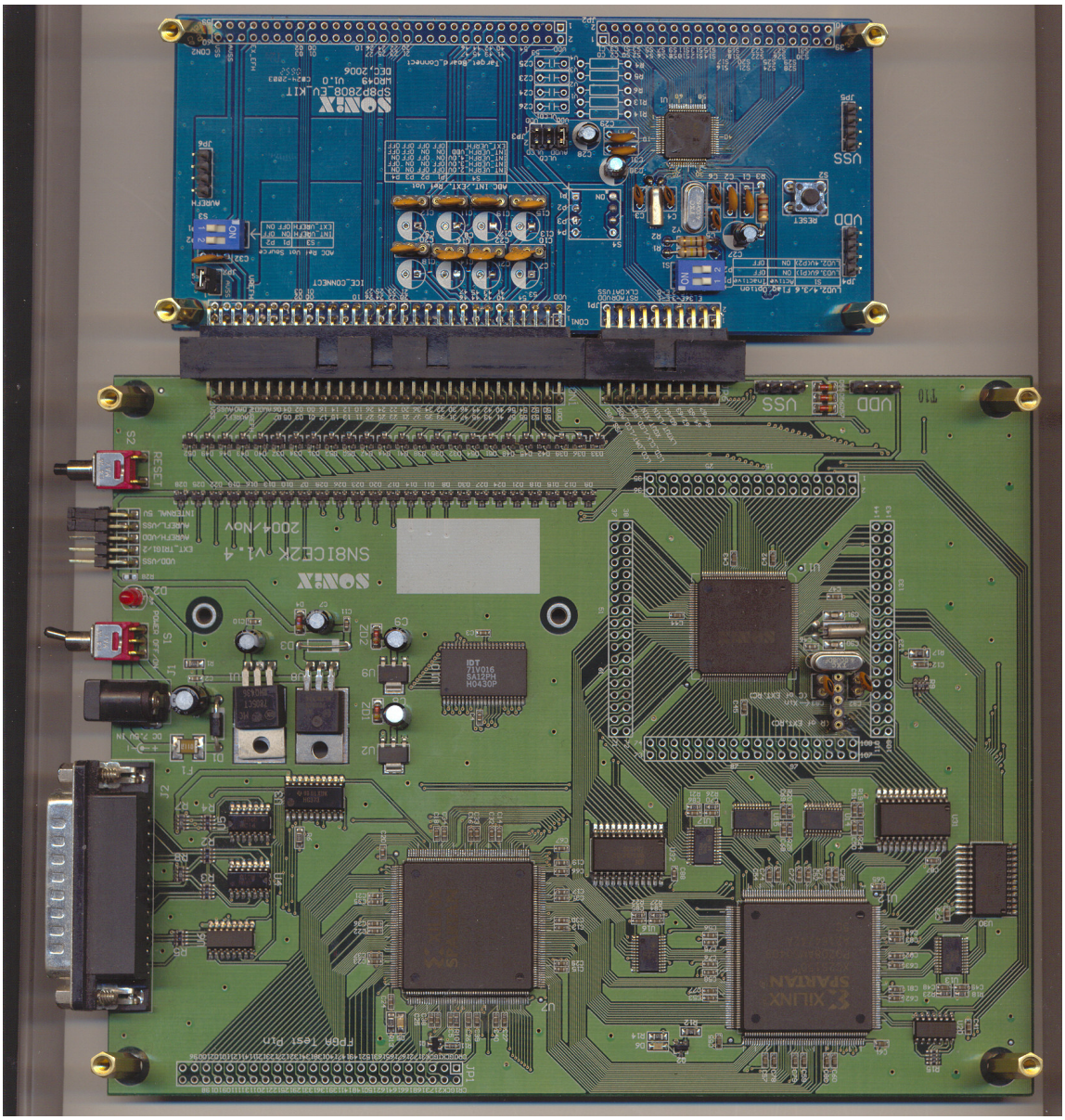
S3: Select the different reference voltage source for ADC Reference Voltage.

S3.	P1	P2
INT_VREFH	ON	OFF
EXT_VREFH	OFF	ON

As user enabled **ADENB** and **REFS** bits, user must select **INT_VREFH** as **ON** and **EXT_VREFH** as **OFF**. Besides, user have to open the "**AVREFH/VDD**" JUMP on the **SN8ICE2K** board. As user enabled **ADENB** bit and disabled **REFS** bit, user must select **EXT_VREFH** as **ON** and **INT_VREFH** as **OFF**. User can apply the desired voltage to **EX_EFH** pin on the **CON2**.

13.4.2 SN8P2808 EV KIT CONNECT TO SN8ICE2K

The connection from SN8P2808 EV KIT to SN8ICE 2K is as following.



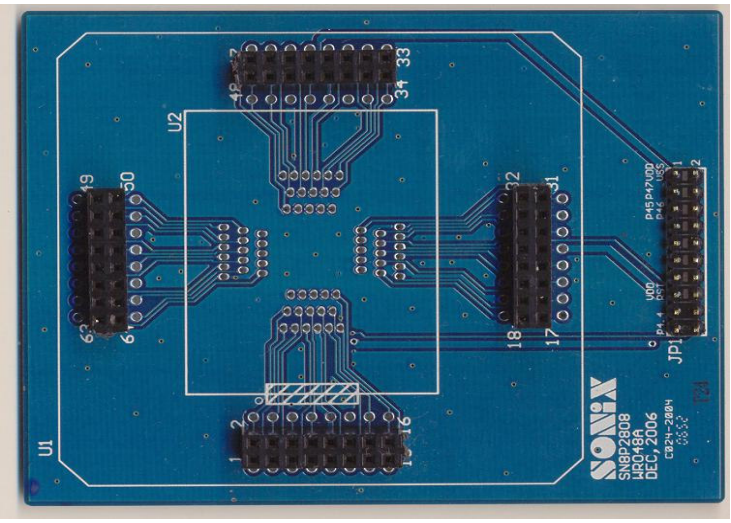
13.5 TRANSITION BOARD FOR OTP PROGRAMMING

13.5.1 SN8P2808 TRANSITION BOARD FOR LQFP 64 PIN

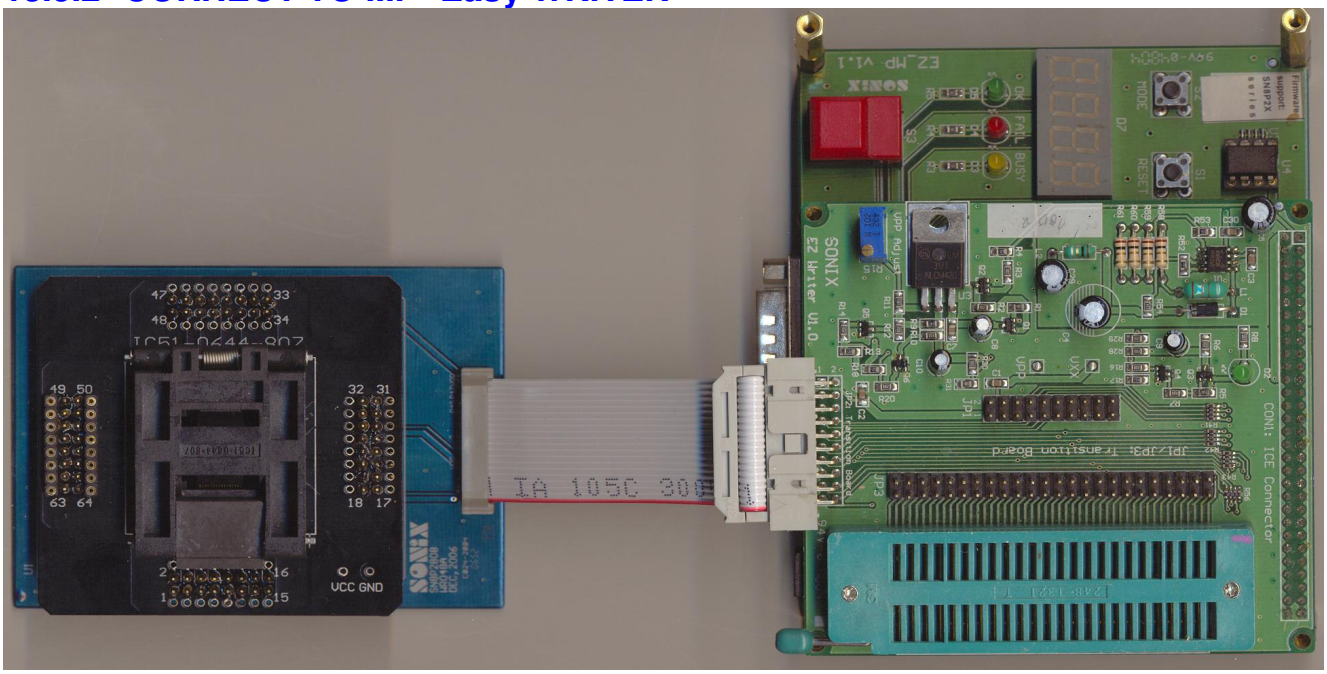
SN8P2808 transition board is for SN8P2808 OTP programming by LQFP 64 pin socket connection.

JP1: Connect to EZ writer or EZ_MP writer.

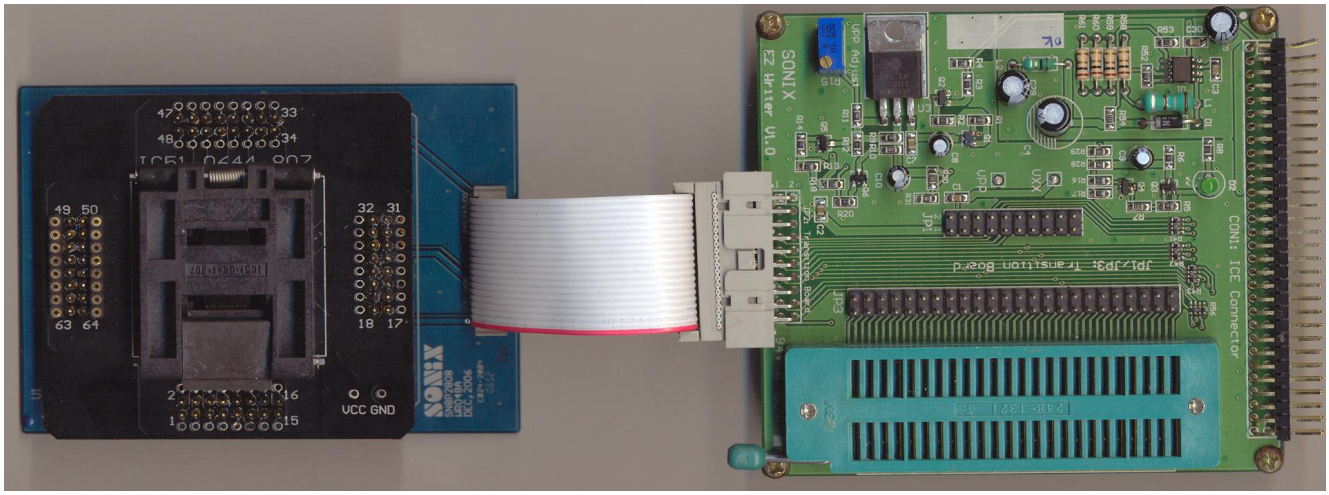
U1: LQFP 64 pin socket.



13.5.2 CONNECT TO MP- Easy WRITER



13.5.3 CONNECT TO Easy WRITER



➢ ***Note: Connect Easy Writer to ICE must be through an 60-pin cable***

13.6 13.6 OTP PROGRAMMING PIN

13.6.1 13.6.1 EASY WRITER TRANSITION BOARD SOCKET PIN ASSIGNMENT:

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

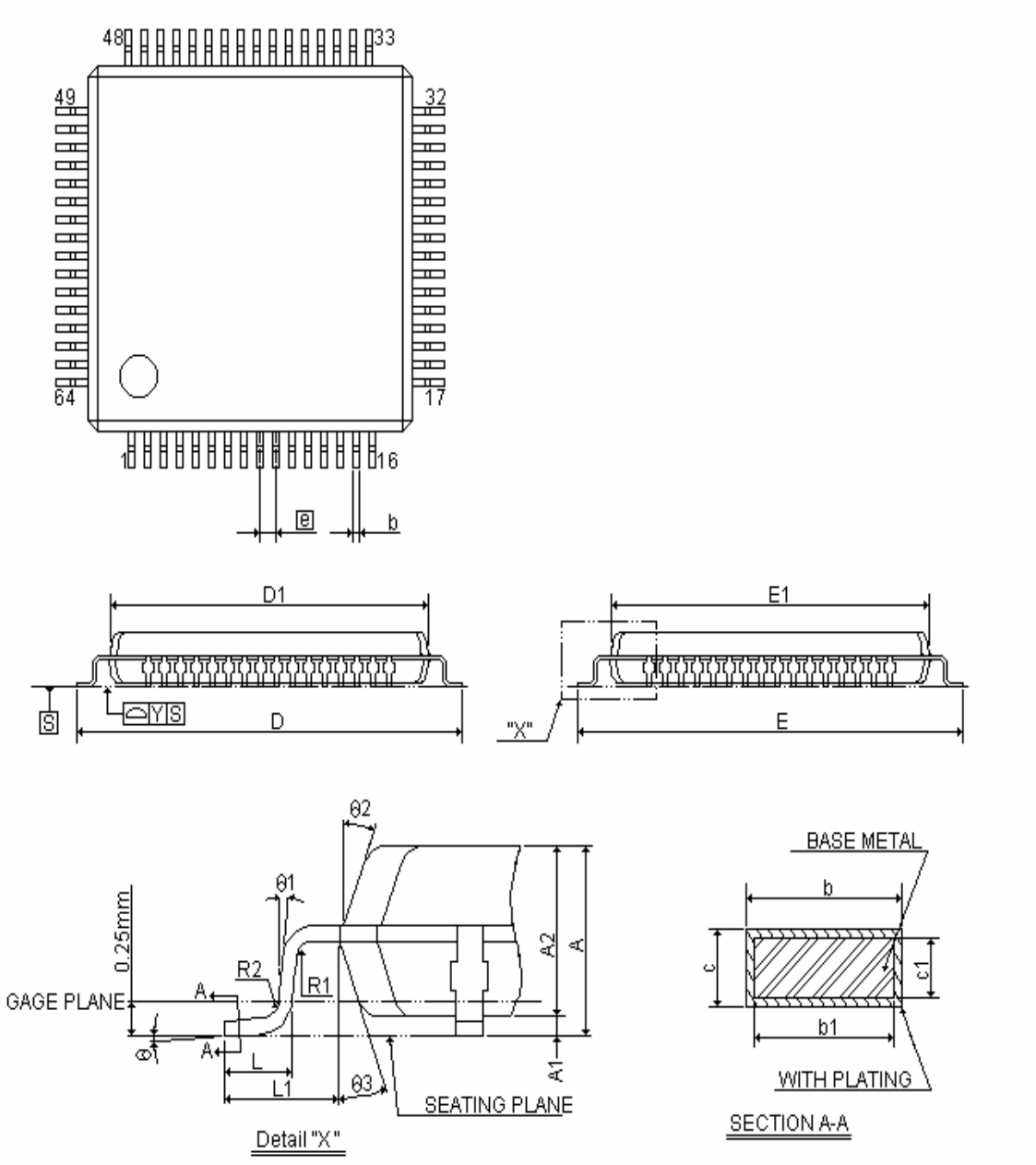
JP3 for MP transition board

13.6.2 13.6.2 SN8P2808 PROGRAMMING PIN MAPPING:

Programming Information of SN8P2800 Series									
Chip Name		SN8P2808Q		SN8P2807Q					
EZ Writer Connector		OTP IC / JP3 Pin Assigment							
Number	Name	Number	Pin	Number	Pin				
1	VDD	10,25,40	VDD	8,23,41	VDD				
2	GND	20	VSS	3	VSS				
3	CLK	19	P4.7	2	P4.7				
4	CE	-	-	-	-				
5	PGM	17	P4.5	48	P4.5				
6	OE	18	P4.6	1	P4.6				
7	D1	-	-	-	-				
8	D0	-	-	-	-				
9	D3	-	-	-	-				
10	D2	-	-	-	-				
11	D5	-	-	-	-				
12	D4	-	-	-	-				
13	D7	-	-	-	-				
14	D6	-	-	-	-				
15	VDD	-	-	-	-				
16	VPP	26	RST	9	RST				
17	HLS	-	-	-	-				
18	RST	-	-	-	-				
19	-	-	-	-	-				
20	ALSB/PDB	16	P4.4	47	P4.4				

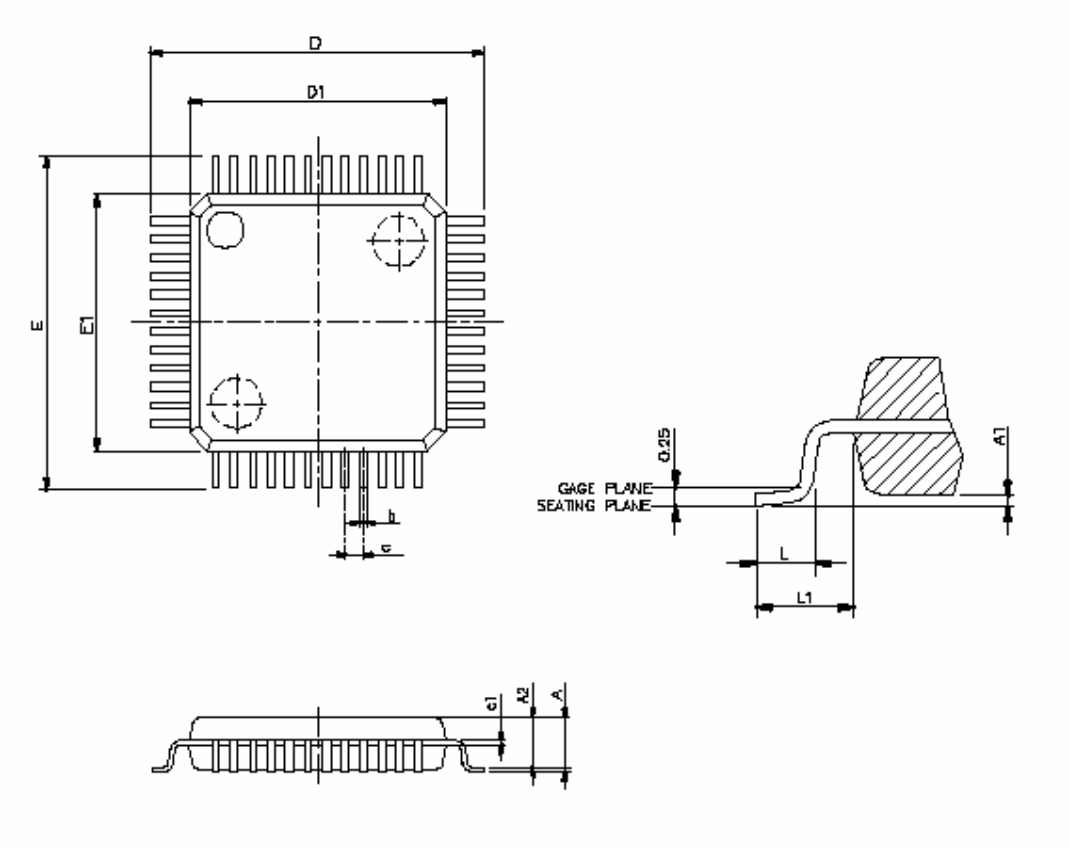
14 PACKAGE INFORMATION

14.1 LQFP 64 PIN



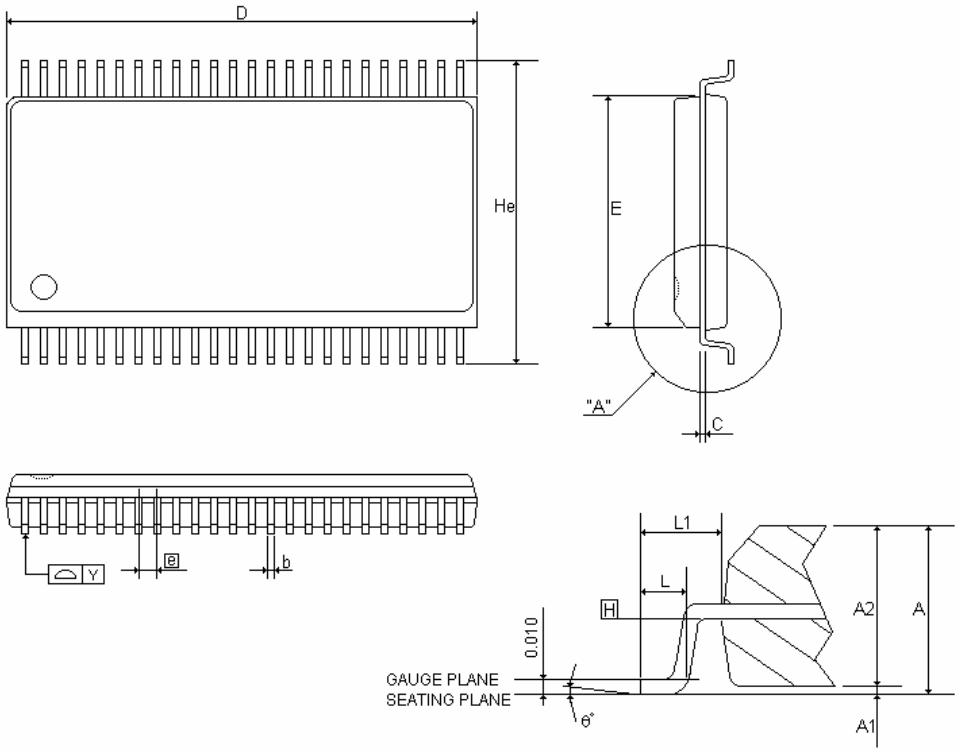
SYMBLE	DIMENSION (MM)			DIMENSION (MIL)		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A			1.60			63
A1	0.05	1.40	0.15	2	55	6
A2	1.36	0.22	1.45	35	9	57
b	0.17	0.22	0.27	7	8	11
b1	0.17		0.23	7		12
c	0.09		0.20	4		8
c1	0.09		0.16	4		6
D	11.75	12.00	12.25	463	473	483
D1	9.95	10.00	10.05	392	394	396
E	11.75	12.00	12.25	463	473	483
E1	9.95	10.00	10.05	392	394	396
[e]		0.50			20	
L	0.45	0.60	0.75	18	24	30
L1	0.9	1	1.1		39	
R1	0.08			3		
R2	0.08		0.20	3		8
Y			0.075			3
θ	0°	3.5°	7°	0°	3.5°	7°
θ 1	0°			0°		
θ 2	11°	12°	13°	11°	12°	13°
θ 3	11°	12°	13°	11°	12°	13°

14.2 LQFP 48 PIN



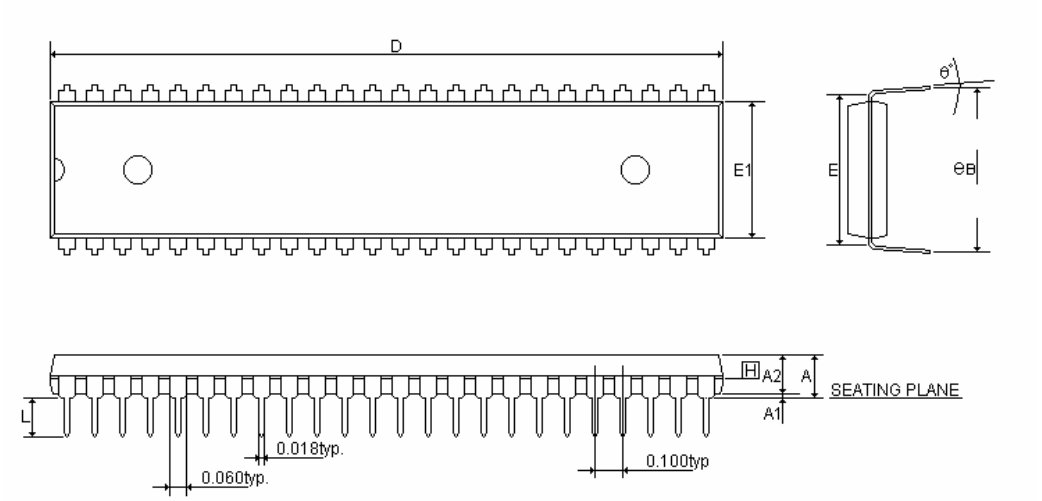
SYMBOLS	MIN	NOR	MAX
	(mm)		
A	-	-	1.6
A1	0.05	-	0.15
A2	1.35	-	1.45
c1	0.09	-	0.16
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
e	0.5 BSC		
B	0.17	-	0.27
L	0.45	-	0.75
L1	1 REF		

14.3 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°

14.4 P-DIP 48 PIN



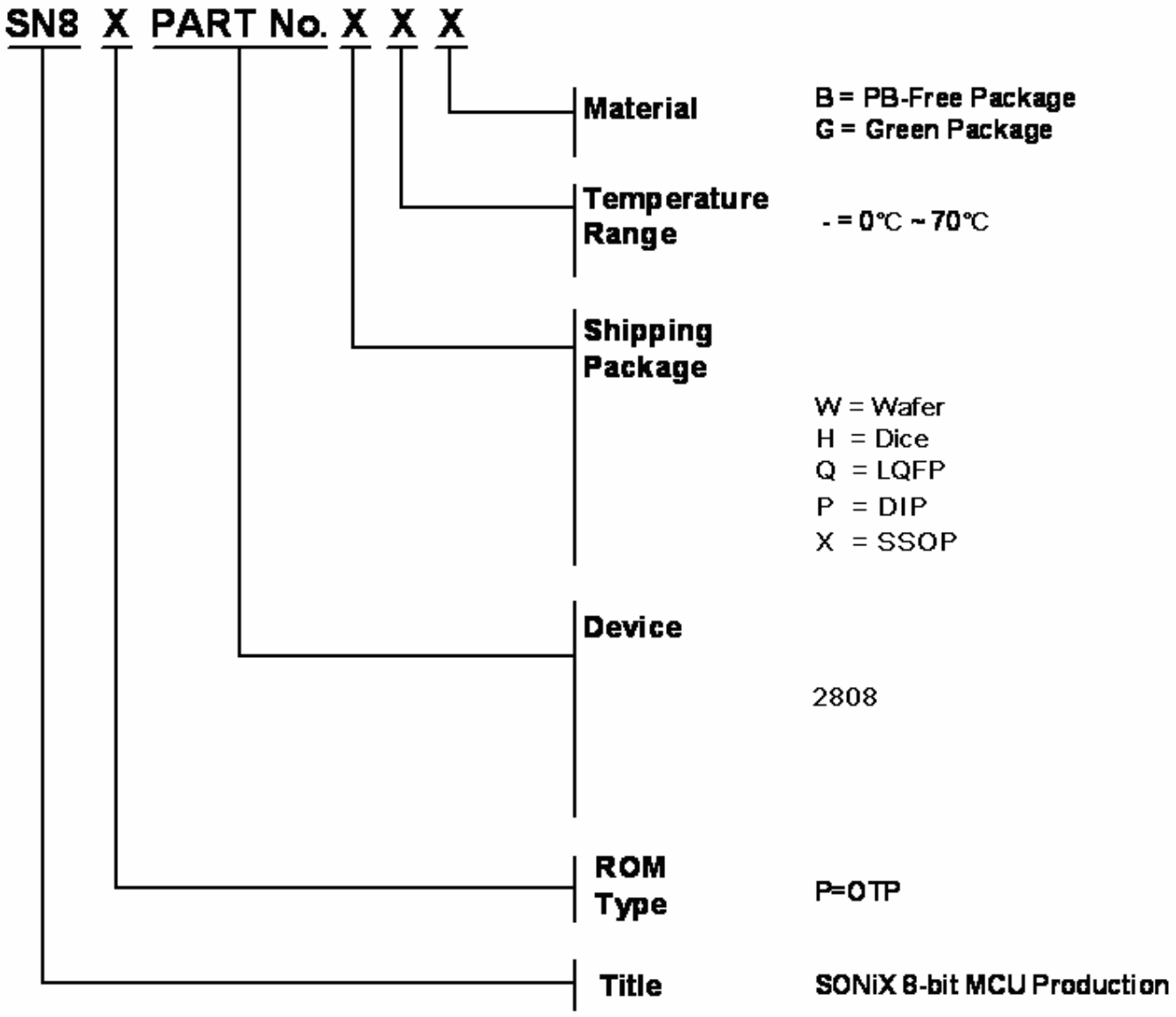
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.220	-	-	5.588
A1	0.015	-	-	0.381	-	-
A2	0.150	0.155	0.160	3.810	3.937	4.064
D	2.400	2.450	2.550	60.960	62.230	64.770
E		0.600			15.240	
E1	0.540	0.545	0.550	13.716	13.843	13.970
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.630	0.650	0.067	16.002	16.510	1.702
θ°	0°	7°	15°	0°	7°	15°

15 Marking Definition

15.1 INTRODUCTION

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

15.2 MARKING INDETIFICATION SYSTEM

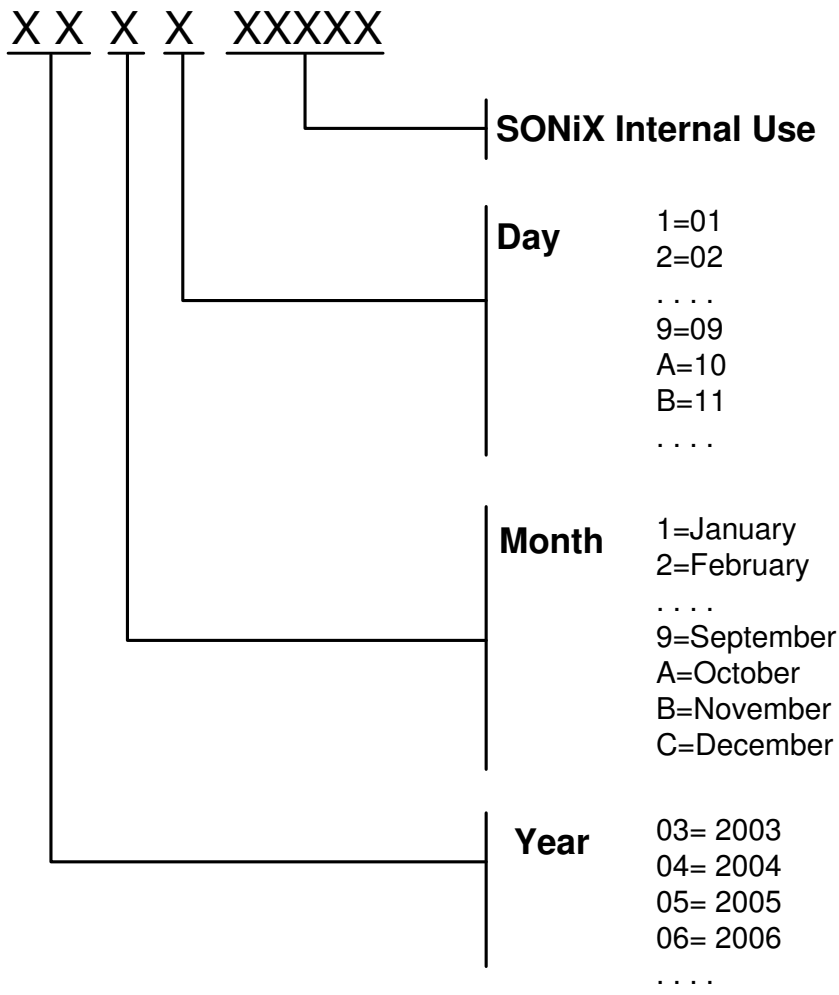


15.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P2808QG	OTP	TA01	LQFP	0℃~70℃	Green Package
SN8P2808QB	OTP	TA01	LQFP	0℃~70℃	PB-Free Package

15.4 Datecode system

There are total 8~9 letters of SONiX datecode system. The final four or five char. are for Sonix inside use only, and the first 4 indicate the Package date including Year/Month/Date. The detail information is following:



Main Office:
Address: 9F, NO. 8, Hsien Cheng 5th St, Chupei City, Hsinchu, Taiwan R.O.C.
Tel: 886-3-551 0520
Fax: 886-3-551 0523

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