SN8P1929

USER'S MANUAL

Preliminary Specification Version 0.5

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 0.1	July 2008	First issue.
VER 0.2	July 2008	Update pin assignment in LQFP package.
VER 0.3	Sep. 2008	Cancel LQFP64 package.
VER 0.4	Oct. 2008	1. Modified VACM driving capacity Spec. 2. Modified IHRC frequency Spec. 3. Annotation of WTCKS function. 4. Error correction of NT0 and NTP. 5. Cancel EZ writer programming. 6. Cancel OTP programming pin to transition board mapping. 7. Modified Development tools chapter.
VER 0.5	Dec. 2008	1. Modified AVE+, ACM, AVDDR Spec. 2. Modified Marking Identification “F = LQFP” 3. Modified “LCD Drive Waveform, 1/4 duty, 1/3 bias and 1/2 bias”. 4. Modified P41 and P42 description in “Low Battery Detect Register”. 5. Modified “Low Battery Detect Register” 6. Modified Reset flag NT0:NPD = [11] when LVD and Power On Reset occurring.

Table of Content

1 PRODUCT OVERVIEW

1.1 SELECTION TABLE

CHIP	ROM	RAM	Stack	LCD	Timer			I/O	ADC (Bit)	PWM Buzzer	SIO	Wakeup Pin no.	Package
					T0	TC0	TC1						
SN8P1908	8K*16	512*8	8	4*24	V	V	V	17	16	2	-	7	LQFP64
SN8P1909	8K*16	512*8	8	4*32	V	V	V	20	16	2	1	7	LQFP80
SN8P1919	6K*16	256*8	8	4*32	V	V	V	22	16	2	-	7	LQFP80
SN8P1929	4K*16	256*8	8	4*24	V	V	V	16	16	2	-	6	LQFP80

Table 1-1 Selection table of SN8P1929

1.2 FEATURES

- ***Memory configuration***
  OTP ROM size: 4K * 16 bits
  RAM size: 256 * 8 bits (bank 0, bank 1)
  8-levels stack buffer
  LCD RAM size: 4*24 bits
- ***I/O pin configuration***
  Input only: P0
  Bi-directional: P1, P2, P4, P5
  Wakeup: P0, P1
  Pull-up resisters: P0, P1, P2, P4, 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**
  Three ADC channel configurations:
  Two fully differential channel
  One differential and Two single-ended channels
  Four single-ended channels
- ***Five interrupt sources***
  Three internal interrupts: T0, TC0, TC1
  Two external interrupts: INT0, INT1
- ***Single power supply: 2.4V ~5.5V***
- ***On-chip watchdog timer***
- ***On-chip charge-pump 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 ADC reference voltage V(R+,R-)=0.8V , 0.64V or 0.4V.***
- ***Build In Temperature Sensor.***
- ***LCD driver:***
  1/3 or 1/2 bias voltage.
  4 common * 24 segment
- ***Dual clock system offers four operating modes***
  Internal high clock: RC type up to 16 MHz
  External high clock: Crystal type up to 8 MHz
  Normal mode: Both high and low clock active.
  Slow mode: External Low clock and Internal low RC clock.
  Green mode: Period wake up by T0 and TC0
  Sleep mode: Both high and low clock stop.
- ***Package***
  LQFP80/Dice

1.3 SYSTEM BLOCK DIAGRAM

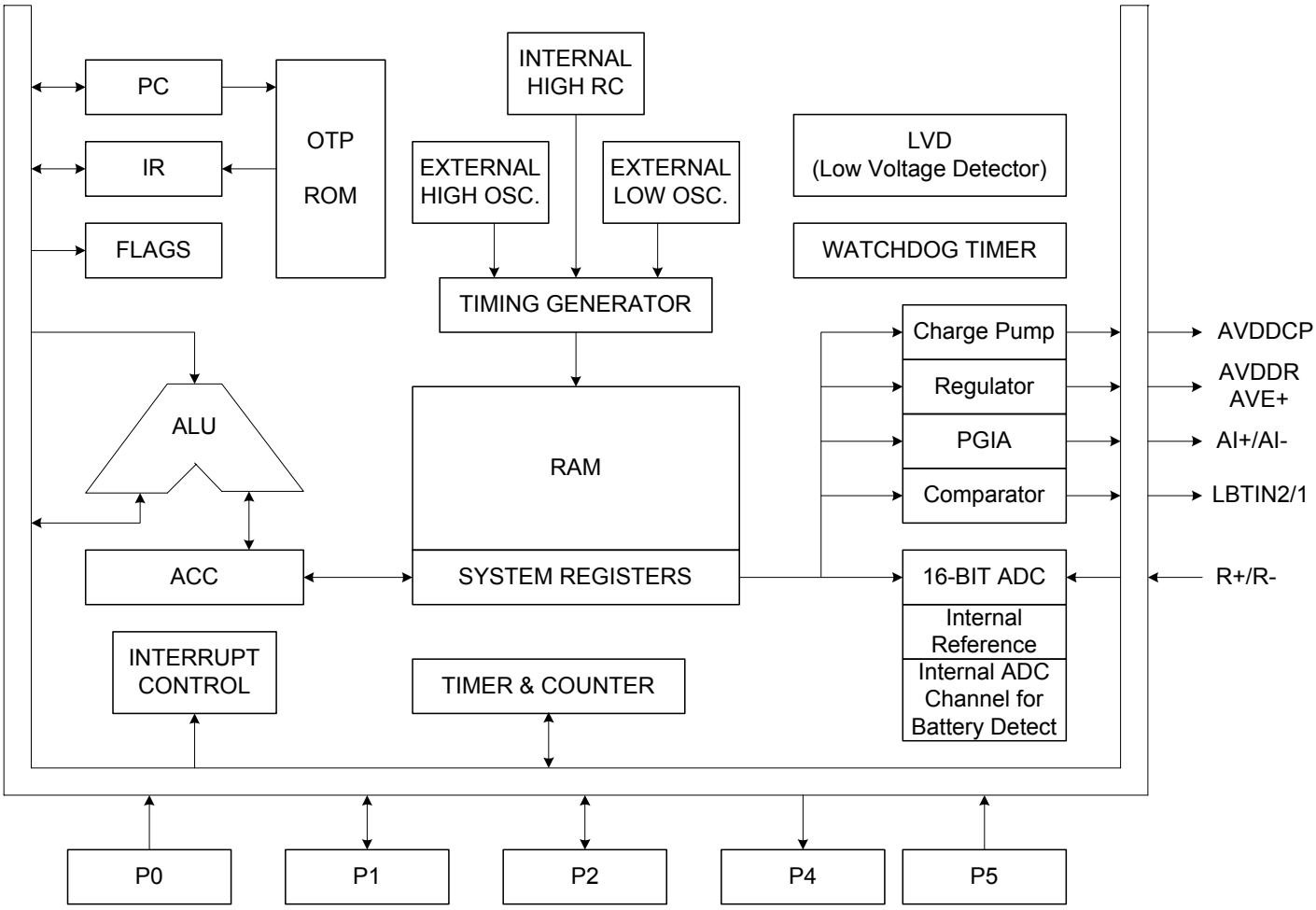
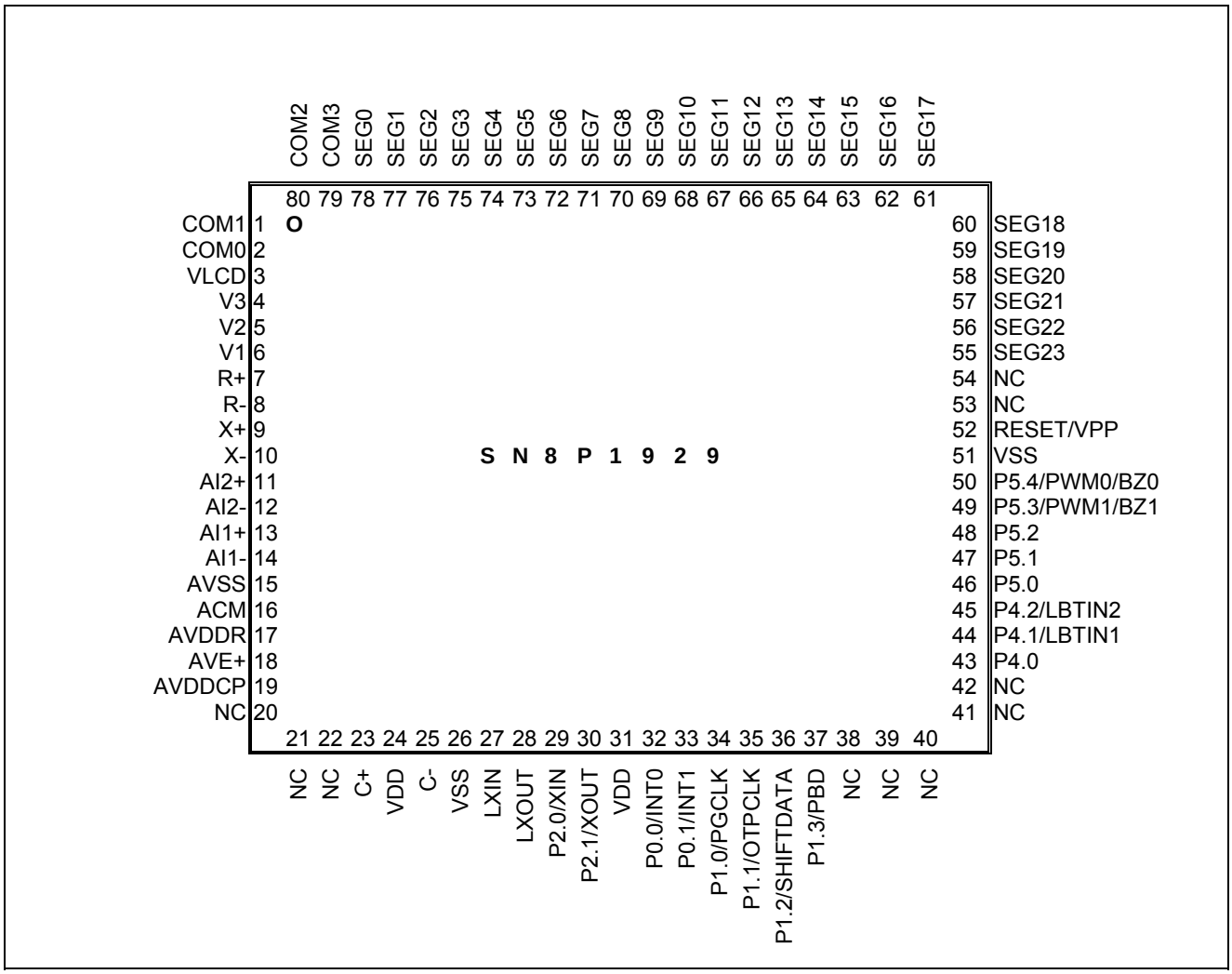


Figure 1-1 Simplified system block diagram

1.4 PIN ASSIGNMENT

SN8P1929 LQFP80

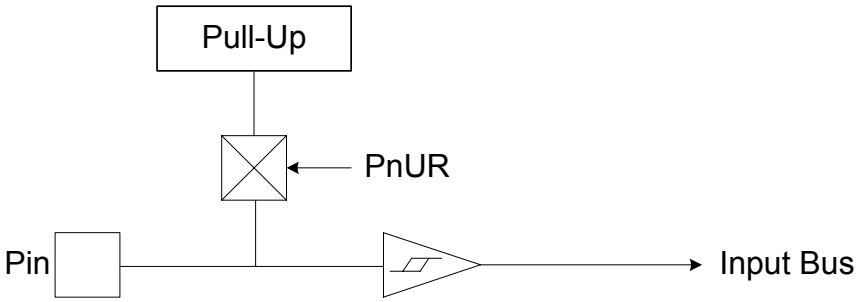


1.5 PIN DESCRIPTIONS

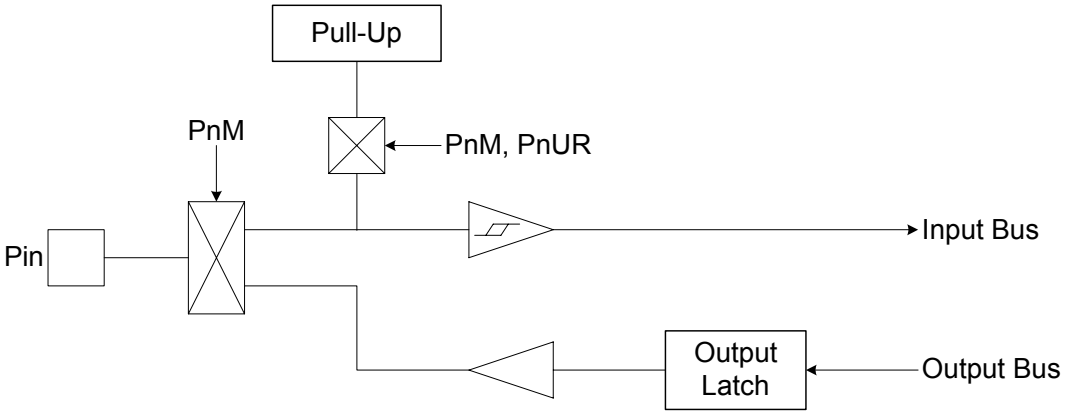
PIN NAME	TYPE	DESCRIPTION
VDD, VSS, AVSS	P	Power supply input pins for digital / analog circuit.
VLCD	P	LCD Power supply input.
AVDDR	P	Regulator power output pin, Voltage of 3.8V.
AVE+	P	Regulator output =3.0V /2.4V/1.5V for Sensor. Maximum output current of 10mA.
ACM	P	Band Gap Voltage output of 1.2V.
AVDDCP	P	Charge Pump Voltage output. (connect a 10uF or higher capacitor to ground)
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.
AI1+,AI2+	AI	Positive analog input channel.
AI-, AI2-	AI	Negative analog input channel.
C+	A	Positive capacitor terminal for charge pump regulator.
C-	A	Negative capacitor terminal for charge pump regulator.
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. No RC mode
P0.0 / INT0	I	P0.0 shared with INT0 trigger pin (Schmitt trigger) / Built-in pull-up resisters.
P0.1 / INT1	I	P0.1 shared with INT1 trigger pin (Schmitt trigger) / Built-in pull-up resisters.
P1 [3:0]	I/O	P1.0 ~ P1.3 bi-direction pins / wakeup pins/ Built-in pull-up resisters.
P2 [1:0]	I/O	P2.0 ~ P2.1 bi-direction pins / Built-in pull-up resisters. Shared with XIN/XOUT.
P4 [2:0]	I/O	bi-direction pins / Built-in pull-up resisters.
P5 [2:0]	I/O	bi-direction pins / Built-in pull-up resisters.
P5 [4:3]	I/O	bi-direction pins / Built-in pull-up resisters / Shared with PWM, TCOUT.
LBTIN1/2	I	Low BatTery detect Input pins shared with P4.1, P4.2.
COM [3:0]	O	COM0~COM3 LCD driver common port.
SEG0 ~ SEG23	O	LCD driver segment pins.

1.6 PIN CIRCUIT DIAGRAMS

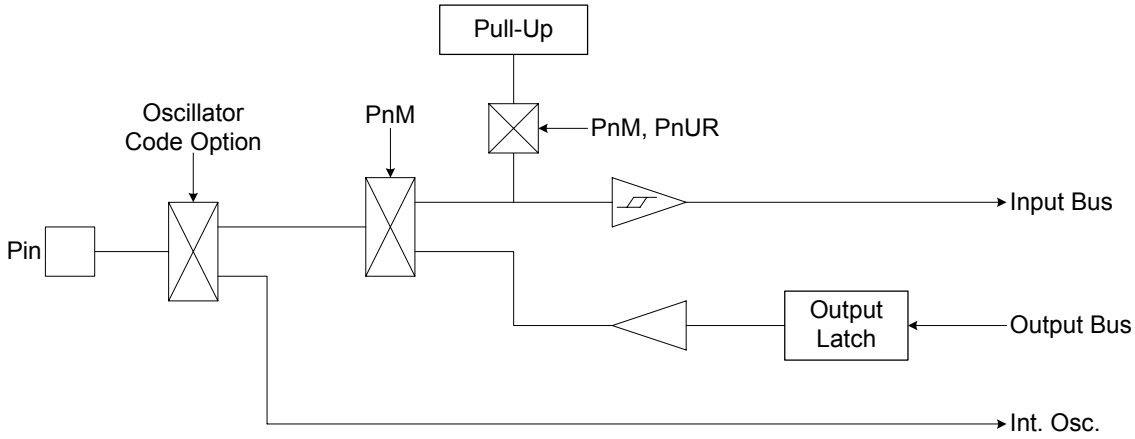
Port 0 structure:



Port1, Port4 and Port5 structure:



Port2 structure:



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ ***4K 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		
.		
.		
FFEH		End of user program
FFFH	***Code Option***	Code option address.

2.1.2 RESET VECTOR (0000H)

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

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

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

➢ **Example: Defining Reset Vector**

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

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


                ENDP                ; End of program
```

2.1.2.1 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.***

2.1.2.2 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
                …
```

2.1.2.3 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
```

2.1.2.4 CHECKSUM CALCULATION

The last ROM address is 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
```

2.1.3 CODE OPTION TABLE

Code Option	Content	Function Description
High_Clk	IHRC	High speed internal 16MHz RC. XIN/XOUT become to P2.0/P2.1 bi-direction I/O pins.
	4M X'tal	Standard crystal /resonator (e.g. 4M) for external high clock oscillator.
Watch_Dog	Enable	Enable Watchdog function
	Disable	Disable Watchdog 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
Noise Filter	Enable	Enable Noise Filter in High Noisy Environment.
	Disable	Disable Noise Filter.
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" and Enable Noise Filter is strongly recommended.***
2. ***Fcpu code option is only available for High Clock. Fcpu of slow mode is Flosc/4.***
3. ***In high noisy environment, disable "Low Power" is strongly recommended.***
4. ***The side effect is to increase the lowest valid working voltage level if enable "Low Power" and "Noise Filter" code option.***
5. ***Enable "Low Power" option will reduce operating current except in slow mode.***

2.1.4 DATA MEMORY (RAM)

☞ ***256 X 8-bit RAM***

		RAM location	
BANK 0	000h	General purpose area	; 000h~07Fh of Bank 0 = To store general
		.	; purpose data (128 bytes).
	07Fh	.	
	080h	System register	; 080h~0FFh of Bank 0 = To store system
		.	; registers (128 bytes).
	0FFh	End of bank 0 area	
BANK 1	100h	General purpose area	; 100h~17Fh of Bank 1 = To store general
		.	; purpose data (128 bytes).
	17Fh	.	;
BANK 15	F00h	LCD RAM area	; Bank 15 = To store LCD display data
		.	; (24bytes).
	F17h	End of LCD RAM	;

2.1.5 SYSTEM REGISTER

2.1.5.1 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	ADCM	ADCKS	CPM	CPCKS	DFM	ADCDL	ADCDH	LBTM	-	-	-	-	-
A	ROMADRH	ROMADRL	ROMDAH	ROMDAL	ROMCNT	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M	P3M	-	P5M	-	-	INTRQ	INTEN	OSCM	-	-	TC0R	PCL	PCH
D	P0	P1	P2	P3	-	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	P2UR	P3UR	-	P5UR	@HL	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.1.5.2 SYSTEM REGISTER DESCRIPTION

L, H = Working & @HL 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
CPCKS = Charge pump 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
T0C = Timer 0 counting register
TC1M = Timer/Counter 1 mode register
TC1C = Timer/Counter 1 counting register
STKP = Stack pointer buffer
@HL = RAM HL indirect addressing index pointer
@YZ = RAM YZ indirect addressing index pointer
R = Working register and ROM look-up data buffer

OPTION = RCLK options.
RBANK= RAM bank select register
AMPCHS = PGIA channel selection
ADCM = ADC's mode register
CPM = Charge pump mode
DFM = Decimation filter mode
ADCDH = ADC high-byte data buffer
P1W = Port 1 wakeup register
P_NUR = Port N pull-up register
INTRQ = Interrupt request register
OSCM = Oscillator mode register
PCH, PCL = Program counter
TC0M = Timer/Counter 0 mode register
TC0C = Timer/Counter 0 counting register
TC0R = Timer/Counter 0 auto-reload data buffer
LBTM = Low Battery Detect Register
STK0~STK7 = Stack 0 ~ stack 7 buffer
ROMADRH/L= ISP ROM Address
ROMDAH/L= ISP ROM Data
ROMCNT = ISP ROM Counter

2.1.5.3 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	NT0	NPD	-	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	RBNKS3	RBNKS2	RBNKS1	RBNKS0	R/W	RBANK
088H								RCLK	R/W	OPTION
089H	LCDREF1	LCDREF0	LCDBNK	-	LCDENB	LCDBIAS			R/W	LCDM1
090H	CHPENB	BGRENB	FDS1	FDS0	GS2	GS1	GS0	AMPENB	R/W	AMPM
091H	-	-	-	-	CHS3	CHS2	CHS1	CHS0	R/W	AMPCHS
092H	-	-	-	-	-	AMPCKS2	AMPCKS1	AMPCKS0	W	AMPCKS
093H	-	-	-	-	IRVS	RVS1	RVS0	ADCENB	R/W	ADCM
094H	ADCKS7	ADCKS6	ADCKS5	ADCKS4	ADCKS3	ADCKS2	ADCKS1	ADCKS0	W	ADCKS
095H	ACMENB	AVDDRENB	AVENB	AVESEL1	AVESEL0	CPAUTO	CPON	CPRENB	R/W	CPM
096H	-	-	-	-	CPCKS3	CPCKS2	CPCKS1	CPCKS0	W	CPCKS
097H			-	-	-	WRS0		DRDY	R/W	DFM
098H	ADCB9	ADCB8	ADCB7	ADCB6	ADCB5	ADCB4	ADCB3	ADCB2	R	ADCDL
099H	ADCB17	ADCB16	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	R	ADCDH
09AH	-	-	-	-	-	LBTO	P41IO	LBTENB	R/W	LBTM
09BH	-	-	-	-	-	-	-	CPSAVE	R/W	CPMTEST
0A0H	VPPCHK	-	-	-	ROMADR11	ROMADR10	ROMADR9	ROMADR8	R/W	ROMADRH
0A1H	ROMADR7	ROMADR6	ROMADR5	ROMADR4	ROMADR3	ROMADR2	ROMADR1	ROMADR0	R/W	ROMADRL
0A2H	ROMDA15	ROMDA14	ROMDA13	ROMDA12	ROMDA11	ROMDA10	ROMDA9	ROMDA8	R/W	ROMDAH
0A3H	ROMDA7	ROMDA6	ROMDA5	ROMDA4	ROMDA3	ROMDA2	ROMDA1	ROMDA0	R/W	ROMDAL
0A4H	ROMCNT7	ROMCNT6	ROMCNT5	ROMCNT4	ROMCNT3	ROMCNT2	ROMCNT1	ROMCNT0	W	ROMCNT
0BFH	PEDGEN	-	-	P00G1	P00G0	-	-	-	R/W	PEDGE
0C0H	-	-	-	-	P13W	P12W	P11W	P10W	W	P1W
0C1H	-	-	-	-	P13M	P12M	P11M	P10M	R/W	P1M
0C2H	-	-	-	-	-	-	P21M	P20M	R/W	P2M
0C4H	-	-	-	-	-	P42M	P41M	P40M	R/W	P4M
0C5H	-	-	-	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H	-	TC1IRQ	TC0IRQ	T0IRQ	-	-	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	-	TC1IEN	TC0IEN	T0IEN	-	-	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	-	-	-	-	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	-	-	-	-	-	-	P01	P00	R	P0
0D1H	-	-	-	-	P13	P12	P11	P10	R/W	P1
0D2H	-	-	-	-	-	-	P21	P20	R/W	P2
0D4H	-	-	-	-	-	P42	P41	P40	R/W	P4
0D5H	-	-	-	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	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE	-	-	-		STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	-	-	-	-	-	-	P01R	P00R	W	P0UR
0E1H	-	-	-	-	P13R	P12R	P11R	P10R	W	P1UR
0E2H	-	-	-	-	-	-	P21R	P20R	W	P2UR
0E4H	-	-	-	-	-	P42R	P41R	P40R	W	P4UR
0E5H	-	-	-	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	-	-	-		S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	-	-	-	-	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	-	-	-	-	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H

0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	-	-	-	-	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	-	-	-	-	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	-	-	-	-	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	-	-	-	-	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	-	-	-	-	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ ***Note:***

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

2.1.5.4 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.***

2.1.6 PROGRAM FLAG

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

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

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

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

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

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

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

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

2.1.6.1 PROGRAM COUNTER

The program counter (PC) is a 12-bit binary counter separated into the high-byte 4 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

Besides, it can be replaced with specific address by executing CALL or JMP instruction. When JMP or CALL instruction is executed, the destination address will be inserted to bit 0 ~ bit 11.

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

☞ **ONE ADDRESS SKIPPING**

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

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

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

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

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

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

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

INCS instruction:

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

INCMS instruction:

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

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

2.1.7 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
        …
        …
```

2.1.7.1 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.***

2.1.7.2 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
        …
```

2.1.7.3 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.
```

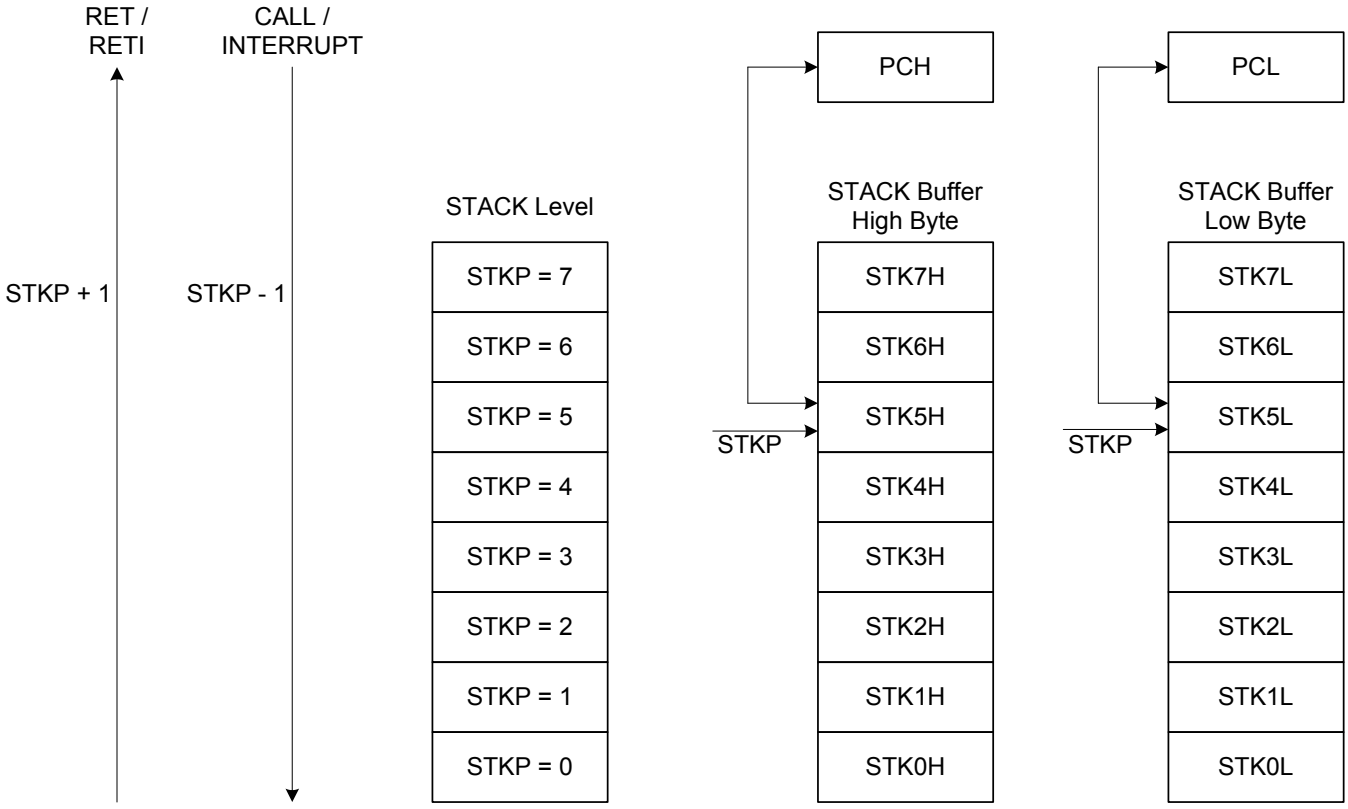
Example: Indirectly addressing mode with @YZ register

```
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 4-bit register to store the address used to access the stack buffer, 12-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

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

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

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

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

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

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

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

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

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

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

2.3.3 STACK OPERATION EXAMPLE

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

Stack Level	STKP Register				Stack Buffer		Description
	STKPB3	STKPB2	STKPB1	STKPB0	High Byte	Low Byte	
0	1	1	1	1	Free	Free	-
1	1	1	1	0	STK0H	STK0L	-
2	1	1	0	1	STK1H	STK1L	-
3	1	1	0	0	STK2H	STK2L	-
4	1	0	1	1	STK3H	STK3L	-
5	1	0	1	0	STK4H	STK4L	-
6	1	0	0	1	STK5H	STK5L	-
7	1	0	0	0	STK6H	STK6L	-
8	0	1	1	1	STK7H	STK7L	-
> 8	0	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
	STKPB3	STKPB2	STKPB1	STKPB0	High Byte	Low Byte	
8	0	1	1	1	STK7H	STK7L	-
7	1	0	0	0	STK6H	STK6L	-
6	1	0	0	1	STK5H	STK5L	-
5	1	0	1	0	STK4H	STK4L	-
4	1	0	1	1	STK3H	STK3L	-
3	1	1	0	0	STK2H	STK2L	-
2	1	1	0	1	STK1H	STK1L	-
1	1	1	1	0	STK0H	STK0L	-
0	1	1	1	1	Free	Free	-

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

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

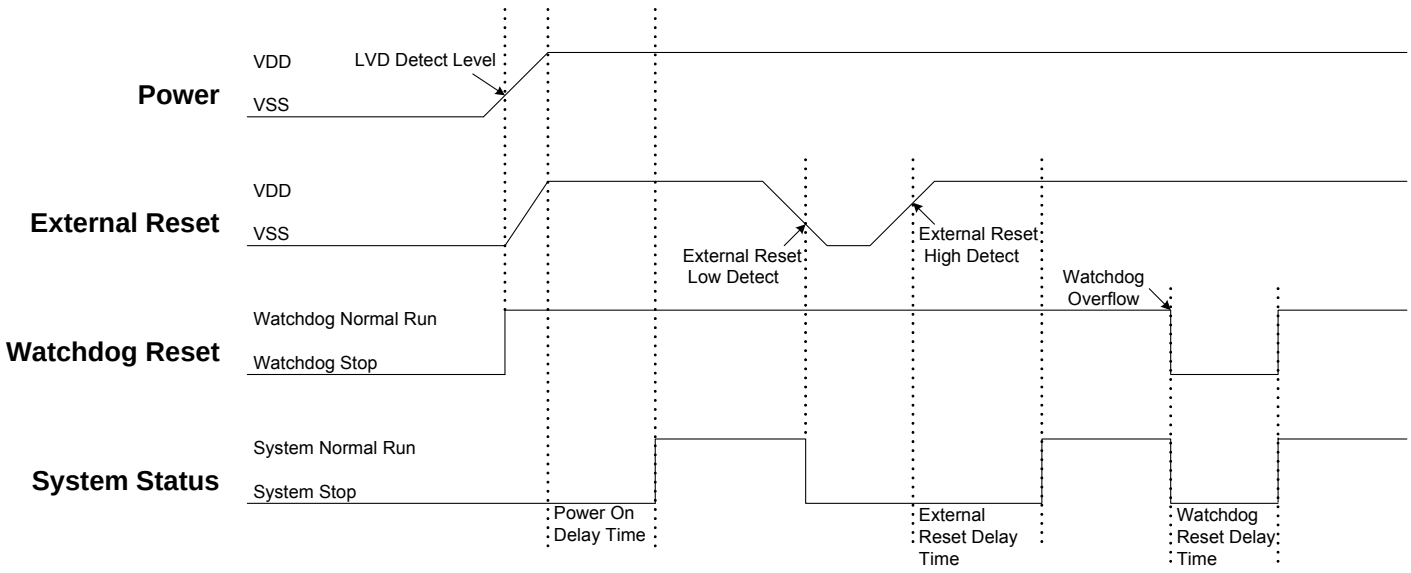
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

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

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

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

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



3.2 POWER ON RESET

The power on reset depend on LVD operation for most power-up situations. The power supplying to system is a rising curve and needs some time to achieve the normal voltage. Power on reset sequence is as following.

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

3.3 WATCHDOG RESET

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

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

Watchdog timer application note is as following.

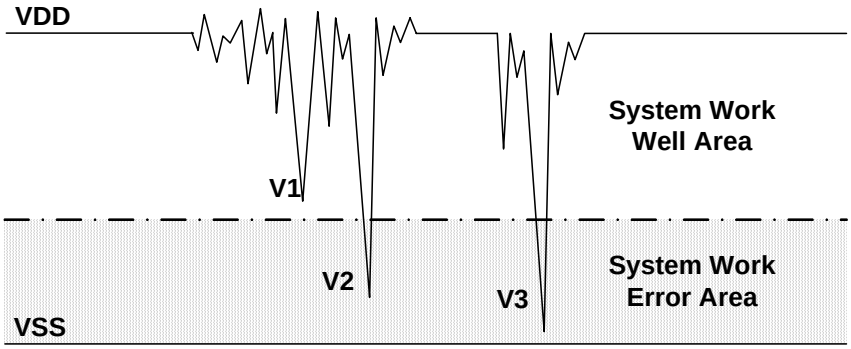
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

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

3.4 BROWN OUT RESET

3.4.1 BROWN OUT DESCRIPTION

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



Brown Out Reset Diagram

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

DC application:

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

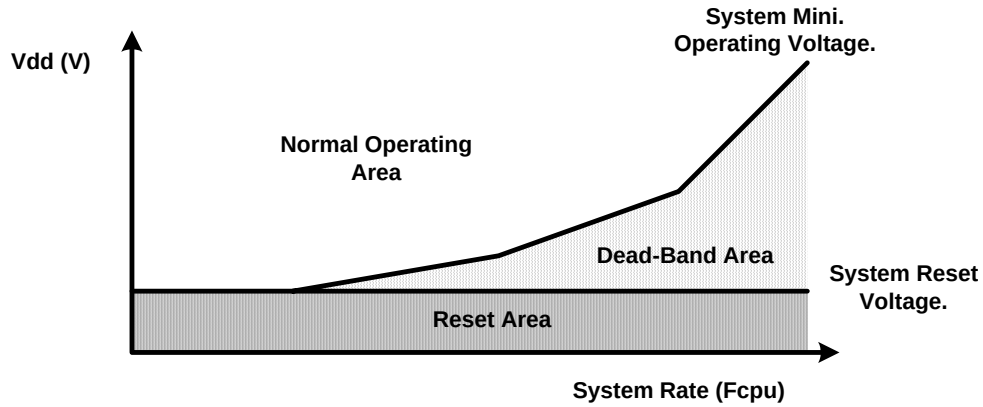
AC application:

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

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

3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION

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



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

3.4.3 BROWN OUT RESET IMPROVEMENT

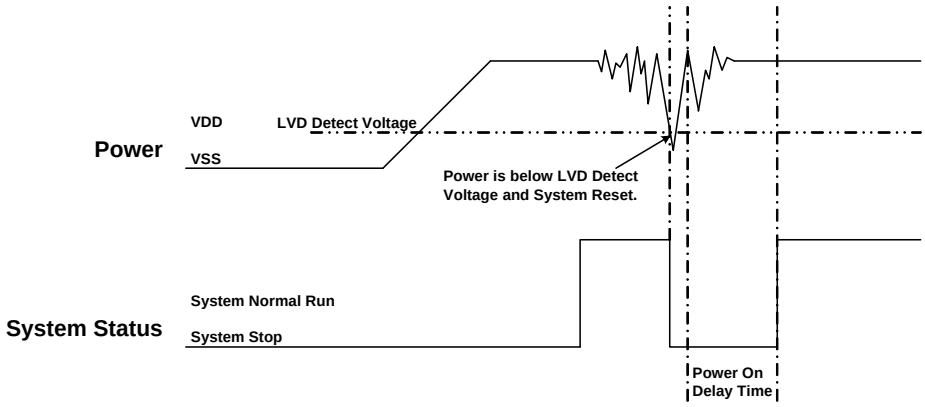
How to improve the brown reset condition? There are some methods to improve brown out reset as following.

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

✷ ***Note:***

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

LVD reset:



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

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

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

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

3.5 EXTERNAL RESET

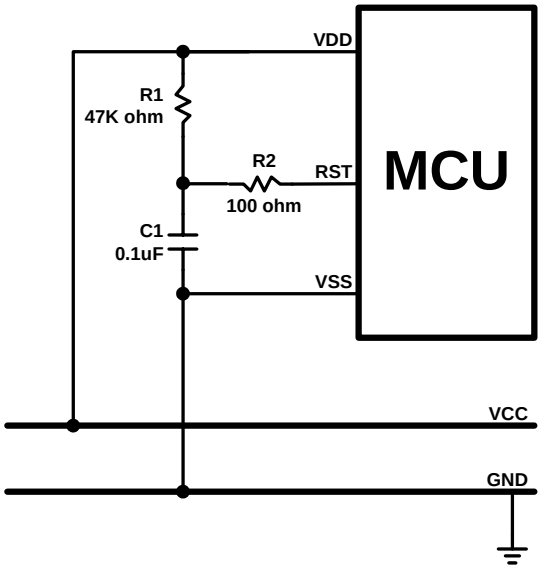
External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the low voltage and the system is reset. The external reset operation actives in power on and normal running mode. During system power-up, the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

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

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

3.6 EXTERNAL RESET CIRCUIT

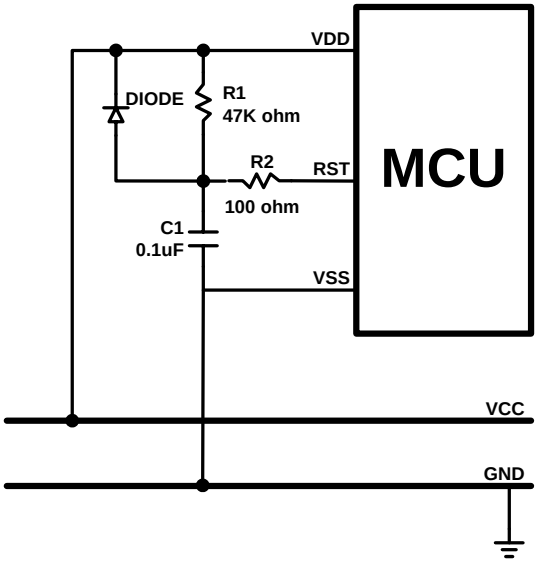
3.6.1 Simply RC Reset Circuit



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

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

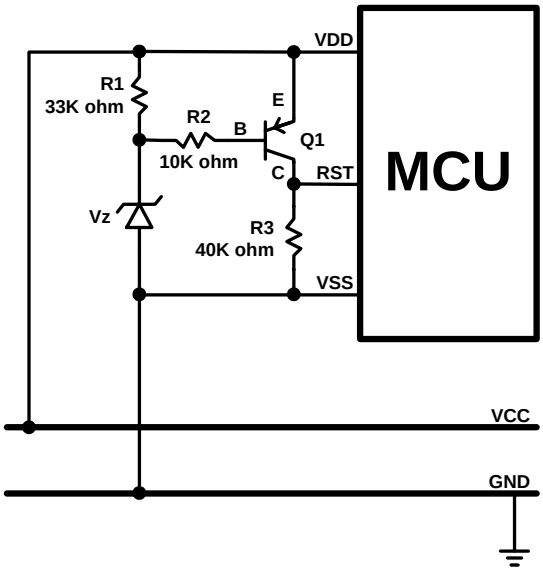
3.6.2 Diode & RC Reset Circuit



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

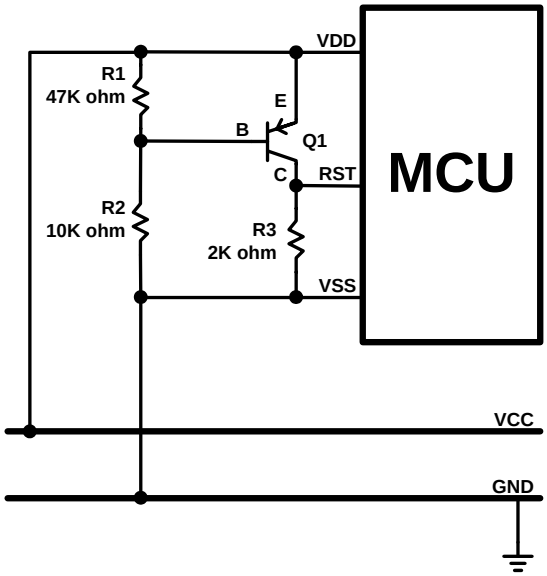
✸ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.6.3 Zener Diode Reset Circuit



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

3.6.4 Voltage Bias Reset Circuit

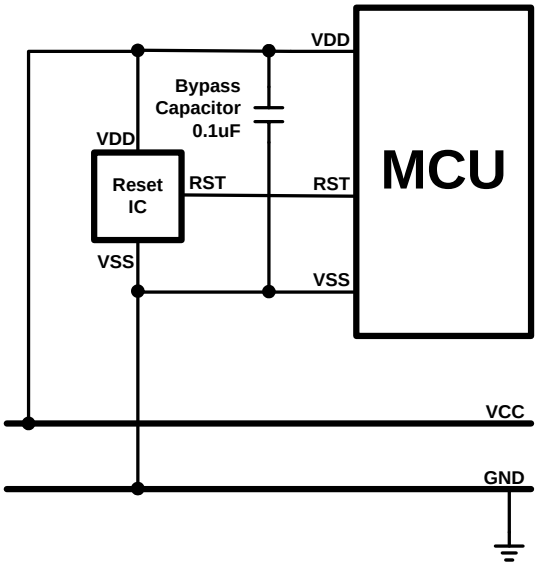


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to "0.7V x (R1 + R2) / R1",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

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

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

3.6.5 External Reset IC



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

4 SYSTEM CLOCK

4.1 OVERVIEW

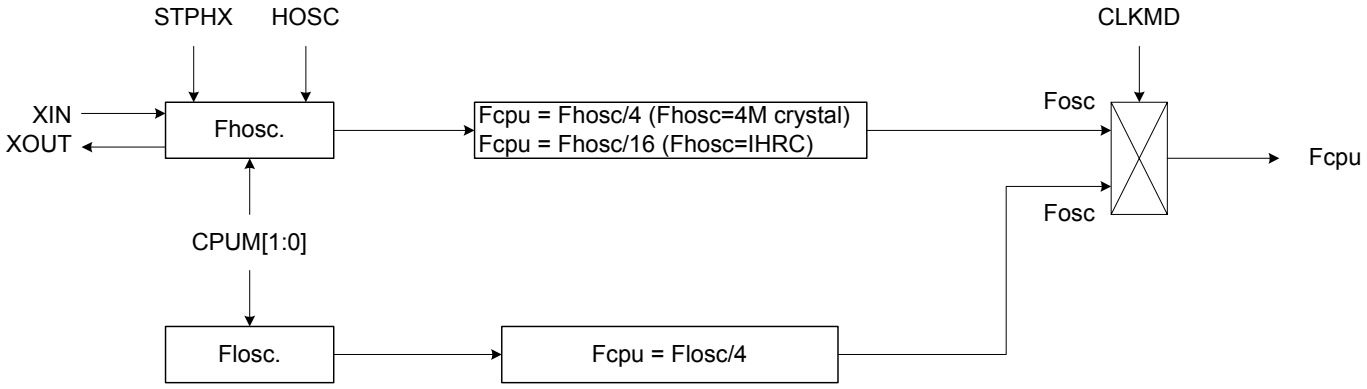
The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit or on-chip 16MHz high-speed RC oscillator circuit (IHRC 16MHz). The low-speed clock is generated from LXIN/LXOUT by 32768 crystal or RC oscillator circuit

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

☞ **Normal Mode (High Clock):** **Fcpu = Fhosc / 4**, (Fhosc= 4M/8M crystal)
Fcpu = Fhosc / 16, (Fhosc=IHRC)

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

4.2 CLOCK BLOCK DIAGRAM



- HOSC: High_Clk code option.
- Fhosc: External high-speed clock / Internal high-speed RC 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. Internal low-speed RC oscillator is still running.

Bit 2 **CLKMD:** System high/Low clock mode control bit.
0 = Normal (dual) mode. System clock is high clock.
1 = Slow mode. System clock is 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.
(The WTCKS bit will be set as "1" when Int_16k_RC "Always_On" selected in the code option)

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 internal low-speed oscillator will be stopped.**

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

4.4 SYSTEM HIGH CLOCK

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

High_Clk Code Option	Description
IHRC_16M	The high clock is internal 16MHz oscillator RC type. XIN and XOUT pins are general purpose I/O pins.
4M	The high clock is external oscillator. The typical frequency is 4MHz.

4.4.1 INTERNAL HIGH RC

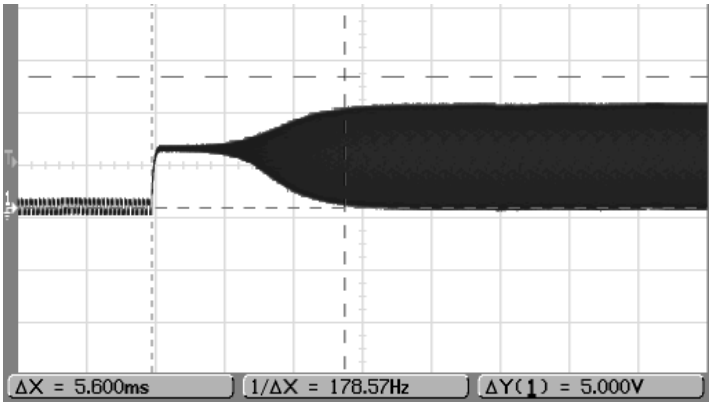
The chip is built-in RC type internal high clock (16MHz) controlled by “IHRC_16M” code options. In “IHRC_16M” mode, the system clock is from internal 16MHz RC type oscillator and XIN / XOUT pins are general-purpose I/O pins.

- **IHRC:** High clock is internal 16MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.

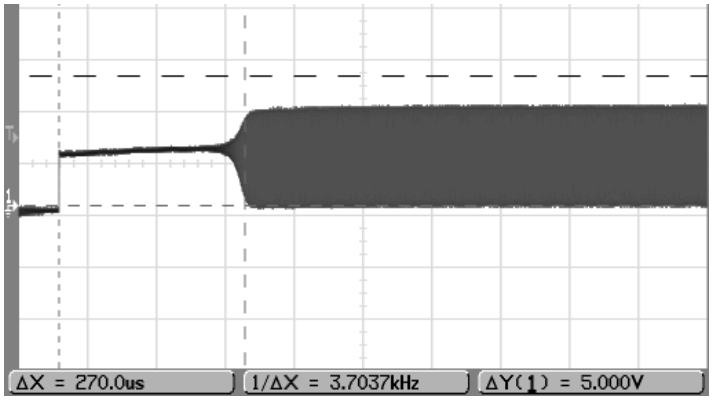
4.4.2 EXTERNAL HIGH CLOCK

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

4MHz Crystal

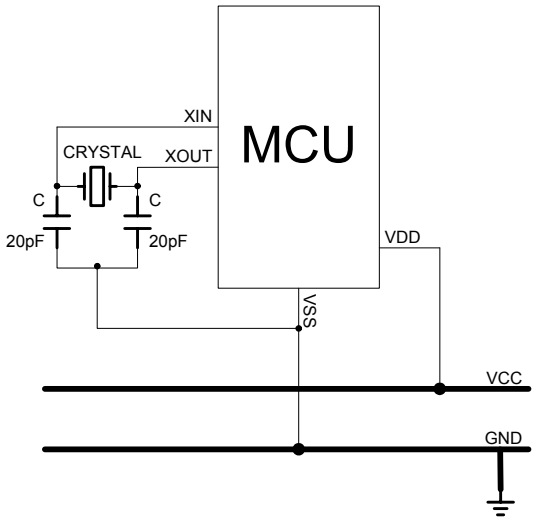


4MHz Ceramic



4.4.2.1 CRYSTAL/CERAMIC

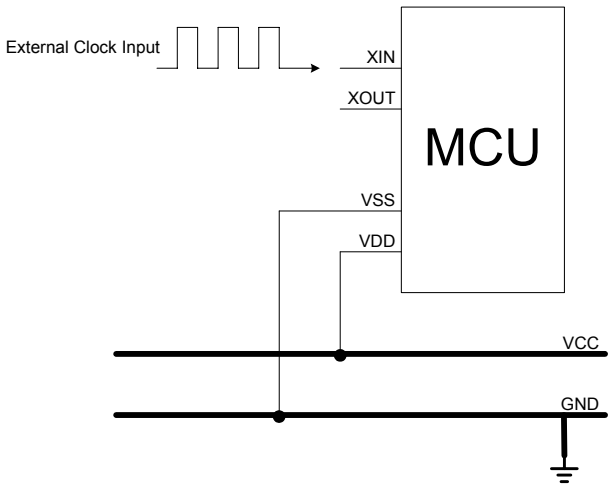
Crystal/Ceramic devices are driven by XIN, XOUT pins. For high/normal/low frequency, the driving currents are different. High_Clk code option supports different frequencies. 12M option is for high speed (ex. 12MHz). 4M option is for normal speed (ex. 4MHz).



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

4.4.2.2 EXTERNAL CLOCK SIGNAL

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



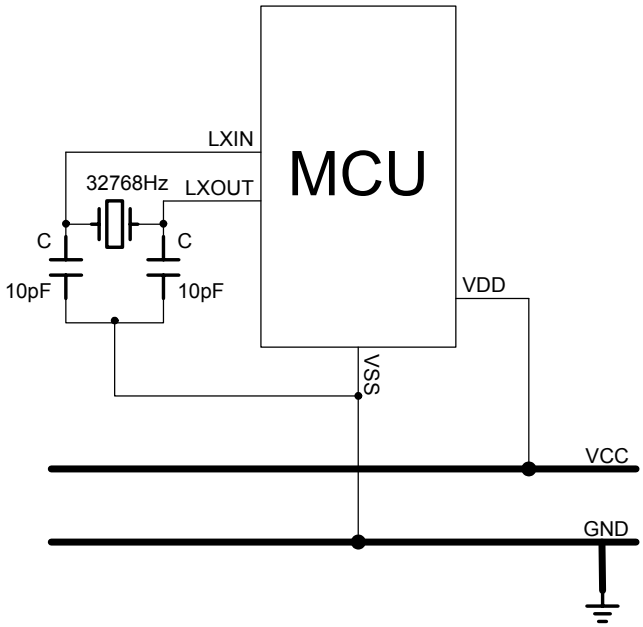
✷ ***Note: The GND of external oscillator circuit must be as near as possible to VSS pin of micro-controller.***

4.5 SYSTEM LOW CLOCK

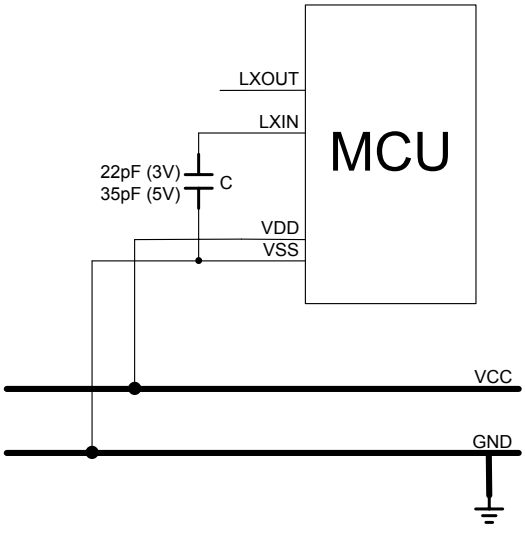
The system low clock source is the external low-speed oscillator. The low-speed oscillator can use 32768 crystal or RC type oscillator circuit.

4.5.1.1 CRYSTAL

Crystal devices are driven by LXIN, LXOUT pins. The 32768 crystal and 10uF 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.***

4.5.2 SYSTEM CLOCK MEASUREMENT

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

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

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

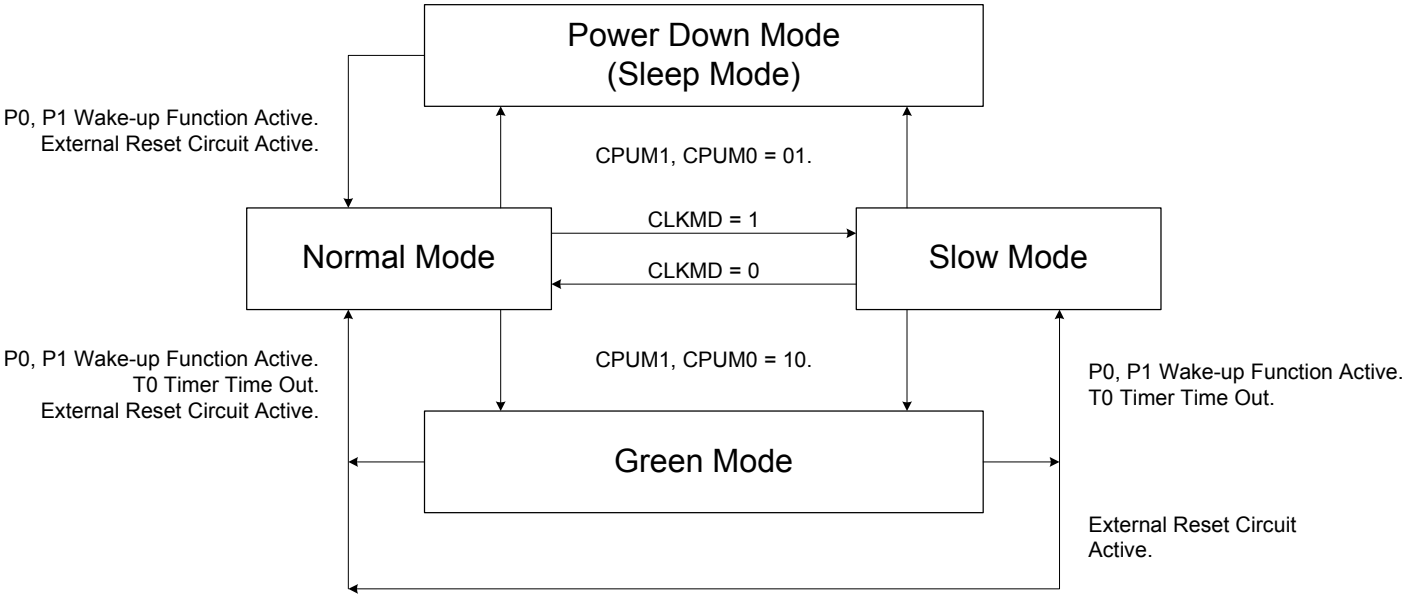
✸ ***Note: Do not measure the RC frequency directly from XIN; the probe impendence will affect the RC frequency.***

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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



System Mode Switching Diagram

Operating mode description

MODE	NORMAL	SLOW	GREEN	POWER DOWN (SLEEP)	REMARK
EHOSC	Running	By STPHX	By STPHX	Stop	
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) or green mode, program doesn't execute. The wakeup trigger can wake the system up to normal mode or slow mode. The wakeup trigger sources are external trigger (P0, P1 level change) and internal trigger (T0/TC0 timer overflow).

- Power down mode is waked up to normal mode. The wakeup trigger is only external trigger (P0, P1 level change)
- Green mode is waked up to last mode (normal mode or slow mode). The wakeup triggers are external trigger (P0, P1 level change) and internal trigger (T0/TC0 timer overflow).

5.3.2 WAKEUP TIME

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

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

The value of the wakeup time is as the following.

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

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

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

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

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.

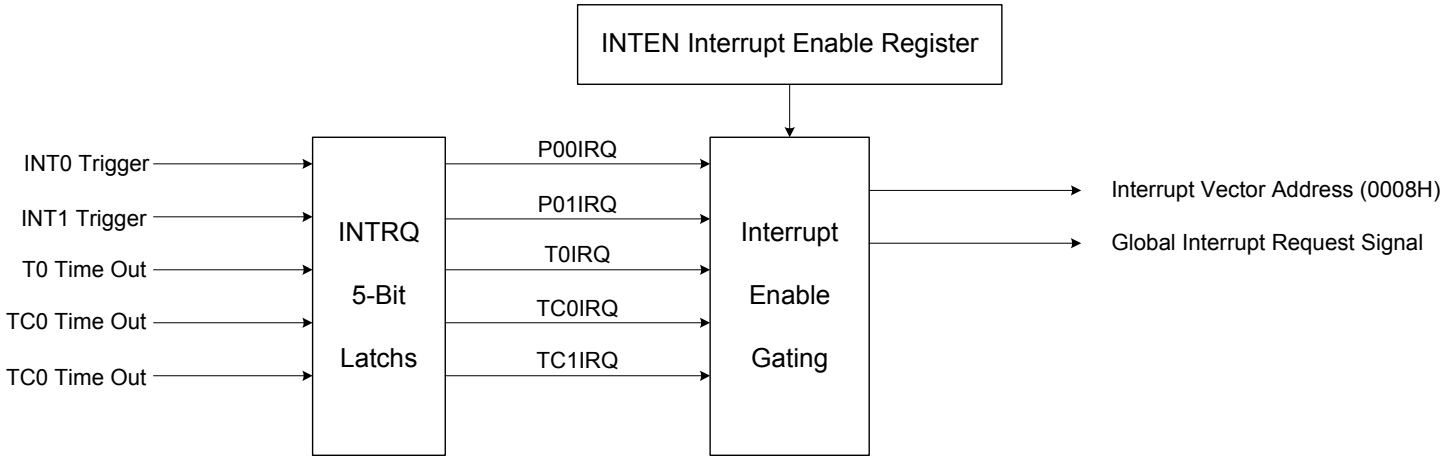
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	-	-	-	-	P13W	P12W	P11W	P10W
Read/Write	-	-	-	-	W	W	W	W
After reset	-	-	-	-	0	0	0	0

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

6 INTERRUPT

6.1 OVERVIEW

This MCU provides three interrupt sources, including three internal interrupts (T0/TC0/TC1) 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. Once interrupt service is executed, the GIE bit in STKP register will clear to "0" for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to "1" to accept the next interrupts' request. All of the interrupt request signals are stored in INTRQ register.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

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

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

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

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

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

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

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

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

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

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

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

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

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

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 MULTI-INTERRUPT OPERATION

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

Interrupt Name	*Trigger Event Description*
P00IRQ	P0.0 trigger controlled by PEDGE
P01IRQ	P0.1 trigger controlled by PEDGE
T0IRQ	T0C overflow
TC0IRQ	TC0C overflow
TC1IRQ	TC1C overflow

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     INTT0CHK        ; Jump check to next interrupt
        B0BTS0  FP01IRQ         ; Check P01IRQ
        JMP     INTP01
INTT0CHK:                       ; Check T0 interrupt request
        B0BTS1  FT0IEN          ; Check T0IEN
        JMP     INTTC0CHK       ; Jump check to next interrupt
        B0BTS0  FT0IRQ          ; Check T0IRQ
        JMP     INTT0           ; Jump to T0 interrupt service routine
INTTC0CHK:                      ; Check TC0 interrupt request
        B0BTS1  FTC0IEN         ; Check TC0IEN
        JMP     INTTC1CHK       ; Jump check to next interrupt
        B0BTS0  FTC0IRQ         ; Check TC0IRQ
        JMP     INTTC0          ; Jump to TC0 interrupt service routine
INTTC1CHK:                      ; Check T1 interrupt request
        B0BTS1  FTC1IEN         ; Check TC1IEN
        JMP     INT_EXIT        ; Jump check to next interrupt
        B0BTS0  FTC1IRQ         ; Check TC1IRQ
        JMP     INTTC1          ; 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 expects input mode only of port0.

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

0C2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2M	-	-	-	-	-	-	P21M	P20M
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

0C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	-	-	-	-	-	P42M	P41M	P40M
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	-	P54M	P53M	P52M	P51M	P50M
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	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 0 is input only port***
3. ***Port 2 is shared with XIN and XOUT***

➢ **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	-	-	-	-	P13R	P12R	P11R	P10R
Read/Write	-	-	-	-	W	W	W	W
After reset	-	-	-	-	0	0	0	0

0E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2UR	-	-	-	-	-	-	P21R	P20R
Read/Write	-	-	-	-	-	-	W	W
After reset	-	-	-	-	-	-	0	0

0E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	-	-	-	-	-	P42R	P41R	P40R
Read/Write	-	-	-	-	-	W	W	W
After reset	-	-	-	-	-	0	0	0

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

✸ ***Note: PnUR is Write Only Register.***

➢ **Example: I/O Pull up Register**

```
MOV     A, #0FFH        ; Enable Port1 Pull-up register,
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	-	-	-	-	P13	P12	P11	P10
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

0D2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2	-	-	-	-	-	-	P21	P20
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	-	-	-	-	-	P42	P41	P40
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After reset	-	-	-	-	-	0	0	0

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

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

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

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

```
MOV     A, #0FFH  ; Write data FFH to all Port.
B0MOV   P1, A
B0MOV   P2, A
B0MOV   P4, 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”.
```

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
```

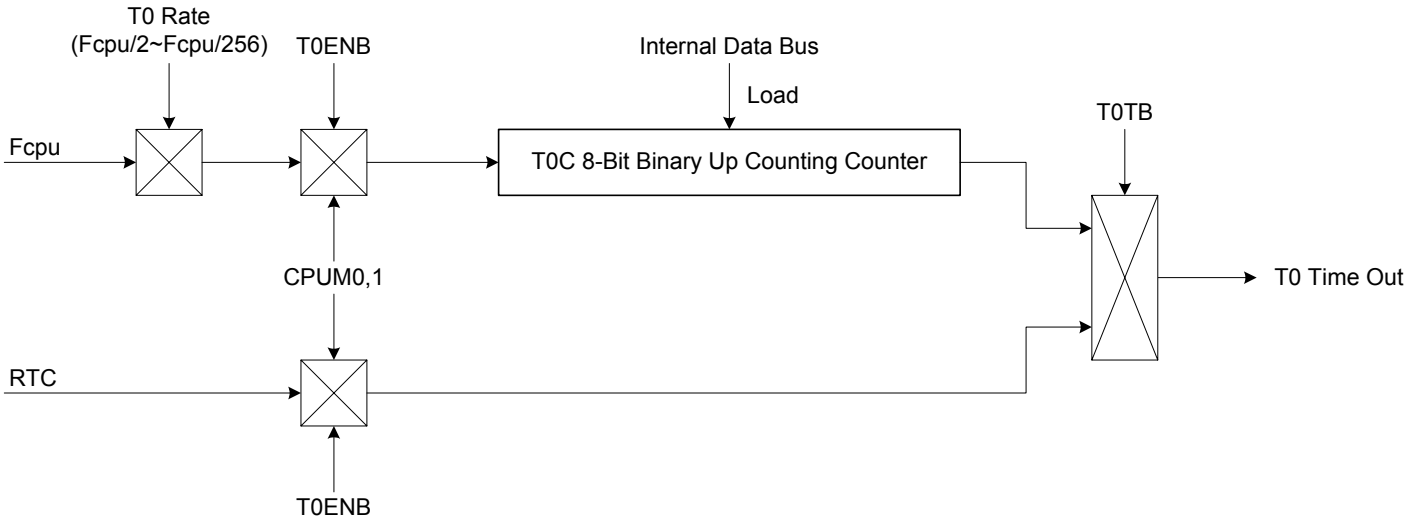
8.2 TIMER 0 (T0)

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.
```

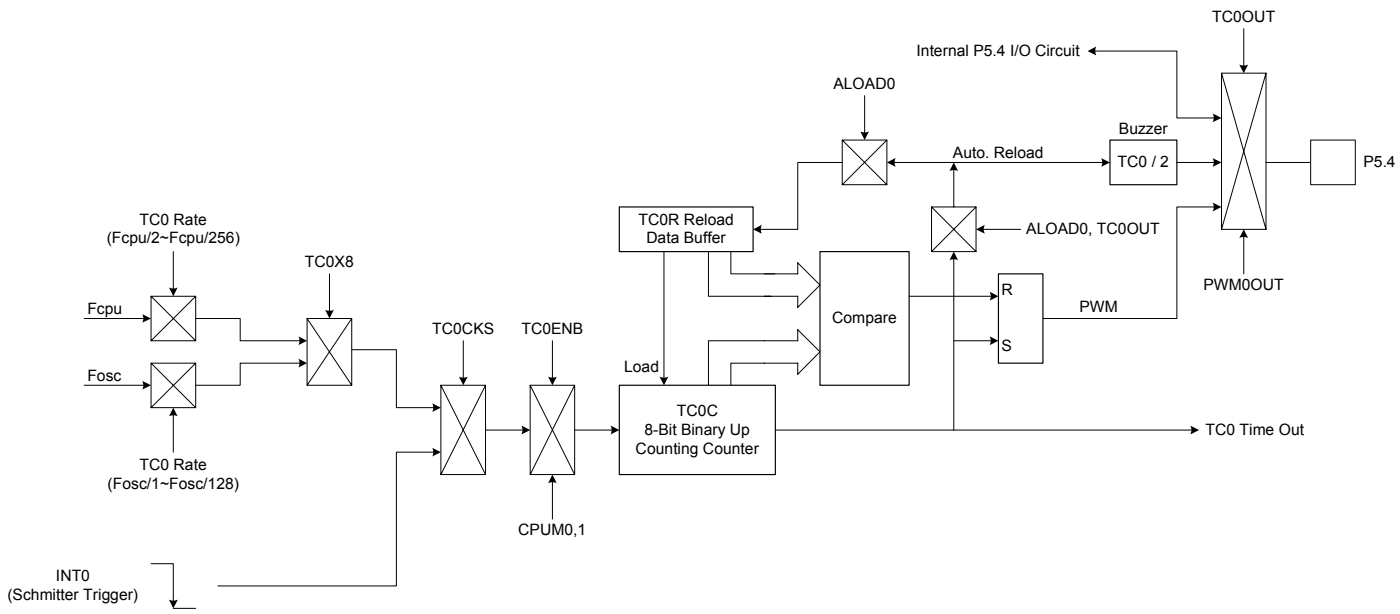
8.3 TIMER/COUNTER 0 (TC0)

8.3.1 OVERVIEW

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

The main purposes of the TC0 timer is as following.

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



8.3.2 TC0M MODE REGISTER

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

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

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

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

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

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

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

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

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

8.3.3 TC1X8, TC0X8, TC0GN FLAGS

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	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.

✸ ***Note: Under TC0 event counter mode (TC0CKS=1), TC0X8 bit and TC0RATE are useless.***

8.3.4 TC0C COUNTING REGISTER

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

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

The equation of TC0C initial value is as following.

TC0C initial value = N - (TC0 interrupt interval time * input clock)

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

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

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

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

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

TC0RATE	TC0CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 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 = N - (TC0 interrupt interval time * input clock)

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

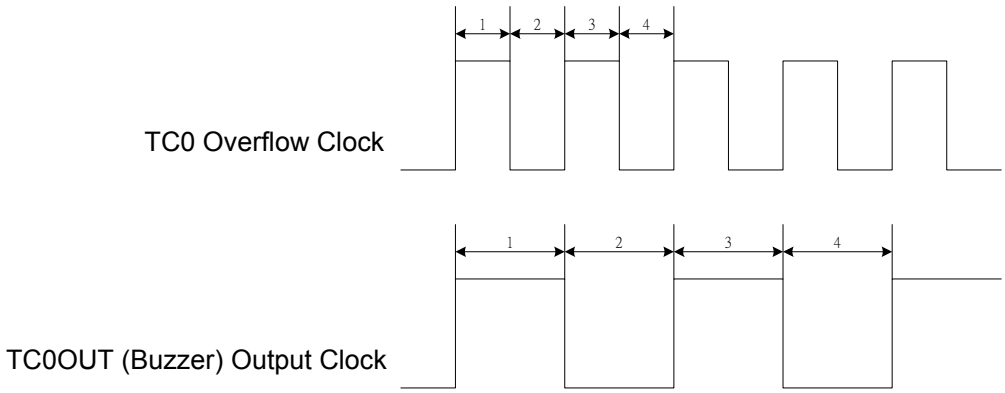
TC0CKS	TC0X8	PWM0	ALOAD0	TC0OUT	N	TC0R valid value	TC0R value binary type
0	0 (Fcpu/2~ Fcpu/256)	0	x	x	256	0x00~0xFF	00000000b~11111111b
		1	0	0	256	0x00~0xFF	00000000b~11111111b
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b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

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

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

8.3.6 TC0 CLOCK FREQUENCY OUTPUT (BUZZER)

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



➢ **Example: Setup TC0OUT output from TC0 to TC0OUT (P5.4). The external high-speed clock is 4MHz. The TC0OUT frequency is 0.5KHz. Because the TC0OUT signal is divided by 2, set the TC0 clock to 1KHz. The TC0 clock source is from external oscillator clock. T0C rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 131.**

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

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

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

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

8.3.7 TC0 TIMER OPERATION SEQUENCE

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

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

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

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

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

☞ **Set TC0 timer clock source.**

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

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

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

☞ **Set TC0 timer auto-load mode.**

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

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

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

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

☞ **Set TC0 timer function mode.**

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

☞ **Enable TC0 timer.**

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

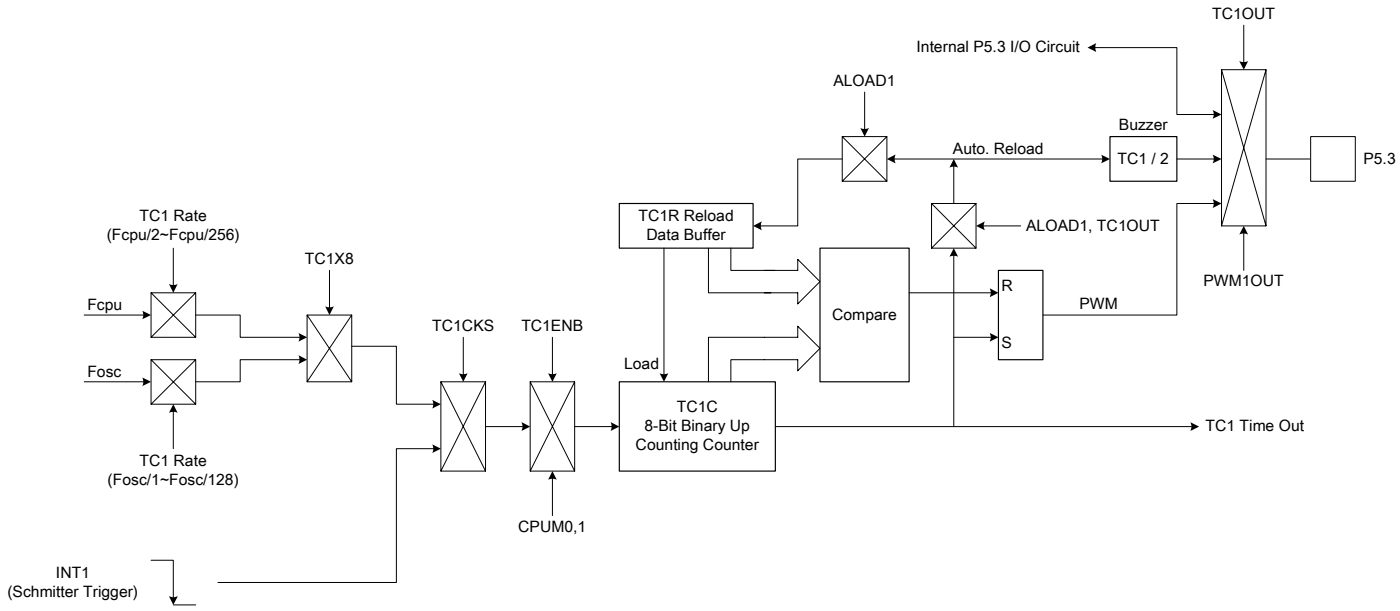
8.4 TIMER/COUNTER 1 (TC1)

8.4.1 OVERVIEW

The TC1 is an 8-bit binary up counting timer. TC1 has two clock sources including internal clock and external clock for counting a precision time. The internal clock source is from Fcpu or Fosc controlled by TC1X8 flag to get faster clock source (Fosc). The external clock is INT1 from P0.1 pin (Falling edge trigger). Using TC1M register selects TC1C's clock source from internal or external. If TC1 timer occurs an overflow, it will continue counting and issue a time-out signal to trigger TC1 interrupt to request interrupt service. TC1 overflow time is 0xFF to 0X00 normally. Under PWM mode, TC1 overflow is decided by PWM cycle controlled by ALOAD1 and TC1OUT bits.

The main purpose of the TC1 timer is as following.

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



8.4.2 TC1M MODE REGISTER

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

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

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

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

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

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

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

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

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

8.4.3 TC1X8, 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.

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

8.4.4 TC1C COUNTING REGISTER

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

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

The equation of TC1C initial value is as following.

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

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

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

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

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

The basic timer table interval time of TC1, TC1X1 = 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 = N - (TC1 interrupt interval time * input clock)

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

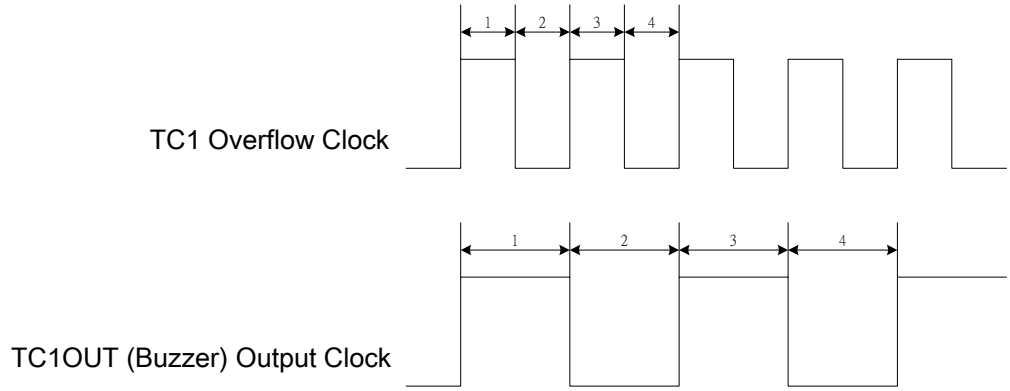
TC1CKS	TC1X8	PWM1	ALOAD1	TC1OUT	N	TC1R valid value	TC1R value binary type
0	0 (Fcpu/2~Fcpu/256)	0	x	x	256	0x00~0xFF	00000000b~11111111b
		1	0	0	256	0x00~0xFF	00000000b~11111111b
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b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

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

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

8.4.6 TC1 CLOCK FREQUENCY OUTPUT (BUZZER)

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



➢ **Example: Setup TC1OUT output from TC1 to TC1OUT (P5.3). The external high-speed clock is 4MHz. The TC1OUT frequency is 0.5KHz. Because the TC1OUT signal is divided by 2, set the TC1 clock to 1KHz. The TC1 clock source is from external oscillator clock. TC1 rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 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 interrupt function is disabled.
```

☞ **Set TC1 timer clock source.**

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

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

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

☞ **Set TC1 timer auto-load mode.**

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

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

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

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

☞ **Set TC0 timer function mode.**

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

☞ **Enable TC0 timer.**

```
        B0BSET      FTC0ENB         ; Enable TC1 timer.
```

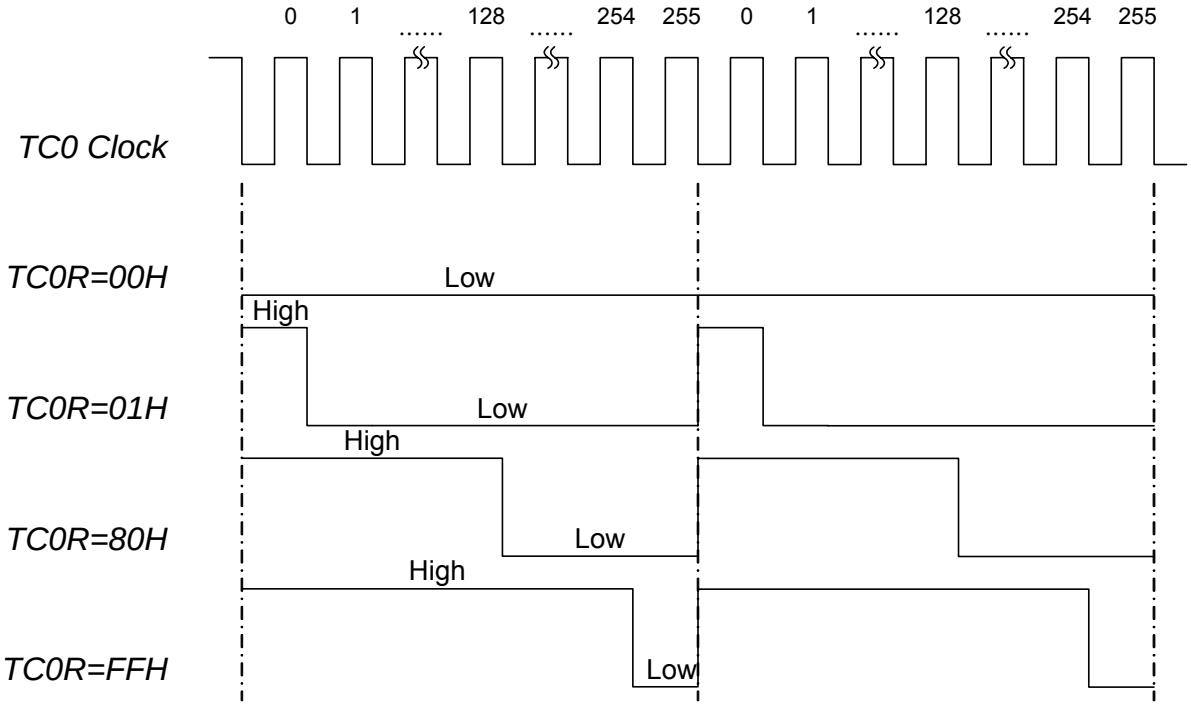
8.5 PWM0 MODE

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.

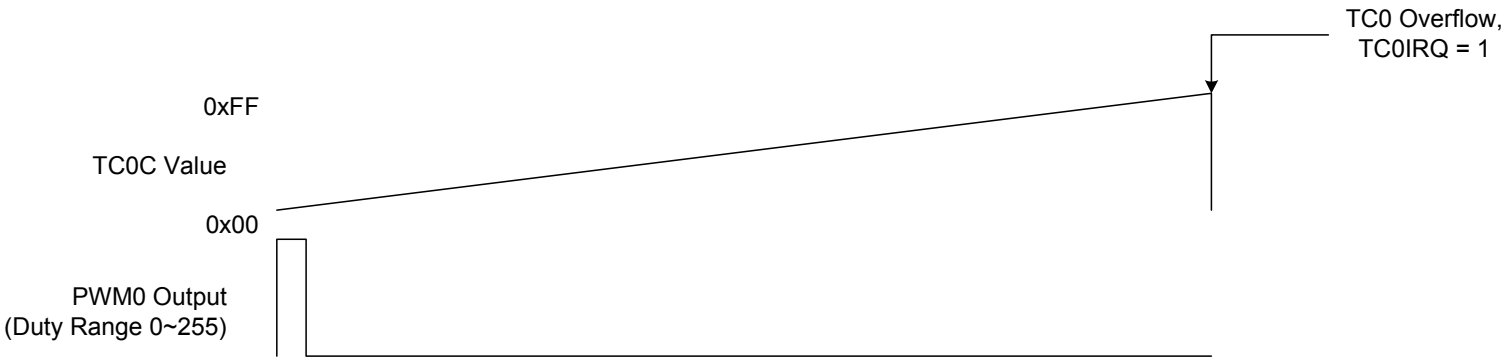
PWM duty range	TC0C valid value	TC0R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count

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.**

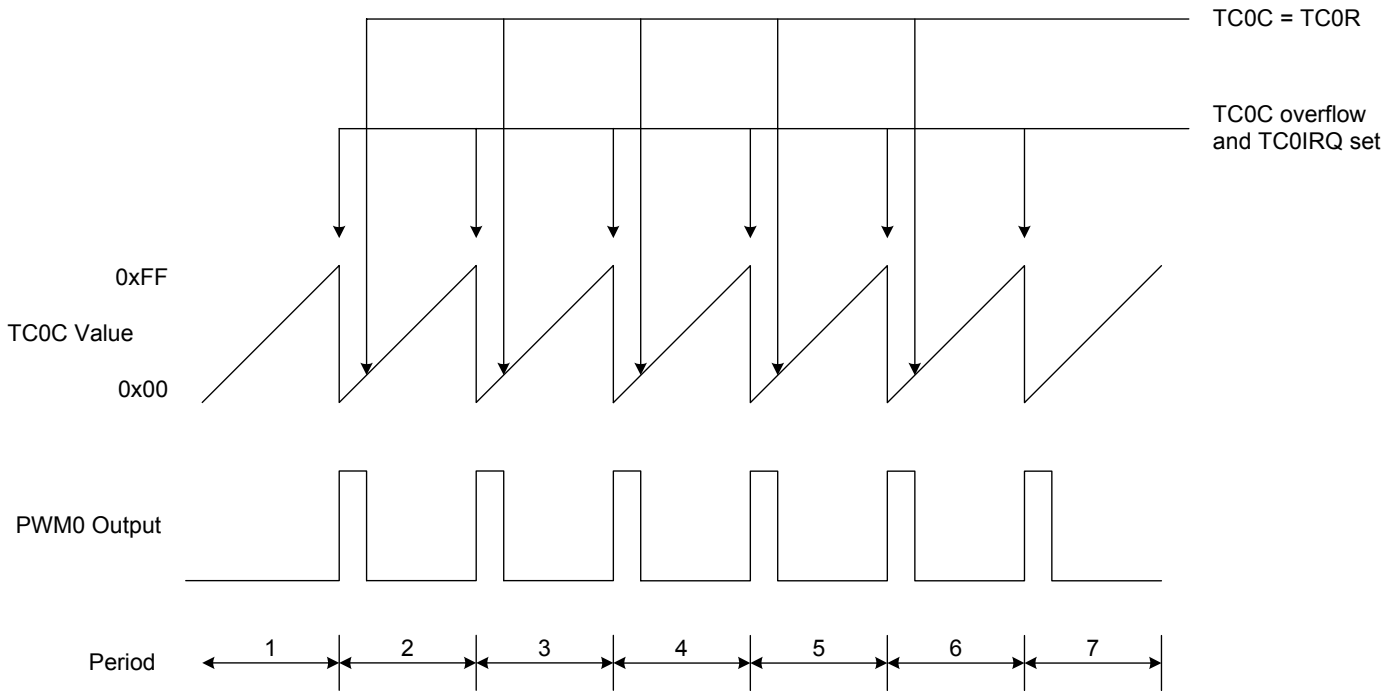
```
MOV       A, #30H         ; Input a number using B0MOV instruction.
B0MOV     TC0R, A

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.***

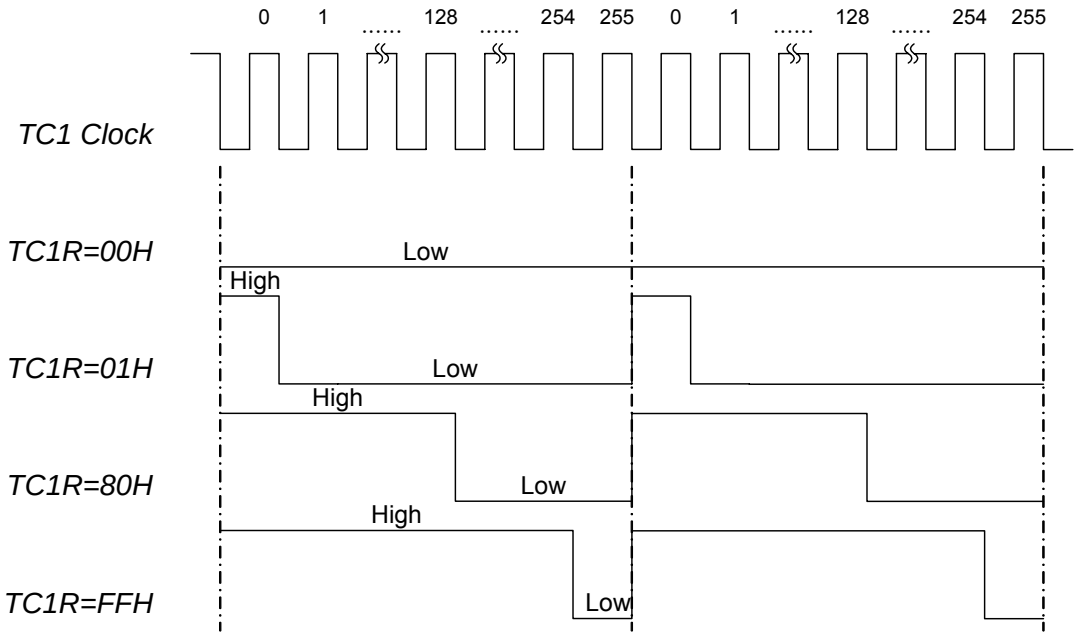
8.6 PWM1 MODE

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,

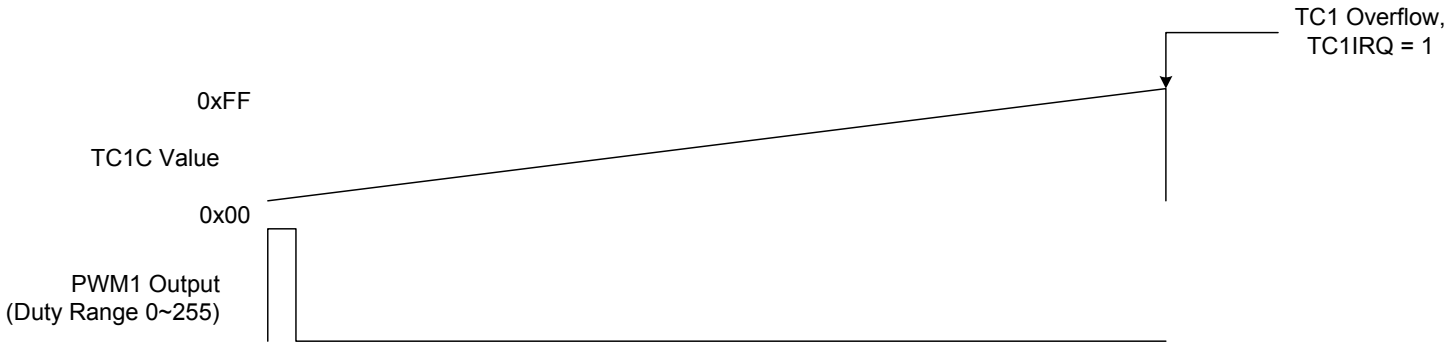
PWM duty range	TC1C valid value	TC1R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count

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.**

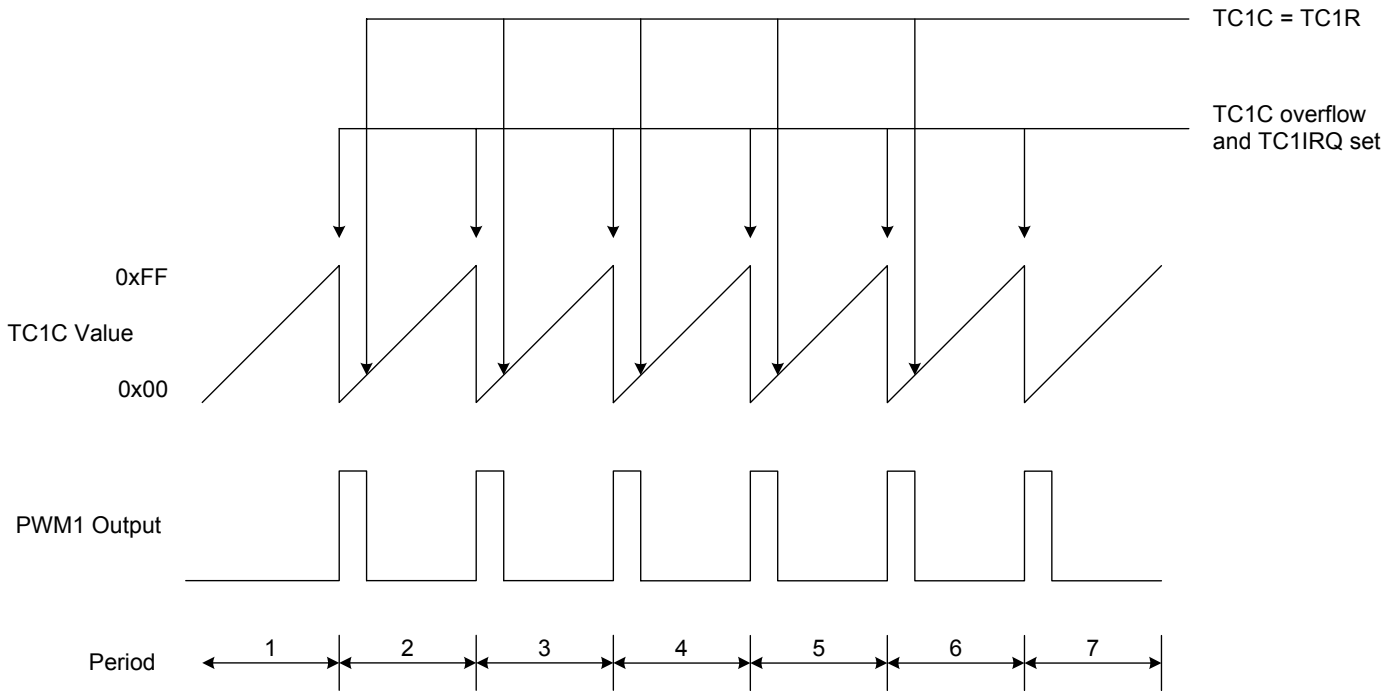
```
MOV       A, #30H         ; Input a number using B0MOV instruction.
B0MOV     TC1R, A

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.***

9 LCD DRIVER

There are 4 common pins and 24 segment pins in the SN8P1929. The LCD scan timing is 1/4 duty and 1/2 OR 1/3 bias structure to yield 96 dots LCD driver.

9.1 LCDM1 REGISTER

LCDM1 register initial value = 000x 00xx

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM1	LCDREF1	LCDREF0	LCDBNK	-	LCDENB	LCDBIAS	-	-
R/W	R/W	R/W	R/W	-	R/W	R/W	-	-
After Reset	0	0	0	-	0	0	-	-

Bit[7:6] **LCDREF[0,1]: :** Selective range of resistance for LCD Bias Voltage-division.
00 = 400k resistance
01 = 200k resistance
10 = 100k resistance
11 = 50k resistance

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
1 = Enable.

Bit2 **LCDBIAS:** LCD Bias Selection Bit
0 = LCD Bias is 1/3 Bias
1 = LCD Bias is 1/2 Bias

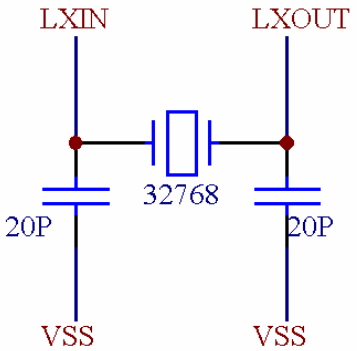
9.2 OPTION REGISTER DESCRIPTION

OPTION initial value = xxxx xxx0

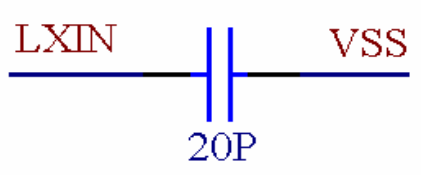
088H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPTION	-	-	-	-	-	-	-	RCLK
R/W	-	-	-	-	-	-	-	R/W
After Reset	-	-	-	-	-	-	-	0

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.***



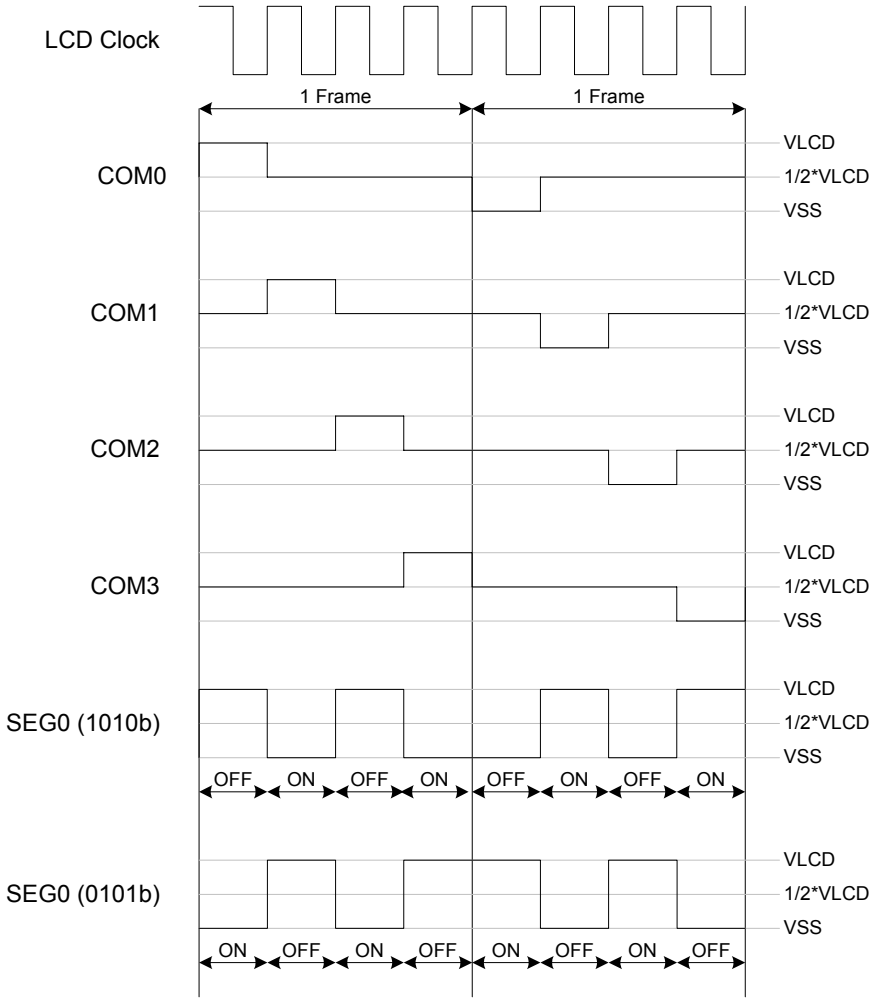
➢ ***Note2: Circuit diagram when "RCLK=1" will enable external Low Clock sets as RC mode.***



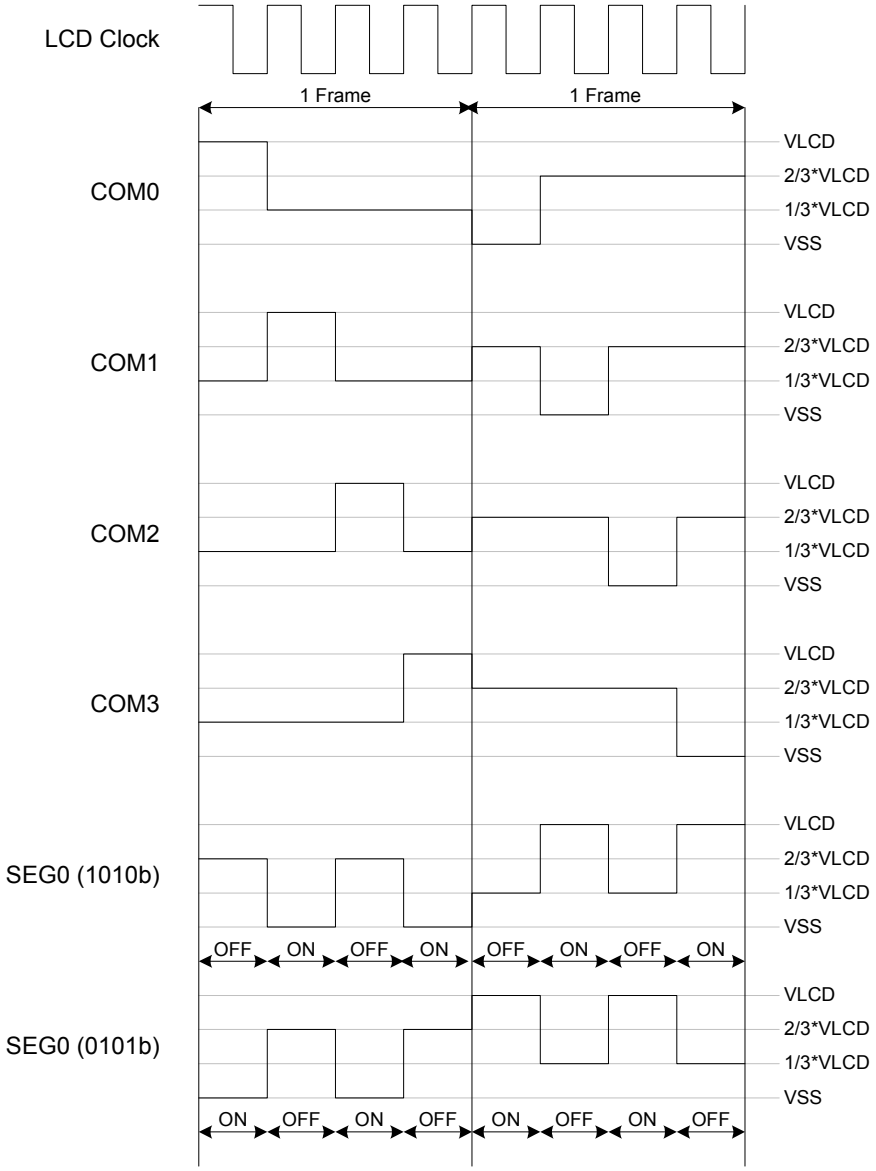
- ***Connect the C as near as possible to the VSS pin of micro-controller. The frequency of external low RC is decided by the capacitor value. Adjust capacitor value to about 32KHz frequency.***
- ***LCD frame rate is supplied from external Low clock and frame rate is 64Hz (32768Hz/512)***

9.3 LCD TIMING

LCD frame rate is always supplied from external Low clock and frame rate is 64Hz (32768Hz/512)



LCD Drive Waveform, 1/4 duty, 1/2 bias



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 23	17H.0	17H.1	17H.2	17H.3	-	-	-	-

➢ **Example: Enable LCD function.**

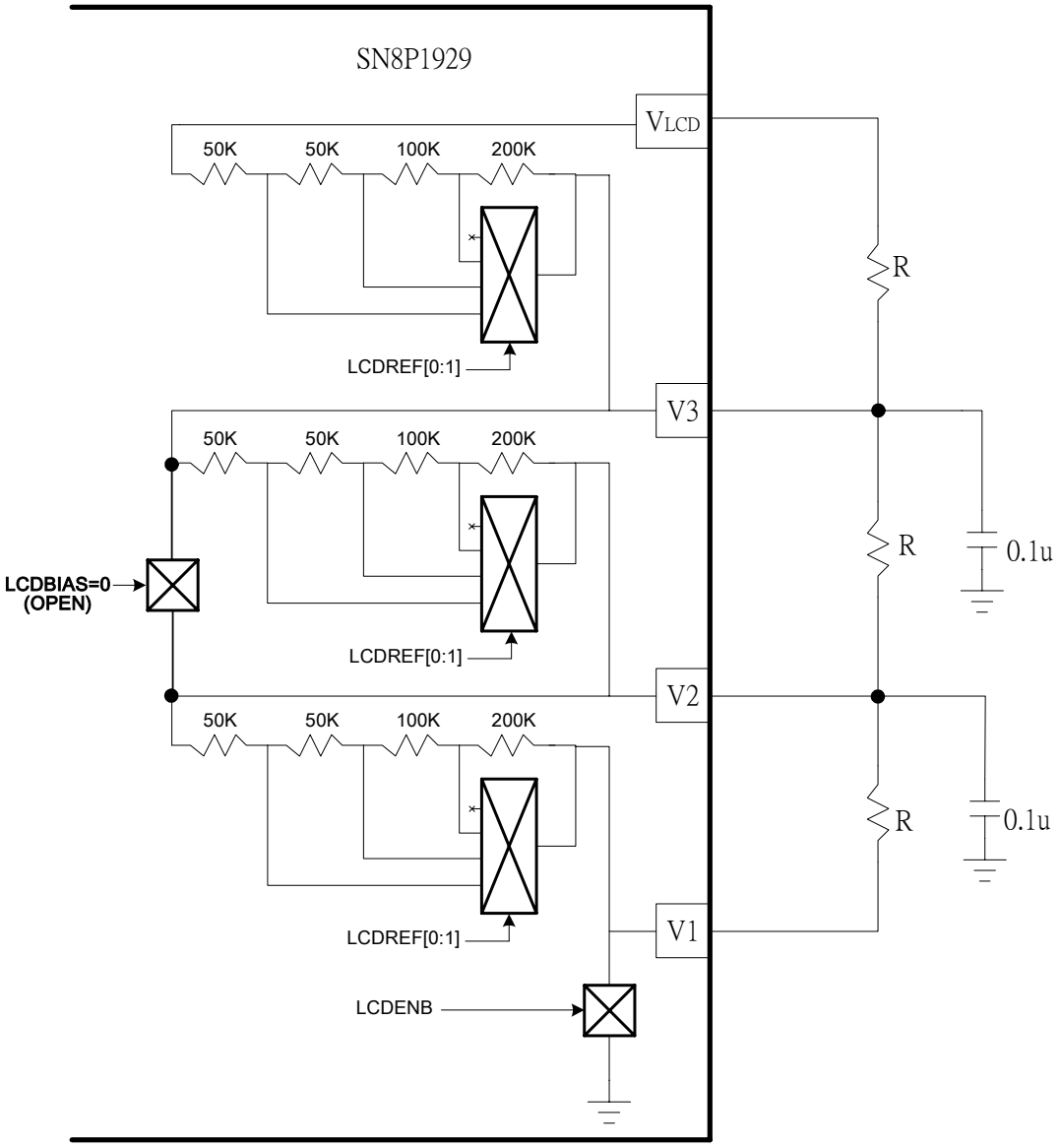
Set the LCD control bit (LCDENB) and program LCD RAM to display LCD panel.

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

9.5 LCD Circuit

SN8P1929 in the LCD electric circuit, builds in selective resistance for Voltage-division. User can add resistance between VLCD / V3 / V2 / V1 for more driving current.
Build in register can be selected in four resistor value, 400K, 200K, 100K, and 50K controlled by LCDREF0 and LCDREF1 of the OPTION register
V1, V2, V3 only available for Dice form and LQFP80 package

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

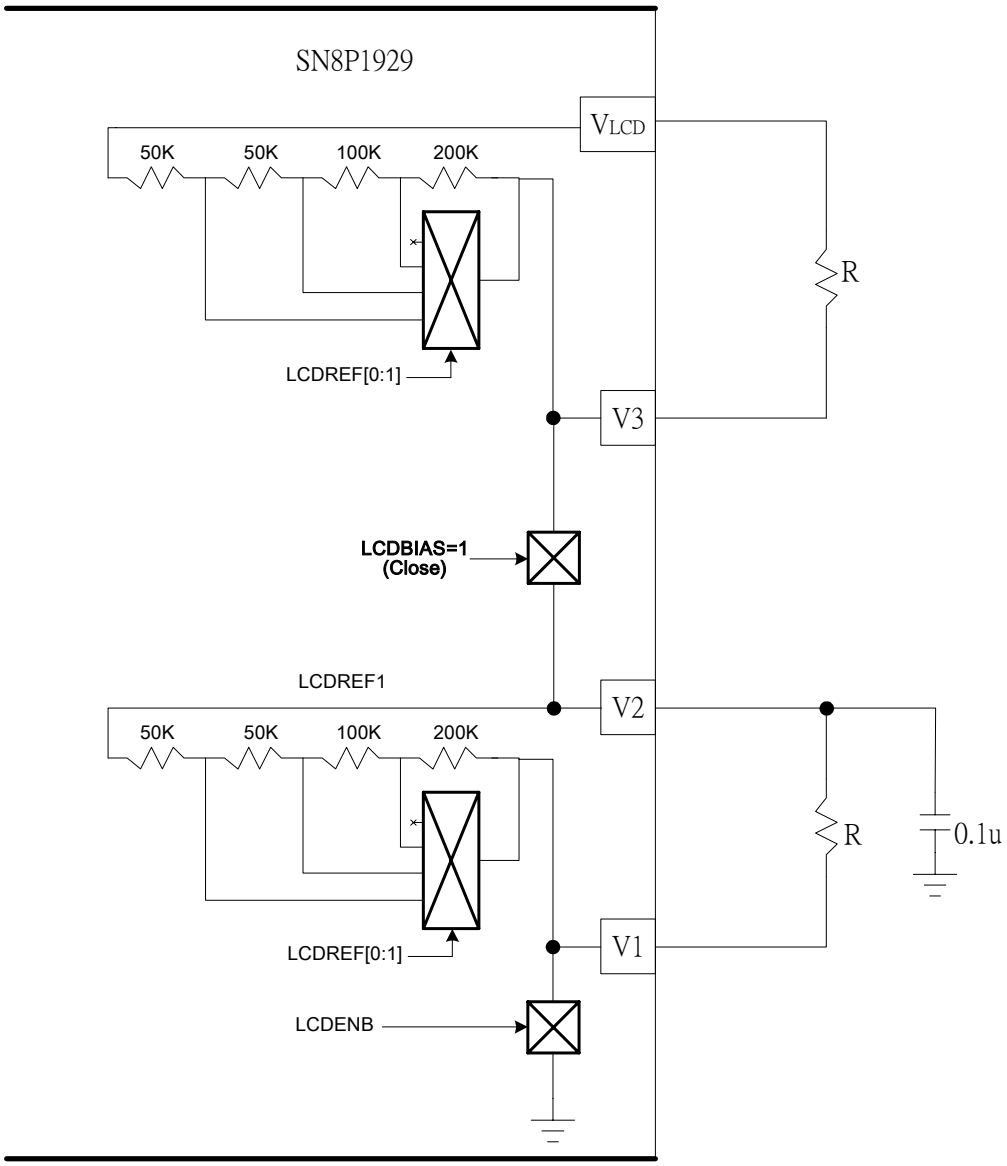


$$\text{LCD Current consumption} = \frac{\text{VLCD}}{\left(\frac{400\text{K}\times\text{R}}{400\text{K}+\text{R}}\right)\times 3}, \text{when LCDREF}[0{:}1]=00$$

Note: If used external resister, the LCD current consumption from VLCD always existence, even under power down mode.

Note: V2=1/3*VLCD、V3=2/3*VLCD。

LCD Drive Waveform, 1/4 duty, 1/2 bias



$$\text{LCD Current consumption} = \frac{\text{VLCD}}{\left(\frac{400\text{K}\times\text{R}}{400\text{K}+\text{R}}\right)\times 2}, \text{ when LCDREF[0:1]} = 00$$

Note: If used external resister, the LCD current consumption from VLCD always existence, even under power down mode.

Note: V2=V3=2/3*VLCD。

10 IN SYSTEM PROGRAM ROM

10.1 OVERVIEW

In-System-Program ROM (ISP ROM), provided user an easy way to storage data into Read-Only-Memory. Choice any ROM address and executing ROM programming instruction – ROMWRT and supply 12.5V voltage on VPP/RST pin, after programming time which controlled by ROMCNT, ROMDAH/ROMDAL data will be programmed into address ROMADRH/ROMADRL.

10.2 ROMADRH/ROMADRL REGISTER

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMADRH	VPPCHK	ROMADR14	ROMADR13	ROMADR12	ROMADR11	ROMADR10	ROMADR9	ROMADR8
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMADRL	ROMADR7	ROMADR6	ROMADR5	ROMADR4	ROMADR3	ROMADR2	ROMADR1	ROMADR0
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

VPPCHK: VPP pin Programming Voltage. Check
0 = VPP's Voltage NOT reached 12.5V. Can't program ISP ROM
1 = VPP's Voltage reached 12.5V. Can program ISP ROM

✷ ***Note 2: Using Marco @B0BTS1_FVPPCHK and @B0BTS0_FVPPCHK for checking VPP voltage status.***

ROMADR[14:0] : ISP ROM Programming Address.
ROM Address which will be Programmed

10.3 ROMDAH/ROMADL REGISTERS

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMDAH	ROMDA15	ROMDA14	ROMDA13	ROMDA12	ROMDA11	ROMDA10	ROMDA9	ROMDA8
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

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMDAL	ROMDA7	ROMDA6	ROMDA5	ROMDA4	ROMDA3	ROMDA2	ROMDA1	ROMDA0
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

ROMDA[15:0] : ISP ROM Programming Data
ROM Data which want to Programming into ROM area..

10.4 ROMCNT REGISTERS and ROMWRT INSTRUCTION

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMCNT	ROMCNT7	ROMCNT6	ROMCNT5	ROMCNT4	ROMCNT3	ROMCNT2	ROMCNT1	ROMCNT0
Read/Write	W	W	W	W	W	W	W	W
After reset	-	-	-	-	-	-	-	-

Bit[7:0] ROMCNT**[7:0]:** ISP ROM Programming Time Counter
The ISP ROM Programming Time was controlled by ROMCNT[7:0]
Programming will be (256-ROMCNT)*4/Fcpu
The Suggestion Programming is 1ms

Fcpu	ROMCNT	Programming Time
1MIPs	6	1ms

When all setting was done, execute **ROMWRT** instruction to program data ROMDA[15:0] into address ROMADR[14:0]

- ***Note1: Please Keep VDD=5V when accessing ISP ROM.***
- ***Note2: After access ROMWRT, at least 3 NOP instruction delay is necessary.***
- ***Note3:Please executing ISP function in room temperature(25°C)***

10.5 ISP ROM ROUTINE EXAMPLE

➢ **Example :**

```
; Reserved ISP ROM Area as 0xFFFF
ORG             0100H
@CALDATA:
                DW              0xFFFF
                ……..
                ……..
; Program Data 0xAA55 into address @CALDATA
                MOV             A, #@CALDATA$L
                B0MOV           ROMADRL, A          ;Move Low Byte Address to ROMADRL
                MOV             A, #@CALDATA$H
                B0MOV           ROMADRH, A          ;Move Low Byte Address to ROMADRH
                MOV             A, #0X55
                B0MOV           ROMDAL, A           ;Move Low Byte Data to ROMDAL
                MOV             A, #0XAA
                B0MOV           ROMDAH, A           ;Move Low Byte Data to ROMADRH
;VPP Voltage Check
                @B0BTS1_ FVPPCHK                    ;Check VPP Voltage is 12.5V or not
                JMP             $-1                 ;If VPP not reach 12.5V, Keep waiting.

;Set programming counter and Accessing ISP ROM
@ROM_WRT:       MOV             A,#6                ;Set Programming Counter
                B0MOV           ROMCNT,A
                ROMWRT                              ;Programming ISP ROM
                NOP                                 ;NOP Delay
                NOP                                 ;NOP Delay
                NOP                                 ;NOP Delay
;VPP Voltage Check                                  ;Set VPP as VDD voltage.
                @B0BTS0_ FVPPCHK                    ;Check VPP Voltage is VDD or not
                JMP             $-1                 ;If VPP still reach 12.5V, Keep waiting.

;Check Programmed Data
                B0MOV           Z, #@CALDATA$L
                B0MOV           Y, #@CALDATA$H
                MOVC                                ;MOVE ISP ROM Data into A and R

                CMPRS           A,#0x55
                JMP             @WRT_ERR
                B0MOV           A, R
                CMPRS           A,#0xAA
                JMP             @WRT_ERR            ;Check ISP ROM Data Correction.

                ……..
```

11 Charge-Pump, PGIA and ADC

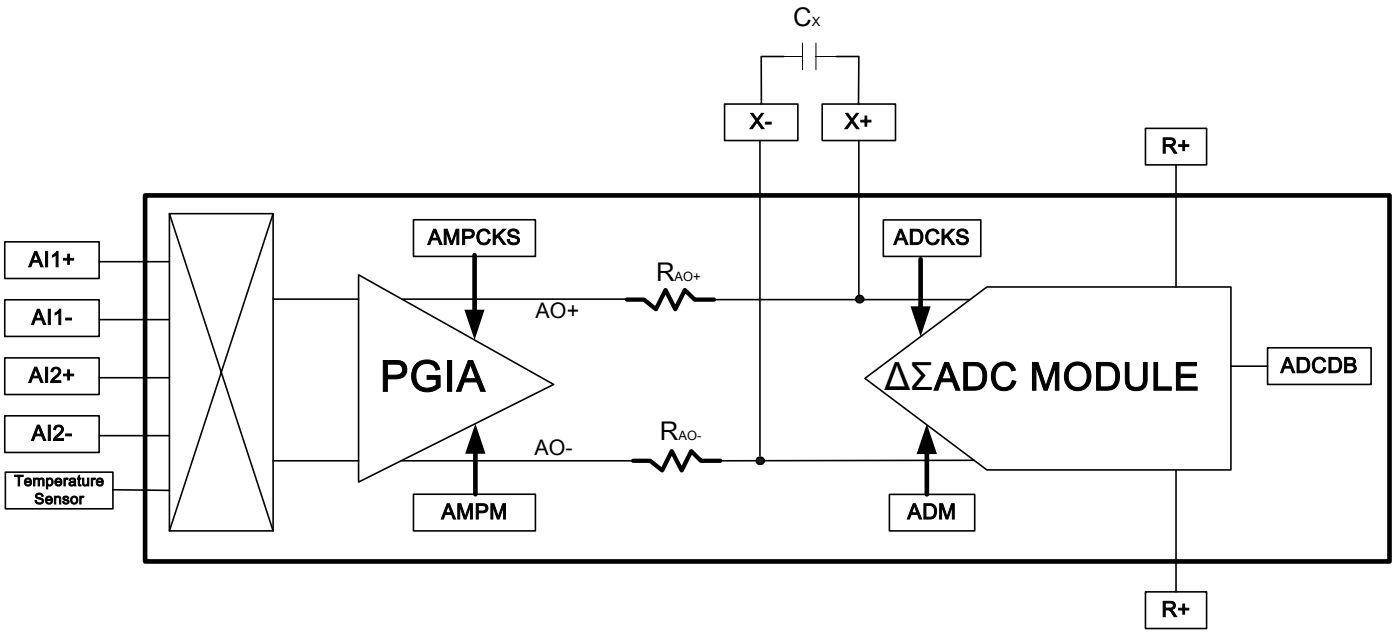
11.1 OVERVIEW

The SN8P1929 has a built-in Voltage Charge-Pump/Regulator (CPR) to support a stable voltage 3.8V from pin AVDDR and 3.0V/2.4v/1.5V from pin AVE+ with maximum 10mA current driving capacity. This CPR provides stable voltage for internal circuits (PGIA, ADC from AVDDR) and external sensor (load cell or thermistor from AVE+). The SN8P1929 series also integrated Δ Σ Analog-to-Digital Converters (ADC) to achieve 16-bit performance and up to 62500-step resolution. The ADC has THREE different input channel modes: (1) Two fully differential inputs (2) One fully differential input and Two single-ended inputs (3) Four single-ended 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.

11.2 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 Δ Σ 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 ADCM register and GS[2:0] bits (Gain Selection) in AMPM register.



Block Diagram of PGIA/ADC module

- ***Note 1: The low pass filter (C_X) will filter out chopper frequency of PGIA.***
- ***Note 2: The recommend value of C_X is 0.1 μF . This capacitor needs to place as close chip as possible.***

11.3 Voltage Charge Pump / Regulator (CPR)

SN8P1929 is built in a CPR, which can provide a stable 3.8V (pin AVDDR) and 3.0V/2.4V/1.5V (pin AVE+) with maximum 10mA current driving capacity. Register CPM can enable or disable CPR and controls CPR working mode, another register CPCKS sets CPR working clock to 4KHz. 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 AVDDCP. In addition, the CP will need at least 10ms for output voltage stabilization after set CPRENB to high.

11.3.1 CPM-Charge Pump Mode Register

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPM	ACMENB	AVDDRENB	AVENB	AVESEL1	AVESEL0	CPAUTO	CPON	CPRENB
R/W	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

Bit0: **CPRENB:** Charge Pump / Regulator function enable control bit.
0 = Disable charge pump and regulator,
1 = Enable charge pump and regular.

Bit1: **CPON:** Change Pump always ON function control bit (CPRENB must = “1”)
0 = Charge Pump On / Off controlled by bit CPAUTO.
1 = Always turn ON the charge pump regulator.

Bit2: **CPAUTO:** Charge Pump Auto Mode function control bit
0 = Disable charge pump auto mode.
1 = Enable charge pump auto mode.

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 Regulator and AVDDR Output voltage 3.8V
1 = Enable Regulator and 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 CPRENB = “1”.***
- ***Note2: All current consumptions from AVDDR and AVE+ (including PGIA and ADC) will time 2, when Charge Pump was Enabled.***
- ***Note3: Before Enable Charge pump/Regulator , Must enable Band Gap Reference (BGRENB=1) first.***
- ***Note4 Before Enable ACM voltage, Enable AVDDR voltage first.***
- ***Note5: Before Enable PGIA and ADC , Must enable Band Gap Reference (BGRENB=1), ACM (ACMENB=1) and AVDDR(AVDDRENB).***
- ***Note6: CPR can work in slow mode, but CPCKS, AMPCKS register value must be reassigned.***

Bit CPRENB, CPON, and CPAUTO are Charge-Pump working mode control bit. By these three bits, Charge-Pump can be set as OFF, Always ON, or Auto mode.

CPRENB	CPON	CPAUTO	AVDDRENB	Charge-Pump Status	Regulator Status	AVDDR	PGIA, ADC Function
0	X	X	0	OFF	OFF	0V	**Not Available**
1	0	0	1	OFF	ON	**See Note1**	**See Note1**
1	0	1	1	Auto Mode	ON	3.8V	Available
1	1	0	1	Always ON	ON	3.8V	Available

In Auto Mode, Charge-Pump ON/OFF depended on VDD voltage.

Auto-Mode Description:

CPRENB	CPON	CPAUTO	AVDDRENB	VDD	Charge-Pump Status	Regulator Status	AVDDR Output	PGIA, ADC Function
1	0	1	1	>4.1V	OFF	ON	3.8V	Available
				≦4.1V	ON	ON	3.8V	Available

✸ ***Note 1: When Charge-Pump is OFF and Regulator is ON, VDD voltage must be higher than 4.1V to make sure AVDDR output voltage for PGIA, and ADC functions are working well.***

CPRENB	CPON	CPAUTO	AVDDRENB	VDD	Charge-Pump Status	Regulator Status	AVDDR Output	PGIA, ADC Function
1	0	0	1	>4.1V	OFF	ON	3.8V	Available
				≦4.1V	OFF	ON	**VDD**	**Not Available**

✸ ***Note 1: For normally application, set CP as Auto mode (CPAUTO = 1) is strongly recommended.***
✸ ***Note 2: If VDD is higher than 5.0V, don't set Charge-Pump as Always ON mode.***
✸ ***Note 3: Band Gap Reference voltage must be enable first (FBRGENB), before following function accessing: (Reference AMPM register for detail information)***
(1) Charge pump /Regulator.
(2) PGIA function.
(3) 16- bit ADC function.
(4) Low Battery Detect function

11.3.2 CPCKS-Charge Pump Clock Register

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPCKS					CPCKS3	CPCKS2	CPCKS1	CPCKS0
R/W					W	W	W	W
After Reset					0	0	0	0

CPCKS [3:0] register sets the Charge-Pump working clock; the suggestion Charge-Pump clock is 13K~15K Hz.@ Normal mode, 2K@Slow mode

Charge-Pump Clock= Fcpu / 4 / (2^CPCKS[3:0])

Refer to the following table for CPCKS [3:0] register value setting in different Fosc frequency.

CPCKS3	CPCKS2	CPCKS1	CPCKS0	Fosc				
				32768Hz	2M	3.58M	4M/IHRC	8M
0	0	0	0	2.048K	125K	223.75K	250K	500K
0	0	0	1	NA	62.5K	111.88K	125K	250K
0	0	1	0	NA	31.25K	55.94K	62.5K	125K
0	0	1	1	NA	15.625K	27.97K	31.25K	62.5K
0	1	0	0	NA	7.8125K	13.985K	15.625K	31.25K
0	1	0	1	NA	3.90625K	6.99K	7.8125K	15.625K
0	1	1	0	NA	1.953215K	3.495K	3.90625K	7.8125K
0	1	1	1	NA	0.976K	1.75K	1.953215K	3.90625K
1	0	0	0	NA	0.488K	0.875K	0.976K	1.953215K
1	0	0	1	NA	0.244K	0.438K	0.488K	0.976K
1	0	1	0	NA	0.122K	0.219K	0.244K	0.488K
1	0	1	1	NA	0.61K	0.11K	0.122K	0.244K
1	1	0	0	NA	0.3K	0.055K	0.061K	0.122K
1	1	0	1	NA	0.15K	0.028K	0.03K	0.61K
1	1	1	0	NA	0.075K	0.014K	0.015K	0.3K
1	1	1	1	NA	0.037K	0.007K	0.008K	0.15K

- ***Note1: When enable charge pump, Set Charge pump clock as "1011" to avoid VDD dropped.***
- ***Note2: In general application, CP working clock is about 13K~15K Hz in normal mode, 2K Hz in slow mode (External Low Clock mode).***
- ***Note3: The Faster of Charge pump clock, AVE+ can load more current.***
- ***Note4: In slow mode or Green mode, Set CPCKS=0x00 for AVDDR/AVE+/ACM can supply the max current.***

Example: Charge-Pump setting (Fosc = 4M X'tal)

```
@CPREG_Init:
            XB0BSET     FBGRENB     ;Enable Band Gap Reference voltage.

            MOV         A, #00001011b
            XB0MOV      CPCKS, A    ; Set CPCKS as slowest clock to void VDD dropping.

            MOV         A, #00011100B   ;
            XB0MOV      CPM, A      ; Set AVE+=3.0V ,CP as Auto mode and Disable AVDDR,
                                    AVE+, ACM voltage before enable Charge pump

@CP_Enable:
            XB0BSET     FCPRENB     ; Enable Charge-Pump
            CALL        @Wait_200ms ; Delay 200ms for Charge-Pump Stabilize

            MOV         A, #0000100b
            XB0MOV      CPCKS, A    ; Set CPCKS as 15.6K for 10mA current loading.
            CALL        @Wait_100ms ; Delay 100ms for Voltage Stabilize

@AVDDR_Enable:
            XB0BSET     FAVDDRENB   ; Enable AVDDR Voltage=3.8V
            CALL        @Wait_10ms  ; Delay 10ms for AVDDR Voltage Stabilize

@ACM_Enable:
            XB0BSET     FACMENB     ; Enable ACM Voltage=1.2v
            CALL        @Wait_5ms   ; Delay 5ms for ACM Voltage Stabilize

@AVE_Enable:
            XB0BSET     FAVENB      ; Enable AVE+ Voltage=3.0V/2.4V/1.5V
            CALL        @Wait_10ms  ; Delay 10ms for AVE+ Voltage Stabilize

            …
            …
```

- ***Note1: The Charge pump delay (200ms and 100ms) can avoid VDD drop when CR2032 battery application. If VDD source came from AA or AAA dry battery, the delay time can be shorten to 50ms.***
- ***Note2: Please refer the SN8P1929 EV_Board manual for the detail XB0MOV, XB0BSET command.***

11.4 PGIA -Programmable Gain Instrumentation Amplifier

SN8P1929 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) Two fully differential input (2) One fully differential input and Two single-ended inputs (3) Four single-ended inputs, it was defined by register AMPCHS.

11.4.1 AMPM- Amplifier Mode Register

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

Bit7: **CHPENB:** Chopper clock Enable control pin
0 = Disable Chopper clock- Chopper clock set to High .
1 = Enable Chopper clock

✸ ***Note: Set CHPENB=1 for all applications.***

11.4.2 AMPCKS- PGIA CLOCK SELECTION

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPCKS	-	-	-	-	-	AMPCKS1	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/IHRC	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 Internal Low clock mode.***

11.4.3 AMPCHS-PGIA CHANNEL SELECTION

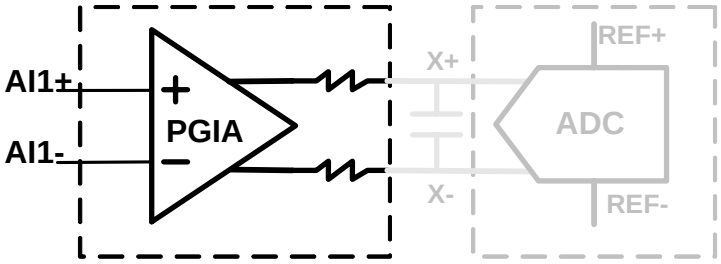
091H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPCHS	-	-	-	-	CHS3	CHS2	CHS1	CHS0
R/W	-	-	-	-	R/W	R/W	R/W	R/W
After Reset	-	-	-	-	0	0	0	0

CHS [3:0]: PGIA Channel Selection

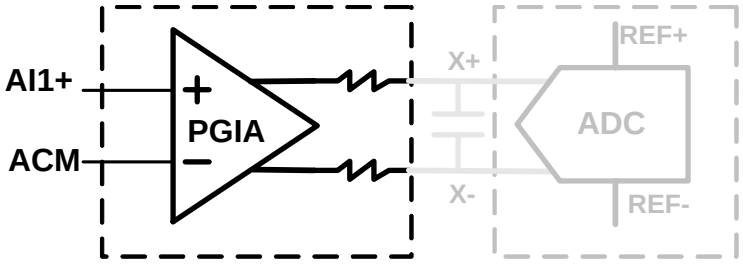
CHS [3:0]	Selected Channel	V (X+, X-) Output	Input-Signal Type
0000	AI1+, AI1-	V (AI1+, AI1-) × PGIA Gain	Differential
0001	AI2+, AI2-	V (AI2+, AI2-) × PGIA Gain	Differential
0010	AI1+, ACM	V (AI1+, ACM) × PGIA Gain	Single-ended
0011	AI1-, ACM	V (AI1-, ACM) × PGIA Gain	Single-ended
0100	AI2+, ACM	V (AI2+, ACM) × PGIA Gain	Single-ended
0101	AI2-, ACM	V (AI2-, ACM) × PGIA Gain	Single-ended
0110	ACM, ACM	V (ACM, ACM) × PGIA Gain	Input-Short
0111	Reserved	N/A	N/A
1000	Temperature Sensor	V (V_{TS}, 0.8V) × 1	N/A
Others	Reserved	N/A	N/A

- ***Note 1: V (AI+, AI-) = (AI+ voltage - AI- voltage)***
- ***Note 2: V (AI-, ACM) = (AI- voltage - ACM voltage)***
- ***Note 3: The purpose of Input-Short mode is only for PGIA offset testing.***
- ***Note 4: When CPR 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.***

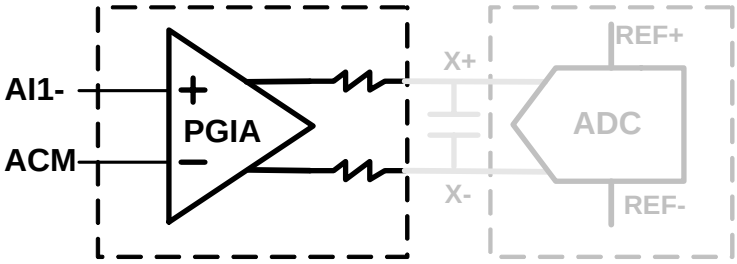
AMPCHS[3:0]="0000"



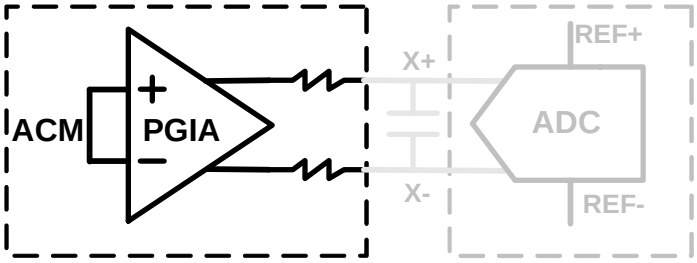
AMPCHS[3:0]="0010"



AMPCHS[3:0]="0011"

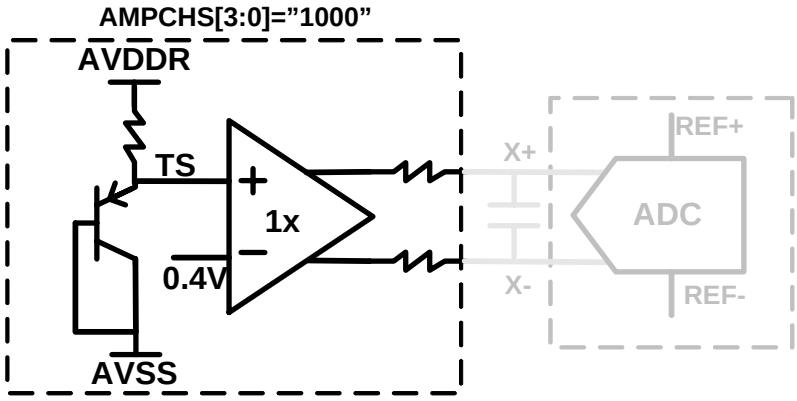


AMPCHS[2:0]="0110"



11.4.4 Temperature Sensor (TS)

In applications, sensor characteristic might change in different temperature also. To get the temperature information, SN8P1929 build in a temperature senor (TS) for temperature measurement. Select the respective PGIA channel to access the Temperature Sensor ADC output.



- ***Note1: When selected Temperature Sensor, PGIA gain must set to 1x, or the result will be incorrect.***
- ***Note2: Under this setting, X+ will be the V(TS) voltage, and X- will be 0.8V.***
- ***Note3: The Temperature Sensor was just a reference data not real air temperature. For precision application, please use external Thermister sensor.***

In 25C, V(TS) will be about 0.8V, and if temperature rise 10C, V(TS) will decrease about 15mV, if temperature drop 10C, V(TS) will increase about 15mV,

Example:

Temperature	V(TS)	V(REF+,REF-)	ADC output
15	0.815V	0.8V	16211
25	0.800V	0.8V	15625
35	0.785V	0.8V	15039

By ADC output of V(TS), can get temperature information and compensation the syste.

- ***Note1: The V(TS) voltage and temperature curve of each chip might different. Calibration in room temperature is necessary when application temperature sensor.***
- ***Note2: The typical temperature parameter of Temperature Sensor is 1.5mV/C.***

Example: PGIA setting (Fosc = 4M X'tal)

```
@CPREG_Init:
            XB0BSET     FBGRENB         ;Enable Band Gap Reference voltage.

            MOV         A, #00001011b
            XB0MOV      CPCKS, A        ; Set CPCKS as slowest clock to void VDD dropping.

            MOV         A, #00011100B   ;
            XB0MOV      CPM, A          ; Set AVE+=3.0V ,CP as Auto mode and Disable AVDDR, AVE+,
                                        ACM voltage, before enable Charge pump

@CP_Enable:
            XB0BSET     FCPRENB         ; Enable Charge-Pump
            CALL        @Wait_200ms     ; Delay 200ms for Charge-Pump Stabilize

            MOV         A, #0000100b
            XB0MOV      CPCKS, A        ; Set CPCKS as 15.6K for 10mA current loading.
            CALL        @Wait_100ms     ; Delay 100ms for Voltage Stabilize

@AVDDR_Enable:
            XB0BSET     FAVDDRENB       ; Enable AVDDR Voltage=3.8V
            CALL        @Wait_10ms      ; Delay 10ms for AVDDR Voltage Stabilize

@ACM_Enable:
            XB0BSET     FACMENB         ; Enable ACM Voltage=1.2v
            CALL        @Wait_5ms       ; Delay 5ms for ACM Voltage Stabilize

@AVE_Enable:
            XB0BSET     FAVENB          ; Enable AVE+ Voltage=3.0V/2.4V/1.5V
            CALL        @Wait_10ms      ; Delay 10ms for AVE+ Voltage Stabilize

@PGIA_Init:
            MOV         A, #11110110B
            XB0MOV      AMPM, A         ; Enable Band Gap, Set :FDS="11" ,CHPENB=1, PGIA Gain=200
            MOV         A, #00000100B
            XB0MOV      AMPCKS, A       ; Set AMPCKS = "100" for PGIA working clock = 1.9K @ 4M X'tal
            MOV         A, #00h
            XB0MOV      AMPCHS, A       ; Selected PGIA differential input channel= AI1+, AI1-
@PGIA_Enable:
            XB0BSET     FAMPENB         ; Enable PGIA function
            …                           ; V (X+, X-) Output = V (AI1+, AI1-) x 200
```

- ***Note 1: Enable Charge-Pump/Regulator before PGIA working***
- ***Note 2: Please set PGIA relative registers first, then enable PGIA function bit.***

Example: PGIA channel change:

```
@PGIA_Init:
        MOV     A, #11110110B
        XB0MOV  AMPM, A         ; Enable Band Gap, Set :FDS="11" ,CHPENB=1, PGIA Gain=200
        MOV     A, #00000100B
        XB0MOV  AMPCKS, A       ; Set AMPCKS = "100" for PGIA working clock = 1.9K @ 4M X'tal
        MOV     A, #00000000B
        XB0MOV  AMPCHS, A       ; Selected PGIA differential input channel= AI1+, AI1-
@PGIA_Enable:
        XB0BSET FAMPENB         ; Enable PGIA function
        …                       ; V (X+, X-) Output = V (AI1+, AI1-) x 200

@PGIA_Sensor:
        MOV     A, #11110111B   ;Don't Disable PGIA when change PGIA CH.
        XB0MOV  AMPM, A         ; Enable Band Gap, Set FDS="11", CHPENB=1, PGIA Gain=200
        MOV     A, #00000001B
        XB0MOV  AMPCHS, A       ; Selected PGIA as Differential channel.
        …                       ; V (X+, X-) Output = V(AI2+,AI2-) x 200

@PGIA_TS:
        MOV     A, #11110001B   ;Don't Disable PGIA when change PGIA CH.
        XB0MOV  AMPM, A         ; Enable Band Gap, Set FDS="11", CHPENB=1, PGIA Gain=1x
        MOV     A, #00001000B
        XB0MOV  AMPCHS, A       ; Selected PGIA as Temperature Sensor ch.
        …                       ; V (X+, X-) Output = V (TS, 0.4) x 1.
```

11.5 16-Bit ADC

11.5.1 ADCM- ADC Mode Register

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCM	-	-	-	-	IRVS	RVS1	RVS0	ADCENB
R/W	-	-	-	-	R/W	R/W	R/W	R/W
After Reset	-	-	-	0	0	0	0	0

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

Bit1: **RVS 0**: ADC Reference Voltage Selection bit
0 = Selection ADC as normal operation from X+,X-.
1 = Selection ADC as VDD voltage detect

Bit2: **RVS 1**: ADC Reference Voltage Selection bit 1
0 = Selection ADC Reference voltage from **External** reference R+,R-.
1 = 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**
X	0	1	-	R+	R-	VDD *3/16	VDD* 2/16	**External Ref. Voltage**
0	1	1	11 (AVE+=3.0V)	0.8V	0.4V			(AVE+=3.0V)
1	1	1		1.2V	0.4V			
0	1	1	10 (AVE+=2.4V)	0.64V	0.32V			(AVE+=2.4V)
1	1	1		0.96V	0.32V			

✸ ***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.***

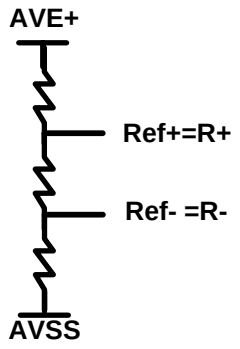
✸ ***Note2: The Internal Reference Voltage is divided from AVE+, so the voltage will follow the changing with AVE+(3.0V/2.4V/1.5V) which selected by AVESEL[1:0].***

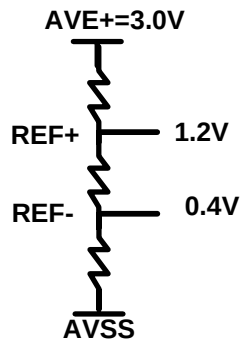
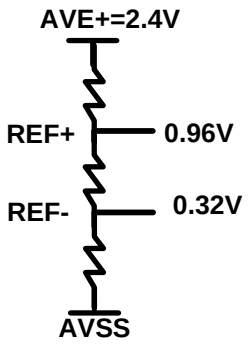
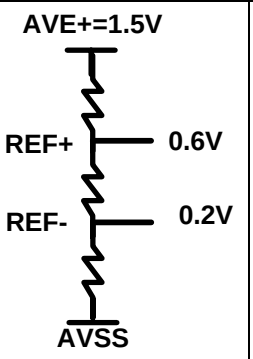
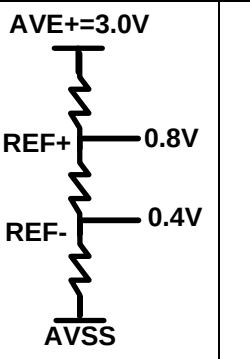
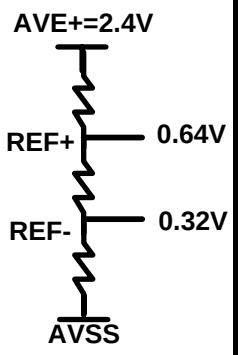
$$(ADCIN+) > (ADCIN-) \Rightarrow ADCConversionData = +\frac{(ADCIN+)-(ADCIN-)}{(REF+)-(REF-)} X\,31250$$

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

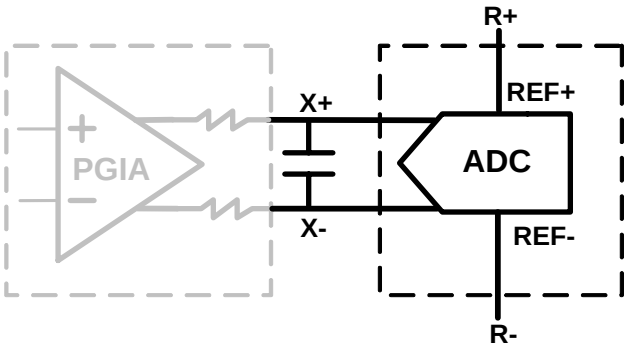
✸ ***Note: The internal reference voltage are generated from AVE+ voltage.***

External and Internal Reference Circuit Table:

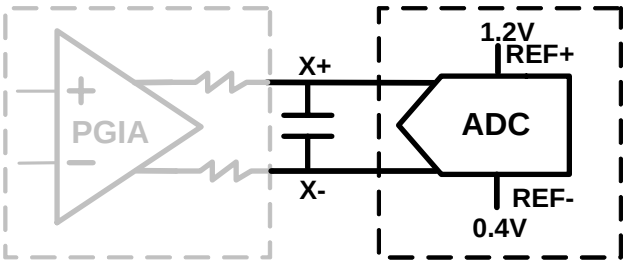
External Ref. Circuit
RVS1=0


Internal Reference Circuit				
RVS1=1,				
IRVS=1, AVE+=3.0V	IRVS=1, AVE+=2.4V	IRVS=1, AVE+=1.5V	IRVS=0,AVE+=3.0V	IRVS=0,AVE+=2.4V
				

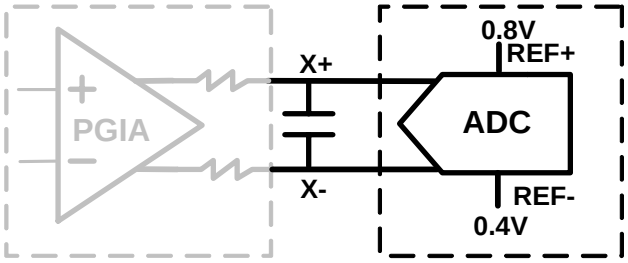
ADCM=#xxx0x00xB, V(REF+, REF-) = V(R+, R-), ADC Reference Voltage from External R+,R-.



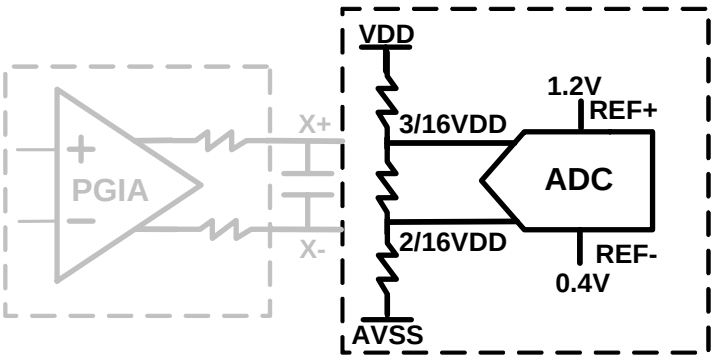
ADCM=#xxx0110xB, V(REF+, REF-) = V(1.2V, 0.4V)=0.8V (AVE+=3.0V) ADC Reference Voltage from Internal 1.2V and 0.4V.



ADCM=#xxx0010xB, V(REF+, REF-) = V(0.8V, 0.4V)=0.4V (AVE+=3.0V), ADC Reference Voltage from Internal 0.8V and 0.4V.



ADCM=#xxx0111xB, V(REF+, REF-) = V(1.2V, 0.4V)=0.8V (AVE+=3.0V), ADC Reference Voltage from Internal 1.2V and 0.4V, and ADC output is Voltage measurement result.



11.5.2 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
R/W	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.***

11.5.3 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/W	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

11.5.4 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/W	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,***
- ***Note5: Because of the ADC design limitation, the ADC Linear range is +28125~-28125 (decimal). The MAX ADC output must keep inside this range.***

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

11.5.5 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
After Reset	-	-	-	-	-	0	-	0

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.***

Example: Charge-Pump, PGIA and ADC setting (Fosc = 4M X'tal)

```
@CPREG_Init:
            XB0BSET   FBGRENB       ;Enable Band Gap Reference voltage.

            MOV       A, #00001011b
            XB0MOV    CPCKS, A      ; Set CPCKS as slowest clock to void VDD dropping.

            MOV       A, #00011100B ;
            XB0MOV    CPM, A        ; Set AVE+=3.0V ,CP as Auto mode and Disable AVDDR, AVE+,
                                    ACM voltage ,before enable Charge pump

@CP_Enable:
            XB0BSET   FCPRENB       ; Enable Charge-Pump
            CALL      @Wait_200ms   ; Delay 200ms for Charge-Pump Stabilize

            MOV       A, #0000100b
            XB0MOV    CPCKS, A      ; Set CPCKS as 15.6K for 10mA current loading.
            CALL      @Wait_100ms   ; Delay 100ms for Voltage Stabilize

@AVDDR_Enable:
            XB0BSET   FAVDDRENB     ; Enable AVDDR Voltage=3.8V
            CALL      @Wait_10ms    ; Delay 10ms for AVDDR Voltage Stabilize

@ACM_Enable:
            XB0BSET   FACMENB       ; Enable ACM Voltage=1.2v
            CALL      @Wait_5ms     ; Delay 5ms for ACM Voltage Stabilize

@AVE_Enable:
            XB0BSET   FAVENB        ; Enable AVE+ Voltage=3.0V/1.5V
            CALL      @Wait_10ms    ; Delay 10ms for AVE+ Voltage Stabilize

@PGIA_Init:
            MOV       A, #11110110B
            XB0MOV    AMPM, A       ;Enable Band Gap, Set :FDS="11" ,CHPENB=1 PGIA Gain=200
            MOV       A, #00000100B
            XB0MOV    AMPCKS, A     ; Set AMPCKS = "100" for PGIA working clock = 1.9K @ 4M X'tal
            MOV       A, #00h
            XB0MOV    AMPCHS, A     ; Selected PGIA differential input channel= AI1+, AI1-
@PGIA_Enable:
            XB0BSET   FAMPENB       ; Enable PGIA function
            …                       ; V (X+, X-) Output = V (AI1+, AI1-) x 200
@ADC_Init:
            MOV       A, #00000000B
            XB0MOV    ADCM, A       ; Selection ADC Reference voltage = V(R+, R-)
            MOV       A, #0236
            XB0MOV    ADCKS, A      ; Set ADCKS = 236 for ADC working clock = 100K @ 4M X'tal
            MOV       A, #00h
            XB0MOV    DFM, A        ; Set ADC as continuous mode and WRS0 = "0"
@ADC_Enable:                        ; ADC conversion rate =25 Hz
            XB0BSET   FADCENB       ; Enable ADC function
@ADC_Wait:
            XB0BTS1   FDRDY         ; Check ADC output new data or not
            JMP       @ADC_Wait     ; Wait for Bit DRDY = 1
@ADC_Read:                          ; Output ADC conversion word
            XB0BCLR   FDRDY
            XB0MOV    A, ADCDH
            B0MOV     Data_H_Buf, A ; Move ADC conversion High byte to Data Buffer.
            XB0MOV    A, ADCDL
            B0MOV     Data_L_Buf, A ; Move ADC conversion Low byte to Data Buffer.
            …
```

✸ ***Note: Please set ADC relative registers first, than enable ADC function bit.***

Example: ADC Reference Voltage Changes:

```
@ADC_Init:
            MOV         A, #00000000B
            XBMOV       ADCM, A             ; Selection ADC Reference voltage = V(R+, R-)
            MOV         A, #0236
            XB0MOV      ADCKS, A            ; Set ADCKS = 236 for ADC working clock = 100K @ 4M
                                            X'tal
            MOV         A, #00h
            XB0MOV      DFM, A              ; Set ADC as continuous mode and WRS0 = "0" 25 Hz
@ADC_Enable:
            XB0BSET     FADCENB             ; Enable ADC function
@ADC_Wait:
            XB0BTS1     FDRDY               ; Check ADC output new data or not
            JMP         @ADC_Wait           ; Wait for Bit DRDY = 1
@ADC_Read:                                  ; Output ADC conversion word
            XB0BCLR     FDRDY
            XB0MOV      A, ADCDH
            B0MOV       Data_H_Buf, A       ; Move ADC conversion High byte to Data Buffer.
            XB0MOV      A, ADCDL
            B0MOV       Data_L_Buf, A       ; Move ADC conversion Low byte to Data Buffer.
            …
            …
@ADC_RVS1:
            MOV         A, #00001101B       ;Don't disable ADC when change Reference Votlage
            XB0MOV      ADCM, A             ; Selection ADC Reference voltage internal V(1.2V,0.4V)

@@:
            XB0BTS1     FDRDY               ; Check ADC output new data or not
            JMP         @B                  ; Wait for Bit DRDY = 1
                                            ; Output ADC conversion word
            XB0BCLR     FDRDY
            XB0MOV      A, ADCDH
            B0MOV       Data_H_Buf, A       ; Move ADC conversion High byte to Data Buffer.
            XB0MOV      A, ADCDL
            B0MOV       Data_L_Buf, A       ; Move ADC conversion Low byte to Data Buffer.
            …
            …
@ADC_RVS2:
            MOV         A, #00001111B       ;Don't disable ADC when change Reference Votlage
            XBMOV       ADCM, A             ; Selection ADC as Voltage Measure.

@@:
            XB0BTS1     FDRDY               ; Check ADC output new data or not
            JMP         @B                  ; Wait for Bit DRDY = 1
                                            ; Output ADC conversion word
            XB0BCLR     FDRDY
            XB0MOV      A, ADCDH
            B0MOV       Data_H_Buf, A       ; Move ADC conversion High byte to Data Buffer.
            XB0MOV      A, ADCDL
            B0MOV       Data_L_Buf, A       ; Move ADC conversion Low byte to Data Buffer.
            …
```

11.5.6 LBTM : Low Battery Detect Register

SN8P1929 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 P4.1, bit LBTO will output the P4.2 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	P41IO	LBTENB
R/W	-	-	-	-	-	R	R/W	R/W
After Reset	-	-	-	-	-	0	0	0

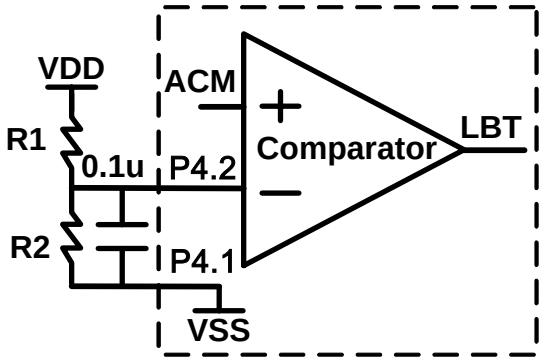
Bit0 **LBTENB:** Low Battery Detect mode control Bit.
0 = Disable Low Battery Detect function,
1 = Enable Low Battery Detect function

Bit1: **P41IO:** Port 4.1 Input/LBT function control bit.
0 = Set P41 as I/O Port,
1 = Set P41 as LBT function

Bit2: **LBTO:** Low Battery Detect Output Bit.
0 = P4.2/LBT voltage Higher than ACM (1.2V)
1 = P4.2/LBT voltage Lower than ACM (1.2V)

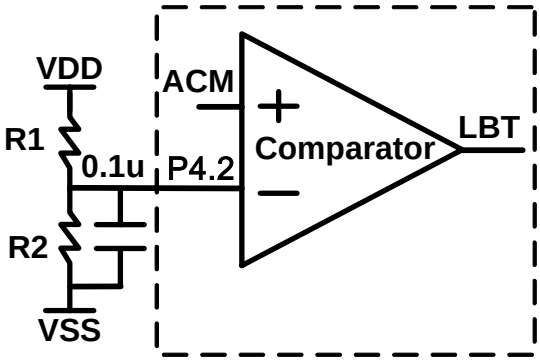
There are two circuit connections for LBT application, One is using P4.0 and P4.1, which can avoid power consumption in sleep mode, the another is using P4.0 only. The second way will leak a small current in power down mode but can use P4.1 for Input application. These two circuits are following:

LBTENB=1, P41IO=1



P5.1 as LBT function, No leakage current in sleep mode

LBTENB=1, P41IO=0



P4.1 as Input port, Leak current in sleep mode.

Low Battery Voltage	R1	R2	LBTO=1
2.4V	1MΩ	1MΩ	VDD<2.4V
3.6V	1.33MΩ	0.66MΩ	VDD<3.6V
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 to make sure the Low Battery signal is stable.***

11.5.7 Analog Setting and Application

The most applications of SN8P1929 were for DC measurement ex. Weight scale, Pressure measure. 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 CR2032 battery, AA/AAA dry battery or external Regulator

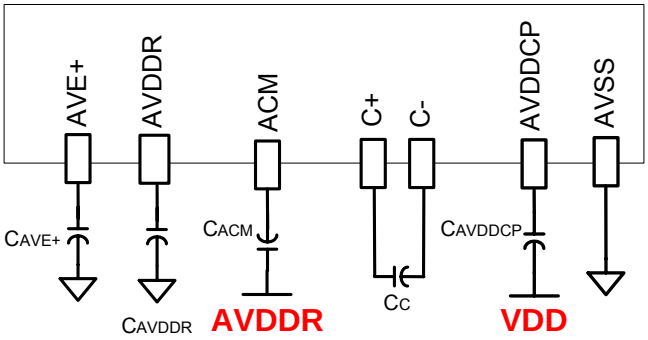
Resistance and Capacitor Table:

Power type	AI+	AI-	X+/X-	R+/R-	ACM	AVDDR	AVE+	AVDDCP	C+/C-	VDD (Pin24)	VDD (Pin31)
	C_{AI+}	C_{AI-}	C_X	C_R	C_{ACM}	C_{AVDDR}	C_{AVE+}	C_{AVDDCP}	C_C	C_{AVDD}	C_{DVDD}
CR2032 (2.4~3V)	0.1uF	0.1uF	0.1uF	0.1uF	1uF	1uF	2.2uF	10uF	1uF	10uF	0.1uF
CR2032 ((4.4~6V))	0.1uF	0.1uF	0.1uF	0.1uF	1uF	1uF	2.2uF	No	No	10uF	0.1uF
AA/AAA Bat.(2.4~3V)	0.1uF	0.1uF	0.1uF	0.1uF	1uF	1uF	4.7uF	10uF	1uF	10uF	0.1uF
AA/AAA Bat.(4.4~6V)	0.1uF	0.1uF	0.1uF	0.1uF	1uF	1uF	4.7uF	No	No	10uF	0.1uF
External 5V Reg.	0.1uF	0.1uF	0.1uF	0.1uF	1uF	1uF	4.7uF	No	No	10uF	0.1uF

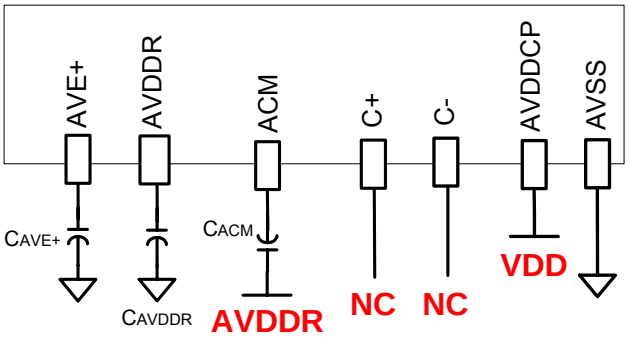
- ***Note 1: When MCU source from CR2032 battery, the AVE+ loading can't over 3mA, for example the Load cell resistance can't over 1K.***
- ***Note 2: In AA/AAA battery application, the AVE+ can loading 10mA current, so that the Load cell can be up to 330 ohm.***
- ***Note 3: If VDD always over 4.2V, Set Charge pump as Auto or Disable mode so that charge pump will disable and current consumption will not time 2 from AVDDR and AVE+. Capacitors of AVDDR and C+/C- can be removed and Connect AVDDCP to VDD.***

- ***Note 1:The positive note of C_{AVDDCP} connect to AVDDCP and Negative note connect to VDD***
- ***Note2: The positive note of C_{ACM} connect to AVDDR and Negative note connect to ACM***

VDD=2.4V~4.2V Analog Capacitor Connection



VDD=4.2V~5.5V Analog Capacitor Connection



Delay Time:

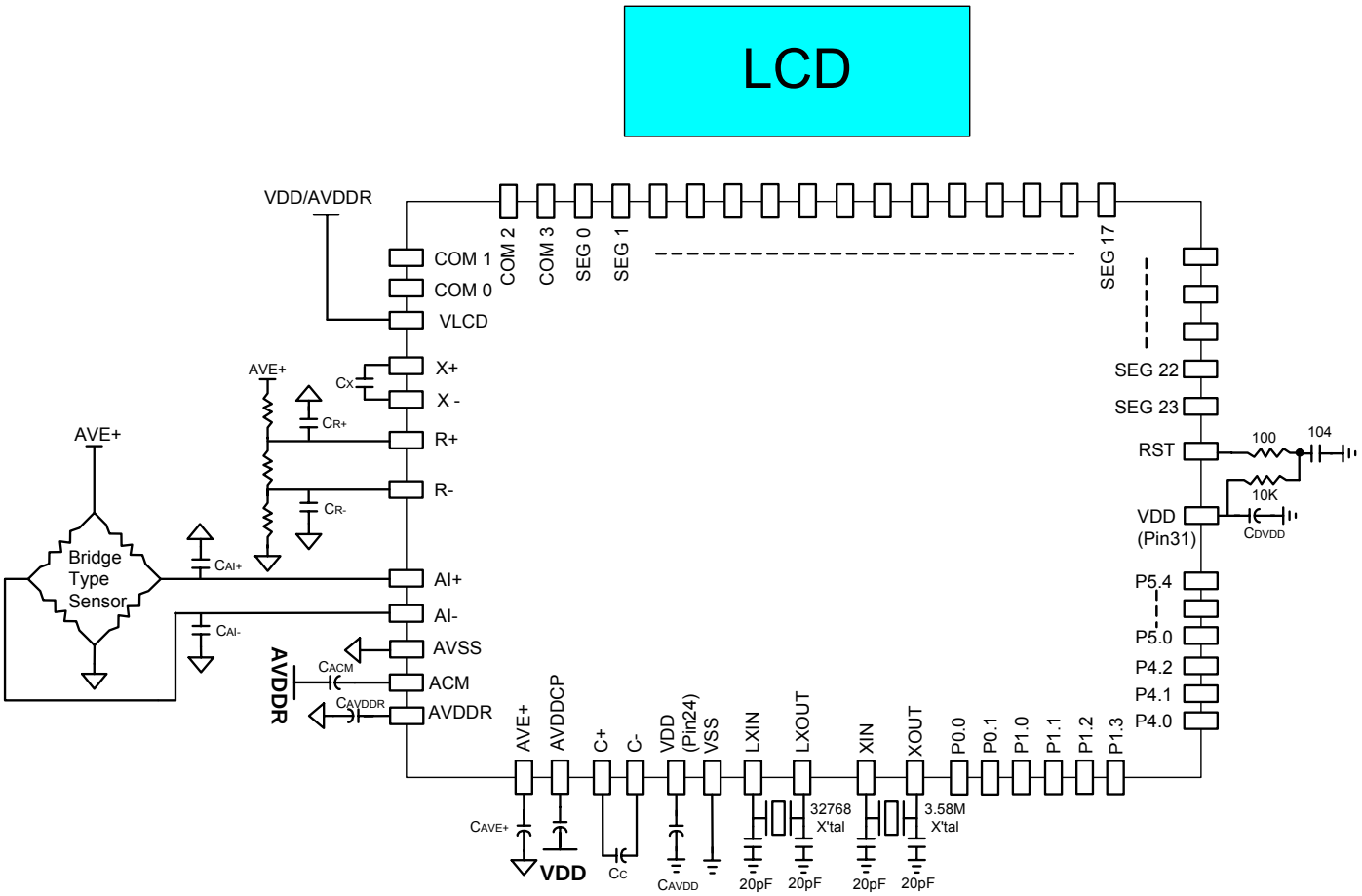
Power Type	Charge Pump Enable Delay		Enable ACM	Enable AVDDR	Enable AVE+
	Step 1 CPCKS=#00001011B	Step 2 CPCKS=#00000100B			
CR2032 (2.4~3V)	200ms	100ms	5ms	50ms	50ms
CR2032 ((4.4~6V))	-	-	5ms	50ms	50ms
AA/AAA Bat.(2.4~3V)	100ms	50ms	5ms	50ms	50ms
AA/AAA Bat.(4.4~6V)	-	-	5ms	50ms	50ms
External 5V Reg.	-	-	5ms	50ms	50ms

- ***Note 1: In CR2032 application, Please set enough delay time or the VDD will drop when Charge pump enable***
- ***Note 2: IF VDD always over 4.2V, set Charge pump as Auto or Disable mode to disable charge pump.***
- ***Note 3: In AA/AAA dry battery application, delay time is shorter than CR2032 application.***

12 APPLICATION CIRCUIT

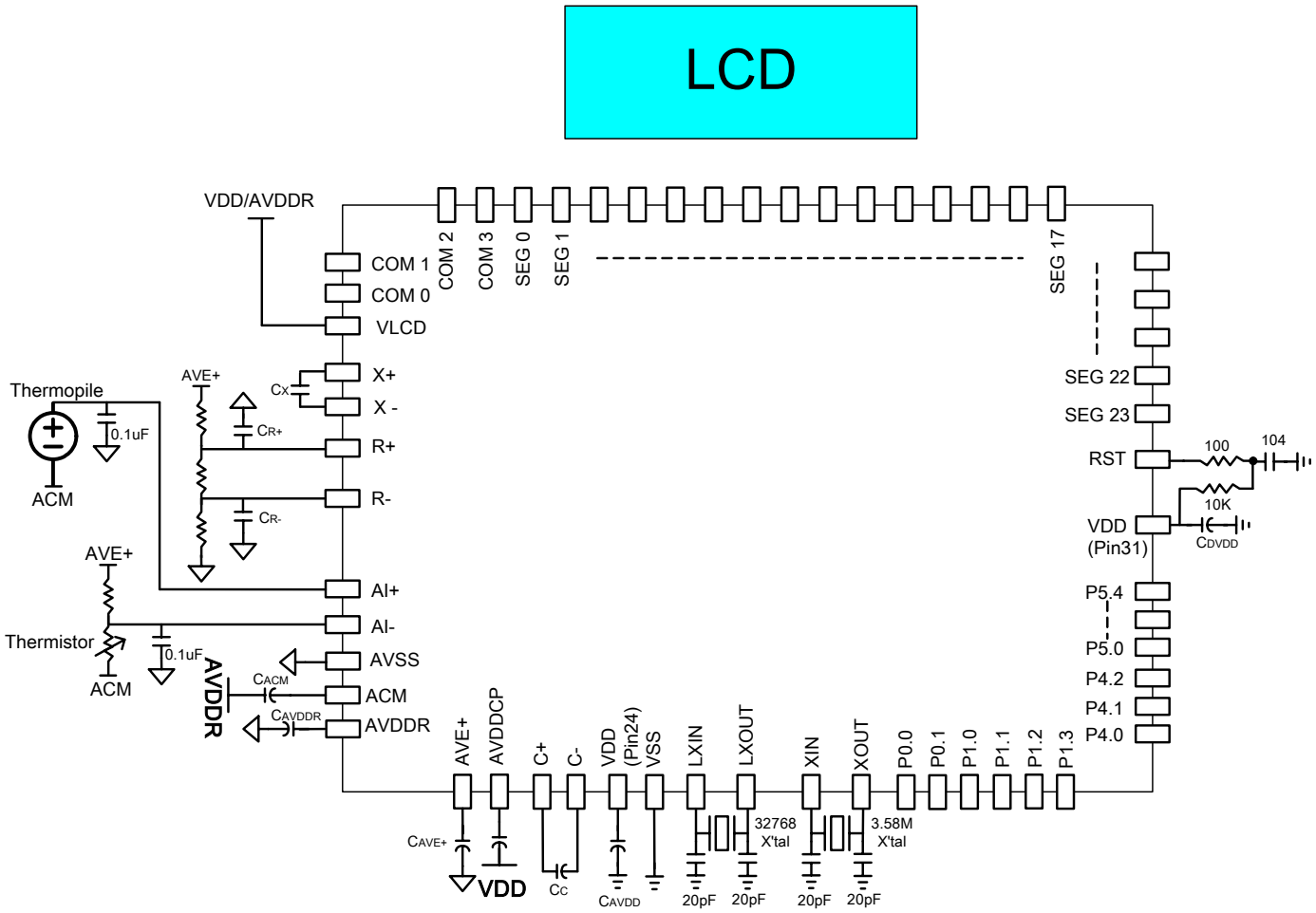
12.1 Scale (Load Cell) Application Circuit

✸ ***Note : Please refer 10.5.7 for capacitor setting.***



12.2 Thermometer Application Circuit

✸ ***Note : Please refer 10.5.7 for capacitor setting.***



13 INSTRUCTION TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bnak 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	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
ARITHMETIC	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
	ADD	A,M	A ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
	ADD	M,A	M ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1
	ADD	A,I	A ← A + I, if occur carry, then C=1, else C=0	√	√	√	1
	SBC	A,M	A ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
	SUB	A,M	A ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
	SUB	M,A	M ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
	DAA		To adjust ACC's data format from HEX to DEC.	√	-	-	1
	MUL	A,M	R, A ← A * M, The LB of product stored in Acc and HB stored in R register. ZF affected by Acc.	-	-	√	2
LOGIC	AND	A,M	A ← A and M	-	-	√	1
	AND	M,A	M ← A and M	-	-	√	1
	AND	A,I	A ← A and I	-	-	√	1
	OR	A,M	A ← A or M	-	-	√	1
	OR	M,A	M ← A or M	-	-	√	1
	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
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1
	CLR	M	M ← 0	-	-	-	1
	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
BRANCH	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1 + S
	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1 + S
	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1 + S
	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1 + S
	BTS0	M.b	If M.b = 0, then skip next instruction	-	-	-	1 + S
	BTS1	M.b	If M.b = 1, then skip next instruction	-	-	-	1 + S
	B0BTS0	M.b	If M(bank 0).b = 0, then skip next instruction	-	-	-	1 + S
	B0BTS1	M.b	If M(bank 0).b = 1, then skip next instruction	-	-	-	1 + S
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
MISC	RET		PC ← Stack	-	-	-	2
	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
	PUSH		To push working registers (080H~087H) into buffers	-	-	-	1
	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".

14 Development Tools

14.1 Development Tool Version

14.1.1 ICE (In circuit emulation)

- **SN8ICE 1K: (S8KD-2)** Full function emulates SN8P1929 series

✷ **SN8ICE1K ICE emulation notice**
- **Operation voltage of ICE: 3.0V~5.0V.**
- **Recommend maximum emulation speed at 5V: 4 MIPS (e.g. 16MHZ crystal and Fcpu = Fhosc/4).**
- **Use SN8P1929 EV-KIT to emulation Analog Function.**

✷ **Note: S8ICE2K doesn't support SN8P1929 serial emulation.**

14.1.2 OTP Writer

MPIII Writer：ON/OFF line operation to support SN8P1929 mass production.

✷ **Note: Writer 3.0 doesn't support SN8P1929 OTP programming.**

14.1.3 IDE (Integrated Development Environment)

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

- For SN8ICE 1K: SN8IDE 1.99Z04 or later
- Easy Writer and MP-Easy Writer doesn't support SN8P1929.
- M2IDE V1.0X doesn't support SN8P1929.

14.2 SN8P1929 EV-KIT

14.2.1 INTRODUCTION

Sonix provides a complete EV-KIT for SN8P1929, which includes an ICE with S8KD chip, SN8P1929 EV Board, Sonix Assembler and Complier. Users are able to do the programming on the computer and to simulate the program code using the software or the ICE itself. On the other hand, when executing the program and monitoring the RAM status, users can user various functions such as Breakpoint, Single step etc. This makes debug much easier for most programmers. Also, the system has build-in 5.0V power supply.

14.2.2 PCB DESCRIPTION

Sonix provides SN8P1929 EV board for all functions emulation shown in FIG.1

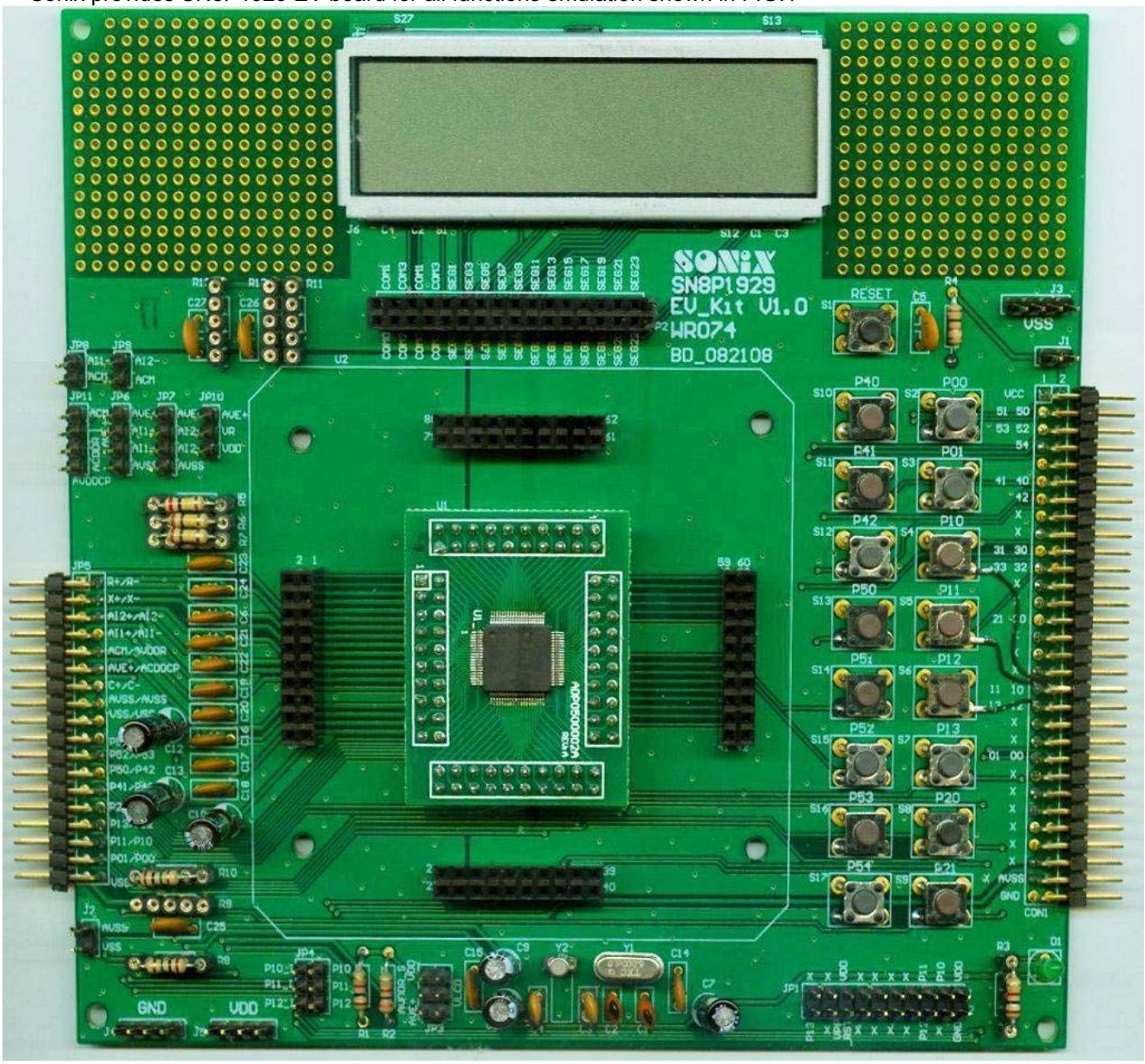


FIG.1 SN8P1929 EV board

14.2.3 EV BOARD SETTING

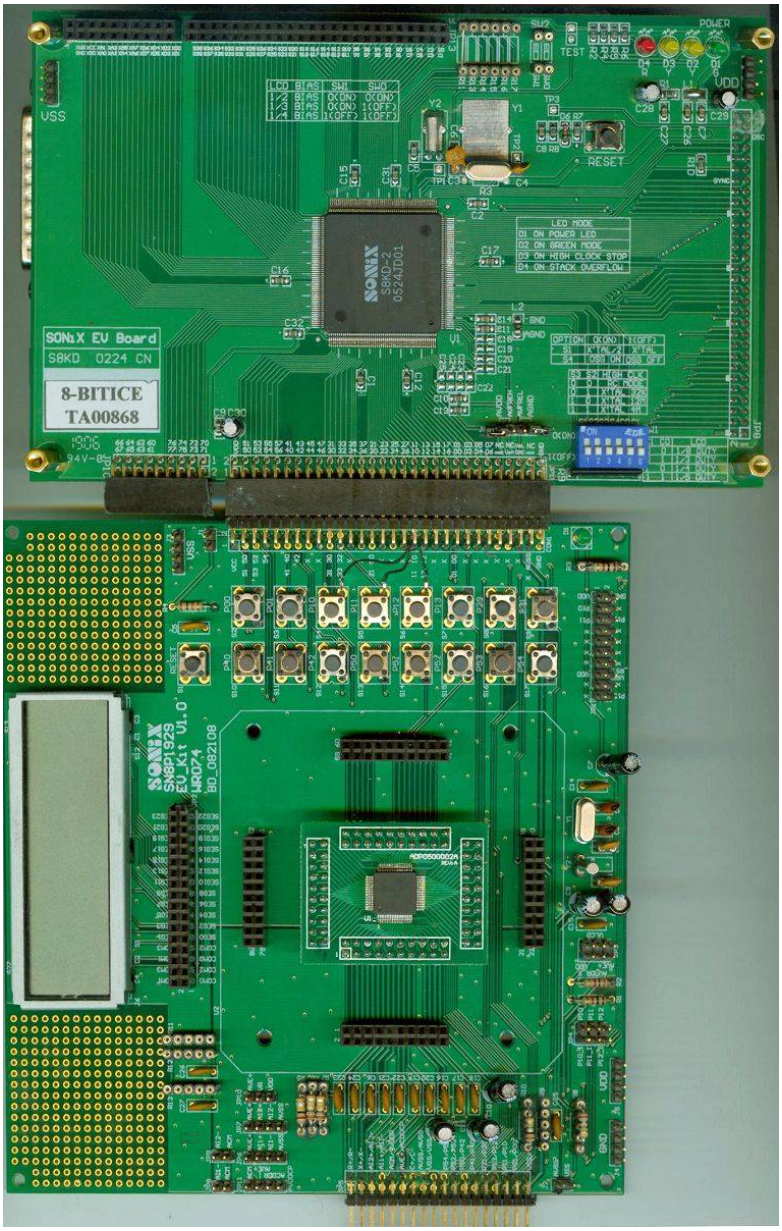
1. CON1 : connecting with the ICE PORT
2. J1 : Switch it “SHORT” position when using the power from ICE, and switch it “OPEN” when using connecting the power from other source.
3. D1 : Power indicator
4. S1 : RESET KEY
5. JP1 : OTP write-in PORT
6. JP2 : LCD PORT
7. JP3 : select different voltage setting for VLCD.
8. JP4 : When separate ICE and Ev board, switch JP4 to “SHORT”，or else the relative IO port won’t work.
9. JP5 :Target board connector.
10. JP6/JP7: Differential Input Channel 1/2.
11. JP10: Selection of external reference voltage V(R+, R-) power.
12. JP11: ACM/ACE+/AVDDR/AVDDCP output port.
13. JP12: This jumper short as “RST/VPP = VDD” for normal operation or short as “RST/VPP = 12.5V” for executing ISP function.
14. J2 : AVSS and VSS “SHORT” pin.
15. R5/R6/R7: Resistors for ADC external reference voltage.
16. R8/R9/R10: LBT function resistance.
17. R23/R24/R25: LCD External resistors for voltage division. User can add resistance between VLCD / V3 / V2 / V1 for more driving current.

	Ev board connects ICE	Ev board connects WRITER	Only EV board
JP1	Disconnect	Write-in SN8P1929	Disconnect
JP2(LCD)	No Function	-	Depends on actual situation
JP3 (VLCD)	Connect to VDD or AVE+	Connect to VDD	Depends on actual situation
JP4	Disconnect all	Disconnect all	Connect all
JP5	Depends on actual situation	Disconnect	Depends on actual situation
JP6/7/8/9/10/11	Depends on actual situation	-	Depends on actual situation
J1	Depends on actual situation	Short	Short
J2	Short	Short	Short
U1	SN8P1929 EV-Link IC only	Blank SN8P1929 IC for programming	SN8P1929 IC with system application code
R8/R9/R10(LBT)	Depends on actual situation	-	Depends on actual situation

14.2.4 SN8P1929 EV BOARD CONNECT WITH ICE

1. EV Board U1 requires IC board embedded with connection program when attach to each other.
2. When power source is supply from the ICE，J1 should be in “SHORT”.
3. When connecting to the develop system, the JP4 should be “OPEN”. Or else the system will not be able to reset successfully.
4. Use the same oscillator for both EV Board and ICE.
5. JP3 is allowed to connect to AVE+、VDD or AVDDR，it is suggested to be connect to VDD.
6. When simulate both EV Board and developer system，J2 needed to be in “SHORT”(Digital GND and Analog GND both connected).
7. R8/R9/R10 are for LBT function. Connect the resistance for this function.

The connection from SN8P1929 EV board to SN8ICE 1K is shown in following.



14.2.5 STAND ALONG EV BOARD

1. Separate EV board and ICE.
2. Attached the IC board with pre-write-in program code to Ev Board U1.
3. When using the stand along Ev Board JP4 should all be in “SHORT”，in order to prevent malfunction for relative IO ports.
4. Confirm J2’s connectivity status，J2 should be in “SHORT”(Digital GND and Analog GND both connected).
5. JP3 is used for monitoring LCD voltage，users are allow to connect to AVE+ or VDD.

14.2.6 SONIX ASSEMBLER

SONIX provides SN8ASM Assembler program, and all sort developer related SN8P1929 software, and as well as complier. The combination of the SONIX ICE and SN8P1929 Emulation Board are used for the purpose of actual circuitry verification and testing, thus to save the costs and time for development.

14.2.7 SYSTEM REQUIREMENT

Operationg system:

SONIX assembler software supports WIN95，win98，WINME，WIN200 and WINXP operation system.

It’s necessary to install device driver under WIN2000 and WINXP.

Files Description:

SN8IDE_xxxx.EXE	Assembler software package，xxx means version.
SN8ASMxxxx.EXE	Main execution program. xxxx mean version.
MACRO1.H	Reference macro one
MACRO2.H	Reference macro two
MACRO3.H	Reference macro three
SN8P1929.INC	Define SN8P1929 all function
1929Ev.H	Constant and Macro definition for SN8P1929 Ev Kit emulation code
1929Ev.ASM	SN8P1929 EV Kit interface subroutine. User must include this file to communication with Ev. Kit
1929_EV_Demo.ASM	SN8P1929 Demo code

14.2.8 NOTE FOR SOFTWARE INSTALLATION

1. Check if the SN8P1929.INC has been included in the designated folder for SONIX Assembler。(Default directory is C:\Sonix\Sn8IDE_xxxx\use_inc2)
2. Check if the main program #include 1929Ev.H and 1929Ev.ASM ， please refer to 1929_TEMPLATE.ASM for detailed instruction。
3. The first line of the main program are listed below：

   ICE_Mode EQU 1

   CHIP SN8P1929
4. When done with the programming, users must set the first line in the main program to real chip mode for OTP programming, example is as follows：

   ICE_Mode EQU 0

   CHIP SN8P1929

Warning:

1. When monitoring the special registers of XB0MOV table over the ICE, we suggest users to copy the XB0MOV to USER RAM area.
2. Do not use special instruction table(ie. XB0MOV) that has not been descried in the previous chapter to avoid unexpected malfunction.

14.2.9 EXAMPLE: PROGRAM CODE STRUCTURE

```
;********************************************************************************
; FILENAME : 1929_Demo.ASM
; AUTHOR : SONiX
; PURPOSE : Demo Code for SN8P1929
;********************************************************************************
;* (c) Copyright 2008, SONiX TECHNOLOGY CO., LTD.
;********************************************************************************
    ICE_Mode EQU 1          ; 1 for ICE , 0 for real chip
    ;ICE_Mode EQU 0
CHIP SN8P1929 ; Select the CHIP
;--------------------------------------------------------------------------------
; Include Files
;--------------------------------------------------------------------------------
.nolist                     ; do not list the macro file
    INCLUDESTD MACRO1.H
    INCLUDESTD MACRO2.H
    INCLUDESTD MACRO3.H
    INCLUDE 1929Ev.h        ; for ICE linking emulation board
```

```
.list                           ; Enable the listing function
;-----------------------------------------------------------------------
; Constants Definition
;-----------------------------------------------------------------------
;       ONE EQU 1
;-----------------------------------------------------------------------
; Variables Definition
;-----------------------------------------------------------------------
.DATA
        org 0h                  ;Bank 0 data section start from RAM address 0x000
Wk00B0  DS 1                    ;Temporary buffer for main loop
Iwk00B0 DS 1                    ;Temporary buffer for ISR
AccBuf  DS 1                    ;Accumulater buffer
PflagBuf DS 1                   ;PFLAG buffer
;-----------------------------------------------------------------------
; Bit Flag Definition
;-----------------------------------------------------------------------
Wk00B0_0        EQU     Wk00B0.0        ;Bit 0 of Wk00B0
Iwk00B0_1       EQU     Iwk00B0.1       ;Bit 1 of Iwk00
;-----------------------------------------------------------------------
; Code section
;-----------------------------------------------------------------------
.CODE
        ORG 0                           ;Code section start
        jmp         Reset               ;Reset vector
                                        ;Address 4 to 7 are reserved
        ORG 8
        Jmp         Isr                 ;Interrupt vector
        ORG 10h
;-----------------------------------------------------------------------
; Program reset section
;-----------------------------------------------------------------------
Reset:
        mov         A,#07Fh             ;Initial stack pointer and
        b0mov       STKP,A              ;disable global interrupt
        b0mov       PFLAG,#00h          ;pflag = x,x,x,x,x,c,dc,z
        b0mov       RBANK,#00h          ;Set initial RAM bank in bank 0
        call        ClrRAM              ;Clear RAM
        call        SysInit             ;System initial
;-----------------------------------------------------------------------
INIT_1929Ev ; for ICE linking emulation board
;-----------------------------------------------------------------------
```

```
    b0bclr    FGIE          ;Enable global interrupt
;-------------------------------------------------------------------------
; Main routine
;-------------------------------------------------------------------------
Main:
    b0bset    FWDRST        ;Clear watchdog timer
    mov       a, #04h ;
    XB0MOV    CPM,a         ;using XB0MOV command for CPM setting
    mov       a, #00000110B ;
    XB0MOV    CPCKS,a       ;using XB0MOV command for CPCKS setting
    Xb0bset   FCPRENB       ;Enable Charge Pump/ Regulator
    Call @Delay_10ms        ;Delay 10ms for CPR stable.
    jmp       Main
;-------------------------------------------------------------------------
INCLUDE 1929Ev.asm      ; SN8P1929 Ev. Kit interface code
;-------------------------------------------------------------------------
ENDP
```

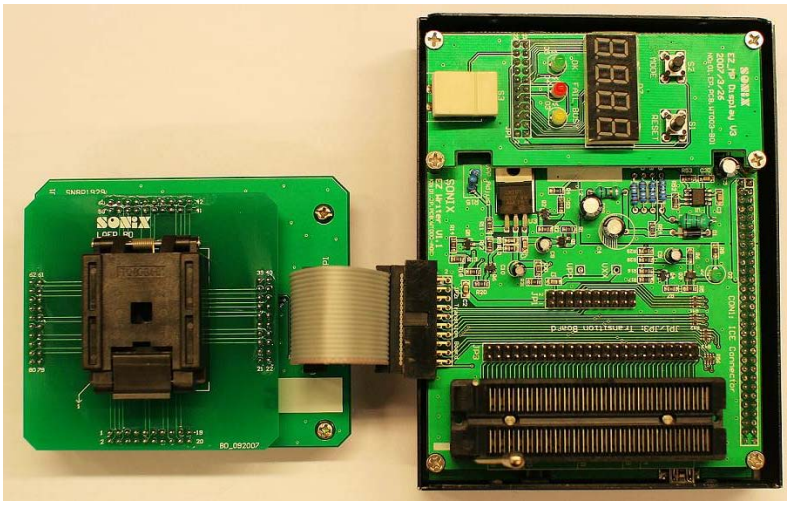
Please be aware of the position of the listed file names marked in red.

14.2.10 OTP WRITE-IN STEP

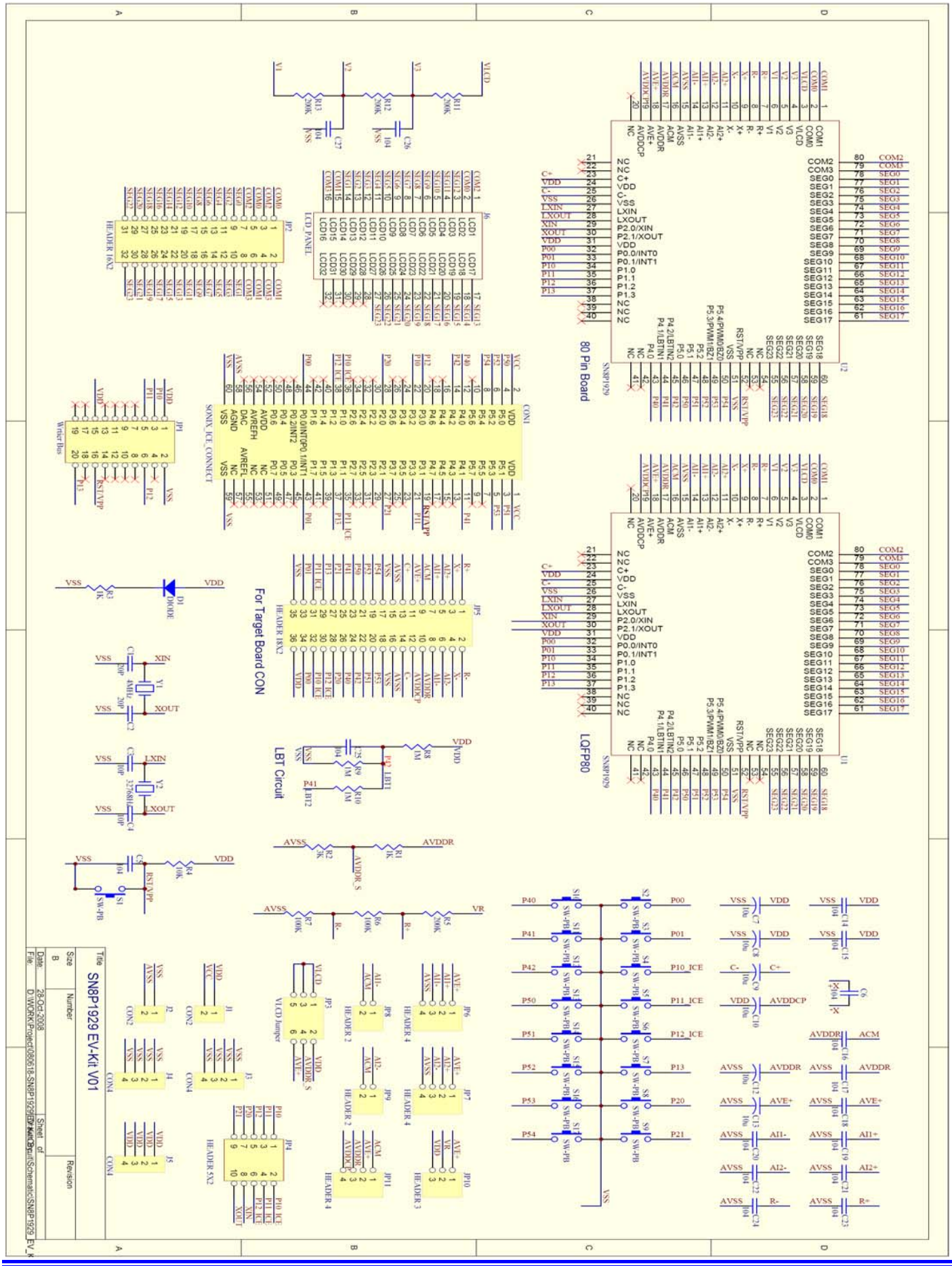
Using SN8P1929 EV-board to program is shown in following step:

1. Separate EV board and ICE.
2. Plug in the empty OTP IC board to U1.
3. Confirm JP3 JUMPER connectivity to VDD.
4. Connect JP1 to MPIII writer.
5. Confirm J2's connecting status, J2 is required to be in "SHORT"。

14.2.11 SN8P1929 PROGRAMMING BOARD CONNECT TO MPIII WRITER



APPENDIX A: EV-KIT BOARD CIRCUIT



15 ELECTRICAL CHARACTERISTIC

15.1 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

15.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to V_{SS}, V_{DD} = 5.0V,F_{OSC} = 4MHz,Fcpu=1MHZ, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION			MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd			2.4	5.0	5.5	V
RAM Data Retention voltage	Vdr				-	1.5	-	V
V_{DD} rise rate	V_{POR}	VDD rise rate to ensure power-on reset			0.05	-	-	V/ms
Input Low Voltage	ViL1	All input pins			Vss	-	0.3Vdd	V
	ViL2	Reset pin			Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input pins			0.7Vdd	-	Vdd	V
	ViH2	Reset pin			0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd			-	-	5	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V			100	200	300	KΩ
		Vin = Vss , Vdd = 5V			50	100	180	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd			-	-	2	uA
I/O Port source current	IoH	Vop = Vdd - 0.5V			8	12	-	mA
sink current	IoL	Vop = Vss + 0.5V			8	15	-	
INTn trigger pulse width	Tint0	INT0 ~ INT1 interrupt request pulse width			2/fcpu	-	-	cycle
Supply Current	Idd1	Normal Mode (Low Power Disable, Analog Parts OFF)		Vdd= 5V 4MHz / IHRC	-	2.2	4	mA
				Vdd= 3V 4MHz / IHRC	-	1	2	mA
	Idd2	Normal Mode (Low Power Enable, Analog Parts OFF)		Vdd= 5V 4MHz / IHRC	-	1.8	4	mA
				Vdd= 3V 4MHz / IHRC	-	0.8	2	mA
	Idd3	Normal Mode (Low Power Disable, Analog Parts ON)		Vdd= 5V 4MHz / IHRC	-	3	5	mA
				Vdd= 3V 4MHz / IHRC	-	2.2	4.5	mA
	Idd4	Normal Mode (Low Power Enable, Analog Parts ON)		Vdd= 5V 4MHz / IHRC	-	2.5	5	mA
				Vdd= 3V 4MHz / IHRC	-	2.2	4	mA
	Idd5	Slow mode (Stop High Clock, LCD OFF, CPR OFF)		Vdd= 5V Ext.32768Hz	-	20	30	uA
				Vdd= 3V Ext.32768Hz	-	8	20	uA
	Idd6	Slow mode (Stop High Clock, LCD ON 200K, CPR OFF)		Vdd= 5V Ext.32768Hz	-	30	50	uA
				Vdd= 3V Ext.32768Hz	-	15	30	uA
	Idd7	Slow mode (Stop High Clock, LCD ON 200K, CPR ON)		Vdd= 5V Ext.32768Hz	-	300	600	uA
				Vdd= 3V Ext.32768Hz	-	250	500	uA
	Idd8	Green mode *Stop High Clock *LCD OFF *CPR OFF	By_CPUM	Vdd= 5V Ext.32768Hz	-	10	20	uA
				Vdd= 3V Ext.32768Hz	-	4	4	uA
	Idd9		Internal_RC always on	Vdd= 5V Ext.32768Hz	-	15	30	uA
				Vdd= 3V Ext.32768Hz	-	6	12	uA
	Idd10	Green mode *Stop High Clock *LCD ON 200K	By_CPUM	Vdd= 5V Ext.32768Hz	-	21	40	uA
				Vdd= 3V Ext.32768Hz	-	10	20	uA

	Idd11	*CPR OFF	Internal_RC always on	Vdd= 5V Ext.32768Hz	-	25	50	uA
				Vdd= 3V Ext.32768Hz	-	12	25	uA
	Idd12	Green mode *Stop High Clock *LCD ON 200K *CPR ON	By_CPUM	Vdd= 5V Ext.32768Hz	-	300	600	uA
				Vdd= 3V Ext.32768Hz	-	250	500	uA
			Internal_RC always on	Vdd= 5V Ext.32768Hz	-	300	300	uA
				Vdd= 3V Ext.32768Hz	-	250	500	uA
	Idd13	Sleep Mode		Vdd= 5V	-	1	5	uA
				Vdd= 3V	-	0.7	5	uA
LVD Detect Level	V_{LVD}	Internal POR detect level		25℃	1.9	2.0	2.1	V
				40℃~85℃	1.8	2.0	2.2	V
Internal High Clock Freq.	F_{IHRC}	Internal High RC Oscillator Frequency			14	16	18	MHz

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

- **Note: Analog Parts including Charge Pump Regulator (CPR), PGIA and ADC.**

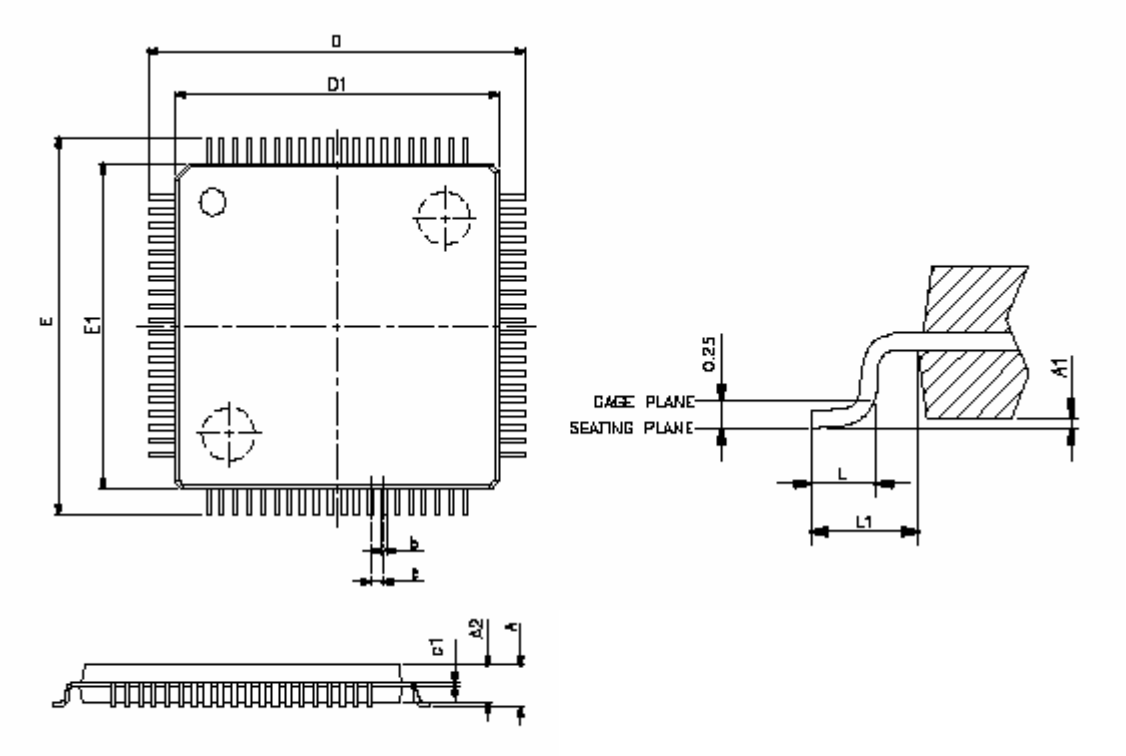
(All of voltages refer to Vdd=3.8V F_{OSC} = 4MHz, ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
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	uA
Conversion rate	F_{SMP}	ADCKS: 200KHz			25	sps
Reference Voltage Input Voltage	Vref	R+, R- Input Range (External Ref.)	0.4		2.0	V
		R+, R- Input Range (Internal Ref.)	0.2		2.0	V
Differential non-linearity	DNL	ADC range ±28125		±0.5	±0.5	LSB
Integral non-linearity	INL	ADC range ± 28125		±1	±4	LSB
No missing code	NMC	ADC range ± 28125	16			bit
Noise free code	NFC	ADC range ± 28125		14	16	bit
Effective number of bits	ENOB	ADC range ±28125		14	16	bit
ADC Input range	V_{AIN}		0.4		2.0	V
Temperature Sensor inaccuracy	E_{TS}	Inaccuracy range vs. real Temp.		±8		℃
PGIA						
Current consumption	I_{DD_PGIA}	Run mode @ 3.8V		300	500	uA
Power down current	I_{PDN}	Stop mode @ 3.8V			0.1	uA
Input offset voltage	Vos			25	50	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}		1.18	1.23	1.28	V
Reference Voltage Temperature Coefficient	T_{ACM}			50*		PPM/℃
Operating current	I_{BG}	Run mode @ 3.8V		50	100	uA
Charge pump regulator						
Supply voltage	V_{CPS}	Normal mode	2.4		5.5	V
Regulator output voltage AVDDR	V_{AVDDR}		3.65	3.8	4.0	V
Regulator output voltage AVE+	V_{AVE+}	AVE+ set as 3.0V	2.9	3.0	3.3	V
Analog common voltage	V_{ACM}		1.18	1.23	1.28	V
Regulator output current capacity	I_{VA+}		10			mA
Quiescent current	I_{QI}			700	1400	uA
V_{ACM} driving capacity	I_{SRC}		10			uA
V_{ACM} sinking capacity	I_{SNK}		1			mA
In-System-Program ROM Function						
ISP operating temperature	T_{ISP}			25	30	℃

✷ ***Note : When Charge Pump enable, current consumption will be time 2 of ADC, PGIA, CPR and Loading from AVE+, AVDDR.***

16 PACKAGE INFORMATION

16.1 LQFP 80 PIN



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

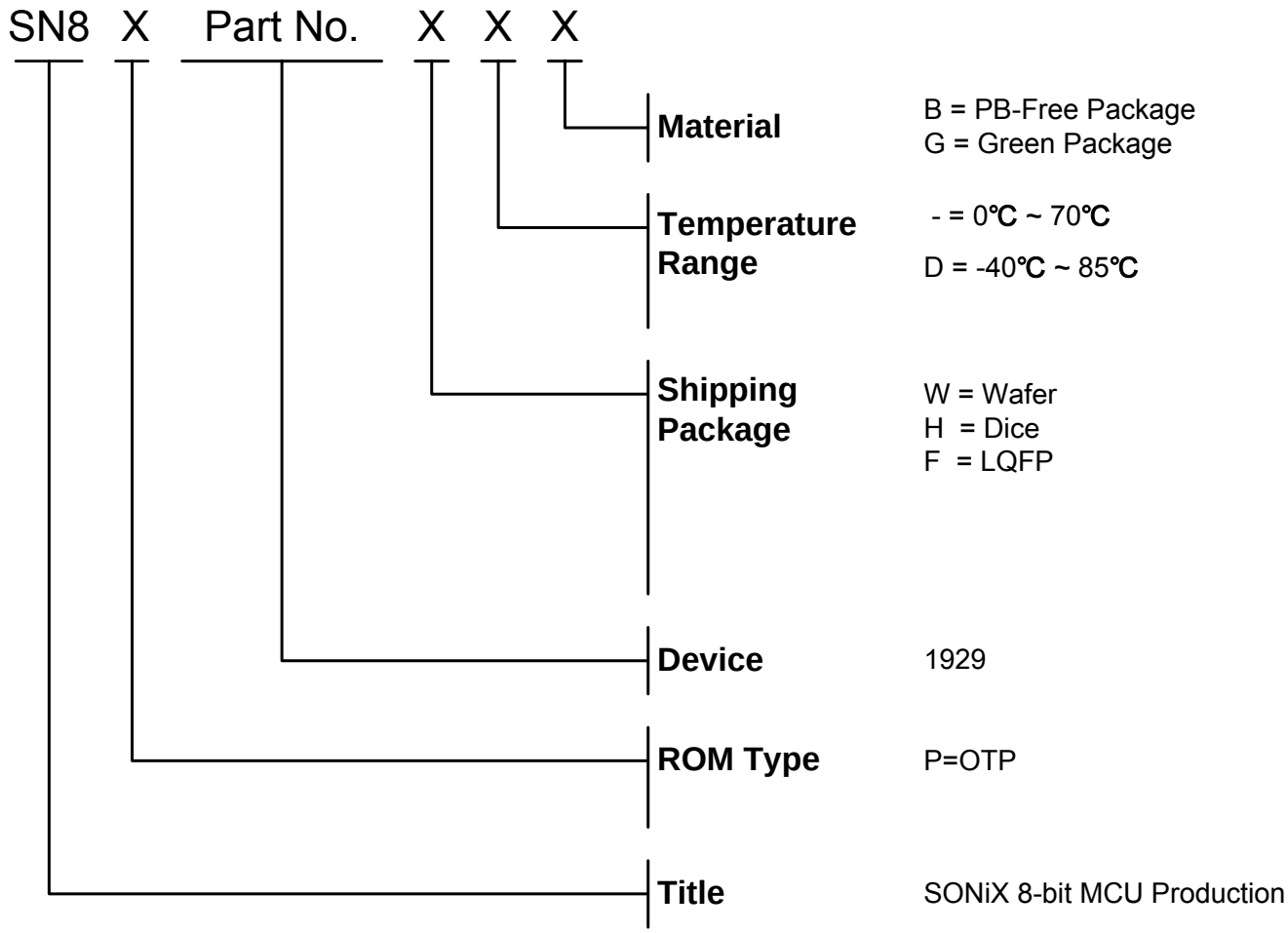
SYMBOLS	MIN.	MAX.
A	--	1.6
A1	0.05	0.15
A2	1.35	1.45
c1	0.09	0.16
D	12 BSC	
D1	10 BSC	
E	12 BSC	
E1	10 BSC	
e	0.4 BSC	
b	0.17	0.27
L	0.45	0.75
L1	1 REF	

17 Marking Definition

17.1 INTRODUCTION

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

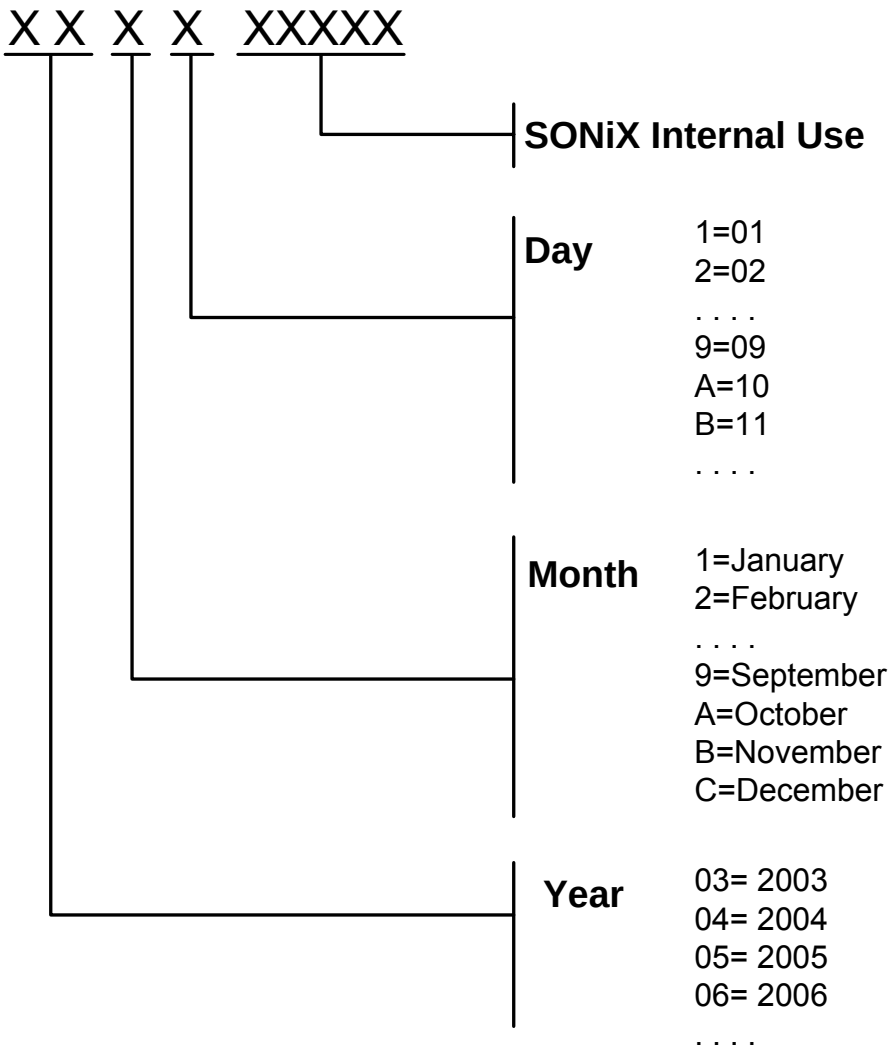
17.2 MARKING IDENTIFICATION SYSTEM



17.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P1929FB	OTP	1929	LQFP	0℃~70℃	PB-Free Package
SN8P1929FG	OTP	1929	LQFP	0℃~70℃	Green Package

17.4 DATECODE SYSTEM



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

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

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

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