SN8P1917

USER'S MANUAL

Specification Version 1.6

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 0.1	Aug. 2005	First issue.
VER 0.2	Sep. 2005	1. Add LCD Bias register 2. Chapter 10 sample modified 3. Change FDS setting 4. Change Programming pin of P1.1/P1.2 5. 10.4.3 Note Error. 6. ADC Clock Rater change 7. ADD BROWN-OUT circuit.
VER. 0.3	Nov. 2005	1. For New Version Chip (B) Application 2. AMPCKS Selection Update 3. ADCM Register update for RVS and IRVS
VER. 0.4	Nov. 2005	1. ADD ADC current. 2. Modify Topr value.
VER.0.5	Dec. 2005	1. Modify 8P1917_PreV04 Brown-Out Reset description. 2. Modify ELECTRICAL CHARACTERISTIC. 3. Remove Calibration Function.
VER.0.6	Jan. 2006	1. Change X+/X- capacitor to 0.01uF 2. Internal Reference voltage depend on AVE+ 3. Chopper Frequency suggestion value is 2K 4. Remove P1W, P0UR 5. Add V1/V2 description. 6. Change ADENB to ADCENB 7. Change AVE+, AVDDR characteristic data.
VER.1.0	Apr. 2006	1. Add Instruction Set table 2. Remove IREFENB bit 3. RAM data-retention spec. is MIN. 4. Remove Migration table of PGIA Temp. resistance and ADC work in Slow mode 5. AVDDCP capacitor is 10 uF of Pin description 6. Remove P4CON register 7. Correct IRVS and ADCM setting of reference table 8. Add VSS pin (Pin40) into program table
VER.1.1	May. 2006	1. Temp sensor ch. should be 101 not 100. (on graphic) 2. Add VLCD pin connect to VDD when OTP programming 3. AVDDCP /ACM capacitor connection on 5V and 3V 4. Update application circuit.
VER.1.2	Sep. 2006	1. Modified CPCKS/AMPCKS/ADCKS as write mode register 2. Limit the ADC Linear range as ± 28125 in ADC chapter and elec. Char. 3. Add TS-Temperature sensor elec. Spec. 4. Change CPCKS as 15.6K 5. Modified ACM elec. Spec. 6. Modified PGIA offset of elec. Spec.
VER.1.3	Dec. 2006	1. Modified P00G[1:0] function description.
VER.1.4	Jan. 2007	1. Add Marking Definition. 2. Modify ELECTRICAL CHARACTERISTIC.
VER.1.5	Apr. 2007	1. Add LCD circuit.
VER 1.6	Sep. 2007	1. Remove DIP48 package.

Table of Content

1 PRODUCT OVERVIEW

1.1 SELECTION TABLE

CHIP	ROM	RAM	Stack	LCD	Timer			I/O	ADC	PWM Buzzer	SIO	Wakeup Pin no.	Package
					T0	TC0	TC1						
SN8P1907	2K*16	128*8	8	4*12	V	-	-	11	16-bit	-	-	5	SSOP48
SN8P1917	2K*16	128*8	8	4*12	V	-	-	13	16-bit	-	-	5	SSOP48

Table 1-1 Selection table of SN8P1917

1.2 MIGRATION TABLE

Migration SN8P1917 Series to SN8P1907 Series

Item	SN8P1917	SN8P1907
PGIA Gain setting	**1x, 12.5x, 50x, 100x, 200x**	**1x, 16x, 32x, 64x, 128x**
PGIA Temperature Drift	**Good**	**No Good**
AVE+ Voltage	**3.0V or 1.5V**	**3.0V only**
ADC Reference Voltage V(R+, R-)	**0.8V or 0.4V**	**0.8V only**
Battery Detect Method	**By Comparator or By ADC**	**By ADC only**
Temperature Sensor	**Build In**	**External**
ACM (1.2V) Voltage	**Not Change with Sink current**	**Change with Sink current**
Charge pump clock frequency (CPCKS)	**4-Bit Selection**	**2-Bit Selection**
Chopper clock frequency (AMPCKS)	**3-Bit Selection**	**2-Bit Selection**
Charge pump Regulator working in slow mode	**Yes**	**No**
Operating Current Consumption	**Less**	**More**
Slow mode Current Consumption	**Less**	**More**
LCD Bias Voltage	**1/3 or 1/2 Bias**	**1/2 Bias only**
Internal 16M RC Oscillator	**Yes**	**No**
P2 [1:0] I/O	**Available when Fosc=IHRC**	**No**
OTP Programming Method	**Serial Method**	**Parallel Method**

Table 1-2 SN8P1917 Migration Table

1.3 FEATURES

- ***Memory configuration***
  OTP ROM size: 2K * 16 bits
  RAM size: 128 * 8 bits (bank 0)
  8-levels stack buffer
  LCD RAM size: 4*12 bits
- ***I/O pin configuration***
  Input only: P0, P4
  Bi-directional: P1, P2
  Output only: P5
  Wakeup: P0, P1
  Pull-up resisters: P0, P1, P2, P4
  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***
- ***Build In PGIA Temperature Compensation Resistance***
- **16-bit Delta-Sigma ADC with 14-bit noise free**
  Two ADC channel configuration:
  One fully differential channel
  Two single channels
- ***Two interrupt sources***
  One internal interrupts: T0
  One external interrupts: INT0
- ***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/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 or 0.4V.***
- ***Build In Temperature Sensor.***
- ***LCD driver:***
  1/3 or 1/2 bias voltage.
  4 common * 12 segment
- ***Dual clock system offers four operating modes***
  External high clock: RC type up to 10 MHz
  External high clock: Crystal type up to 8 MHz
  Normal mode: Both high and low clock active.
  Slow mode: Low clock only.
  Sleep mode: Both high and low clock stop.
- ***Package***
  SSOP48

1.4 SYSTEM BLOCK DIAGRAM

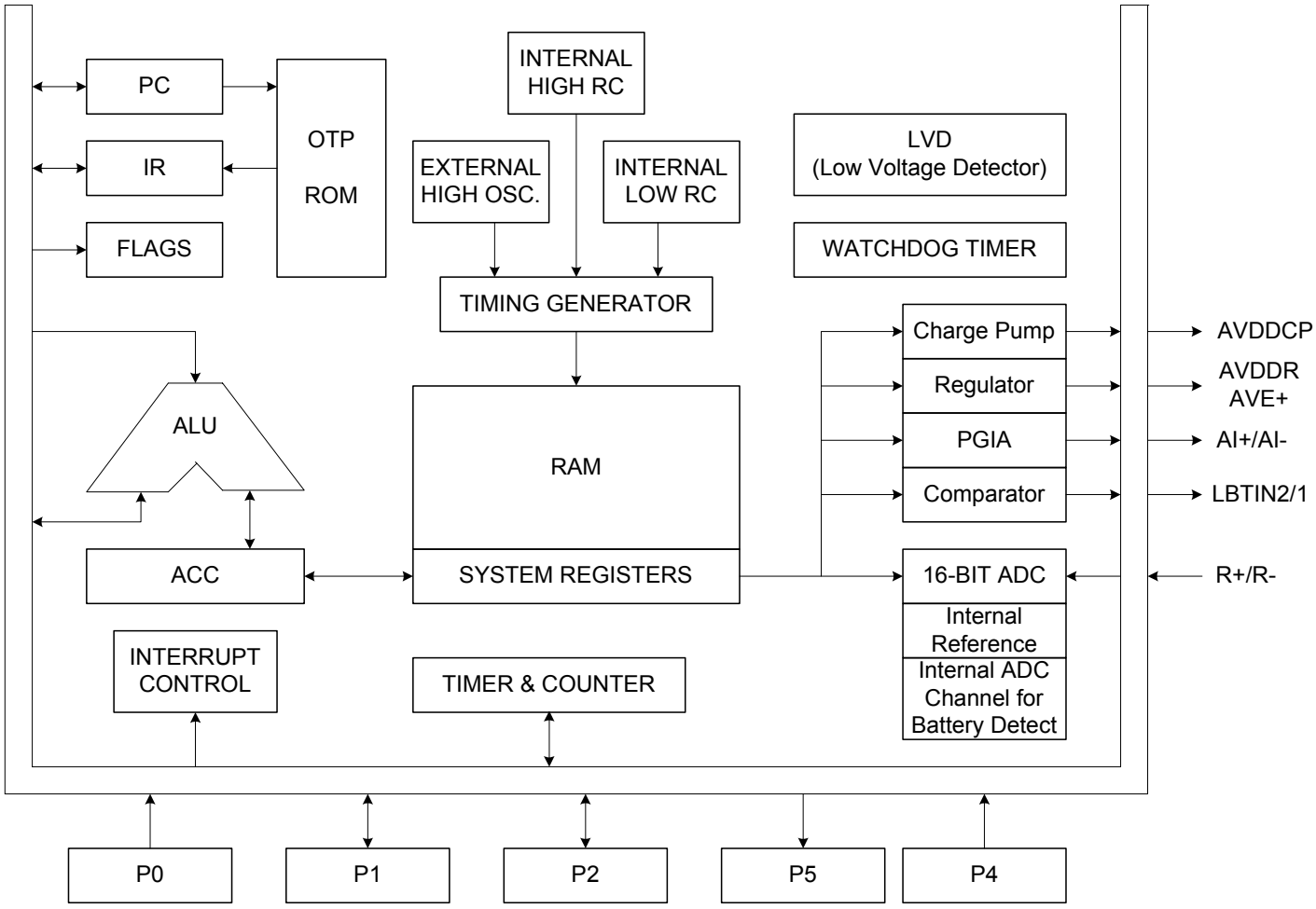


Figure 1-1 Simplified system block diagram

1.5 PIN ASSIGNMENT

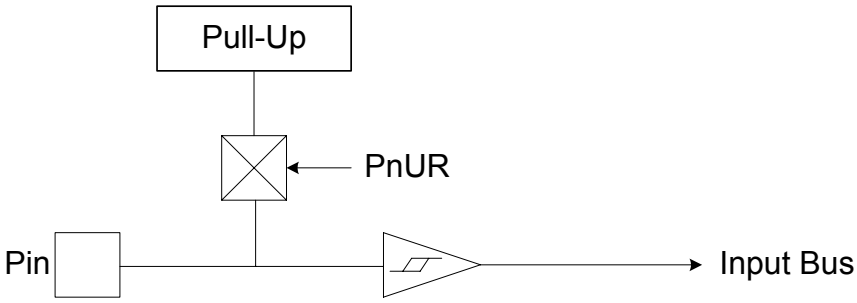
SEG4	1	48	SEG5
SEG3	2	47	SEG6
SEG2	3	46	SEG7
SEG1	4	45	SEG8
SEG0	5	44	SEG9
COM3	6	43	SEG10
COM2	7	42	SEG11
COM1	8	41	VPP/RST
COM0	9	40	VSS
VLCD	10	39	P5.2
R+	11	38	P5.1
R-	12	37	P5.0
X+	13	36	P4.2/LBTIN2
X-	14	35	P4.1/LBTIN1
AI+	15	34	P4.0
AI-	16	33	P1.3
AVSS	17	32	P1.2
ACM	18	31	P1.1
ADDR	19	30	P1.0
AVE+	20	29	P0.0/INT0
AVDDCP	21	28	VDD
C+	22	27	XOUT/P2.1
VDD	23	26	XIN/P2.0
C-	24	25	VSS

1.6 PIN DESCRIPTIONS

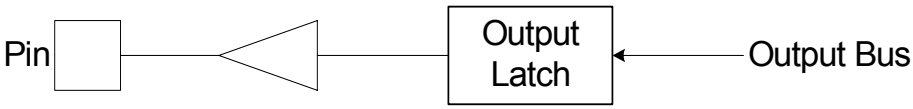
PIN NAME	TYPE	DESCRIPTION
VDD, VSS, AVSS	P	Power supply input pins for digital / analog circuit.
VLCD	P	LCD Power supply input
AVDDR	P	Regulator power output pin, Voltage=3.8V.
AVE+	P	Regulator output =3.0V or 1.5V for Sensor. Maximum output current=10 mA
ACM	P	Band Gap Voltage output =1.2V
AVDDCP	P	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
AI+	AI	Positive analog input channel
AI-	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. RC mode from XIN.
P0.0 / INT0	I	Port 0.0 and shared with INT0 trigger pin (Schmitt trigger) / Built-in pull-up resisters.
P1 [3:0]	I/O	Port 1.0~Port 1.3 bi-direction pins / wakeup pins/ Built-in pull-up resisters.
P2 [1:0]	I/O	Port 2.0~Port 2.1 bi-direction pins / Built-in pull-up resisters. Shared with XIN/XOUT
P4 [2:0]	I	Port 4.0~Port 4.2 Input pins/ Built-in pull-up resisters
P5 [2:0]	O	Port 5.0~Port 5.2 output pins
LBTIN1/2	I I	Low BatTery detect Input pins shared with P4.1, P4.2
COM [3:0]	O	COM0~COM3 LCD driver common port
SEG0 ~ SEG11	O	LCD driver segment pins.

1.7 PIN CIRCUIT DIAGRAMS

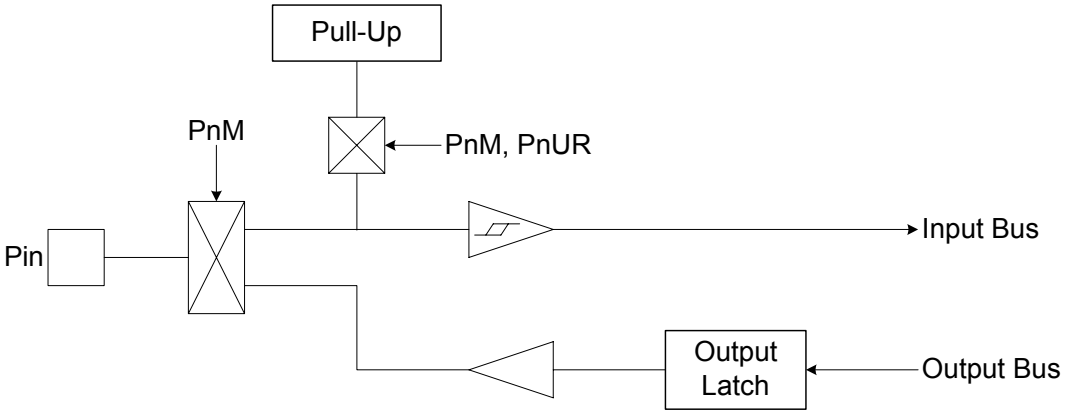
Port 0, Port 4structure:



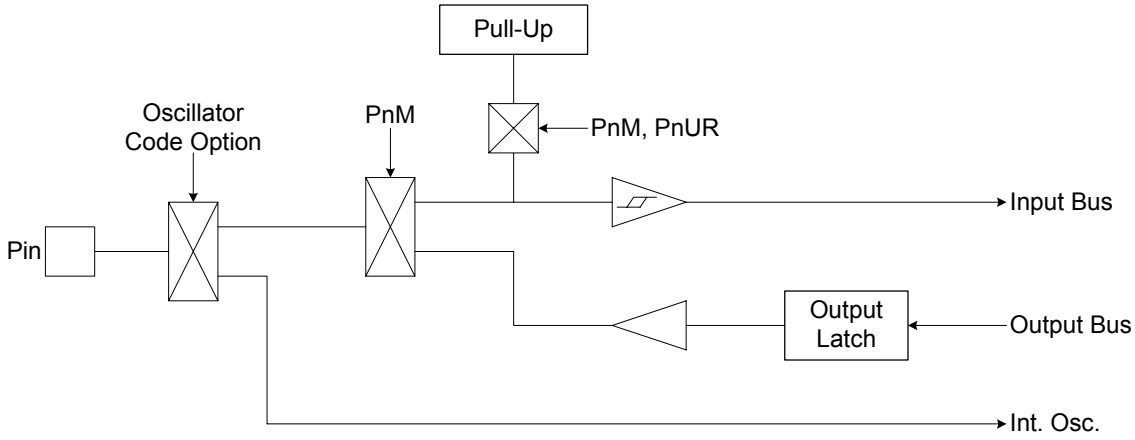
Port 5 structure:



Port1 structure:



Port2 structure:



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ ***2K 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		
.		
.		
7FEH		End of user program
7FFH	***Reserved***	

2.1.1.1 RESET VECTOR (0000H)

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

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

After power on reset, external reset or watchdog timer overflow reset, then the chip will restart the program from address 0000h and all system registers will be set as default values. The following example shows the way to define the reset vector in the program memory.

➢ **Example: Defining Reset Vector**

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

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

                ENDP                ; End of program
```

2.1.1.2 INTERRUPT VECTOR (0008H)

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

✸ ***Note: Users have to save and load ACC and PFLAG register 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.
            PFLAGBUF   DS   1          ; Define PFLAGBUF for store PFLAG data.

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

            ORG        8               ; Interrupt vector.
            B0XCH      A, ACCBUF       ; Save ACC in a buffer.
            B0MOV      A, PFLAG
            B0MOV      PFLAGBUF, A     ; Save PFLAG register in a buffer.
            …
            …
            B0MOV      A, PFLAGBUF
            B0MOV      PFLAG, A        ; Restore PFLAG register from buffer.
            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.
            PFLAGBUF   DS   1        ; Define PFLAGBUF for store PFLAG 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.
            B0MOV      A, PFLAG
            B0MOV      PFLAGBUF, A   ; Save PFLAG register in a buffer.
            …
            …
            B0MOV      A, PFLAGBUF
            B0MOV      PFLAG, A      ; Restore PFLAG register from buffer.
            B0XCH      A, ACCBUF     ; Restore ACC from buffer.
            RETI                     ; End of interrupt service routine.
            …

            ENDP                     ; End of program.
```

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

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

2.1.1.3 LOOK-UP TABLE DESCRIPTION

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

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

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

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

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

➢ **Example: INC_YZ macro.**

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

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

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

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

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

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

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

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

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

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

2.1.1.4 JUMP TABLE DESCRIPTION

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

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

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

➢ **Example: Jump table.**

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

        B0ADD   PCL, A          ; PCL = PCL + ACC, the PCH can’t be changed.
        JMP     A0POINT         ; ACC = 0, jump to A0POINT
        JMP     A1POINT         ; ACC = 1, jump to A1POINT
        JMP     A2POINT         ; ACC = 2, jump to A2POINT
        JMP     A3POINT         ; ACC = 3, jump to A3POINT
```

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

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

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

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

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

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

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

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

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

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

➢ **Example: "@JMP_A" operation.**

; Before compiling program.

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

; After compiling program.

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

2.1.1.5 CHECKSUM CALCULATION

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

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

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

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

2.1.2 CODE OPTION TABLE

Code Option	Content	Function Description
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
Low Power	Enable	Enable Low Power function to save Operating current
	Disable	Disable Low Power function

✷ **Note:**

1. In high noisy environment, set Watch_Dog as "Enable" and INT_16K_RC as "Always_ON" is strongly recommended.

2. 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" code option.

5. Enable "Low Power" option will reduce operating current except in slow mode.

2.1.3 DATA MEMORY (RAM)

☞ ***128 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 15	F00h	LCD RAM area	; Bank 15 = To store LCD display data
		.	; (12 bytes).
	F0Bh	End of LCD Ram	;

2.1.4 SYSTEM REGISTER

2.1.4.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	-	-	R	Z	Y	-	PFLAG	RBANK	-	LCDM1	-	-	-	-	-	-
9	AMPM	AMPCHS	AMPCKS	ADCM	ADCKS	CPM	CPCKS	DFM	ADCDL	ADCDH	LBTM	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	VERFH
B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	PEDGE
C	-	P1M	P2M	-	-	-	-	-	INTRQ	INTEN	OSCM	-	-	-	PCL	PCH
D	P0	P1	P2	-	P4	P5	-	-	T0M	T0C	-	-	-	-	-	STKP
E	-	P1UR	P2UR	-	-	-	-	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.1.4.2 SYSTEM REGISTER DESCRIPTION

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
LBTM= Low Battery Detect Register

R = Working register and ROM look-up data buffer
AMPCHS = PGIA channel selection
ADCM = ADC's mode register
CPM = Charge pump mode
DFM = Decimation filter mode
ADCDH = ADC high-byte data buffer
P_NUR = Port N pull-up register
INTRQ = Interrupt request register
OSCM = Oscillator mode register
PCH, PCL = Program counter
STK0~STK7 = Stack 0 ~ stack 7 buffer
@YZ = RAM YZ indirect addressing index pointer
STKP = Stack pointer buffer

2.1.4.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Name
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
086H	-	-	-	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	RBNKS3	RBNKS2	RBNKS1	RBNKS0	R/W	RBANK
089H	-	-	LCDBNK	-	LCDENB	LCDBIAS	LCDRATE	LCDCLK	R/W	LCDM1
090H	-	BGRENB	FDS1	FDS0	GS2	GS1	GS0	AMPENB	R/W	AMPM
091H	-	-	-	-	-	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	AVESEL	AVENB	CPSTS	CPAUTO	CPON	CPRENB	R/W	CPM
096H	-	-	-	-	CPCKS3	CPCKS2	CPCKS1	CPCKS0	W	CPCKS
097H			-	-	-	WRS0		DRDY	R/W	DFM
098H	ADCB7	ADCB6	ADCB5	ADCB4	ADCB3	ADCB2	ADCB1	ADCB0	R	ADCDL
099H	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB9	ADCB8	R	ADCDH
09AH	-	-	-	-	-	LBTO	P41IO	LBTENB	R/W	LBTM
0BFH	PEDGEN	-	-	P00G1	P00G0	-	-	-	R/W	PEDGE
0C1H	-	-	-	-	P13M	P12M	P11M	P10M	R/W	P1M
0C2H	-	-	-	-	-	-	P21M	P20M	R/W	P2M
0C8H	-	-	-	T0IRQ	-	-	-	P00IRQ	R/W	INTRQ
0C9H	-	-	-	T0IEN	-	-	-	P00IEN	R/W	INTEN
0CAH	WTCKS	WDRST	WDARTE	-	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	-	-	PC10	PC9	PC8	R/W	PCH
0D0H	-	-	-	-	-	-	-	P00	R	P0
0D1H	-	-	-	-	P13	P12	P11	P10	R/W	P1
0D2H	-	-	-	-	-	-	P21	P20	R/W	P2
0D4H	-	-	-	-	-	P42	P41	P40	R	P4
0D5H	-	-	-	-	-	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0RATE2	T0RATE1	T0RATE0	-	-	-	-	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DFH	GIE	-	-	-	STKPB3	STKPB2	STKPB1	STKPB0	R/W	STKP
0E1H	-	-	-	-	P13R	P12R	P11R	P10R	W	P1UR
0E2H	-	-	-	-	-	-	P21R	P20R	W	P2UR
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	-	-	-			S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	-	-	-			S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	-	-	-			S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	-	-	-			S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	-	-	-			S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	-	-	-			S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	-	-	-			S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	-	-	-			S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ *Note:*

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

2.1.4.4 ACCUMULATOR

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

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

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

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

The system doesn't store ACC and PFLAG value when interrupt executed. ACC and PFLAG data must be saved to other data memories by program.

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

```
.DATA           ACCBUF      DS   1          ; Define ACCBUF for store ACC data.
                PFLAGBUF    DS   1          ; Define PFLAGBUF for store PFLAG data.
.CODE
INT_SERVICE:
                B0XCH       A, ACCBUF       ; Save ACC in a buffer.
                B0MOV       A, PFLAG
                B0MOV       PFLAGBUF, A     ; Save PFLAG register in a buffer.
                …
                …
                B0MOV       A, PFLAGBUF
                B0MOV       PFLAG, A        ; Restore PFLAG register from buffer.
                B0XCH       A, ACCBUF       ; Restore ACC from buffer.
                RETI                        ; Exit interrupt service vector
```

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

2.1.4.5 PROGRAM FLAG

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

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

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

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

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

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

2.1.4.6 PROGRAM COUNTER

The program counter (PC) is a 11-bit binary counter separated into the high-byte 3 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

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

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	-	-	-	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	-	-	-	-	-	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 can't carry to PCH when PCL overflow automatically after executing addition instructions. Users have to take care program counter result and adjust PCH value by program. For jump table or others applications, users have to calculate PC value to avoid PCL overflow making PC error and program executing error.

✸ ***Note: Program counter can't carry to PCH when PCL overflow automatically after executing addition instructions. Users have to take care program counter result and adjust PCH value by program.***

➢ **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
                …
                …
```

2.1.4.7 Y, Z REGISTERS

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

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

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

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

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

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

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

```
        B0MOV   Y, #0       ; Y = 0, bank 0
        B0MOV   Z, #07FH    ; Z = 7FH, the last address of the data memory area

CLR_YZ_BUF:
        CLR     @YZ         ; Clear @YZ to be zero

        DECMS   Z           ; Z – 1, if Z= 0, finish the routine
        JMP     CLR_YZ_BUF  ; Not zero

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

2.1.4.8 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 (Y/Z).

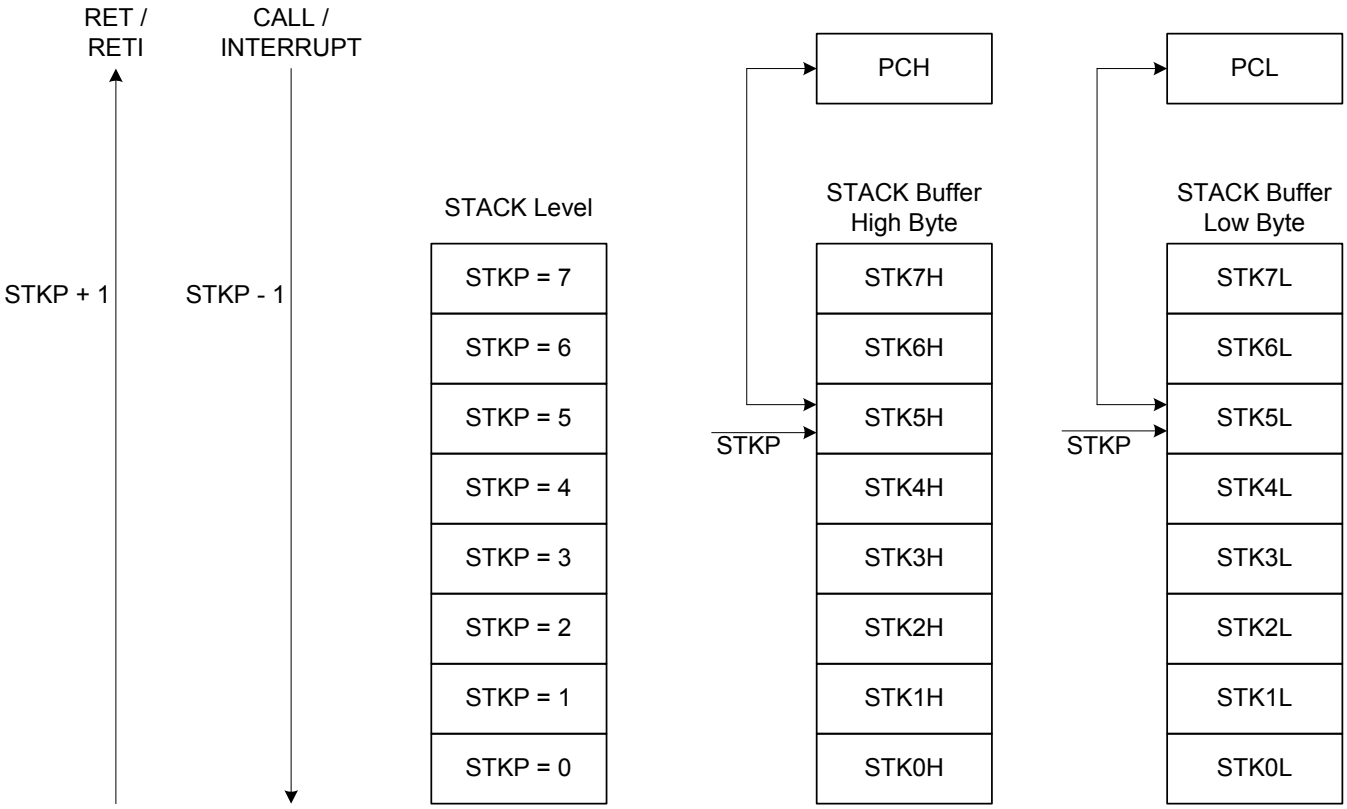
- **Example: Indirectly addressing mode with @YZ register.**

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

2.3 STACK OPERATION

2.3.1 OVERVIEW

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



2.3.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 11-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

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

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

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

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

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

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

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

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

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

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

2.3.3 STACK OPERATION EXAMPLE

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

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

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

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

3 RESET

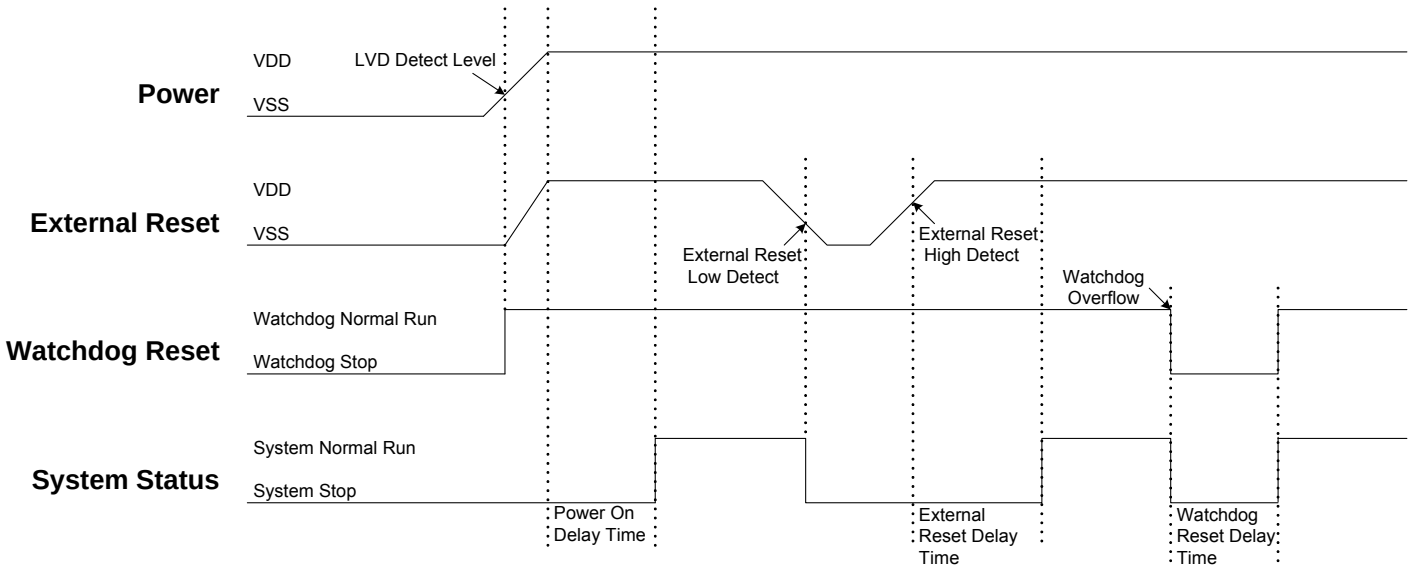
3.1 OVERVIEW

The system would be reset in three conditions as following.

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

When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0.

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



3.2 POWER ON RESET

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

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

3.3 WATCHDOG RESET

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

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

Watchdog timer application note is as following.

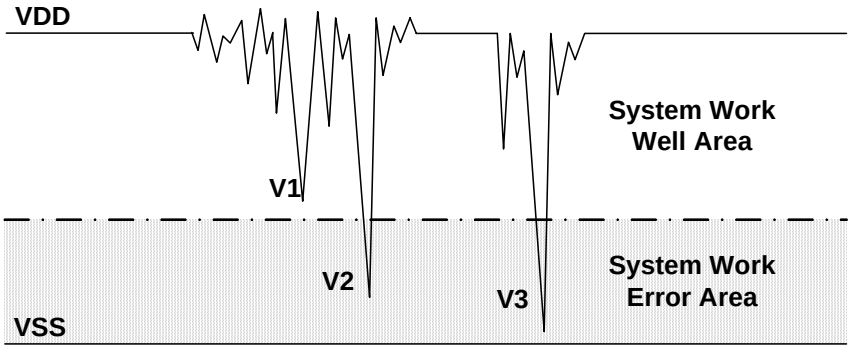
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

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

3.4 BROWN OUT RESET

3.4.1 BROWN OUT DESCRIPTION

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



Brown Out Reset Diagram

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

DC application:

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

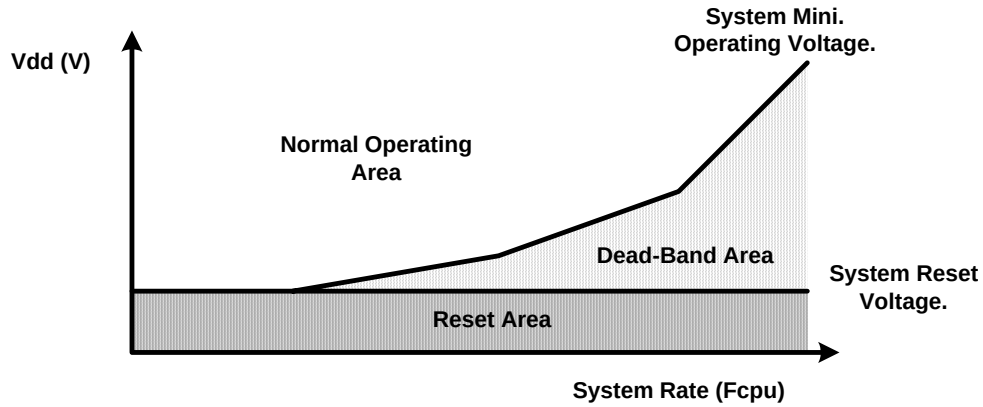
AC application:

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

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

3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION

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



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

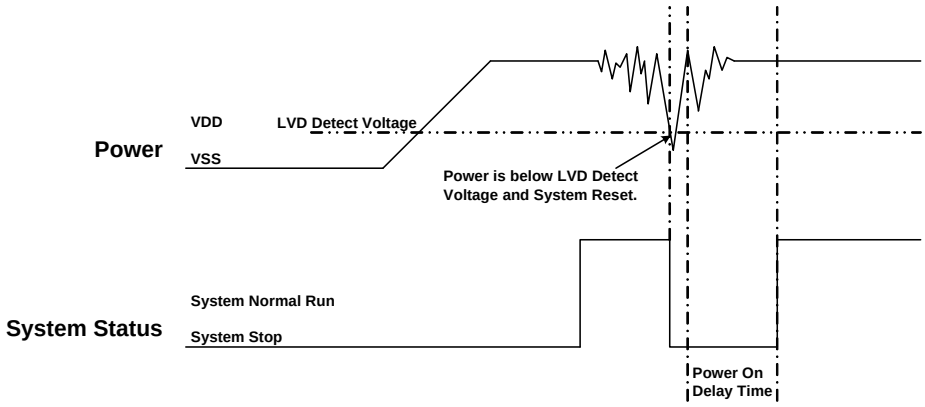
3.4.3 BROWN OUT RESET IMPROVEMENT

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

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

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

LVD reset:



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

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

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

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

3.5 EXTERNAL RESET

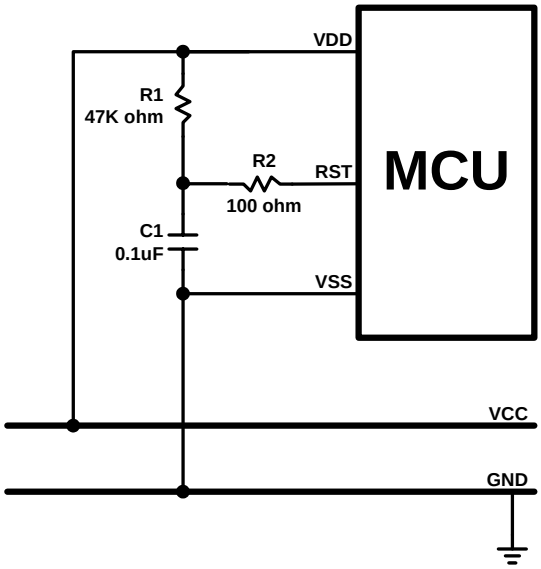
External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the low voltage and the system is reset. The external reset operation actives in power on and normal running mode. During system power-up, the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

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

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

3.6 EXTERNAL RESET CIRCUIT

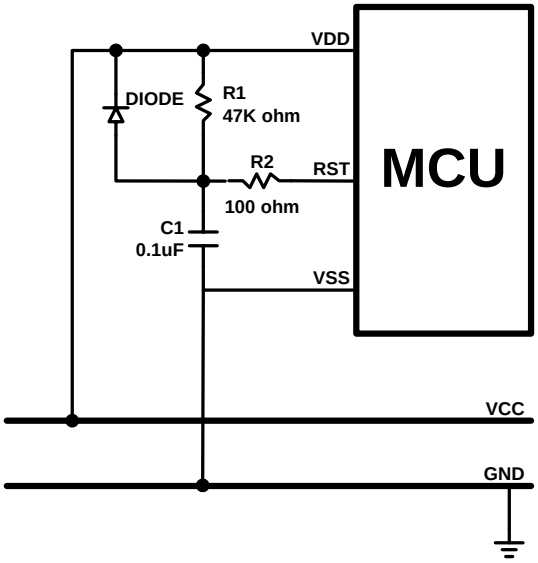
3.6.1 Simply RC Reset Circuit



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

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

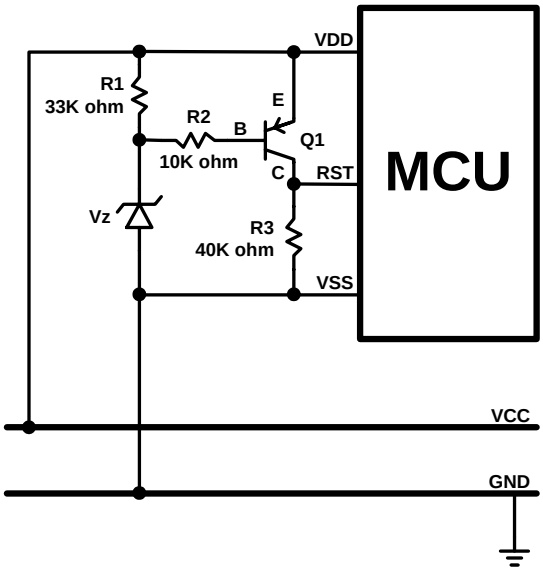
3.6.2 Diode & RC Reset Circuit



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

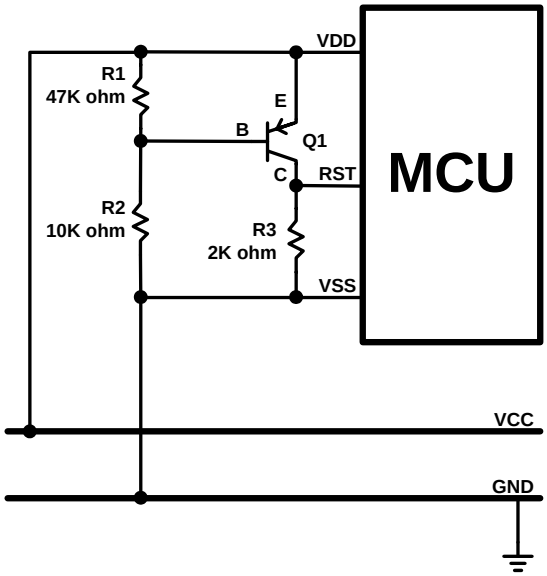
✸ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.6.3 Zener Diode Reset Circuit



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

3.6.4 Voltage Bias Reset Circuit

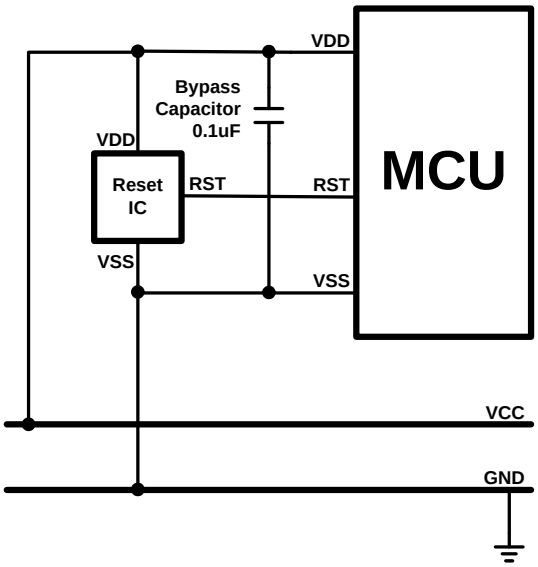


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to "0.7V x (R1 + R2) / R1",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

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

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

3.6.5 External Reset IC



4 SYSTEM CLOCK

4.1 OVERVIEW

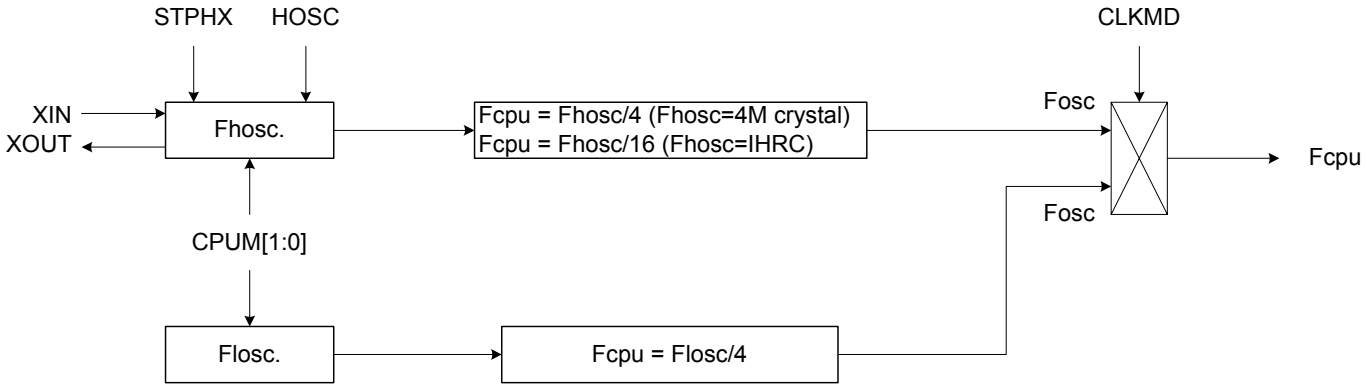
The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit or on-chip 16MHz high-speed RC oscillator circuit (IHRC 16MHz). The low-speed clock is generated from on-chip low speed RC oscillator circuit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

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

☞ **Normal Mode (High Clock):** **Fcpu = Fhosc / 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: Internal low-speed RC clock .(About 16KHz@3V, 32KHz@5V)
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.3 OSCM REGISTER

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

0CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	WTCKS	WDRST	WDRATE	-	CPUM0	CLKMD	STPHX	0
Read/Write	R/W	R/W	R/W	-	R/W	R/W	R/W	-
After reset	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] **CPUM0:** CPU operating mode control bits.
0 = normal.
1 = sleep (power down) mode.

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

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

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

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

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

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

➢ **Example: When entering the power down mode (sleep mode), both high-speed oscillator and 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	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

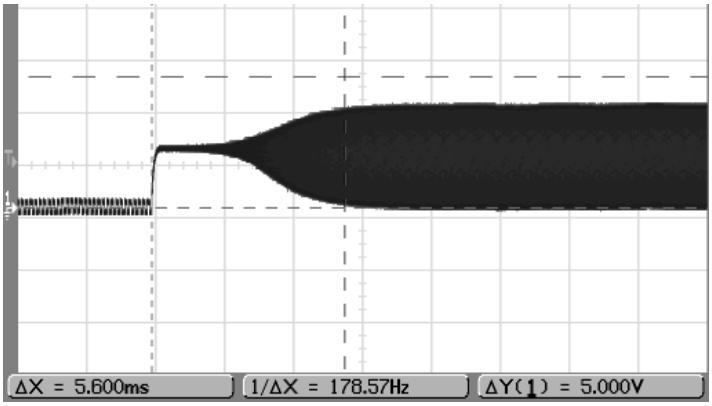
The chip is built-in RC type internal high clock (16MHz) controlled by “IHRC_16M” code options. In “IHRC_16M” mode, the system clock is from internal 16MHz RC type oscillator and XIN / XOUT pins are general-purpose I/O pins.

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

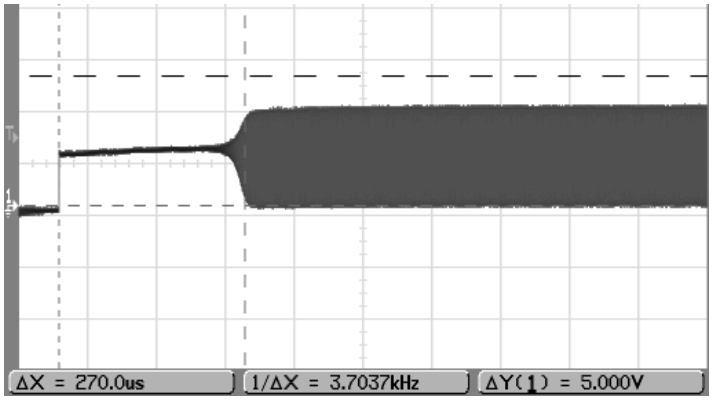
4.4.2 EXTERNAL HIGH CLOCK

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

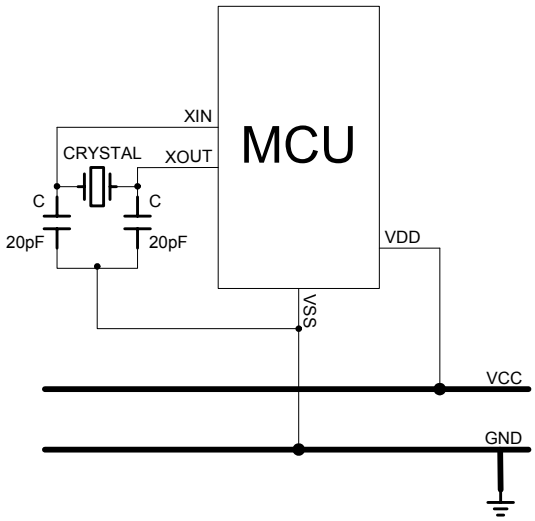


4MHz Ceramic



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. 4M option is for normal speed (ex. 4MHz).



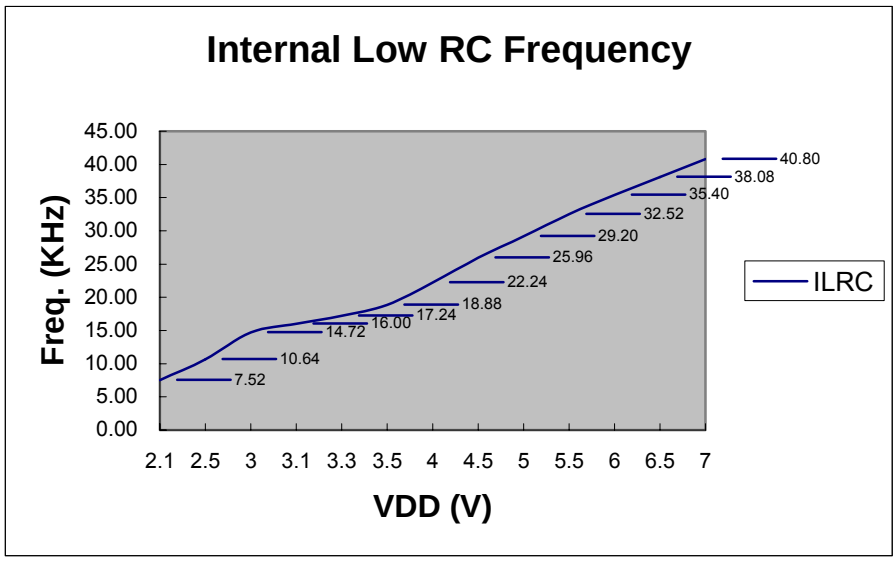
✷ ***Note: Connect the Crystal/Ceramic and C as near as possible to the XIN/XOUT/VSS pins of micro-controller.***

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.

4.5 SYSTEM LOW CLOCK

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



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

☞ ***Flosc = Internal low RC oscillator (about 16KHz @3V, 32KHz @5V).***

☞ ***Slow mode Fcpu = Flosc / 4***

There are two conditions to stop internal low RC. One is power down mode, and the other is green mode of 32K mode and watchdog disable. If system is in 32K mode and watchdog disable, only 32K oscillator actives and system is under low power consumption.

➢ **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 internal low-speed clock can't be turned off individually. It is controlled by CPUM0 bits of OSCM register.***

4.5.1 SYSTEM CLOCK MEASUREMENT

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

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

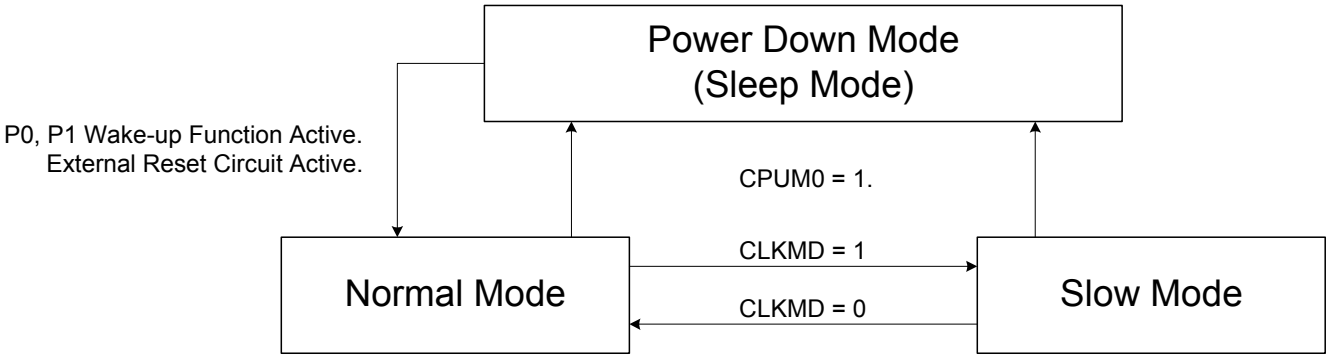
```
        B0BSET    P1M.0       ; Set P1.0 to be output mode for outputting Fcpu toggle signal.
@@:
        B0BSET    P1.0        ; Output Fcpu toggle signal in low-speed clock mode.
        B0BCLR    P1.0        ; Measure the Fcpu frequency by oscilloscope.
        JMP       @B
```

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

- High-speed mode
- Low-speed mode
- Power-down mode (Sleep mode)



System Mode Switching Diagram

Operating mode description

MODE	NORMAL	SLOW	POWER DOWN (SLEEP)	REMARK
EHOSC	Running	By STPHX	Stop	
IHRC	Running	By STPHX	Stop	
ILRC	Running	Running	Stop	
CPU instruction	Executing	Executing	Stop	
T0 timer	*Active	*Active	Inactive	* Active if T0ENB=1
Watchdog timer	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	All inactive	
External interrupt	All active	All active	All inactive	
Wakeup source	-	-	P0, P1, Reset	

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

5.2 SYSTEM MODE SWITCHING EXAMPLE

➢ **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 10mS for external clock stable.

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

        MOV         A, #27          ; If VDD = 5V, internal RC=32KHz (typical) will delay
        B0MOV       Z, A
@@:     DECMS       Z               ; 0.125ms X 81 = 10.125ms for external clock stable
        JMP         @B
                                    ;
        B0BCLR      FCLKMD          ; Change the system back to the normal mode
```

-

5.3 WAKEUP

5.3.1 OVERVIEW

Under power down mode (sleep mode) , program doesn't execute. The wakeup trigger can wake the system up to normal mode. The wakeup trigger sources are external trigger (P0, P1 level change)

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

5.3.2 WAKEUP TIME

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

The value of the wakeup time is as the following.

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

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

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

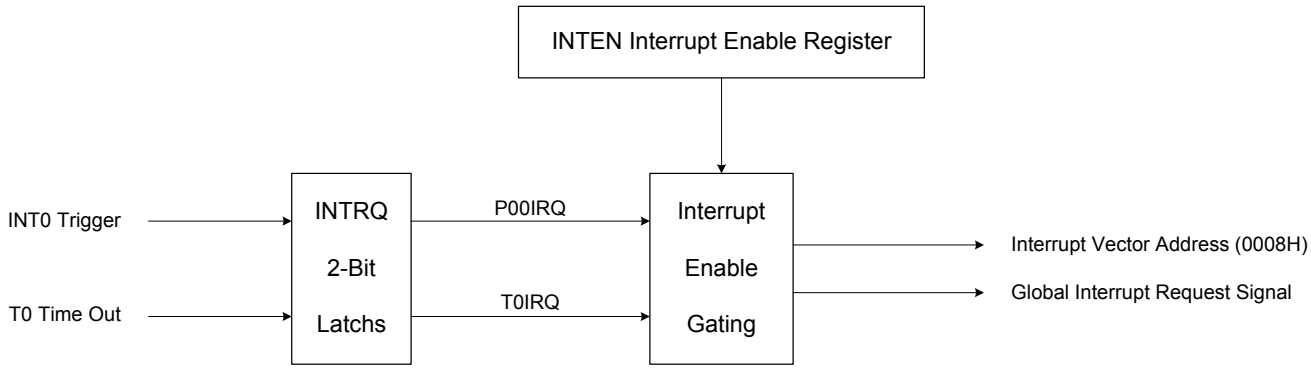
The wakeup time = 1/Fosc * 2048 = 0.512 ms (Fosc = 4MHz)

The total wakeup time = 0.512 ms + oscillator start-up time

6 INTERRUPT

6.1 OVERVIEW

This MCU provides three interrupt sources, including one internal interrupt (T0) and one external interrupt (INT0). 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	-	-	-	T0IEN	-	-	-	P00IEN
Read/Write	-	-	-	R/W	-	-	-	R/W
After reset	-	-	-	0	-	-	-	0

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

Bit 4 **T0IEN:** T0 timer interrupt control bit.
0 = Disable T0 interrupt function.
1 = Enable T0 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	-	-	-	T0IRQ	-	-	-	P00IRQ
Read/Write	-	-	-	R/W	-	-	-	R/W
After reset	-	-	-	0	-	-	-	0

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

Bit 4 **T0IRQ:** T0 timer interrupt request flag.
0 = None T0 interrupt request.
1 = T0 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 doesn't have any special instructions to process ACC, PFLAG registers when into interrupt service routine. Users have to save ACC, PFLAG by program, Using "B0XCH" to save/load ACC buffer, "B0MOV" to save/load PFLAG and avoid main routine error after interrupt service routine finishing.

✸ ***Note: To save/load ACC data, users must be "B0XCH" instruction, or else the PFLAG register might be modified by ACC operation.***

➢ **Example: Store ACC and PAFLG data by program when interrupt service routine executed.**

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

.CODE
            ORG        0
            JMP        START

            ORG        8
            JMP        INT_SERVICE

            ORG        10H
START:
            …

INT_SERVICE:
            B0XCH      A, ACCBUF     ; Save ACC to ACCBUF buffer.
            B0MOV      A, PFLAG
            B0MOV      PFLAGBUF, A   ; Save PFLAG to PFLAGBUF buffer.
            …
            …
            B0MOV      A, PFLAGBUF
            B0MOV      PFLAG, A      ; Load PFLAG from PFLAGBUF buffer.
            B0XCH      A, ACCBUF     ; Load ACC from ACCBUF 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.
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 T0 INTERRUPT OPERATION

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

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

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

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

        B0BSET      FGIE        ; Enable GIE
```

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

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

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

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

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

        RETI                    ; Exit interrupt vector
```

6.8 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
T0IRQ	T0C 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         INTT0CHK        ; Jump check to next interrupt
                B0BTS0      FP00IRQ         ; Check P00IRQ
                JMP         INTP00          ; Jump to INT0 interrupt service routine
INTT0CHK:                                   ; Check T0 interrupt request
                B0BTS1      FT0IEN          ; Check T0IEN
                JMP         INT_EXIT        ; Jump to exit of IRQ
                B0BTS0      FT0IRQ          ; Check T0IRQ
                JMP         INTT0           ; Jump to T0 interrupt service routine
INT_EXIT:
                …                           ; Pop routine to load ACC and PFLAG from buffers.

                RETI                        ; Exit interrupt vector
```

7 I/O PORT

7.1 I/O PORT MODE

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

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

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 4 is Input only port.***
3. ***Port 5 is output only port.***
4. ***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

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

✷ ***Note: Pull up Resistance of Port 0 and Port 4 is always existance.***

➢ **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	-	-	-	-	-	-	-	P00
Read/Write	-	-	-	-	-	-	-	R/W
After reset	-	-	-	-	-	-	-	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	R	R
After reset	-	-	-	-	-	0	0	0

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

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

```
B0MOV   A, P0     ; Read data from Port 0
B0MOV   A, P1     ; Read data from Port 4
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   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	-	CPUM0	CLKMD	STPHX	0
Read/Write	R/W	R/W	R/W	-	R/W	R/W	R/W	-
After reset	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
```

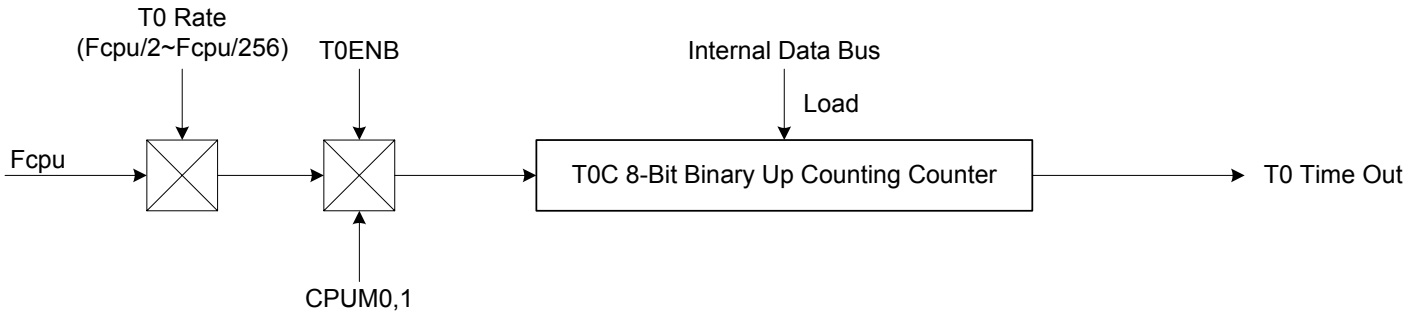
8.2 TIMER 0 (T0)

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.
- **Green mode wakeup function:** T0 can be green mode wake-up time as T0ENB = 1. System will be wake-up by T0 time out.



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

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

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

8.2.3 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

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 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.
```

9 LCD DRIVER

There are 4 common pins and 12 segment pins in the SN8P1917. The LCD scan timing is 1/4 duty and 1/2 bias or 1/3 bias structure to yield 48 dots LCD driver.

9.1 LCDM1 REGISTER

LCDM1 register initial value = xx0x 0011

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

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

Bit1 **LCDRATE:** LCD clock rate control when LCD CLK=1.
0 = LCD clock= Internal RC/ 64
1 = LCD clock= Internal RC/ 32

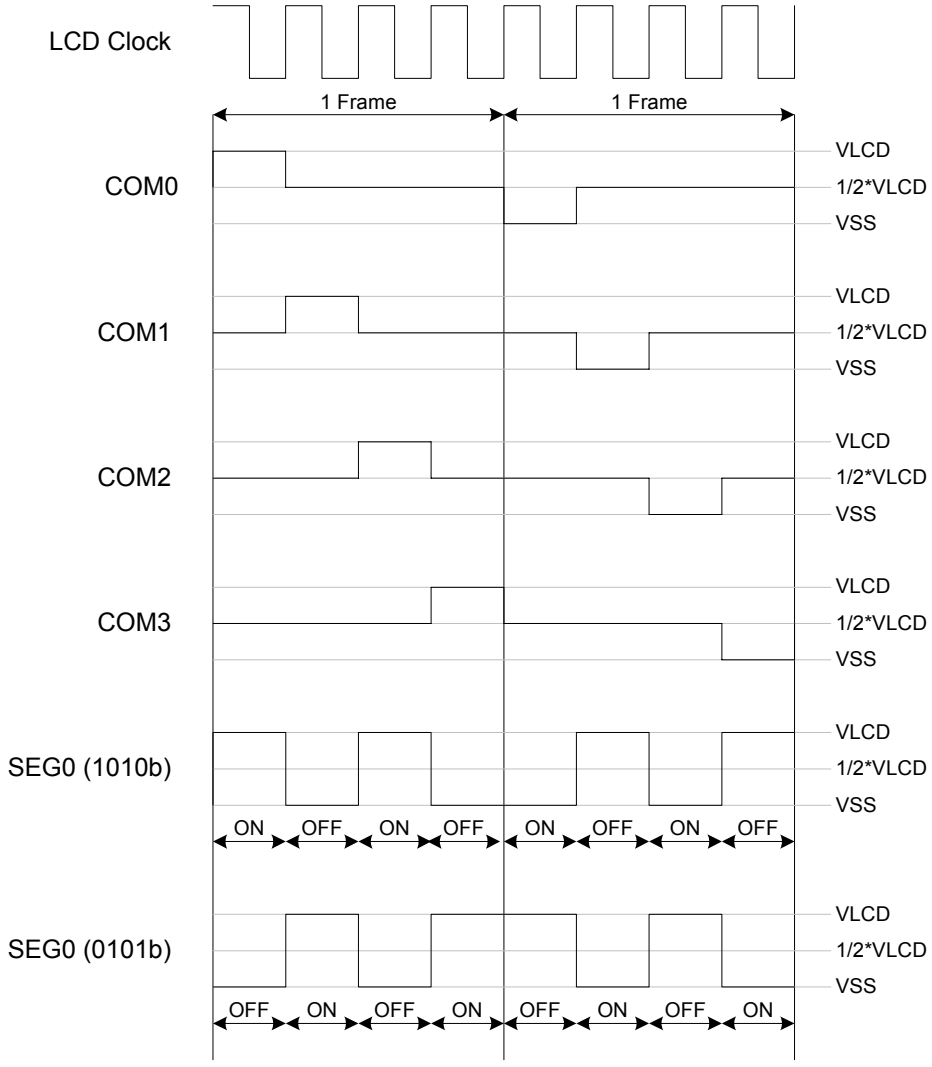
Bit1 **LCDCLK:** LCD clock source slection control bit.
0 = LCD clock = External Clock /2^14, Frame rate = LCD clock / 4
Ex. High clock = 4M, LCD clock = 244.14 Hz , Frame rate= 244.14/4=61.03
High clock = 3.58M, LCD clock = 218.51 Hz , Frame rate= 218.51/4=54.62
1 = LCD clock = Internal RC /32(LCDRATE=1) or Internal RC = 64 (LCDRATE=0),
Frame rate = LCD clock/4

- ***Note 1: In Dice form package of SN8P1917, two external pads of V1/V2 are available for fine tune the LCD bias voltage and current.***
- ***Note2: In 1/3 bias setting V1=1/3 VLCD, V2=2/3 VLCD. In 1/2 bias setting, please short V1 and V2, then V1=V2=1/2VLCD.***
- ***Note3: Pads V1/V2 only available in Dice form.***

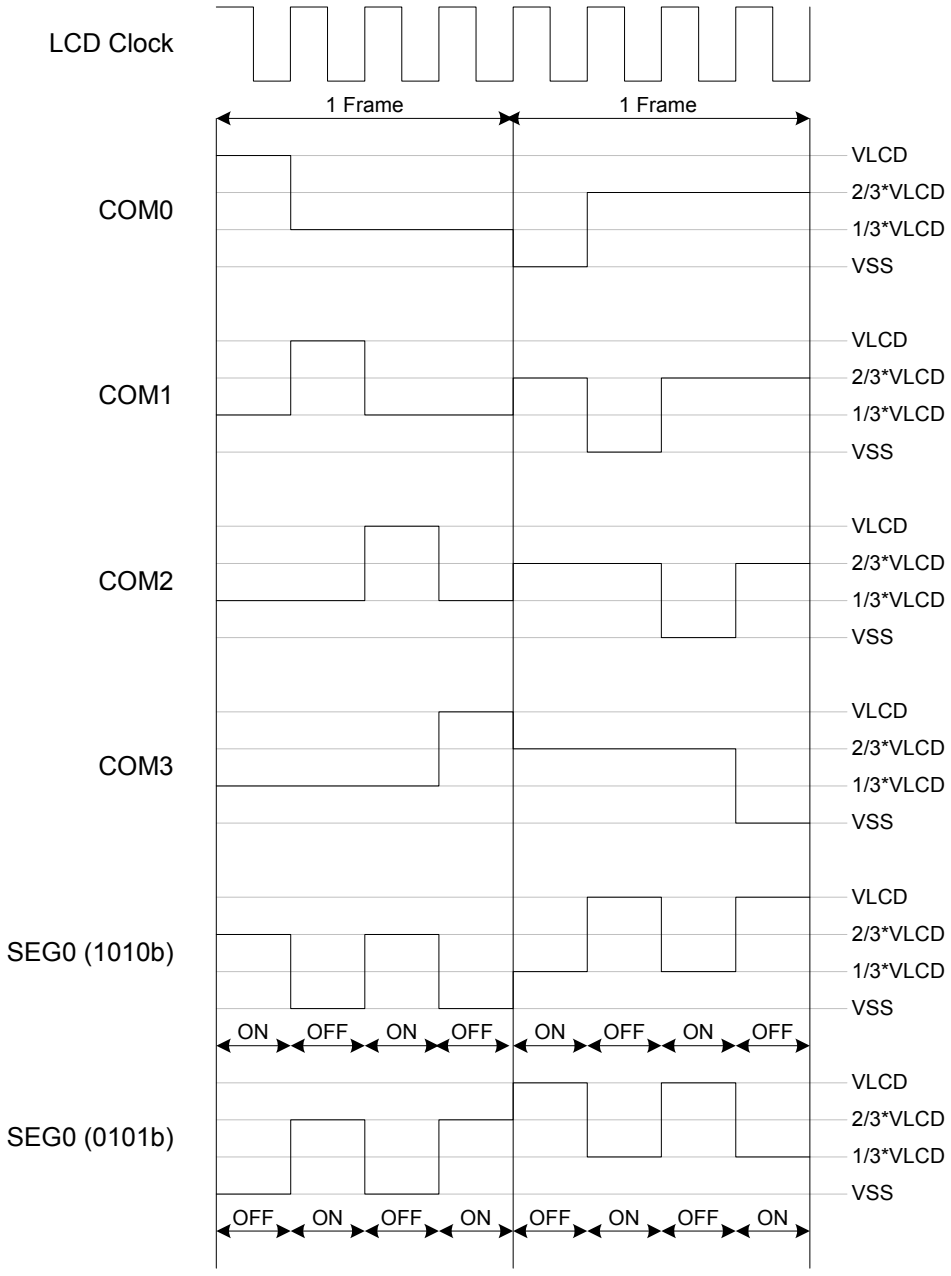
9.2 LCD TIMING

LCD Timing Table

LCDCLK	LCD clock source	LCDRATE	LCD Clock	Frame=LCD clock/4
0	Fhosc	X	4M/2^14=244.14Hz@4M	244.14/4=61.03Hz
0	Fhosc	X	3.58M/2^14=218.51Hz@3.58M	218.51/4=54.6Hz
1	Flosc	0	16K/64=250Hz@3V	250/4=62.5Hz
1	Flosc	1	16K/32=500Hz@3V	500/4=125Hz
1	Flosc	0	32K/64=500Hz@5V	500/4=125Hz
1	Flosc	1	32K/32=1000Hz@3V	1000/4=250Hz



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



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

9.3 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 11	11H.0	11H.1	11H.2	11H.3	-	-	-	-

➢ **Example: Enable LCD function.**

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

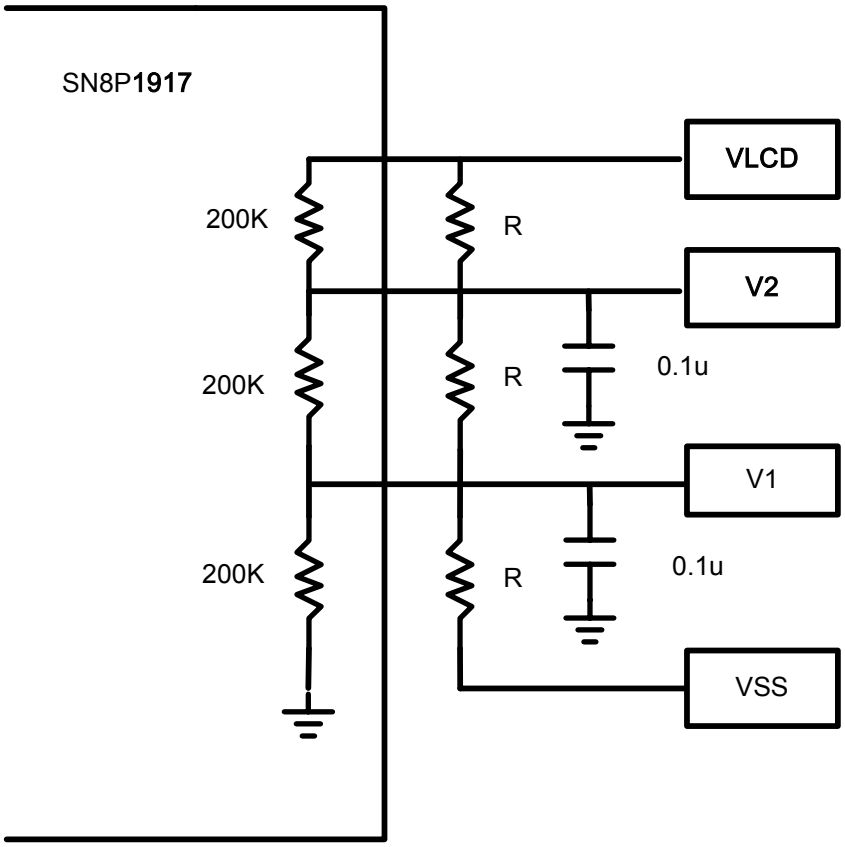
```
B0BSET      FLCDENB      ; LCD driver.
```

9.4 LCD Circuit

SN8P1917 in the LCD electric circuit, builds in 200K ohm Voltage-division resistance. User can add resistance between VLCD / V2 / V1 / VSS for more driving current.

Note: V1, V2 only available for Dice form, not support package type.

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

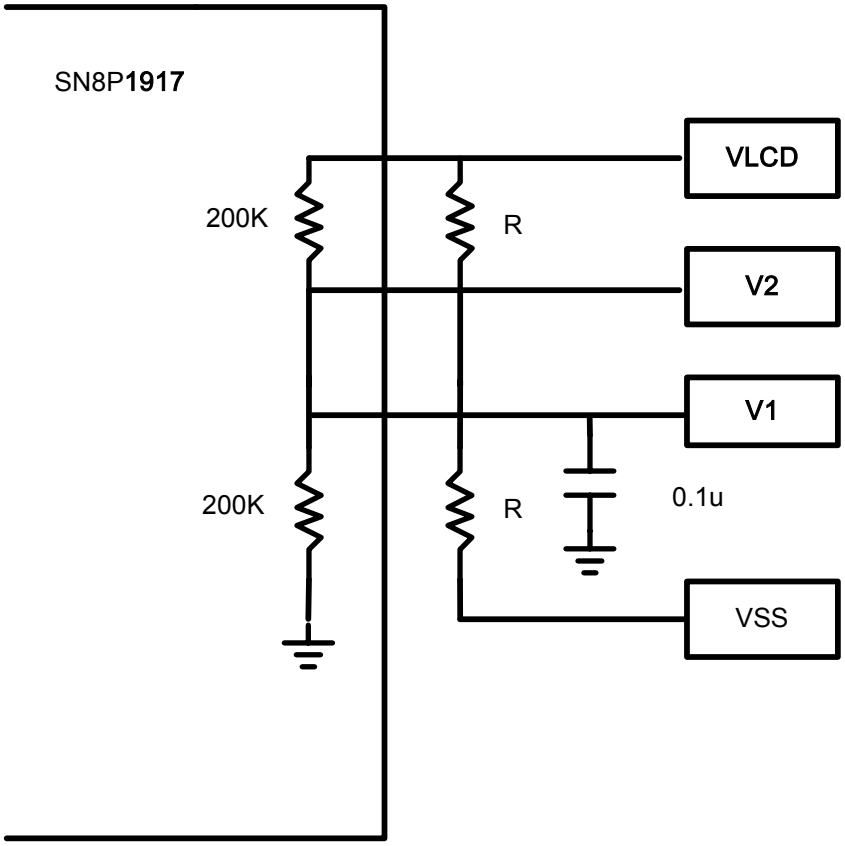


$$\text{LCD Current consumption} = \frac{\text{VLCD}}{\left(\frac{200\text{K}\times\text{R}}{200\text{K}+\text{R}}\right)\times 3}$$ 。

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

Note: **V1=1/3*VLCD、V2=2/3*VLCD。**

LCD Circuit, 1/4 duty, 1/2 bias



$$\text{LCD Current consumption} = \frac{\text{VLCD}}{\left(\frac{200\text{K}\times\text{R}}{200\text{K}+\text{R}}\right)\times 2}$$ 。

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

Note: **V1=V2=1/2*VLCD**。

10 Charge-Pump, PGIA and ADC

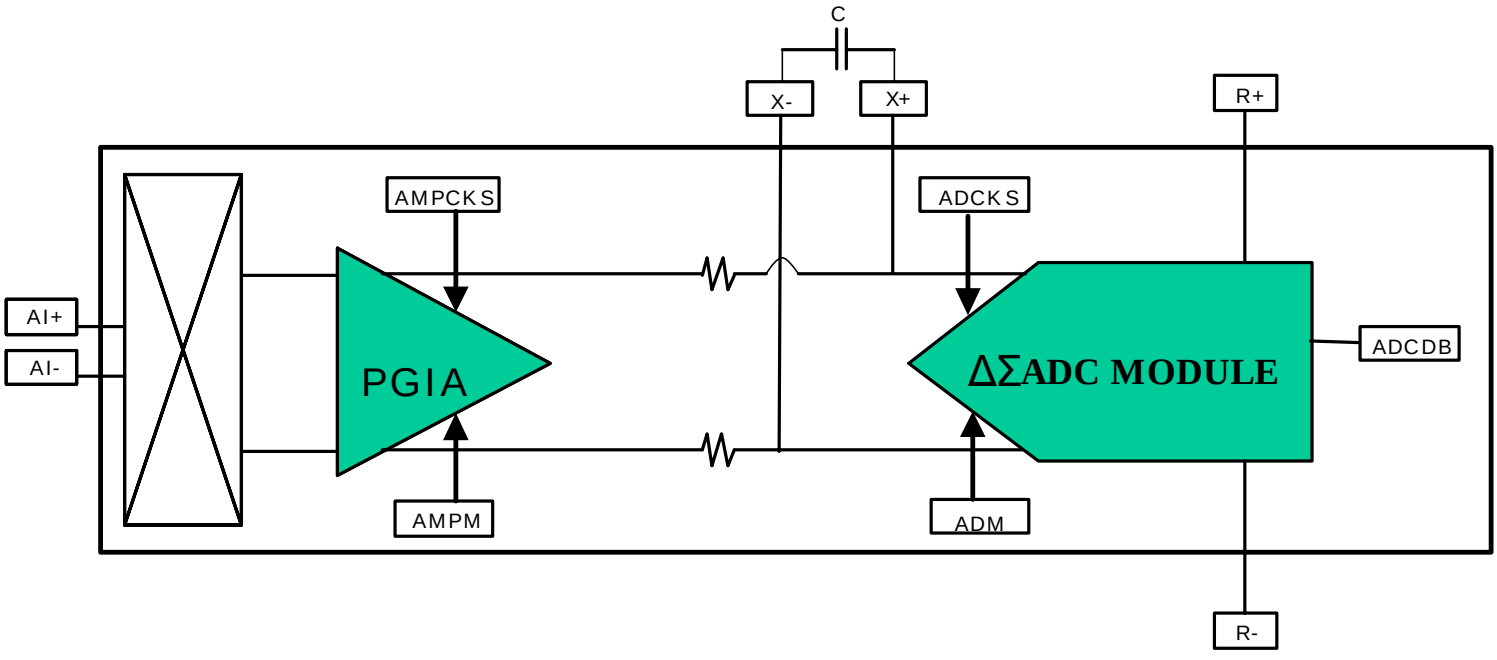
10.1 OVERVIEW

The SN8P1917 has a built-in Voltage Charge-Pump/Regulator (CPR) to support a stable voltage 3.8V from pin AVDDR and 3.0V/1.5V from pin AVE+ with maximum 10mA current driving capacity. This CPR provides stable voltage for internal circuits (PGIA, ADC) and external sensor (load cell or thermistor). The SN8P1917 series also integrated Δ Σ Analog-to-Digital Converters (ADC) to achieve 16-bit performance and up to 62500-step resolution. The ADC has 2 different input channel modes: (1) One fully differential inputs (2) Two 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.

10.2 ANALOG INPUT

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

- ***Note 1: The low pass filter (R, R and C) will filter out chopper frequency of PGIA.***
- ***Note 2: The recommend value of C is 0.01 μF. This capacitor needs to place as close chip as possible.***

10.3 Voltage Charge Pump / Regulator (CPR)

SN8P1917 is built in a CPR, which can provide a stable 3.8V (pin AVDDR) and 3.0V/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.

10.3.1 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	AVESEL	AVENB	CPSTS	CPAUTO	CPON	CPRENB
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

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: **CPSTS:** Charge-Pump status bit in AUTO Mode (Only available when CPAUTO = “1”)
0 = Charge-Pump is OFF in Auto mode.
1 = Charge-Pump is ON in Auto mode.

Bit4: **AVENB:** AVE+ voltage output control bit.
0 = Disable AVE+ output Voltage
1 = Enable AVE+ output Voltage

Bit5: **AVESEL**: AVE+ voltage selection control bit.
0 = AVE+ output 1.5V
1 = AVE+ output 3.0V

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 PGIA and ADC , Must enable Band Gap Reference (BGRENB=1), ACM (ACMENB=1) and AVDDR(AVDDRENB).***
- ***Note5: CPR, PGIA and ADC 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	CPSTS	AVDDR	PGIA, ADC Function
0	X	X	0	OFF	OFF	N/A	0V	**Not Available**
1	0	0	1	OFF	ON	N/A	**See Note1**	**See Note1**
1	0	1	1	Auto Mode	ON	0/1	3.8V	Available
1	1	0	1	ON	ON	N/A	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	CPSTS	Regulator Status	AVDDR Output	PGIA, ADC Function
1	0	1	1	>4.1V	OFF	0	ON	3.8V	Available
				≦4.1V	ON	1	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 (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

10.3.2 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 [4:0] register sets the Charge-Pump working clock; the suggestion Charge-Pump clock is 13K~15.6 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	High Clock			
				2M	3.58M	4M/IHRC	8M
0	0	0	0	125K	223.75K	250K	500K
0	0	0	1	62.5K	111.88K	125K	250K
0	0	1	0	31.25K	55.94K	62.5K	125K
0	0	1	1	15.625K	27.97K	31.25K	62.5K
0	1	0	0	7.8125K	13.985K	15.625K	31.25K
0	1	0	1	3.90625K	6.99K	7.8125K	15.625K
0	1	1	0	1.953215K	3.495K	3.90625K	7.8125K
0	1	1	1	0.976K	1.75K	1.953215K	3.90625K
1	0	0	0	0.488K	0.875K	0.976K	1.953215K
1	0	0	1	0.244K	0.438K	0.488K	0.976K
1	0	1	0	0.122K	0.219K	0.244K	0.488K
1	0	1	1	0.61K	0.11K	0.122K	0.244K
1	1	0	0	0.3K	0.055K	0.061K	0.122K
1	1	0	1	0.15K	0.028K	0.03K	0.61K
1	1	1	0	0.075K	0.014K	0.015K	0.3K
1	1	1	1	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 (Internal Low Clock mode).***
- ***Note3: The Faster of Charge pump clock, AVE+ can load more 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, #00100100B   ;
            XB0MOV      CPM, A          ; Set AVE+=3.0V ,CP as Auto mode and Disable AVDDR,
                                        AVE+, ACM voltage and 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 5ms for ACM Voltage Stabilize

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

@AVDDR_Enable:
            XB0BSET     FAVDDRENB       ; Enable AVDDR Voltage=3.8V
            CALL        @Wait_50ms      ; Delay 50ms for AVDDR Voltage Stabilize

@AVE_Enable:
            XB0BSET     FAVENB          ; Enable AVE+ Voltage=3.0V/1.5V
            CALL        @Wait_50ms      ; Delay 50ms 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.***

10.4 PGIA -Programmable Gain Instrumentation Amplifier

SN8P1917 includes a low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selection gains of 1x, 12.5x, 50x, 100x, and 200x by register AMPM. The PGIA also provides two types channel selection mode: (1) One fully differential input (2) Two single-ended inputs, it was defined by register AMPCHS.

10.4.1 AMPM- Amplifier Mode Register

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

10.4.2 AMPCKS- PGIA CLOCK SELECTION

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPCKS	-	-	-	-	-	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.***

10.4.3 AMPCHS-PGIA CHANNEL SELECTION

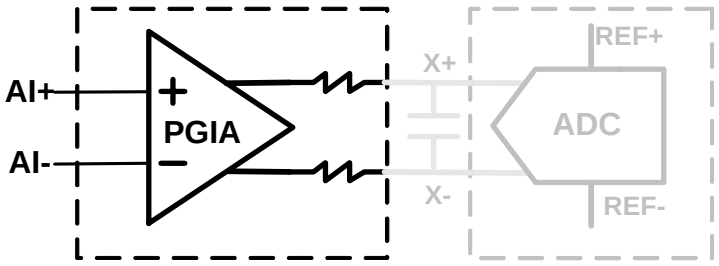
091H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPCHS	-	-	-	-	-	CHS2	CHS1	CHS0
	-	-	-	-	-	R/W	R/W	R/W

CHS [2:0]: PGIA Channel Selection

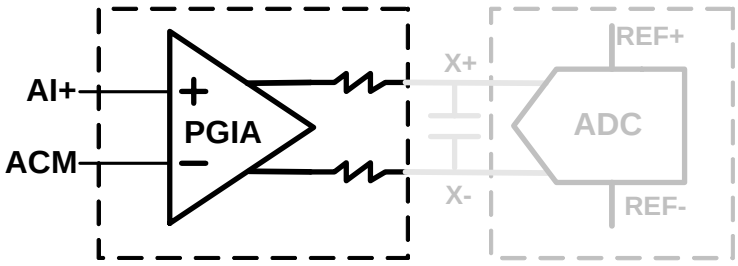
CHS [2:0]	Selected Channel	V (X+, X-) Output	Input-Signal Type
000	AI+, AI-	V (AI+, AI-) × PGIA Gain	Differential
001	AI+, ACM	V (AI+, ACM) × PGIA Gain	Single-ended
010	AI-, ACM	V (AI-, ACM) × PGIA Gain	Single-ended
011	ACM, ACM	V (ACM, ACM) × PGIA Gain	Input-Short
100	Reserved	-	-
101	Temperature Sensor	V (V_{TS}, 0.8V) × 1	N/A
110,111			

- ***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.***

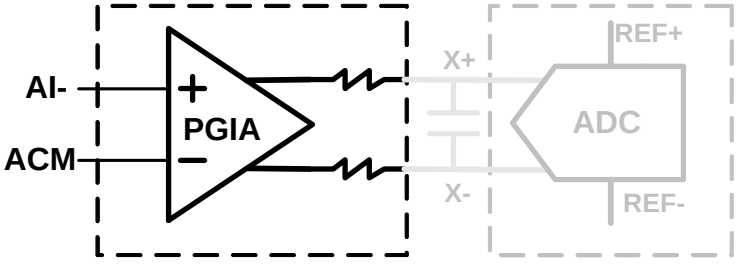
AMPCHS[2:0]="000"



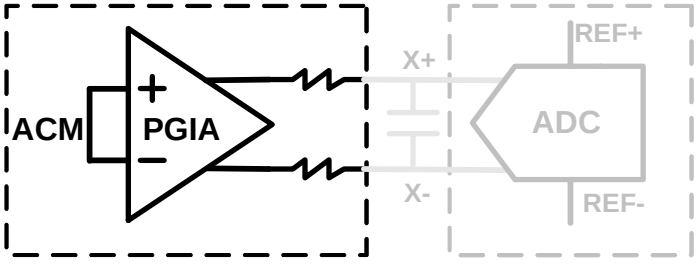
AMPCHS[2:0]="001"



AMPCHS[2:0]="010"

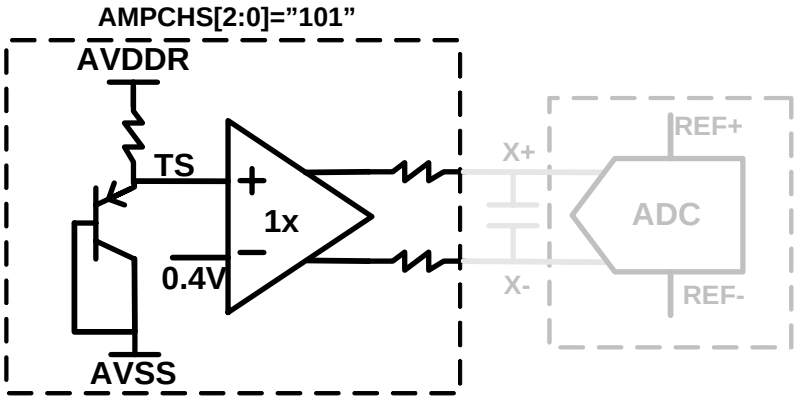


AMPCHS[2:0]="011"



10.4.4 Temperature Sensor (TS)

In applications, sensor characteristic might change in different temperature also. To get the temperature information, SN8P1917 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 0.4V.***
- ***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 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 system.

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

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, #00100100B   ;
            XB0MOV      CPM, A          ; Set AVE+=3.0V ,CP as Auto mode and Disable AVDDR,
                                        AVE+, ACM voltage and 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 5ms for ACM Voltage Stabilize

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

@AVDDR_Enable:
            XB0BSET     FAVDDRENB       ; Enable AVDDR Voltage=3.8V
            CALL        @Wait_50ms      ; Delay 50ms for AVDDR Voltage Stabilize

@AVE_Enable:
            XB0BSET     FAVENB          ; Enable AVE+ Voltage=3.0V/1.5V
            CALL        @Wait_50ms      ; Delay 50ms for AVE+ Voltage Stabilize

@PGIA_Init:
            MOV         A, #01110110B
            XB0MOV      AMPM, A         ; Enable Band Gap, Set :FDS="11" and 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= AI+, AI-
@PGIA_Enable:
            XB0BSET     FAMPENB         ; Enable PGIA function
            …                           ; V (X+, X-) Output = V (AI+, AI-) 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, #01110110B
        XB0MOV  AMPM, A         ; Enable Band Gap, Set :FDS="11" and 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= AI+, AI-
@PGIA_Enable:
        XB0BSET FAMPENB         ; Enable PGIA function
        …                       ; V (X+, X-) Output = V (AI+, AI-) x 200

@PGIA_Sensor:
        MOV     A, #01110111B   ;Don't Disable PGIA when change PGIA CH.
        XB0MOV  AMPM, A         ; Enable Band Gap, Set :FDS="11" and PGIA Gain=200
        MOV     A, #00000000B
        XB0MOV  AMPCHS, A       ; Selected PGIA as Differential channel.
        …                       ; V (X+, X-) Output = V(AI+,AI-) x 200

@PGIA_TS:
        MOV     A, #01110001B   ;Don't Disable PGIA when change PGIA CH.
        XB0MOV  AMPM, A         ; Enable Band Gap, Set :FDS="11" and PGIA Gain=1x
        MOV     A, #00000101B
        XB0MOV  AMPCHS, A       ; Selected PGIA as Temperature Sensor ch.
        …                       ; V (X+, X-) Output = V (TS, 0.4) x 1.
```

10.5 16-Bit ADC

10.5.1 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
	-	-	-	-	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+/1.33 (When AVE+=3.0V, V(REF+,REF-)=0.4V)
1 = Internal Reference Voltage V(REF+,REF-) is AVE+/2.660. (When AVE+=3.0V, V(REF+,REF-)=0.8V)

IRVS	RVS1	RVS0	AD Reference Voltage		AD Channel Input		Note
			REF+	REF-	ADCIN+	ADCIN-	
X	0	0	R+	R-	X+	X-	V (X+, X-) < V (R+, R-)
0	1	0	0.8V	0.4V			V (X+, X-) < 0.4V (AVE+=3.0V)
1	1	0	1.2V	0.4V			V (X+, X-) < 0.8V (AVE+=3.0V)
1	1	0	0.6V	0.2V			V (X+, X-) < 0.4V (AVE+=1.5V)
X	0	1	R+	R-	VDD *3/16	VDD* 2/16	ADC input = 1/16 VDD For battery monitor (AVE+=3.0V)
0	1	1	0.8V	0.4V			
1	1	1	1.2V	0.4V			

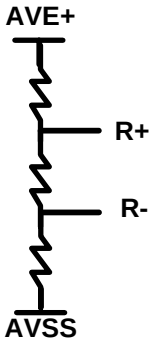
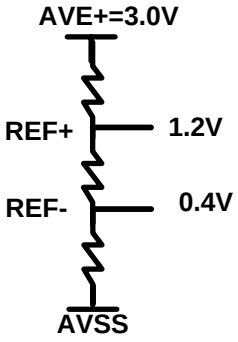
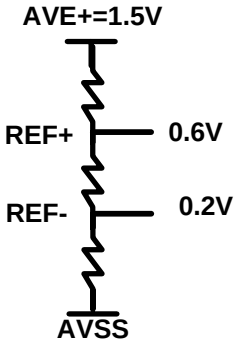
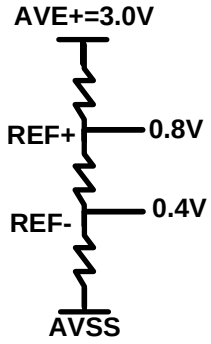
✸ ***Note1: The ADC conversion data is combined with ADCDH and ADCDL register in 2's compliment with sign bit numerical format, and Bit ADCB15 is the sign bit of ADC data. Refer to following formula to calculate ADC conversion data value.***
✸ ***Note2: The Internal Reference Voltage is divided from AVE+, so the voltage will follow the changing with AVE+(3.0V/1.5V).***

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

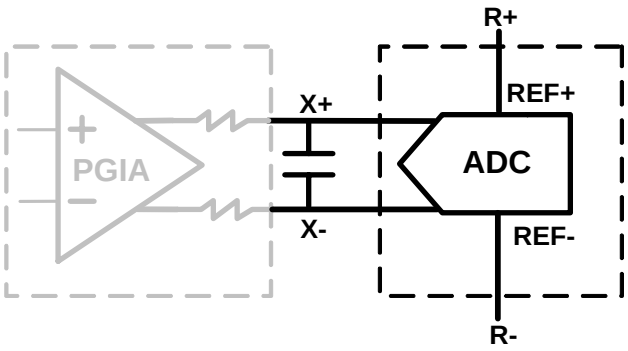
$$(ADCIN+) < (ADCIN-) \Rightarrow ADCConversionData = -\frac{(ADCIN+)-(ADCIN-)}{(REF+)-(REF-)} X\,31250$$

✸ ***Note2: The internal 1.2V, 0.8V and 0.4V reference voltage are generated from Band Gap reference voltage.***

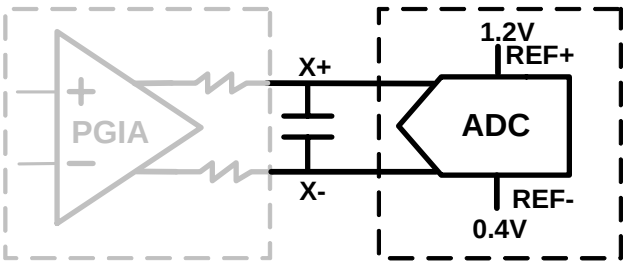
External and Internal Reference Circuit Table:

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

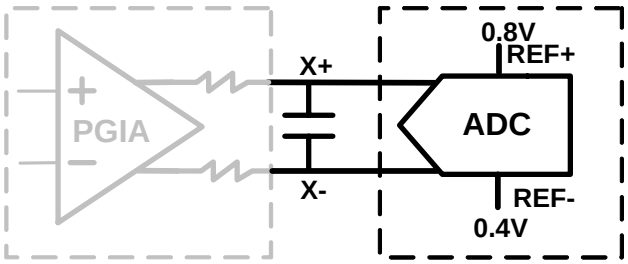
ADCM=#xxxxx00xB, V(REF+, REF-) = V(R+, R-), ADC Reference Voltage from External R+,R-.



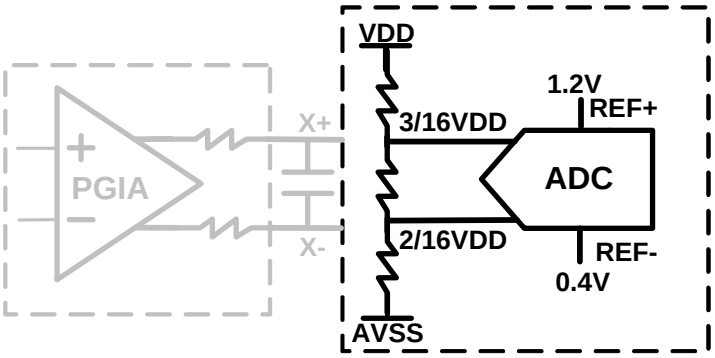
ADCM=#xxxx110xB, V(REF+, REF-) = V(1.2V, 0.4V)=0.8V (AVE+=3.0V) ADC Reference Voltage from Internal 1.2V and 0.4V.



ADCM=#xxxx010xB, V(REF+, REF-) = V(0.8V, 0.4V)=0.4V (AVE+=3.0V), ADC Reference Voltage from Internal 0.8V and 0.4V.



ADCM=#xxxx111xB, 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.



10.5.2 ADCKS- ADC Clock Register

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

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

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

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

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

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

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

10.5.3 ADCDL- ADC Low-Byte Data Register

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

10.5.4 ADCDH- ADC High-Byte Data Register

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

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

.

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

10.5.5 DFM-ADC Digital Filter Mode Register

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

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

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

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

- ***Note 1: When STOD=0, the ADC is designed for continuous mode, so it needn't read each conversion data every time. User only needs to read the data when conversion result is required. Note that if the conversion data is not read immediately, it could be lost and be replaced by the next new conversion data.***
- ***Note2: When STOD = 1, although no more conversion word will update to ADC output buffer, but ADC is still working and will output conversion word when STOD= 0.***
- ***Note 3: AC power 50 Hz noise will be filter out when output word rate = 25Hz***
- ***Note 4: AC power 60 Hz noise will be filter out when output word rate = 20Hz***
- ***Note 5: Both AC power 50 Hz and 60 Hz noise will be filter out when output word rate = 10Hz***
- ***Note 6: Clear Bit DRDY after got ADC data or this bit will keep High all the time.***
- ***Note 7: 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, #00100100B   ;
            XB0MOV      CPM, A          ; Set AVE+=3.0V ,CP as Auto mode and Disable AVDDR,
                                        AVE+, ACM voltage and 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 5ms for ACM Voltage Stabilize
@ACM_Enable:
            XB0BSET     FACMENB         ; Enable ACM Voltage=1.2v
            CALL        @Wait_5ms       ; Delay 5ms for ACM Voltage Stabilize
@AVDDR_Enable:
            XB0BSET     FAVDDRENB       ; Enable AVDDR Voltage=3.8V
            CALL        @Wait_50ms      ; Delay 50ms for AVDDR Voltage Stabilize
@AVE_Enable:
            XB0BSET     FAVENB          ; Enable AVE+ Voltage=3.0V/1.5V
            CALL        @Wait_50ms      ; Delay 50ms for AVE+ Voltage Stabilize
@PGIA_Init:
            MOV         A, #01110110B
            XB0MOV      AMPM, A         ; Enable Band Gap, Set :FDS="11" and 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= AI+, AI-
@PGIA_Enable:
            XB0BSET     FAMPENB         ; Enable PGIA function
            …                           ; V (X+, X-) Output = V (AI+, AI-) x 200
@ADC_Init:
            MOV         A, #00000110B
            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, #00000110B
            XBMOV       ADCM, A             ; Selection ADC Reference voltage = V(R+, R-)
            MOV         A, #0216
            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, #00011011B       ;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, #00011001B       ;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.
            …
```

10.5.6 LBTM : Low Battery Detect Register

SN8P1917 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 P4.2, bit LBTO will output the P4.2 voltage Higher or Lower than ACM(1.2V)

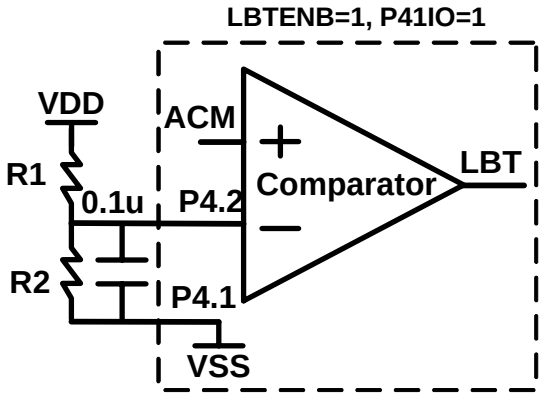
09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LBTM	-	-	-	-	-	LBTO	P41IO	LBTENB
R/W	-	-	-	-	-	R	R/W	R/W
After Reset	-	-	-	-	-	0	0	0

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

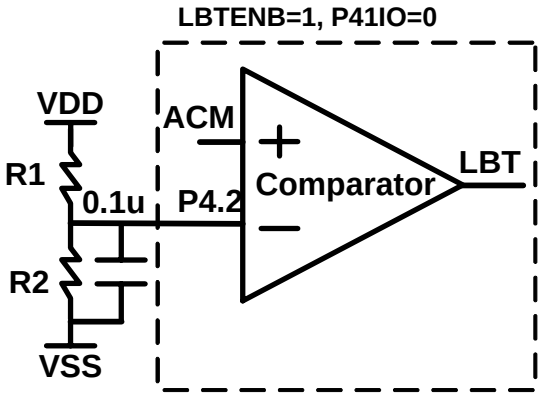
Bit1: **P41IO:** Port 4.1 Input/LBT function control bit.
0 = Set P41 as Input Port,
1 = Set P41 as LBT function

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 P4.2 and P4.1, which can avoid power consumption in sleep mode, the another is using P4.2 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:



P4.1 as LBT function, No leakage current in sleep mode



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 or more to make sure the Low Battery signal is stable.***

10.5.7 Analog Setting and Application

The most applications of SN8P1917 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

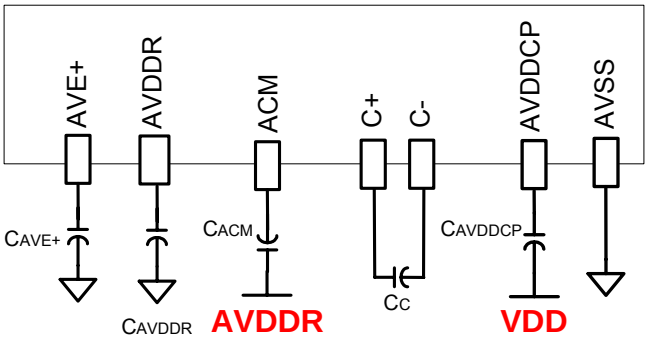
Capacitor Table:

Power type	AI+	AI-	X+/X-	R+/R-	ACM	AVDDR	AVE+	AVDDCP	C+/C-	VDD (Pin23)	VDD (Pin28)
	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	0.01uF	0.1uF	1uF	1uF	2.2uF	10uF	1uF	10uF	0.1uF
CR2032 ((4.4~6V))	0.1uF	0.1uF	0.01uF	0.1uF	1uF	1uF	2.2uF	No	No	10uF	0.1uF
AA/AAA Bat.(2.4~3V)	0.1uF	0.1uF	0.01uF	0.1uF	1uF	1uF	4.7uF	10uF	1uF	10uF	0.1uF
AA/AAA Bat.(4.4~6V)	0.1uF	0.1uF	0.01uF	0.1uF	1uF	1uF	4.7uF	No	No	10uF	0.1uF
External 5V Reg.	0.1uF	0.1uF	0.01uF	0.1uF	1uF	1uF	4.7uF	No	No	10uF	0.1uF

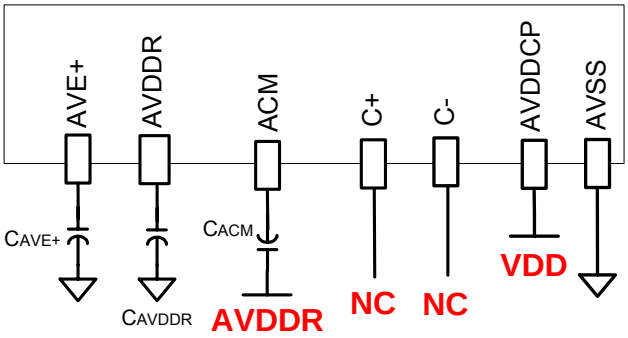
- ***Note 1: When MCU source from CR2032battery, the AVE+ loading can't over 3mA, for example the Load cell resistance can't over 1K.***
- ***Note 2: In AA/AAA battery application, the AVE+ can loading 10mA current, so that the Load cell can be up to 330 ohm.***
- ***Note 3: If VDD always over 4.2V, 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



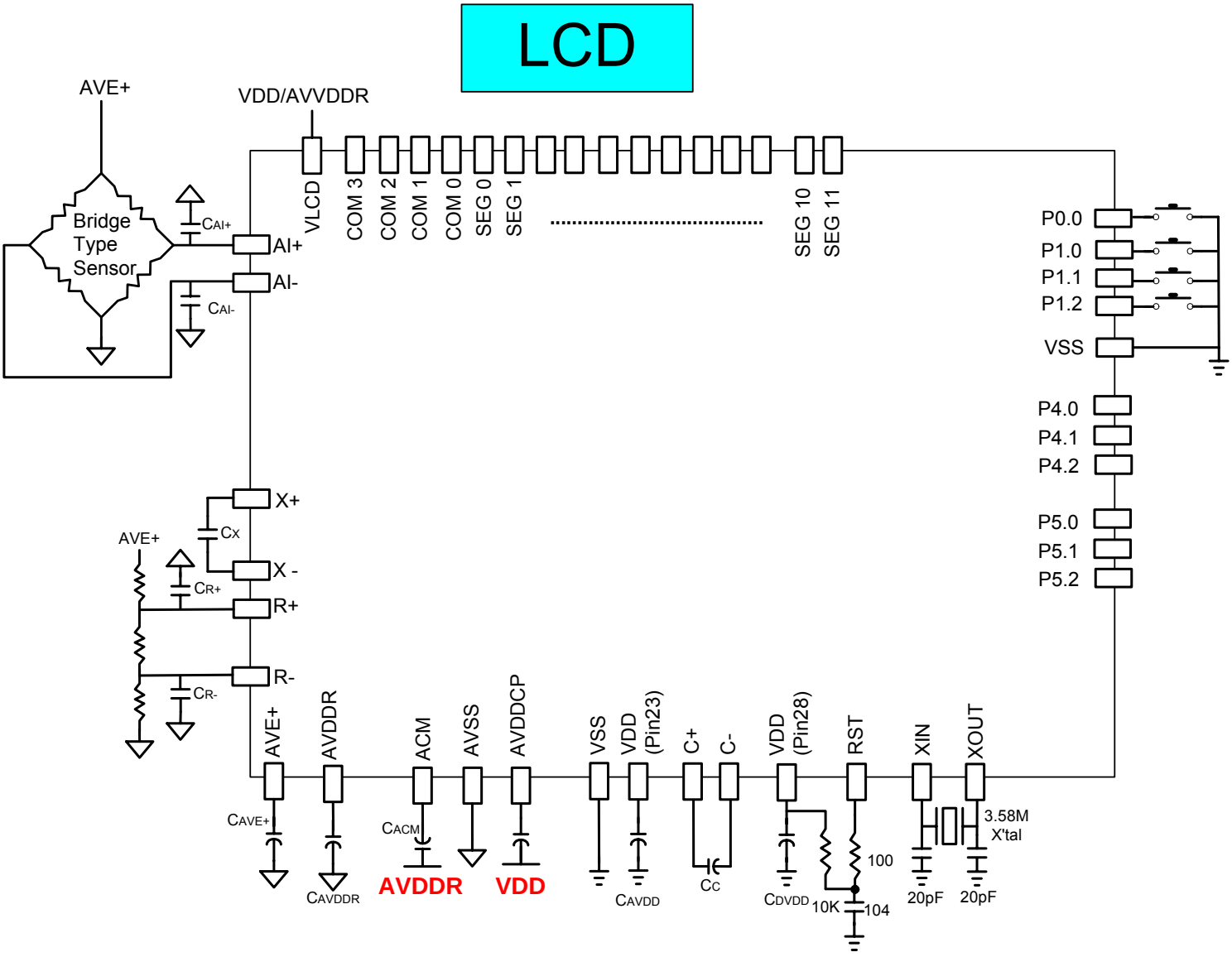
Delay Time:

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.***

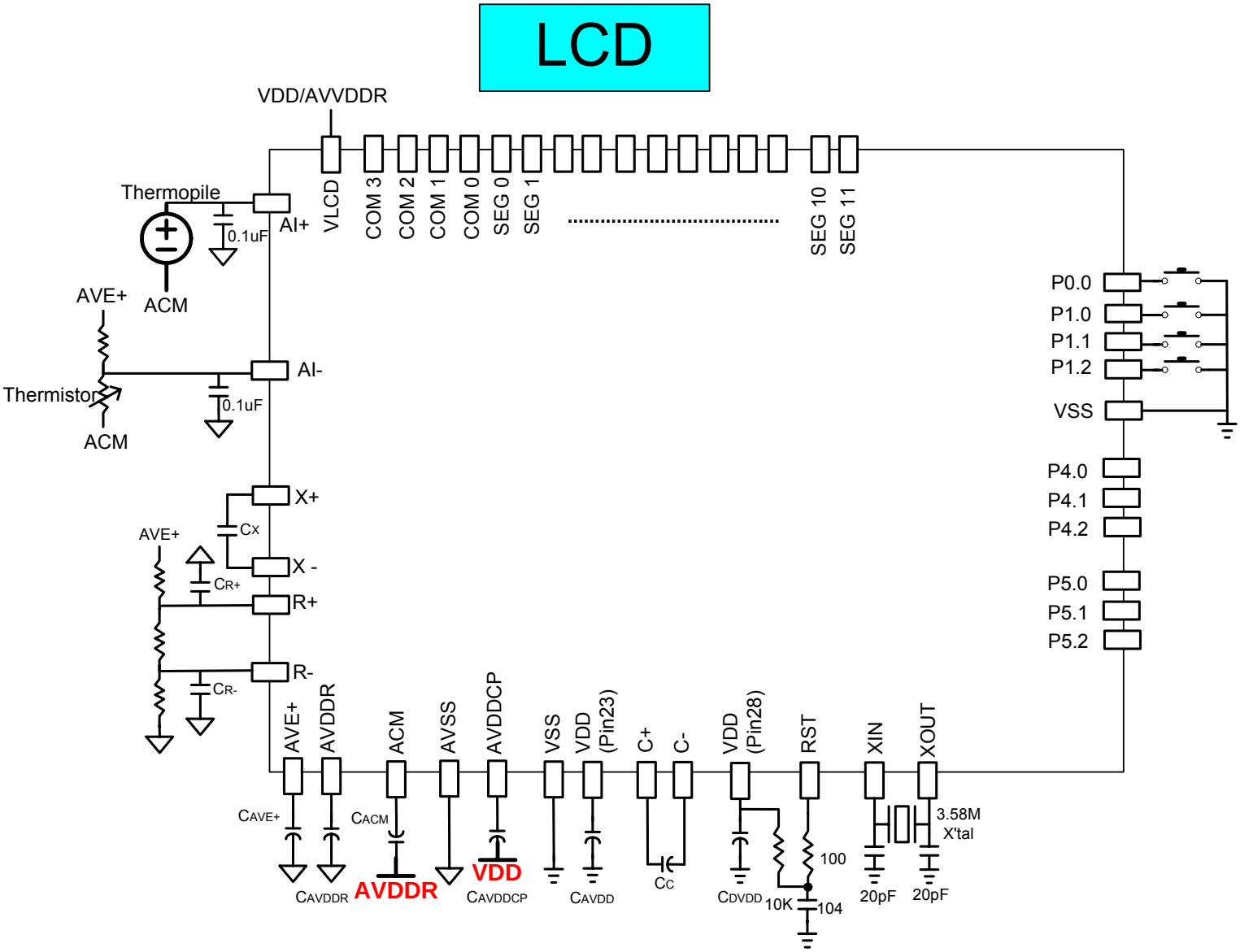
11 APPLICATION CIRCUIT

11.1 Scale (Load Cell) Application Circuit



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

11.2 Thermometer Application Circuit



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

12 INSTRUCTION SET TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
	MOV	A,M	A ← M	-	-	√	1
M	MOV	M,A	M ← A	-	-	-	1
O	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
V	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
E	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I, M = only supports 0x80~0x87, (e.g. R, Y, Z , RBANK ,PFLAG…….)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
A	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
R	ADD	A,M	A ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
I	ADD	M,A	M ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1
H	ADD	A,I	A ← A + I, if occur carry, then C=1, else C=0	√	√	√	1
M	SBC	A,M	A ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
E	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
T	SUB	A,M	A ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
I	SUB	M,A	M ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
C	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
	DAA		To adjust ACC's data format from HEX to DEC.	√	-	-	1
	AND	A,M	A ← A and M	-	-	√	1
L	AND	M,A	M ← A and M	-	-	√	1
O	AND	A,I	A ← A and I	-	-	√	1
G	OR	A,M	A ← A or M	-	-	√	1
I	OR	M,A	M ← A or M	-	-	√	1
C	OR	A,I	A ← A or I	-	-	√	1
	XOR	A,M	A ← A xor M	-	-	√	1
	XOR	M,A	M ← A xor M	-	-	√	1
	XOR	A,I	A ← A xor I	-	-	√	1
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1
R	RRC	M	A ← RRC M	√	-	-	1
O	RRCM	M	M ← RRC M	√	-	-	1
C	RLC	M	A ← RLC M	√	-	-	1
E	RLCM	M	M ← RLC M	√	-	-	1
S	CLR	M	M ← 0	-	-	-	1
S	BCLR	M.b	M.b ← 0	-	-	-	1
	BSET	M.b	M.b ← 1	-	-	-	1
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1
	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
B	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
R	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1 + S
A	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1 + S
N	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1 + S
C	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1 + S
H	BTS0	M.b	If M.b = 0, then skip next instruction	-	-	-	1 + S
	BTS1	M.b	If M.b = 1, then skip next instruction	-	-	-	1 + S
	B0BTS0	M.b	If M (bank 0).b = 0, then skip next instruction	-	-	-	1 + S
	B0BTS1	M.b	If M (bank 0).b = 1, then skip next instruction	-	-	-	1 + S
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
M	RET		PC ← Stack	-	-	-	2
I	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
S	NOP		No operation	-	-	-	1
C							

13 Development Tools

13.1 Development Tool Version

13.1.1 ICE (In circuit emulation)

- **SN8ICE 1K: (S8KD-2)** Full function emulates SN8P1917 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 SN8P1917 EV-KIT to emulation Analog Function.**

✷ **Note: S8ICE2K doesn't support SN8P1917 serial emulation.**

13.1.2 OTP Writer

- **Easy Writer V1.0:** OTP programming is controlled by ICE without firmware upgrade suffers. Please refer easy writer user manual for detailed information.
- **MP-EZ Writer V1.0:** Stand-alone operation to support SN8P1917 mass production

✷ **Note: Writer 3.0 doesn't support SN8P1917 OTP programming.**

13.1.3 IDE (Integrated Development Environment)

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

- **For SN8ICE 1K:** SN8IDE 1.99V or later
- **For Easy Writer and MP-Easy Writer:** SN8IDE 1.99W or later
- **M2IDE V1.0X doesn't support SN8P1917.**

13.2 OTP Programming Pin to Transition Board Mapping

13.2.1 The pin assignment of Easy and MP EZ Writer transition board socket:

Easy Writer JP1/JP2

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

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

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

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

JP3 for MP transition board

13.2.2 The pin assignment of Writer V3.0 transition board socket:

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

Writer V3.0 JP1 Pin Assignment

13.2.3 SN8P1917 Series Programming Pin Mapping:

OTP Programming Pin of SN8P1917 Series			
Chip Name		SN8P1917	
Easy, MP-EZ Writer And Writer V3.0		OTP IC / JP3 Pin Assignment	
Number	Pin	Number	Pin
1	VDD	10,23,28	VDD
2	GND	17,25,40	VSS
3	CLK	30	P1.0
4	CE	-	-
5	PGM	31	P1.1
6	OE	32	P1.2
7	D1	-	-
8	D0	-	-
9	D3	-	-
10	D2	-	-
11	D5	-	-
12	D4	-	-
13	D7	-	-
14	D6	-	-
15	VDD	10,23,28	VDD
16	VPP	41	RST
17	HLS	-	-
18	RST	-	-
19	-	-	-
20	ALSB/PDB	33	P1.3

14 ELECTRICAL CHARACTERISTIC

14.1 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

14.2 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	V_{DD}	Normal mode, V_{PP} = V_{DD}		2.4	5.0	5.5	V
RAM Data Retention voltage	V_{DR}			1.5		-	V
V_{DD} rise rate	V_{POR}	V_{DD} rise rate to ensure power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.9Vdd	-	Vdd	V
Reset pin leakage current	I_{LEKG}	V_{IN} = V_{DD}		-	-	2	uA
I/O port pull-up resistor	R_{UP}	V_{IN} = V_{SS}, V_{DD} = 3V		100	200	300	KΩ
		V_{IN} = V_{SS}, V_{DD} = 5V		50	100	180	KΩ
I/O port input leakage current	I_{LEKG}	Pull-up resistor disable, V_{IN} = V_{DD}		-	-	2	uA
I/O output source current	I_{OH}	V_{OP} = V_{DD} - 0.5V		8	12	-	mA
sink current	I_{OL}	V_{OP} = V_{SS} + 0.5V		8	15	-	mA
INT_N trigger pulse width	$T_{INT}0$	INT0 ~ INT1 interrupt request pulse width		2/F_{CPU}	-	-	Cycle
Supply Current	Idd1	normal Mode (Low Power Disable, Analog Parts OFF)	Vdd= 5V 4Mhz crystal	-	1.5	3	mA
			Vdd= 3V 4Mhz crystal	-	0.5	1	mA
	Idd2	normal Mode (Low Power Enable Analog Parts OFF)	Vdd= 5V 4Mhz crystal	-	0.9	1.8	mA
			Vdd= 3V 4Mhz crystal	-	0.35	0.7	mA
	Idd3	normal Mode (Low Power Disable Analog Parts ON)	Vdd= 5V 4Mhz crystal	-	2.5	5	mA
			Vdd= 3V 4Mhz crystal	-	2.2	4.4	mA
	Idd4	normal Mode (Low Power Enable Analog Parts ON)	Vdd= 5V 4Mhz crystal	-	2	4	mA
			Vdd= 3V 4Mhz crystal	-	1.5	3	mA
	Idd5	normal Mode (Low Power Disable, Analog Parts OFF)	Vdd= 5V IHRC	-	1.7	3.4	mA
			Vdd= 3V IHRC	-	0.9	1.8	mA
	Idd6	normal Mode (Low Power Enable Analog Parts OFF)	Vdd= 5V IHRC	-	1.1	2.2	mA
			Vdd= 3V IHRC	-	0.7	1.4	mA
	Idd7	normal Mode (Low Power Disable Analog Parts ON)	Vdd= 5V IHRC	-	2.5	5	mA
			Vdd= 3V IHRC	-	2.2	4.4	mA

Supply Current	Idd8	normal Mode (Low Power Enable Analog Parts ON)	Vdd= 5V IHRC	-	2.1	4.2	mA
			Vdd= 3V IHRC	-	1.9	3.8	mA
	Idd9	Slow mode (Stop High Clock, LCD OFF, CPR OFF)	Vdd= 5V ILRC 32Khz	-	10	20	uA
			Vdd= 3V ILRC 16Khz	-	3	6	uA
	Idd10	Slow mode (Stop High Clock, LCD ON, CPR OFF)	Vdd= 5V ILRC 32Khz	-	25	50	uA
			Vdd= 3V ILRC 16Khz	-	12	24	uA
	Idd11	Slow mode (Stop High Clock, LCD ON, CPR ON)	Vdd= 5V ILRC 32Khz	-	300	600	uA
			Vdd= 3V ILRC 16Khz	-	250	500	uA
	Idd12	Sleep mode	Vdd= 5V	-	1	2	uA
			Vdd= 3V	-	0.7	1.5	uA
Internal High Clock Freq.	F_{IHRC}	Internal High RC Oscillator Frequency (Fcpu= F_{IHRC}/16)		14	16	18	MHz
LVD detect level	V_{LVD}	Internal POR detect level		1.7	2.0	2.3	V

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

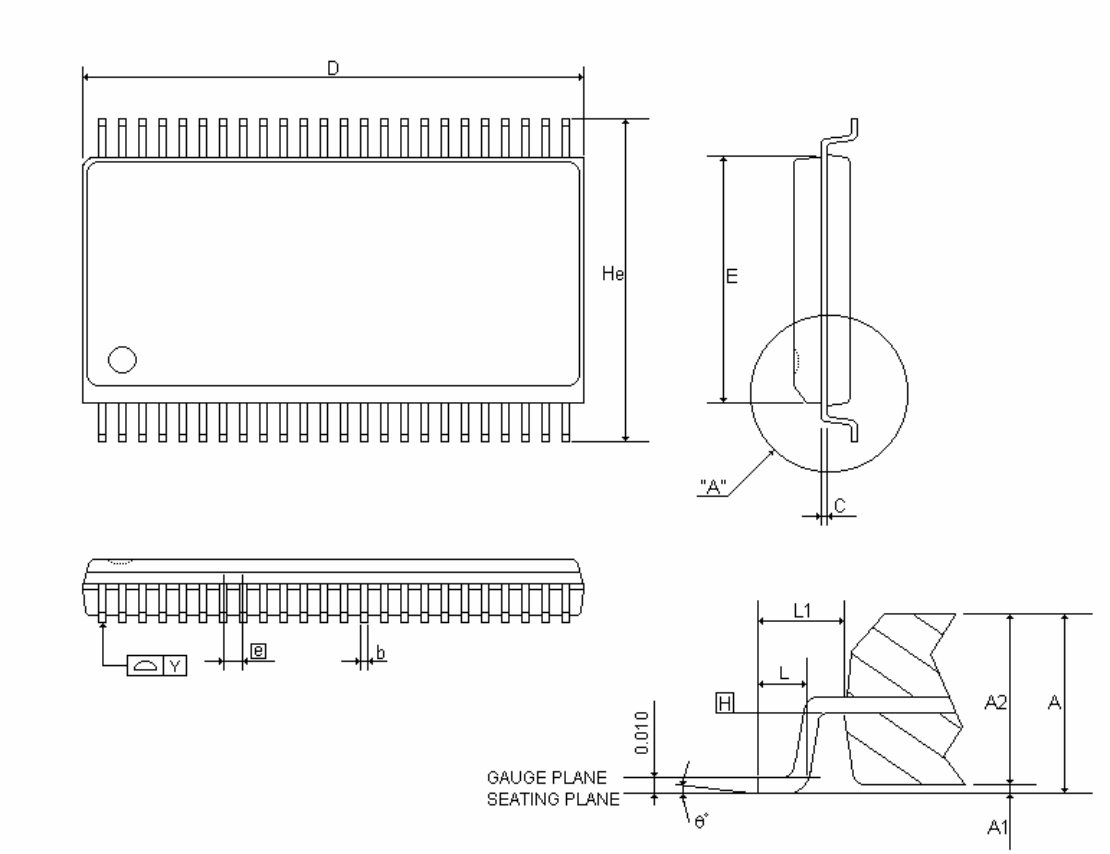
(All of voltages refer to Vdd=3.8V F_{OSC} = 4MHz, ambient temperature is 25°C unless otherwise note.)

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	μA
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	μA
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}		1.160	1.210	1.270	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}		3.5	3.75	4.1	V
Regulator output voltage AVE+	V_{AVE+}	AVE+ set as 3.0V	2.8	3.0	3.3	V
Analog common voltage	V_{ACM}		1.15	1.21	1.27	V
Regulator output current capacity	I_{VA+}		10			mA
Quiescent current	I_{QI}			700	1400	uA
V_{ACM} driving capacity	I_{SRC}		10	-	-	μA
V_{ACM} sinking capacity	I_{SNK}		1	-	-	mA

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

15 PACKAGE INFORMATION

15.1 SSOP 48 PIN



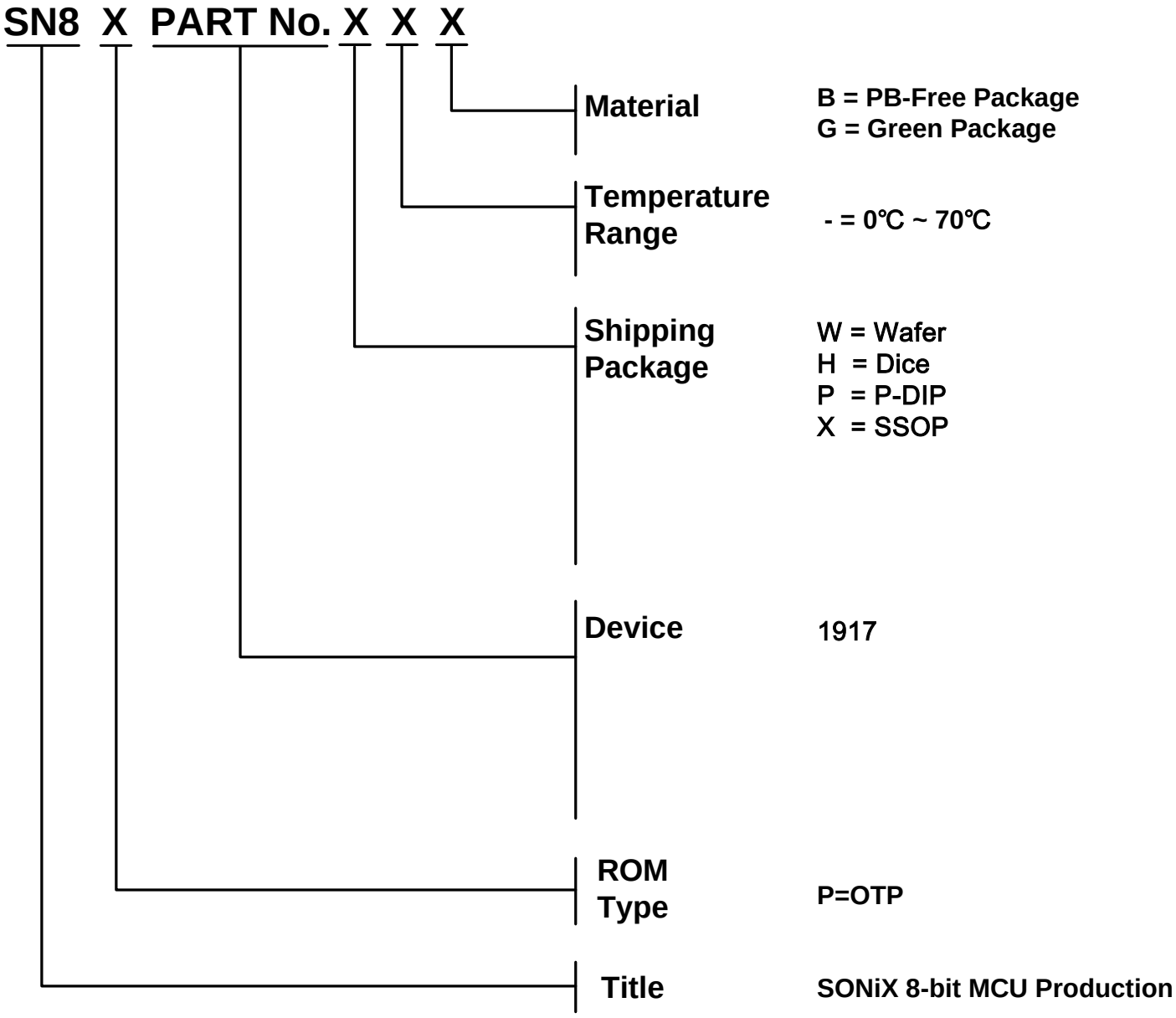
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.095	0.102	0.110	2.413	2.591	2.794
A1	0.008	0.012	0.016	0.203	0.305	0.406
A2	0.089	0.094	0.099	2.261	2.388	2.515
b	0.008	0.010	0.030	0.203	0.254	0.762
C	-	0.008	-	-	0.203	-
D	0.620	0.625	0.630	15.748	15.875	16.002
E	0.291	0.295	0.299	7.391	7.493	7.595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

16 Marking Definition

16.1 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.

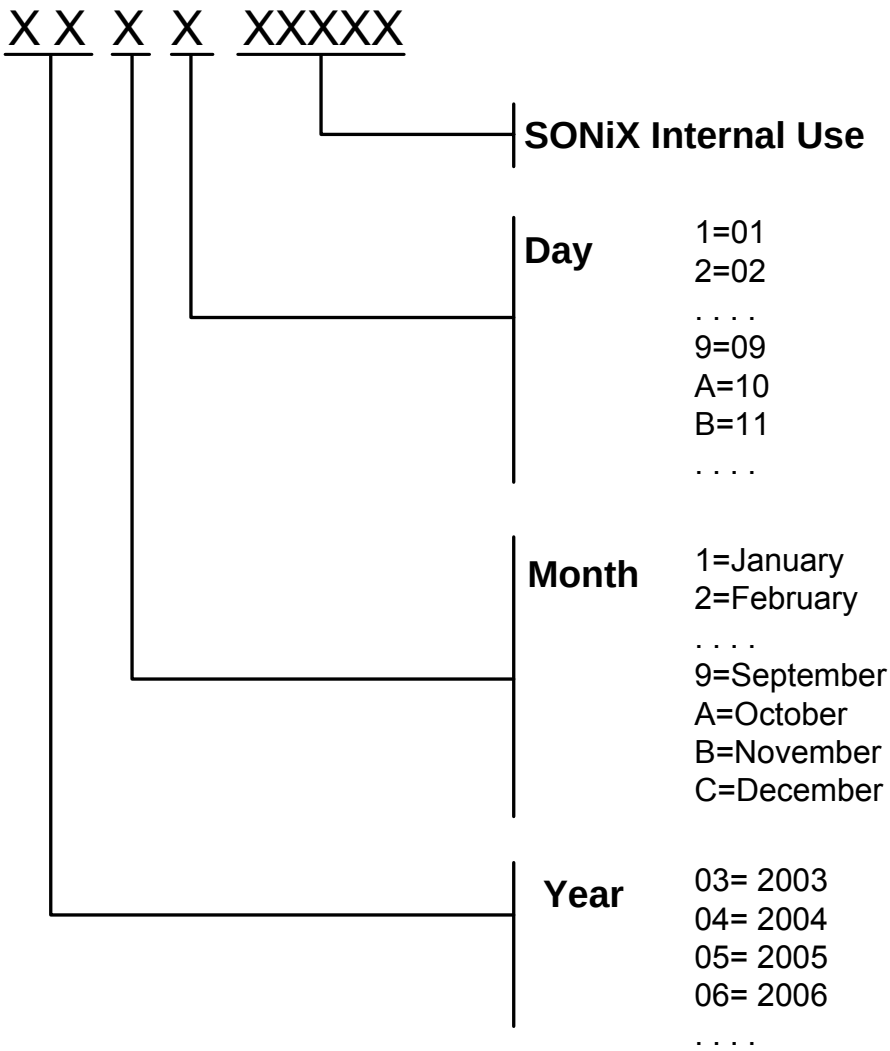
16.2 MARKING INDETIFICATION SYSTEM



16.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P1917PB	OTP	1917	P-DIP	0℃~70℃	PB-Free Package
SN8P1917PG	OTP	1917	P-DIP	0℃~70℃	Green Package

16.4 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