SN8P1989

USER'S MANUAL

Preliminary Specification Version 1.0

SONiX 8-Bit Micro-Controller

SONIX reserves the right to make change without further notice to any products herein to improve reliability, function or design. SONIX does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. SONIX products are not designed, intended, or authorized for us as components in systems intended, for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SONIX product could create a situation where personal injury or death may occur. Should Buyer purchase or use SONIX products for any such unintended or unauthorized application. Buyer shall indemnify and hold SONIX and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that SONIX was negligent regarding the design or manufacture of the part.

AMENDENT HISTORY

Version	Date	Description
VER V1.0	May. 2009	V1.0 First issue

Table of Content

1 PRODUCT OVERVIEW

1.1 FEATURES

- ***Memory configuration***
  OTP ROM size: 12K * 16 bits
  RAM size: 512 * 8 bits (bank 0)
  8-levels stack buffer
  LCD RAM size: 28*4 bits
- ***I/O pin configuration***
  Input only: P0

  Bi-directional: P1, P2, P5

  Wakeup: P0, P1

  Pull-up resisters: P0, P1, P2, P5

  External interrupt: P0

- ***Powerful instructions***
  Four clocks per instruction cycle
  All instructions are one word length
  Most of instructions are 1 cycle only.
  Maximum instruction cycle is “2”.
  JMP instruction jumps to all ROM area.
  All ROM area look-up table function (MOVC)

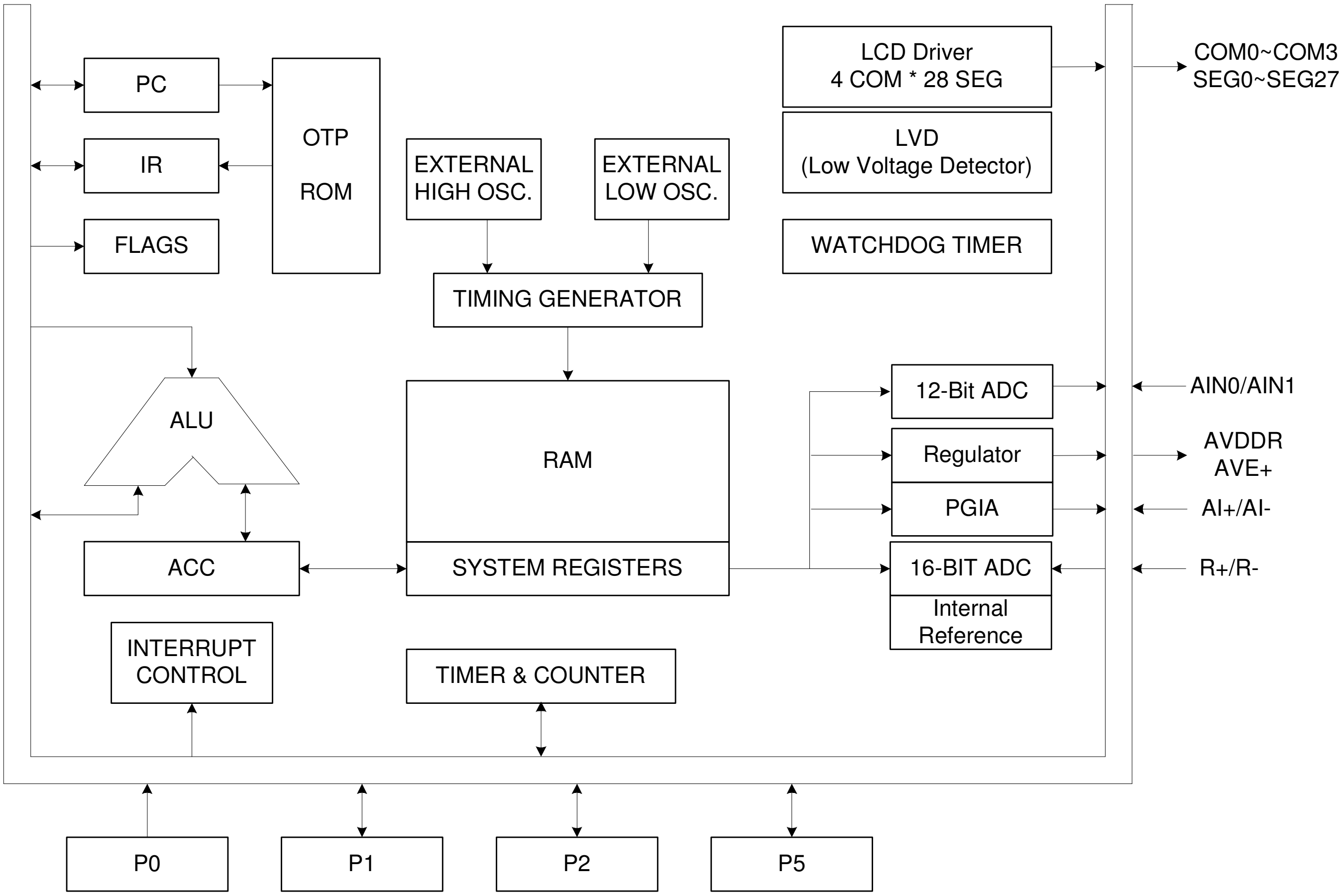
- ***Programmable gain instrumentation amplifier***
- ***Gain option: 1x/12.5x/50x/100x/200x***
- **16-bit Delta-Sigma ADC with 14-bit noise free with one fully differential ADC input channel**
- **12-bit ADC with Two channel single-ended Input and built in Battery Detect.**
- **Timer**
  An 8-bit basic timer with green mode wakeup function.
  Two 8-bit timer counter with PWM or Buzzer
  On chip watchdog timer
  Real Time Clock(RTC) with 0.5 second.

- ***Seven interrupt sources***
  Three internal timer interrupts: T0, TC0, TC1
  Two external interrupts: INT0, INT1
  16-bit ADC end of conversion interrupt

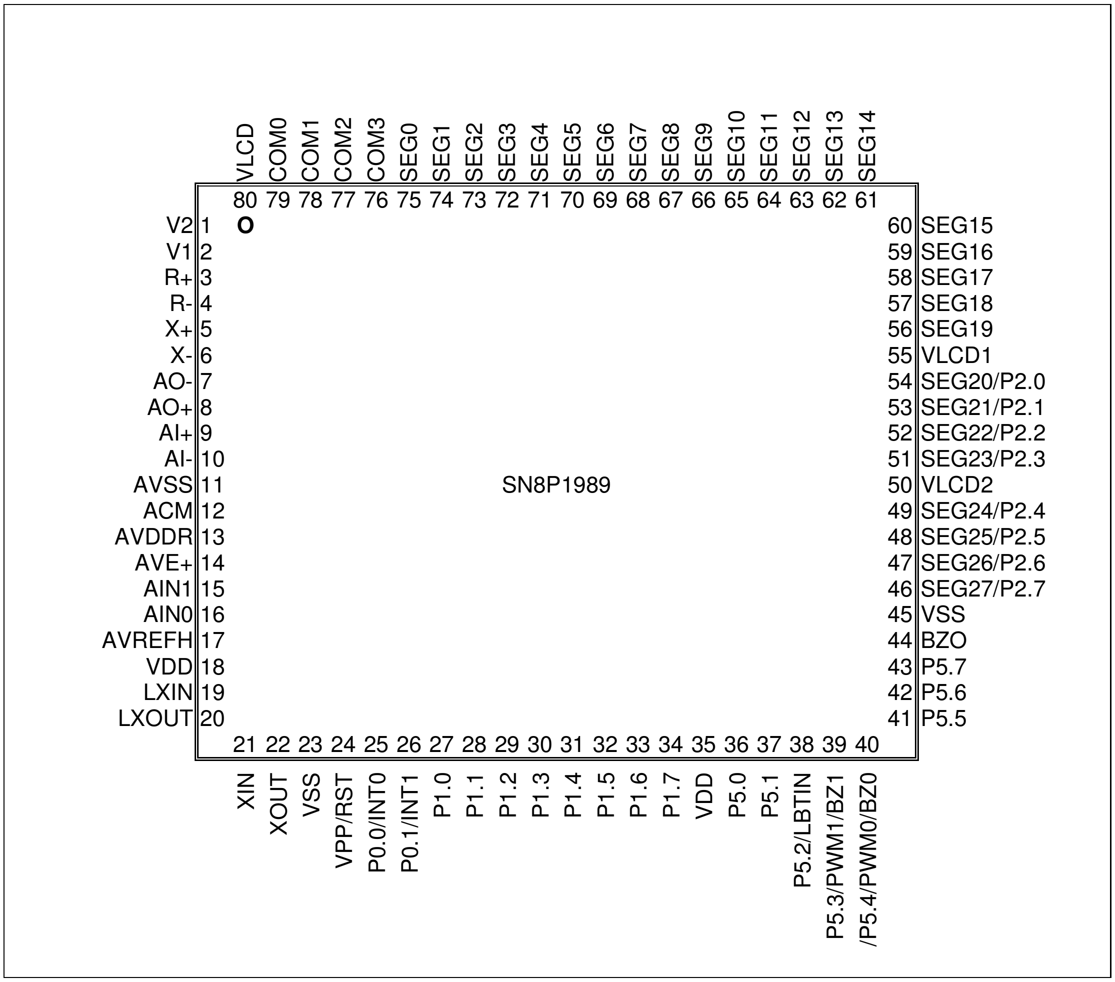
- ***Single power supply: 4.2V ~5.5V***
- ***On-chip watchdog timer***
- ***On-chip Regulator with 3.8V voltage output and 10mA driven current.***
- ***On chip regulator with 3.0V/2.4V/1.5V output voltage***
- ***On-chip 1.2V Band gap reference for battery monitor.***
- ***On chip Voltage Comparator.***
- ***Build in 16-bit ADC reference voltage V(R+,R-)=0.8V /0.64V/0.4V.***
- **Build in Buzzer output (BZO) of 6.25K or 50K.**
- ***LCD driver:***
  1/3 bias voltage.
  4 common * 28 segment

- ***Dual clock system offers four operating modes***
  External high clock: up to 8 MHz
  Normal mode: Both high and low clock active.
  Slow mode: Low clock only.
  Green mode with T0 period wake-up
  Sleep mode: Both high and low clock stop.
- ***Package***
  LQFP 80, Dice

1.2 SYSYEM BLOCK



1.3 PIN ASSIGNMENT

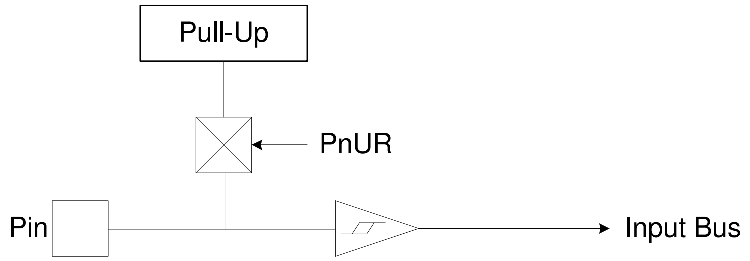


1.4 PIN DESCRIPTIONS

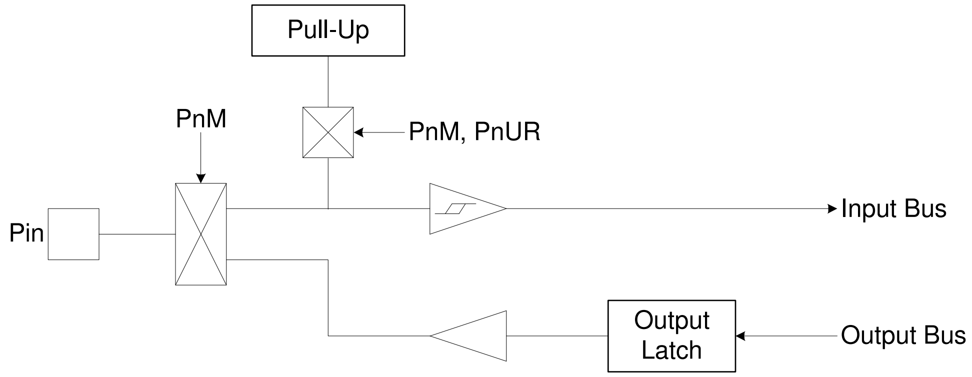
PIN NAME	***TYPE***	***DESCRIPTION***
VDD, VSS, AVSS	P	Power supply input pins for digital / analog circuit. VDD pin function description: Pin 38: For Digital function and IO power. Pin 21: For 12-bit ADC circuit Power Pin 17: For Regulator/PGIA/16-bit ADC Power.
VLCD/VLCD1/VLCD2	P	LCD Power supply input
AVDDR	P	Regulator power output pin, Voltage=3.8V.
AVE+	P	Regulator output =3.0V or 1.5V for Sensor. Maximum output current=10 mA
ACM	P	Band Gap Voltage output =1.2V
R+	AI	Positive reference input
R-	AI	Negative reference input
X+	AI	Positive ADC differential input, a 0.1uF capacitor connect to pin X-
X-	AI	Negative ADC differential input
AI+/-	AI	PGIA/16 bit ADC Analog input channel
AIN0, AIN1	I/O	12-bit ADC Input channel
VPP/ RST	P, I	OTP ROM programming pin. System reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”.
XIN, XOUT	I, O	External High clock oscillator pins.
LXIN, LXOUT	I, O	External Low clock oscillator pins.
P0.0 / INT0	I	Port 0.0 and shared with INT0 trigger pin (Schmitt trigger) / Built-in pull-up resisters.
P0.1 / INT1	I	Port 0.1 and shared with INT1 trigger pin (Schmitt trigger) / Built-in pull-up resisters.
P1 [7:0]	I/O	Port 1.0~Port 1.7 bi-direction pins / wakeup pins/ Built-in pull-up resisters.
P2 [7:0]	I/O	Port 2.0~Port 2.7 bi-direction pins / Built-in pull-up resisters. Shared with LCD
P5 [2:0], P5 [5:7]	I/O	Port 5.0~Port 5.2, P5.5~P5.7, bi-direction pins / Built-in pull-up resisters.
P5.3 / BZ1 / PWM1	I/O	Port 5.3 bi-direction pin, Built-in pull-up resisters. Buzzer1 or PWM1 output pin.
P5.4 / BZ0 / PWM0	I/O	Port 5.4 bi-direction pin, Built-in pull-up resisters. Buzzer0 or PWM0 output pin.
LBTIN	I	Low Battery detect input pin shared with P5.2
COM [3:0]	O	COM0~COM3 LCD driver common port
SEG0 ~ SEG27	O	LCD driver segment pins.
BZO	O	Build in Buzzer output of 6.25K to 250K.

1.5 PIN CIRCUIT DIAGRAM

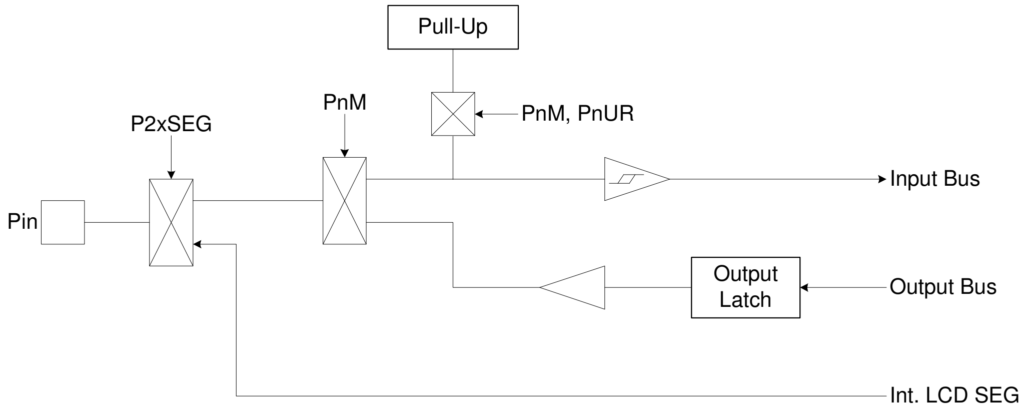
Port 0



Port 1, Port5



Port 2



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ ***12K words ROM***

	ROM	
0000H	***Reset vector***	User reset vector
0001H	***General purpose area***	Jump to user start address
0002H		Jump to user start address
0003H		Jump to user start address
0004H	***Reserved***	
0005H		
0006H		
0007H		
0008H	***Interrupt vector***	User interrupt vector
0009H	***General purpose area***	User program
.		
.		
000FH		
0010H		
0011H		
.		
.		
2FFBH		End of user program
2FFCH	***Reserved***	
.		
2FFFH		

2.1.1.1 RESET VECTOR (0000H)

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

☞ **Power On Reset**
☞ **Watchdog Rese**
☞ **External Reset**

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. The following example shows the way to define the reset vector in the program memory.

➢ **Example: Defining Reset Vector**

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

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


                ENDP                      ; End of program
```

2.1.1.2 INTERRUPT VECTOR (0008H)

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

> ✸ ***Note: "PUSH", "POP" instructions only process 0x80~0x87 working registers and PFLAG register. Users have to save and load ACC by program as interrupt occurrence.***

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

```
.DATA           ACCBUF    DS   1          ; Define ACCBUF for store ACC data.

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

                ORG       8               ; Interrupt vector.
                B0XCH     A, ACCBUF       ; Save ACC in a buffer
                PUSH                      ; Save 0x80~0x87 working registers and PFLAG register to
                                          buffers.
                …
                …
                POP                       ; Load 0x80~0x87 working registers and PFLAG register
                                          from buffers.
                B0XCH     A, ACCBUF       ; Restore ACC from buffer
                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.**

```
.DATA           ACCBUF     DS   1          ; Define ACCBUF for store ACC data.

.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.
                B0XCH      A, ACCBUF       ; Save ACC in a buffer
                PUSH                       ; Save 0x80~0x87 working registers and PFLAG register to
                                           buffers.
                …
                …
                POP                        ; Load 0x80~0x87 working registers and PFLAG register
                                           from buffers.
                B0XCH      A, ACCBUF       ; Restore ACC from buffer
                RETI                       ; End of interrupt service routine.
                …
                ENDP                       ; End of program.
```

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

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

2.1.1.3 LOOK-UP TABLE DESCRIPTION

In the ROM's data lookup function, X register is pointed to high byte address (bit 16~bit 23), Y register is pointed to middle byte address (bit 8~bit 15) and Z register is pointed to low byte address (bit 0~bit 7) of ROM. After MOVC instruction executed, the low-byte data will be stored in ACC and high-byte data stored in R register.

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

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

                                        ; Increment the index address for next address.
                INCMS   Z               ; Z+1
                JMP     @F              ; Z is not overflow.
                INCMS   Y               ; Z is overflow, Y=Y+1.
                JMP     @F              ; Y is not overflow.
                INCMS   X               ; Y is overflow, X=X+1.
                NOP
                                        ;
 @@:            MOVC                    ; To lookup data, R = 51H, ACC = 05H.
                …                       ;
TABLE1:         DW      0035H           ; To define a word (16 bits) data.
                DW      5105H
                DW      2012H
                …
```

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

➢ **Example: INC_XYZ macro.**

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

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

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

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

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

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

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

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

```
                B0MOV     X, #TABLE1$H    ; To set lookup table1's high address
                B0MOV     Y, #TABLE1$M    ; To set lookup table1's middle address
                B0MOV     Z, #TABLE1$L    ; To set lookup table's low address.

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

                B0BTS1    FC              ; Check the carry flag.
                JMP       GETDATA         ; FC = 0
                INCMS     Y               ; FC = 1. Y+1.
                JMP       GETDATA         ; Y is not overflow.
                INCMS     X               ; Y is overflow, X=X+1.
                NOP

GETDATA:                                  ;
                MOVC                      ; To lookup data. If BUF = 0, data is 0x0035
                                          ; If BUF = 1, data is 0x5105
                                          ; If BUF = 2, data is 0x2012
                …

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

2.1.1.4 JUMP TABLE DESCRIPTION

The jump table operation is one of multi-address jumping function. Add low-byte program counter (PCL) and ACC value to get one new PCL. 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).

When carry flag occurs after executing of "ADD PCL, A", it will not affect PCH register. Users have to check if the jump table leaps over the ROM page boundary or the listing file generated by SONIX assembly software. If the jump table leaps over the ROM page boundary (e.g. from xxFFH to xx00H), move the jump table to the top of next program memory page (xx00H). **Here one page mean 256 words.**

> ✸ ***Note: Program counter can't carry from PCL to PCH when PCL is overflow after executing addition instruction.***

➢ **Example: Jump table.**

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

        B0ADD       PCL, A          ; PCL = PCL + ACC, the PCH can't be changed.
        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
```

In following example, the jump table starts at 0x00FD. When execute B0ADD PCL, A. If ACC = 0 or 1, the jump table points to the right address. If the ACC is larger then 1 will cause error because PCH doesn't increase one automatically. We can see the PCL = 0 when ACC = 2 but the PCH still keep in 0. The program counter (PC) will point to a wrong address 0x0000 and crash system operation. It is important to check whether the jump table crosses over the boundary (xxFFH to xx00H). A good coding style is to put the jump table at the start of ROM boundary (e.g. 0100H).

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

```
ROM Address
    …
    …
    …
  0X00FD    B0ADD       PCL, A          ; PCL = PCL + ACC, the PCH can't be changed.
  0X00FE    JMP         A0POINT         ; ACC = 0
  0X00FF    JMP         A1POINT         ; ACC = 1
  0X0100    JMP         A2POINT         ; ACC = 2   ← jump table cross boundary here
  0X0101    JMP         A3POINT         ; ACC = 3
    …
    …
```

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

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

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

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

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

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

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

- **Example: "@JMP_A" operation.**

; Before compiling program.

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

; After compiling program.

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

2.1.1.5 CHECKSUM CALCULATION

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

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

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

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

2.1.2 CODE OPTION TABLE

Code Option	Content	Function Description
Watch_Dog	Enable	Enable Watchdog function
	Disable	Disable Watchdog function
Noise Filter	Enable	Enable Noise Filter function
	Disable	Disable Noise Filter function
Security	Enable	Enable ROM code Security function
	Disable	Disable ROM code Security function
INT_16K_RC	Always_ON	Force Watch Dog Timer clock source come from INT 16K RC. Also INT 16K RC never stop both in power down and green mode that means Watch Dog Timer will always enable both in power down and green mode.
	By_CPUM	Enable or Disable internal 16K(@ 3V) RC clock by CPUM register
Low Power	Enable	Enable Low Power function to save Operating current
	Disable	Disable Low Power function

✸ **Note:**

1. ***In high noisy environment, set Watch_Dog as "Enable" and INT_16K_RC as "Always_ON" is strongly recommended.***
2. ***In high noisy environment, disable "Low Power" is strongly recommended.***
3. ***The side effect is to increase the lowest valid working voltage level if enable "Low Power" code option.***
4. ***Enable "Low Power" option will reduce operating current except in slow mode.***

2.1.3 DATA MEMORY (RAM)

☞ ***512 X 8-bit RAM***

		RAM location	
	000h	General purpose area	; 000h~07Fh of Bank 0 = To store general
		.	; purpose data (128 bytes).
BANK 0	07Fh	.	
	080h	System register	; 080h~0FFh of Bank 0 = To store system
		.	; registers (128 bytes).
	0FFh	End of bank 0 area	
	0100h	General purpose area	; 0100h~01FFh of Bank 1 = To store general
		.	; purpose data (256 bytes).
BANK1		.	
		.	
	01FFh	End of bank1area	
	0200h	General purpose area	; 0200h~027Fh of Bank 2 = To store general
BANK 2		.	; purpose data (128 bytes).
	027Fh	End of bank 2 area	
	F00h	LCD RAM area	; Bank 15 = To store LCD display data
BANK 15		.	; (28 bytes).
	F1Ch	End of LCD Ram	;

2.1.4 SYSTEM REGISTER

2.1.4.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	X	PFLAG	RBANK	OPTION	LCDM1	-	-	-	-	-	-
9	AMPM	AMPCHS	AMPCKS	ADC16M	ADCKS	REGM		DFM	ADCDL	ADCDH	LBTM	BZC	BZM	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	ADC12M	ADB	ADR	-	-	-	-	-	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	-	TC0R	PCL	PCH
D	P0	P1	P2	-	-	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	P2UR	-	-	P5UR	@HL	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.1.4.2 SYSTEM REGISTER DESCRIPTION

L, H = Working, @HL and ROM addressing register
Y, Z = Working, @YZ and ROM addressing register
PFLAG = ROM page and special flag register
AMPM = PGIA mode register
AMPCKS = PGIA clock selection
ADCKS = ADC clock selection

ADCDL = ADC low-byte data buffer
P_NM = Port N input/output mode register
P_N = Port N data buffer
INTEN = Interrupt enable register
LCDM1= LCD mode register
T0M = Timer 0 mode register
@HL = RAM HL indirect addressing index pointer.

LBTM= Low Battery Detect Register
T0C = Timer 0 counting register.
TC1M = Timer/Counter 1 mode register.
TC1C = Timer/Counter 1 counting register.

X = Working register and ROM look-up data buffer
R = Working register and ROM look-up data buffer
AMPCHS = PGIA channel selection
ADC16M = 16 bit ADC's mode register
REGM = Regulator mode
DFM = Decimation filter mode
ADCDH = ADC high-byte data buffer
P_NUR = Port N pull-up register
INTRQ = Interrupt request register
OSCM = Oscillator mode register
PCH, PCL = Program counter
STK0~STK7 = Stack 0 ~ stack 7 buffer
T0C = Timer 0 counting register
@YZ = RAM YZ indirect addressing index pointer.
STKP = Stack pointer buffer
TC0M = Timer/Counter 0 mode register.
TC0C = Timer/Counter 0 counting register.
TC0R = Timer/Counter 0 auto-reload data buffer.
TC1R = Timer/Counter 1 auto-reload data buffer.

2.1.4.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Name
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
085H	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0	R/W	X
086H	-	-	-	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	RBNKS3	RBNKS2	RBNKS1	RBNKS0	R/W	RBANK
088H	-	-	-	-	-	-	-	RCLK	R/W	OPTION
089H	-	-	LCDBNK	-	LCDENB	-	P2HSEG	P2LSEG	R/W	LCDM1
090H	-	BGRENB	FDS1	FDS0	GS2	GS1	GS0	AMPENB	R/W	AMPM
091H	-	-	-	-	ADC16CHS3	ADC16CHS2	ADC16CHS1	ADC16CHS0	R/W	AMPCHS
092H	-	-	-	-	-	AMPCKS2	AMPCKS1	AMPCKS0	W	AMPCKS
093H	-	-	-	-	IRVS	RVS1	RVS0	ADC16ENB	R/W	ADC16M
094H	ADCKS7	ADCKS6	ADCKS5	ADCKS4	ADCKS3	ADCKS2	ADCKS1	ADCKS0	W	ADCKS
095H	ACMENB	AVDDRENB	AVENB	AVESEL1	AVESEL0	-	-	REGENB	R/W	REGM
097H			-	-	-	WRS0		DRDY	R/W	DFM
098H	ADCB7	ADCB6	ADCB5	ADCB4	ADCB3	ADCB2	ADCB1	ADCB0	R	ADCDL
099H	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB9	ADCB8	R	ADCDH
09AH	-	-	-	-	-	LBTO	-	LBTENB	R/W	LBTM
09BH	BZC7	BZC6	BZC 5	BZC4	BZC3	BZC2	BZC1	BZC0	W	BZC
09CH	BZOENB	BZORATE2	BZORATE1	BZORATE0	-	BZOX8	-	-	R/W	BZM
0B1H	ADC12ENB	ADS	EOC	GCHS	-	-	ADC12CHS1	ADC12CHS0	R/W	ADC12M
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0B3H	-	AD12CKS1	-	AD12CKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0BFH	PEDGEN	-	-	P00G1	P00G0	-	-	-	R/W	PEDGE
0C0H	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	R/W	P1W
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M
0C5H	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H	-	TC1IRQ	TC0IRQ	T0IRQ	-	ADC16IRQ	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	-	TC1IEN	TC0IEN	T0IEN	-	ADC16IEN	P01IEN	P00IEN	R/W	INTEN
0CAH	WTCKS	WDRST	WDRATE	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	PC13	PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	-	-	-	-	-	-	P01	P00	R	P0
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2
0D5H	P57	P56	P55	P54	P53	P52	P51	P50	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	-	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0	-	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	-	-	-	STKPB3	STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	-	-	-	-	-	-	P01R	P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2UR
0E5H	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ

0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	-	-	S7PC13	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	-	-	S6PC13	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	-	-	S5PC13	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	-	-	S4PC13	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	-	-	S3PC13	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	-	-	S2PC13	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	-	-	S1PC13	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	-	-	S0PC13	S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ ***Note:***

1. ***To avoid system error, make sure to put all the "0" and "1" as it indicates in the above table.***
2. ***All of register names had been declared in SN8ASM assembler.***
3. ***One-bit name had been declared in SN8ASM assembler with "F" prefix code.***
4. ***"b0bset", "b0bclr", "bset", "bclr" instructions are only available to the "R/W" registers.***

2.1.4.4 ACCUMULATOR

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

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

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

```
; Read ACC data and store in BUF data memory

                MOV         BUF, A

; Write a immediate data into ACC

                MOV         A, #0FH

; Write ACC data from BUF data memory

                MOV         A, BUF
```

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

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

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

                RETI                        ; Exit interrupt service vector
```

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

2.1.4.5 PROGRAM FLAG

The PFLAG register contains the arithmetic status of ALU operation, C, DC, Z bits indicate the result status of ALU operation.

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

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

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

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

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

2.1.4.6 PROGRAM COUNTER

The program counter (PC) is a 13-bit binary counter separated into the high-byte 6 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

Besides, it can be replaced with specific address by executing CALL or JMP instruction. When JMP or CALL instruction is executed, the destination address will be inserted to bit 0 ~ bit 10.

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

☞ **ONE ADDRESS SKIPPING**

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

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

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

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

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

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

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

INCS instruction:

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

INCMS instruction:

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

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

DECS instruction:

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

DECMS instruction:

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

☞ **MULTI-ADDRESS JUMPING**

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

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

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

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

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

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

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

2.1.5 H, L REGISTERS

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

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

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

2.1.5.1 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.***

2.1.5.2 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
                …
```

2.1.5.3 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.
```

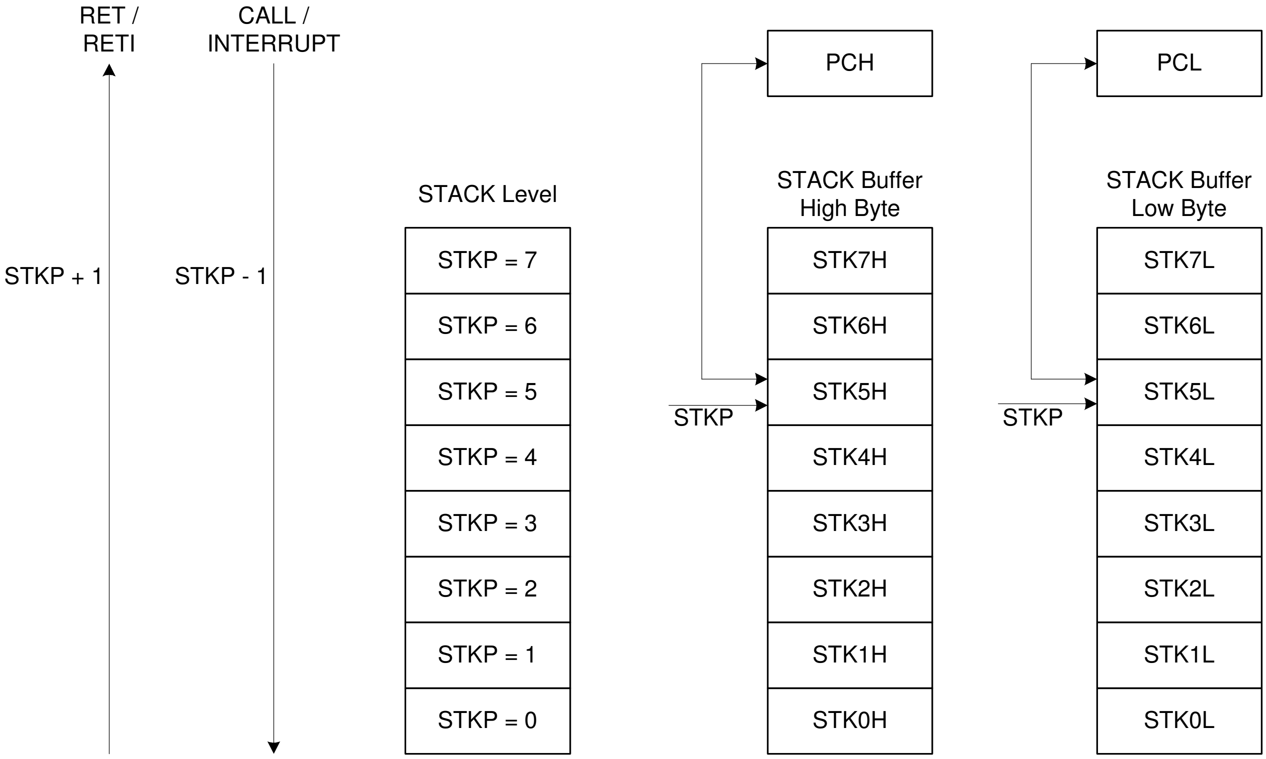
Example: Indirectly addressing mode with @YZ register

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

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

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

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

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

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

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

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

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

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

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

2.3.3 STACK OPERATION EXAMPLE

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

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

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

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

3 RESET

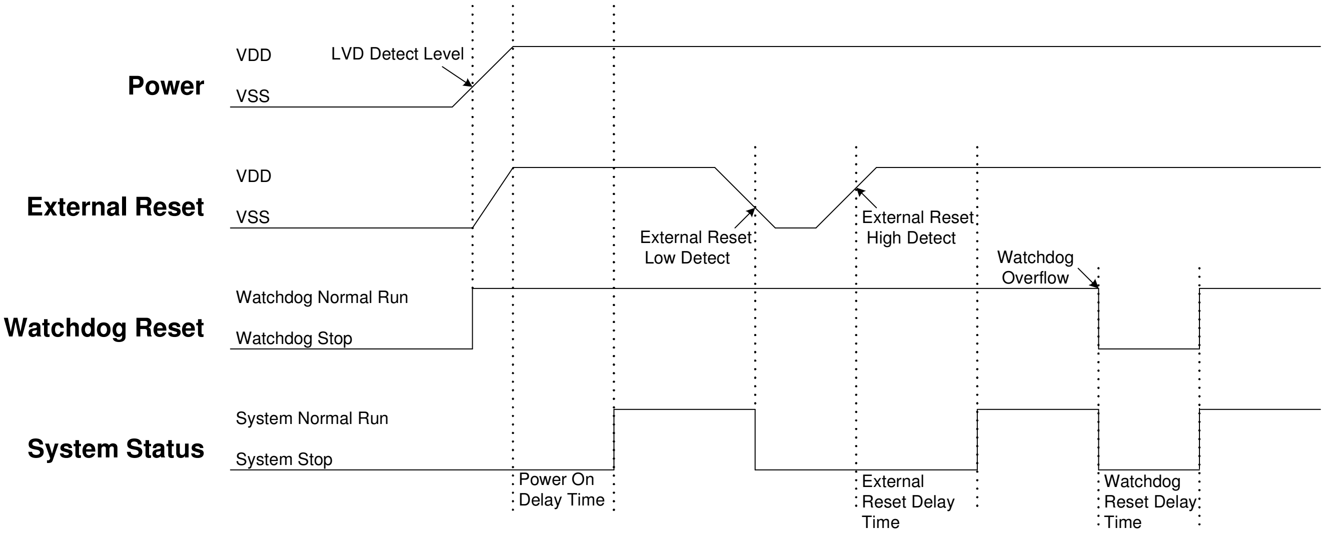
3.1 OVERVIEW

The system would be reset in three conditions as following.

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

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.

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



3.2 POWER ON RESET

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

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

3.3 WATCHDOG RESET

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

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

Watchdog timer application note is as following.

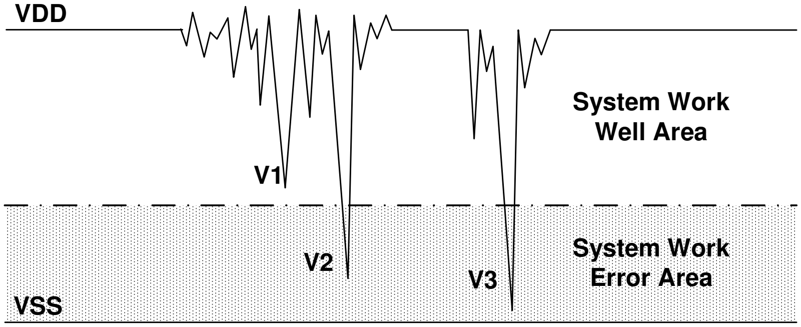
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

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

3.4 BROWN OUT RESET

3.4.1 BROWN OUT DESCRIPTION

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



Brown Out Reset Diagram

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

DC application:

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

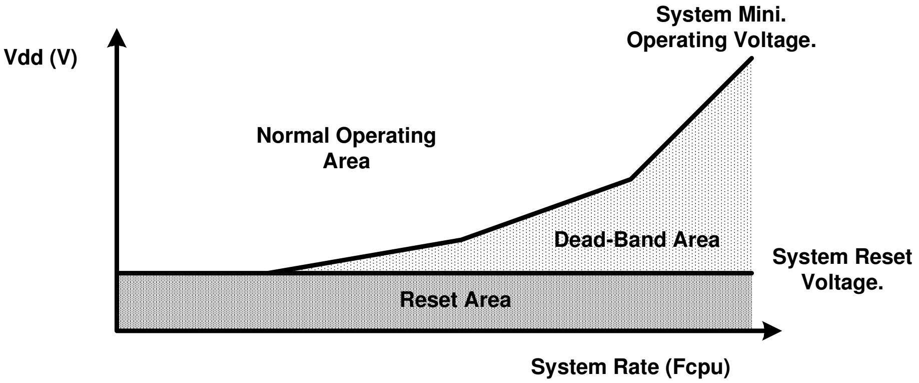
AC application:

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

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

3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION

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



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

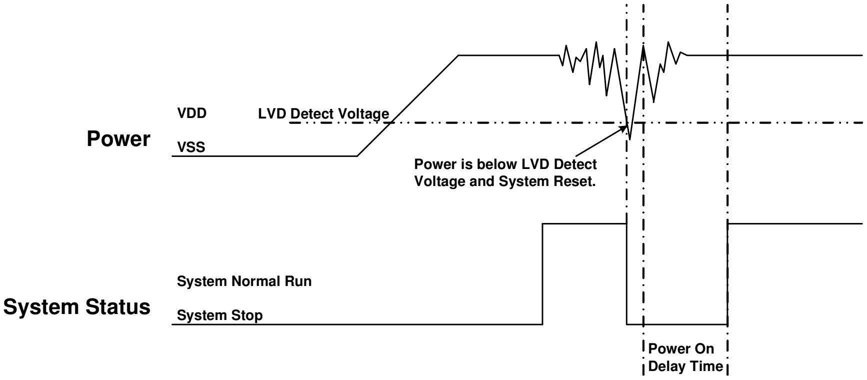
3.4.3 BROWN OUT RESET IMPROVEMENT

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

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

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

LVD reset:



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

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

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

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

3.5 EXTERNAL RESET

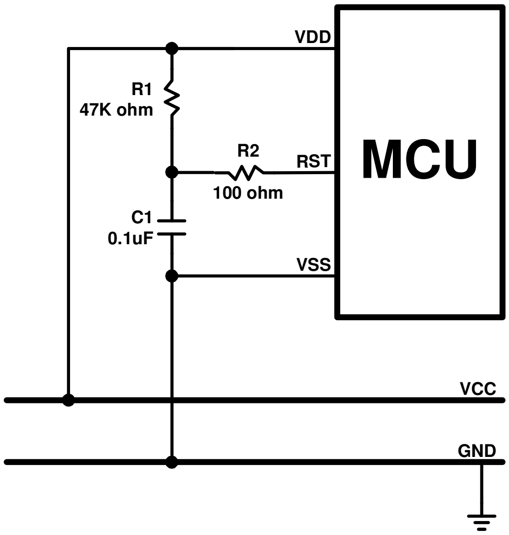
External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the low voltage and the system is reset. The external reset operation actives in power on and normal running mode. During system power-up, the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

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

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

3.6 EXTERNAL RESET CIRCUIT

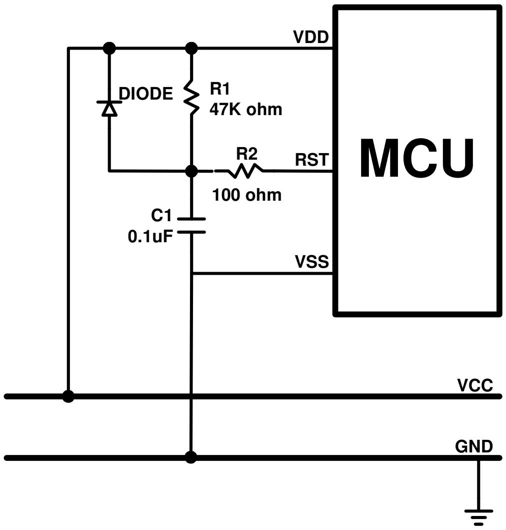
3.6.1 Simply RC Reset Circuit



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

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

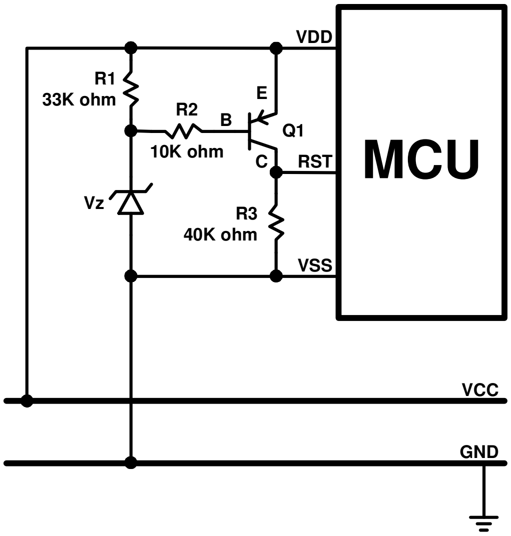
3.6.2 Diode & RC Reset Circuit



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

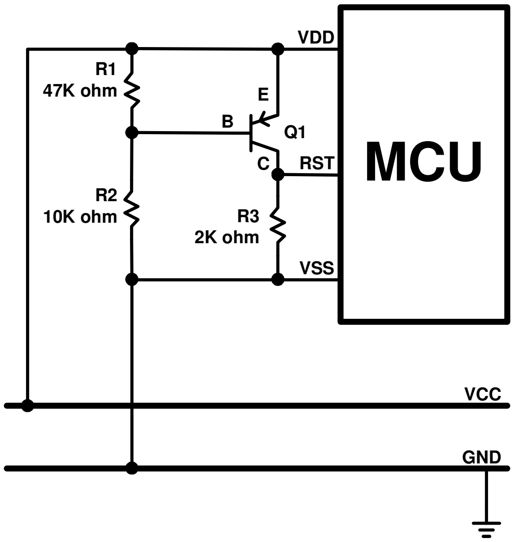
✸ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.6.3 Zener Diode Reset Circuit



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

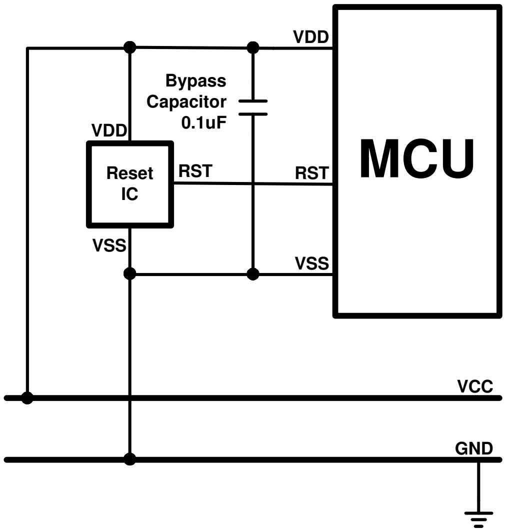
3.6.4 Voltage Bias Reset Circuit



The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to "0.7V x (R1 + R2) / R1", the C terminal of the PNP transistor outputs high voltage and MCU operates normally. When VDD is below "0.7V x (R1 + R2) / R1", the C terminal of the PNP transistor outputs low voltage and MCU is in reset mode.
Decide the reset detect voltage by R1, R2 resistances. Select the right R1, R2 value to conform the application. In the circuit diagram condition, the MCU's reset pin level varies with VDD voltage variation, and the differential voltage is 0.7V. If the VDD drops and the voltage lower than reset pin detect level, the system would be reset. If want to make the reset active earlier, set the R2 > R1 and the cap between VDD and C terminal voltage is larger than 0.7V. The external reset circuit is with a stable current through R1 and R2. For power consumption issue application, e.g. DC power system, the current must be considered to whole system power consumption.

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

3.6.5 External Reset IC



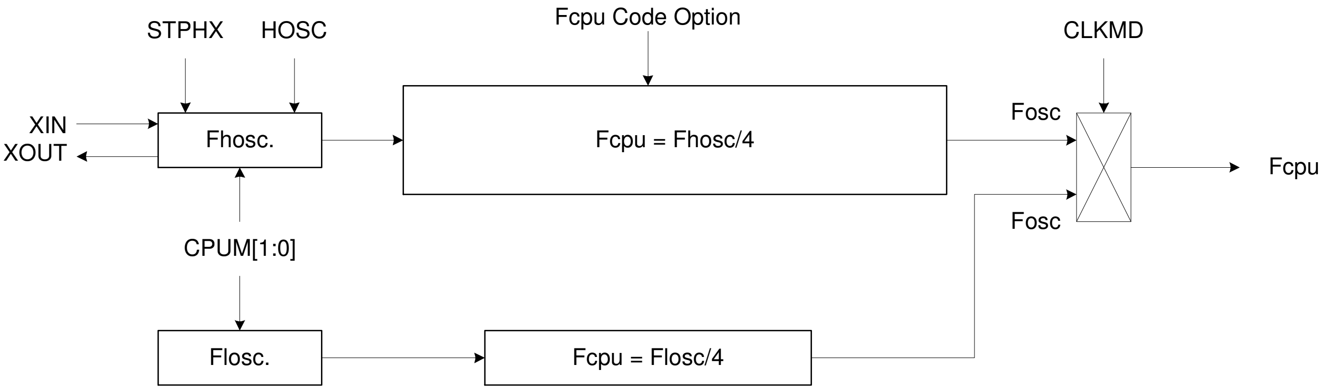
4 SYSTEM CLOCK

4.1 OVERVIEW

The micro-controller is a single clock system. The high-speed clock is generated from the external oscillator circuit. The high-speed clock can be system clock (Fosc). The system clock is divided by 4 to be the instruction cycle (Fcpu).

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

4.2 CLOCK BLOCK DIAGRAM



- HOSC: High_Clk code option.
- Fhosc: External high-speed clock.
- FLosc: External 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	WTCKS	WDRST	WDRATE	CPUM1	CPUM0	CLKMD	STPHX	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
After reset	0	0	0	0	0	0	0	-

Bit 1 **STPHX:** External high-speed oscillator control bit.
0 = External high-speed oscillator free run.

1 = External high-speed oscillator free run stop. External 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 external 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.

Bit5 **WDRATE:** Watchdog timer rate select bit.
0 = $F_{CPU} \div 2^{14}$
1 = $F_{CPU} \div 2^{8}$

Bit6 **WDRST:** Watchdog timer reset bit.
0 = No reset
1 = clear the watchdog timer's counter.
(The detail information is in watchdog timer chapter.)

Bit7 **WTCKS:** Watchdog clock source select bit.
0 = F_{CPU}
1 = internal RC low clock.

WTCKS	WTRATE	CLKMD	Watchdog Timer Overflow Time
0	0	0	1 / (fcpu ÷ 2^{14} ÷ 16) = 293 ms, Fosc=3.58MHz
0	1	0	1 / (fcpu ÷ 2^{8} ÷ 16) = 500 ms, Fosc=32768Hz
0	0	1	1 / (fcpu ÷ 2^{14} ÷ 16) = 65.5s, Fosc=16KHz@3V
0	1	1	1 / (fcpu ÷ 2^{8} ÷ 16) = 1s, Fosc=16KHz@3V
1	-	-	1 / (16K ÷ 512 ÷ 16) ~ 0.5s @3V

➢ **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 external low-speed oscillator will be stopped.**

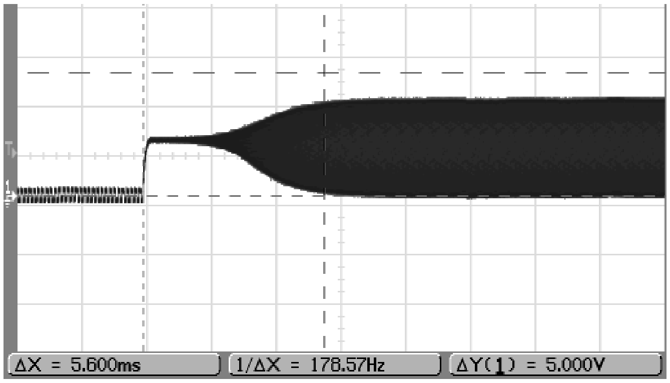
```
		B0BSET		FCPUM0			; To stop external high-speed oscillator and external low-speed
							; oscillator called power down mode (sleep mode).
```

4.4 SYSTEM HIGH CLOCK

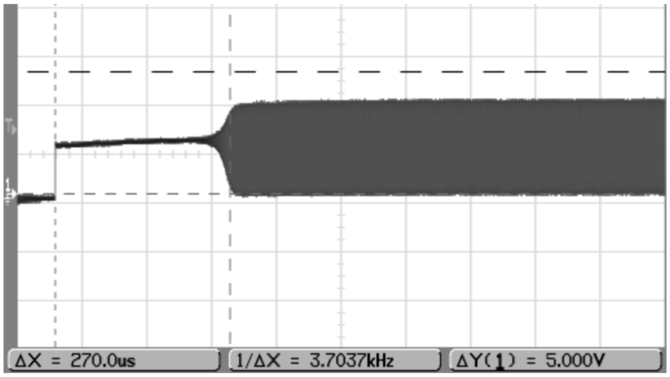
4.4.1 EXTERNAL HIGH CLOCK

External high clock includes Crystal/Ceramic modules .. The start up time of is longer. The oscillator start-up time decides reset time length.

4MHz Crystal

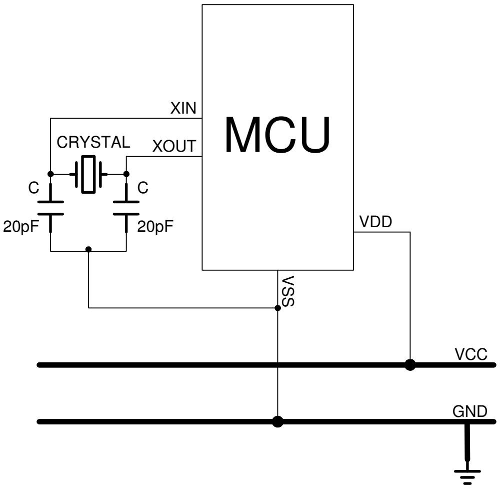


4MHz Ceramic



4.4.1.1 CRYSTAL/CERAMIC

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



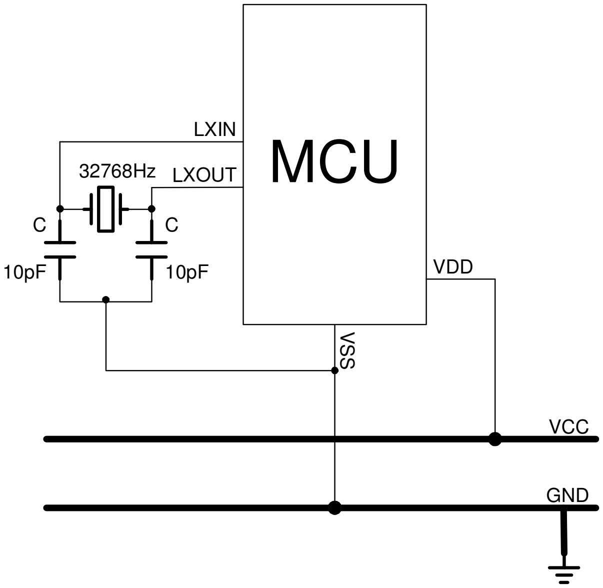
✸ ***Note: Connect the Crystal/Ceramic and C as near as possible to the XIN/XOUT/VSS pins of micro-controller.***

4.5 SYSTEM LOW CLOCK

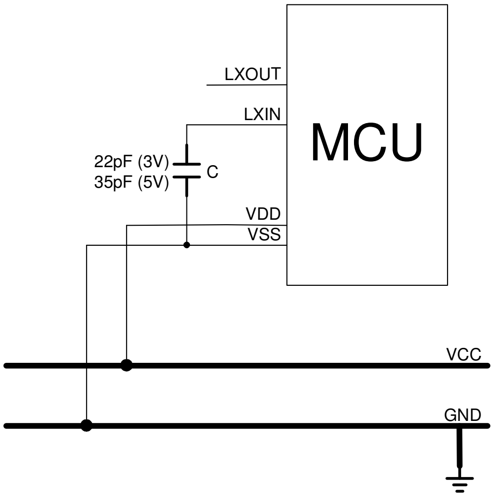
The system low clock source is the external low-speed oscillator. The low-speed oscillator can use 32768 crystal or RC type oscillator circuit.

4.5.1.1 CRYSTAL

Crystal devices are driven by LXIN, LXOUT pins. The 32768 crystal and 10pF capacitor must be as near as possible to MCU.



4.5.1.2 RC Type



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

- ☞ ***Flosc = External low oscillator***
- ☞ ***Slow mode Fcpu = Flosc / 4***

In power down mode the external low clock will be Stop.

➢ **Example: Stop internal low-speed oscillator by power down mode.**

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

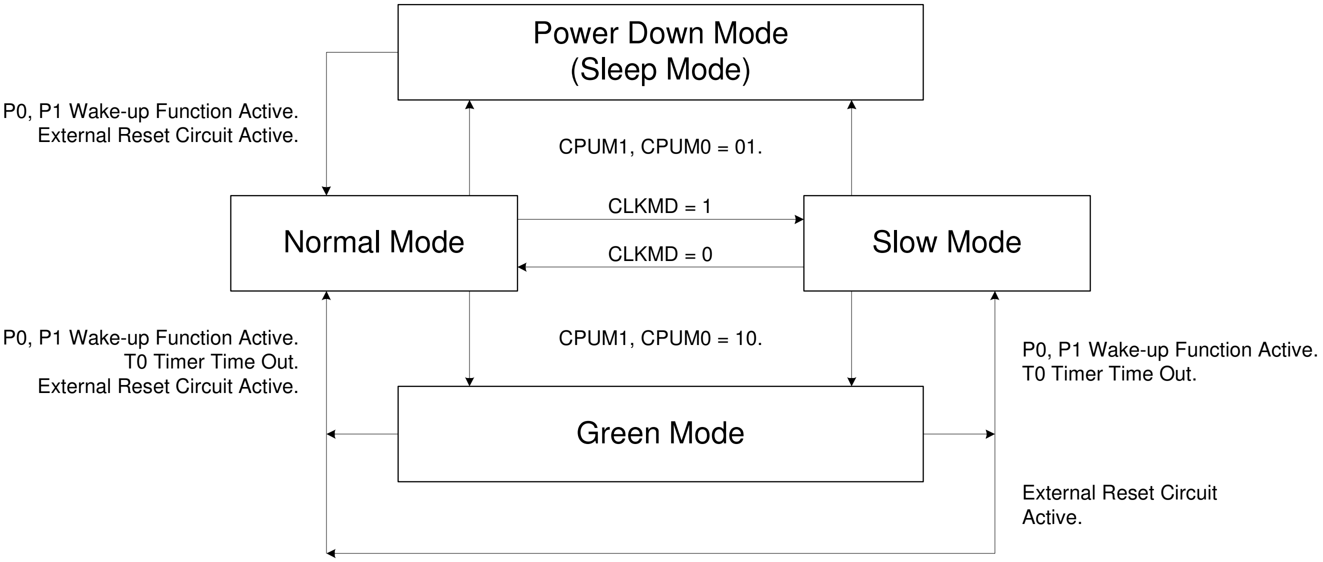
➢ ***Note: The external low-speed clock can’t be turned off individually. It is controlled by CPUM0, CPUM1 bits of OSCM register.***

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	
Ext. LRC	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	Inactive	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, TC0 Reset	P0, P1, Reset	

EHOSC: External high clock
Ext. LRC: External low clock

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/TC0 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) , program doesn’t execute. The wakeup trigger can wake the system up to normal mode. The wakeup trigger sources are external trigger (P0, P1 level change)

- Power down mode is waked up to normal mode. The wakeup trigger is only external trigger (P0, P1 level change)

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.

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

5.3.3 P1W WAKEUP CONTROL REGISTER

Under power down mode (sleep mode) and green mode, the I/O ports with wakeup function are able to wake the system up to normal mode. The Port 0 and Port 1 have wakeup function. Port 0 wakeup function always enables, but the Port 1 is controlled by the P1W register.

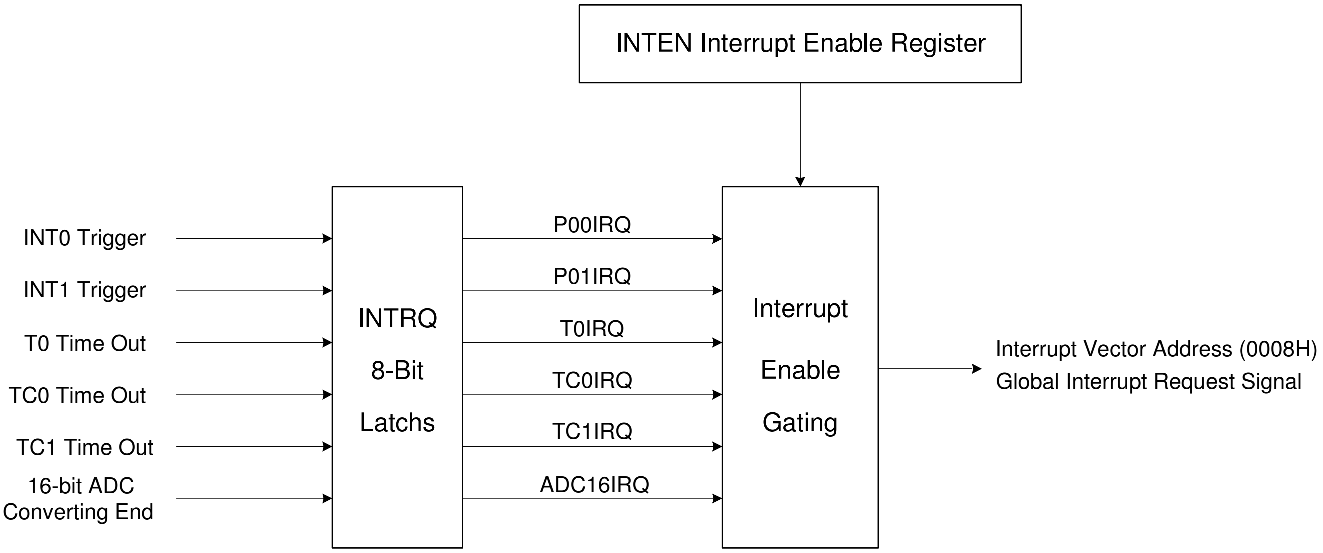
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

6 INTERRUPT

6.1 OVERVIEW

This MCU provides eight interrupt sources, including four internal interrupt (T0/TC0/TC1/16bit 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.***

6.2 INTEN INTERRUPT ENABLE REGISTER

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	-	TC1IEN	TC0IEN	T0IEN	-	ADC16IEN	P01IEN	P00IEN
Read/Write	-	R/W	R/W	R/W	-	R/W	R/W	R/W
After reset	-	0	0	0	-	0	0	0

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

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

Bit 2 **ADC16IEN:** 16bit ADC Converting end interrupt control bit.
0 = Disable 16bit ADC interrupt function.
1 = Enable16bit ADC 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.

6.3 INTRQ INTERRUPT REQUEST REGISTER

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

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	-	TC1IRQ	TC0IRQ	T0IRQ	-	ADC16IRQ	P01IRQ	P00IRQ
Read/Write	-	R/W	R/W	R/W	-	R/W	R/W	R/W
After reset	-	0	0	0	-	0	0	0

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

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

Bit 2 **ADC16IRQ:** 16bit ADC Converting end interrupt request flag.
0 = None 16bit ADC interrupt request.
1 = 16bit ADC 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.

6.4 GIE GLOBAL INTERRUPT OPERATION

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

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

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

Example: Set global interrupt control bit (GIE).

```
B0BSET      FGIE            ; Enable GIE
```

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

6.5 PUSH, POP ROUTINE

When any interrupt occurs, system will jump to ORG 8 and execute interrupt service routine. It is necessary to save ACC, PFLAG data. The chip includes "PUSH", "POP" for in/out interrupt service routine. The two instruction only save working registers 0x80~0x87 including **PFLAG** data into buffers. The ACC data must be saved by program.

✸ ***Note:***

1. ***"PUSH", "POP" instructions only process 0x80~0x87 working registers and PFLAG register. Users have to save and load ACC by program as interrupt occurrence.***
2. ***The buffer of PUSH/POP instruction is only one level and is independent to RAM or Stack area.***

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

```
.DATA           ACCBUF      DS 1            ; ACCBUF is ACC data buffer.

.CODE
                ORG         0
                JMP         START

                ORG         8
                JMP         INT_SERVICE

                ORG         10H
START:
                …

INT_SERVICE:
                B0XCH       A, ACCBUF       ; Save ACC in a buffer
                PUSH                        ; Save 0x80~0x87 working registers and PFLAG register to
                                            buffers.
                …
                …
                POP                         ; Load 0x80~0x87 working registers and PFLAG register
                                            from buffers.
                B0XCH       A, ACCBUF       ; Restore ACC from buffer

                RETI                        ; Exit interrupt service vector
                …
                ENDP
```

6.6 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	PEDGEN	-	-	P00G1	P00G0	-	-	-
	R/W	-	-	R/W	R/W	-	-	-

Bit7 **PEDGEN:** Interrupt and wakeup trigger edge control bit.
0 = Disable edge trigger function.
Port 0: Low-level wakeup trigger and falling edge interrupt trigger.
Port 1: Low-level wakeup trigger.
1 = Enable edge trigger function.
P0.0: Both Wakeup and interrupt trigger are controlled by P00G1 and P00G0 bits.
P0.1: Wakeup trigger and interrupt trigger is Level change (falling or rising edge).
Port 1: Wakeup trigger is Level change (falling or rising edge).

Bit[4:3] **P00G[1:0]:** Port 0.0 edge select bits.
00 = reserved,
01 = falling edge,
10 = rising edge,
11 = rising/falling bi-direction.

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

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

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

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

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

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

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

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

            RETI                            ; Exit interrupt vector
```

6.7 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.

> ✸ ***Note: The interrupt trigger direction of P0.1 is controlled by PEDGEN bit.***

- **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
```

6.8 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.

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

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

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

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


            RETI                        ; Exit interrupt vector
```

6.9 TC0 INTERRUPT OPERATION

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

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

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

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

                B0BSET      FGIE             ; Enable GIE
```

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

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

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

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

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

                RETI                         ; Exit interrupt vector
```

6.10 TC1 INTERRUPT OPERATION

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

Example: TC1 interrupt request setup.

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

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

                B0BSET          FGIE                ; Enable GIE
```

Example: TC1 interrupt service routine.

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

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

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

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

                RETI                                ; Exit interrupt vector
```

6.11 16 bit ADC INTERRUPT OPERATION

When the 16 bit ADC converting successfully, the ADC16IRQ will be set to “1” no matter the ADC16IEN is enable or disable. If the ADC16IEN and the trigger event ADC16IRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the ADC16IEN = 0, the trigger event ADC16IRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the ADC16IEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

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

```
            B0BCLR      FADC16IEN       ; Disable ADC interrupt service

            B0BSET      FADC16IEN       ; Enable ADC interrupt service
            B0BCLR      FADC16IRQ       ; Clear ADC interrupt request flag
            B0BSET      FGIE            ; Enable GIE

            B0BSET      FAD16ENB        ; 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      FADC16IRQ       ; Check 16 bit ADCIRQ
            JMP         EXIT_INT        ; ADCIRQ = 0, exit interrupt vector

            B0BCLR      FADC16IRQ       ; Reset 16 bit ADCIRQ
            …                           ; ADC interrupt service routine
            …
EXIT_INT:
            …                           ; Pop routine to load ACC and PFLAG from buffers.

            RETI                        ; Exit interrupt vector
```

6.12 MULTI-INTERRUPT OPERATION

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

Interrupt Name	***Trigger Event Description***
P00IRQ	P0.0 trigger controlled by PEDGE
P01IRQ	P0.1 trigger controlled by PEDGE
ADC16IRQ	16-bit ADC converting end.
T0IRQ	T0C overflow
TC0IRQ	TC0C overflow
TC1IRQ	TC1C overflow

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          FP01IEN             ; Check P01IEN
                JMP             INTP01CHK           ; Jump check to next interrupt
                B0BTS0          FP01IRQ             ; Check P01IRQ
                JMP             INTP01
INTDADCCHK:                                         ; Check 16-bit ADC interrupt request
                B0BTS1          FADC16IEN           ; Check DADCIEN
                JMP             INTT0CHK            ; Jump check to next interrupt
                B0BTS0          FADC16IRQ           ; Check DADCIRQ
                JMP             INTADC16
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             INTSADCHK           ; Jump check to next interrupt
                B0BTS0          FTC1IRQ             ; Check TC1IRQ
                JMP             INTT1               ; Jump to TC1 interrupt service routine
INT_EXIT:
                …                                   ; Pop routine to load ACC and PFLAG from buffers.

                RETI                                ; Exit interrupt vector
```

7 I/O PORT

7.1 I/O PORT MODE

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

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

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

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	P57M	P56M	P55M	P54M	P53M	P522M	P51M	P50M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit[7:0] **PnM[7:0]:** Pn mode control bits. (n = 0~5).
0 = Pn is input mode.
1 = Pn is output mode.

✸ ***Note:***

1. ***Users can program them by bit control instructions (B0BSET, B0BCLR).***
2. ***Port 2 is shared with LCD***

➢ **Example: I/O mode selecting**

```
        CLR         P1M             ; Set all ports to be input mode.
        CLR         P2M


        MOV         A, #0FFH        ; Set all ports to be output mode.
        B0MOV       P1M,A
        B0MOV       P2M, A

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

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

7.2 I/O PULL UP REGISTER

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

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

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

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

- **Example: I/O Pull up Register**

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

7.3 I/O PORT DATA REGISTER

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

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

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

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	P57	P56	P55	P54	P53	P52	P51	P50
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

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

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

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

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

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

```
        B0BSET          P1.0                    ; Set P1.0 to be “1”.

        B0BCLR          P1.0                    ; Set P1.0 to be “0”.
```

7.4 Port 2 IO/LCD Selection

The port 2 is shared with LCD and can be set as IO pin by pin

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM1	-	-	LCDBNK	-	LCDENB	-	P2HSEG	P2LSEG
R/W	-	-	R/W	-	R/W	-	R/W	R/W
After Reset	-	-	0	-	0	-	0	0

Bit5 **LCDBNK:** LCD blank control bit.
0 = Normal display
1 = All of the LCD dots off.

Bit3 **LCDENB:** LCD driver enable control bit.
0 = Disable LCD function
1 = Enable LCD function

Bit1 P2HSEG**:** SEG20~23 LCD/IO selection bit.
0 = SEG20~23 as LCD function. VLCD1 connect to VLCD
1 = SEG20~23 as IO function (P2.0~P2.3). VLCD1 connect to VDD

Bit0 P2LSEG**:** SEG24~27 LCD/IO selection bit.
0 = SEG24~27 as LCD function. VLCD2 connect to VLCD
1 = SEG24~27 as IO function (P2.4~P2.7). VLCD1 connect to VDD

8 TIMERS

8.1 WATCHDOG TIMER (WDT)

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. The instruction that clears the watchdog timer (“ B0BSET FWDRST “) should be executed within a certain period. If an instruction that clears the watchdog timer is not executed within the period and the watchdog timer overflows, reset signal is generated and system is restarted.

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

Bit5 **WDRATE:** Watchdog timer rate select bit.
0 = $F_{CPU} \div 2^{14}$
1 = $F_{CPU} \div 2^{8}$

Bit6 **WDRST:** Watchdog timer reset bit.
0 = No reset
1 = clear the watchdog timer’s counter.
(The detail information is in watchdog timer chapter.)

Bit7 **WTCKS:** Watchdog clock source select bit.
0 = F_{CPU}
1 = internal RC low clock.

Watchdog timer overflow table.

WTCKS	WTRATE	CLKMD	Watchdog Timer Overflow Time
0	0	0	1 / (fcpu ÷ 2^{14} ÷ 16) = 293 ms, Fosc=3.58MHz
0	1	0	1 / (fcpu ÷ 2^{8} ÷ 16) = 500 ms, Fosc=32768Hz
0	0	1	1 / (fcpu ÷ 2^{14} ÷ 16) = 65.5s, Fosc=16KHz@3V
0	1	1	1 / (fcpu ÷ 2^{8} ÷ 16) = 1s, Fosc=16KHz@3V
1	-	-	1 / (16K ÷ 512 ÷ 16) ~ 0.5s @3V

✸ ***Note: The watchdog timer can be enabled or disabled by the code option.***

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
```

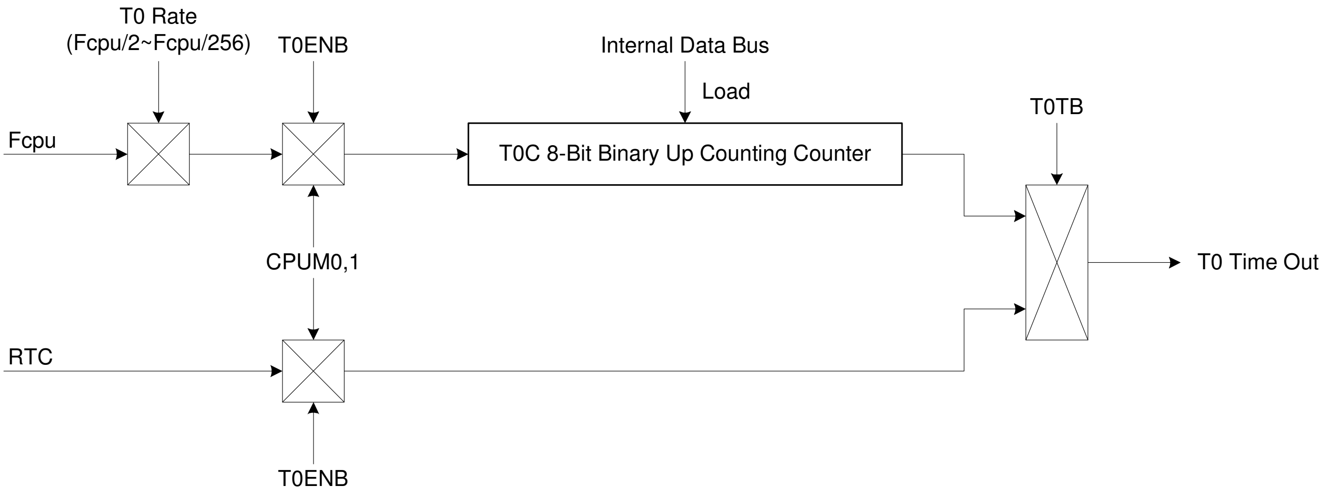
8.2 TIMER 0 (T0)

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

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

8.2.4 T0 TIMER OPERATION SEQUENCE

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

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

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

☞ **Set T0 timer rate.**

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

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

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

or

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

☞ **Set T0 interrupt interval time.**

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

☞ **Set T0 timer function mode.**

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

☞ **Enable T0 timer.**

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

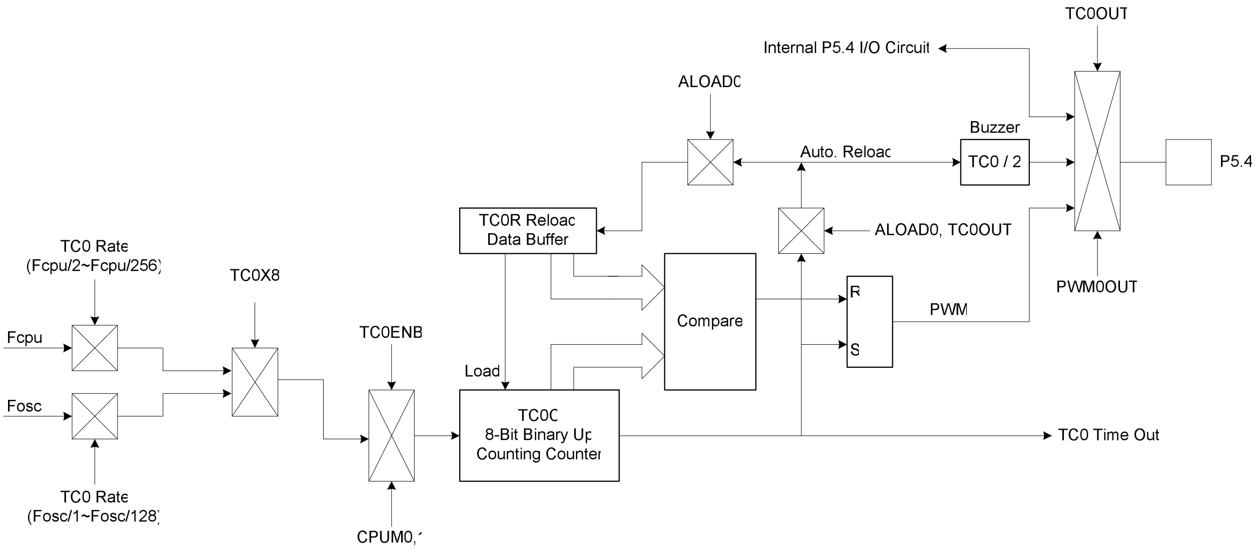
8.3 TIMER/COUNTER 0 (TC0)

8.3.1 OVERVIEW

The TC0 is an 8-bit binary up counting timer. TC0 clock sources came internal 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). 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.
☞ **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	-	ALOAD0	TC0OUT	PWM0OUT
Read/Write	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
After reset	0	0	0	0	-	0	0	0

Bit 0 **PWM0OUT:** PWM output control bit.
0 = Disable PWM output.
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 [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.

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	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.

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.

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 = 256 - (TC0 interrupt interval time * input clock)

TC0X8	**TC0C valid value**	**TC0C value binary type**	**Remark**
0 (Fcpu/2~ Fcpu/256)	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
1 (Fosc/1~ Fosc/128)	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).**

$$\begin{aligned} \text{TC0C initial value} &= N - (\text{TC0 interrupt interval time} * \text{input clock}) \\ &= 256 - (10ms * 4MHz / 4 / 64) \\ &= 256 - (10^{-2} * 4 * 10^{6} / 4 / 64) \\ &= 100 \\ &= 64H \end{aligned}$$

The basic timer table interval time of TC0, TC0X8 = 0.

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

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

TC0RATE	TC0CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fosc/128	8.192 ms	32 us	1000 ms	7812.5 us
001	Fosc/64	4.096 ms	16 us	500 ms	3906.25 us
010	Fosc/32	2.048 ms	8 us	250 ms	1953.125 us
011	Fosc/16	1.024 ms	4 us	125 ms	976.563 us
100	Fosc/8	0.512 ms	2 us	62.5 ms	488.281 us
101	Fosc/4	0.256 ms	1 us	31.25 ms	244.141 us
110	Fosc/2	0.128 ms	0.5 us	15.625 ms	122.07 us
111	Fosc/1	0.064 ms	0.25 us	7.813 ms	61.035 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.

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

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 = 256 - (TC0 interrupt interval time * input clock)

These parameters decide TC0 overflow time and valid value as follow table.

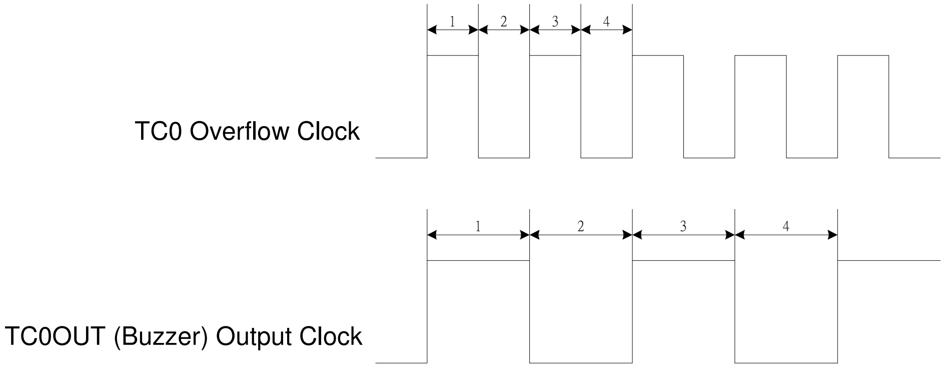
TC0X8	TC0R valid value	TC0R value binary type
0 (Fcpu/2~Fcpu/256)	0x00~0xFF	00000000b~11111111b
1 (Fosc/1~Fosc/128)	0x00~0xFF	00000000b~11111111b

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

$$
\begin{aligned}
\textit{TC0R initial value} &= 256 - (\textit{TC0 interrupt interval time} * \textit{input clock})\\
&= 256 - (10ms * 4MHz / 4 / 64)\\
&= 256 - (10^{-2} * 4 * 10^{6} / 4 / 64)\\
&= 100\\
&= 64H
\end{aligned}
$$

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 urce.**

```
            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.**

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

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

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

☞ **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.
```

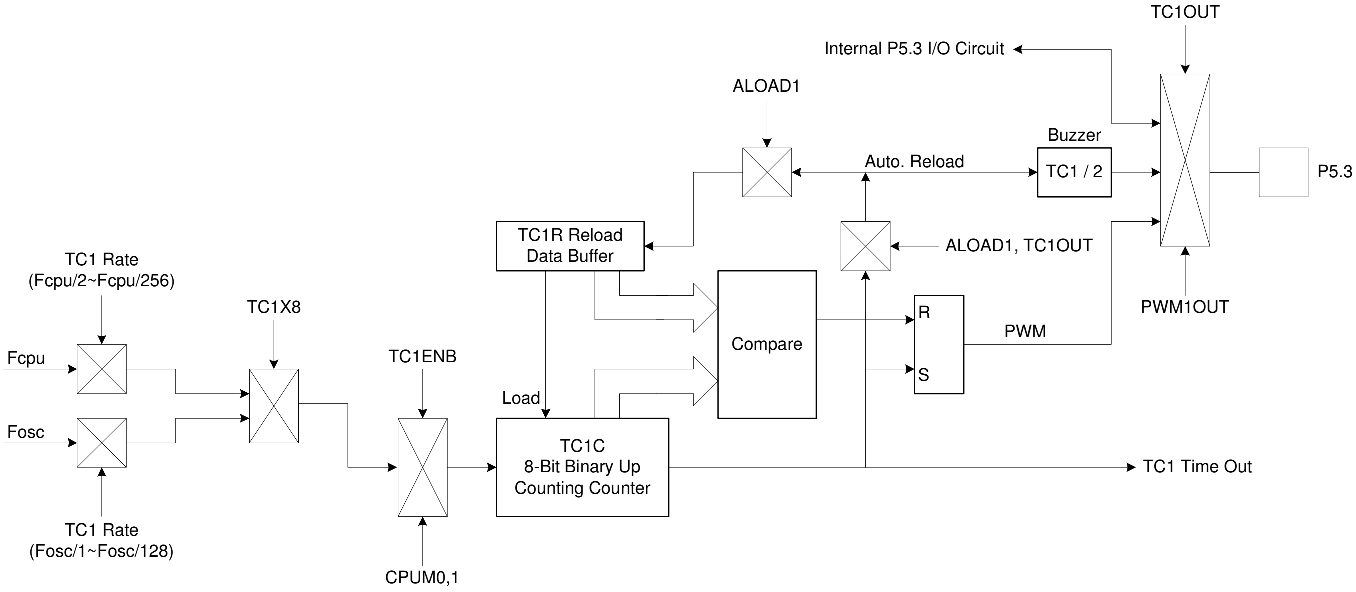
8.4 TIMER/COUNTER 1 (TC1)

8.4.1 OVERVIEW

The TC1 is an 8-bit binary up counting timer. TC1 clock source came from internal 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). 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.
☞ **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	-	ALOAD1	TC1OUT	PWM1OUT
Read/Write	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
After reset	0	0	0	0	-	0	0	0

Bit 0 **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 [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.

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.

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

These parameters decide TC1 overflow time and valid value as follow table.

TC1X8	TC1C valid value	TC1C value binary type	Remark
0 (Fcpu/2~Fcpu/256)	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
1 (Fosc/1~Fosc/128)	0x00~0xFF	00000000b~11111111b	Overflow per 256 count

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

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

The basic timer table interval time of TC1, TC1X8 = 0.

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

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

TC1RATE	TC1CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fosc/128	8.192 ms	32 us	1000 ms	7812.5 us
001	Fosc/64	4.096 ms	16 us	500 ms	3906.25 us
010	Fosc/32	2.048 ms	8 us	250 ms	1953.125 us
011	Fosc/16	1.024 ms	4 us	125 ms	976.563 us
100	Fosc/8	0.512 ms	2 us	62.5 ms	488.281 us
101	Fosc/4	0.256 ms	1 us	31.25 ms	244.141 us
110	Fosc/2	0.128 ms	0.5 us	15.625 ms	122.07 us
111	Fosc/1	0.064 ms	0.25 us	7.813 ms	61.035 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.

> ✸ ***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 = 256 - (TC1 interrupt interval time * input clock)

These parameters decide TC1 overflow time and valid value as follow table.

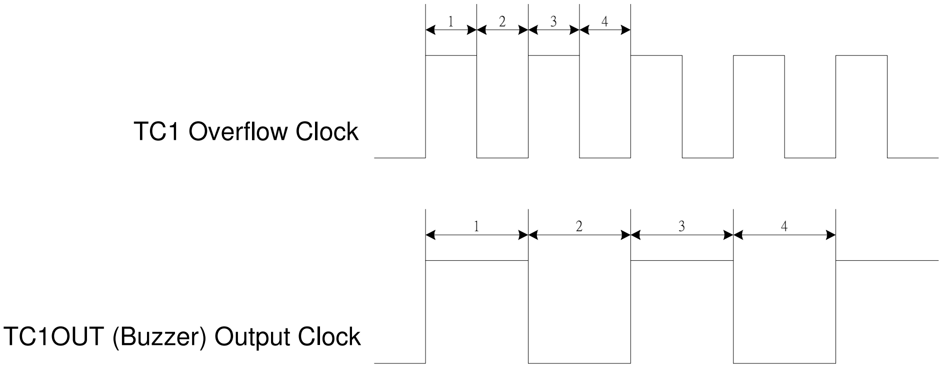
TC1X8	**TC1R valid value**	**TC1R value binary type**
0 (Fcpu/2~Fcpu/256)	0x00~0xFF	00000000b~11111111b
1 (Fosc/1~Fosc/128)	0x00~0xFF	00000000b~11111111b

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

*TC1R initial value = 256 - (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 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.**

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

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

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

☞ **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.
```

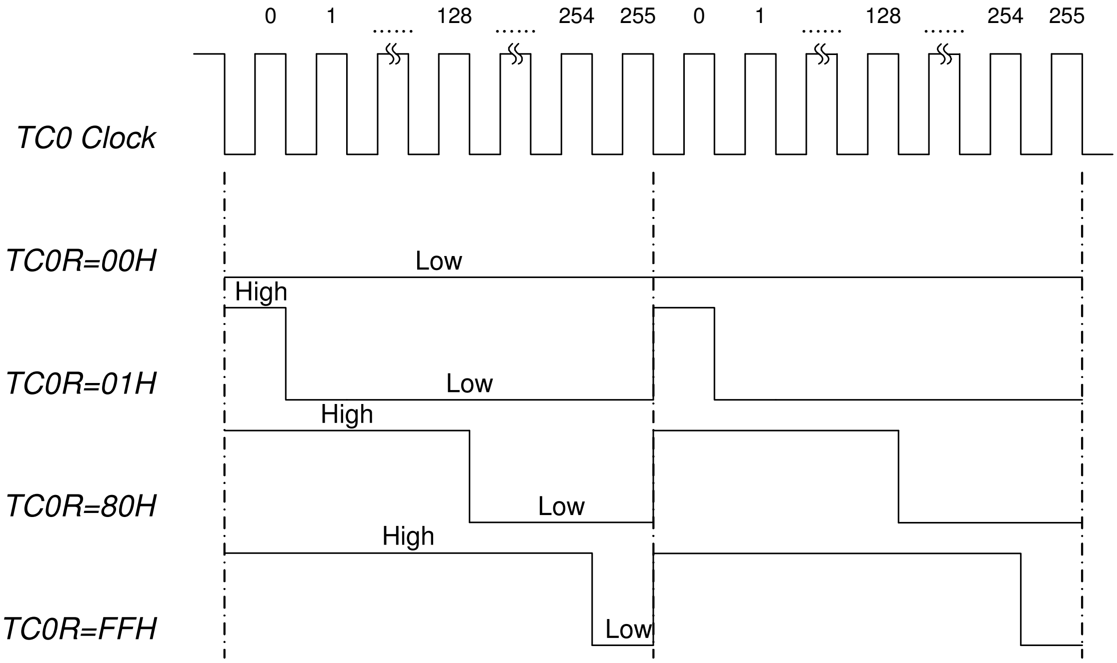
8.5 PWM0 MODE

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.

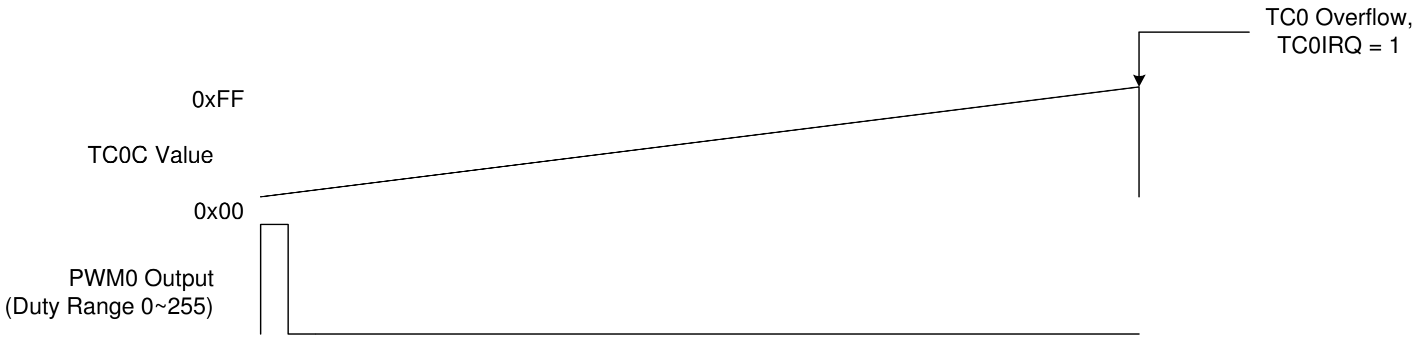
PWM duty range	TC0C valid value	TC0R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count

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.**

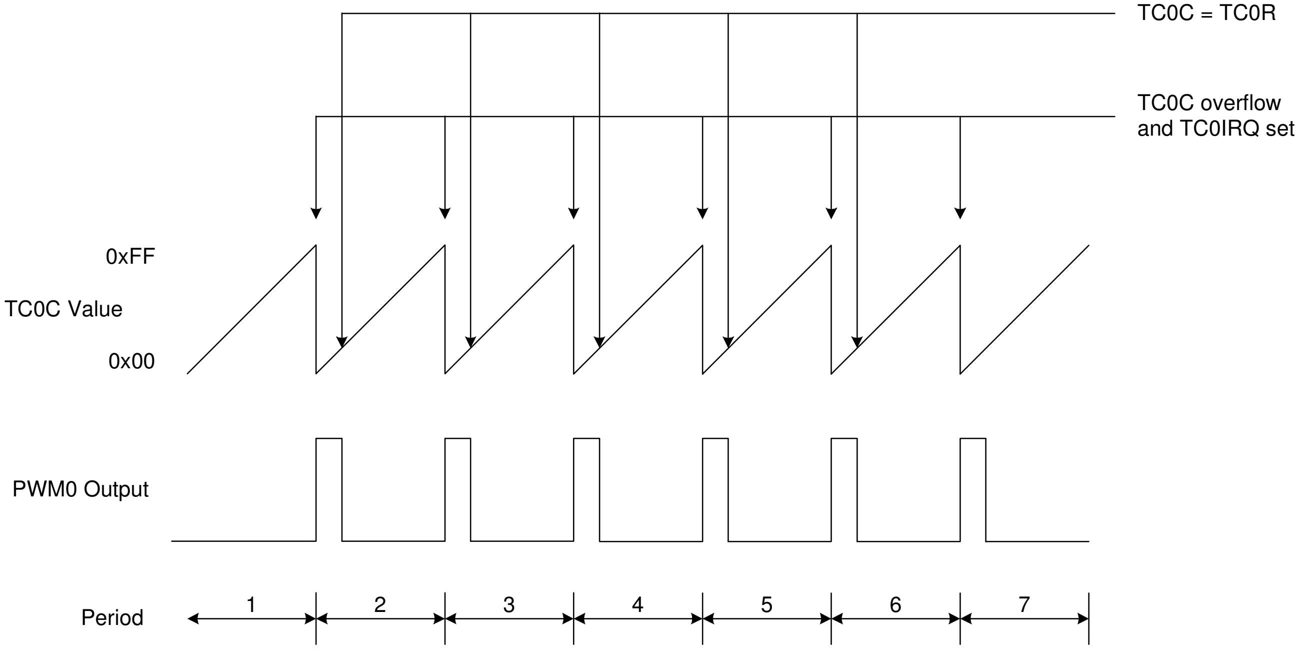
```
MOV       A, #30H         ; Input a number using B0MOV instruction.
B0MOV     TC0R, A

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.***

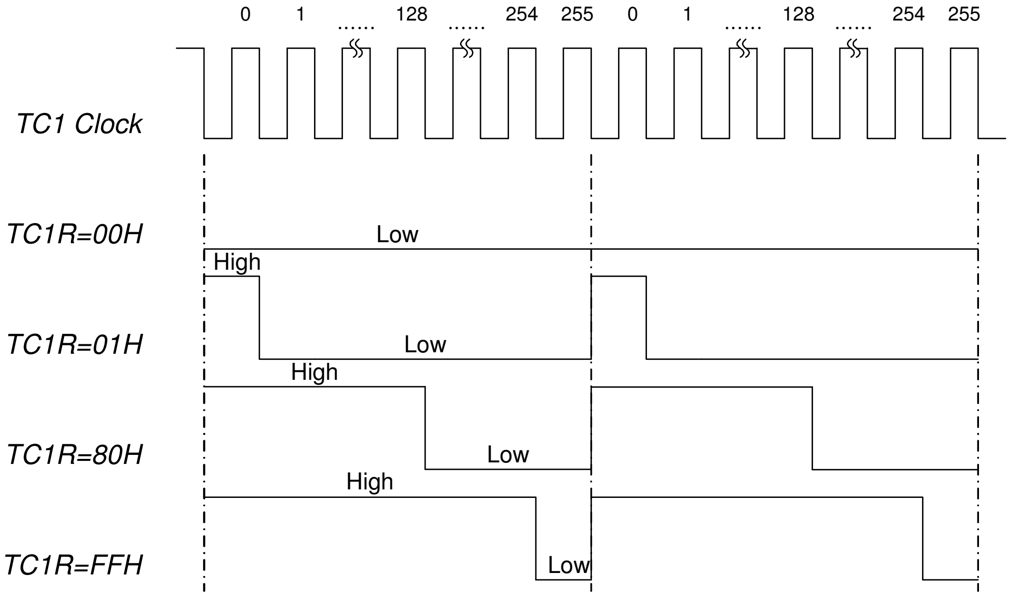
8.6 PWM1 MODE

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,

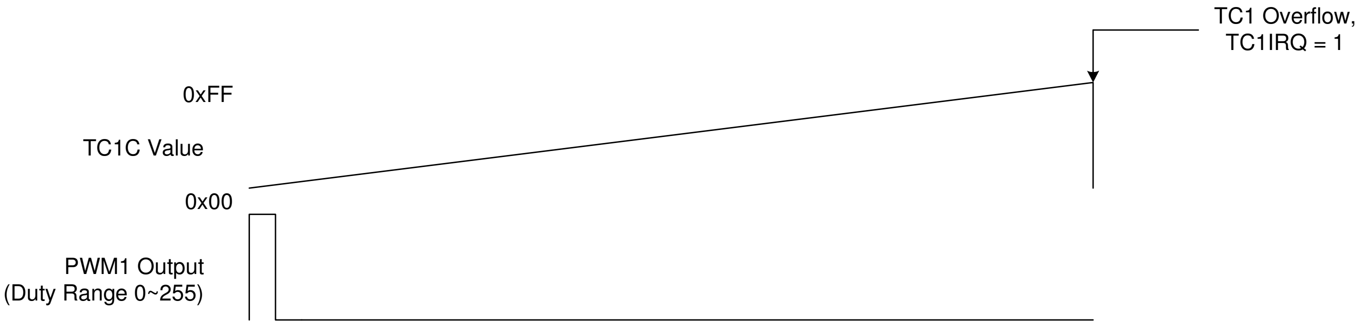
PWM duty range	TC1C valid value	TC1R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count

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.**

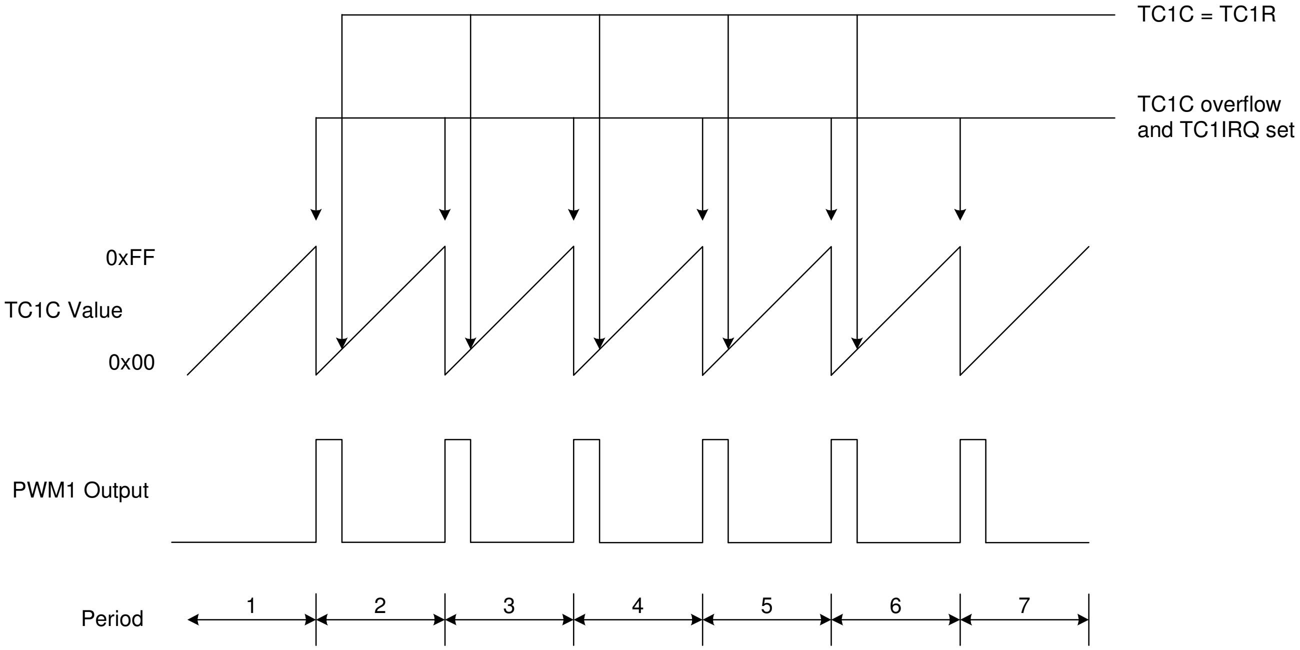
```
MOV         A, #30H             ; Input a number using B0MOV instruction.
B0MOV       TC1R, A

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.



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

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

8.7 BZO Timer

SN8P1989 build in a Buzzer output with controllable frequency change by BZC and BZM register and output in BZO pin.

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BZM	BZOENB	BZORATE2	BZORATE1	BZORATE1	-	BZOX8		
Read/Write	R/W	R/W	R/W	R/W	-	R/W		
After reset	0	0	0	0	-	0		

Bit 2 **BZOX8:** BZO internal clock source control bit.
0 = BZO internal clock source is Fcpu. BZORATE is from Fcpu/2~Fcpu/256.
1 = BZO internal clock source is Fosc. BZORATE is from Fosc/1~Fosc/128.

Bit [6:4] **BZORATE[2:0]:** BZO internal clock select bits.

BZORATE [2:0]	**BZOX8 = 0**	**BZOX8 = 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 **BZOENB:** BZO buzzer-output enable control bit.
0 = Disable BZO buzzer output.
1 = Enable BZO buzzer output

09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BZC	BZC7	BZC6	BZC5	BZC4	BZC3	BZC2	BZC1	BZC0
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

BZC initial value = N - (BZO interval time * input clock)

- **Example: To set 6.25K (160us) Buzzer output in BZO. The BZO counter clock source is Fcpu/2 (BZOX8=0, BZORATE[2:0]=111). High clock is external 4MHz. Fcpu=Fosc/4.**

*BZC initial value = 256 - (Buzzer interrupt interval time * Input clock)*
*= 256 - (160us * 4MHz / 4 / 2/2)*
*= 256 - (160*10*$^{-6}$ ** 4 * 10*^{6} */ 4 / 2/2)*
= 256-40
= 216
= D8H

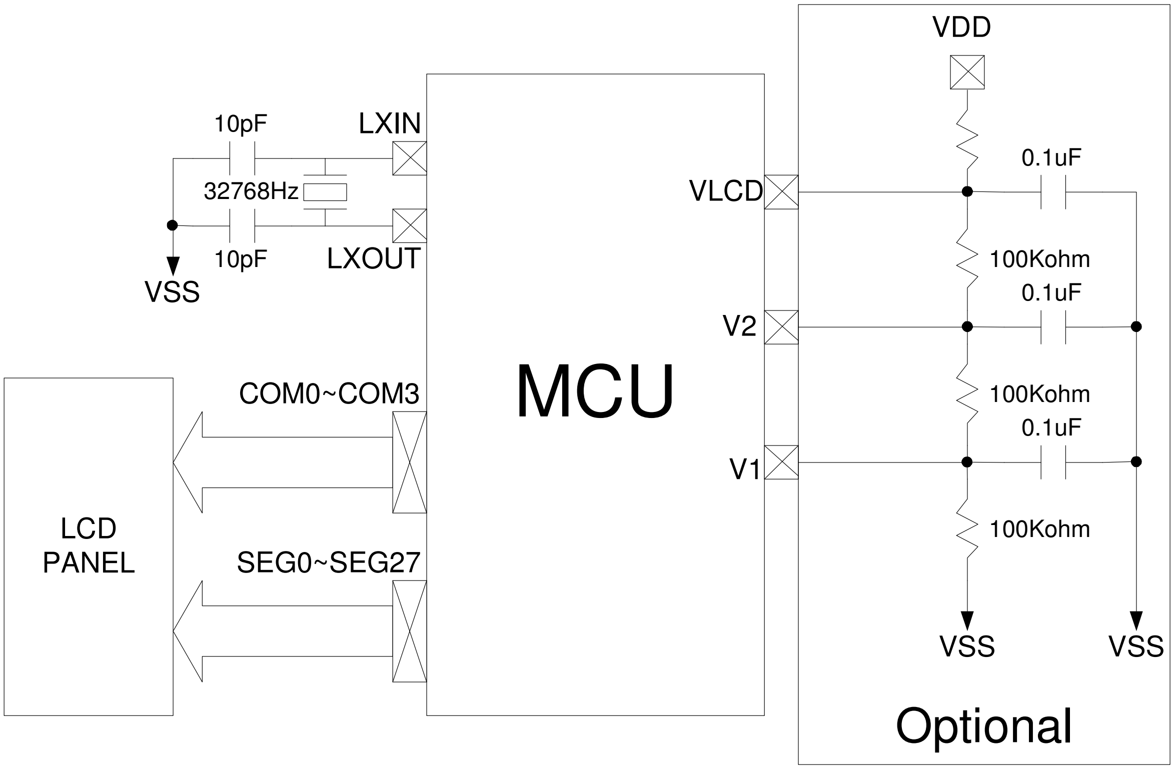
The basic timer table of BZO output clcok.

BZC	BZOX8	BZ0RATE	BZ Clock	BZO output frequency
D8H	0	111	Fcpu/2	6.25K
FBH	0	111	Fcpu/2	50K
FBH	1	101	Fosc/4	100K
FFH	0	111	Fcpu/2	250K

9 LCD DRIVER

9.1 OVERVIEW

There are 4 common pins and 28 segment pins in the SN8P1989. The LCD scan timing is 1/4 duty and 1/3 bias structure to yield 112 dots LCD driver. User can add resistance between VLCD/V2/V1 for more driving current.



Basic LCD Circuit

9.2 LCDM1 REGISTER

LCDM1 register initial value = xx0x 0xxx

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM1	-	-	LCDBNK	-	LCDENB	-	P2HSEG	P2LSEG
R/W	-	-	R/W	-	R/W	-	R/W	R/W
After Reset	-	-	0	-	0	-	0	0

Bit5 **LCDBNK:** LCD blank control bit.
0 = Normal display
1 = All of the LCD dots off.

Bit3 **LCDENB:** LCD driver enable control bit.
0 = Disable LCD function
1 = Enable LCD function

Bit1 P2HSEG**:** SEG20~23 LCD/IO selection bit.
0 = SEG20~23 as LCD function. VLCD1 connect to VLCD
1 = SEG20~23 as IO function (P2.0~P2.3). VLCD1 connect to VDD

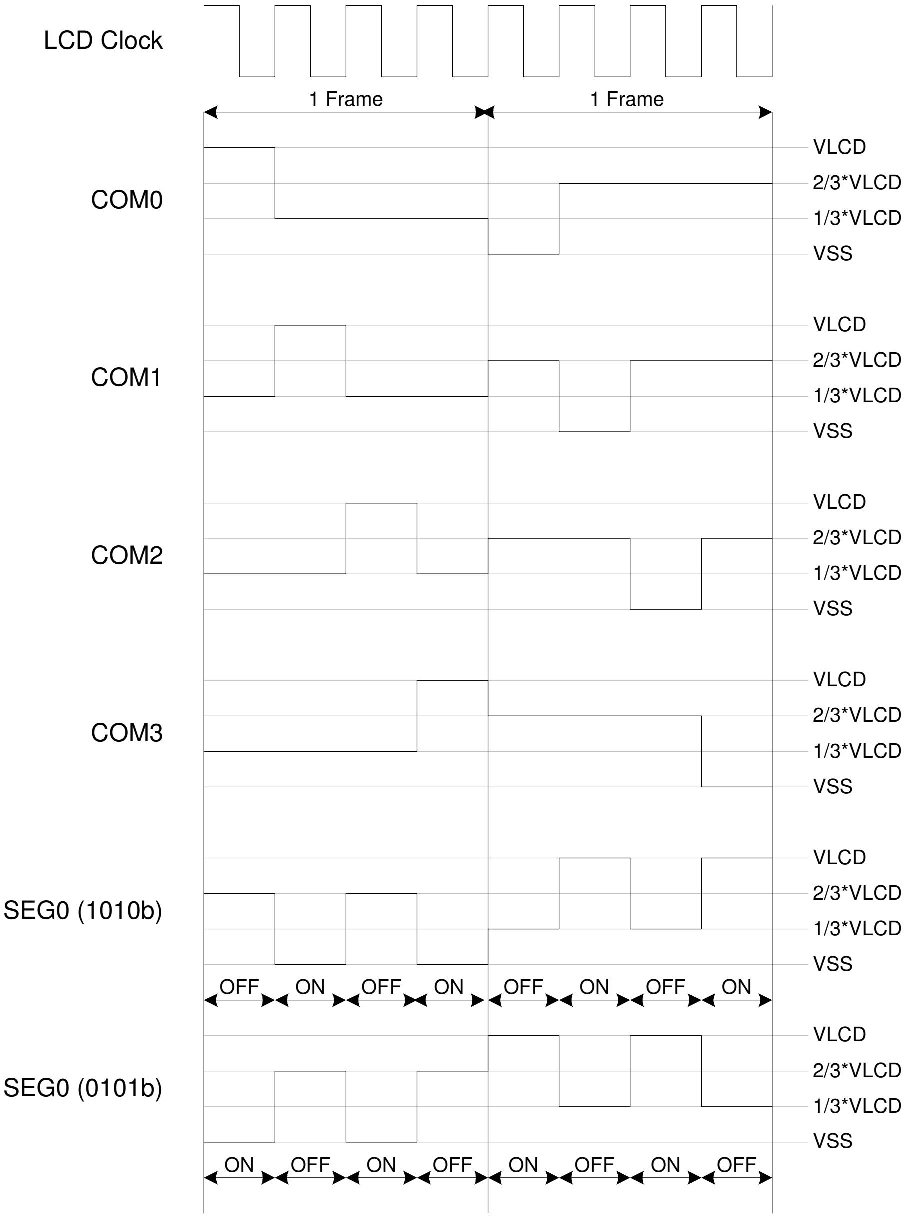
Bit0 P2LSEG**:** SEG24~27 LCD/IO selection bit.
0 = SEG24~27 as LCD function. VLCD1 connect to VLCD
1 = SEG24~27 as IO function (P2.4~P2.7). VLCD2 connect to VDD

9.3 LCD TIMING

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.



LCD Drive Waveform, 1/4 duty, 1/3 bias

9.4 LCD RAM LOCATION

RAM bank 15's address vs. Common/Segment pin location

	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
	COM0	COM1	COM2	COM3	-	-	-	-
SEG 0	00H.0	00H.1	00H.2	00H.3	-	-	-	-
SEG 1	01H.0	01H.1	01H.2	01H.3	-	-	-	-
SEG 2	02H.0	02H.1	02H.2	02H.3	-	-	-	-
SEG 3	03H.0	03H.1	03H.2	03H.3	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
SEG 1C	1CH.0	1CH.1	1CH.2	1CH.3	-	-	-	-

- **Example: Enable LCD function.**

Set the LCD control bit (LCDENB) and program LCD RAM to display LCD panel.

```
		B0BSET		FLCDENB		; LCD driver.
```

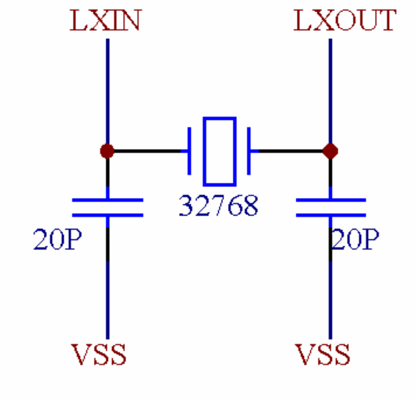
9.5 OPTION REGISTER DESCRIPTION

OPTION initial value = xxxx xxx0

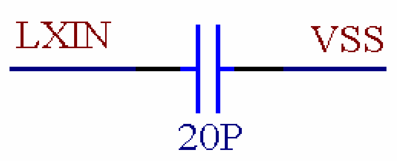
088H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPTION	-	-	-	-	-	-	-	RCLK
R/W	-	-	-	-	-	-	-	R/W
After Reset	-	-	-	-	-	-	-	0

RCLK: External low oscillator type control bit.
0 = Crystal Mode
1 = RC mode.

- ***Note1: Circuit diagram when RCLK=0 –External Low Clock sets as Crystal mode.***



- ➢ ***Note2: Circuit diagram when “RCLK=1” will enable external Low Clock sets as RC mode.***



✸ ***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.***

10 Regulator, PGIA and ADC

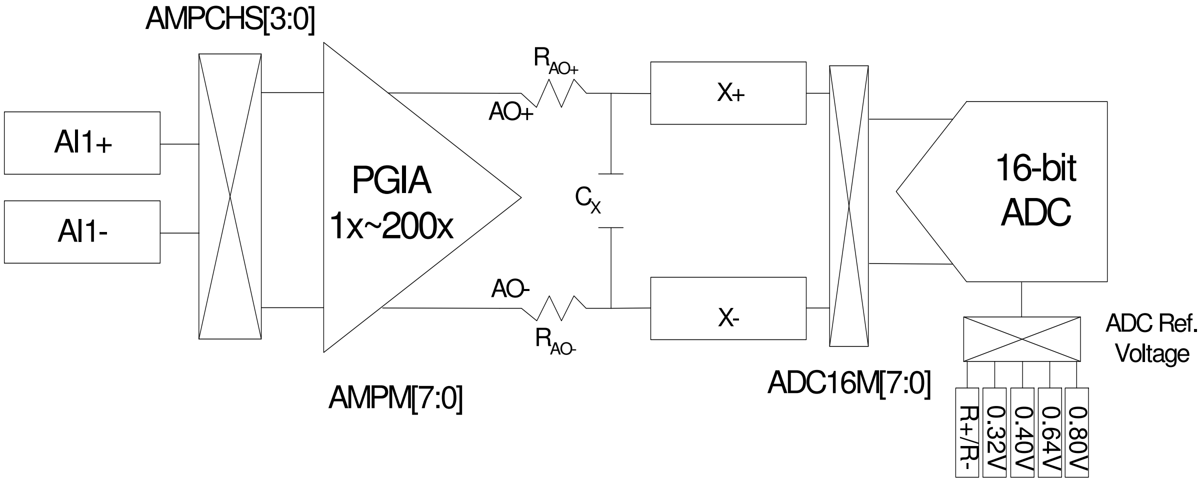
10.1 OVERVIEW

The SN8P1989 has a built-in Voltage Regulator (REG) to support a stable voltage 3.8V from pin AVDDR and 3.0V from pin AVE+ with maximum 10mA current driving capacity. This REG provides stable voltage for internal circuits (PGIA, ADC) and external sensor (load cell or thermistor). The SN8P1989 series also integrated ΔΣAnalog-to-Digital Converters (ADC) to achieve 16-bit performance and up to 62500-step resolution. The ADC has 1 different input channel modes: One fully differential inputs . This ADC is optimized for measuring low-level unipolar or bipolar signals in weight scale and medical applications. A very low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selectable gains of 1x, 12.5x, 50x, 100x, and 200x in the ADC to accommodate these applications.

10.2 ANALOG INPUT

Following diagram illustrates a block diagram of the PGIA and ADC module. The front end consists of a multiplexer for input channel selection, a PGIA (Programmable Gain Instrumentation Amplifier), and the ΔΣ ADC modulator.

To obtain maximum range of ADC output, the ADC maximum input signal voltage V (X+, X-) should be close to but can't over the reference voltage V(R+, R-), Choosing a suitable reference voltage and a suitable gain of PGIA can reach this purpose. The relative control bits are RVS [1:0] bits (Reference Voltage Selection) in ADC16M register and GS[2:0] bits (Gain Selection) in AMPM register.



10.3 Voltage Regulator (REG)

SN8P1989 was built in a REG, which can provide a stable 3.8V (pin AVDDR) and 3.0V/1.5V (pin AVE+) with maximum 10mA current driving capacity. Register REGM can enable or disable REG and controls REG working mode. Because the power of PGIA and ADC is come from AVDDR, turn on AVDDR (AVDDRENB = 1) first before enabling PGIA and ADC. The AVDDR voltage was regulated from VDD. In addition, it will need at least 30ms delay for output voltage stabilization after set REGENB to high.

10.3.1 REGM- Regulator Mode Register

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
REGM	ACMENB	AVDDRENB	AVENB	AVESEL1	AVESEL0	-	-	REGENB
R/W	R/W	R/W	R/W	R/W	R/W	-	-	R/W
After Reset	0	0	0	0	0	-	-	0

Bit0: **REGENB:** Regulator function enable control bit.
0 = Disable Internal Regulator
1 = Enable Internal Regular.

Bit3,4 **AVESEL[1:0]:** AVE+ voltage selection control bit.

AVESEL1	AVESEL0	AVE+ Voltage
1	1	3.0V
1	0	2.4V
0	1	1.5V
0	0	Reserved

Bit5: **AVENB:** AVE+ voltage output control bit.
0 = Disable AVE+ output Voltage
1 = Enable AVE+ output Voltage

Bit6: **AVDDRENB:** Regulator (AVDDR) voltage Enable control bit.
0 = Disable AVDDR Output voltage 3.8V
1 = Enable AVDDR Output voltage 3.8V

Bit7: **ACMENB:** Analog Common Mode (ACM) voltage Enable control bit.
0 = Disable Analog Common Mode and ACM Output voltage 1.2V
1 = Enable Analog Common Mode and ACM Output voltage 1.2V

- ***Note1: 30ms delay is necessary for output voltage stabilization after set REGENB = "1".***
- ***Note2: Before Enable Regulator , Must enable Band Gap Reference (BGRENB=1) first.***
- ***Note3: Before Enable PGIA and ADC , Must enable Band Gap Reference (BGRENB=1), ACM (ACMENB=1) and AVDDR(AVDDRENB).***
- ***Note4: Regulator, PGIA and ADC can work in slow mode, but AMPCKS register value must be reassigned.***

10.4 PGIA -Programmable Gain Instrumentation Amplifier

SN8P1989 includes a low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selection gains of 1x, 12.5x, 50x, 100x, and 200x by register AMPM. The PGIA also provides two types channel selection mode: (1) One fully differential input (2) Two single-ended inputs, it was defined by register AMPCHS.

10.4.1 AMPM- Amplifier Mode Register

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPM	-	BGRENB	FDS1	FDS0	GS2	GS1	GS0	AMPENB
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	-	0	1	1	1	1	1	0

Bit0: **AMPENB:** PGIA function enable control bit.
0 = Disable PGIA function
1 = Enable PGIA function

Bit[3:1]: **GS [2:0]:** PGIA Gain Selection control bit

GS [2:0]	PGIA Gain
000	12.5
001	50
010	100
011	200
100,101,110	Reserved
111	1

✸ ***Note: When selected gain is 1x, PGIA can be disabled (AMPENB=0) for power saving.***

Bit[5:4] **FDS [1:0]:** Chopper Low frequency setting

Note:Set FDS[1:0] = “11” for all applications.

Bit6: **BGRENB:** Band Gap Reference voltage enable control bit.
0 = Disable Band Gap Reference Voltage
1 = Enable Band Gap Reference Voltage

✸ ***Note1: Band Gap Reference voltage must be enable (FBRGENB), before following function accessing***

1. **Regulator.**
2. **PGIA function.**
3. **16- bit ADC function.**
4. **Low Battery Detect function**

✸ ***Note2: PGIA can’t work in slow mode, unless gain selection is 1x.***

10.4.2 AMPCKS- PGIA CLOCK SELECTION

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPCKS	-	-	-	-	-	AMPCKS2	AMPCKS1	AMPCKS0
R/W	-	-	-	-	-	W	W	W
After Reset	-	-	-	-	-	0	0	0

Bit[2:0] **AMPCKS [2:0]** register sets the PGIA Chopper working clock. The suggestion Chopper clock is 1.95K Hz.@ 4MHz, 1.74K @ 3.58MHz.

PGIA Clock= Fcpu / 32 / (2^AMPCKS)

Refer to the following table for AMPCKS [2:0] register value setting in different Fosc frequency.

AMPCKS2	AMCKS1	AMPCKS0	High Clock			
			2M	3.58M	4M	8M
0	0	0	15.625K	27.968K	31.25K	62.5K
0	0	1	7.8125K	13.98K	15.625K	31.25K
0	1	0	3.90625K	6.99K	7.8125K	15.625K
0	1	1	1.953125K	3.49K	3.90625K	7.8125K
1	0	0	976Hz	1.748K	1.953125K	3.90625K
1	0	1	488Hz	874Hz	976Hz	1.953125K
1	1	0	244Hz	437Hz	488Hz	976Hz
1	1	1	122Hz	218Hz	244Hz	488Hz

✸ ***Note: In general application, set PGIA Chopper working clock is ~2K Hz, but set clock to 250Hz when High clock is 32768 crystal or in external Low clock mode.***

10.4.3 AMPCHS-PGIA CHANNEL SELECTION

091H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPCHS	-	-	-	-	ADC16CHS3	ADC16CHS2	ADC16CHS1	ADC16CHS0
R/W	-	-	-	-	R/W	R/W	R/W	R/W
After Reset	-	-	-	-	0	0	0	0

ADC16CHS0: PGIA Channel Selection

ADC16CHS [3:0]	Selected Channel	V (X+, X-) Output	Input-Signal Type
0000	AI1+, AI1-	V (AI1+, AI1-) × PGIA Gain	Differential
0110	ACM, ACM	V (ACM, ACM) × PGIA Gain	Input-Short
Others	Reserved	-	-

- ✸ ***Note 1: V (AI+, AI-) = (AI+ voltage - AI- voltage)***
- ✸ ***Note 2: The purpose of Input-Short mode is only for PGIA offset testing.***
- ✸ ***Note 3: When REG is Disable or system in stop mode, signal on analog input pins must be Zero ("0"V, including AI+, AI-, X+, X-, R+ and R-) or it will cause the current consumption from these pins.***

10.5 16-Bit ADC

10.5.1 ADC16M- ADC Mode Register

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADC16M	-	-	-	-	IRVS	RVS1	RVS0	ADC16ENB
	-	-	-	-	R/W	R/W	R/W	R/W
				-	0	0	0	0

Bit0: **ADC16ENB:** ADC function control bit:
0 = Disable 16-bit ADC,
1 = Enable 16-bit ADC

Bit[2:1]: **RVS [1:0]**: ADC Reference Voltage Selection bit 1
00 = Selection ADC Reference voltage from **External** reference R+,R-.
10 = Selection ADC Reference voltage from **Internal** reference

Bit3: **IRVS:** Internal Reference Voltage Selection.
0 = Internal Reference Voltage V(REF+,REF-) is AVE+/0.133 (When AVE+=3.0V, V(REF+,REF-)=0.4V)
1 = Internal Reference Voltage V(REF+,REF-) is AVE+/0.266 (When AVE+=3.0V, V(REF+,REF-)=0.8V)

Bit4: Always Set to "0"

IRVS	RVS1	RVS0	AVESEL[1:0]	AD Reference Voltage		AD Channel Input		Note
				REF+	REF-	ADCIN+	ADCIN-	
X	0	0	-	R+	R-	X+	X-	External Ref. Voltage
0	1	0	11 (AVE+=3.0V)	0.8V	0.4V			**V (X+, X-) < 0.4V**
0	1	0	10 (AVE+=2.4V)	0.64V	0.32V			**V (X+, X-) < 0.32V**
1	1	0	11 (AVE+=3.0V)	1.2V	0.4V			**V (X+, X-) < 0.8V**
1	1	0	10 (AVE+=2.4V)	0.96V	0.32V			**V (X+, X-) < 0.64V**
1	1	0	01 (AVE+=1.5V)	0.6V	0.2V			**V (X+, X-) < 0.4V**

✸ ***Note1: The ADC conversion data is combined with ADCDH and ADCDL register in 2's compliment with sign bit numerical format, and Bit ADCB15 is the sign bit of ADC data. Refer to following formula to calculate ADC conversion data value.***

✸ ***Note2: The Internal Reference Voltage is divided from AVE+, so the voltage will follow the changing with AVE+(3.0V/2.4V/1.5V) which selected by AVESEL[1:0].***

✸ ***Note1: The ADC conversion data is combined with ADCDH and ADCDL register in 2's compliment with sign bit numerical format, and Bit ADCB15 is the sign bit of ADC data. Refer to following formula to calculate ADC conversion data value.***

$$(ADCIN+) > (ADCIN-) \Rightarrow ADCConversionData = +\frac{(ADCIN+)-(ADCIN-)}{(REF+)-(REF-)} X\,31250$$

$$(ADCIN+) < (ADCIN-) \Rightarrow ADCConversionData = -\frac{(ADCIN+)-(ADCIN-)}{(REF+)-(REF-)} X\,31250$$

10.5.2 ADCKS- ADC Clock Register

094H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCKS	ADCKS7	ADCKS6	ADCKS5	ADCKS4	ADCKS3	ADCKS2	ADCKS1	ADCKS0
	W	W	W	W	W	W	W	W
After Reset	0	0	0	0	0	0	0	0

ADCKS [7:0] register sets the ADC working clock, the suggestion ADC clock is 100K Hz.

Refer the following table for ADCKS [7:0] register value setting in different Fosc frequency.

ADC Clock= (Fosc / (256-ADCKS [7:0]))/2

ADCKS [7:0]	F_{OSC}	ADC Working Clock
246	4M	(4M / 10)/2 = 200K
236	4M	(4M / 20)/2 = 100K
243	4M	(4M / 13)/2 = 154K
231	4M	(4M / 25)/2 = 80K

ADCKS [7:0]	F_{OSC}	ADC Working Clock
236	8M	(8M / 20)/2 = 200K
216	8M	(8M / 40)/2 = 100K
231	8M	(8M / 25)/2 = 160K
206	8M	(8M / 50)/2 = 80K

➢ ***Note: In general application, ADC working clock is 100K Hz.***

10.5.3 ADCDL- ADC Low-Byte Data Register

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCDL	ADCB7	ADCB6	ADCB5	ADCB4	ADCB3	ADCB2	ADCB1	ADCB0
	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

10.5.4 ADCDH- ADC High-Byte Data Register

099H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCDH	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB8	ADCB9
	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

ADCDL [7:0]: Output low byte data of ADC conversion word.
ADCDH [7:0]: Output high byte data of ADC conversion word.

.

- ➢ ***Note1: ADCDL [7:0] and ADCDH [7:0] are both read only registers.***
- ➢ ***Note2: The ADC conversion data is combined with ADCDH, ADCDL in 2's compliment with sign bit numerical format, and Bit ADCB15 is the sign bit of ADC data.***
  ADCB15=0 means data is Positive value, ADCB15=1 means data is Negative value.
- ➢ ***Note3: The Positive Full-Scale-Output value of ADC conversion is 0x7A12.***
- ➢ ***Note4: The Negative Full-Scale-Output value of ADC conversion is 0x85EE,***

ADC conversion data (2's compliment, Hexadecimal)	***Decimal Value***
0x7A12	31250
…	…
0x4000	16384
…	…
0x1000	4096
…	…
0x0002	2
0x0001	1
0x0000	0
0xFFFF	-1
0xFFFE	-2
…	…
0xF000	-4096
…	…
0xC000	-16384
…	…
0x85EE	-31250

10.5.5 DFM-ADC Digital Filter Mode Register

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DFM	-	-	-	-	-	WRS0	-	DRDY
	-	-	-	-	-	R/W	-	R/W

Bit0: **DRDY:** ADC Data Ready Bit.
1 = ADC output (update) new conversion data to ADCDH, ADCDL.
0 = ADCDH, ADCDL conversion data are not ready.

Bit2: **WRS [1:0]:** ADC output Word Rate Selection:

WRS0	**Output Word Rate**		
	ADC clock = 200K	**ADC clock = 100K**	**ADC clock = 80K**
0	50Hz	25 Hz	20 Hz
1	25Hz	12.5 Hz	10 Hz

- ✷ ***Note 1: AC power 50 Hz noise will be filter out when output word rate = 25Hz***
- ✷ ***Note 2: AC power 60 Hz noise will be filter out when output word rate = 20Hz***
- ✷ ***Note 3: Both AC power 50 Hz and 60 Hz noise will be filter out when output word rate = 10Hz***
- ✷ ***Note 4: Clear Bit DRDY after got ADC data or this bit will keep High all the time.***
- ✷ ***Note 5: Adjust ADC clock (ADCKS) and bit WRS0 can get suitable ADC output word rate.***

10.5.6 Analog Setting and Application

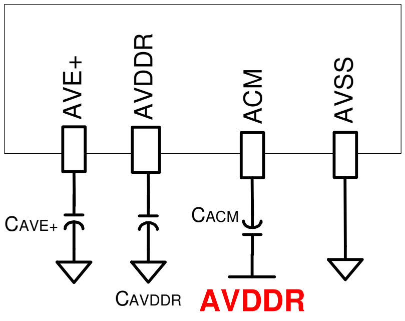
The most applications of SN8P1989 were for DC measurement In different applications had each Analog capacitor setting to avoid VDD drop when Charge pump enable or can save cost. Following table indicate different applications setting which MCU power source came from, AA/AAA dry battery or external Regulator

Resistance and Capacitor Table:

Power type	AI+	AI-	X+/X-	AO+ /AO-	R+/R-	ACM	AVDDR	AVE+	AVDD (pin17)	VDD (pin21/38)
	C_{AI+}	C_{AI-}	C_X	$R_{AO+/}$ R_{AO-}	C_R	C_{ACM}	C_{AVDDR}	C_{AVE+}	C_{AVDD}	C_{DVDD}
AA/AAA Bat.(4.4~5.5V)	0.1uF	0.1uF	0.1uF	100K	0.1uF	1uF	1uF	4.7uF	10uF	0.1uF/2.2uF
External 5V Reg.	0.1uF	0.1uF	0.1uF	100K	0.1uF	1uF	1uF	4.7uF	10uF	0.1uF/2.2uF

✸ ***Note: The positive note of C_{ACM} connect to AVDDR and Negative note connect to ACM***

VDD=4.2V~5.5V Analog Capacitor Connection



Delay Time:

Power Type	Enable ACM	Enable AVDDR	Enable AVE+
AA/AAA Bat.(4.4~6V)	5ms	50ms	50ms
External 5V Reg.	5ms	50ms	50ms

10.5.7 LBTM : Low Battery Detect Register

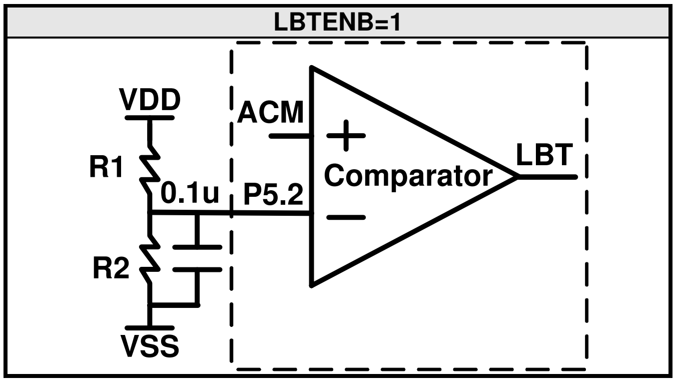
SN8P1989 provided two different way to measure Power Voltage. One is from ADC reference voltage selection. It will be more precise but take more time and a little bit complex. The another way is using build in Voltage Comparator, divide power voltage and connect to P5.2, bit LBTO will output the P5.2 voltage Higher or Lower than ACM(1.2V)

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LBTM	-	-	-	-	-	LBTO	-	LBTENB
R/W	-	-	-	-	-	R	-	R/W
After Reset	-	-	-	-	-	0	-	0

Bit0 **LBTENB:** Low Battery Detect mode control Bit.
0 = Disable Low Battery Detect function,
1 = Enable Low Battery Detect function

Bit2: **LBTO:** Low Battery Detect Output Bit.
0 = P52/LBT voltage Higher than ACM (1.2V)
1 = P52/LBT voltage Lower than ACM (1.2V)

The LBT circuit will leak a small current in power down mode . These two circuit is following:



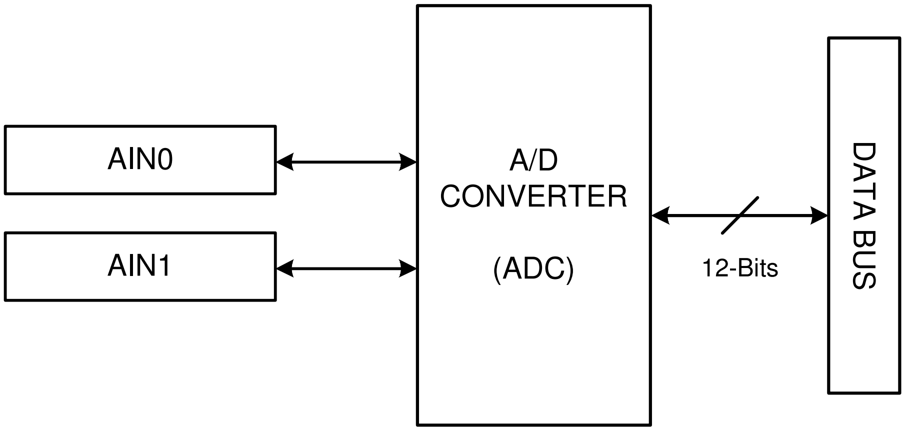
Low Battery Voltage	R1	R2	LBTO=1
4.2V	2.5MΩ	1MΩ	VDD<4.2V
4.8V	1.5MΩ	0.5MΩ	VDD<4.8V

✸ ***Note: Get LBTO=1 more 10 times in a raw every certain period, ex. 20 ms or more to make sure the Low Battery signal is stable.***

11 2 CHANNEL ANALOG TO DIGITAL CONVERTER

11.1 OVERVIEW

This analog to digital converter has 2-input sources with up to 4096-step resolution to transfer analog signal into 12-bits digital data. The sequence of ADC operation is to select input source (AIN0 ~ AIN1) at first, then set GCHS and ADS bit to “1” to start conversion. When the conversion is complete, the ADC circuit will set EOC bit to “1” and final value output in ADB register. This ADC circuit can select between 8-bit and 12-bit resolution operation by programming ADLEN bit in ADR register.



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

✸ ***Note: ADC programming notice:***

1. Disable ADC before enter power down (sleep) mode to save power consumption.

2. Delay 100uS after enable ADC (set ADENB = “1”) to wait ADC circuit ready for conversion.

3. Disable ADC (set ADENB = “0”) before enter sleep mode to save power consumption.

11.2 ADC12M REGISTER

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADC12M	ADC12ENB	ADS	EOC	GCHS	-	ADC12CHS2	ADC12CHS1	ADC12CHS0
Read/Write	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
After reset	0	0	0	0	-	0	0	0

Bit 7 **ADC12ENB:** 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] **ADC12CHS[2:0]:** ADC input channels select bit.

ADC12CHS [2:0]	Selected Channel
000	AIN0
001	AIN1
Others	Reserved

11.3 ADR REGISTERS

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	-	AD12CKS1	-	AD12CKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	-	R/W	-	R/W	R	R	R	R
After reset	-	0	-	0	-	-	-	-

Bit 6,4 **AD12CKS [1:0]:** ADC's clock source select bit.

AD12CKS1	AD12CKS0	ADC Clock Source	ADC Clock Source
0	0	Fcpu/4	Both validate in Normal mode and Slow mode
0	1	Fcpu/2	Both validate in Normal mode and Slow mode
1	0	Fhosc	Only validate in Normal mode
1	1	Fhosc/2	Only validate in Normal mode

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

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

11.4 ADB REGISTERS

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

Bit[7:0] **ADB[11:4]:** 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 12-bit ADC mode, the ADC data is stored in ADB and ADR registers.

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

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

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

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

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

11.5 ADC CONVERTING TIME

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

Fcpu = 1MHz (High clock, Fosc is 4MHz and Fcpu = Fosc/4)

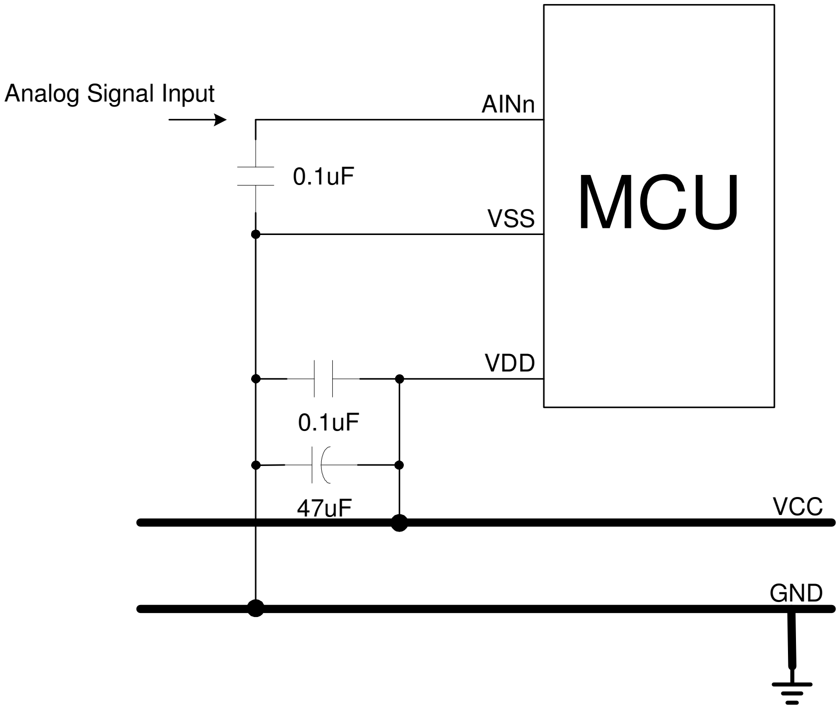
ADLEN	ADCKS1	ADCKS0	ADC Clock	ADC conversion time
1 (12-bit)	0	0	Fcpu/16	1/(1MHz/16/4)*16 = 1024 us
	0	1	Fcpu/8	1/(1MHz/8/4)*16 = 512 us
	1	0	Fcpu	1/(1MHz/4)*16 = 64 us
	1	1	Fcpu/2	1/(1MHz/2/4)*16 = 128 us

11.6 ADC ROUTINE EXAMPLE

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

```
ADC0:
             B0BSET      FADENB          ; Enable ADC circuit
             CALL        Delay100uS      ; Delay 100uS to wait ADC circuit ready for conversion
             MOV         A, #60H
             B0MOV       ADR, A          ; To set 12-bit ADC and ADC clock = Fosc.
             MOV         A,#90H
             B0MOV       ADC12M,A        ; To enable ADC and set AIN0 input
             B0BSET      FADS            ; To start conversion
WADC0:
             B0BTS1      FEOC            ; To skip, if end of converting =1
             JMP         WADC0           ; else, jump to WADC0
             B0MOV       A,ADB           ; To get AIN0 input data bit11 ~ bit4
             B0MOV       Adc_Buf_Hi, A
             B0MOV       A,ADR           ; To get AIN0 input data bit3 ~ bit0
             AND         A, 0Fh
             B0MOV       Adc_Buf_Low, A
Power_Down   .           .
             B0BCLR      FADENB          ; Disable ADC circuit
             B0BCLR      FCPUM1
             B0BSET      FCPUM0          ; Enter sleep mode
```

11.7 ADC CIRCUIT



ADC reference high voltage is from VDD pin.

12 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 (bnak 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 for Working registers R, Y, Z , RBANK & PFLAG)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1
	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
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
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1
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
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
C	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
	DAA		To adjust ACC's data format from HEX to DEC.	√	-	-	1
	MUL	A,M	R, A ← A * M, The LB of product stored in Acc and HB stored in R register. ZF affected by Acc.	-	-	√	2
	AND	A,M	A ← A and M	-	-	√	1
L	AND	M,A	M ← A and M	-	-	√	1
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
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
	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
R	RRC	M	A ← RRC M	√	-	-	1
O	RRCM	M	M ← RRC M	√	-	-	1
C	RLC	M	A ← RLC M	√	-	-	1
E	RLCM	M	M ← RLC M	√	-	-	1
S	CLR	M	M ← 0	-	-	-	1
S	BCLR	M.b	M.b ← 0	-	-	-	1
	BSET	M.b	M.b ← 1	-	-	-	1
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1
	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 + 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 + 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 working registers (080H~087H) into buffers	-	-	-	1
C	POP	To pop working registers (080H~087H) from buffers	√	√	√	1
	NOP	No operation	-	-	-	1

***Note:** 1. Processing OSCM register needs to add extra one cycle.*

2. If branch condition is true then "S = 1", otherwise "S = 0".

13 ELECTRICAL CHARACTERISTIC

13.1 ABSOLUTE MAXIMUM RATING

Supply voltage (V_{DD})……………………………………………….……………………………………	- 0.3V ~ 6.0V
Input in voltage (V_{IN})……………………………...……………….………………………	V_{SS} - 0.2V ~ V_{DD} + 0.2V
Operating ambient temperature (T_{OPR})…………………………………………………	0°C ~ + 70°C
Storage ambient temperature (T_{STOR})…………………..………..…………………………	–40°C ~ + 125°C

13.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to V_{SS}, V_{DD} = 5.0V,F_{OSC} = 4MHz, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	V_{DD}	Normal mode, V_{PP} = V_{DD}		4.2	5.0	5.5	V
RAM Data Retention voltage	V_{DR}			1.5		-	V
V_{DD} rise rate	V_{POR}	V_{DD} rise rate to ensure 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	I_{LEKG}	V_{IN} = V_{DD}		-	-	2	uA
I/O port pull-up resistor	R_{UP}	V_{IN} = V_{SS}, V_{DD} = 3V		100	200	300	KΩ
		V_{IN} = V_{SS}, V_{DD} = 5V		50	100	180	KΩ
I/O port input leakage current	I_{LEKG}	Pull-up resistor disable, V_{IN} = V_{DD}		-	-	2	uA
I/O output source current	I_OH	V_{OP} = V_{DD} - 0.5V		9	-	-	mA
sink current	I_OL	V_{OP} = V_{SS} + 0.5V		10	-	-	mA
INT_N trigger pulse width	$T_{INT}0$	INT0 ~ INT1 interrupt request pulse width		$2/F_{CPU}$	-	-	Cycle
AVREFH input voltage	Varfh	Vdd = 5.0V		2V	-	Vdd	V
AIN0 ~ AIN1 input voltage	Vani	Vdd = 5.0V		0	-	Varfh	V
12-bit ADC current consumption	I_{ADC}	Vdd=5.0V		-	0.6*	-	mA
12-bit ADC enable time	Tast	Ready to start convert after set ADENB = “1”		100	-	-	uS
12-bit ADC Clock Frequency	F_{ADCLK}	VDD=5.0V		32K		8M	Hz
12-bit ADC Conversion Cycle Time	F_{ADCYL}	VDD=2.4V~5.5V		64			$1/F_{ADCLK}$
12-bit ADC Sampling Rate (Set FADS=1 Frequency)	F_{ADSMP}	VDD=5.0V				125	K/sec
12-bit ADC Differential Nonlinearity	DNL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K		±1	±2	±4	LSB
12-bit ADC Integral Nonlinearity	INL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K		±2	±4	±8	LSB
12-bit ADC No Missing Code	NMC	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K		10			Bits
Supply Current	Idd1	normal Mode (Analog Parts OFF)	Vdd= 5V 4Mhz crystal	-	1.5	3	mA
	Idd4	normal Mode (Analog Parts ON)	Vdd= 5V 4Mhz crystal	-	2.5	4	mA
	Idd9	Slow mode (Stop High Clock, LCD OFF, REG OFF)	Vdd= 5V ILRC 32Khz	-	10	20	uA

	Idd10	Slow mode (Stop High Clock, LCD ON, REG OFF)	Vdd= 5V ILRC 32Khz	-	25	50	uA
	Idd11	Slow mode (Stop High Clock, LCD ON, REG ON)	Vdd= 5V ILRC 32Khz	-	300	600	uA
	Idd12	Sleep mode	Vdd= 5V	-	1	2	uA
			Vdd= 3V	-	0.7	1.5	uA
LVD detect level	V_{LVD}	Internal POR detect level		2.1	2.4	2.6	V

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

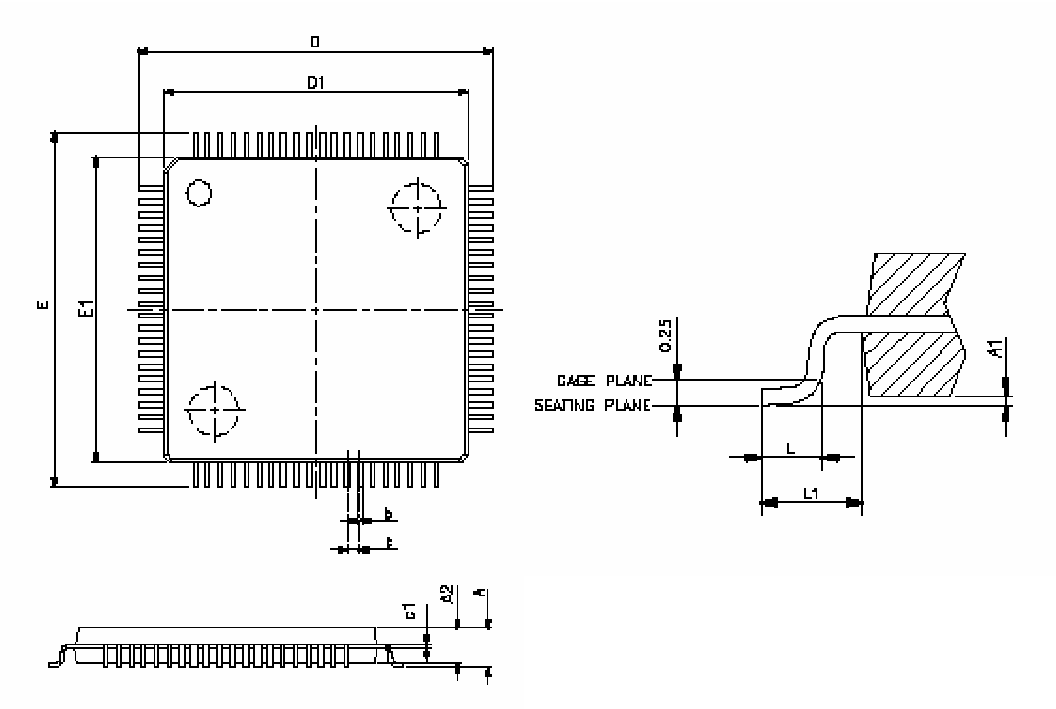
➢ **Note: Analog Parts including Regulator (REG), PGIA and ADC.**

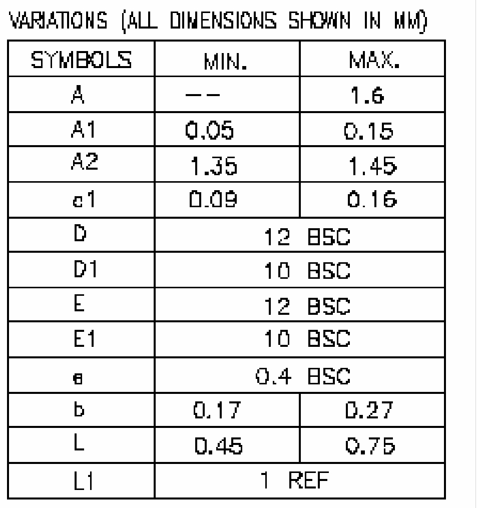
(All of voltages refer to Vdd=3.8V F_{OSC} = 4MHz, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
16-Bit Analog to Digital Converter						
Operating current	I_{DD_ADC}	Run mode @ 3.8V		800	1000	uA
Power down current	I_{PDN}	Stop mode @ 3.8V		0.1	1	μA
Conversion rate	F_{SMP}	ADCKS: 200KHz			25	sps
Reference Voltage Input Voltage	Vref	R+, R- Input Voltage	0.4		2.0	V
Differential non-linearity	DNL			±0.5	±0.5	LSB
Integral non-linearity	INL			±1	±4	LSB
No missing code	NMC		16			bit
Noise free code	NFC			14	16	bit
Effective number of bits	ENOB			14	16	bit
ADC Input range	V_{AIN}		0.4		2.0	V
PGIA						
Current consumption	I_{DD_PGIA}	Run mode @ 3.8V		300	500	uA
Power down current	I_{PDN}	Stop mode @ 3.8V			0.1	μA
Input offset voltage	Vos				2	uV
Bandwidth	BW				100	Hz
PGIA Gain Range (Gain=200x)	GR	VDD = 3.8V	180	200	250	
PGIA Input Range	Vopin	VDD = 3.8V	0.4		2	V
PGIA Output Range	Vopout	VDD = 3.8V	0.4		2	V
Band gap Reference (Refer to ACM)						
Band gap Reference Voltage	V_{BG}		1.150	1.220	1.250	V
Reference Voltage Temperature Coefficient	T_{ACM}			50*		PPM/℃
Operating current	I_{BG}	Run mode @ 3.8V		50	100	uA
Regulator						
Supply voltage	V_{CPS}	Normal mode	2.4		5.5	V
Regulator output voltage AVDDR	V_{CPO1}		3.70	3.8	3.95	V
Regulator output voltage AVE+	V_{CPO2}		2.95	3.1	3.15	V
Analog common voltage	V_{ACM}		1.19	1.22	1.25	V
Regulator output current capacity	I_{VA+}		10			mA
Quiescent current	I_{QI}		-	700	1400	uA
V_{ACM} driving capacity	I_{SRC}		-	-	10	μA
V_{ACM} sinking capacity	I_{SNK}		-	-	1	mA

14 PACKAGE INFORMATION

14.1 LQFP80





VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

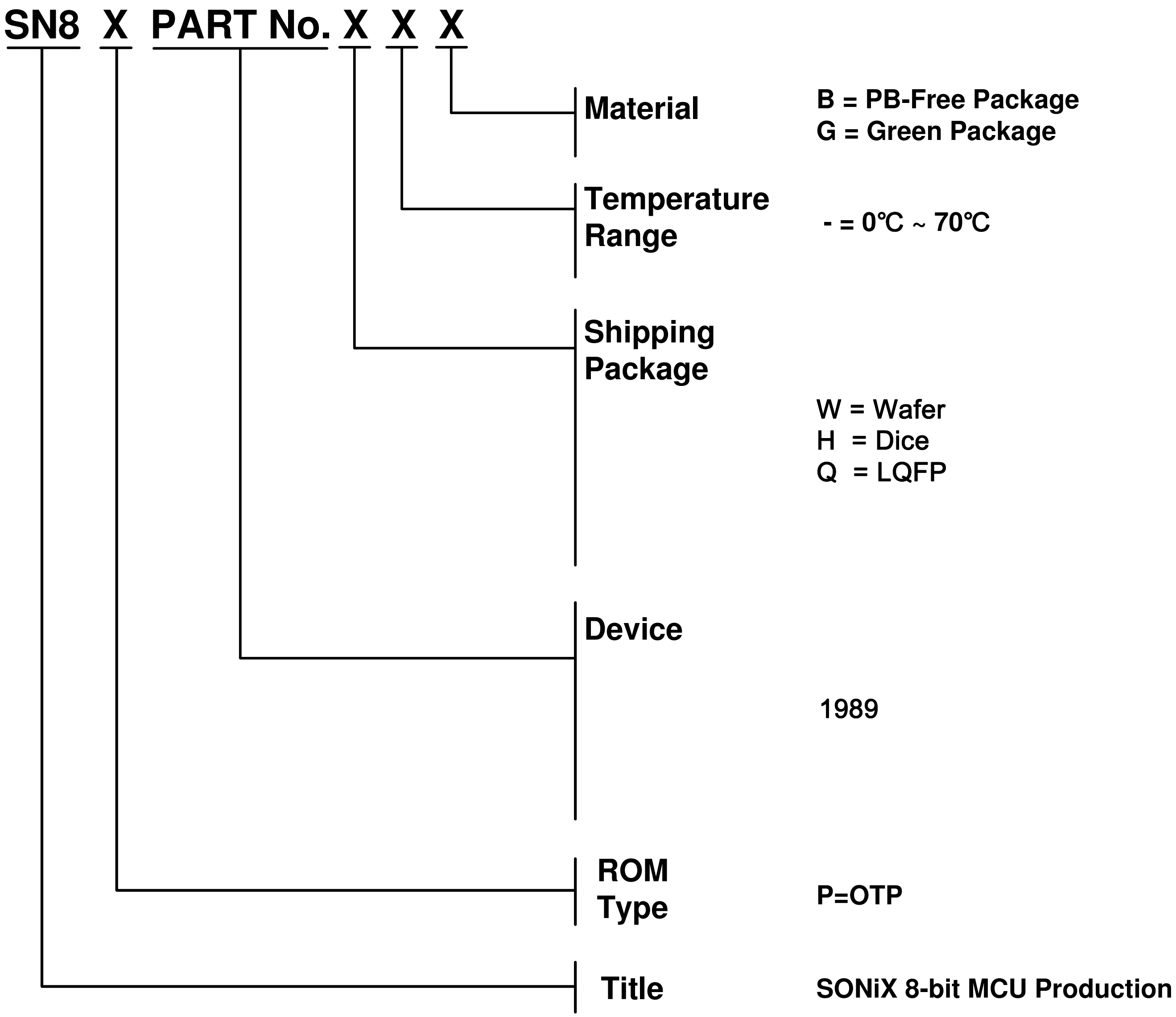
SYMBOLS	MIN.	MAX.
A	--	1.6
A1	0.05	0.15
A2	1.35	1.45
c1	0.09	0.16
D	12 BSC	
D1	10 BSC	
E	12 BSC	
E1	10 BSC	
e	0.4 BSC	
b	0.17	0.27
L	0.45	0.75
L1	1 REF	

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



SN8 X PART No. X X X

Material	B = PB-Free Package
	G = Green Package

Temperature Range	- = 0℃ ~ 70℃

Shipping Package	W = Wafer
	H = Dice
	Q = LQFP

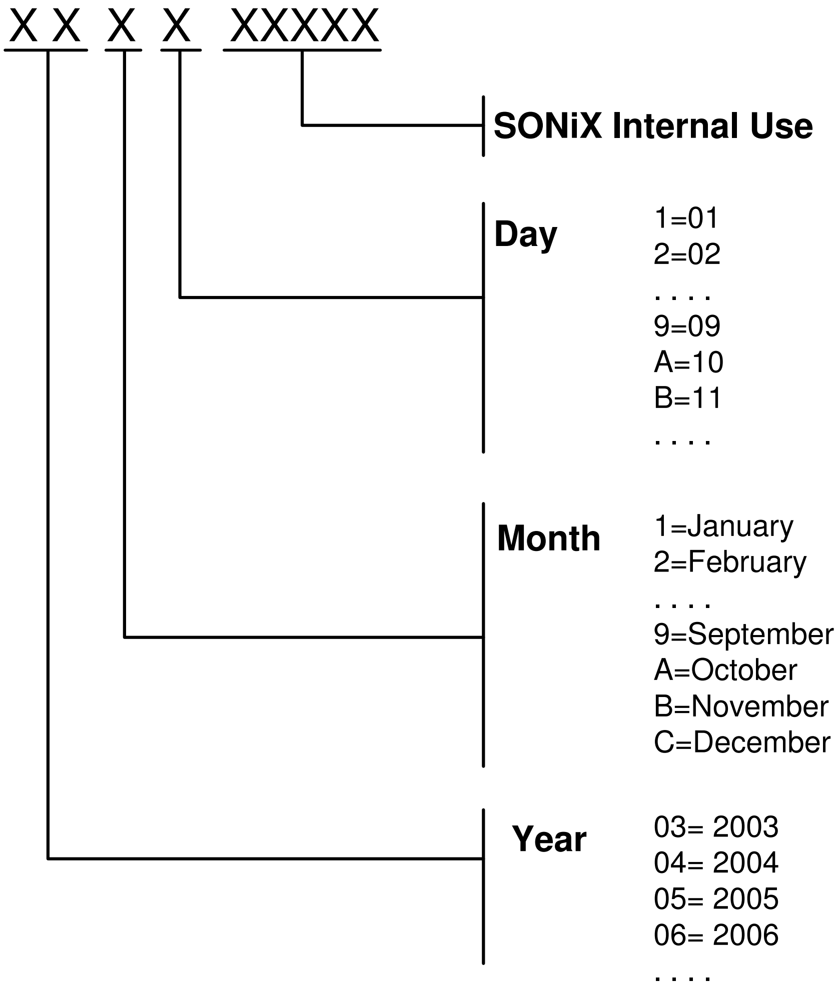
Device	1989
ROM Type	P=OTP
Title	SONiX 8-bit MCU Production

15.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P1989QG	OTP	1989	LQFP	0℃~70℃	Green Package
SN8P1989QB	OTP	1989	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:



SONIX reserves the right to make change without further notice to any products herein to improve reliability, function or design. SONIX does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. SONIX products are not designed, intended, or authorized for us as components in systems intended, for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SONIX product could create a situation where personal injury or death may occur. Should Buyer purchase or use SONIX products for any such unintended or unauthorized application. Buyer shall indemnify and hold SONIX and its officers , employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that SONIX was negligent regarding the design or manufacture of the part.

Main Office:
Address: 10F-1, No.36, Taiyuan Street, Chupei City, Hsinchu, Taiwan.
Tel: 886-3-5600 888
Fax: 886-3-5600 889

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

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

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