SN8P2977A

USER'S MANUAL

Specification V1.8

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY(2977A)

Version	Date	Description
Ver1.0	2017.10	New Version start. (Refer to 2977 Ver1.7). The 2977A improve ADC structure, it makes the EMS-RS better than before.
Ver1.1	2017.11	1.Add PARAMETER(TACM) OF Band gap Reference in chapter 17.(page 135)
Ver1.2	2017.11	1.Modify ISP voltage 7.5V→6.5V. (page 102) 2.Modify AVMPENB→AMPENB.(page 108) 3.Modify OTP Programming Pin to Transition Board Mapping. (page 129)
Ver1.3	2018.3	1.Modify typical consumption current 1.2mA→1.6mA in normal mode. (page 134) 2.Modify PUSH/POP description in Chapter 15. (page 127) 3.Modify system low clock figure in chapter 4.5. (page 49) 4.Add Internal Low clock frequency characteristic in chapter 17. (page 135) 5.Add RBANK register descriptions in chapter 2.1.4. (page 27)
Ver1.4	2018.8	1.Modify reserved ROM size from FFC to FF8 in complier.(To reserve 4 word use in unique code/rolling code)(page 17)
Ver1.5	2018.11	1.Modify The basic timer table interval time of TC0, TC0X8 = 1 table. (page79)
Ver1.6	2018.12	1.The SSOP48 type have been phased out. 2.Add note_4 in Chapter 13.8(page 124)
Ver1.7	2021.05	1. Add STOP note that LCD must set all off mode before into stop mode.(page 96) 2. Update Sleep demo code that it adds LCD all off mode setting before into STOP mode. (page 120) 3. Update Green demo code that it adds LCD all off mode setting before into Green mode. (page 119)
Ver1.8	2021.12	1. Remove PUSH and POP of instruction.(page 127)

Table of Content

1 PRODUCT OVERVIEW

1.1 SELECTION TABLE

CHIP	ROM	RAM	Stack	LCD	Timer			I/O	ADC	PWM0/ Buzzer0	Buzzer	RTC	Wakeup Pin no.	UART	Package
					T0	TC0	TC1								
SN8P2967	2K*16	192*8	8	4*12	V	V	-	10	20-bit	1	-	1	8	1	DIP48/LQFP48
SN8P2962	2K*16	192*8	8	-	V	V	-	10	20-bit	1	-	1	8	1	SOP18
SN8P2977A	4K*16	256*8	8	4*16	V	V	-	26	24-bit	1	1	1	8	1	DIP48/LQFP48
SN8P2972A	4K*16	256*8	8	-	V	V	-	26	24-bit	1	1	1	8	1	SOP18

Table 1-1 Selection table of SN8Px9x7 serial

1.2 MIGRATION TABLEUART

Item	SN8P2967	SN8P2977A
PGIA Gain setting	1x, 16x, 32x, 64x, 128x (ADC Gain option ~ 2x)	1x, 16x, 32x, 64x, 128x, 200x (ADC Gain option ~ 2x)
PGIA Temperature Drift	Good	Good
AVE+ Voltage	No	2.0V or 1.5V /1V or 0.75V
AVE+ Capacitor	No Capacitor	1uf
ACM Voltage	No	No
ACM Capacitor	No Capacitor	No Capacitor
AVDDR	2.4V / 2.8V /3.2V	2.4V / 2.8V /3.2V
Load cell Power	AVDDR	AVDDR or AVE
ADC Input Channel	1 fully differential Input	2 fully differential Input 4 single-ended Input
ADC Reference External Voltage V(R+, R-)	No	Yes
ADC Reference Internal Voltage	0.36V ~ 1.2V	0.23V ~ 1.6V
ADC output Rate	7.6Hz~5.2kHz	7.6Hz~5.2kHz
ADC Interrupt	Yes	Yes
ADC Run in Green Mode (with wake-up function)	Yes	Yes
Battery Detect Method	By Comparator or By ADC	By Comparator or By ADC
Temperature Sensor	Build In	Build In
Chopper clock frequency	31.25K	31.25K
AVDDR/AVE+ working in slow mode	Yes	Yes
Operating/Slow mode Current Consumption	Less	Less
INT Interrupt Channel	INT0	INT0/INT1
LCD Bias Voltage	1/3 or 1/2 Bias	1/3 or 1/2 Bias
LCD Type	R type / C type	R type / C type
VLCD Voltage	Adjustable (2.6V~3.3V)	Adjustable (2.6V~3.3V)
VLCD Capacitors	1-Cap (CVLCD)	1-Cap (CVLCD)
Timer	T0: base timer / RTC TC0: timer/counter/buzzer/PWM	T0: base timer / RTC TC0: timer/counter/buzzer/PWM
RTC	Yes	Yes
UART	Yes	Yes
OTP Programming Method	Serial Method	Serial Method
In-System Programmer ROM	Yes(Internal 7.5V)	Yes(Internal 6.5V)
Total Capacitors	5-Cap	5-Cap

Table 1-2 SN8P29x7 Migration Table

1.3 FEATURES

◆ ***Memory configuration***
OTP ROM size: 4K * 16 bits
RAM size: 256 * 8 bits
8-levels stack buffer
LCD RAM size: 4*16 bits

◆ ***I/O pin configuration***
Bi-directional: P0, P1, P2, P3
Wakeup: P0
Pull-up resistors: P0, P1
External interrupt: P0
High-Sink Current I/O: P2, P3

◆ ***Powerful instructions***
One 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)
Fcpu : IHRC/4, /8, /16, /32

◆ ***Two 8-Bit Timer***
T0: Basic Timer. Build in 0.5 sec RTC mode
◆ TC0:Auto-reload Timer/Counter/PWM0/Buzzer
◆ ***One Buzzer output***
Buzzer Output P03: 0.98K / 1.96K / 3.9K/ 7.8K
◆ ***Programmable Gain Instrumentation Amplifier***
PGIA Gain option: 1x/4x/8x/16x/32x/64x/128x /200x
◆ **20-bit Delta-Sigma ADC**
ADC Gain selection: 1x, 2x
ADC Offset selection: (-1/4, -2/4, -3/4) * Vref)
ADC Interrupt and Green Mode wakeup function
4-ADC channels configuration:
- Two fully differential channels
- Four single channel

◆ ***Five interrupt sources***
Three internal interrupts: T0, TC0, ADC
Three external interrupts: INT0 / INT1, UART(RX/TX)

◆ ***Single power supply:*** 2.4V ~ 3.6V (Analog Part)
2.0V ~ 3.6V (Digital Part)

◆ ***On-chip watchdog timer***
On-chip Regulator (AVDDR) with 2.4V voltage output and maximum 10mA driven current (Option:2.4V,2.8V,3.2V)

◆ ***On-chip Regulator AVE with selectable 2V/1.5V/0.75/1V***
◆ ***On-chip 1.2V Band gap reference for battery monitor***
◆ ***Internal LBT 2.2V~3.6V; or external P10 input LBT***
◆ ***Build in ADC reference voltage = 0.23V~1.6V***
◆ ***In-system Programmer ROM with Int. 6.5V generation***
◆ ***One Temperature Sensor***
◆ ***LCD driver:***
1/3 or 1/2 bias voltage.
4 common * 16 segment
Both R type and C type LCD
1C-type LCD Voltage: 2.6 ~3.3V
R-type Mode
◆ ***Seven-segment display driver:***
P2/P3 are High Current Sink I/O and applied for LED display

◆ ***Dual clock system offers four operating modes***
Internal high clock: RC type up to 8 MHz
Internal Low clock: RC type up to 32kHz
Normal mode: Both high and low clock active
Slow mode: Low clock active and 32768 X'tal
Green mode: Low clock active and optional high clock
Wakeup by P0/T0/TC0/ADC
Sleep mode: Both high and low clock stop

◆ ***Package***
Dice/LQFP48/DIP48/QFN32
TSSOP28/SSOP20/SOP18/SOP16

1.4 SYSTEM BLOCK DIAGRAM

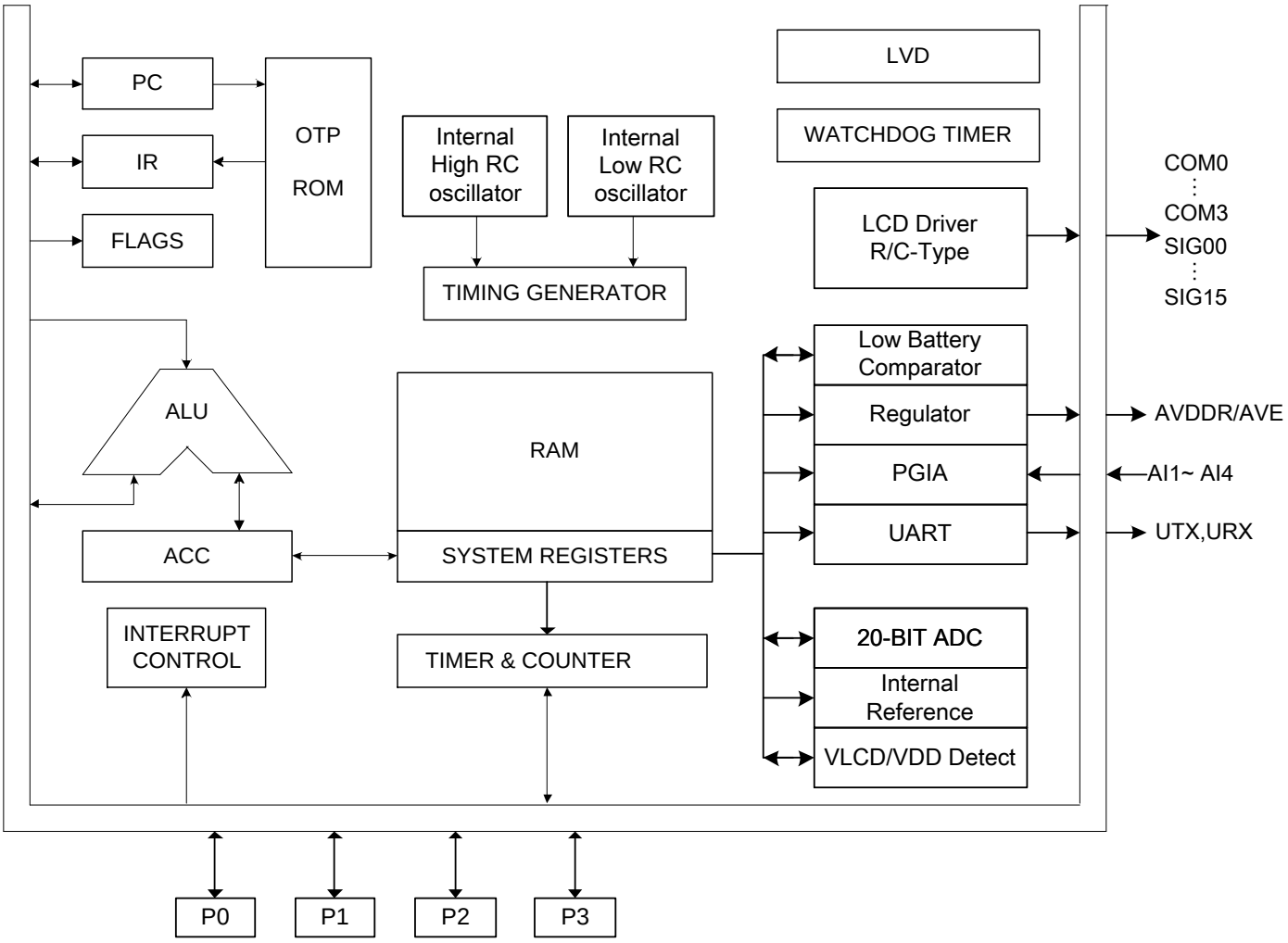


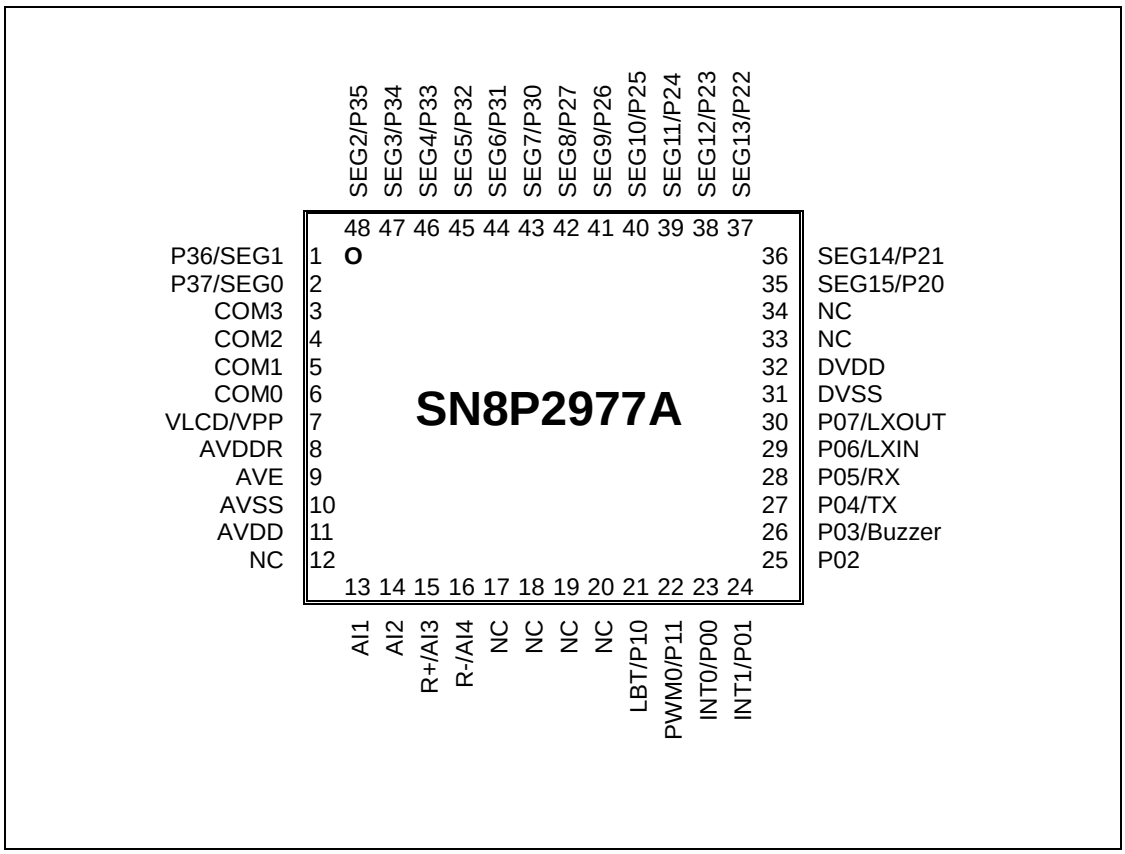
Figure 1-1 Simplified system block diagram

1.5 PIN ASSIGNMENT

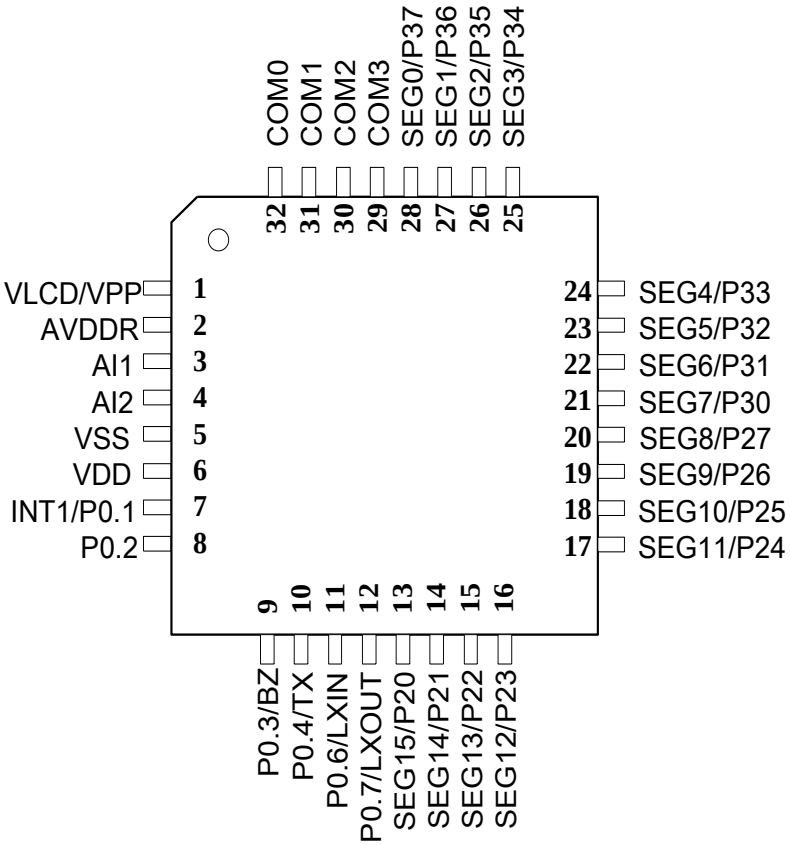
SN8P2977A DIP48

SEG6/P31	1	48	P30/SEG7
SEG5/P32	2	47	P27/SEG8
SEG4/P33	3	46	P26/SEG9
SEG3/P34	4	45	P25/SEG10
SEG2/P35	5	44	P24/SEG11
SEG1/P36	6	43	P23/SEG12
SEG0/P37	7	42	P22/SEG13
COM3	8	41	P21/SEG14
COM2	9	40	P20/SEG15
COM1	10	39	NC
COM0	11	38	NC
VLCD/VPP	12	37	DVDD
NC	13	36	DVSS
NC	14	35	P07/LXOUT
AVDDR	15	34	P06/LXIN
AVE	16	33	P05/RX
AVSS	17	32	P04/PDB/TX
AVDD	18	31	P03/SHIFTDATA/Buzzer
AI1	19	30	P02/OTPCLK
AI2	20	29	P01/INT1/PGCLK
R+/AI3	21	28	P00/INT0
R-/AI4	22	27	NC
NC	23	26	P11/PWM0
NC	24	25	P10/LBT

SN8P2977A LQFP48



SN8P2975A QFN32



SN8P2974A TSSOP28

COM2	1	28	COM3
COM1	2	27	P37/SEG0
COM0	3	26	P36/SEG1
VLCD/VPP	4	25	P35/SEG2
AVDDR	5	24	P34/SEG3
AI1	6	23	P33/SEG4
AI2	7	22	P32/SEG5
VDD	8	21	P31/SEG6
VSS	9	20	P30/SEG7
INT1/P01	10	19	P27/SEG8
P02	11	18	P26/SEG9
Buzzer/P03	12	17	P25/SEG10
TX/P04	13	16	P24/SEG11
P22/SEG13	14	15	P23/SEG12

SN8P2973A SSOP20

VLCD/VPP	1	20	P27/SEG8
AVDDR	2	19	P26/SEG9
AVDD	3	18	P25/SEG10
AI1	4	17	P24/SEG11
AI2	5	16	P23/SEG12
INT1/P01	6	15	P22/SEG13
P02	7	14	P21/SEG14
Buzzer/P03	8	13	P20/SEG15
TX/P04	9	12	DVDD
RX/P05	10	11	DVSS

SN8P2972A SOP18

VLCD/VPP	1	18	DVDD
AVDDR	2	17	DVSS
AVSS	3	16	P07/LXOUT
AVDD	4	15	P06/LXIN
AI1	5	14	P05/RX
AI2	6	13	P04/TX
P10/LBT	7	12	P03/Buzzer
P11/PWM0	8	11	P02
P00/INT0	9	10	P01/INT1

SN8P2971A SOP16

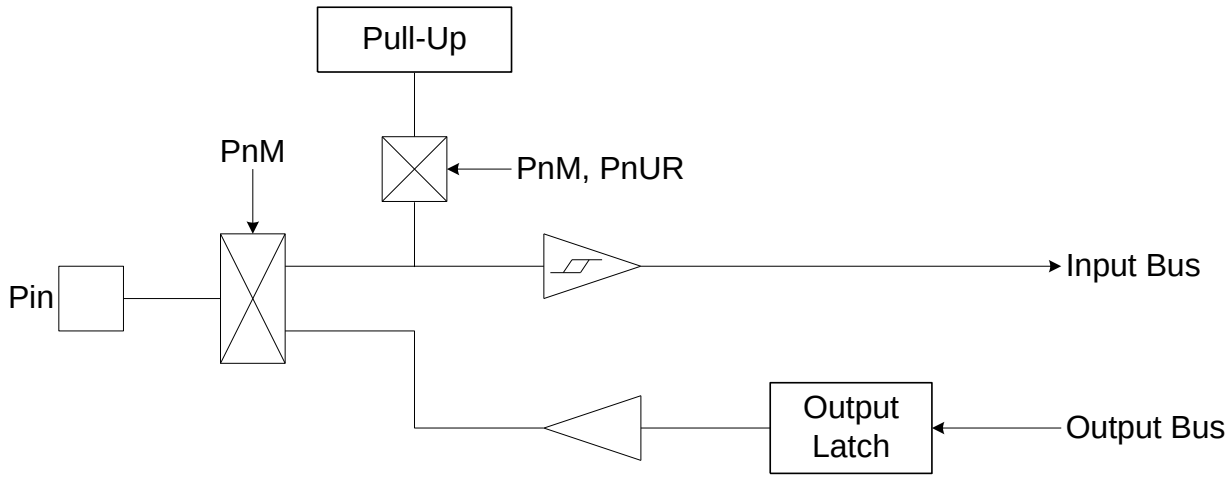
VLCD/VPP	1	16	DVDD
AVDDR	2	15	DVSS
AI1	3	14	P07/LXOUT
AI2	4	13	P06/LXIN
P1.0/LBT	5	12	P05/RX
P1.1/PWM0	6	11	P04/TX
P0.0/INT0	7	10	P03/Buzzer
P0.1/INT1	8	9	P02

1.6 PIN DESCRIPTIONS

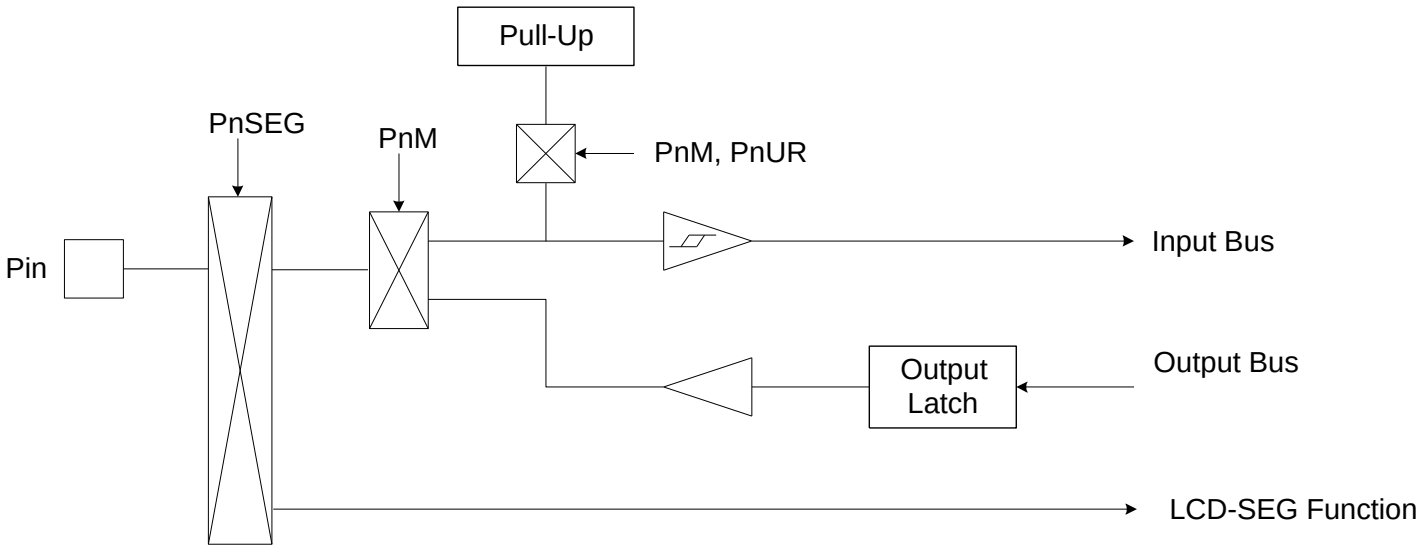
PIN NAME	TYPE	DESCRIPTION
DVDD, AVDD DVSS, AVSS	P	Power supply input pins for digital / Analog circuit
AVDDR	P	Regulator power output pin, Voltage = 2.4V, 2.8V, 3.2V
AVE	P	Regulator power output pin, Voltage = 0.75V/1V(sink only), 1.5V/2.0V
AI1 ~ AI4	P	Analog input channel of PGIA. (AI3/AI4 share with R+/R- function)
VLCD/VPP	P	VPP: OTP ROM programming pin only. (No reset function.) VLCD: LCD driver power pin. (When R-Type LCD mode, VLCD = VDD auto connection)
COM [3:0]	O	COM0~COM3 LCD driver common port
SEG0 ~ SEG15	O	LCD driver segment pins
P37/SEG0	I/O	P37 IO function share with LCD SEG0.
P36/SEG1	I/O	P36 IO function share with LCD SEG1.
P35/SEG2	I/O	P35 IO function share with LCD SEG2.
P34/SEG3	I/O	P34 IO function share with LCD SEG3.
P33/SEG4	I/O	P33 IO function share with LCD SEG4.
P32/SEG5	I/O	P32 IO function share with LCD SEG5.
P31/SEG6	I/O	P31 IO function share with LCD SEG6.
P30/SEG7	I/O	P30 IO function share with LCD SEG7.
P27/SEG8	I/O	P27 IO function share with LCD SEG8.
P26/SEG9	I/O	P26 IO function share with LCD SEG9.
P25/SEG10	I/O	P25 IO function share with LCD SEG10.
P24/SEG11	I/O	P24 IO function share with LCD SEG11.
P23/SEG12	I/O	P23 IO function share with LCD SEG12.
P22/SEG13	I/O	P22 IO function share with LCD SEG13.
P21/SEG14	I/O	P21 IO function share with LCD SEG14.
P20/SEG15	I/O	P20 IO function share with LCD SEG15.
P0 [7:0]	I/O	P00~P07 bi-direction pins / wakeup pins/ Built-in pull-up resisters
P00	I/O	IO share with INT0
P01	I/O	IO share with INT1
P03	I/O	IO share with Buzzer function
P04	I/O	IO share with UART-TX
P05	I/O	IO share with UART-RX
P06	I/O	IO share with LXIN 32768 Oscillator.
P07	I/O	IO share with LXOUT 32768 Oscillator.
P1 [1:0]	I/O	P10~P11 bi-direction pins / Built-in pull-up resisters (optional)
P10	I/O	I/O shire with LBT function (Low battery detect, comparator input)
P11	I/O	I/O shire with PWM0/TC0OUT.

1.7 PIN CIRCUIT DIAGRAMS

Port 0, Port 1 structure:



Port 2, Port 3 structure:



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

4K words ROM

	ROM	
0000H	***Reset vector***	User reset vector
0001H	***General purpose area***	Jump to user start address
0002H		Jump to user start address
0003H		Jump to user start address
0004H	***Reserved***	
0005H		
0006H		
0007H		
0008H	***Interrupt vector***	User interrupt vector
0009H	***General purpose area***	User program
.		
.		
000FH		
0010H		
0011H		
.		
.		
FF7H		End of user program
FF8H	***Reserved***	
.		
FFFH		

2.1.2 RESET VECTOR (0000H)

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

Power On Reset
Watchdog Reset

After power on 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
```

2.1.2.1 INTERRUPT VECTOR (0008H)

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

Note: 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:

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

2.1.2.1 LOOK-UP TABLE DESCRIPTION

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

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

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

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

Example: INC_YZ macro.

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

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

Example: Modify above example by "INC_YZ" macro.

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

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

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

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

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

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

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

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

2.1.2.2 JUMP TABLE DESCRIPTION

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

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

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

Example: Jump table.

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

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

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

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

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

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

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

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

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

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

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

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

Example: "@JMP_A" operation.

; Before compiling program.

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

; After compiling program.

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

2.1.2.3 CHECKSUM CALCULATION

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

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

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

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

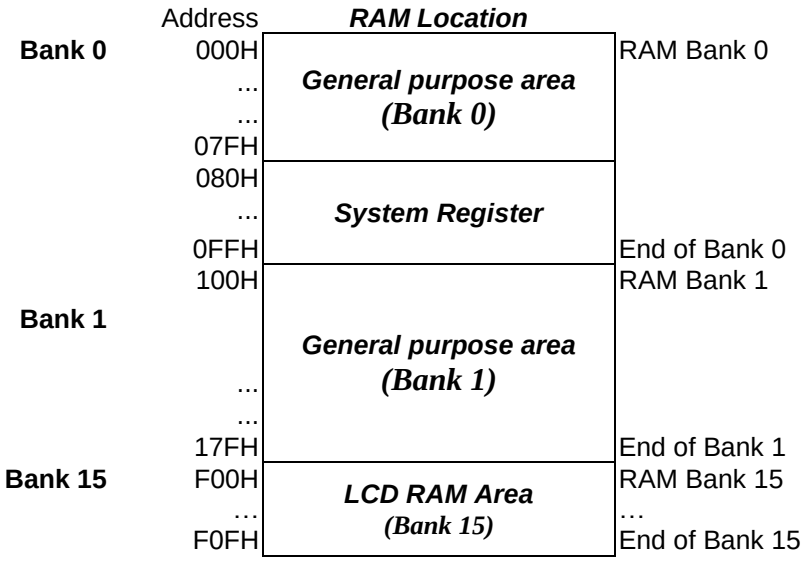
2.1.3 CODE OPTION TABLE

Code Option	Content	Function Description
Watch_Dog	Enable	Enable Watchdog function (WDT work in Normal/slow Mode).
	Disable	Disable Watchdog function.
	Always On	Enable Watchdog function (WDT work in Normal/slow/Green/Sleep Mode).
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
High_Clk_Div	Fhosc/4	High clock Fcpu = IHRC/4 = 2MHz.
	Fhosc/8	High clock Fcpu = IHRC/8 = 1MHz.
	Fhosc/16	High clock Fcpu = IHRC/16 = 0.5MHz.
	Fhosc/32	High clock Fcpu = IHRC/32 = 0.25MHz.
Low_Power	Disable	Disable low power mode
	Enable	Enable low power mode

Note1: In high noisy environment, set Watch_Dog as "Always_On" is strongly recommended.
Note2: Fcpu code option is only available for High Clock. Fcpu of slow mode is Flosc/4.

2.1.4 DATA MEMORY (RAM)

☞ ***256 X 8-bit RAM***



The 256-byte general purpose RAM is separated into Bank0, Bank1 and LCD Ram Bank15. Accessing the three banks' RAM is controlled by "RBANK" register. When RBANK = 0, the program controls Bank 0 RAM directly. When RBANK = 1, the program controls Bank 1 RAM directly. Under one bank condition and need to access the other bank RAM, setup the RBANK register is necessary. When interrupt occurs, RBANK register is saved by user, RAM bank must be reserved last state. User can select RAM bank through setup RBANK register during processing interrupt service routine. When RETI is executed to leave interrupt operation, RBANK register is reloaded by user, and RAM bank returns to last data. Sonix provides "Bank 0" type instructions (e.g. b0mov, b0add, b0bts1, b0bset…) to control Bank 0 RAM in non-zero RAM bank condition directly.

087H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RBANK	-	-	-	-	-	-	RBNKS1	RBNKS0
Read/Write	-	-	-	-	-	-	R/W	R/W
After Reset	-	-	-	-	-	-	0	0

Bit 0 RBNKS[1:0]: Bank selection Bit.

00 = Bank0 selected.

01 = Bank1 selected.

1x = Bank15 selected. (LCD RAM)

2.1.5 SYSTEM REGISTER

2.1.5.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	-	PFLAG	RBANK	-	LCDM1	LCDM2	P2SEG	P3SEG	BZRM	-	-
9	VREG	CHS	AMPM	ADCM1	ADCM2	LBTM	ADCDH	ADCDM	ADCDL	-	-	-	-	-	-	-
A	ROMADRH	ROMADRL	ROMDAH	ROMDAL	-	-	-	-	-	-	-	-	-	-	-	
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	-	P1M	P2M	P3M	-	-	-	-	INTRQ	INTEN	OSCM	-	WDTR	-	PCL	PCH
D	P0	P1	P2	P3	-	-	-	-	T0M	T0C	TC0M	TC0C	TC0R	P2UR	P3UR	STKP
E	P0UR	P1UR	URTX	URRX	URCR	UTXD	URXD	@YZ	@HL	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.1.5.2 SYSTEM REGISTER DESCRIPTION

L , H = Working register, @LH and ROM addressing register
Y, Z = Working register, @YZ and ROM addressing register
LCDM1 = LCD mode register
P2SEG = Port 2 function control register
BZRM = Buzzer function control register
CHS = PGIA input channel control register
ADCM1 = ADC control register1
LBTM = Low Battery Detect Register
ADCDM = ADC medium-byte data buffer
ROMADRH/L= ISP ROM Address
P_NM = Port N input/output mode register
P_NUR = Port N pull-up register
INTRQ = Interrupt request register
OSCM = Oscillator mode register
PCH, PCL = Program counter
T0C = Timer 0 counting register
TC0C = Timer/Counter 0 Counting register
STKP = Stack pointer buffer
URRX = UART received control register
UTXD = UART transmitted data buffer.
@YZ = RAM YZ indirect addressing index pointer
STK0~STK7= Stack 0 ~ stack 7 Buffer

R= Working register and ROM look-up data buffer
PFLAG = ROM page and special flag register
RBANK = Bank selection register
LCDM2 = LCD mode register
P3SEG = Port 3 function control register
VREG = Voltage Regulators control register
AMPM = PGIA mode selection register
ADCM2 = ADC control register2
ADCDH = ADC high-byte data buffer
ADCDL = ADC low-byte data buffer
ROMDARH/L= ISP Program Data(H/L)
PEDGE = INT0/INT1 edge selection register
P_N = Port N data buffer
INTEN = Interrupt enable register
WDTR = Watchdog timer register
T0M = Timer 0 mode register
TC0M = Timer/Counter 0 mode register
TC0R = Timer/Counter 0 auto-reload data buffer
URTX = UART transmitter control register
URCR = UART baud rate control register
URXD = UART received data buffer.
@HL = RAM HL indirect addressing index pointer

2.1.5.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Name
80H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
81H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	-	-	-	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	-	RBNKS1	RBNKS0	R/W	RBANK
089H	LCDBNK	-	LCDMOD1	LCDMOD0	LCDENB	LCDBIAS	LCDRATE	LCDCLK	R/W	LCDM1
08AH	VAR1	VAR0	DISQ	-	VPPINTL	VCP2	VCP1	VCP0	R/W	LCDM2
08BH	P27SEG	P26SEG	P25SEG	P24SEG	P23SEG	P22SEG	P21SEG	P20SEG	R/W	P2SEG
08CH	P37SEG	P36SEG	P35SEG	P34SEG	P33SEG	P32SEG	P31SEG	P30SEG	R/W	P3SEG
08DH						BZRENB	BZRCKS1	BZRCKS0	W	BZRM
090H	BGRENB	-	AVENB	AVESEL1	AVESEL0	AVDDRENB	AVDDRSEL1	AVDDRSEL0	R/W	VREG
091H	-	MUXP2	MUXP1	MUXP0	-	MUXN2	MUXN1	MUXN0	R/W	CHS
092H	AMPCKS2	AMPCKS1	AMPCKS0	PCHPENB	GS2	GS1	GS0	AMPENB	R/W	AMPM
093H	IRVS3	IRVS2	IRVS1	IRVS0	ACHPENB	-	ADGN	ADCENB	R/W	ADCM1
094H	ADCKS	OSR2	OSR1	OSR0	-	OFSEL1	OFSEL0	DRDY	R/W	ADCM2
095H	-	P11IO	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTO	LBTENB	R/W	LBTM
096H	ADCB23	ADCB22	ADCB21	ADCB20	ADCB19	ADCB18	ADCB17	ADCB16	R	ADCDH
097H	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB9	ADCB8	R	ADCDM
098H	ADCB7	ADCB6	ADCB5	ADCB4	ADCB3	ADCB2	ADCB1	ADCB0	R	ADCDL
0A0H	ROMADR15				ROMADR11	ROMADR10	ROMADR9	ROMADR8	R/W	ROMADRH
0A1H	ROMADR7	ROMADR6	ROMADR5	ROMADR4	ROMADR3	ROMADR2	ROMADR1	ROMADR0	R/W	ROMADRL
0A2H	ROMDA15	ROMDA14	ROMDA13	ROMDA12	ROMDA11	ROMDA10	ROMDA9	ROMDA8	W	ROMDAH
0A3H	ROMDA7	ROMDA6	ROMDA5	ROMDA4	ROMDA3	ROMDA2	ROMDA1	ROMDA0	W	ROMDAL
0B8H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0BFH	-	P01G1	P01G0	P00G1	P00G0	-	-	-	R/W	PEDGE
0C1H	-	-	-	-	-	-	P11M	P10M	R/W	P1M
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M
0C3H	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M	R/W	P3M
0C8H	ADCIRQ	-	TC0IRQ	T0IRQ	URXIRQ	UTXIRQ	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	-	TC0IEN	T0IEN	URXIEN	UTXIEN	P01IEN	P00IEN	R/W	INTEN
0CAH	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	-	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	-	-	-	-	-	-	P11	P10	R/W	P1
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2
0D3H	P37	P36	P35	P34	P33	P32	P31	P30	R/W	P3
0D8H	T0ENB	T0RATE2	T0RATE1	T0RATE0	-	TC0x8	TC0GN	T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0RATE2	TC0RATE1	TC0RATE0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0DDH	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2UR
0DEH	P37R	P36R	P35R	P34R	P33R	P32R	P31R	P30R	W	P3UR
0DFH	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R	W	P0UR
0E1H	-	-	-	-	-	-	P11R	P10R	W	P1UR
0E2H	UTXEN	UTXPEN	UTXPS	UTXBRK	URXBZ	UTXBZ	-	-	R/W	URTX
0E3H	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0	R/W	URRX
0E4H	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0	R/W	URCR
0E5H	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0	R/W	UTXD
0E6H	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0	R/W	URXD
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ

0E8H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	-	-	-		S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	-	-	-		S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	-	-	-		S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	-	-	-		S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	-	-	-		S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	-	-	-		S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	-	-	-		S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	-	-	-		S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✷ ***Note:***

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

2.1.6 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.***

2.1.7 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.***

2.1.8 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 0x01 to 0x00, the PC will add 2 steps to skip next instruction.

DECS instruction:

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

DECMS instruction:

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

2.1.9 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
                …
                …
```

2.1.10 Y, Z REGISTERS

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

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

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

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

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

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

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

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

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

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

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

2.1.11 H, L REGISTERS

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

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

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

080H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
L	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

➢ **Example: Uses H, L register as the data pointer to access data in the RAM address 020H of bank0.**

```
        B0MOV   H, #00H     ; To set RAM bank 0 for H register
        B0MOV   L, #20H     ; To set location 20H for L register
        B0MOV   A, @HL      ; To read a data into ACC
```

➢ **Example: Uses the H, L register as data pointer to clear the RAM data.**

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

CLR_HL_BUF:
        CLR     @HL         ; Clear @HL to be zero

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

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

2.1.12 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	0	0	0	0	0	0	0	0

✷ ***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 @HL register.**

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

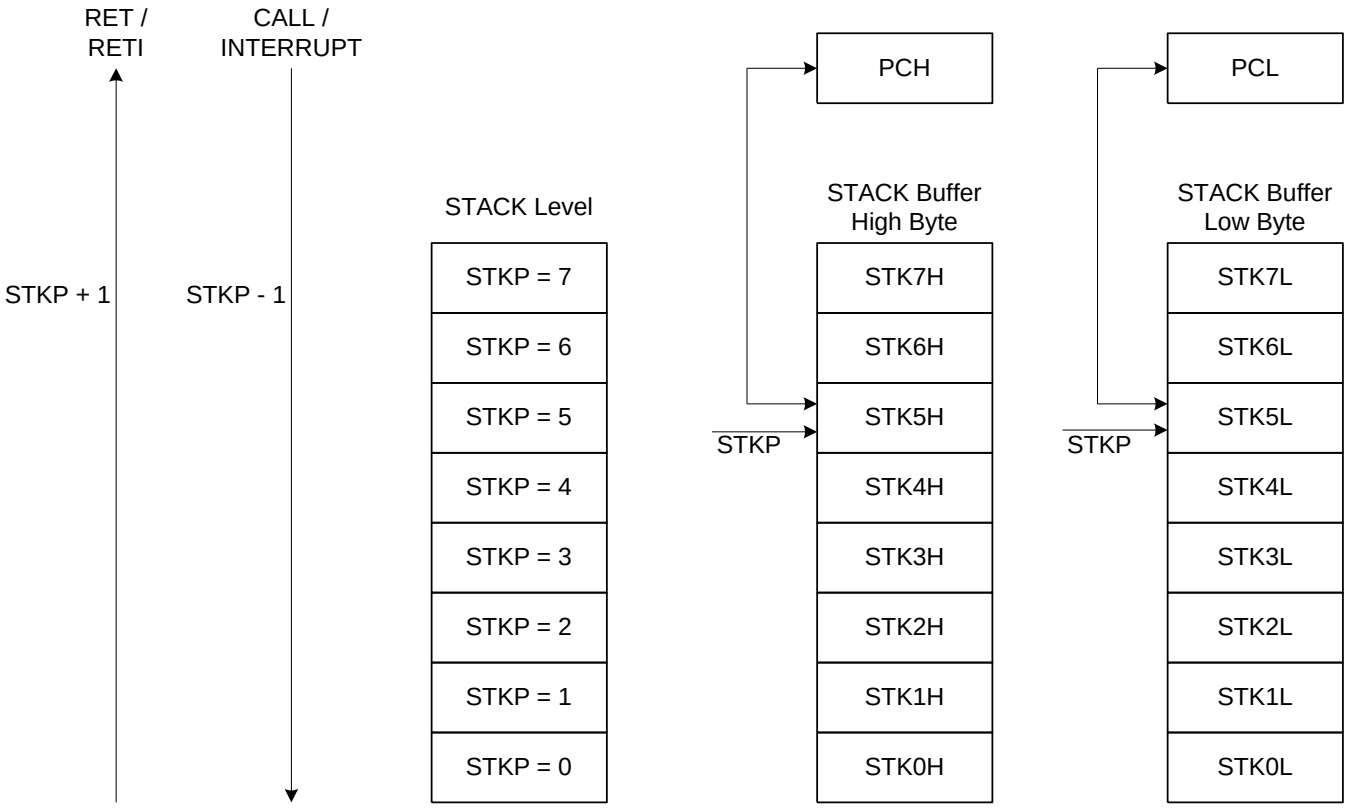
- **Example: Indirectly addressing mode with @YZ register.**

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

2.3 STACK OPERATION

2.3.1 OVERVIEW

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



2.3.2 STACK REGISTERS

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

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

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

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

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

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

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

```
MOV     A, #01111111B
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

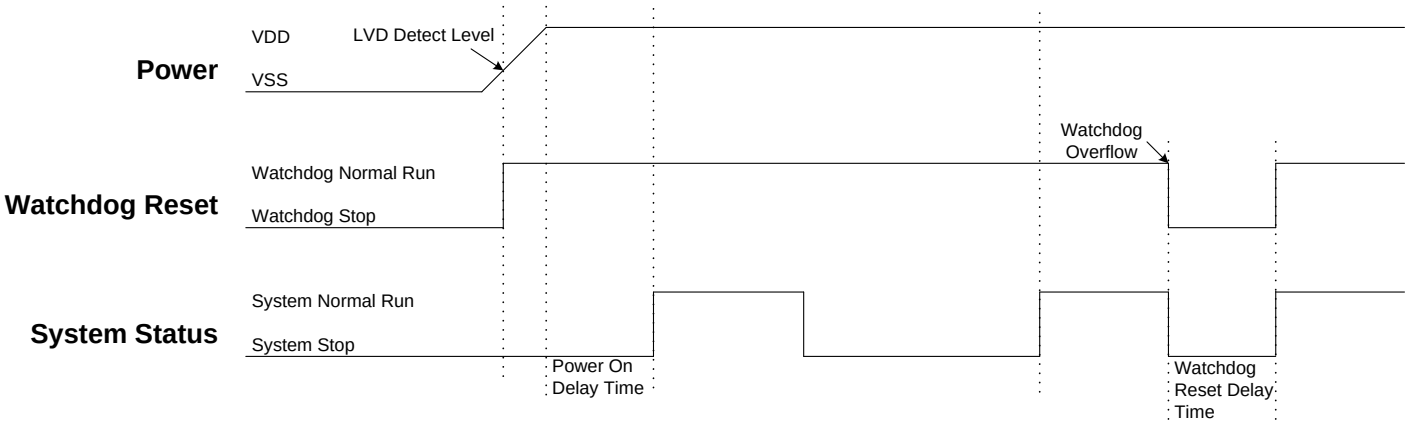
3.1 OVERVIEW

The system would be reset in three conditions as following.

- Power on reset
- Watchdog reset
- Brown out reset

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.
- **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.

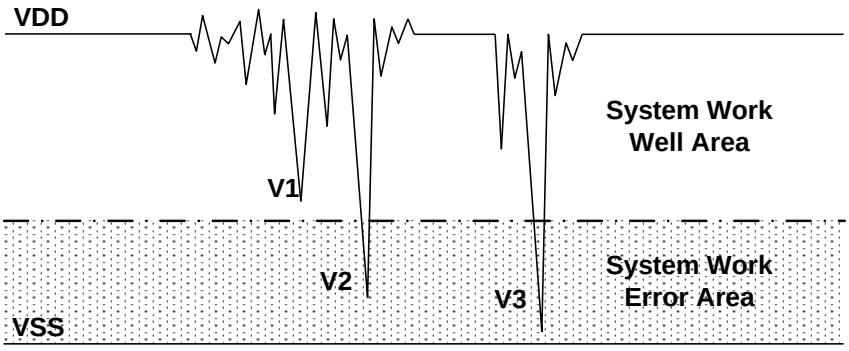
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

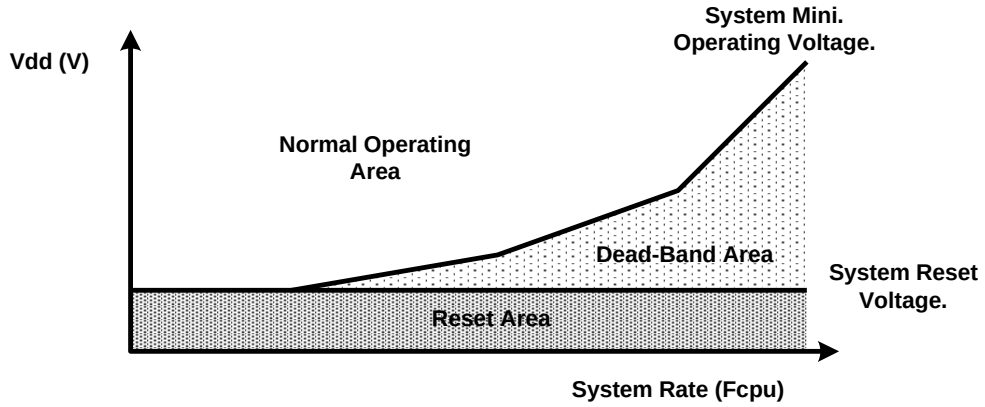
AC application:

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.



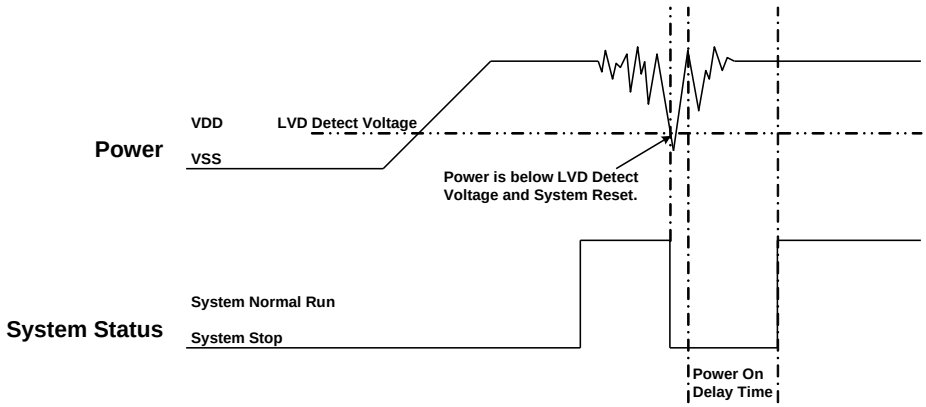
Normally the system operation voltage area is higher than the system reset voltage to VDD, and the reset voltage is decided by LVD detect level. The system minimum operating voltage rises when the system executing rate upper even higher than system reset voltage. The dead-band definition is the system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

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.

4 SYSTEM CLOCK

4.1 OVERVIEW

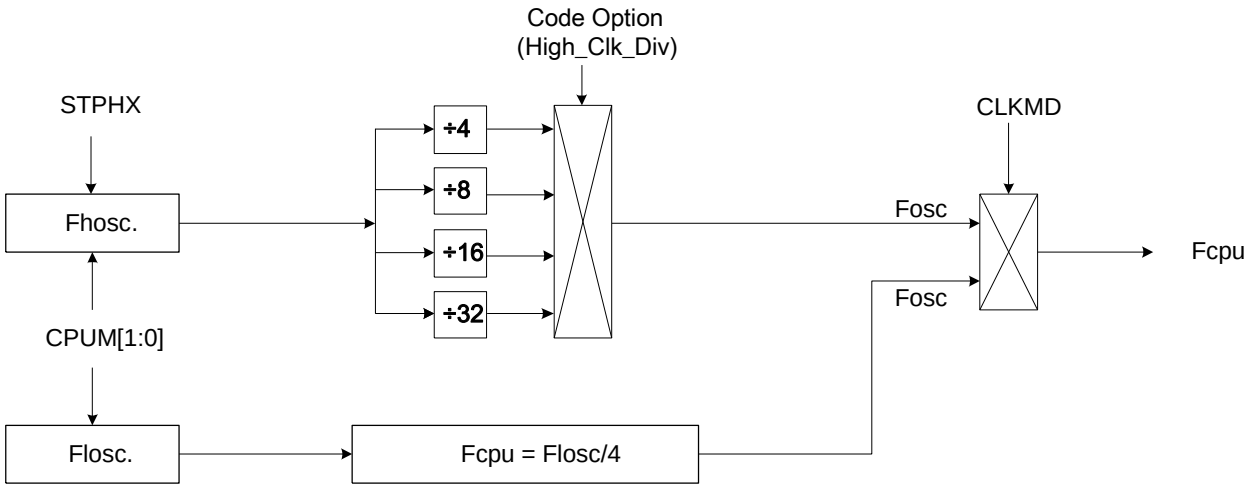
The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is only generated from internal 8MHz high-speed RC oscillator circuit (IHRC 8MHz). The low-speed clock is generated from internal RC oscillator circuit or external 32768Hz Crystal.

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 = 2MHz**. (Fhosc= 8MHz IHRC)
Fcpu = Fhosc / 8 = 1MHz. (Fhosc= 8MHz IHRC)
Fcpu = Fhosc / 16 = 500 kHz. (Fhosc= 8MHz IHRC)
Fcpu = Fhosc / 32 = 250 kHz. (Fhosc= 8MHz IHRC)

☞ **Slow Mode (Low Clock):** **Fcpu = Flosc/4.** (Flosc= ILRC or 32768Hz)

4.2 CLOCK BLOCK DIAGRAM



- Fhosc: System high clock source is from internal high RC (IHRC).
- Flosc: System low clock source is from internal low RC (ILRC) or external 32k.
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.3 OSCM REGISTER

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

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

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

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

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

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

```
B0BSET      FSTPHX          ; To stop IHRC 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 IHRC and internal low-speed
                            ; oscillator called power down mode (sleep mode).
```

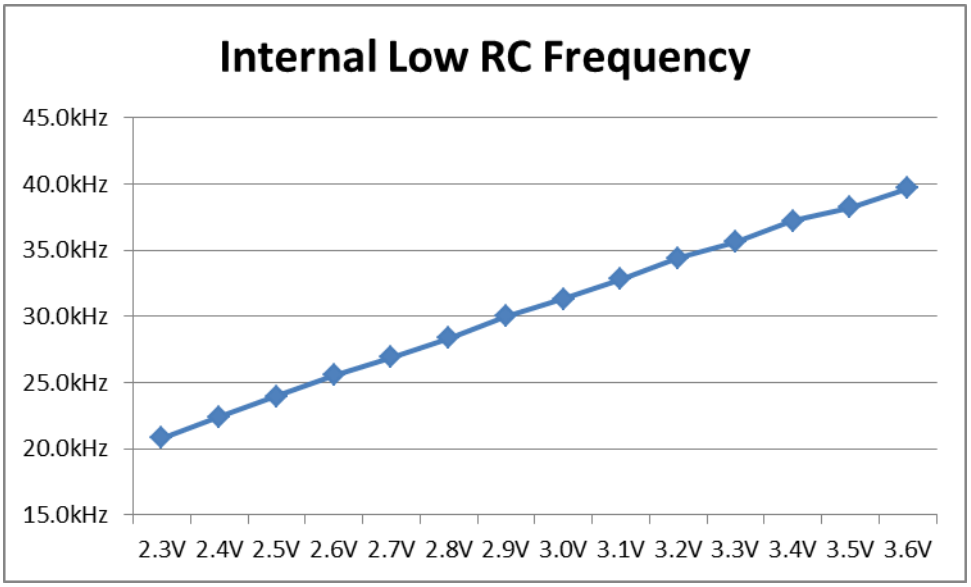
4.4 SYSTEM HIGH CLOCK

The system high clock is only from internal 8MHz oscillator RC type (IHRC). The system clock in normal mode is divided by 4 ,8,16 or 32 controlled by “High_Clk_Div of code option”, to be the instruction cycle (Fcpu).

4.5 SYSTEM LOW CLOCK

The system low clock source is only from internal RC oscillator (ILRC).The system clock in slow mode is divided by 4 to be the instruction cycle (Fcpu).

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 32 KHz at 3.1V. 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 32KHz@3.1V).***

☞ ***Slow mode Fcpu = Flosc / 4***

The only one condition to stop ILRC is the system into power down mode with watchdog disable or enable. If watchdog set “Always_On” and system into power down mode, the ILRC actives well and system will be reset until watchdog overflow occuring.

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

```
B0BSET      FCPUM0          ; To stop IHRC and ILRC or 32k crystal called power down mode
                            ; (sleep mode).
```

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

4.5.1 SYSTEM CLOCK MEASUREMENT

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

Example: Fcpu instruction cycle of external oscillator.

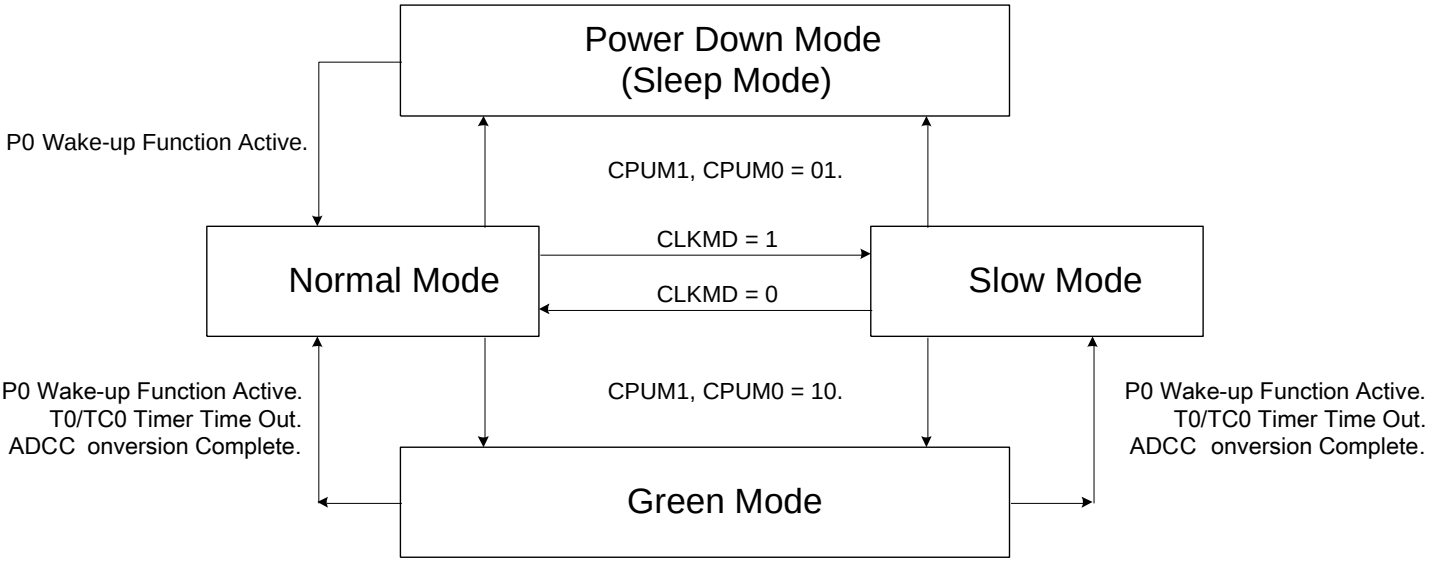
```
        B0BSET    P1M.0       ; Set P1.0 to be output mode for outputting Fcpu toggle signal.
@@:
        B0BSET    P1.0        ; Output Fcpu toggle signal in low-speed clock mode.
        B0BCLR    P1.0        ; Measure the Fcpu frequency by oscilloscope.
        JMP       @B
```

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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



System Mode Switching Diagram

Operating mode description

MODE	NORMAL	SLOW	GREEN	POWER DOWN (SLEEP)	REMARK
Fhosc	Running	By STPHX	By STPHX	Stop	
Flosc	Running	Running	Running	Stop	
CPU instruction	Executing	Executing	Stop	Stop	
T0 timer	* Active	*Active	*Active	Inactive	* Active if T0EN=1
TC0 Timer	Active	Active	Controlled by TC0GN	Inactive	
ADC	Active	*Active	*Active	Stop	*Active if high clock still running. (STPHX=0)
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	Refer to code option description
Internal interrupt	T0, TC0, *ADC,UART	T0, TC0, *ADC,UART	T0, *TC0, *ADC	All inactive	* Active if high clock still running. (STPHX=0) **Active if TC0GN=1
External interrupt	P00 & P01	P00 & P01	P00 & P01	P00 & P01	
Wakeup source	-	-	P0, T0, **TC0 *ADC	P0	*Active if high clock still running. (STPHX=0) **Active if TC0GN=1

- ***Note_1: When system into green mode with conditions of UART function enable and TX or RX still running, the system will be wakeup. @ICE MODE***
- ***Note_2: P04 and P05 can not wakeup IC when UART function enable. @Real chip***
- ***Note_3: When system into green mode with conditions of Code option IHRC_RTC function enable and P06/P07 still running, the system will be wakeup. @ICE MODE***

5.2 SYSTEM MODE SWITCHING

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

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

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

Example: Switch normal mode to slow mode.

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

Example: Switch slow mode to normal mode (The IHRC oscillator is still running)

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

Example: Switch slow mode to normal mode (The IHRC oscillator stops)

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

```
        B0BCLR      FSTPHX          ; Turn on the IHRC oscillator.

        B0MOV       Z, #54          ; If VDD = 3.2V, ILRC =32KHz (typical) will delay
@@:     DECMS       Z               ; 0.125ms X 162 = 20.25ms for external clock stable
        JMP         @B

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

Example: Switch normal/slow mode to green mode.

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

> ***Note: If T0 timer wakeup function is disabled in the green mode, the wakeup pin P0 can wakeup the system backs to the previous operation mode.***

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

```
; Set T0 timer wakeup function.
        B0BCLR      FT0IEN      ; To disable T0 interrupt service
        B0BCLR      FT0EN       ; To disable T0 timer
        MOV         A,#20H      ;
        B0MOV       T0M,A       ; To set T0 clock = Fcpu / 64
        MOV         A,#74H
        B0MOV       T0C,A       ; To set T0C initial value = 74H (To set T0 interval = 10 ms)

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

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

➢
✸ ***Note: During the green mode with T0 wake-up function, the wakeup pins and T0 can wakeup the system back to the last mode. T0 wake-up period is controlled by program and T0EN must be set.***

Example: Switch normal/slow mode to Green mode and enable ADC wakeup function.

```
; Set ADC timer wakeup function.
        MOV         A,#11111111b
        B0MOV       VREG,A      ; To Turn On all analog voltage regulators.
        MOV         A,#00000111b
        B0MOV       AMPM1,A     ; To Set PGIA function.
        B0BSET      FADCENB     ; To enable ADC Function

; Go into green mode with high clock running
        B0BSET      FCPUM1      ;To set CPUM0 = 1
```

➢
✸ ***Note_1: when system into green mode with conditions of ADC function enable and high clock still running, the system will be wakeup when ADC conversion complete.***
✸ ***Note_2: The ADC green mode wakeup function is disable when ADCEN=0 or stop high clock (STPHX=1) is set before into green mode.***

5.3 WAKEUP

5.3.1 OVERVIEW

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

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

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 64 internal high-speed RC oscillator (IHRC) clocks as the wakeup time to stable the oscillator circuit. After the wakeup time, the system goes into the normal mode.

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

The value of the power down wakeup time is as the following.

The Wakeup time = 1/Fhosc * 64 (sec) + high clock start-up time

- ***Note: The high clock start-up time is depended on the VDD. In general, high clock start-up time will be several micro-second (us) at VDD=3V.***

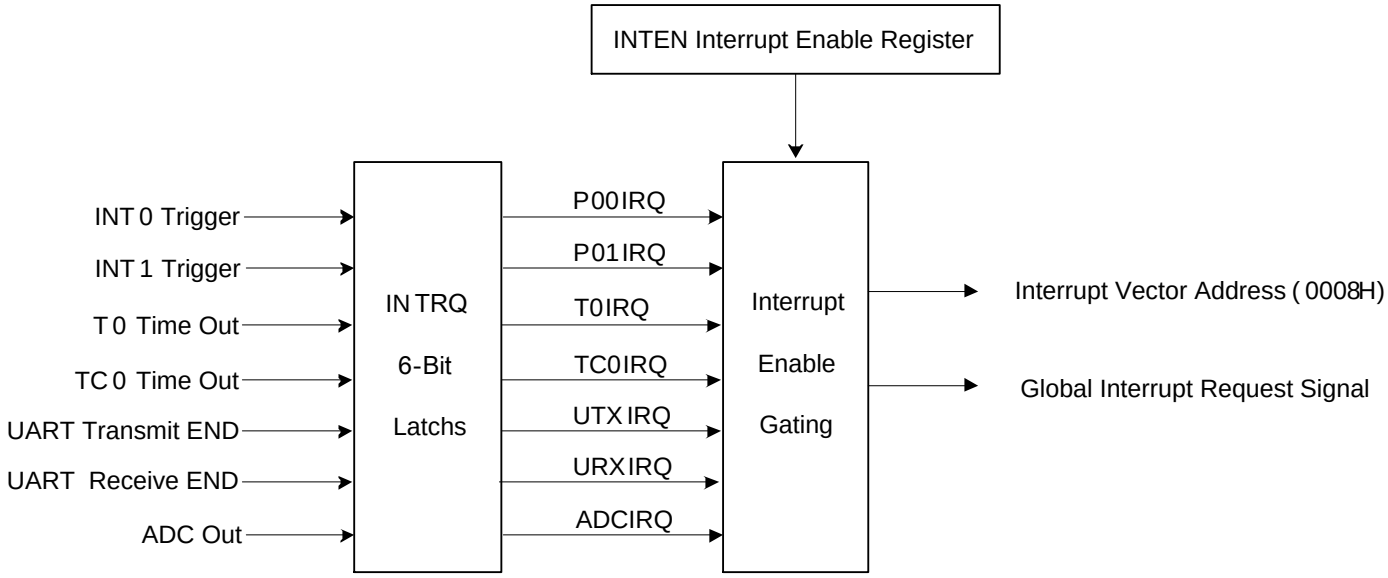
- **Example: The system is waked up from power down (sleep mode) by P0 level chande. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.**

The wakeup time = 1/Fhosc * 64 = 8us (Fhosc = 8MHz)
The total wakeup time = 8us + oscillator start-up time (5us)

6 INTERRUPT

6.1 OVERVIEW

This MCU provides five interrupt sources, including three internal interrupt (T0/TC0/ADC) and two external interrupt (INT0/UART). The external interrupt can wakeup the chip while the system is switched from power down mode to high-speed normal mode, and interrupt request is latched until return to normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to “0” for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to “1” to accept the next interrupts’ request. All of the interrupt request signals are stored in INTRQ register.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	ADCIEN	-	TC0IEN	T0IEN	URXIEN	UTXIEN	P01IEN	P00IEN
Read/Write	R/W	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	-	0	0	0	0	0	0

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

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

Bit 2 **UTXIEN:** UART transmit interrupt control bit.
0 = Disable UART transmit interrupt function.
1 = Enable UART transmit interrupt function.

Bit 3 **URXIEN:** UART receive interrupt control bit.
0 = Disable UART receive interrupt function.
1 = Enable UART receive interrupt function.

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

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

Bit 7 **ADCIEN:** ADC interrupt control bit.
0 = Disable ADC interrupt function.
1 = Enable ADC 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	ADCIRQ	-	TC0IRQ	T0IRQ	URXIRQ	UTXIRQ	P01IRQ	P00IRQ
Read/Write	R/W	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	-	0	0	0	0	0	0

Bit 0 P00IRQ: External P0.0 interrupt request bit.
0 = Non INT0 interrupt request.
1 = INT0 interrupt request.

Bit 1 P01IRQ: External P0.1 interrupt request bit.
0 = Non INT1 interrupt request.
1 = INT1 interrupt request.

Bit 2 UTXIRQ: UART transmit interrupt request flag..
0 = None UART transmit interrupt request..
1 = UART transmit interrupt request.

Bit 3 URXIRQ: UART receive interrupt request flag.
0 = None UART receive interrupt request..
1 = UART receive interrupt request.

Bit 4 T0IRQ: T0 timer interrupt request control bit.
0 = Non T0 interrupt request.
1 = T0 interrupt request.

Bit 5 TC0IRQ: TC0 timer interrupt request control bit.
0 = Non TC0 interrupt request.
1 = TC0 interrupt request.

Bit 7 ADCIRQ: ADC interrupt request controls bit.
0 = Non ADC interrupt request.
1 = ADC 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

Bit [3:0] **STKPBn:** Stack pointer, n = 0~3.

➢ **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 EXTERNAL INTERRUPT OPERATION

When the INT0/INT1 trigger occurs, the P00IRQ/P01IRQ will be set to "1" no matter the P00IEN/P01IEN is enable or disable. If the P00IEN/P01IEN = 1 and the trigger event P00IRQ/P01IRQ is also set to be "1". As the result, the system will execute the interrupt vector (ORG 8). If the P00IEN/P01IEN = 0 and the trigger event P00IRQ/P01IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the P00IRQ/P01IRQ is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

✷ ***Note: The interrupt trigger direction of P0.0/P0.1 is control by PEDGE register.***

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

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

Bit[6:5] **P01G[1:0]:** P0.1 interrupt trigger edge control bits.
00 = reserved.
01 = rising edge.
10 = falling edge.
11 = rising/falling bi-direction (Level change trigger).

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

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

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

Example: INT0 interrupt service routine.

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

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

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

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

        RETI                    ; Exit interrupt vector
```

6.7 MULTI-INTERRUPT OPERATION

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

Interrupt Name	*Trigger Event Description*
P00IRQ	P0.0 trigger controlled by PEDGE
P01IRQ	P0.1 trigger controlled by PEDGE
T0IRQ	T0C overflow
TC0IRQ	TC0C overflow
ADCIRQ	ADC converting end.
UTXIRQ	UART transmit end
URXIRQ	UART receive end

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

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

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

INTP00CHK:                                  ; Check INT0 interrupt request
                B0BTS1      FP00IEN         ; Check P00IEN
                JMP         INTP01CHK       ; Jump check to next interrupt
                B0BTS0      FP00IRQ         ; Check P00IRQ
                JMP         INTP00
INTP01CHK:
                B0BTS1      FP01IEN         ; Check P01IEN
                JMP         INTT0CHK        ; Jump check to next interrupt
                B0BTS0      FP01IRQ         ; Check P01IRQ
                JMP         INTP01

INTT0CHK:                                   ; Check T0 interrupt request
                B0BTS1      FT0IEN          ; Check T0IEN
                JMP         INTTC0CHK       ; Jump check to next interrupt
                B0BTS0      FT0IRQ          ; Check T0IRQ
                JMP         INTT0           ; Jump to T0 interrupt service routine

INTTC0CHK:                                  ; Check TC0 interrupt request
                B0BTS1      FTC0IEN         ; Check TC0IEN
                JMP         INTTC0CHK       ; Jump check to next interrupt
                B0BTS0      FTC0IRQ         ; Check TC0IRQ
                JMP         INTTC0          ; Jump to TC0 interrupt service routine

INTADCHK:                                   ; Check ADC interrupt request
                B0BTS1      FADCIEN         ; Check ADCIEN
                JMP         INTUTXCHK       ; Jump check to next interrupt
                B0BTS0      FADCIRQ         ; Check ADCIRQ
                JMP         INTADC          ; Jump to ADC interrupt service routine
```

```
INTUTXCHK:                              ; Check UTX interrupt request
            B0BTS1      FUTXIEN         ; Check UTXIEN
            JMP         INTURXCHK       ; Jump check to next interrupt
            B0BTS0      FUTXIRQ         ; Check UTXIRQ
            JMP         INTUTX          ; Jump to UART TX interrupt service routine

INTURXCHK:                              ; Check URX interrupt request
            B0BTS1      FURXIEN         ; Check URXIEN
            JMP         INT_EXIT        ; Jump to exit of IRQ
            B0BTS0      FURXIRQ         ; Check URXIRQ
            JMP         INTURX          ; Jump to UART RX interrupt service routine

INT_EXIT:
            …                           ; Pop routine to load ACC and PFLAG from buffers.

            RETI                        ; Exit interrupt vector
```

7 I/O PORT

7.1 I/O PORT MODE

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

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
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

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

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

0C3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3M	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

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

✷ ***Note:***
1. ***Users can program them by bit control instructions (B0BSET, B0BCLR).***
2. ***Port 0 ~ Port 3 are bi-direction I/O port.***
3. ***Port P2 ~ P3 is high-sink I/O Pin, It can drive seven-segment display.***

7.2 I/O PIN SHARE WITH LCD FUNCTION

The microcontroller builds in LCD functions. The LCD driver output pins share with GPIO function, which can be configured by setting PxSEG registers.

08BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2SEG	P27SEG	P26SEG	P25SEG	P24SEG	P23SEG	P22SEG	P21SEG	P20SEG
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	1	1	1	1	1	1	1	1

Bit[7:0] **P2nSEG:** Port 2 function control bit
0 = Set as LCD function Pin. (SEG8~SEG15)
1 = Set as IO function Pin. (P27~P20)

08CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3SEG	P37SEG	P36SEG	P35SEG	P34SEG	P33SEG	P32SEG	P31SEG	P30SEG
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	1	1	1	1	1	1	1	1

Bit[7:0] **P3nSEG:** Port 3 function control bit
0 = Set as LCD function Pin. (SEG0~SEG7)
1 = Set as IO function Pin. (P37~P30)

Example: I/O or LCD function selecting

```
CLR     P2SEG       ; Set P2 ports to be LCD mode.
CLR     P3SEG       ; Set P3 ports to be LCD mode.

MOV     A,#0FFh
B0MOV   P2SEG,A     ; Set P2 ports to be I/O mode.

MOV     A,#0Fh
B0MOV   P3SEG,A     ; Set P34~P37 ports to be LCD mode.
                    ; Set P30~P33 ports to be I/O mode.
```

Example: I/O mode selecting

```
CLR     P0M         ; Set all ports to be input mode.
CLR     P1M
CLR     P2M
CLR     P3M

MOV     A, #0FFH    ; Set all ports to be output mode.
B0MOV   P0M,A
B0MOV   P1M, A
B0MOV   P2M,A
B0MOV   P3M, A

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

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

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

B0BSET  P3M.0       ; Set P3.0 to be output mode.
```

✸ ***Note:***

1. **P2/P3 I/O is share pin with LCD function**
2. **Port P2 ~ P3 is high-sink I/O Pin, It can drive seven-segment display.**

7.3 I/O PULL UP REGISTER

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R
Read/Write	W	W	W	W	W	W	W	W
After Reset	0	0	0	0	0	0	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	-	-	-	-	-	-	P11R	P10R
Read/Write	-	-	-	-	-	-	W	W
After reset	-	-	-	-	-	-	0	0

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

0DEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3UR	P37R	P36R	P35R	P34R	P33R	P32R	P31R	P30R
Read/Write	W	W	W	W	W	W	W	W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **Pn0R~Pn7R:** I/O port pull-up resistor control bit. (n = 0~1).
0 = Disable pull-up resistor.
1 = Enable pull-up resistor.

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

Example: I/O Pull up Register

```
MOV     A, #0FFH        ; Enable Port1 Pull-up register,
B0MOV   P1UR,A
```

7.4 I/O PORT DATA REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	P07	P06	P05	P04	P03	P02	P01	P00
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

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	-	-	-	-	-	-	P11	P10
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

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

0D3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3	P37	P36	P35	P34	P33	P32	P31	P30
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **Pn0~Pn7:** I/O port data buffer. (n = 0~1).
0 = Input low or output low.
1 = Input high or output high.

Example: Read data from input port.

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

Example: Write data to output port.

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

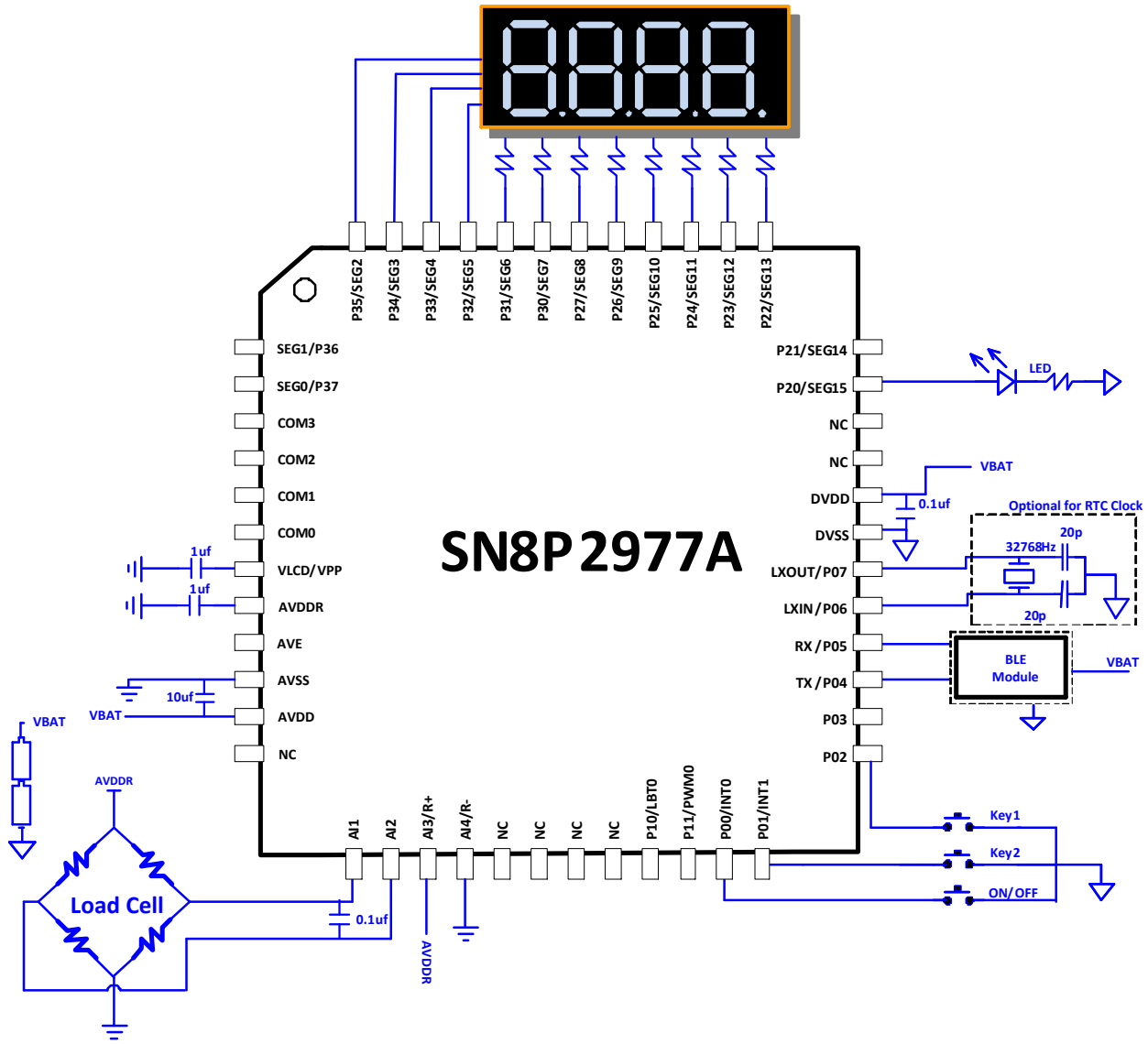
Example: Write one bit data to output port.

```
B0BSET  P1.0        ; Set P1.0 to be “1”.
B0BCLR  P1.0        ; Set P1.0 to be “0”.
```

7.5 High-sink current I/O PORT

The SN8P2977A has a built-in High sink I/O to support drive seven-segment display from port P2/P3 and it with typical 60mA current sinking capacity. when the pin is LOW, it can only accept that much current flowing to ground.
When P2/P3 is high-current sink I/O mode, Advisable AI3/R+ pin to AVDDR, AI4/R- pin to ground and ADC reference voltage set as External reference, avoid high current sink grounding influence ADC operation.

Weight Scale Application Circuit
LQFP48-Pin, LED4*8



✸ ***Note: In High-current sink I/O mode, Recommend ADC reference voltage set external reference IRVS[3:0]=11xx and AI3/R+ pin to AVDDR, AI4/R- pin to AVSS.***

8 TIMERS

8.1 WATCHDOG TIMER

The watchdog timer (WDT) is a binary up counter designed for monitoring program execution. If the program goes into the unknown status by noise interference, WDT overflow signal raises and resets MCU. Watchdog clock controlled by code option and the clock source is internal low-speed oscillator (32 KHz @3V).

Watchdog overflow time = 16384/ Internal Low-Speed oscillator (sec).

VDD	*Internal Low RC Freq.*	*Watchdog Overflow Time*
3.3V	32KHz	512ms

1. Note:

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

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

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

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

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

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

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

Watchdog timer application note is as following.

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

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

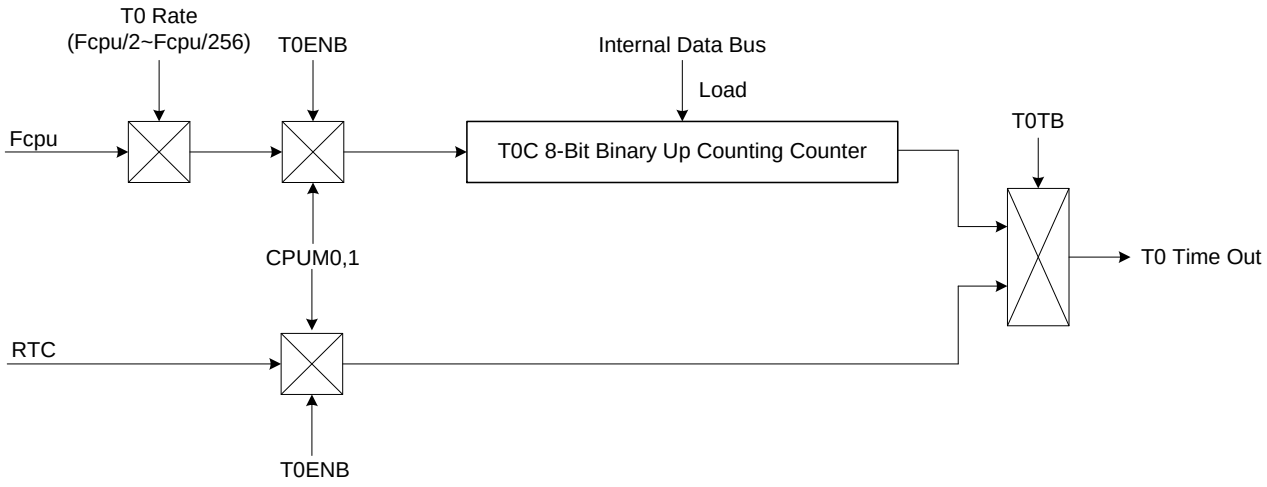
```
Main:
        …                       ; Check I/O.
        …                       ; Check RAM
Err:    JMP $                   ; I/O or RAM error. Program jump here and don't
                                ; clear watchdog. Wait watchdog timer overflow to reset IC.

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

8.2 TIMER 0 (T0)

8.2.1 OVERVIEW

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

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



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

8.2.2 T0M MODE REGISTER

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	TC0X8	TC0GN	T0TB
Read/Write	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
After reset	0	0	0	0	-	0	0	0

Bit 0 **T0TB:** RTC clock source control bit.
0 = Disable RTC (T0 clock source from Fcpu).
1 = Enable RTC, T0 will be 0.5 sec RTC (Low clock must be 32768 crystal).

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

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

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

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

✸ ***Note: In RTC mode, the T0 interval time is fixed at 0.5 sec and T0C is 256 counts.***

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 8MHz IHRC. Fcpu=Fosc/8. Select T0RATE=010 (Fcpu/64).**

*T0C initial value = 256 - (T0 interrupt interval time * input clock)*
*= 256 - (10ms * 8MHz / 8 / 64)*
*= 256 - (10^{-2} * 8* 10^{6} / 8 / 64)*
= 100
= 64H

The basic timer table interval time of T0.

T0RATE	T0CLOCK	High speed mode (Fcpu = 8MHz / 8)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fcpu/256	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.125 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

✸ ***Note: In RTC mode, T0C is 256 counts and generatesT0 0.5 sec interval time. Don't change T0C value in RTC mode.***

8.2.4 T0 TIMER OPERATION SEQUENCE (High_Clk = IHRC)

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

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

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

☞ **Set T0 timer rate.**

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

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

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

or

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

☞ **Set T0 interrupt interval time.**

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

☞ **Set T0 timer function mode.**

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

☞ **Enable T0 timer.**

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

8.2.5 RTC OPERATION SEQUENCE (High_Clk ="IHRC_RTC" and "T0TB = 1")

T0 timer with RTC operation sequence **(High_Clk code option = "IHRC_RTC" and "T0TB = 1"**) of setup T0 timer is as following.

- **Declare buffer.**

```
.DATA
        OLDT0C      DS          1
        NEWT0C      DS          1
        T0FLAG      DS          1
                    T0IRQFLAG   EQU         T0FLAG.0
```

- **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 T0M register.**

```
        MOV         A, #00000000b
        B0MOV       T0M,A
```

- **Set T0 clock source from RTC.**

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

- **Set T0 interrupt interval time.**

```
        CLR         T0C         ; Clear T0C value.
        CLR         OLDT0C      ; Clear OLDT0C value.
        CLR         NEWT0C      ; Clear NEWT0C value.
        CLR         T0FLAG      ; Clear T0FLAG before execute Main routing.
```

- **Disable T0 timer function mode.**

```
        B0BCLR      FT0IEN      ; Disable T0 interrupt function.
```

- **Enable T0 timer with RTC function.**

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

- **Execute MAIN routing polling T0 timer.(Execute MAIN routing interval time is not more than 200ms).**

```
        B0BCLR      FT0IRQ      ; Clear FT0IRQ.
MAIN:
        CALL        CKT_T0CVAL  ; Check T0C value overflow
        CALL        CKT_T0FLAG  ; Check T0C overflow Flag and update time.
        .
        .
        JMP         MAIN        ; Jmp MAIN.
```

- **CKT_T0CVAL sub-routing (Check T0C value status).**

```
CKT_T0CVAL:
        MOV         A, T0C      ; Read T0C value
        MOV         NEWT0C, A   ; Save to NEWT0C
        SUB         A, OLDT0C   ; A sub OLDT0C value.
        B0BTS0      FC          ; If FC = 0, borrow
```

```
                JMP         EXIT_CKTT0CVAL:  ; If FC = 1, jmp EXIT_CKTT0CVAL

                B0BSET      T0IRQFLAG        ; Set T0IRQFLAG (T0C counts overflow.).
EXIT_CKTT0CVAL:
                MOV         A, NEWT0C
                MOV         OLDT0C, A        ; Update T0C value
                RET                          ; Exit sub-routing.
```

➢ **CKT_T0FLAG sub-routing (Check T0 timer overflow flag).**

```
CKT_T0FLAG:
                B0BTS1      T0IRQFLAG        ; Check T0IRQ status.
                JMP         EXIT_CKTT0FLAG   ; Jmp EXIT_CKTT0FLAG.

                B0BCLR      T0IRQFLAG        ; Clear T0IRQFLAG.
                CALL        DELAY            ; Call delay time = over 1/32.768ms (for RTC limit).
                B0BCLR      FT0IRQ           ; Clear FT0IRQ.
                .
                CALL        UPDATE_TIME      ; Update time.
                .
EXIT_CKTT0FLAG:
                RET                          ; Exit sub-routing.
```

➢ **Into green mode before.**

```
                CALL        CKT_T0CVAL       ; Check T0C value overflow
                CALL        CKT_T0FLAG       ; Check T0C overflow Flag and update time.
```

➢ **Process green mode after wakeup.**

```
INTO_GREENMODE: .
                B0BCLR      FCPUM0
                B0BSET      FCPUM1           ; Into green mode

WAKEUP:
                B0BTS1      FT0IRQ           ; Check FT0IRQ
                JMP         CKT_OTHER        ; Check other trigger wakeup source.

                CALL        DELAY            ; Call delay time = over 1/32.768ms (for RTC limit).

                CLR         T0FLAG           ; Clear T0FLAG.
                B0BCLR      FT0IRQ           ; Clear FT0IRQ.

                MOV         A, T0C
                MOV         OLDT0C, A        ; Update T0C value

                CALL        UPDATE_TIME      ; Update time.
                .
CKT_OTHER:
                .
                .
```

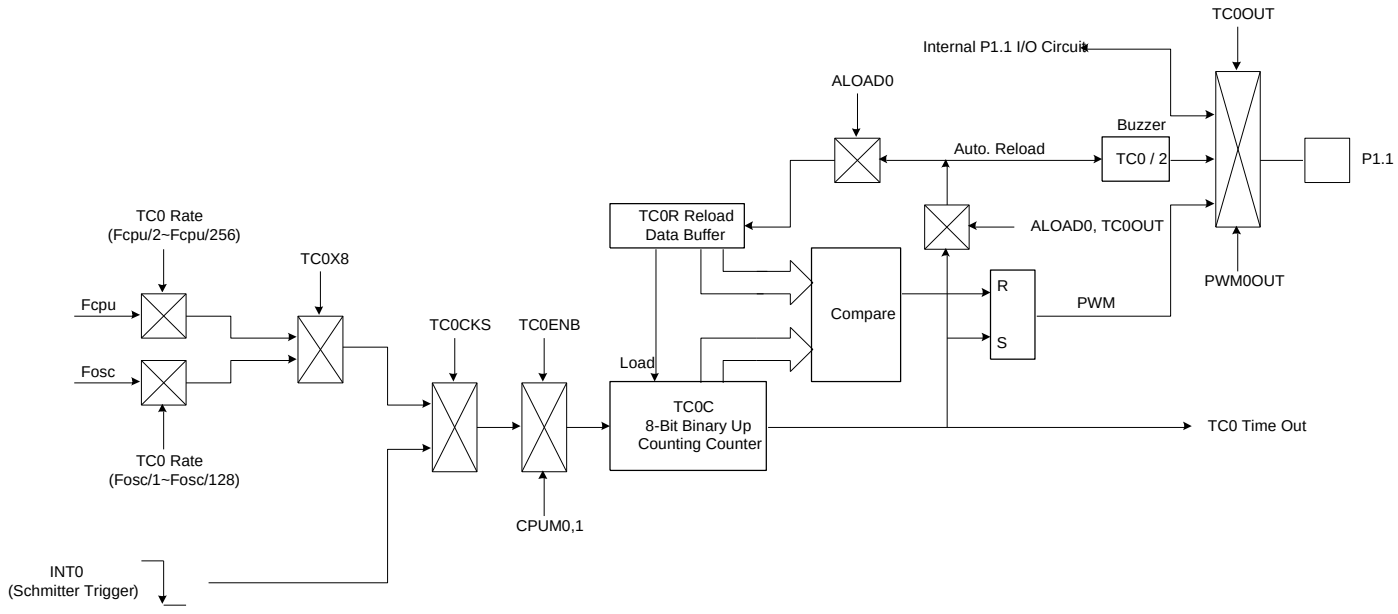
8.3 TIMER/COUNTER 0 (TC0)

8.3.1 OVERVIEW

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

The main purposes of the TC0 timer is as following.

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



8.3.2 TC0M MODE REGISTER

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

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

Bit 1 **TC0OUT:** TC0 time out toggle signal output control bit. **Only valid when PWM0OUT = 0.**
0 = Disable, P1.1 is I/O function.
1 = Enable, P1.1 is output TC0OUT signal.

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

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

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

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

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

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

8.3.3 TC0X8, TC0GN FLAGS

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	TC0X8	TC0GN	T0TB
Read/Write	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
After reset	0	0	0	0	-	0	0	0

Bit 0 **T0TB:** RTC clock source control bit.
0 = Disable RTC (T0 clock source from Fcpu).
1 = Enable RTC.

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

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

Bit [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.

2.
✸ ***Note: Under TC0 event counter mode (TC0CKS=1), TC0X8 bit and TC0RATE are useless.***

8.3.4 TC0C COUNTING REGISTER

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

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

The equation of TC0C initial value is as following.

TC0C initial value = 256 - (TC0 interrupt interval time * input clock)

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

☞ **Example: To set 10ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=0, TC0X8=0) and no PWM output (PWM0=0). High clock is external 8MHz. Fcpu=Fosc/8. Select TC0RATE=010 (Fcpu/64).**

*TC0C initial value = N - (TC0 interrupt interval time * input clock)*
*= 256 - (10ms * 8MHz / 8 / 64)*
*= 256 - (10^{-2} * 8 * 10^{6} / 8 / 64)*
= 100
= 64H

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

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

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

TC0RATE	TC0CLOCK	High speed mode (Fosc = 8MHz)	
		Max overflow interval	One step = max/256
000	Fosc/128	4.096 ms	16 us
001	Fosc/64	2.048 ms	8 us
010	Fosc/32	1.024 ms	4 us
011	Fosc/16	0.512 ms	2 us
100	Fosc/8	0.256 ms	1 us
101	Fosc/4	0.128 ms	0.5 us
110	Fosc/2	0.064 ms	0.25 us
111	Fosc/1	0.032 ms	0.125 us

8.3.5 TC0R AUTO-LOAD REGISTER

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

3.

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

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

The equation of TC0R initial value is as following.

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

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

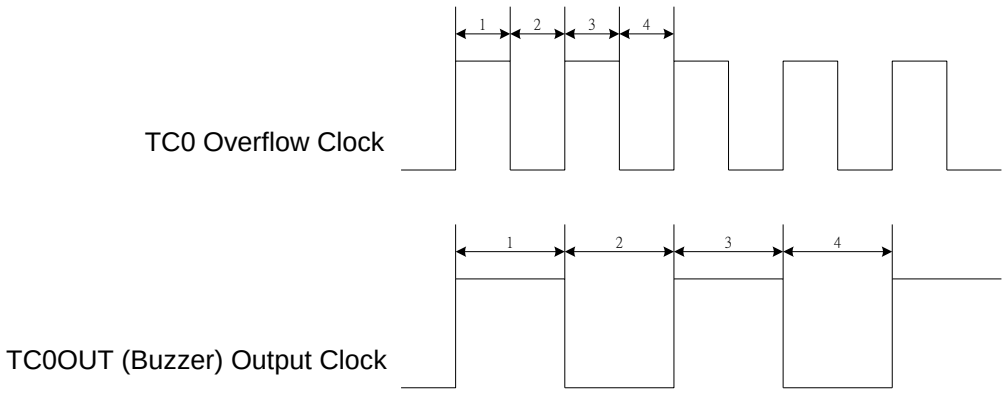
TC0X8	TC0R valid value	TC0R value binary type
0 (Fcpu/2~Fcpu/256)	0x00~0xFF	00000000b~11111111b
1 (Fosc/1~Fosc/128)	0x00~0xFF	00000000b~11111111b

☞ **Example: To set 10ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=0, TC0X8=0) and no PWM output (PWM0=0). High clock is external 8MHz. Fcpu=Fosc/8. Select TC0RATE=010 (Fcpu/64).**

*TC0R initial value = N - (TC0 interrupt interval time * input clock)*
*= 256 - (10ms * 8MHz / 8 / 64)*
*= 256 - (10^{-2} * 8 * 10^{6} / 8 / 64)*
= 100
= 64H

8.3.6 TC0 CLOCK FREQUENCY OUTPUT (BUZZER)

Buzzer output (TC0OUT) is from TC0 timer/counter frequency output function. By setting the TC0 clock frequency, the clock signal is output to P1.1 and the P1.1 general purpose I/O function is auto-disable. The TC0OUT frequency is divided by 2 from TC0 interval time. TC0OUT frequency is 1/2 TC0 frequency. The TC0 clock has many combinations and easily to make difference frequency. The TC0OUT frequency waveform is as following.



☞ **Example: Setup TC0OUT output from TC0 to TC0OUT (P1.1). The external high-speed clock is 8MHz. (Fcpu=Fosc/8) The TC0OUT frequency is 0.5KHz. Because the TC0OUT signal is divided by 2, set the TC0 clock to 1KHz. The TC0 clock source is from external oscillator clock. T0C rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 6.**

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

MOV     A,#6            ; Set the auto-reload reference value
B0MOV   TC0C,A
B0MOV   TC0R,A

B0BSET  FTC0OUT         ; Enable TC0 output to P1.1 and disable P1.1 I/O function
B0BSET  FALOAD0         ; Enable TC0 auto-reload function
B0BSET  FTC0ENB         ; Enable TC0 timer
```

4.

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

8.3.7 TC0 TIMER OPERATION SEQUENCE

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

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

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

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

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

☞ **Set TC0 timer clock source.**

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

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

5.

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

☞ **Set TC0 timer auto-load mode.**

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

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

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

☞ **Set TC0 timer function mode.**

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

☞ **Enable TC0 timer.**

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

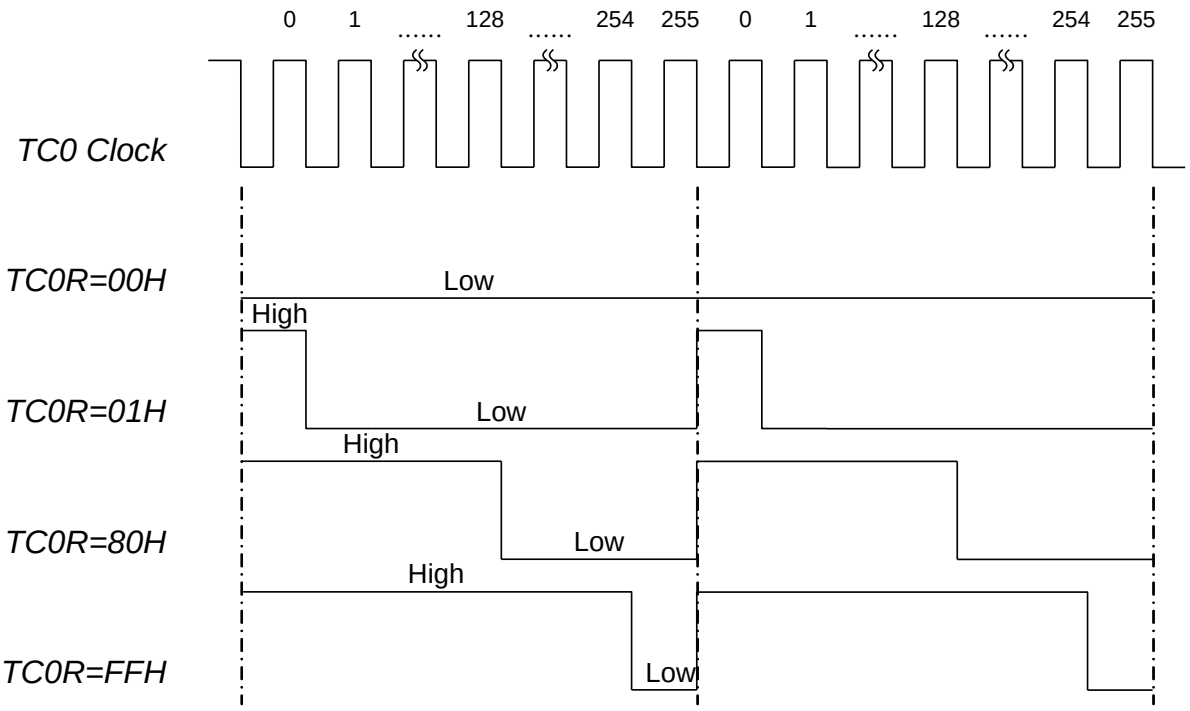
8.4 PWM0 MODE

8.4.1 OVERVIEW

PWM function is generated by TC0 timer counter and output the PWM signal to PWM0OUT pin (P1.1). The 8-bit counter counts modulus 256 bits. The value of the 8-bit counter (TC0C) is compared to the contents of the reference register (TC0R). When the reference register value (TC0R) is equal to the counter value (TC0C), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The ratio (duty) of the PWM0 output is TC0R/256.

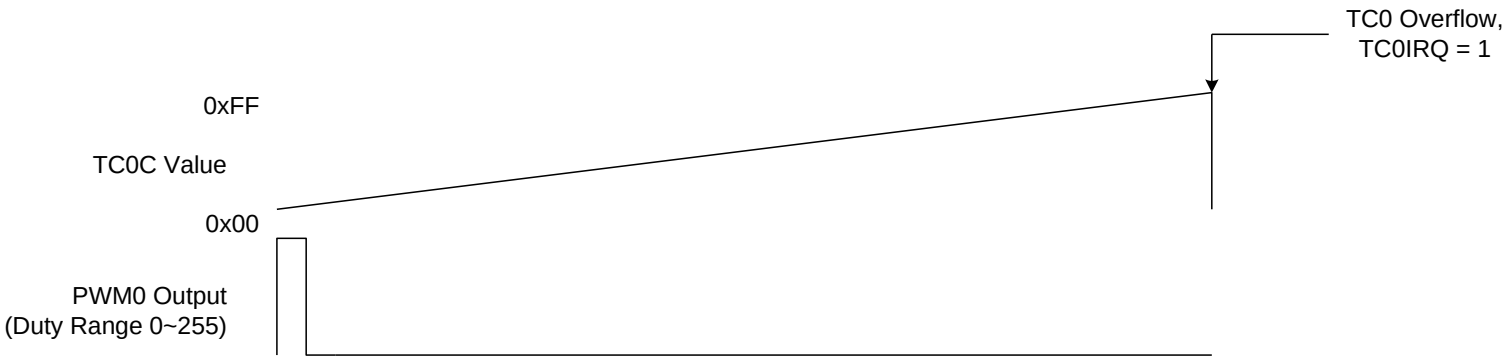
PWM duty range	TC0C valid value	TC0R valid bits value	MAX. PWM Frequency (Fcpu = 8MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	31.25K	Overflow per 256 count

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



8.4.2 TC0IRQ AND PWM DUTY

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



8.4.3 PWM PROGRAM EXAMPLE

☞ **Example: Setup PWM0 output from TC0 to PWM0OUT (P1.1). The external high-speed oscillator clock is 8MHz. Fcpu = Fosc/8. The duty of PWM is 30/256. The PWM frequency is about 1KHz. The PWM clock source is from external oscillator clock. TC0 rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 30.**

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

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

B0BSET    FPWM0OUT        ; Enable PWM0 output to P1.1 and disable P1.1 I/O function
B0BSET    FTC0ENB         ; Enable TC0 timer
```

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

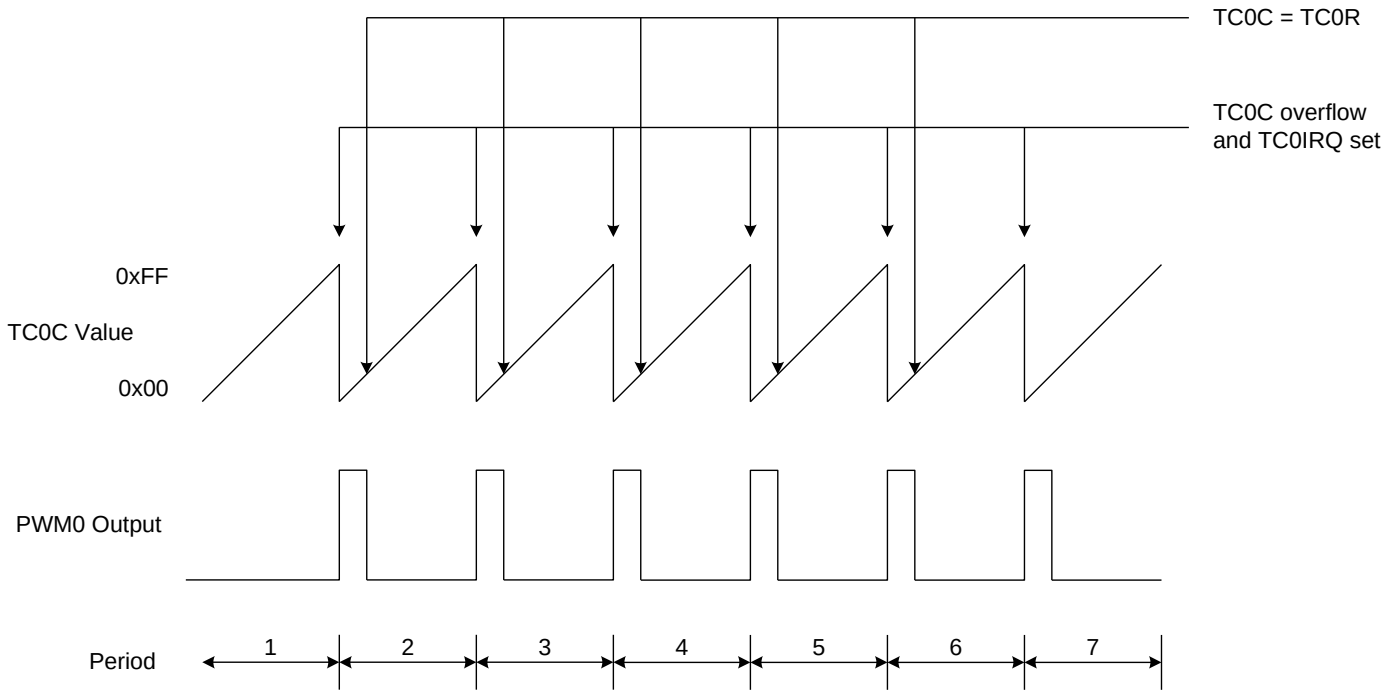
☞ **Example: Modify TC0R registers' value.**

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

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

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

8.4.4 PWM0 DUTY CHANGING NOTICE

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



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

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

9 UART

9.1 OVERVIEW

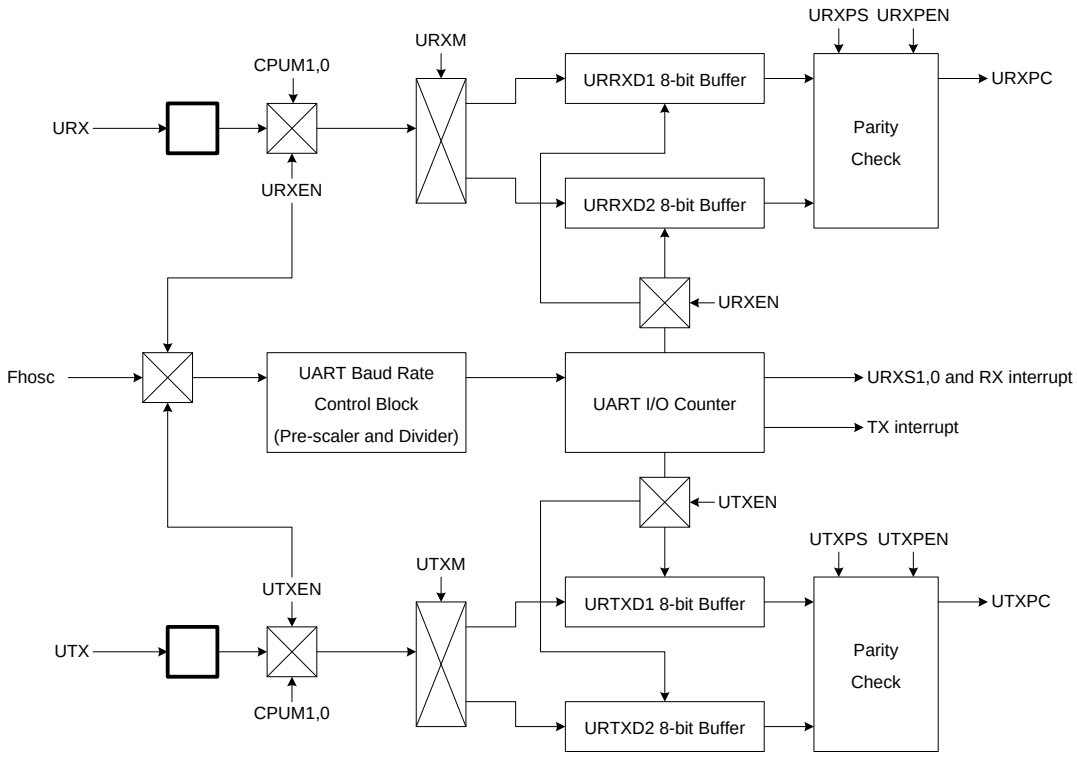
The UART interface is an universal asynchronous receiver/transmitter method. The serial interface is applied to low speed data transfer and communicate with low speed peripheral devices. The UART transceiver of Sonix 8-bit MCU allows RS232 standard and supports one byte data length. The transfer format has start bit, 8-bit data, parity bit and stop bit. Programmable baud rate supports different speed peripheral devices.

The UART features include the following:

- **Full-duplex, 2-wire asynchronous data transfer.**
- **Programmable baud rate.**
- **8-bit data length.**
- **Odd and even parity bit.**
- **End-of-Transfer interrupt.**
- **Support break pocket function.**
- **Support wide range baud rate.**

9.2 UART OPERATION

The UART RX (P05) and TX (P04) pins are shared with GPIO. When UART enables (RXDEN=1, TXDEN=1), the UART shared pins transfers to UART purpose and disable GPIO function automatically. When UART disables, the UART pins returns to GPIO last status. The UART data buffer length supports 1-byte. After UART RX operation finished, the RXIRQ sets as “1”. After UART TX operation finished, the TXIRQ sets as “1”. The UART IRQ bits are cleared by program. If the RXIEN or TXIEN set to enable, the RXIRQ and TXIRQ triggers the interrupt request and program counter jumps to interrupt vector to execute interrupt service routine.



UART Interface Circuit Diagram

9.3 UART TRANSFER FORMAT

The UART also builds in “Busy Bit” to indicate UART bus status. URXBZ bit is UART RX operation indicator. UTXBZ bit is UART TX operation indicator. If bus is transmitting, the busy bit is “1” status. If bus is finishing operation or in idle status, the busy bit is “0” status.

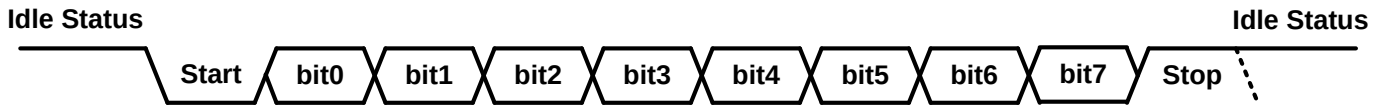
UART TX operation is controlled by loading UTXD data buffer. After UART TX configuration, load transmitted data into UTXD 8-bit buffer, and then UART starts to transmit the pocket following UART TX configuration.

UART RX operation is controlled by receiving the start bit from master terminal. After UART RX configuration, URX pin detects the falling edge of start bit, and then UART starts to receive the pocket from master terminal.

UART provides URXPC bit and UFMER bit to check received pocket. URXPC bit is received parity bit checker. If received parity is error, URXPC sets as “1”. If URXPC bit is zero after receiving pocket, the parity is correct. UFMER bit is received stream frame checker. The stream frame error definition includes “Start bit error”, “Stop bit error”, “Stream length error”, “UART baud rate error”... Each of frame error conditions makes UFMER bit sets as “1” after receiving pocket.

The UART transfer format includes “Bus idle status”, “Start bit”, “8-bit Data”, “Parity bit” and “Stop bit” as following.

UART Transfer Format with Parity Bit



UART Transfer Format without Parity Bit

Bus Idle Status
The bus idle status is the bus non-operating status. The UART receiver bus idle status of MCU is floating status and tied high by the transmitter device terminal. The UART transmitter bus idle status of MCU is high status. The UART bus will be set when URXEN and UTXEN are enabled.

Start Bit
UART is a asynchronous type of communication and need a attention bit to offer receiver the transfer starting. The start bit is a simple format which is high to low edge change and the duration is one bit period. The start bit is easily recognized by the receiver.

8-bit Data
The data format is 8-bit length, and LSB transfers first following start bit. The one bit data duration is the unit of UART baud rate controlled by register.

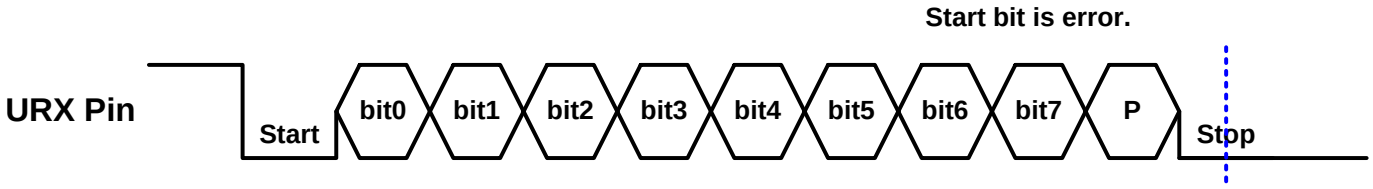
Parity Bit
The parity bit purpose is to detect data error condition. It is an extra bit following the data stream. The parity bit includes odd and even check methods controlled by URXPS/UTXPS bits. After receiving data and parity bit, the parity check executes automatically. The URXPC bit indicates the parity check result. The parity bit function is controlled by URXPEN/UTXPEN bits. If the parity bit function is disabled, the UART transfer contents remove the parity bit and the stop bit follows the data stream directly.

Stop Bit
The stop bit is like start bit using a simple format to indicate the end of UART transfer. The stop bit format is low to high edge change and the duration is one bit period.

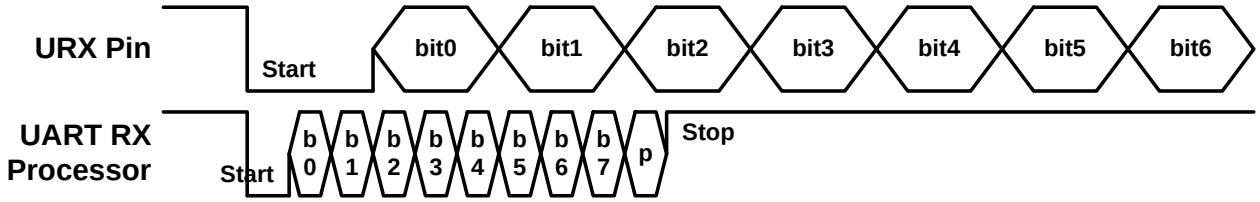
9.4 ABNORMAL POCKET

The abnormal pocket occurs in UART RX mode. Break pocket is one abnormal pocket of the UART architecture. The abnormal pocket includes Stream period error, start bit error, stop bit error…When UART receives abnormal pocket, the UFMER bit will be set “1”, and UART issues URXIRQ. The system finds the abnormal pocket through firmware. UART changes to initial status until detecting next start bit.

UART check the start bit is error and issue UFMER flag, but the UART still finishes receiving the pocket.



UART check the stop bit is error and issue UFMER flag, but the UART still finishes receiving the pocket.



If the host’s UART baud rate isn’t match to receiver terminal, the received pocket is error. But it is not easy to differentiate the pocket is correct or not, because the received error pocket maybe match UART rule, but the data is error. Use checking UFMER bit and URXPC bit status to decide the stream. If the two conditions seem like correct, but the pocket is abnormal, UART will accept the pocket as correct one.

9.5 UART BAUD RATE

UART clock is 2-stage structure including a pre-scaler and an 8-bit buffer. UART clock source is generated from system oscillator called Fhosc. Fhosc passes through UART pre-scaler to get UART main clock called Fuart. UART pre-scaler has 8 selections (Fhosc/1, Fhosc/2, Fhosc/4, Fhosc/8, Fhosc/16, Fhosc/32, Fhosc/64, Fhosc/128) and 3-bit control bits (URS[2:0]). UART main clock (Fuart) purposes are the front-end clock and through UART 8-bit buffer (URCR) to obtain UART processing clock and decide UART baud rate.

UART Pre-scaler Selection, URS[2:0]	UART Main Clock Rate	Fuart (Fhosc=8Hz)
000b	Fhosc/1	8MHz
001b	Fhosc/2	4MHz
010b	Fhosc/4	2MHz
011b	Fhosc/8	1MHz
100b	Fhosc/16	0.5MHz
101b	Fhosc/32	0.25MHz
110b	Fhosc/64	0.125MHz
111b	Fhosc/128	0.625MHz

0D7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URCR	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0
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 UART baud rate clock source is Fhosc and divided by pre-scalar. The equation is as following.

UART Baud Rate = 1/2 *(Fuart * 1/(256 - URCR))…bps .

Fhosc = 8MHz

Baud Rate	UART Pre-scaler	URS[2:0]	URCR (Hex)	UART Baud Rate	Accuracy (%)
1200	Fhosc/16	100b	30	1202	0.16%
2400	Fhosc/16	100b	98	2404	0.16%
4800	Fhosc/16	100b	CC	4808	0.16%
9600	Fhosc/16	100b	E6	9615	0.16%
19200	Fhosc/16	100b	F3	19231	0.16%
38400	Fhosc/1	000b	98	38462	0.16%
51200	Fhosc/1	000b	B2	51282	0.16%
57600	Fhosc/1	000b	BB	57971	0.64%
102400	Fhosc/1	000b	DA	102564	0.16%
115200	Fhosc/1	000b	DD	114286	-0.79%

☞ ***Note: We strongly recommend not to set URCR = 0xFF, or UART operation would be error.***

9.6 UART RECEIVER CONTROL REGISTER

0E3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRX	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0
Read/Write	R/W	R/W	R/W	R	R	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [2:0] **URS[2:0]:** UART per-scalar select bit.
000 = Fhosc/1, 001 = Fhosc/2, 010 = Fhosc/4, 011 = Fhosc/8, 100 = Fhosc/16, 101 = Fhosc/32,
110 = Fhosc/64, 111 = Fhosc/128.

Bit 3 **UFMER:** UART RX stream frame error flag bit.
0 = Collect UART frame.
1 = UART frame is error including start/stop bit, stream length.

Bit 4 **URXPC:** UART RX parity bit checking flag.
0 = Parity bit is correct or no parity function.
1 = Parity bit is error.

Bit 5 **UTXPS:** UART RX parity bit format control bit.
0 = UART RX parity bit format is even parity.
1 = UART RX parity bit format is odd parity.

Bit 6 **URXPEN:** UART RX parity bit control bit.
0 = Disable UART RX parity bit function. The data stream doesn't include parity bit.
1 = Enable UART RX parity bit function. The data stream includes parity bit.

Bit 7 **URXEN:** UART RX control bit.
0 = Disable UART RX. URX pin is GPIO mode or returns to GPIO status.
1 = Enable UART RX. URX pin exchanges from GPIO mode to UART RX mode.

9.7 UART TRANSMITTER CONTROL REGISTER

0E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTX	UTXEN	UTXPEN	UTXPS	-	URXBZ	UTXBZ	-	-
Read/Write	R/W	R/W	R/W	-	R	R	-	-
After reset	0	0	0	-	0	0	-	-

Bit 2 **UTXBZ:** UART TX operating status flag.
0 = UART TX is idle or the end of processing.
1 = UART TX is busy and processing.

Bit 3 **URXBZ:** UART RX operating status flag.
0 = UART RX is idle or the end of processing.
1 = UART RX is busy and processing.

Bit 5 **UTXPS:** UART TX parity bit format control bit.
0 = UART TX parity bit format is even parity.
1 = UART TX parity bit format is odd parity.

Bit 6 **UTXPEN:** UART TX parity bit control bit.
0 = Disable UART TX parity bit function. The data stream doesn't include parity bit.
1 = Enable UART TX parity bit function. The data stream includes parity bit.

Bit 7 **UTXEN:** UART TX control bit.
0 = Disable UART TX. UTX pin is GPIO mode or returns to GPIO status.
1 = Enable UART TX. UTX pin exchanges from GPIO mode to UART TX mode and idle high status.

☞ ***Note: URXBZ and UTXBZ bits are UART operating indicators. After setting UART RX/TX operations, set a "NOP" instruction is necessary, and then check UART status through URXBZ and UTXBZ bits.***

9.8 UART TRANSMITTER CONTROL REGISTER

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UTXD	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **UTXD:** UART transmitted data buffer.

0E6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URXD	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0
Read	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **URXD:** UART received data buffer.

9.9 UART OPERATION EXAMLPE

UART TX Configuration:

```
; Select parity bit function.
        B0BCLR      FUTXPEN     ; Disable UART TX parity bit function.
;or
        B0BSET      FUTXPEN     ; Enable UART TX parity bit function.

; Select parity bit format.
        B0BCLR      FUTXPS      ; UART TX parity bit format is even parity.
;or
        B0BSET      FUTXPS      ; UART TX parity bit format is odd parity.

; Set UART baud rate.
        MOV         A, #value1  ; Set UART pre-scaler URS[2:0].
        B0MOV       URRX, A
        MOV         A, #value2  ; Set UART baud rate 8-bit buffer.
        B0MOV       URCR, A

; Enable UART TX pin.
        B0BSET      FUTXEN      ; Enable UART TX function and UART TX pin.

; Enable UART TX interrupt function.
        B0BCLR      FUTXIRQ     ; Clear UART TX interrupt flag.
        B0BSET      FUTXIEN     ; Enable UART TX interrupt function.

; Load TX data buffer and execute TX transmitter.
        MOV         A, #value3  ; Load 8-bit data to UTXD data buffer.
        B0MOV       UTXD, A
                                ;After loading UTXD, UART TX starts to transmit.
        NOP                     ; One instruction delay for UTXBZ flag.

; Check TX operation.
        B0BTS0      FUTXBZ      ; Check UTXBZ bit.
        JMP         CHKTX       ; UTXBZ=1, TX is operating.
        JMP         ENDTX       ; UTXBZ=0, the end of TX.
```

✸ ***Note: UART TX operation is started through loading UTXD data buffer.***

10 BUZZER FUNCTION

10.1 OVERVIEW

Buzzer function is controlled by BZRENB bit, which be configured as GPIO mode or Buzzer mode. Buzzer output square wave signal through P0.3 Pin with 50% duty, and output frequency is controllable by setting the BZRCKS[1:0] register.

Buzzer frequency Table

BZRCKS1	*BZRCKS0*	*Buzzer frequency*	*Note*
0	0	0.98k Hz	Buzzer clock source From IHRC= 8MHz
0	1	1.96k Hz	
1	0	3.9k Hz	
1	1	7.8k Hz	

10.2 BUZZER CONTROL REGISTER

08DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BZRM	-	-	-	-	-	BZRENB	BZRCKS1	BZRCKS0
Read/Write	-	-	-	-	-	W	W	W
After reset	-	-	-	-	-	0	1	1

Bit [1:0] **URS[1:0]:** Buzzer output frequency.
00 = 0.98 KHz.
01 = 1.96 KHz.
10 = 3.9 KHz.
11 = 7.8 KHz.

Bit 2 **BZRENB:** Buzzer output control bit.
0 : P03 as GPIO Mode.
1 : P03 as Buzzer Mode. Output Buzzer signal

11 LCD DRIVER

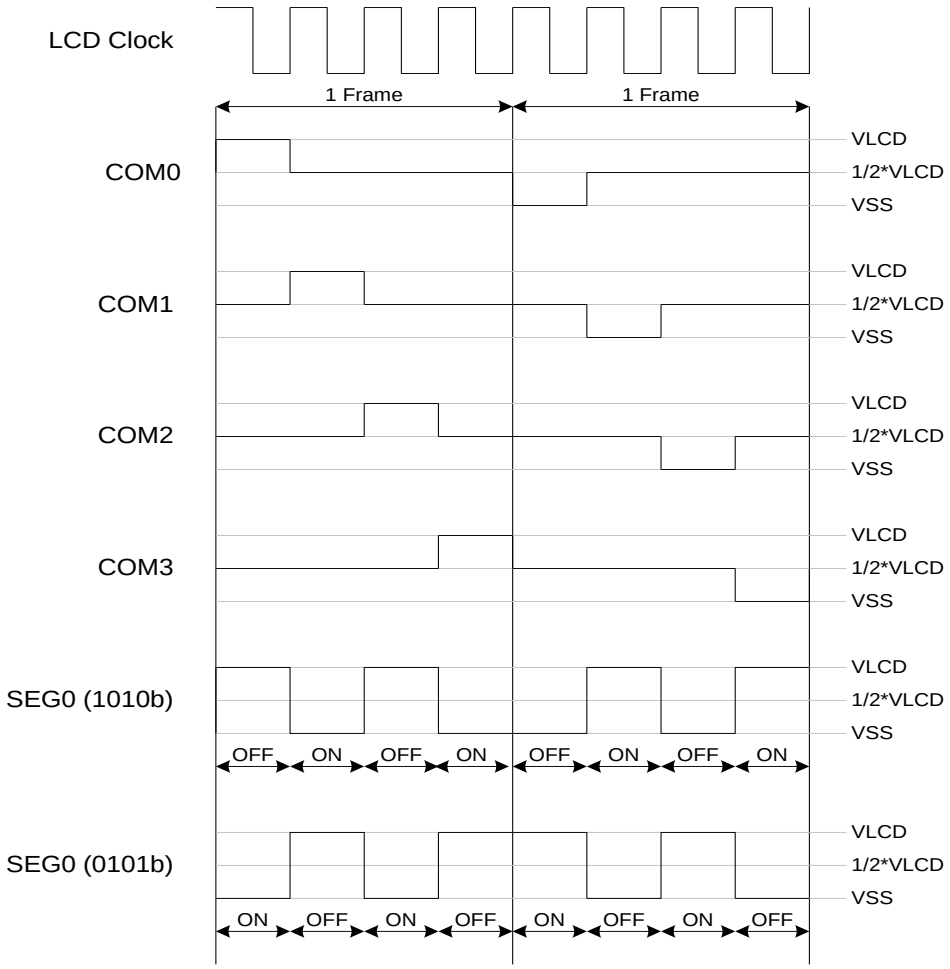
11.1 OVERVIEW

LCD driver includes R-type and C-type structures with 4 common pins and 16 segment pins in the SN8P2977A. The LCD scan timing is 1/4 duty with 1/2 bias or 1/3 bias structure, all support in R-type and C-type mode to yield 64 dots LCD driver. LCD power and bias voltage can be adjusted by additional external bias circuit in R-type LCD driver, and adjusted by setting internal charge pump in C-type LCD driver.

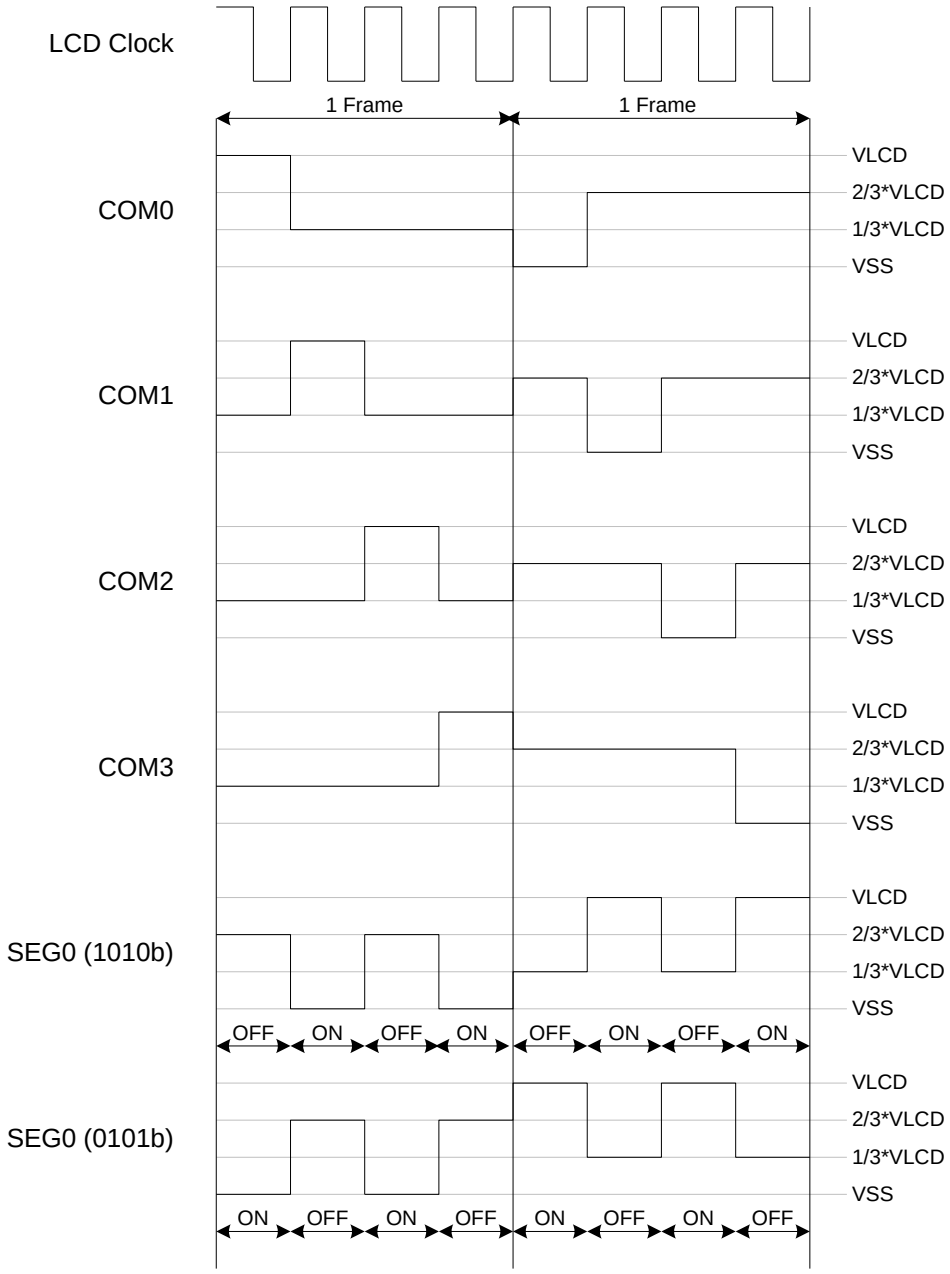
11.2 LCD TIMING

LCD Timing Table

LCDCLK	Clock Source	LCDRATE	LCD Clock	Frame = LCD clock/4	Note
0	Fhosc	X	8MHz / (2^14) = 488Hz	488Hz/4 = 122Hz	IHRC= 8MHz
1	Flosc	0	32kHz /128 = 250Hz	256Hz/4 = 64Hz	ILRC = 32kHz@3.2V
1	Flosc	1	32kHz / 64 = 500Hz	512Hz/4 = 128Hz	



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



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

11.3 LCDM1 REGISTER

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM1	LCDBNK	-	LCDMOD1	LCDMOD0	LCDENB	LCDBIAS	LCDRATE	LCDCLK
R/W	R/W	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	-	1	1	0	0	1	1

Bit0 **LCDCLK:** LCD clock source selection control bit.
0 = LCD clock = Fhosc /64, Frame rate = LCD clock / 4
1 = LCD clock = Flosc /128 or /64. Frame rate = LCD clock / 4

Bit1 **LCDRATE:** LCD clock rate control when LCDCLK=1.
0 = LCD clock rate = Flosc / 128
1 = LCD clock rate = Flosc / 64

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

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

Bit[5:4] **LCDMOD[1:0]:** LCD Mode control bit.
00 = C-Type LCD Mode.
01 = R-Type LCD Mode.
10 = ISP Mode.
11 = LCD Mode All OFF

Bit7 **LCDBNK:** LCD blank control bit.
0 = Normal display.
1 = All of the LCD dots off.

- ***Note1: LCD disable in green or sleep mode, LCDENB is set "0" for power saving.)***
- ***Note2: In C-Type LCD start-up procedure, we recommend to set LCDMOD[1:0]=00 with delay 5ms before LCDENB set "1".***
- ***Note3: If the user wants to set the system into Stop mode, the LCDMOD[1:0] must be set LCD ALL OFF MODE first.***

11.4 LCDM2 REGISTER

08AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM2	VAR1	VAR0	DISQ	**-**	VPPINTL	VCP2	VCP1	VCP0
R/W	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
After Reset	0	1	0	-	0	1	0	0

Bit[2:0] **VCP [2:0]:** LCD Charge pump output voltage

VCP[2:0]	1/3 bias Condition			1/2 bias Condition		
	V2	V3	VLCD	V2	V3	VLCD
000	0.86V	1.73V	2.6V	1.30V	1.30V	2.6V
001	0.9V	1.8V	2.7V	1.35V	1.35V	2.7V
010	0.93V	1.86V	2.8V	1.40V	1.40V	2.8V
011	0.96V	1.93V	2.9V	1.45V	1.45V	2.9V
100	1.00V	2.00V	3.0V	1.50V	1.50V	3.0V
101	1.03V	2.06V	3.1V	1.55V	1.55V	3.1V
110	1.06V	2.13V	3.2V	1.60V	1.60V	3.2V
111	1.10V	2.20V	3.3V	1.65V	1.65V	3.3V

Bit3 **VPPINTL:** Internal VPP Generation control bit.
0 = VLCD is not short to VPP pin internally.
1 = VLCD is short to VPP pin internally for ISP power from internal VLCD

Bit5 **DISQ:** VLCD Pin Discharge Control bit.
0 = VLCD Pin no discharge.
1 = VLCD Pin discharge (After the end of ISP, for discharging VLCD 6.5V down to VDD)

Bit[7:6] **VAR[1:0]:** VLCD pump ripple Control bit.

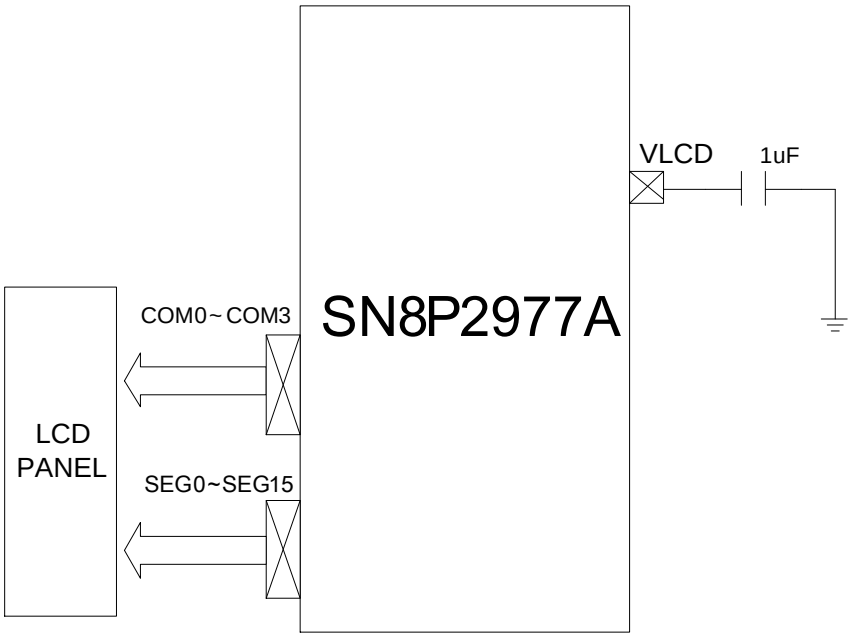
	C-Type	R-Type
VAR[1:0]	**Pump Vripple**	**Power Saving**
00	± 15mV	Disable
01	± 45mV	I
10	± 75mV	II
11	± 105mV	III (Low Power)

☞ In C-Type Mode, Please always set VAR[1:0] = 00.
☞ In R-Type Mode, Power saving level III > II > I > Disable.

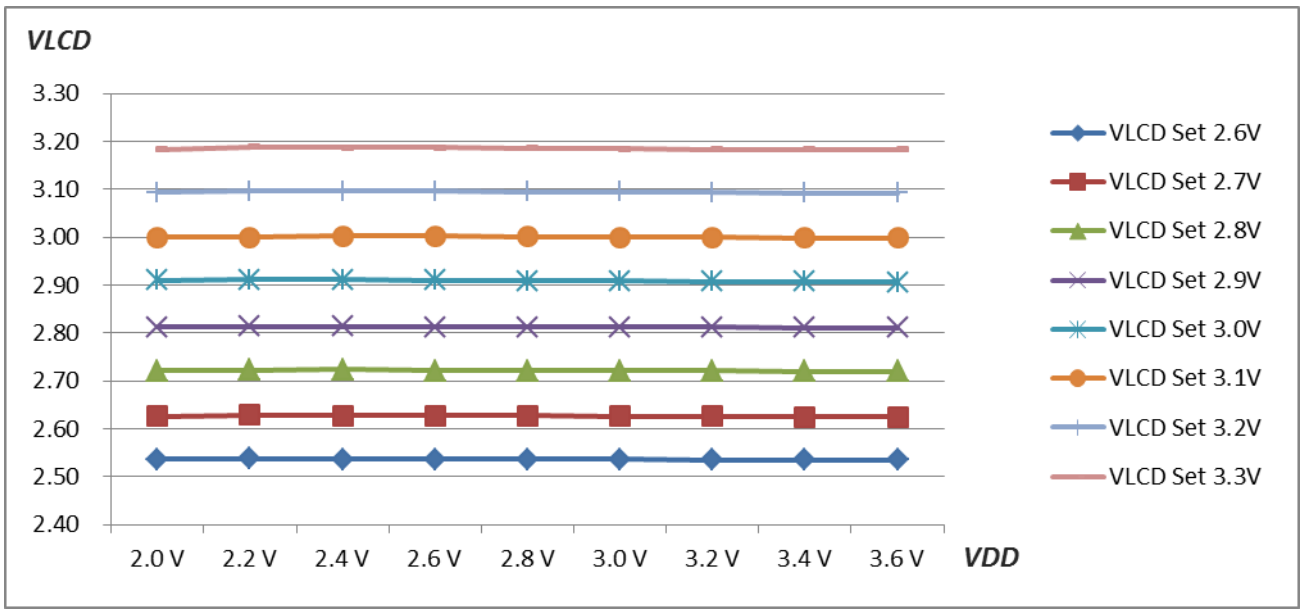
✸ ***Note_1: Macro "RomwrtVpp" instruction cover procedures of internal VPP generation and ROMWRT instruction for ISP function without external 6.5V requirement.***

11.5 C-TYPE LCD DRIVER MODE

The LCD C-type driver mode is support 1/3 and 1/2 bias LCD panel. The LCD power (VLCD) is supplied by internal LCD charge-pump. The C-Type LCD charge-pump voltage level is following VLCD voltage. In C-type LCD mode, the LCDMOD[1:0] bits f LCDM1 register must be “0”. The following are shown the 1/3 and 1/2 bias C-Type LCD application circuit and VLCD output voltage chart.



1/3 and 1/2 Basic C-type LCD Application Circuit

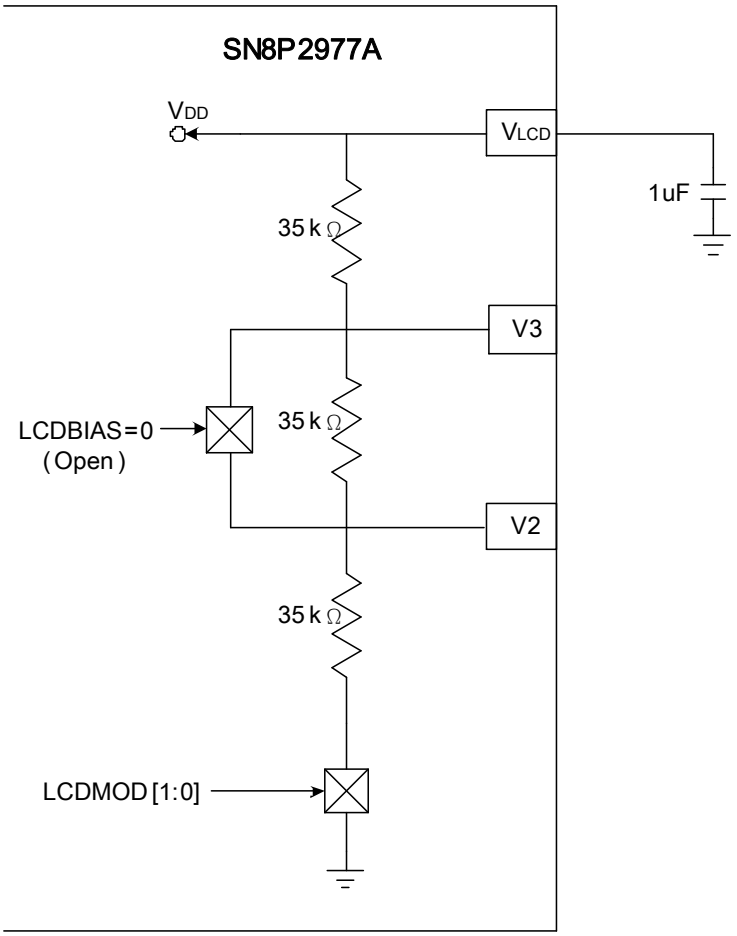


- ***Note1: In C-type mode, a 1uF capacitor is connected to pin VLCD to VSS.***
- ***Note2: VLCD output voltage can be set from 2.6V to 3.3V and with ±0.2V accuracy.***

11.6 R-TYPE LCD DRIVER MODE

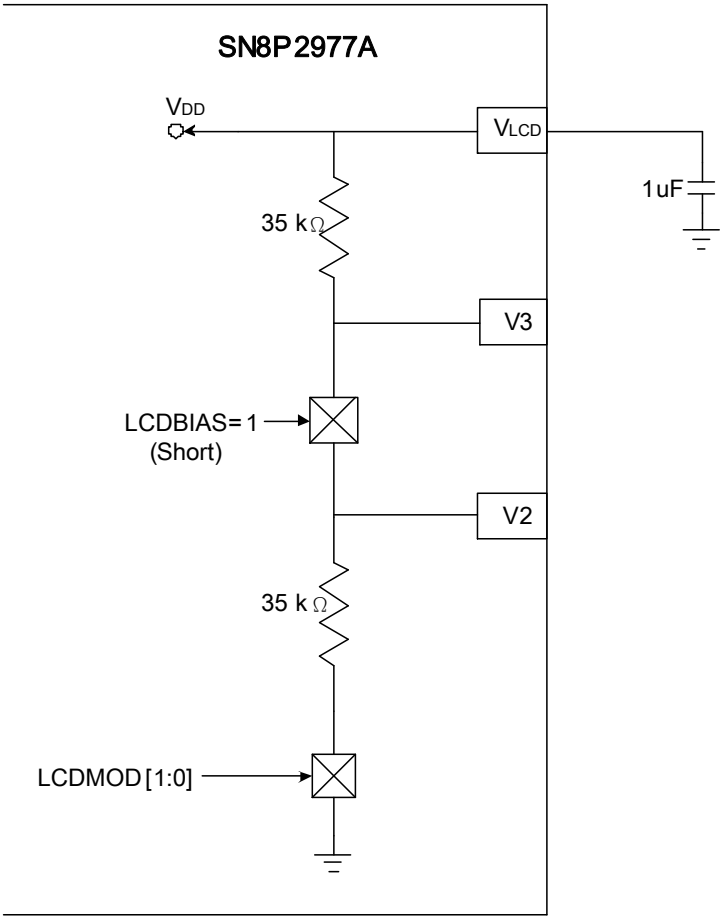
In LCD R-type driver, LCD power (VLCD) is auto connected to VDD via internal circuit. V3 and V2 bias voltage source from internal voltage division by resistors. The following diagram shows the connection of 1/4 duty with 1/3 bias and 1/2 bias.

1/4 duty with 1/3 bias:



R-Type LCD current consumption = $\frac{VLCD}{35k \times 3}$

1/4 duty with 1/2 bias:



$$\text{LCD current consumption} = \frac{VLCD}{35k \times 2}$$

✸ ***Note_1: In R-Type LCD driver mode, VLCD power is auto connected to VDD via internal circuit. Pin VLCD do not connect to any power source.***

✸

11.7 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 15	0FH.0	0FH.1	0FH.2	0FH.3	-	-	-	-

Example: Enable LCD function.

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

R-Type LCD Setting:

```
        B0BSET      FLCDMOD0    ; R-Type LCD.
        B0BCLR      FLCDMOD1    ;
        B0BSET      FLCDENB     ; Enable LCD driver.
```

C-Type LCD Setting:

```
        B0BCLR      FLCDMOD0    ; C-Type LCD.
        B0BCLR      FLCDMOD1    ;
        Delay 5ms               ; Delay time for CP-VLCD output stable.
        B0BSET      FLCDENB     ; Enable LCD driver.
```

12 IN SYSTEM PROGRAM ROM

12.1 OVERVIEW

In-System-Program ROM (ISP ROM), provided user an easy way to storage data into Read-Only-Memory. Choice any ROM address and executing ROM programming instruction – ROMWRT and supply 6.5V voltage on VPP/VLCD pin, after programming time always 200us,ROMDAH/ROMDAL data will be programmed into address ROMADRH / L

12.2 ROMADRH/ROMADRL REGISTER

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMADRH	ROMADR15	-	-	-	ROMADR11	ROMADR10	ROMADR9	ROMADR8
Read/Write	R/W	-	-	-	R/W	R/W	R/W	R/W
After reset	0	-	-	-	0	0	0	0

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMADRL	ROMADR7	ROMADR6	ROMADR5	ROMADR4	ROMADR3	ROMADR2	ROMADR1	ROMADR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

ROMADR[15] : ISP ROM Programming selection control bit. (Always set “0”)
0: User ROM.
1: Information ROM.

ROMADR[11:0] : ISP ROM Programming Address.
ROM Address which will be Programmed

12.3 ROMDAH/ROMADL REGISTERS

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMDAH	ROMDA15	ROMDA14	ROMDA13	ROMDA12	ROMDA11	ROMDA10	ROMDA9	ROMDA8
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMDAL	ROMDA7	ROMDA6	ROMDA5	ROMDA4	ROMDA3	ROMDA2	ROMDA1	ROMDA0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

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

12.4 ISP ROM ROUTINE EXAMPLE

Example : ISP Example with Internal VPP generation:

- ISP ROM Address from 0x0700H to 0x0703H (4-words)
- Using Macro “RomwrtVpp”.

```
Start:
                    MOV             A, #32
                    B0MOV           ISP_Cnt,A
@ISP_WRITE:
;====================== Set ISP Address ======================
                ---------- Start Address = 0x01F0h------------
                    MOV             A, #0F0H
                    B0MOV           ROMADRL,A       ;Move Low Byte Address to ROMADRL
                    MOV             A,#01H
                    B0MOV           ROMADRH,A       ;Move High Byte Address to ROMADRH
;====================== Load data for ISP ======================
@@:                 MOV             A, #12H         Data = 0x1234
                    B0MOV           ROMDAH,A
                    MOV             A, #34H
                    B0MOV           ROMDAL,A
;=============== ISP Write, Using Macro “RomwrtVpp ===============
                    B0BCLR          FGIE            ; Disable Interrupt.
                    RomwrtVpp                       ; ISP ROM Write Macro Instruction
                    B0BSET          FGIE            ; Enable Interrupt if necessary.
                    Incms           ROMADRL         ;ISP Address increase
                    jmp             $+2
                    incms           ROMADRH
                    nop

                    decms           ISP_Cnt         ;Data counter
                    jmp             @b
ISP_End:
                    jmp             $               End of ISP
```

13 Regulator, PGIA and ADC

13.1 OVERVIEW

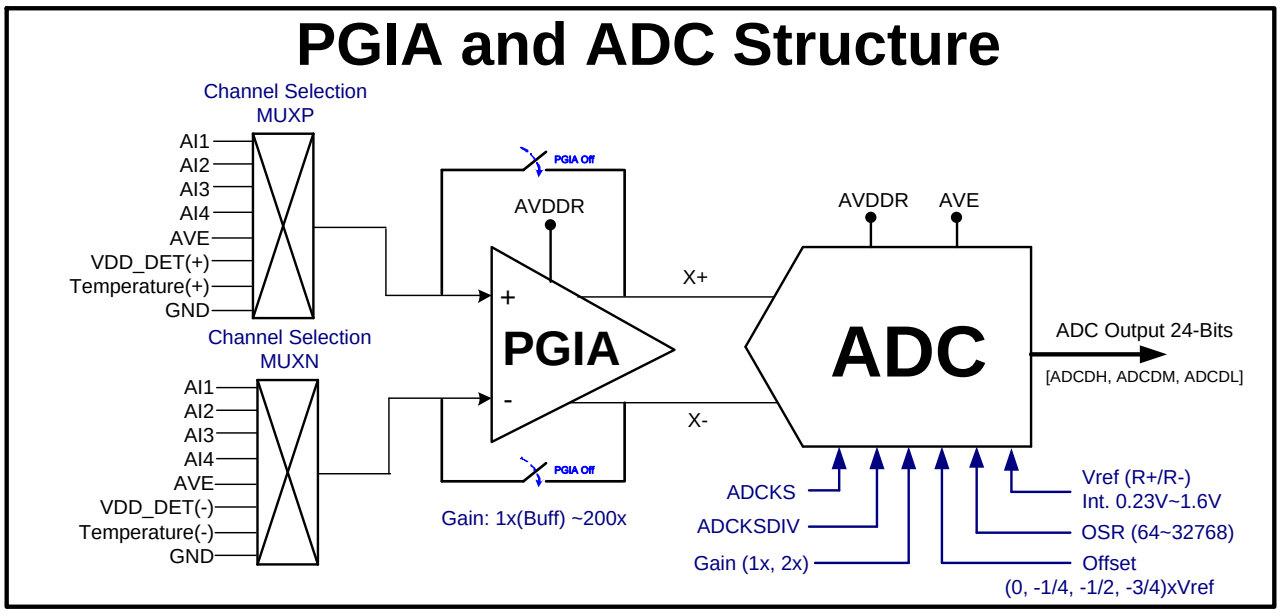
The SN8P2977A has a built-in Voltage Regulator to support a stable voltage 2.4V/2.8V/3.2V from pin AVDDR and 0.75V/1.0V/1.5V/2.0V from pin AVE+ with maximum 10mA current driving capacity. The AVDDR provides stable voltage for internal circuits (PGIA, ADC) and external sensor (load cell or thermistor).

The SN8P2977A series also integrated Δ Σ Analog-to-Digital Converters (ADC) to achieve 24-bit performance and up to 2^24-step resolution. The ADC builds in internal Gain Option with selective range of x1 and x2 for additional signal amplification expect PGIA. The PGIA provides of input channel mode: One fully differential input. 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, 4x, 8x, 16x, 32x, 64x, 128x and 200x in the ADC to accommodate these applications.

13.2 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 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 IRVS bits (Reference Voltage Selection) in ADCM1 register and GS[2:0] bits (Gain Selection) in AMPM2 register.



Block Diagram of ADC module

13.3 Voltage Regulator

SN8P2977A is built in voltage regulators, which can provide a stable 2.4V/2.8V/3.2V (pin AVDDR) and 0.75V/1.0V1.5V/2.0V (pin AVE+) with maximum 10mA current driving capacity. Register VREG can enable or disable AVDDR and AVE output voltage. Because the power of PGIA and ADC are came from AVDDR, turn on AVDDR (AVDDRENB = 1) first before enabling PGIA and ADC. The AVDDR voltage was regulated from VDD.

13.3.1 Voltage Regulator Control Register

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VREG	BGRENB	-	AVENB	AVESEL1	AVESEL0	AVDDRENB	AVDDRSEL1	AVDDRSEL0
R/W	R/W	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	-	0	1	0	0	0	1

Bit7: **BGRENB:** Band Gap Reference voltage enable control bit
0 = Disable Band Gap Reference Voltage
1 = Enable Band Gap Reference Voltage

Bit5: **AVENB:** Regulator (AVE) voltage Enable control bit.
0 = Disable Regulator and AVE Output voltage.
1 = Enable Regulator and AVE Output voltage.
Note: ※When AVE enable & AVE set 1.5V/2.0V, AVE pin have to connect 1uf capacitors to ground.

Bit[4:3]: **AVESEL[1:0]:** Regulator (AVE) voltage output voltage control bit.
00 = AVE 0.75V. (Sink only)
01 = AVE 1.0V. (Sink only)
10 = AVE 1.5V. (Drive only)
11 = AVE 2.0V. (Drive only)

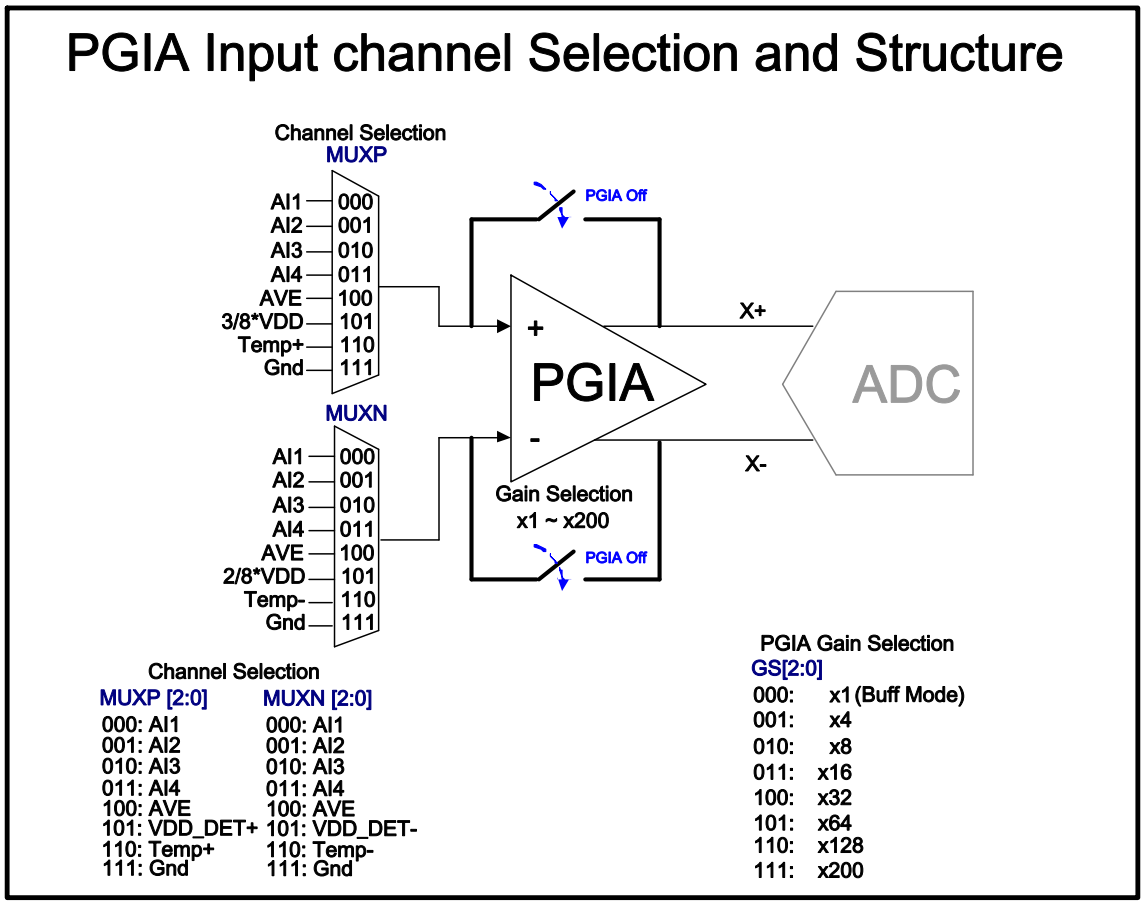
Bit2: **AVDDRENB:** Regulator (AVDDR) voltage Enable control bit.
0 = Disable Regulator and AVDDR Output voltage.
1 = Enable Regulator and AVDDR Output voltage.

Bit[1:0]: **AVDDRSEL[1:0]:** Regulator (AVDDR) voltage output voltage control bit.
00 = Reserved.
01 = AVDDR 2.4V.
10 = AVDDR 2.8V.
11 = AVDDR 3.2V.

- ***Note_1: Band Gap Reference voltage must be enable (FBRGEN), before following function accessing:***
  (Reference AMPM1 and AMPM2 register for detail information)
  (1)Regulators of AVDDR.
  (2)PGIA Function.
  (3)ADC Function.
  (4)Low Battery Detection Function.
- ***Note_2: All current consumptions from AVE+.(ex. Load cell or AVDDR will NOT double.)***
- ***Note_3: PGIA can work in Normal, Slow or Green Mode, when high clock is still running (STPHX=0).***
- ***Note_4: Add 10ms delay time after enabling each regulators, AVDDR/AVE to avoid VDD drop in CR2032 battery application.***

13.4 PGIA -Programmable Gain Instrumentation Amplifier

SN8P2977A includes a low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selection gains of 1x, 4x, 8x, 16x, 32x, 64x, 128x and 200x controlled by register AMPM1. The PGIA also provides two multiplexers. One for positive input channel, the other one is for negative input channel. It was defined by register AMPM1.



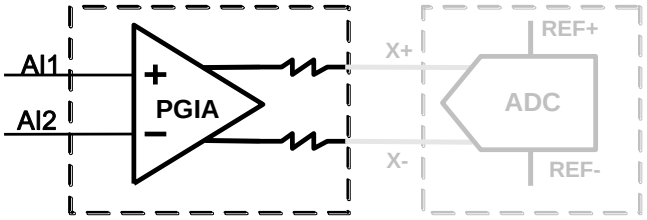
13.4.1 CHS- Analog input signal channel selection Register

091H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CHS	-	MUXP2	MUXP1	MUXP0	-	MUXN2	MUXN1	MUXN0
R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
After Reset	-	0	0	0	-	0	0	1

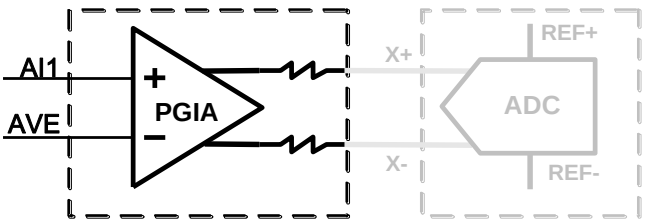
Bit[6:4]: MUXP[2:0]: PGIA Positive input channel selection.
Bit[2:0]: MUXN[2:0]: PGIA Negative input channel selection.

MUXP [2:0]	Positive channel	MUXN [2:0]	Negative channel
000	AI1	000	AI1
001	AI2	001	AI2
010	AI3	010	AI3
011	AI4	011	AI4
100	AVE	100	AVE
101	VDD_DET+	101	VDD_DET-
110	Temperature+	110	Temperature-
111	GND	111	GND

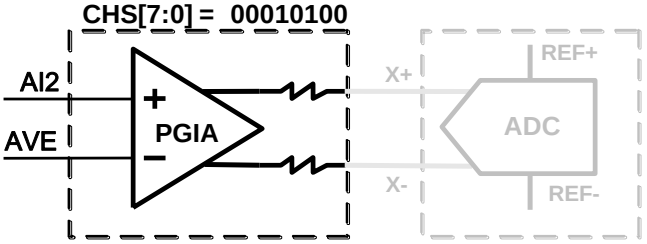
CHS[7:0] = 00000001



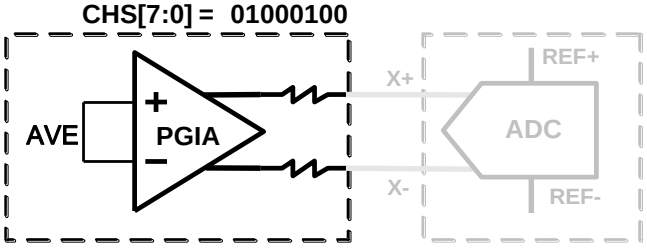
CHS[7:0] = 00000100



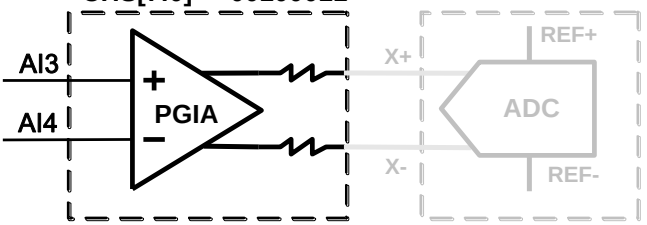
CHS[7:0] = 00010100



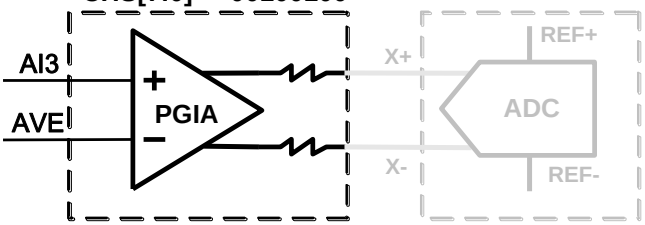
CHS[7:0] = 01000100



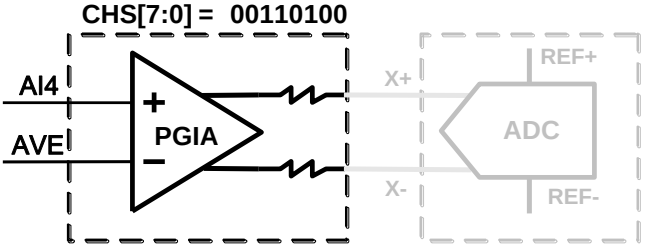
CHS[7:0] = 00100011



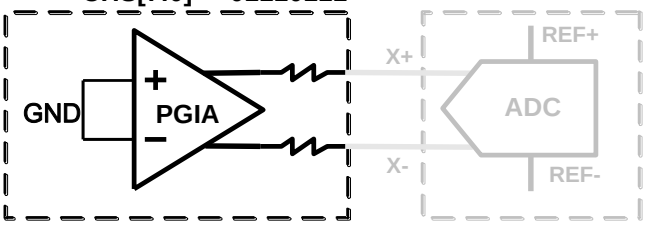
CHS[7:0] = 00100100



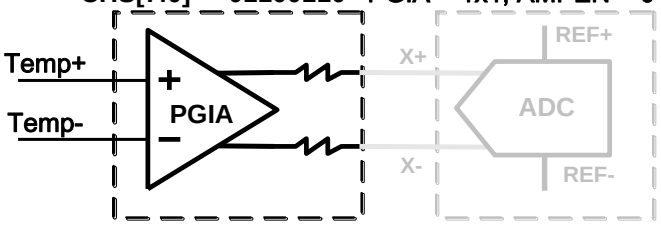
CHS[7:0] = 00110100



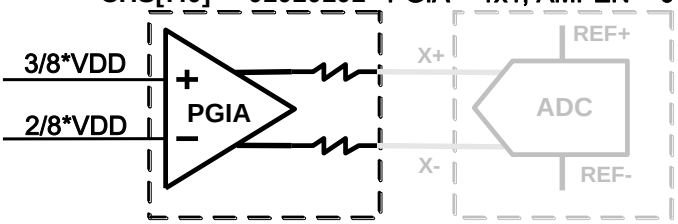
CHS[7:0] = 01110111



CHS[7:0] = 01100110 PGIA = 1x1, AMPEN = 0



CHS[7:0] = 01010101 PGIA = 1x1, AMPEN = 0



- *Note_1: V (AI1, AI2) = (AI1 voltage – AI2 voltage), V (AI3, AI4) = (AI3 voltage – AI4 voltage)*
- *Note_2: V (AVE AVE) = (AVE voltage − AVE voltage). Plaese set AVE 0.75V or 1.0V*
- *Note_3: The purpose of Input-Short mode is only for PGIA offset testing.*
- *Note_4: When CHS[7:0]=01100110 (Temperature detection mode) or CHS[7:0]=01010101 (VDD detection mode), PGIA Gain always set 1x (GS0[2:0]=000) application, the AI+/AI- signal will bypass PGIA and input ADC directly. PGIA can be disabled (AMPENB=0) for power saving.*

13.4.2 AMPM- Amplifier Mode Control Register

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPM	AMPCKS2	AMPCKS1	AMPCKS0	PCHPENB	GS2	GS1	GS0	AMPENB
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	1	0	1	0	0	0	0

Bit[7:5]: **AMPCKS[2:0]:** PGIA chopper selection.(Always set "7.8k Hz")

AMPCKS[1:0]	Chopper Freq.		ADC Clock
000	ADCKS/8	31.25kHz	250 kHz
001	ADCKS/16	15.6kHz	
010	ADCKS/32	7.8kHz	
011	ADCKS/64	3.9kHz	
100	ADCKS/128	1.95kHz	
101	ADCKS/256	0.975kHz	
110	ADCKS/512	0.488kHz	
-	-	-	

Bit4: **PCHPENB:** PGIA Chopper Enable bit.
0 = Disable PGIA Chopper.
1 = Enable PGIA Chopper.

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

GS[2:0]	PGIA GAIN
000	1x PGIA Buff mode @AMPENB=1
001	4x
010	8x
011	16x
100	32x
101	64x
110	128x
111	200x

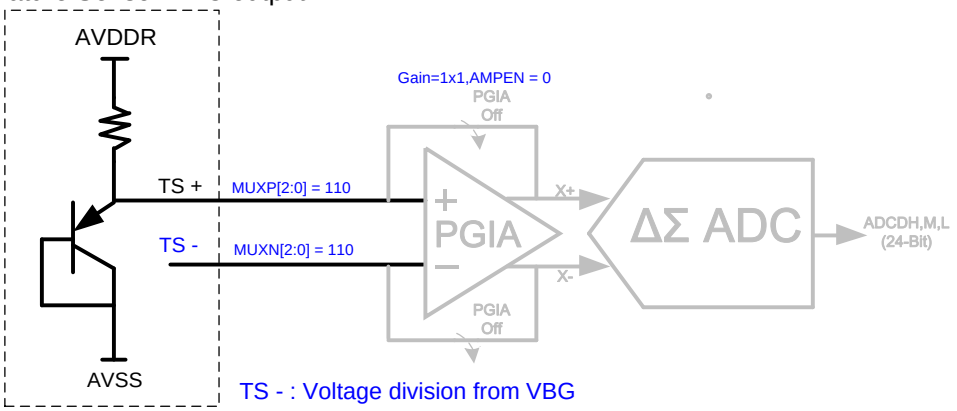
Bit0: **AMPENB:** PGIA function enable control bit.
0 = Disable PGIA function. (PGIA input signal by pass to ADC)
1 = Enable PGIA function.

- ***Note_1: In Temperature detection mode or VDD detection mode, PGIA Gain always set 1x (GS0[2:0]=000) application, the AI+/AI- signal will bypass PGIA and input ADC directly. PGIA can be disabled (AMPENB=0) for power saving.***

- ***Note_2: When PGIA Gain set 1x (GS[2:0]=000) application, the AI+/AI- signal input buffer of PGIA must be enabled (AMPENB=1) for input high impedance characteristic of ADC.***

13.5 Temperature Sensor (TS)

In applications, sensor characteristic might change in different temperature also. To get the temperature information, SN8P2977A build in a temperature senor (TS) for temperature measurement. Select the respective PGIA channel to access the Temperature Sensor ADC output.



Temperature senor setting table:

IRVS[3:0]	Vref source (VBG*2/3)	16Bit-ADC
10xx	0.8V	-65 AD Cnt/℃

- ***Note 1: When selected Temperature Sensor, PGIA gain must set to 1x, or the result will be incorrect.***
- ***Note 2: Under this setting, X+ will be the V(TS) voltage, and X- will be VBG.***
- ***Note 3: The Temperature Sensor was just a reference data not real air temperature. For precision application, please use external thermistor sensor.***

In 25C, V(TS) will be about 1.056V typically, and if temperature rise 10℃, V(TS) will decrease about 17.0mV, if temperature drop 10℃ , V(TS) will increase about 17.0mV.

Example:

Temperature	V(TS)	Vref = (VBG*2/3)	16-Bit ADC output
15℃	V_{TS} Offset + 17mV	0.8V	9635
25℃	V_{TS} Offset		8972
35℃	V_{TS} Offset - 17mV		8309

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

☞ **Example: If temperature 25°C ADC Cnt about 8972, when measured ADC CNT=9635, what's the temperature now?(Slope:-65 Cnt/℃)**

Temperature value = 25℃ + (Measure ADC CNT - 25℃ ADC CNT) / Slope

= 25℃ + (9635 - 8972) / -65

≈ 15℃

- ***Note 1: The V(TS) voltage and temperature curve of each chip might different. Calibration in room temperature is necessary when application temperature sensor.***
- ***Note 2: -1.70mV/C was typical temperature parameter only sensor, every single chip was different to each other.***

Example: PGIA setting (Fhosc = 8MHz IHRC)

```
@CPREG_Init:
            B0BSET      FBGRENB         ; Enable Band Gap Reference voltage.

@AVDDR_Enable:
            B0BSET      FAVDDRENB       ; Enable AVDDR Voltage=2.4V

@PGIA_Setting:
            MOV         A, #00000001B
            B0MOV       CHS, A          ; V (X+, X-) Output = V (AI1, AI2)
            MOV         A, #00011100B   ; PGIA chopper Enable
            B0MOV       AMPM, A         ; PGIA chopper Freq. 31.25kHz.

@PGIA_Enable:   B0BSET      FAMPENB         ; Enable PGIA function
```

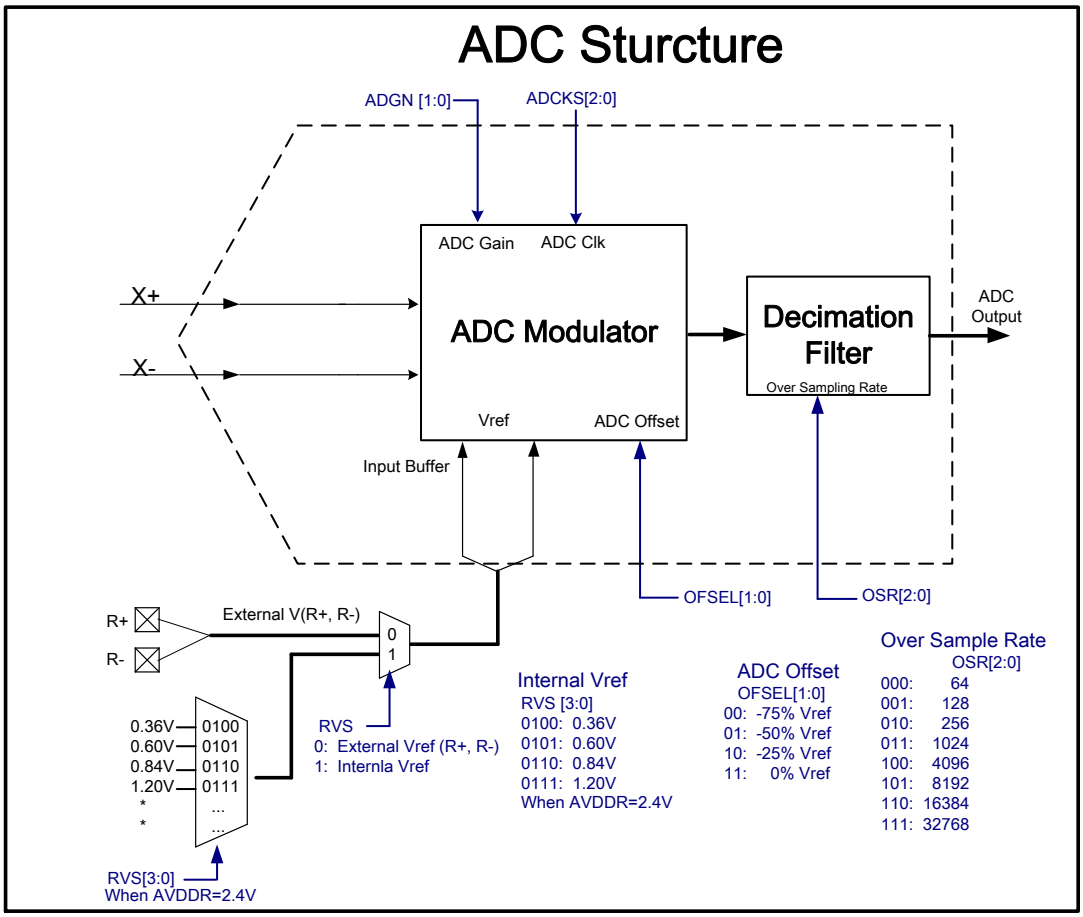
- ***Note_1: Enable AVDDR Regulator before PGIA working***

Example: PGIA channel change:

```
@PGIA_Setting:
            MOV         A, #00000001B
            B0MOV       CHS, A          ; V (X+, X-) Output = V (AI1, AI2)
            MOV         A, #00011100B   ; PGIA chopper Enable, Gain= 128x
            B0MOV       AMPM, A         ; PGIA chopper Freq. 31.25kHz.

@PGIA_Enable:   B0BSET      FAMPENB         ; Enable PGIA function
            …
            …
@PGIA_single-end:
            MOV         A, #00000100B   ; Don't Disable PGIA when change PGIA Channel.
            B0MOV       CHS, A          ; Selected PGIA as Single-end channel
            …                           ; V (X+, X-) Output = V(AI+,AVE)
            …
@PGIA_VDD:
            MOV         A, #01010101B   ; Don't Disable PGIA when change PGIA Channel. PGIA Gain 1x
            B0MOV       CHS, A          ; Selected PGIA as Voltage detection channel
            MOV         A, #00010000B   ; PGIA chopper Enable, Gain= 1x
            B0MOV       AMPM, A         ; PGIA chopper Freq. 31.25kHz.
            B0BCLR      AMPENB          ; Select VDD Voltage Detect function.
            …                           ; V (X+, X-) Output = 1/8*VDD x 1
            …
```

13.6 24-Bit Analog to Digital Converter (ADC)

The SN8P2977A integrated a 24-bit ΔΣ Analog-to-Digital Converters (ADC) with decimation filters can be set for variable throughputs range from 7.6 Hz up to 5.2 kHz. A reference voltage (Vref) is built in internal with selective range from 0.36V to 1.2V in AVDDR=2.4V condition, or an external reference voltage can be used to adjust an adequate range via differential voltage between input pins of R+ and R-. The on-chip input buffers can be used to provide high input impedance for direct connection to sensitive transducers. The ADC builds in internal Gain Option with selective range of x1 and x2 for additional signal amplification expect PGIA.



13.6.1 Analog Inputs and Voltage Operation Range

There are six analog inputs for ADC and PGIA operation, including pins of AI1, AI2, AI3, AI4,R+ and R-. The analog inputs of PGIA, AI1/AI2-, AI3/AI4, are connected to external sensor's output signal, which can be configured as differential mode (AI1 to AI2) or single-end (AI1 to AVE). External Vref for ADC is decided by differential voltage of input pin R+ and R-. All of analog inputs are restricted in absolute voltage range between 0.4V to AVDDR-1V (@AVDDR=2.4V). Moreover, the output signals of PGIA, X+/X-, must also remain within the absolute voltage range.

13.6.2 Reference Voltage

There are two reference voltage (Vref) sources option for ADC operation. One is from internal Vref another is from external Vref. The ADC"s Vref is selected using IRVS[3:0] bits in register ADCM1. Internal Vref source which can be selected value from 0.225V~1.6V setting IRVS[3:0] bits. When IRVS3 & IRVS2 bit is setting to „1", the Vref is from external and the value is decided by differential voltage between Pins of R+ and R-. Detail setting reference register ADCM1

13.6.3 ADC Gain and Offset

The ADC builds in internal Gain Option with selective range of x1 and x2 for additional signal amplification expect PGIA. The ADC Gain setting is controlled by **ADGN [2:0]** bits in register **ADCM1**. The analog signal after ADC Gain amplification, it can be adjusted offset level by subtraction or addition function, to increase the signal operation range of ADC in weigh-scales application. ADC Offset function is controlled by **OFSEL [1:0]** bits in register **ADMC2.**

The following shows ADC output code calculation (differential mode):

16bits: $$\frac{[(AI+)-(AI-)]\times PGIA\times ADC\_Gain+V_{Offset}}{Vref}\times 2^{(16-1)} = +32767 \sim -32768$$

18bits: $$\frac{[(AI+)-(AI-)]\times PGIA\times ADC\_Gain+V_{Offset}}{Vref}\times 2^{(18-1)} = +131071 \sim -131072$$

20bits: $$\frac{[(AI+)-(AI-)]\times PGIA\times ADC\_Gain+V_{Offset}}{Vref}\times 2^{(20-1)} = +524287 \sim -524288$$

Voffset: 0, -1/4, -1/2 or -3/4 x Vref
PGIA: 1x ~ 200x
ADC_Gain: 1x and 2x
Vref Source: Internal Vref
Vref Range: 0.225V ~ 1.6V

13.6.4 Output Word Rate

Delta-Sigma ADC provides variable output word rate (WR) from 7.6 Hz up to 5.2 kHz, which output word rate is decided by setting bits of **ADCKS** and **OSR [2:0].** The ADC output code with slow output word rate is more stable than fast one. In ADC's application, that should be tradeoff between ADC's output word rate and stability (ENOB). The following table shows the ADC output word rate with setting:

ADC Output Word Rate Table:

ADCKS	OSR [2:0]	ADC clock	WR	ADCKS	OSR [2:0]	ADC clock	WR
0	000	250KHz	3.9 kHz	1	000	333KHz	5.2KHz
0	001		1.95 kHz	1	001		2.6KHz
0	010		976 Hz	1	010		1.3KHz
0	011		244 Hz	1	011		325Hz
0	100		61 Hz	1	100		81Hz
0	101		30.5 Hz	1	101		40Hz
0	110		15.2 Hz	1	110		20Hz
0	111		7.6 Hz	1	111		10Hz

13.6.5 ADCM1- ADC Mode1 Register

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCM1	IRVS3	IRVS2	IRVS1	IRVS0	ACHPENB	-	ADGN	ADCENB
R/W	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
After Reset	0	1	1	0	1	-	0	0

Bit[7:4]: IRVS[3:0]: ADC Internal Reference Voltage Selection.

IRVS[3:0]	V_{ref} source		IRVS[3:0]	V_{ref} source		
	AVE= 1.5V	AVE= 2.0V		AVDDR= 2.4V	AVDDR= 2.8V	AVDDR= 3.2V
0000	0.225V	0.3V	0100	0.36V	0.42V	0.48V
0001	0.375V	0.5V	0101	0.60V	0.70V	0.80V
0010	0.525V	0.7V	0110	0.84V	0.98V	1.12V
0011	0.750V	1.0V	0111	1.20V	1.40V	1.60V
IRVS[3:0]	V_{ref} source		IRVS[3:0]	V_{ref} source		
	VBG=1.2V			External Input		
10xx	0.8V		11xx	R+/R-		

Note: (1) Vref source from AVE (1.5V/2.0V), AVE pin connect 1uf capacitors to ground.
(2) When CHS=0x66h Vref voltage have to set IRVS[3:0]=10xx.

Bit3 ACHPENB: ADC Chopper Control bit. (Always set "1")
0: ADC Chopper Disable.
1: ADC Chopper Enable.

Bit1 ADGN: ADC Gain Selection
0 = ADC Gain x 1
1 = ADC Gain x 2

Bit0: ADCENB: ADC function control bit
0 = Disable 24-bit ADC,
1 = Enable 24-bit ADC

- ***Note_1: The Operation range of ADC Reference Voltage (Vref) is from 0.36V to 1.2V@AVDDR=2.4V.***
- ***Note_2: In Temperature detection mode(CHS=0x66) ,The Operation range of ADC Reference Voltage (Vref) is from VBG*2/3= 0.8V (IRVS=10xx).***

13.6.6 ADCM2- ADC Mode2 Register

094H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCM2	ADCKS	OSR2	OSR1	OSR0	-	OFSEL1	OFSEL0	DRDY
R/W	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
After Reset	0	1	1	1	-	1	1	0

Bit7: **ADCKS:** ADC Clock selection Bit.
0 = ADC clock source set 250kHz .
1 = ADC clock source set 333kHz.

Bit[6:4] **OSR [2:0]:** ADC OSR Selection.

OSR [2:0]	OSR
000	64
001	128
010	256
011	1024
100	4096
101	8192
110	16384
111	32768

Bit[2:1] **OFSEL[1:0]:** ADC Offset selection.

OFSEL[1:0]	Off set %
00	-75% Vref
01	-50% Vref
10	-25% Vref
11	00% Vref

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

- ***Note 1: ADC Output Word Rate (WR) = ADC Clock / OSR.***
- ***Note 2: Adjust ADC clock (ADCKS) and OSR can get suitable ADC output word rate.***
- ***Note 3: For High resolution application, OSR set maximum value of 32768 recommended.***
- ***Note 4: Clear Bit DRDY after got ADC data or this bit will keep high all the time.***
- ***Note 5: ADC output stable date at the 3rd data after ADC enable. The 1st ADC output data is dummy after 15us later of ADC enable. The 2nd ADC output data is unstable data after 1/WR later of 1st ADC data. The 3rd, 4th , 5th … are stable data after 1/WR later of each.***

13.6.7 ADC Data Register

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCDH	ADCB23	ADCB22	ADCB21	ADCB20	ADCB19	ADCB18	ADCB17	ADCB16
R/W	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCDM	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB09	ADCB08
R/W	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCDL	ADCB07	ADCB06	ADCB05	ADCB04	ADCB03	ADCB02	ADCB01	ADCB00
R/W	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	1	1	1

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

ADC conversion data (2's compliment, Hexadecimal)	***Decimal Value***
0x7FFFFH	524287
…	…
0x40000H	262144
…	…
0x10000H	65536
…	…
0x00002H	2
0x00001H	1
0x00000H	0
0xFFFFFH	-1
0xFFFFEH	-2
…	…
0xF0000H	-65536
…	…
0xC0000H	-262144
…	…
0x80000H	-524288

- ***Note 1: ADCDH [7:0], ADCDM [7:0] and ADCDL [7:4] are read only registers.***
- ***Note 2: For 16-Bit ADC resolution, please use registers of ADCDH and ADCDM (ADCB23~ADCB08).***
  For 18-Bit ADC resolution, please use registers of ADCDH, ADCDM and ADCDL. (ADCB23~ADCB06).
  For 20-Bit ADC resolution, please use registers of ADCDH, ADCDM and ADCDL. (ADCB23~ADCB04).
- ***Note 3: The ADC conversion data is combined with ADCDH, ADCDM, ADCDL in 2's compliment with sign bit numerical format, and Bit ADCB23 is the sign bit of ADC data.***
  ADCB23=0 means data is Positive value, ADCB23=1 means data is Negative value.
- ***Note 4: The Positive Full-Scale-Output value of ADC conversion is 0x7FFFF.***
- ***Note 5: The Negative Full-Scale-Output value of ADC conversion is 0x80000H.***
- ***Note 6: Because of the ADC design limitation, the ADC Linear range is +29491 ~ -29491 (16-bit). (+0.9*Vref ~ - 0.9*Vref). The MAX ADC output must keep inside this range.***

Following table shows the Noise and ENOB (RMS and peak-to-peak) of the SN8P2977A ADC with different output word rate rates and gain settings. These numbers are typical and are generated using a differential input-short condition, ADC Vref 0.84V and 1024-data of measurement.

Gain (PGIA x ADC)	OSR	WR	Noise Free Resolution (1)	Effective Resolution (2)	Pk-Pk Noise(4)	RMS Noise(3)
1 x 1(buff off)	32768	7.6Hz	18.4	21.1	4.857uV	0.736uV
1 x 1(buff on)	32768	7.6Hz	18.0	20.7	6.409uV	0.971uV
16 x 1	32768	7.6Hz	17.7	20.4	0.493uV	0.075uV
32 x 1	32768	7.6Hz	17.2	19.9	0.349uV	0.053uV
64 x 1	32768	7.6Hz	16.8	19.5	0.230uV	0.035uV
64 x 2	32768	7.6Hz	15.8	18.5	0.230uV	0.035uV
128 x 1	32768	7.6Hz	15.9	18.6	0.215uV	0.033uV
128 x 2	32768	7.6Hz	15	17.7	0.200uV	0.030uV
200 x 1	32768	7.6Hz	15.3	18	0.208uV	0.032uV
128 x 1	16384	15.3Hz	15.5	18.2	0.283uV	0.043uV
128 x 1	8192	31Hz	14.9	17.6	0.429uV	0.065uV
128 x 1	4096	61Hz	14.3	17	0.651uV	0.099uV
128 x 1	1024	244Hz	13.4	16.1	1.214uV	0.184uV
128 x 1	256	977 Hz	12.3	15	2.603uV	0.394uV
128 x 1	128	2.0 kHz	11.7	14.4	3.945uV	0.598uV
128 x 1	64	3.9 kHz	11	13.7	6.409uV	0.971uV

All Test condition: ADC 250kHz, Input-Short, Vref=0.84V, Gain = PGIA x ADC, Collect 1024 ADC date.

(1).Noise Free Resolution = Log2 (Full Scale Range / Peak-Peak Noise)

where Full Scale Range = 2 x Vref / Gain (ex. Vref=0.84V, Gain=128x)

(2).Effective Resolution = Log2 (Full Scale Range / RMS_Noise)

(3).RMS Noise = σ x LSB_Resolution

where LSB_Resolution = Full Scale Range / 2^Bit, Bit=20

σ = standard deviation of 1024 ADC output data.

(4). Peak-Peak Noise = 6.6 x RMS Noise, or code variation range x LSB_Resolution

where Code variation range = ADC counts max-min of 1024 data.

Example: Regulator, PGIA and ADC setting (Fhosc = IHRC 8MHz)

```
@CPREG_Init:      B0BSET      FBGRENB             ;Enable Band Gap Reference voltage

@AVDDR_Enable:    B0BSET      FAVDDRENB           ; Enable AVDDR Voltage=2.4V

@PGIA_Init:       MOV         A, #00000001B
                  B0MOV       CHS, A              ; V (X+, X-) Output = V (AI1, AI2) x 128
                  MOV         A, #00011100B       ; PGIA chopper Enable
                  B0MOV       AMPM, A             ; PGIA chopper Freq. 31.25kHz.
                  B0BSET      FAMPENB             ; Enable PGIA function


@ADC_Init:        MOV         A, #01101000B       ; ADC chopper 31.25KHz
                  B0MOV       ADCM1, A            ; ADC Reference voltage = internal 0.84V, ADC_Gain = 1x.
                  MOV         A, #01110110B
                  B0MOV       ADCM2, A            ; Set ADC clock=250kHz, OSR=32768, offset=0V.
                  B0BSET      FADCENB             ; Enable ADC function

@ADC_Wait:        B0BTS1      FDRDY               ; Check ADC output new data or not
                  JMP         @ADC_Wait           ; Wait for Bit DRDY = 1
                                                  ; Output ADC conversion word
@ADC_Read:        B0BCLR      FDRDY
                  B0MOV       A, ADCDH
                  B0MOV       Data_H_Buf, A       ; Move ADC conversion High byte to Data Buffer
                  B0MOV       A, ADCDM
                  B0MOV       Data_M_Buf, A       ; Move ADC conversion Medium byte to Data Buffer
```

- ***Note 1: Please set ADC relative registers first, than enable ADC function bit.***
- ***Note 2: Before enable ADC function, please set analog function (regulators, PGIA and ADC) and wait 300us for all functions stable.***

Example: VDD Voltage Detection:

```
@CPREG_Init:      B0BSET      FBGRENB       ;Enable Band Gap Reference voltage

@AVDDR_Enable:    B0BSET      FAVDDRENB     ; Enable AVDDR Voltage=2.4V

@VDD_Detection:
@PGIA_Init:       MOV         A, #01010101B
                  B0MOV       CHS, A        ; PGIA differential channel (VDD+, VDD-) and Gain x 1
                  MOV         A, #00010000B ; Set VDD detection function
                  B0MOV       AMPM, A       ; Set PGIA buffer off and PGIA chopper 31.25KHz
                  B0BCLR      FAMPENB       ; Disable PGIA function
                                            ; V (X+, X-) Output = V (VDD+, VDD-) x 1
@ADC_Init:        MOV         A, #01101000B ; ADC chopper 31.25KHz
                  B0MOV       ADCM1, A      ; ADC Reference voltage = internal 0.84V, ADC_Gain = 1x.
                  MOV         A, #01110110B
                  B0MOV       ADCM2, A      ; Set ADC clock=250kHz, OSR=32768, offset=0V.
                  B0BSET      FADCENB       ; Enable ADC function

@ADC_Wait:        B0BTS1      FDRDY         ; Check ADC output new data or not
                  JMP         @ADC_Wait     ; Wait for Bit DRDY = 1
@ADC_Read:        B0BCLR      FDRDY         ; Output ADC conversion word
                  B0MOV       A, ADCDH
                  B0MOV       Data_H_Buf, A ; Move ADC conversion High byte to Data Buffer
                  B0MOV       A, ADCDM
                  B0MOV       Data_M_Buf, A ; Move ADC conversion Medium byte to Data Buffer
                  …
                  …
                  …
```

Example: Fast ADC conversion Rate setting

```
@CPREG_Init:      B0BSET      FBGRENB          ;Enable Band Gap Reference voltage

@AVDDR_Enable:    B0BSET      FAVDDRENB        ; Enable AVDDR Voltage=2.4V

@VDD_Detection:
@PGIA_Init:       MOV         A, #01010101B
                  B0MOV       CHS, A           ; PGIA differential channel (VDD+, VDD-) and PGIA Gain x 1
                  MOV         A, #00010000B    ; Set VDD detection function
                  B0MOV       AMPM, A          ; Set PGIA buffer off and PGIA chopper 31.25KHz
                  B0BCLR      FAMPENB          ; Disable PGIA function
                                               ; V (X+, X-) Output = V (VDD+, VDD-) x 1
@ADC_Init:        MOV         A, #01101000B
                  B0MOV       ADCM1, A         ; ADC Reference voltage = internal 0.84V, ADC_Gain = 1x.
                  MOV         A, #00100110B
                  B0MOV       ADCM2, A         ; Set ADC clock=250kHz, OSR=256, offset=0V. WR=~1KHz
                  B0BSET      FADCENB          ; Enable ADC function
                  Call        Wait_500uS       ; Wait 500us for Regulators and analog function stable

@ADC_Wait:        B0BTS1      FDRDY            ; Check ADC output new data or not
                  JMP         @ADC_Wait        ; Wait for Bit DRDY = 1
@ADC_Read:        B0BCLR      FDRDY            ; Output ADC conversion word
                  B0MOV       A, ADCDH
                  B0MOV       Data_H_Buf, A    ; Move ADC conversion High byte to Data Buffer
                  B0MOV       A, ADCDM
                  B0MOV       Data_M_Buf, A    ; Move ADC conversion Medium byte to Data Buffer
```

Example: Green Mode and Sleep Mode setting

```
Green_Mode_1:     // Wakeup By ADC ready, T0 overflow and P0 level change.
                  MOV         A, #11110000
                  B0MOV       T0M, A           ; T0 overflow and wakeup every 512us. (Fcpu = 1MIP)
                  MOV         A, #00100110B
                  B0MOV       ADCM2, A         ; Set ADC clock=250kHz, OSR=256. WR=976Hz
                  B0BSET      FADCENB          ; Enable ADC and ADC wakeup system every 1024us.
                  GreenMode                    ; System into Green mode. (Macro)

Green_Mode_2:     // Wakeup by T0 overflow and P0 level change. (Current 3uA typ.)
                  B0BSET      FLCKMD           ; Into slow mode.
                  B0BSET      FSTPHX           ; Stop High clock (IHRC).
                  MOV         A, #11000000
                  B0MOV       T0M, A           ; T0 overflow and green mode wakeup every 0.5s.
                  B0BCLR      FLBTENB          ; Disable Low battery detect function.
                  B0BCLR      FAMPENB          ; Disable PGIA function.
                  B0BCLR      FADCENB          ; Disable ADC function.
                  B0BCLR      FAVDDRENB        ; Disable AVDDR.
                  B0BSET      FLCDMOD1         ; Set the LCDMOD[1:0] = ALL OFF Mode
                  B0BSET      FLCDMOD0         ; Set the LCDMOD[1:0] = ALL OFF Mode
                  B0BCLR      FLCDENB          ; Disable LCD display.
                  B0BCLR      FBGRENB          ; Disable Band Gap Voltage.
                  GreenMode                    ; System into Green mode. (Macro)
```

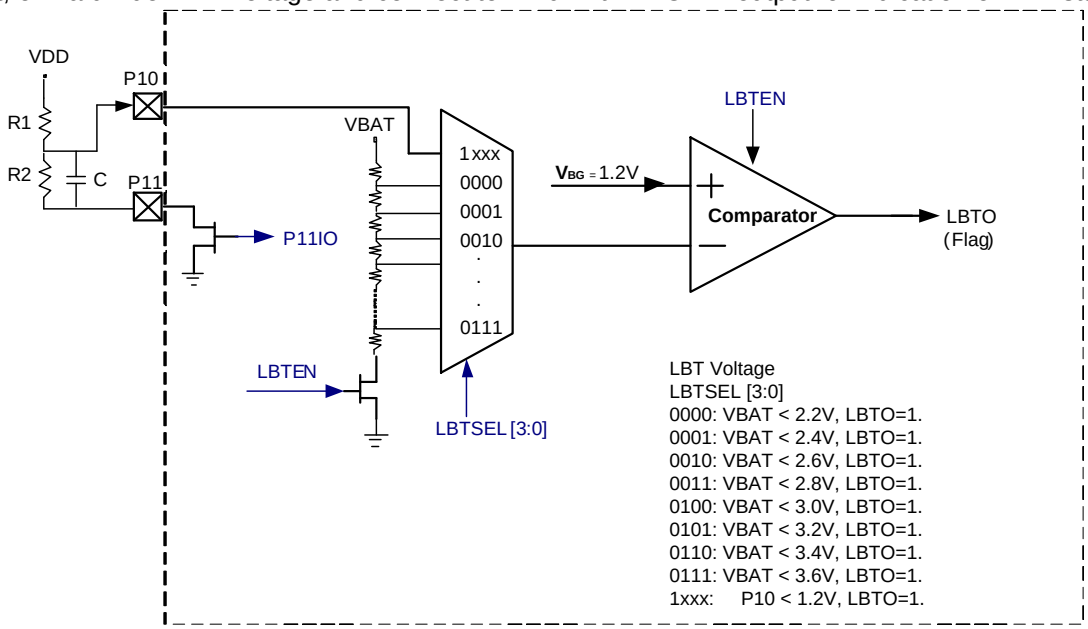
```
Sleep_Mode:     // Wakeup by P0 level change.
                B0BCLR      FLBTENB     ; Disable Low battery detect function.
                B0BCLR      FAMPENB     ; Disable PGIA function.
                B0BCLR      FADCENB     ; Disable ADC function.
                B0BCLR      FAVDDRENB   ; Disable AVDDR.
                B0BSET      FLCDMOD1    ; Set the LCDMOD[1:0] = ALL OFF Mode
                B0BSET      FLCDMOD0    ; Set the LCDMOD[1:0] = ALL OFF Mode
                B0BCLR      FLCDENB     ; Disable LCD display.
                B0BCLR      FBGRENB     ; Disable Band Gap Voltage.
                SleepMode               ; System into Sleep mode. (Macro)
```

LCDMOD1

- ***Note 1: Please set ADC relative registers first before enable ADC function.***
- ***Note 2: Before enable ADC function, please set analog function (regulators, PGIA and ADC) and wait 500us for all functions stable.***
- ***Note 3: In fast ADC conversion application, the third ADC data will available for application.***
- ***Note 4: The second ADC will available after ADC channel switched and in condition of ADC not disable status.***
- ***Note 5: For increasing fast ADC conversion accuracy, recommend averaging several times of ADC row Data for application.***

13.7 LBTM: Low Battery Detect

SN8P2977A provided two different ways to measure VDD Voltage. One is from ADC reference voltage selection. It will be more precise but take more time and a little bit complex. Another way is using build in Voltage Comparator via internal or external input path to detect VDD voltage level. There are four internal level, 2.2V~3.6V can be set for low battery detect, or via divide VDD voltage and connect to P1.0. Bit LBTO will output for indication of LBT status.



13.7.1 LBTM: Low Battery Detect Register

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LBTM	-	P11IO	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTO	LBTENB
R/W	-	R/W	R/W	R/W	R/W	R/W	R	R/W
After Reset	-	0	0	0	0	0	0	0

Bit6: **P11IO:** Port 1.1 Input/LBT function control bit.
0 = Disable P11 as Input Port,
1 = Enable P11 as LBT function

Bit [5:2]: **LBTSL [3:0]:** Low Battery Detect Threshold Voltage.

LBTEN	LBTSL [3:0]	P11IO	LBT Voltage	Note
0	**xxxx**	**X**	**x**	**LBT Disable**
1	**0000**	**0**	**VDD < 2.2V**	**Internal Input**
1	**0001**	**0**	**VDD < 2.4V**	**Internal Input**
1	**0010**	**0**	**VDD < 2.6V**	**Internal Input**
1	**0011**	**0**	**VDD < 2.8V**	**Internal Input**
1	**0100**	**0**	**VDD < 3.0V**	**Internal Input**
1	**0101**	**0**	**VDD < 3.2V**	**Internal Input**
1	**0110**	**0**	**VDD < 3.4V**	**Internal Input**
1	**0111**	**0**	**VDD < 3.6V**	**Internal Input**
1	**1xxx**	**1**	**P10 < 1.2V, LBTO=1**	**External Input**

Bit1: **LBTO:** Low Battery Detect Output Bit.
0 = VDD voltage Higher than Set LBT Voltage.
1 = VDD voltage Lower than Set LBT Voltage.
(Note: After getting LBTO data, Please set LBTEN = 0 to disable LBT Low Battery Detect function.)

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

External Input (P10) LBT functions: (Not available in ICE Emulation)

Low Battery Voltage	R1	R2	LBTO=1
2.3V	470kΩ	500kΩ	VDD<2.3V
2.8V	620kΩ	450kΩ	VDD<2.8V

- ***Note_1: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.***
- ***Note_2: Before enable LBT function, please set LBT function and wait 50us for all functions stable.***
- ***Note_3: After getting LBTO data,Please set LBTENB = 0 to disable LBT Low Battery Detect function.***
- ***Note_4: LBT external input P10 and P11IO function is not available in ICE emulation.***
- ***Note_5: IO input voltage must keep lower than VDD.***

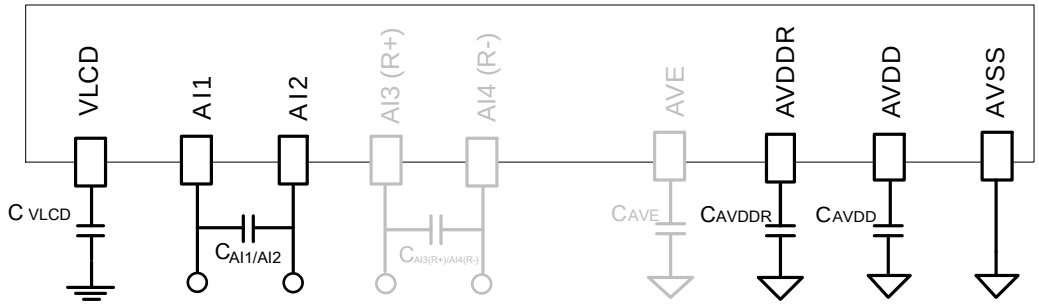
13.8 Analog Setting and Application

The most applications of SN8P2977A were for DC measurement ex. Weight scale, Pressure measure. Following table indicate different applications setting which MCU power source came from CR2032 battery, AA/AAA dry battery or external Regulator.

Setting (AI1, AI2) = (AI1 voltage – AI2 voltage) & Internal Vref source from AVDDR Capacitor Table:

Power type	AI1/AI2	AI3(R+)/AI4(R-)	AVDDR	AVE	VLCD	AVDD	DVDD
	$C_{AI1/AI2}$	$C_{AI3(R+)/AI4(R-)}$	C_{AVDDR}	C_{AVE}	C_{VLCD}	C_{AVDD}	C_{DVDD}
CR2032 (2.4~3V)	0.1uF	NA	1uF	NA	1uF	10uF, 0.1uF	10uF, 0.1uF
AA/AAA Bat.(2.4~3V)	0.1uF	NA	1uF	NA	1uF	10uF, 0.1uF	10uF, 0.1uF

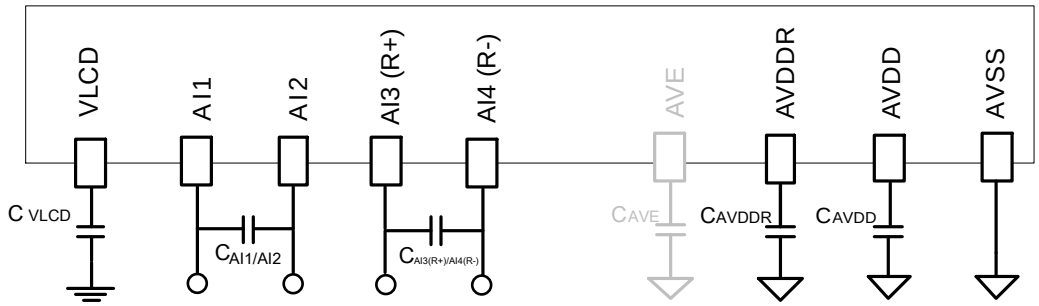
※Note: ※When AVE disable AVE pin can not connect 1uf capacitors to ground.



Setting (AI1, AI2) = (AI1 voltage – AI2 voltage) & External Vref source from AVDDR Capacitor Table:

Power type	AI1/AI2	AI3(R+)/AI4(R-)	AVDDR	AVE	VLCD	AVDD	DVDD
	$C_{AI1/AI2}$	$C_{AI3(R+)/AI4(R-)}$	C_{AVDDR}	C_{AVE}	C_{VLCD}	C_{AVDD}	C_{DVDD}
CR2032 (2.4~3V)	0.1uF	0.1uF	1uF	NA	1uF	10uF, 0.1uF	10uF, 0.1uF
AA/AAA Bat.(2.4~3V)	0.1uF	0.1uF	1uF	NA	1uF	10uF, 0.1uF	10uF, 0.1uF

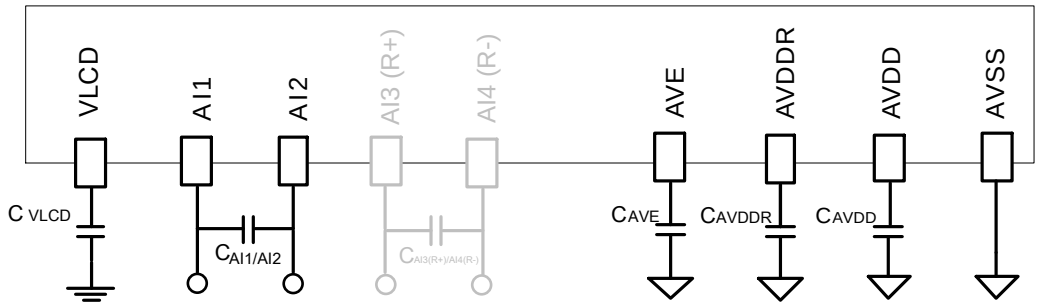
※Note: ※When AVE disable AVE pin can not connect 1uf capacitors to ground.



Setting (AI1, AI2) = (AI1 voltage – AI2 voltage) & Internal Vref source from AVE Capacitor Table:

Power type	AI1/AI2	AI3(R+)/AI4(R-)	AVDDR	AVE	VLCD	AVDD	DVDD
	$C_{AI1/AI2}$	$C_{AI3(R+)/AI4(R-)}$	C_{AVDDR}	C_{AVE}	C_{VLCD}	C_{AVDD}	C_{DVDD}
CR2032 (2.4~3V)	0.1uF	NA	1uF	1uF	1uF	10uF, 0.1uF	10uF, 0.1uF
AA/AAA Bat.(2.4~3V)	0.1uF	NA	1uF	1uF	1uF	10uF, 0.1uF	10uF, 0.1uF

※Note: ※When AVE(2V or 1.5V) enable AVE pin can connect 1uf capacitors to ground.

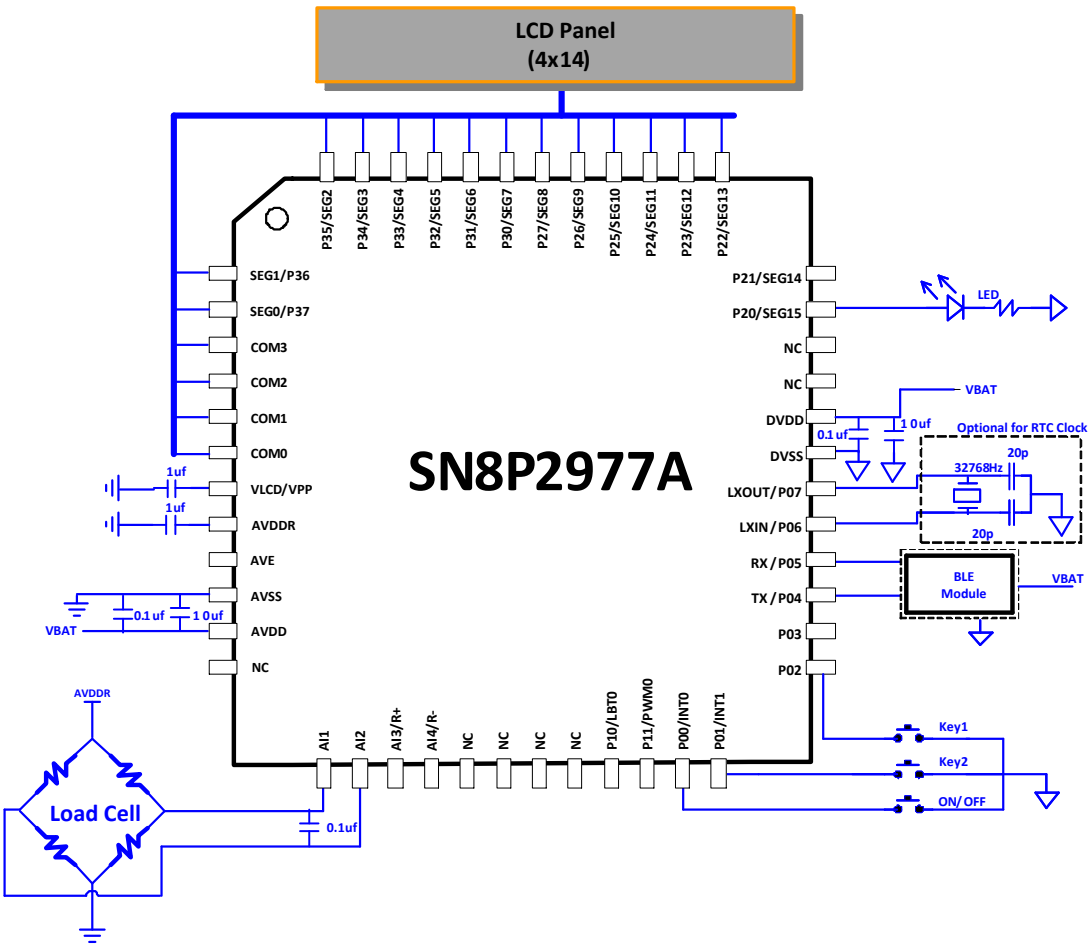


- ***Note_1: AVDDR connect 1uf capacitors, the maximum output current will be limited 5mA typically.***
- ***Note_2: AVE connect 1uf capacitors, the maximum output current will be limited 5mA typically.***
- ***Note_3: When MCU VDD power sources from AA/AAA dry battery directly, not via other LDO, 1uf capacitor can be applied to VDD and without VDD drop when regulator turns on at low battery status.***
- ***Note_4:When AVE=1V or 0.75V,If user care about RC time constant, it is recommended to replace 0.1uF capacitor and connect to AVDDR.***

14 APPLICATION CIRCUIT

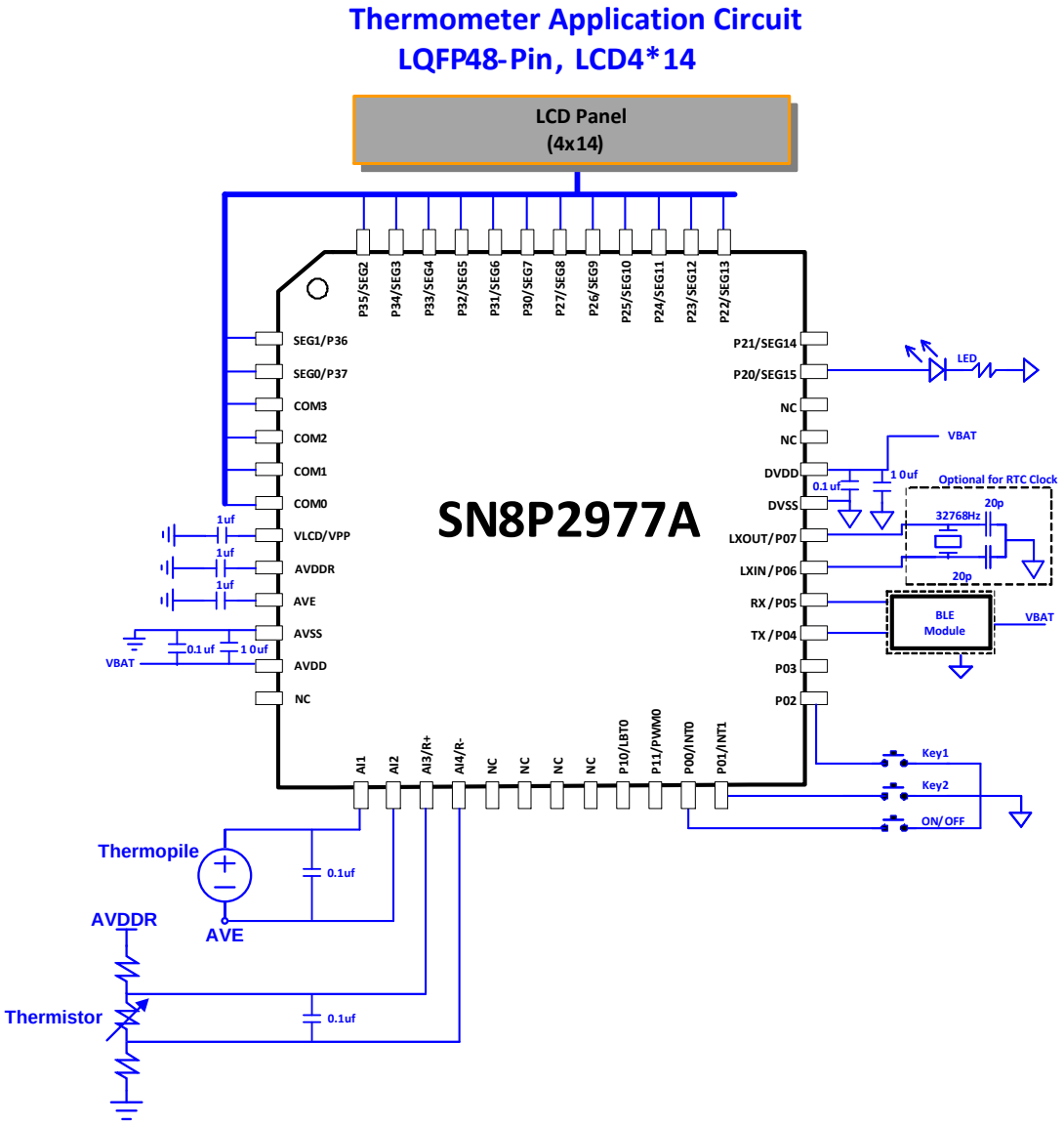
14.1 Scale (Load Cell) Application Circuit

Weight Scale Application Circuit
LQFP48-Pin, LCD4*14



✸ ***Note : DVDD/AVDD capacitors should be as close as possible with pins of IC***

14.2 Thermometer application circuit



✸ ***Note : DVDD/AVDD capacitors should be as close as possible with pins of IC***

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

Note: 1. The “M” is memory including system registers and user defined memory.
Note: 2. If M is system register, the "N" is zero, or the "N" = 1.
Note: 3. If the branch condition is true, the "S=1", or the "S=0".

16 Development Tools

16.1 Development Tool Version

16.1.1 ICE (In circuit emulation)

- **SN8ICE2K Plus II:** Full function emulates SN8P2977A series

✷ **SN8ICE2K ICE emulation notice**
- **Operation voltage of ICE: 3.3V.**
- **Recommend maximum emulation speed at 3.3V: 2 MIPS (e.g. Fcpu = Fhosc/4).**
- **Use SN8P2977A EV-KIT to emulation Analog Function.**
- **Note: S8ICE1K doesn't support SN8P2977A serial emulation.**

16.1.2 OTP Writer

MP Pro Writer：ON/OFF line operation to support SN8P2977A mass production.

✷ **Note: MPIII Writer doesn't support SN8P2977A OTP programming.**

16.1.3 IDE (Integrated Development Environment)

SONiX 8-bit MCU integrated development environment include Assembler, ICE debugger and OTP writer software.
- SN8ICE 2K Plus II
- Easy Writer, MP-Easy and MPIII Writer doesn't support SN8P2977A

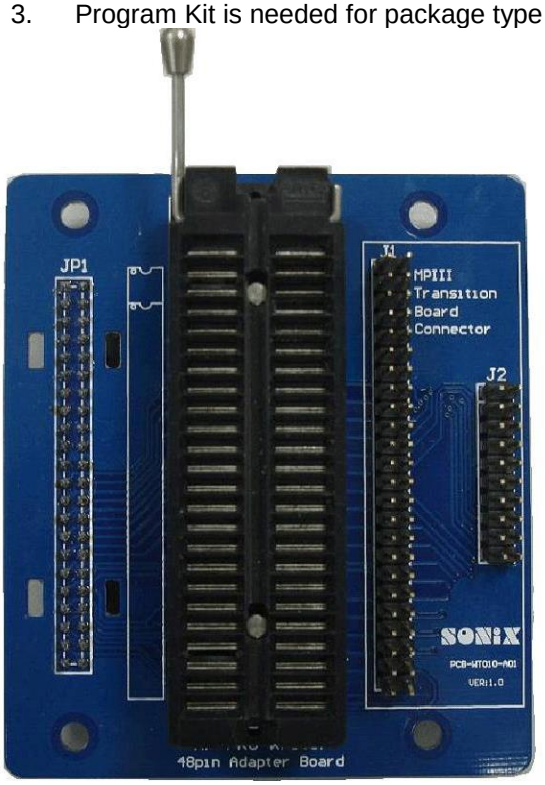
16.2 OTP Programming Pin to Transition Board Mapping

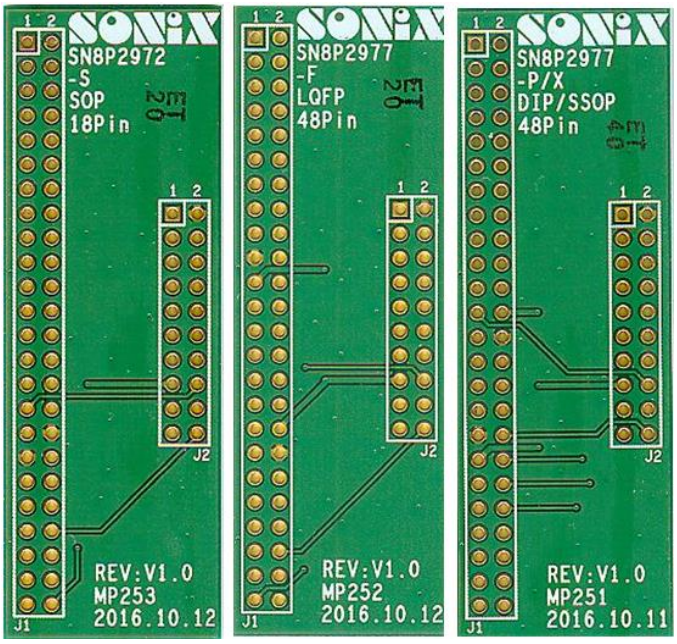
SN8P2977A COB Programming Pin Mapping:

OTP Programming Pin of SN8P2977A Series				
MP PRO Writer		SN8P2977A		
JP3 of 40 PIN Adapter Board		Pin Assignment		
Number	Pin Name	Pad Name	DIP48 PIN Number	LQFP48 PIN Number
1	VDD	DVDD/AVDD	37/18	32/11
2	GND	DVSS/AVSS	36/17	31/10
3	CLK / PGCLK	P0.1	29	24
4	CE	-	-	-
5	PGM / OTPCLK	P0.2	30	25
6	OE / ShiftData	P0.3	31	26
7	D1	-	-	-
8	D0	-	-	-
9	D3	-	-	-
10	D2	-	-	-
11	D5	-	-	-
12	D4	-	-	-
13	D7	-	-	-
14	D6	-	-	-
15	VDD	DVDD/AVDD	37/18	32/11
16	VPP	VLCD	12	7
17	HLS	-	-	-
18	RST	-	-	-
19	-	-	-	-
20	ALSB/PDB	P0.4	32	27

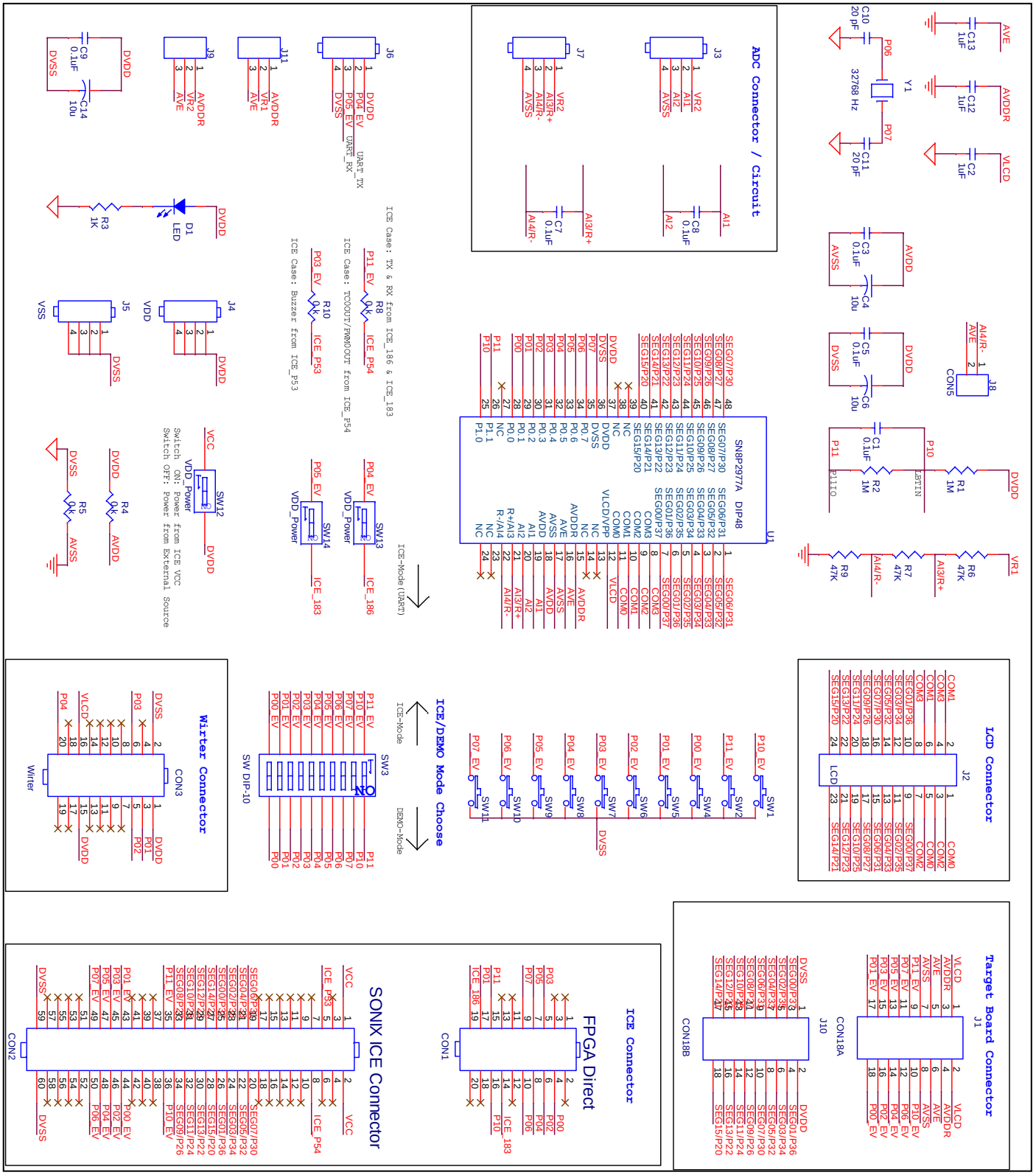
SN8P2977A Package Type Programming with 48 PIN adapter board and with transition board socket:

1. 48 PIN adapter board connect to MP PRO Writer.
2. Plug in the transition board socket, MP251 or MP252 or MP253, on the adapter board (J1/J2).
3. Program Kit is needed for package type of DIP48 and LQFP48 IC.





16.3 APPENDIX A: EV-KIT BOARD CIRCUIT



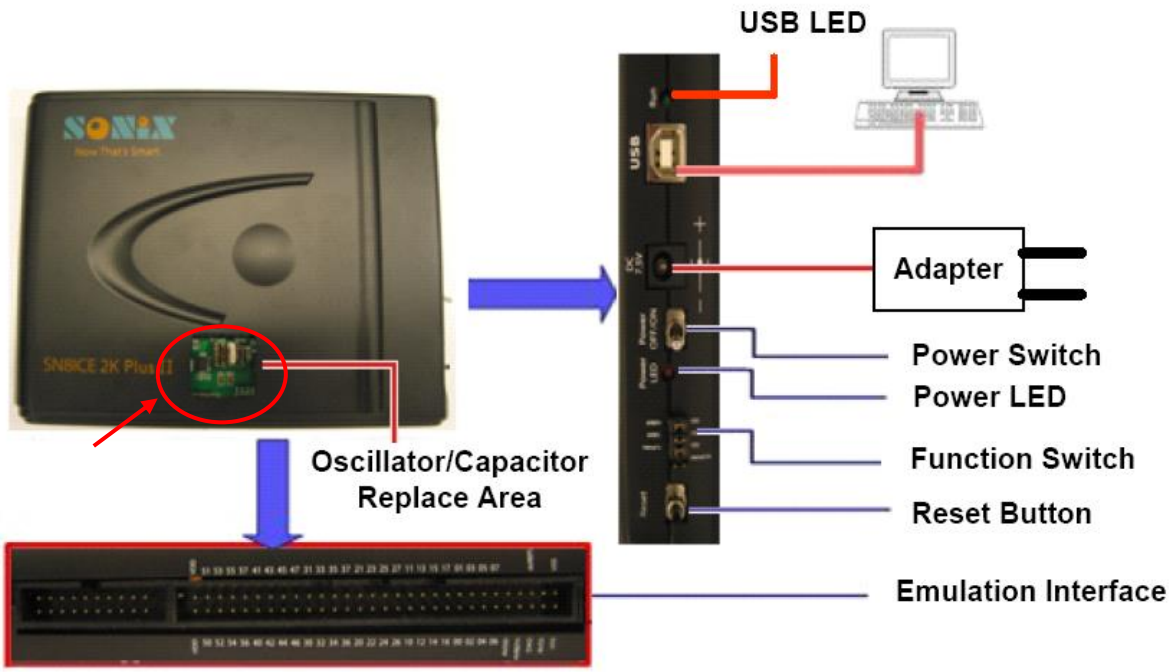
16.4 SN8P2977A Emulation

16.4.1 INTRODUCTION

Sonix provides a complete EV-KIT for SN8P2977A emulation, which includes an ICE SN8ICE2K_Plus_II, a SN8P2977A EV Board, Sonix Assembler and Complier. Users are able to do the programming on the computer and to simulate the program code using the software or the ICE itself. On the other hand, when executing the program and monitoring the RAM status, users can user various functions such as Breakpoint, Single step etc. This makes debug much easier for most programmers.

16.4.2 SN8ICE2K_Plus_II Hardware Setting Notice for SN2977A EV-Kit

1. Chosen 8MHz crystal connects to ICE for System high clock (Fhosc).
2. Check VDD is shorted to Internal_3.3V by jumper. VDD is only 3.3V available for SN8P2977A emulation.
3. Detail setting reference SN8ICE2K Plus II User's Manual.



16.4.3 SN8P2977A EV Board DESCRIPTION

Sonix provides SN8P2977A EV board for all functions emulation shown in FIG.1

FIG.1 SN8P2977A EV board

16.4.4 EV BOARD SETTING

1. CON1/CON2 : connecting with the ICE PORT
2. SW3: Switch it “ON” position when EV board using power from ICE (3.3V), and switch it “Off” when EV board using other power source via J4/J5 input.
3. D1 : Power indicator
4. J1/J10: Target board connector.
5. J2 : LCD COM/SEG connect Pin.
6. J3/J7: Analog Differential (AI+, AI-) input Pin.
7. SW3: Switch to “ICE” position when EV-Board connect to ICE. When separate ICE and Ev board, switch SW3 to “DEMO” position，or else the relative IO port won’t work.
8. C2: C-Type LCD external capacitors.

16.4.5 Notice for EV Emulation

1. ICE VDD must switch to 3.3V. VDD 5V is not available for SN8P2977A EV-Kit
2. SW3 must switch to “ICE” position.
3. Low Battery Detect (LBT) function only supports internal LBT emulation not support P11 Input.
4. X command is canceled in ICE emulation. (XB0MOV, XB0BSET...)

17 ELECTRICAL CHARACTERISTIC

17.1 ABSOLUTE MAXIMUM RATING

Supply voltage (V_{DD})..... - 0.3V ~ 3.6V
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

17.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to V_{SS}, V_{DD} = 3.0V, F_{OSC} = IHRC(8MHz),Fcpu=2MHZ, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd		2.4	3.0	3.6	V
RAM Data Retention voltage	Vdr			-	1.5	-	V
V_{DD} rise rate	V_{POR}	VDD rise rate to ensure power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input pins		Vss		0.3Vdd	V
Input High Voltage	ViH1	All input pins		0.7Vdd	-	Vdd	V
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
I/O port input leakage current	I_{LEKG}	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port source current sink current	P0 IoH	Vop = Vdd - 0.5V		7	10	-	mA
	P1 IoH			15	20	-	
	P2 IoH			15	20		
	P3 IoH			12	20		
	P0 IoL	Vop = Vss + 0.5V		8	10	-	
	P1 IoL			45	60	-	
	P2 IoL			45	60		
	P3 IoL			45	60		
INT0 trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
	Idd1	**Normal Mode**	Vdd=3V, Analog Parts OFF	-	0.7	1.4	mA
	Idd2		Vdd=3V, Analog Parts ON	-	1.6	2.4	mA
	Idd3	**Slow Mode**	Vdd=3V, High Clock On, Analog Parts On, C- LCD On	-	1	2	mA
	Idd4		Vdd=3V, High Clock On, Analog Parts OFF, C-LCD On	-	0.5	1	mA
	Idd5		Vdd=3V, High Clock OFF, Analog Parts OFF, R-LCD (VAR[1:0]=11)	-	5	10	uA
	Idd6		Vdd=3V, High Clock OFF, Analog Parts OFF, LCD OFF	-	3	6	uA
	Idd7	**Green Mode**	Vdd=3V, High Clock On, Analog Parts On, C-LCD On	-	1	2	mA
	Idd8		Vdd=3V, High Clock On, Analog Parts OFF, C- LCD On	-	0.5	1	mA
	Idd9		Vdd=3V, High Clock OFF, Analog Parts OFF, R-LCD (VAR[1:0]=11)	-	4	8	uA
	Idd10		Vdd=3V, High Clock OFF, Analog Parts OFF, LCD OFF	-	2	4	uA
	Idd11	**Sleep Mode**	Vdd= 3V	-	1	2	uA
LVD detect level	V_{LVD}	Internal POR detect level		1.6	1.8	2.0	V
Internal High Clock Freq.	F_{IHRC}	Internal High RC Oscillator Frequency (Vdd = 2.4V ~ 3.6V, Temperature: 25℃)		8-1.5%	8	8+1.5%	MHz

		Internal High RC Oscillator Frequency (Vdd = 2.4V ~ 3.6V, Temperature: 0℃~ 50℃)	8-2.5%	8	8+2.5%	
Internal Low Clock Freq.	FILRC	Internal Low RC Oscillator Frequency (Vdd = 2.4V ~ 3.6V, Temperature: 25℃)	32-40%	32	32+40%	KHz
		Internal Low RC Oscillator Frequency (Vdd = 2.4V ~ 3.6V, Temperature: 0℃~ 50℃)	32-50%	32	32+50%	

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

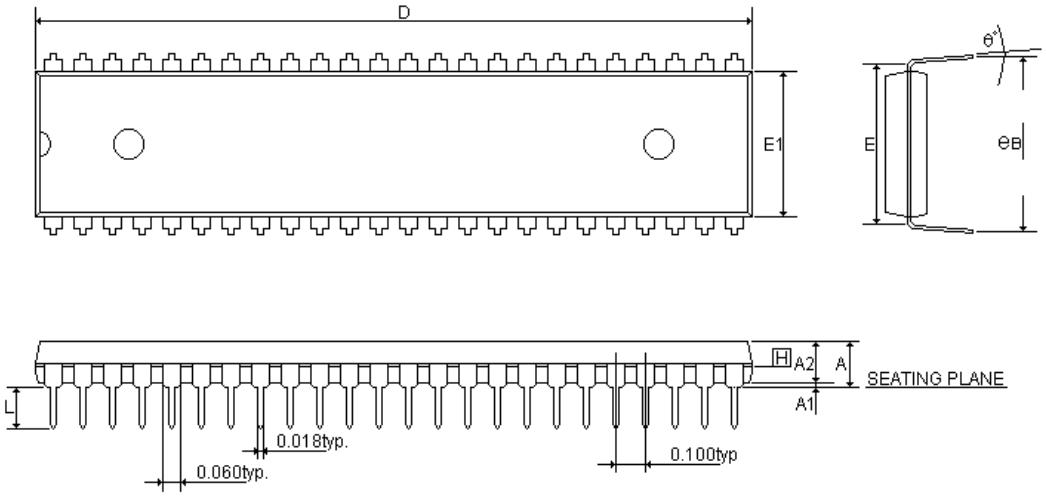
(All of voltages refer to Vdd=3V F_{OSC} = IHRC (8MHz), ambient temperature is 25℃ unless otherwise note.)

PARAMETER	*SYM.*	*DESCRIPTION*	*MIN.*	*TYP.*	*MAX.*	*UNIT*
Analog to Digital Converter						
Operating current	I_{DD_ADC}	Run mode @ 2.4V	-	160	250	uA
Power down current	I_{PDN}	Stop mode @ 2.4V	-	0.1	-	μA
Conversion rate (Word Rare, WR)	F_{WR}	ADC Clock=250KHz, OSR=32768	-	7.6	-	Sps
		ADC Clock=250KHz, OSR=64	-	3.9	-	kSps
Reference Voltage Input absolutely Voltage	V_{AIN}	ADC Internal Vref	0.4	-	1.4	V
Reference Voltage Range	Vref	ADC Reference voltage range	0.36	-	0.96	V
Integral non-linearity	INL	PGIAx128, ADC Input Range ±0.9xVref	-	0.01	-	%FSR
No missing code	NMC	ADC range ± 0.9 x Vref	20	-	-	bit
ADC Noise free bits	NFB	Gain:1, Vref:0.8V, OSR:32768, Input-short	-	18.4	-	bit
		Gain:128, Vref:0.8V, OSR:32768, Input-short	-	15.9	-	bit
Effective number of bits	ENOB	Gain=128, Vref=0.8V, OSR=32768, Input-short	-	18.6	-	bit
		Gain=1, Vref=0.8V, OSR:32768, Input-short	-	21.1	-	bit
ADC Input range	V_{AIN}	ADC input signal, signal after PGIA application	0.4	-	1.4	V
Temperature sensor inaccuracy	E_{TS}	Inaccuracy range vs. real Temp.	-	±10	-	%
PGIA						
PGIA Current consumption	I_{DD_PGIA}	Run mode @ 2.4V	-	150	-	uA
Power down current	I_{PDN}	Stop mode @ 2.4V	-	0.1	-	uA
Input offset voltage	Vos		-	25	-	uV
Bandwidth	BW		-	-	5	kHz
PGIA Gain Range	Gain	VDD = 2.4V, PGIA x 128	110	128	150	Gain
PGIA Input Range	Vopin	AI+, AI- signal input range. (AVDDR = 2.4V)	0.4	-	1.4	V
PGIA Output Range	Vopout	Signal output range. (AVDDR = 2.4V)	0.4	-	1.4	V
Band gap Reference						
Band gap Reference Voltage	V_{BG}	VDD: 2.4V ~ 3.6V	1.18	1.23	1.28	V
Reference Voltage Temperature Coefficient	T_{ACM}			50*		PPM/℃
Operating current	I_{BG}	Run mode @ 2.4V	-	120	-	uA
Regulator						
Regulator output voltage AVDDR	V_{AVDDR}	AVDDR set as 2.4V	2.25	2.4	2.55	V
Regulator output current capacity	I_{VA+}	AVDDR,AVE output current ability	-	5	10	mA
Quiescent current	I_{QI}	AVDDR +AVE	-	120	-	uA
Regulator output voltage AVE+	V_{AVE+}	AVE+ set as 2.0V	1.85	2.0	2.15	V
V_{AVE} sinking capacity	I_{SNK}	Sink Current Capacity @ AVE set 1V or 0.75V	-	-	1	mA
LCD Driver						
R-Type LCD Operation Current	I_{RLCD}	VDD:3V, 1/3 bias, No panel, R-LCD (VAR[1:0]=00)		30	-	uA
		(VDD:3V, 1/3 bias, No panel, R-LCD (VAR[1:0]=01)		8	-	
		VDD:3V, 1/3 bias, No panel, R-LCD (VAR[1:0]=10)		4	-	
		VDD:3V, 1/3 bias, No panel, R-LCD (VAR[1:0]=11)		2	-	
C-Type LCD Operation Current	I_{CLCD}	1/3 bias, LCD Charge pump + Bandgap current		80		uA
C-Type VLCD output Voltage	V_{LCD}	VLCD set 3V,	2.8	3.0	3.2	V
VLCD Variation vs. VDD and Temp		VDD: 2.4~3.6V. Temp.: -10 ~ 50℃	-30	-	30	mV

LBT Driver						
Internal Low-Battery detect voltage	V_{ILBT}	Condition: V_{LBT}=3.6V	3.4	3.6	3.8	V
		Condition: V_{LBT}=3.0V	2.8	3.0	3.2	
		Condition: V_{LBT}=2.4V	2.25	2.4	2.55	
External Low-Battery detect voltage	V_{ELBT}	Condition: VDD =2.2~3.6V, P10 input comparator	1.1	1.2	1.3	

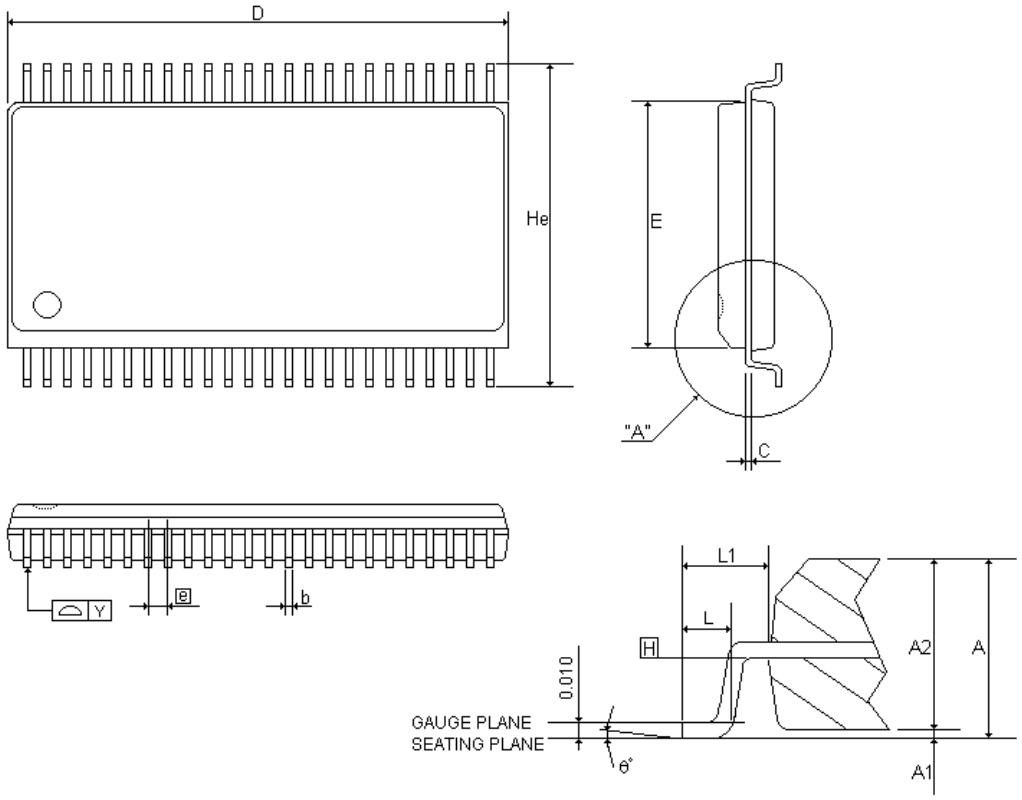
18 PACKAGE INFORMATION

18.1 DIP 48 PIN



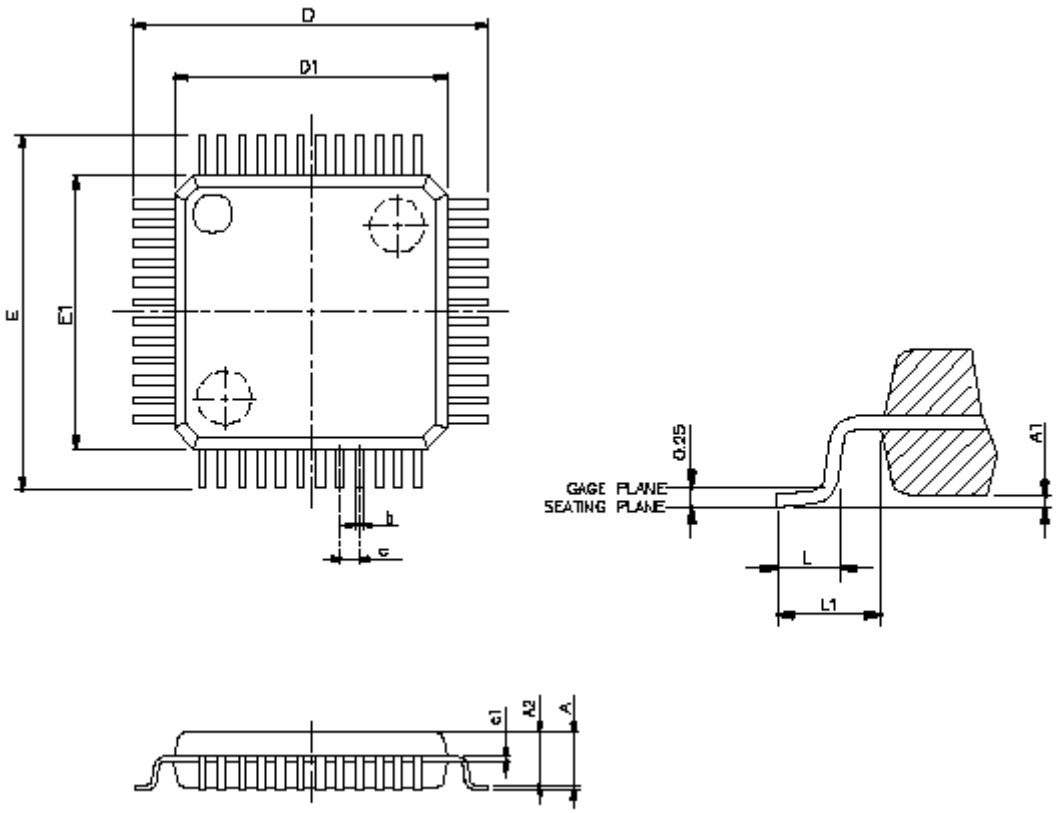
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.220	-	-	5.588
A1	0.015	-	-	0.381	-	-
A2	0.150	0.155	0.160	3.810	3.937	4.064
D	2.400	2.450	2.550	60.960	62.230	64.770
E		0.600			15.240	
E1	0.540	0.545	0.550	13.716	13.843	13.970
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.630	0.650	0.067	16.002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

18.2 SSOP 48 PIN



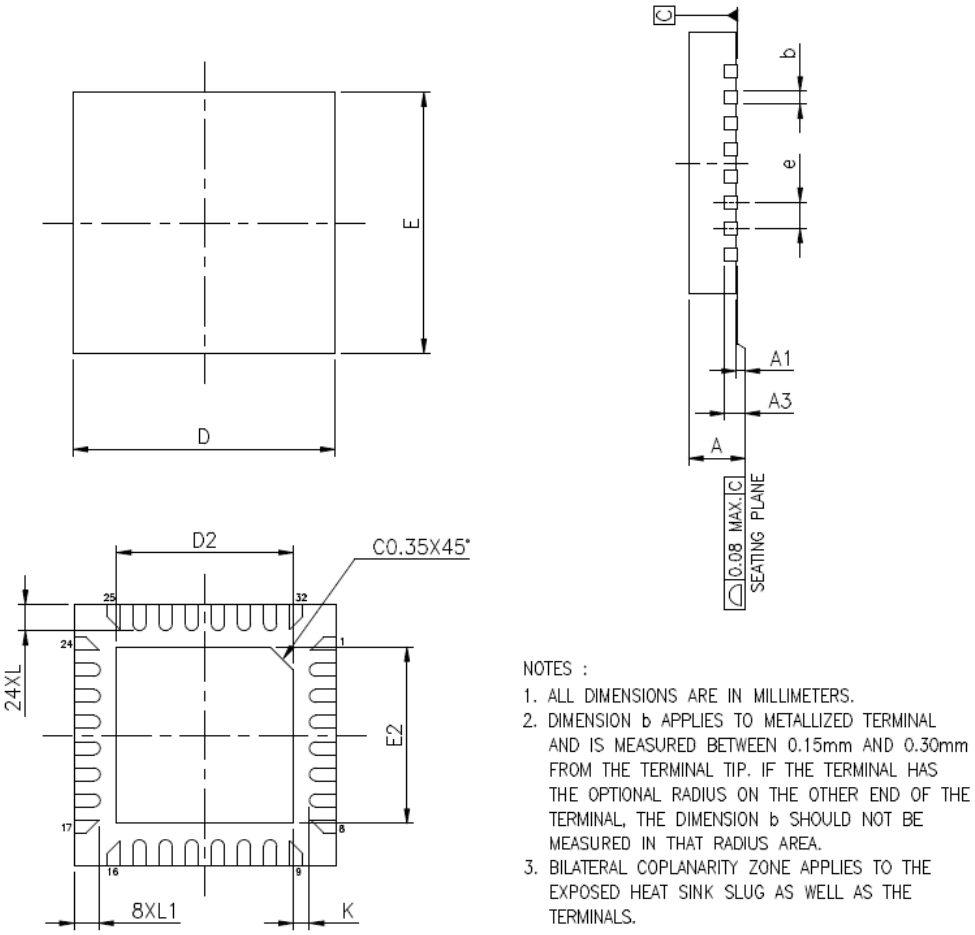
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.095	0.102	0.110	2.413	2.591	2.794
A1	0.008	0.012	0.016	0.203	0.305	0.406
A2	0.089	0.094	0.099	2.261	2.388	2.515
b	0.008	0.010	0.030	0.203	0.254	0.762
C	-	0.008	-	-	0.203	-
D	0.620	0.625	0.630	15.748	15.875	16.002
E	0.291	0.295	0.299	7.391	7.493	7.595
[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°

18.3 LQFP 48 PIN



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

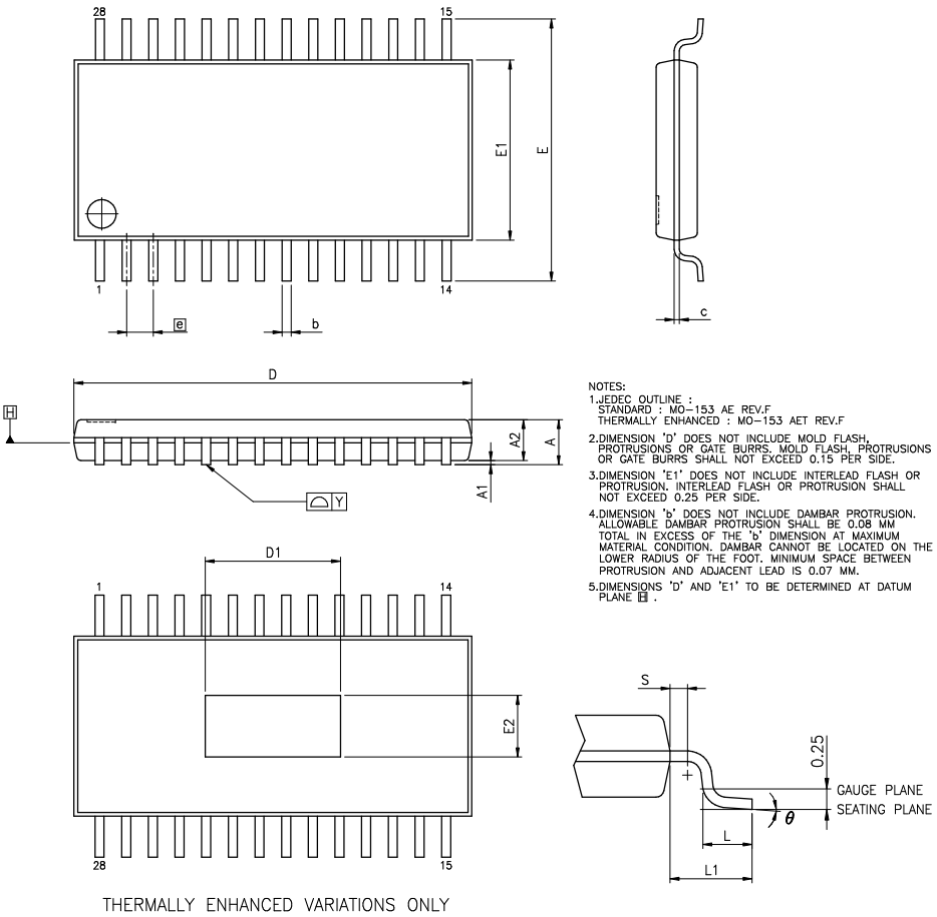
18.4 QFN 32 PIN



NOTES :
1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. DIMENSION b APPLIES TO METALLIZED TERMINAL AND IS MEASURED BETWEEN 0.15mm AND 0.30mm FROM THE TERMINAL TIP. IF THE TERMINAL HAS THE OPTIONAL RADIUS ON THE OTHER END OF THE TERMINAL, THE DIMENSION b SHOULD NOT BE MEASURED IN THAT RADIUS AREA.
3. BILATERAL COPLANARITY ZONE APPLIES TO THE EXPOSED HEAT SINK SLUG AS WELL AS THE TERMINALS.

SYMBOLS	Min	Typical	Max	Min	Typical	Max
	(inch)			(mm)		
A	0.028	0.030	0.031	0.70	0.80	0.90
A1	0	0.001	0.002	0	0.02	0.05
A3	-	0.008 REF	-	-	0.20 REF	-
b	0.006	0.008	0.010	0.15	0.20	0.25
D	-	0.157 BSC	-	-	4.00 BSC	-
E	-	0.157 BSC	-	-	4.00 BSC	-
e	-	0.016 BSC	-	-	0.40 BSC	-
L	0.014	0.016	0.018	0.35	0.40	0.45
L1	0.013	0.015	0.017	0.332	0.382	0.432
K	0.008	-	-	0.20	-	-
D2	0.102	0.106	0.108	2.60	2.70	2.8
E2	0.102	0.106	0.108	2.60	2.70	2.8

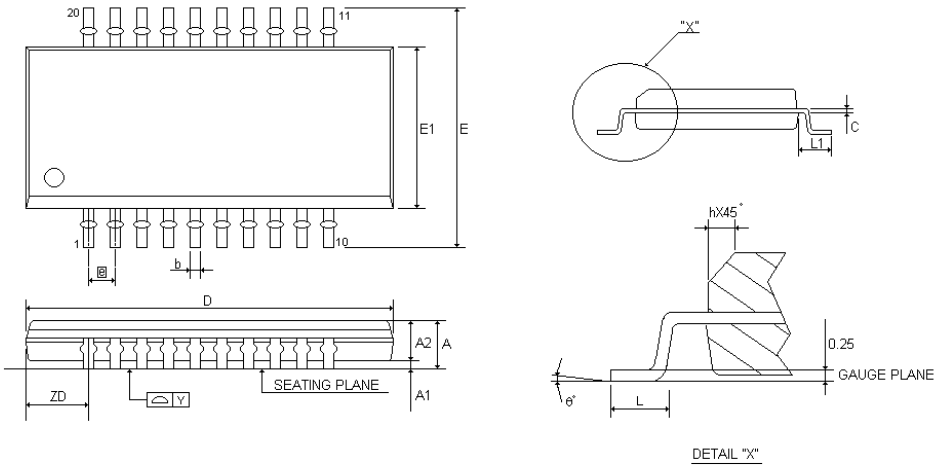
18.5 TSSOP28 PIN



THERMALLY ENHANCED VARIATIONS ONLY

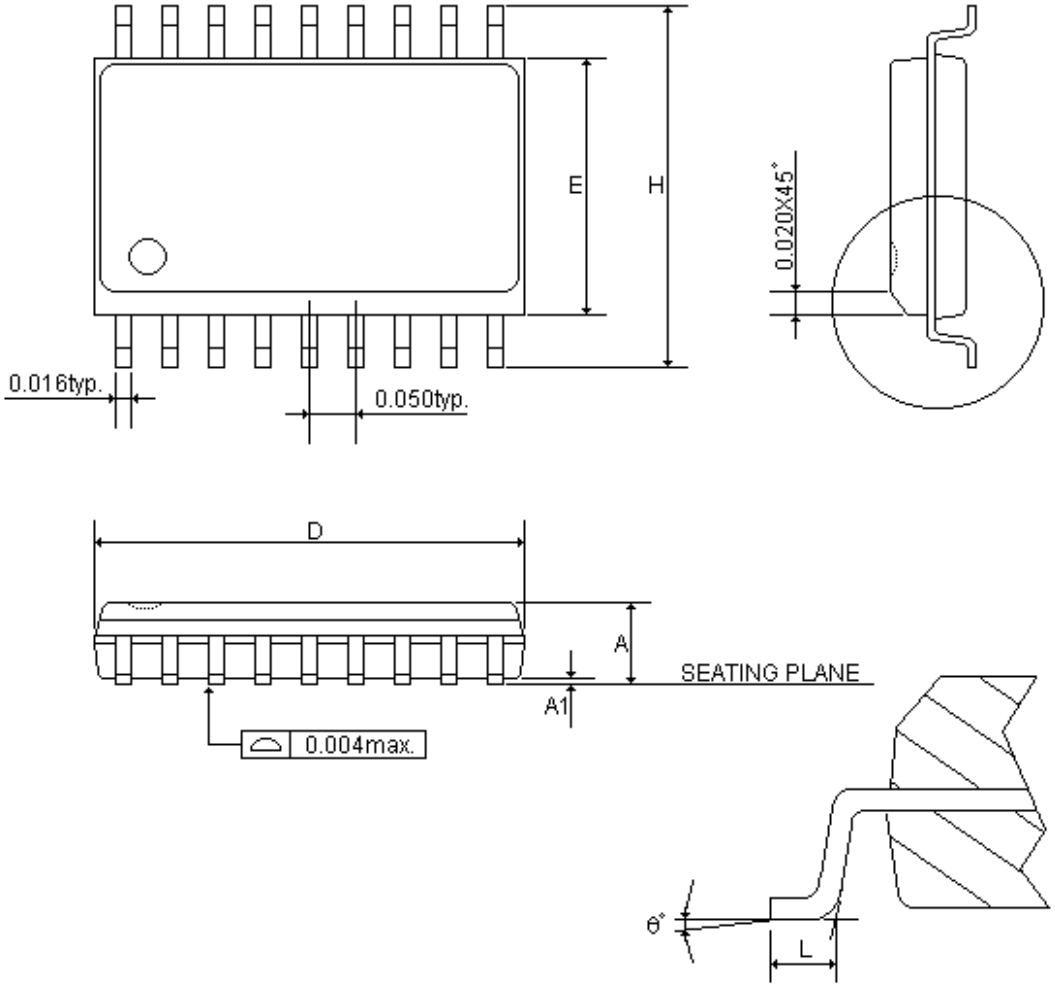
SYMBOLS	Min	Typical	Max	Min	Typical	Max
	(inch)			(mm)		
A	-	-	0.047	-	-	1.2
A1	0.000	-	0.006	0.00	-	0.15
A2	0.031	0.039	0.041	0.80	1.00	1.05
b	0.007	-	0.012	0.19	-	0.30
c	0.004	-	0.008	0.09	-	0.20
D	0.378	0.382	0.386	9.60	9.70	9.80
E	-	0.252 BSC	-	-	6.40 BSC	-
E1	0.169	0.173	0.177	4.30	4.40	4.50
[e]	-	0.026 BSC	-	-	0.65 BSC	-
L	0.018	0.024	0.030	0.45	0.60	0.75
L1	-	0.039 REF	-	-	1.00 REF	-
S	0.008	-	-	0.20	-	-
θ °	0°	-	8°	0°	-	8°
Y	-	0.004	-	-	0.10	-
E2	0.094	-	0.124	2.40	-	3.15
D1	0.174	-	0.219	4.41	-	5.56

18.6 SSOP20 PIN



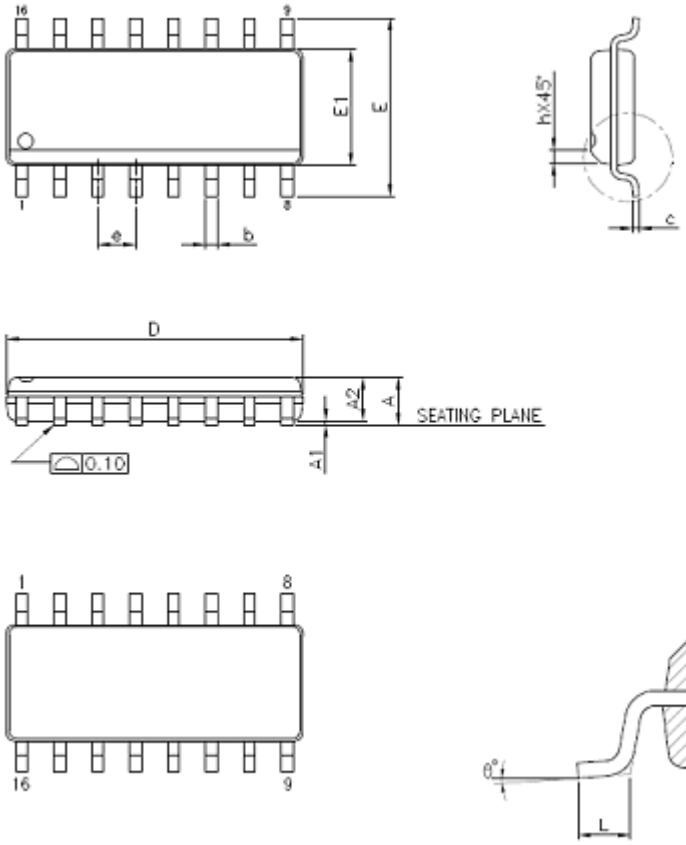
SYMBOLS	Min	Typical	Max	Min	Typical	Max
	(inch)			(mm)		
A	0.053	0.063	0.069	1.350	1.600	1.750
A1	0.004	0.006	0.010	0.100	0.150	0.250
A2	-	-	0.059	-	-	1.500
b	0.008	0.010	0.012	0.200	0.254	0.300
c	0.007	0.008	0.010	0.180	0.203	0.250
D	0.337	0.341	0.344	8.560	8.660	8.740
E	0.228	0.236	0.244	5.800	6.000	6.200
E1	0.150	0.154	0.157	3.800	3.900	4.000
[e]	-	0.025	-	-	0.635	-
h	0.010	0.017	0.020	0.250	0.420	0.500
L	0.016	0.025	0.050	0.400	0.635	1.270
L1	0.039	0.041	0.043	1.000	1.050	1.100
ZD	-	0.059	-	-	1.500	-
Y	-	-	0.004	-	-	0.100
θ °	0°	-	8°	0°	-	8°

18.7 SOP 18 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.447	0.455	0.463	11.354	11.557	11.760
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
θ°	0°	4°	8°	0°	4°	8°

18.8 SOP 16 PIN



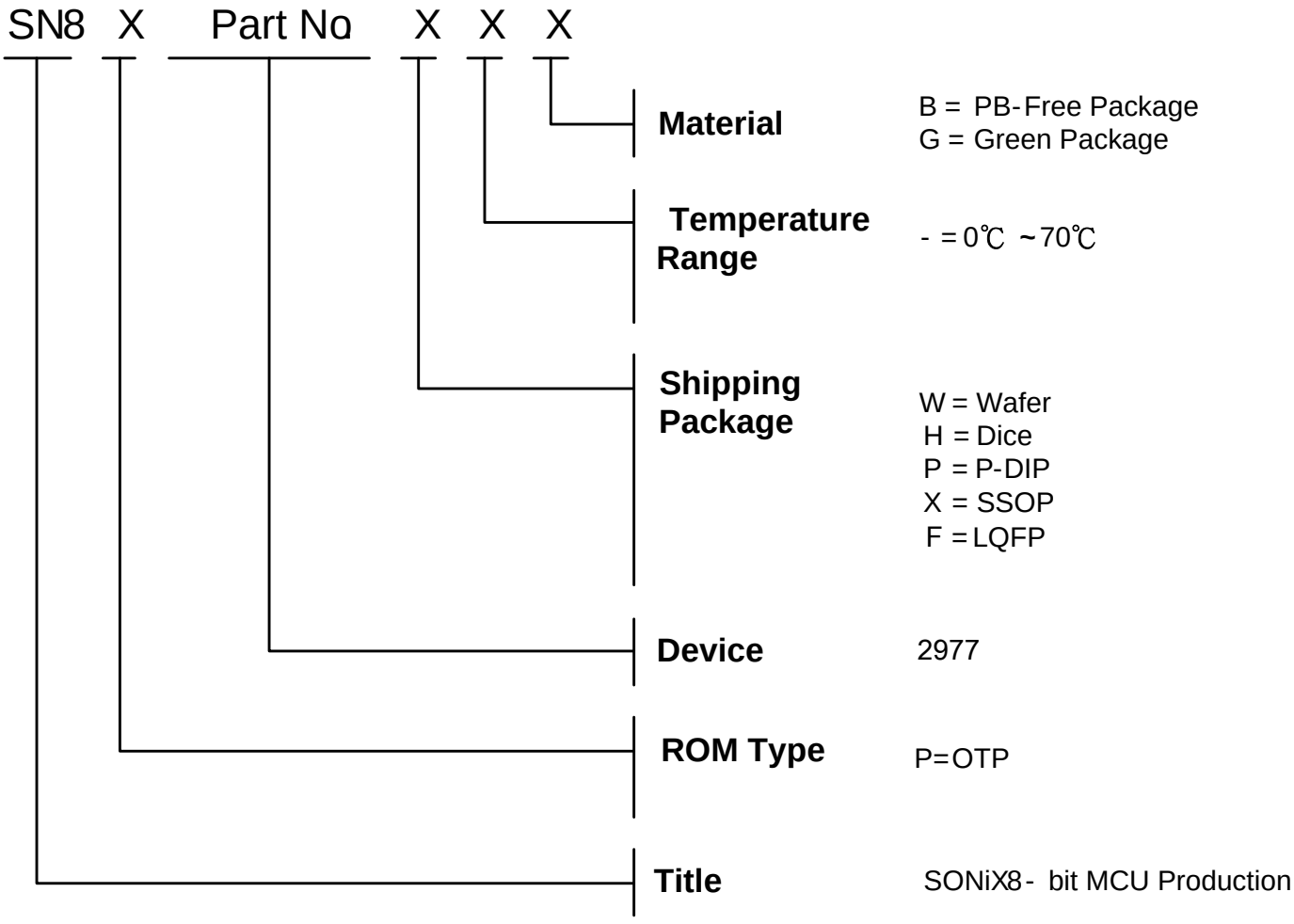
SYMBOLS	Min	Typical	Max	Min	Typical	Max
	(inch)			(mm)		
A	-	-	0.069	-	-	1.75
A1	0.004	-	0.010	0.10	-	0.25
A2	0.049	-	-	1.25	-	-
b	0.012	-	0.020	0.31	-	0.51
c	0.004	-	0.010	0.10	-	0.25
D	-	0.39BSC	-	-	9.90BSC	-
E	-	0.236BSC	-	-	6.00BSC	-
E1	-	0.154BSC	-	-	3.90BSC	-
e	-	0.05BSC	-	-	1.27BSC	-
h	0.016	-	0.050	0.40	-	1.27
L	0.010	-	0.020	0.25	-	0.50
θ °	0°	-	8°	0°	-	8°

19 Marking Definition

19.1 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 obtains information. This definition is only for Blank OTP MCU.

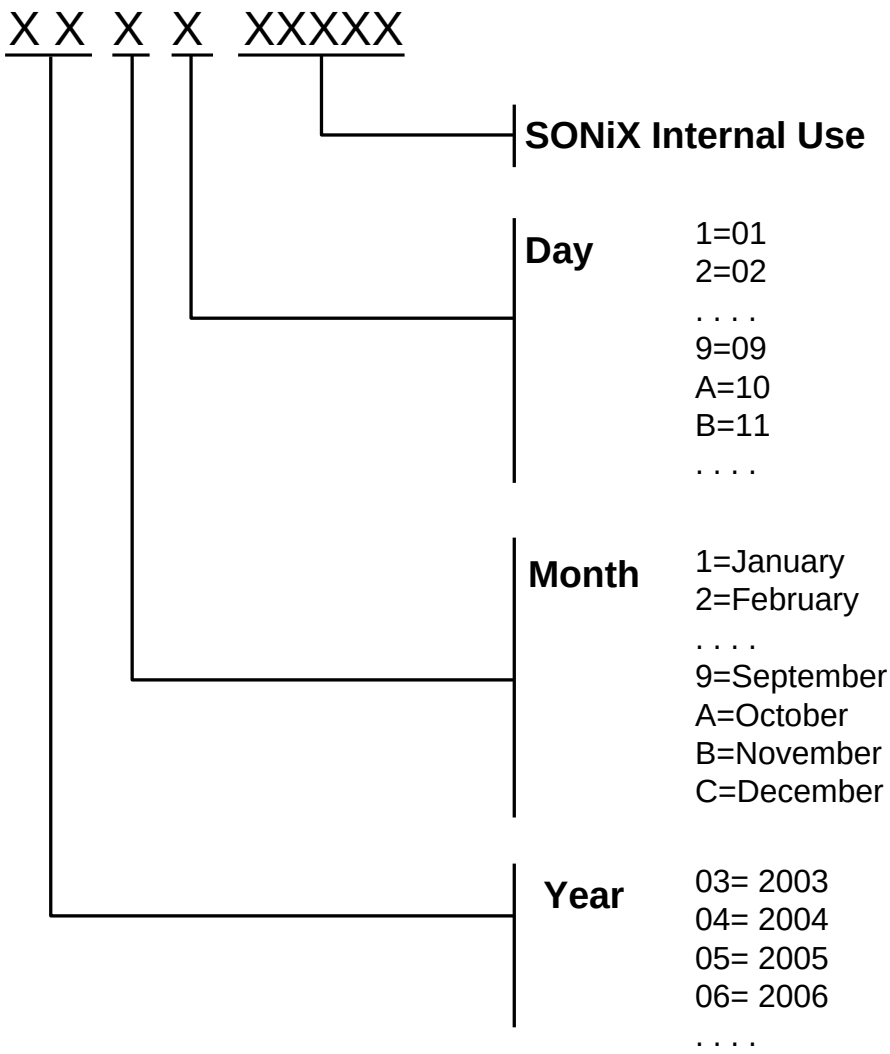
19.2 MARKING INDETIFICATION SYSTEM



19.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P2977APG	OTP	2977A	DIP48	0℃~70℃	Green Package
SN8P2977AFG	OTP	2977A	LQFP48	0℃~70℃	Green Package
SN8P2975AJG	OTP	2977A	QFN32	0℃~70℃	Green Package
SN8P2974ATG	OTP	2977A	TSSOP28	0℃~70℃	Green Package
SN8P2973A1XG	OTP	2977A	SSOP20	0℃~70℃	Green Package
SN8P2972ASG	OTP	2977A	SOP18	0℃~70℃	Green Package
SN8P2971ASG	OTP	2977A	SOP16	0℃~70℃	Green Package

19.4 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