SN8P2947

USER'S MANUAL

Specification V1.6

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
Preliminary	2010.09	First issue Brief version.
PreV02	2010.09	Add EV2947 schematic Modified Block diagram
VER1.0	2010.10	New Version start
VER 1.1	2010.11	Modified ISP function. Add Internal VPP 6.5V generation. Add control bit VCP3 and VPPINTL. Modified ADC Offset source.
VER 1.2	2011.01	Modify VCP[0000] = 2.2V. Modify C-Type VLCD output voltage 2.8V ~ 3.4V with 0.1v step. Modify User program area 000H~7FBH, 7FCH~7FFH reserved. Add illustration of how to turn-On and turn-off LCD Charge pump. Add ISP macro “RomwrtVpp”. Add Fcpu clock option Fhosc/16 and Fhosc/32. Modify electrical characteristic table.
VER 1.3	2011.06	Cancel Fcpu clock option Fhosc/16 and Fhosc/32. Add ADC clock /2 function controlled by “ADCKSDIV”. Modify ADC clock 333kHz to 166.5KHz. Modify Chopper clock 31.25kHz / 15.6KHz / 20.8KHz / 10.4KHz. Modify AVE / AVDDR output max. current spec 5mA. Modify ADC resolution spec. to 16-Bit.
VER 1.4	2011.08	Modify ADC WR table. ADD Temperature sensor and spec. Modify example code “PGIA channel change”. Modify capacitor table of analog setting and application.
VER 1.5	2011.11	Control bit “LCDREF” must set “1” in R-Type LCD driver mode. Error correction: LCD Drive Waveform. Cancel Macro “@B0BTS1_FVPPCHK”. ISP example code modify: add 1ms delay after VPPCHK bit checked “1”.
VER 1.6	2011.11	Interrupt Function must be “disable” before ISP execution. Add ISP Example code.

Table of Content

AMENDENT HISTORY 2
1 PRODUCT OVERVIEW 7
1.1 SELECTION TABLE 7
1.2 MIGRATION TABLE 7
1.3 FEATURES 8
1.4 SYSTEM BLOCK DIAGRAM 9
1.5 PIN ASSIGNMENT 10
1.6 PIN DESCRIPTIONS 12
1.7 PIN CIRCUIT DIAGRAMS 13
2 CENTRAL PROCESSOR UNIT (CPU) 14
2.1 MEMORY MAP 14
2.1.1 PROGRAM MEMORY (ROM) *14*
2.1.2 RESET VECTOR (0000H) *15*
2.1.3 CODE OPTION TABLE 23
2.1.4 DATA MEMORY (RAM) 24
2.1.5 SYSTEM REGISTER *25*
2.1.6 ACCUMULATOR *28*
2.1.7 PROGRAM FLAG *29*
2.1.8 PROGRAM COUNTER *30*
2.1.9 R REGISTERS *34*
2.2 ADDRESSING MODE 35
2.2.1 IMMEDIATE ADDRESSING MODE *35*
2.2.2 DIRECTLY ADDRESSING MODE *35*
2.2.3 INDIRECTLY ADDRESSING MODE *35*
2.3 STACK OPERATION 36
2.3.1 OVERVIEW *36*
2.3.2 STACK REGISTERS *37*
2.3.3 STACK OPERATION EXAMPLE *38*
3 RESET 39
3.1 OVERVIEW 39
3.2 POWER ON RESET 40
3.3 WATCHDOG RESET 40
3.4 BROWN OUT RESET 40
3.4.1 BROWN OUT DESCRIPTION *40*
3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION *41*
3.4.3 BROWN OUT RESET IMPROVEMENT *42*
4 SYSTEM CLOCK 43
4.1 OVERVIEW 43
4.2 CLOCK BLOCK DIAGRAM 43
4.3 OSCM REGISTER 44

4.4 SYSTEM HIGH CLOCK 45
4.5 SYSTEM LOW CLOCK 45
4.5.1 SYSTEM CLOCK MEASUREMENT 46
5 SYSTEM OPERATION MODE 47
5.1 OVERVIEW 47
5.2 SYSTEM MODE SWITCHING 48
5.3 WAKEUP 50
5.3.1 OVERVIEW 50
5.3.2 WAKEUP TIME 50
6 INTERRUPT 51
6.1 OVERVIEW 51
6.2 INTEN INTERRUPT ENABLE REGISTER 52
6.3 INTRQ INTERRUPT REQUEST REGISTER 52
6.4 GIE GLOBAL INTERRUPT OPERATION 53
6.5 PUSH, POP ROUTINE 54
6.6 INT0 (P0.0) INTERRUPT OPERATION 55
6.7 MULTI-INTERRUPT OPERATION 56
7 I/O PORT 57
7.1 I/O PORT MODE 57
7.2 I/O PULL UP REGISTER 58
7.3 I/O PORT DATA REGISTER 59
8 TIMERS 60
8.1 WATCHDOG TIMER 60
8.2 TIMER 0 (T0) 62
8.2.1 OVERVIEW 62
8.2.2 T0M MODE REGISTER 62
8.2.3 T0C COUNTING REGISTER 63
8.2.4 T0 TIMER OPERATION SEQUENCE 64
9 LCD DRIVER 65
9.1 LCD TIMING 65
9.2 LCDM1 REGISTER 67
9.3 LCDM2 REGISTER 68
9.4 C-TYPE LCD DRIVER MODE 69
9.5 R-TYPE LCD DRIVER MODE 70
9.6 LCD RAM LOCATION 72
10 IN SYSTEM PROGRAM ROM 73
10.1 OVERVIEW 73
10.2 ROMADRH/ROMADRL REGISTER 73
10.3 ROMDAH/ROMADL REGISTERS 73
10.4 ROMCNT REGISTERS AND ROMWRT INSTRUCTION 74
10.5 ISP ROM ROUTINE EXAMPLE 75

11 REGULATOR, PGIA AND ADC 77
11.1 OVERVIEW 77
11.2 ANALOG INPUT 77
11.3 VOLTAGE REGULATOR 78
11.3.1 Voltage Regulator Control Register 78
11.4 PGIA -PROGRAMMABLE GAIN INSTRUMENTATION AMPLIFIER 79
11.4.1 AMPM1- Amplifier Mode1 Control Register 79
11.4.2 AMPM2- Amplifier Mode2 Control Register 81
11.5 TEMPERATURE SENSOR (TS) 82
11.6 16-BIT ANALOG TO DIGITAL CONVERTER (ADC) 84
11.6.1 Analog Inputs and Voltage Operation Range 85
11.6.2 Reference Voltage 85
11.6.3 Input Buffer 85
11.6.4 ADC Gain and Offset 85
11.6.5 Output Word Rate 86
11.6.6 ADCM1- ADC Mode1 Register 86
11.6.7 ADCM2- ADC Mode2 Register 87
11.6.8 ADC Data Register 89
11.7 LBTM: LOW BATTERY DETECT 95
11.7.1 LBTM: Low Battery Detect Register 95
11.8 ANALOG SETTING AND APPLICATION 97
12 APPLICATION CIRCUIT 98
12.1 SCALE (LOAD CELL) APPLICATION CIRCUIT 98
12.2 THERMOMETER APPLICATION CIRCUIT 99
13 INSTRUCTION SET TABLE 100
14 DEVELOPMENT TOOLS 101
14.1 DEVELOPMENT TOOL VERSION 101
14.1.1 ICE (IN CIRCUIT EMULATION) 101
14.1.2 OTP WRITER 101
14.1.3 IDE (INTEGRATED DEVELOPMENT ENVIRONMENT) 101
14.2 OTP PROGRAMMING PIN TO TRANSITION BOARD MAPPING 102
14.3 APPENDIX A: EV-KIT BOARD CIRCUIT 104
14.4 SN8P2947 EMULATION 105
14.4.1 INTRODUCTION 105
14.4.2 SN8ICE2K_Plus_II Hardware Setting Notice for SN2947 EV-Kit 105
14.4.3 SN8P2947 EV Board DESCRIPTION 106
14.4.4 EV BOARD SETTING 106
14.4.5 Notice for EV Emulation 107
15 ELECTRICAL CHARACTERISTIC 108
15.1 ABSOLUTE MAXIMUM RATING 108
15.2 ELECTRICAL CHARACTERISTIC 108
16 PACKAGE INFORMATION 110

16.1 SSOP 48 PIN 110
16.2 LQFP 48 PIN 111

17 MARKING DEFINITION 112

17.1 INTRODUCTION 112
17.2 MARKING INDETIFICATION SYSTEM 112
17.3 MARKING EXAMPLE 113
17.4 DATECODE SYSTEM 113

1 PRODUCT OVERVIEW

1.1 SELECTION TABLE

CHIP	ROM	RAM	Stack	LCD	Timer			I/O	ADC	PWM Buzzer	RTC	Wakeup Pin no.	Package
					T0	TC0	TC1						
SN8P1927	2K*16	128*8	8	4*12	V	-	-	13	16-bit	-	-	5	SSOP48/LQFP48
SN8P1937	2K*16	128*8	8	4*12	V	V	-	13	16-bit	1	V	5	LQFP64
SN8P2947	2K*16	128*8	8	4*12	V			10	16-bit	-	-	8	DIP48/SSOP48/LQFP48

Table 1-1 Selection table of SN8Px9x7 serial

1.2 MIGRATION TABLE

Item	SN8P1927	SN8P1937	SN8P2947
PGIA Gain setting	1x, 12.5x, 50x, 100x, 200x	1x, 16x, 32x, 64x, 128x (ADC Gain option ~ 4x)	1x, 16x, 32x, 64x, 128x (ADC Gain option ~ 2x)
PGIA Temperature Drift	Good	Good	Good
AVE+ Voltage	2.0V or 1.5V	1.5V	2.0V or 1.5V
AVE+ current loading when CPR ON	NOT Double Current Consumption	NOT Double Current Consumption	NOT Double Current Consumption
ADC Reference Voltage V(R+, R-)	0.4V	0.6V or 0.3V	0.3V~0.8V
ADC output Rate	Up to 50Hz	Up to 50Hz	2.5Hz~3.9kHz
ADC Interrupt	No	No	Yes
ADC Run in Green Mode (with wake-up function)	No	No	Yes
Battery Detect Method	By Comparator or By ADC	By Comparator or By ADC	By Comparator or By ADC
Temperature Sensor	Build In	Build In	N/A
ACM Voltage	Not Change with Sink current	Not Change with Sink current	1V / 0.75V
Charge pump clock frequency (CPCKS)	4-Bit Selection	No	No
Chopper clock frequency (AMPCKS)	3-Bit Selection	3-Bit Selection	31.25K / 15.6 k
AVDDR/AVE+ working in slow mode	Yes	Yes	Yes
Operating Current Consumption	Less	Least	Least
Slow mode Current Consumption	Less	Least	Least
LCD Bias Voltage	1/3 or 1/2 Bias	1/3 or 1/2 Bias	1/3 or 1/2 Bias
LCD Type	R type only	R type/ C type	R type / C type
VLCD Voltage	Fixed	Adjustable	Adjustable (2.8V~3.4V)
Internal High RC Oscillator	Yes	Yes	Yes
P2 [1:0] I/O	Available when Fosc=IHRC	Available when Fosc=IHRC	N/A
OTP Programming Method	Serial Method	Serial Method	Serial Method
In-System Programmer ROM	Yes	Yes	Yes (Internal 6.5V)

Table 1-2 SN8Px9x7 Migration Table

1.3 FEATURES

- ***Memory configuration***
  OTP ROM size: 2K * 16 bits
  RAM size: 128 * 8 bits (bank 0)
  8-levels stack buffer
  LCD RAM size: 4*12bits
- ***I/O pin configuration***
  Bi-directional: P0, P1
  Wakeup: P0
  Pull-up resistors: P0, P1
  External interrupt: P0
- ***Powerful instructions***
  Four clocks per instruction cycle
  All instructions are one word length
  Most of instructions are 1 cycle only
  Maximum instruction cycle is “2”
  JMP instruction jumps to all ROM area
  All ROM area look-up table function (MOVC)
- ***A 8-Bit Timer***
  T0: Basic Timer
- ***Programmable Gain Instrumentation Amplifier***
- ***PGIA Gain option: 1x/16x/32x/64x/128x***
- **16-bit Delta-Sigma ADC**
  ADC Gain selection: 1x, 2x
  ADC Offset selection: (-1/2, -1/4 or 0) x Vref
  ADC Interrupt and Green Mode wakeup function
  Two ADC channel configuration:
  One fully differential channel
  Two single channels
- ***Two interrupt sources***
  Two internal interrupts: T0, ADC
  One external interrupts: INT0 (P00)
- ***Single power supply: 2.4V ~ 3.6V***
- ***On-chip watchdog timer***
- ***On-chip Regulator (AVDDR) with 2.4V voltage output and 5mA driven current***
- ***On chip regulator with 1.5V / 2.0V output voltage AVE+ Loading current consumption will Not double***
- ***On-chip 1.2V Band gap reference for battery monitor***
- ***On chip Voltage Comparator for Low battery detection***
- ***Build in ADC reference voltage V(R+,R-)= 0.3V~0.8V***
- ***In-system Programmer ROM with internal 6.5V generation***
- ***LCD driver:***
  1/3 or 1/2 bias voltage.
  4 common * 12 segment
  Both R type and C type LCD
  Multiple C-type LCD Voltage: 2.8 ~3.4V
- ***Dual clock system offers four operating modes***
  Internal high clock: RC type up to 4 MHz
  Internal Low clock: RC type up to 32kHz
  Normal mode: Both high and low clock active
  Slow mode: Low clock active and optional high clock
  Green mode: Low clock active and optional high clock
  Sleep mode: Both high and low clock stop
- ***Package***

  Dice /DIP48 /SSOP48/ LQFP48

1.4 SYSTEM BLOCK DIAGRAM

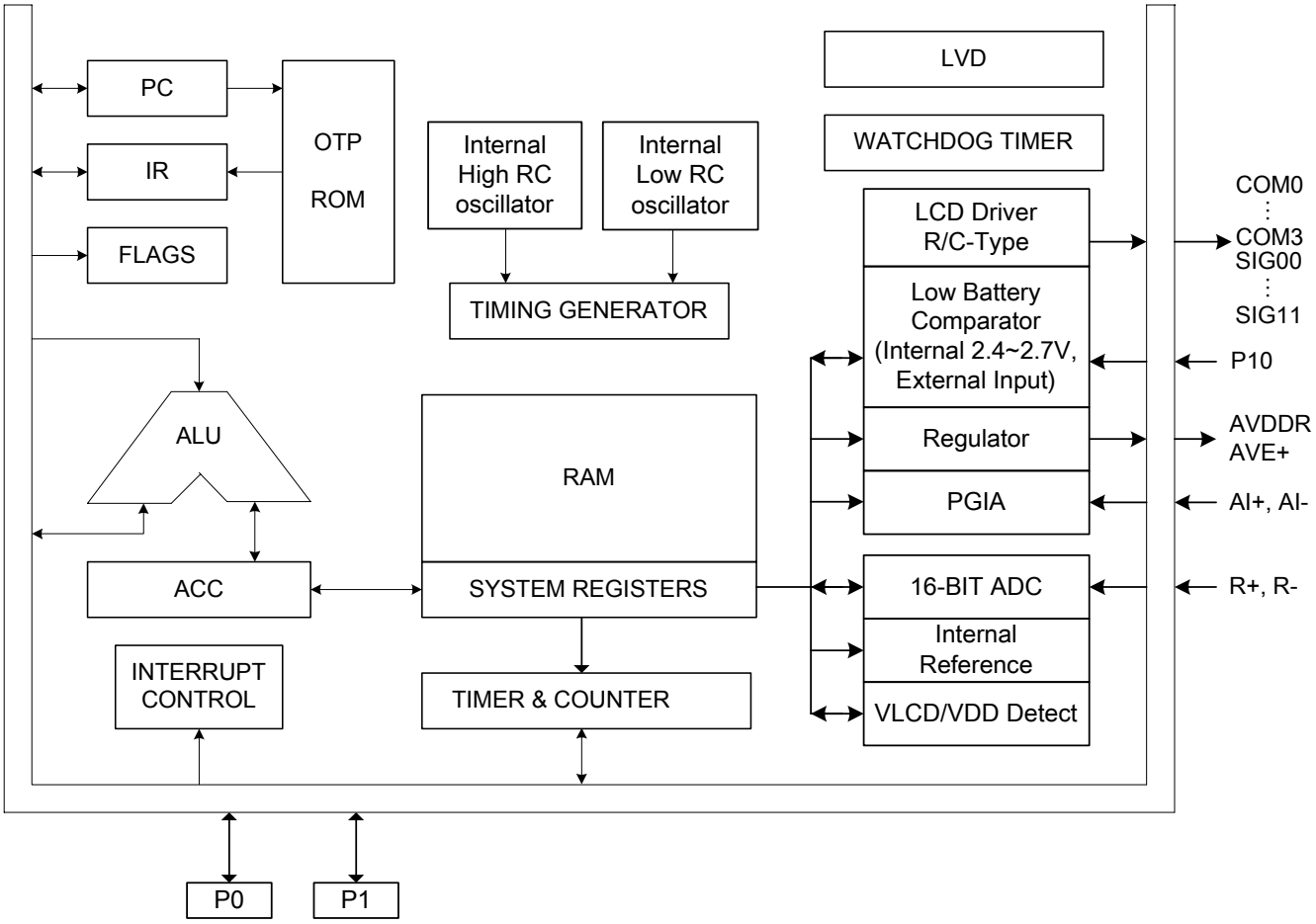


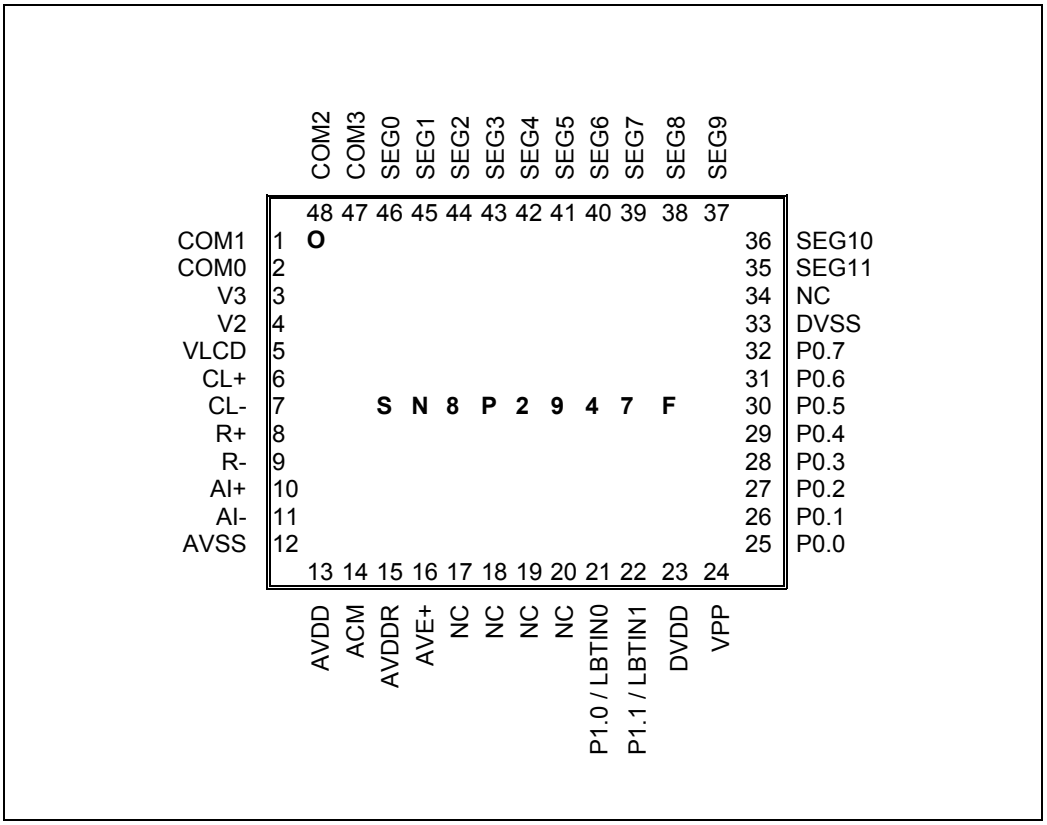
Figure 1-1 Simplified system block diagram

1.5 PIN ASSIGNMENT

SN8P2947 DIP48/SSOP48

SEG4	1	48	SEG5
SEG3	2	47	SEG6
SEG2	3	46	SEG7
SEG1	4	45	SEG8
SEG0	5	44	SEG9
COM3	6	43	SEG10
COM2	7	42	SEG11
COM1	8	41	NC
COM0	9	40	DVSS
V3	10	39	NC
V2	11	38	NC
VLCD	12	37	NC
CL+	13	36	P0.7
CL-	14	35	P0.6
R+	15	34	P0.5
R-	16	33	P0.4
AI+	17	32	P0.3
AI-	18	31	P0.2
AVSS	19	30	P0.1
AVDD	20	29	P0.0
ACM	21	28	VPP
AVDDR	22	27	DVDD
AVE+	23	26	P1.1 / LBTIN1
NC	24	25	P1.0 / LBTIN0

SN8P2947 LQFP48

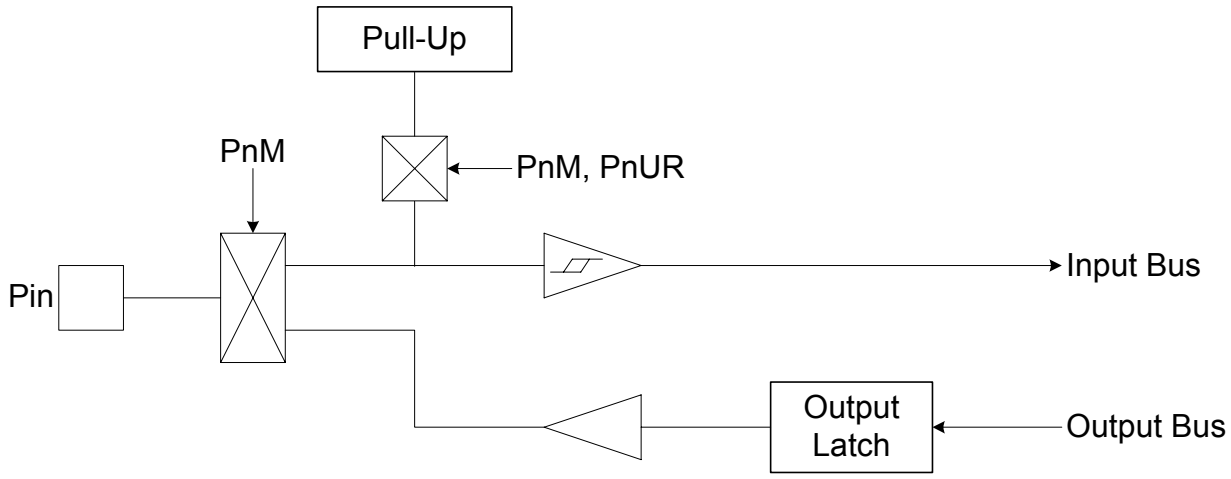


1.6 PIN DESCRIPTIONS

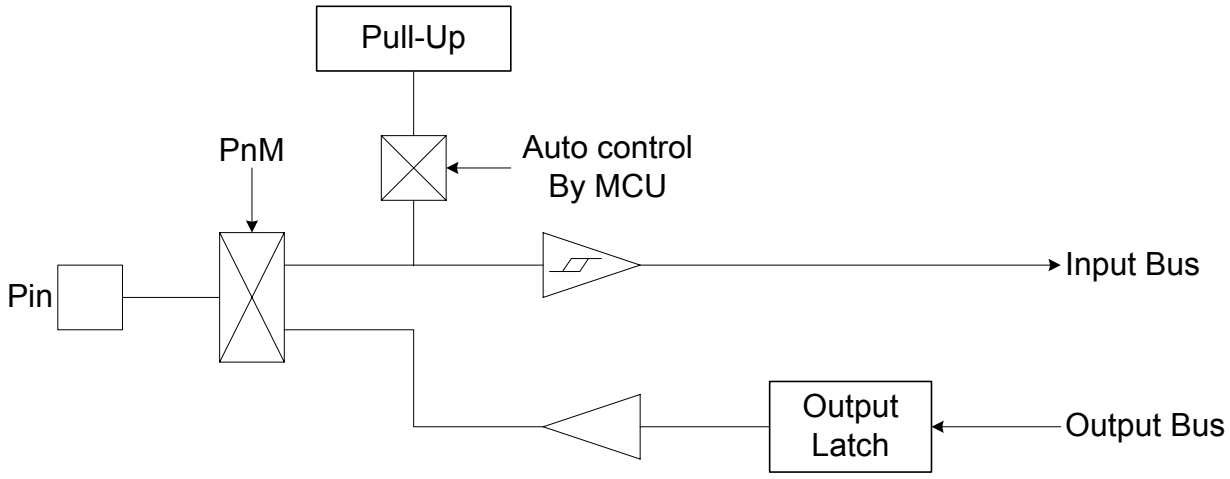
PIN NAME	TYPE	DESCRIPTION
DVDD, DVSS	P	Power supply input pins for digital circuit.
AVDD, DVSS	P	Power supply input pins for analog circuit.
AVDDR	P	Regulator power output pin, Voltage = 2.4V. AVDDR is analog circuit power source. (PGIA, ADC...)
AVE+	P	Regulator output = 2.0/1.5V for Sensor. Maximum output current = 5mA.
ACM	P	ACM Voltage output = 1V /0.75V.
R+	AI	ADC positive reference voltage input.
R-	AI	ADC negative reference voltage input.
AI+	AI	Positive analog input channel of PGIA.
AI-	AI	Negative analog input channel of PGIA.
VPP	P	OTP ROM programming pin only. No reset function.
P0.0 / INT0	I/O	Port 0.0 and share with INT0 trigger pin (Schmitt trigger). Built-in pull-up resisters.
P0 [7:0]	I/O	Port0.0~Port 0.7 bi-direction pins. Green mode or sleep mode wakeup pins. Built-in pull-up resisters.
P1 [1:0]	I/O	Port1.0~Port 1.1 bi-direction pins. Built-in pull-up resisters controlled by P1UR.
COM [3:0]	O	COM0~COM3 LCD driver common port.
LBTIN0/1	I	LBTIN0 (P10) is external input pin for low battery detection. LBTIN1 (P11) is internal grounding pin for low battery detection.
SEG0 ~ SEG11	O	LCD driver segment pins.
CL+, CL-	P	C-Type LCD charge pump capacitor.
VLCD	P	LCD driver power pin. When R-Type LCD mode, VLCD = VDD.
V3	P	LCD bias voltage.
V2	P	LCD bias voltage.

1.7 PIN CIRCUIT DIAGRAMS

Port 0 structure:



Port 1 structure:



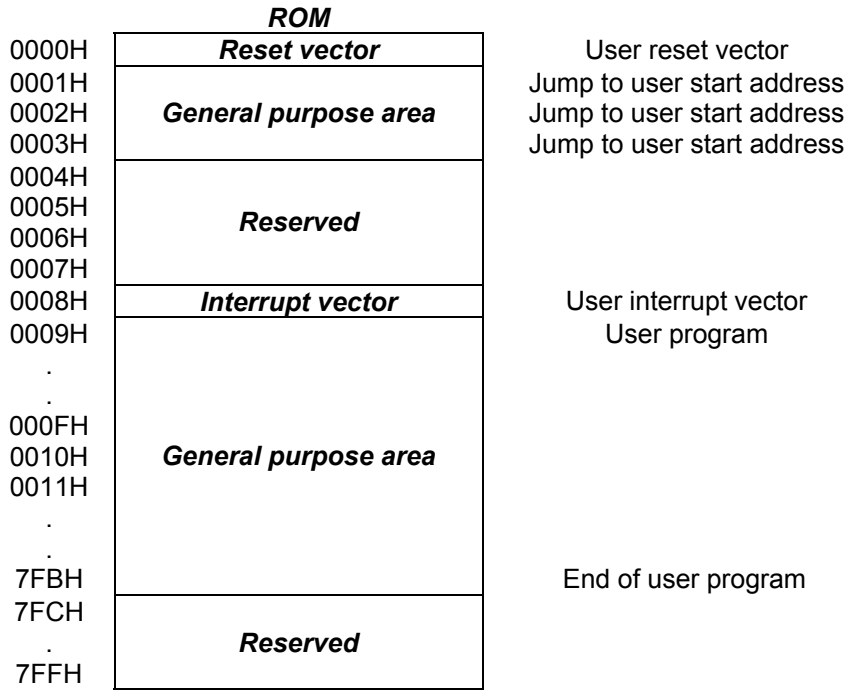
Note: Port_1 as input Mode, P0 pull-up resistor is auto enabled.

2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ ***2K words ROM***



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:***

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

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 = 1MHz.
	Fhosc/8	High clock Fcpu = IHRC/8 = 500kHz.

- ***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)

☞ ***128 X 8-bit RAM***

		RAM location	
BANK 0	000h	General purpose area	; 000h~07Fh of Bank 0 = To store general
		.	; purpose data (128 bytes)
	07Fh	.	
	080h	System register	; 080h~0FFh of Bank 0 = To store system
		.	; registers (128 bytes)
	0FFh	End of bank 0 area	
BANK 15	F00h	LCD RAM area	; Bank 15 = To store LCD display data
		.	; (12 bytes)
	F0Bh	End of LCD Ram	;

2.1.5 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	-	-	R	Z	Y	-	PFLAG	RBANK	-	LCDM1	LCDM2	-	-	-	-	-
9	VREG	AMPM1	AMPM2	ADCM1	ADCM2	LBTM	ADCDH	ADCDM	ADCDL	-	-	-	-	-	-	-
A	ROMADRH	ROMADRL	ROMDAH	ROMDAL	ROMCNT	-	-	-	-	-	-	-	-	-	-	
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	-	P1M	-	-	-	-	-	-	INTRQ	INTEN	OSCM	-	WDTR	-	PCL	PCH
D	P0	P1	-	-	-	-	-	-	T0M	T0C	-	-	-	-	-	STKP
E	-	P1UR	-	-	-	-	-	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.1.5.2 SYSTEM REGISTER DESCRIPTION

R = Working register and ROM look-up data buffer
PFLAG = ROM page and special flag register
LCDM1 = LCD mode register
VREG = Voltage Regulators control register
AMPM2 = PGIA mode selection register2
ADCM2 = ADC control register2
ADCDH = ADC high-byte data buffer
ADCDL = ADC low-byte data buffer
ROMCNT = ISP ROM Counter
P_NUR = Port N pull-up register
INTRQ = Interrupt request register
OSCM = Oscillator mode register
PCH, PCL = Program counter
T0C = Timer 0 counting register
@YZ = RAM YZ indirect addressing index pointer

Y, Z = Working, @YZ and ROM addressing register
RBANK = Bank selection register
LCDM2 = LCD mode register
AMPM1 = PGIA mode selection register1
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_N = Port N data buffer
INTEN = Interrupt enable register
WDTR = Watchdog timer register
T0M = Timer 0 mode register
STK0~STK7 = Stack 0 ~ stack 7 buffer

2.1.5.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Name
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	-	-	-	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	-	-	RBNKS0	R/W	RBANK
089H	LCDREF	LCDPENB	LCDBNK	LCDTYPE	LCDENB	LCDBIAS	LCDRATE	LCDCLK	R/W	LCDM1
08AH	-	-	-	VPPINTL	VCP3	VCP2	VCP1	VCP0	R/W	LCDM2
090H	BGRENB	ACMSEL	ACMENB	AVESEL	AVENB	AVDDRENB	-	-	R/W	VREG
091H	CHS2	CHS1	CHS0	-	GS2	GS1	GS0	AMPENB	R/W	AMPM1
092H	UGBH	GX	GR	AMPCKS	-	-	DTENB	DTSEL		AMPM2
093H	RVS	IRVS2	IRVS1	IRVS0	ADGN2	ADGN1	ADGN0	ADCENB	R/W	ADCM1
094H	ADCKS	OSR2	OSR1	OSR0	ADCKSDIV	OFSEL1	OFSEL0	DRDY	R/W	ADCM2
095H	-	-	P11IO	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	-	-	-	-	R	ADCDL
0A0H	VPPCHK	-	-	-	-	ROMADR10	ROMADR9	ROMADR8	R/W	ROMADRH
0A1H	ROMADR7	ROMADR6	ROMADR5	ROMADR4	ROMADR3	ROMADR2	ROMADR1	ROMADR0	R/W	ROMADRL
0A2H	ROMDA15	ROMDA14	ROMDA13	ROMDA12	ROMDA11	ROMDA10	ROMDA9	ROMDA8	R/W	ROMDAH
0A3H	ROMDA7	ROMDA6	ROMDA5	ROMDA4	ROMDA3	ROMDA2	ROMDA1	ROMDA0	R/W	ROMDAL
0A4H	–	–	–	–	–	–	ROMCNT1	ROMCNT0	W	ROMCNT
0B8H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0BFH	-	-	-	P00G1	P00G0	-	-	-	R/W	PEDGE
0C1H	-	-	-	-	-	-	P11M	P10M	R/W	P1M
0C8H	ADCIRQ	-	-	T0IRQ	-	-	-	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	-	-	T0IEN	-	-	-	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	-	-	-	-	-	PC10	PC9	PC8	R/W	PCH
0D0H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	-	-	-	-	-	-	P11	P10	R/W	P1
0D8H	T0EN	T0RATE2	T0RATE1	T0RATE0	-	-	-	-	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DFH	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0	R/W	STKP
0E1H	-	-	-	-	-	-	P11R	P10R	W	P1UR
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	-	-	-	-	-	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	-	-	-	-	-	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	-	-	-	-	-	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	-	-	-	-	-	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	-	-	-	-	-	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	-	-	-	-	-	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	-	-	-	-	-	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	-	-	-	-	-	S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ ***Note:***

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

2.1.6 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
```

☞ 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
                …
                …
```

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
        …
```

2.1.9 R REGISTERS

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

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

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

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

2.2 ADDRESSING MODE

2.2.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.2.2 DIRECTLY ADDRESSING MODE

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

Example: Move 0x12 RAM location data into ACC.

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

Example: Move ACC data into 0x12 RAM location.

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

2.2.3 INDIRECTLY ADDRESSING MODE

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

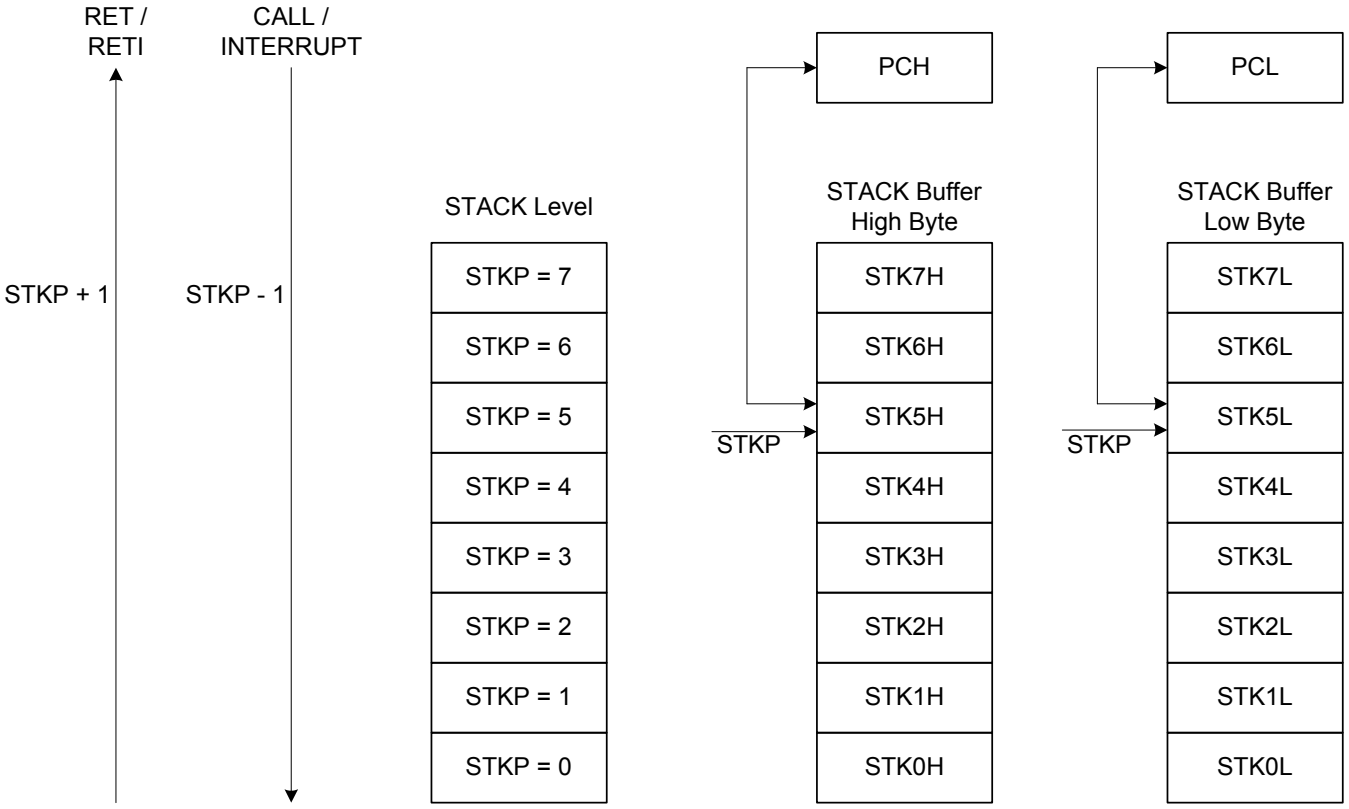
- **Example: Indirectly addressing mode with @YZ register.**

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

2.3 STACK OPERATION

2.3.1 OVERVIEW

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



2.3.2 STACK REGISTERS

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

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

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

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

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

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

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

```
MOV     A, #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

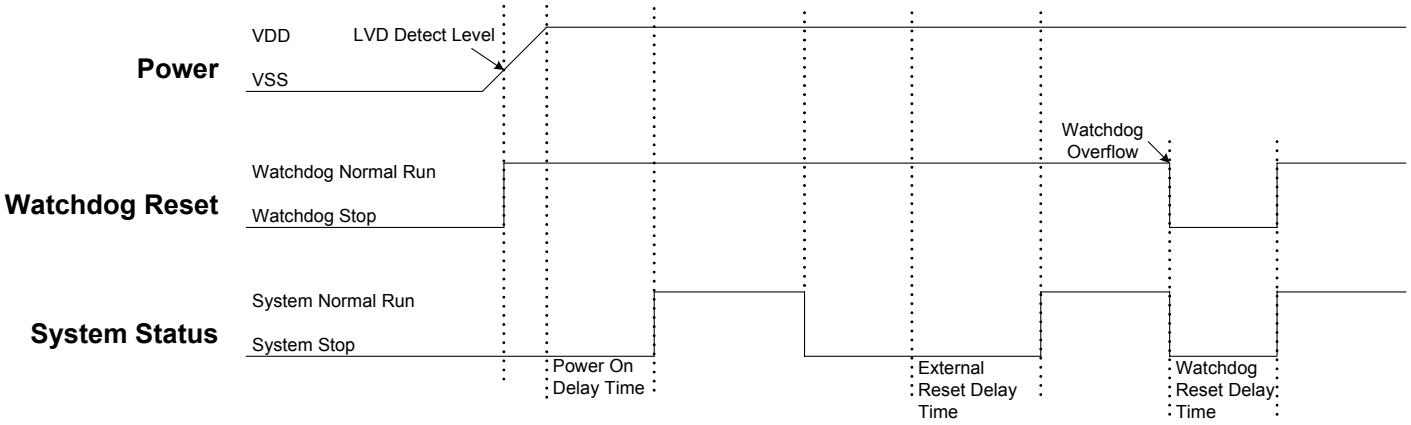
3.1 OVERVIEW

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.

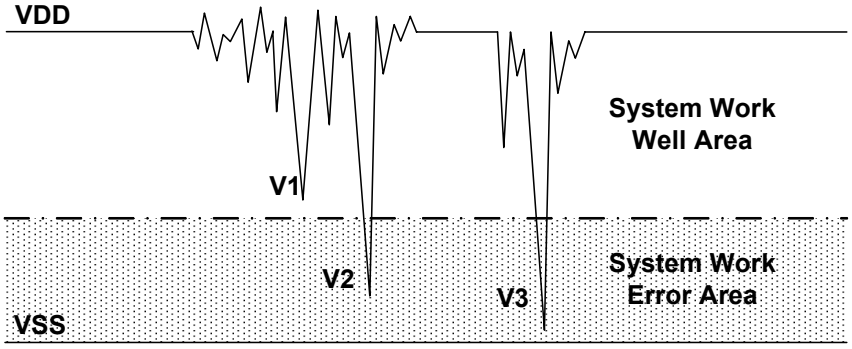
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

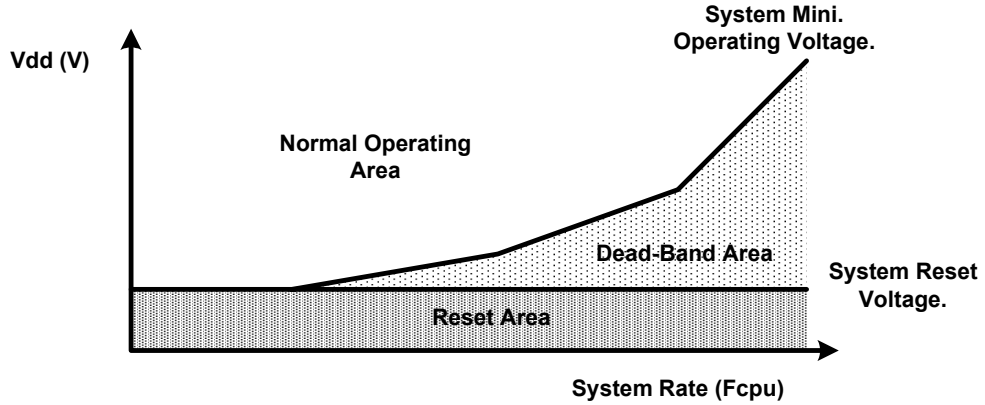
AC application:

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.



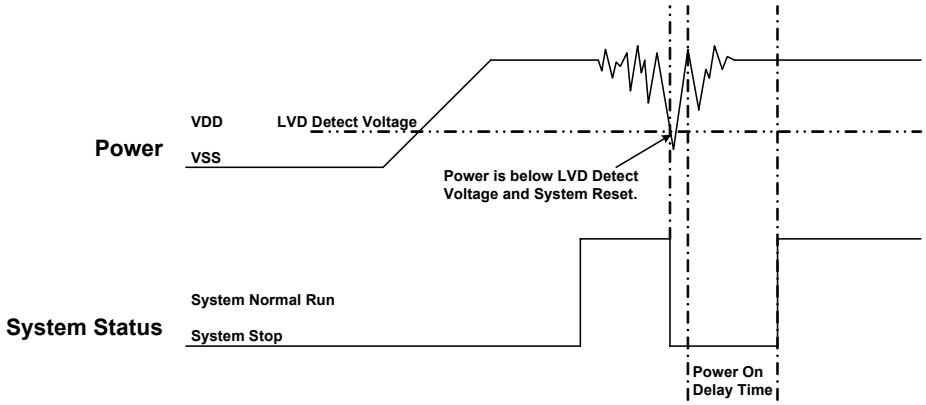
Normally the system operation voltage area is higher than the system reset voltage to VDD, and the reset voltage is decided by LVD detect level. The system minimum operating voltage rises when the system executing rate upper even higher than system reset voltage. The dead-band definition is the system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

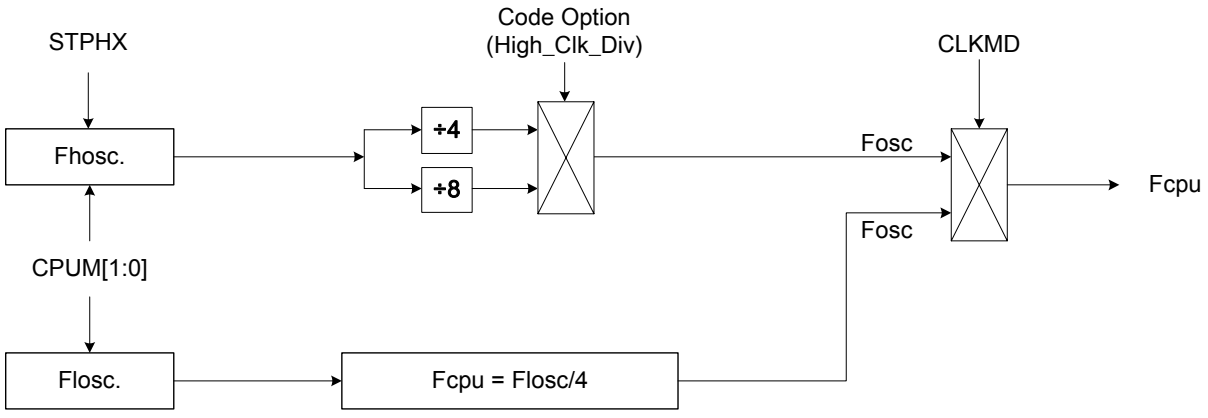
The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is only generated from internal 4MHz high-speed RC oscillator circuit (IHRC 4MHz). The low-speed clock is generated from internal RC oscillator circuit

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

☞ **Normal Mode (High Clock):** **Fcpu = Fhosc / 4 = 1MHz**. (Fhosc= 4MHz IHRC)
Fcpu = Fhosc / 8 = 500 kHz. (Fhosc= 4MHz IHRC)

☞ **Slow Mode (Low Clock):** **Fcpu = Flosc/4.** (Flosc= 32kHz ILRC)

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).
- 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 = External high-speed oscillator free run.
1 = External high-speed oscillator free run stop. Internal low-speed RC oscillator is still running.

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

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

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

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

➢ **Example: When entering the power down mode (sleep mode), both high-speed oscillator and internal low-speed oscillator will be stopped.**

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

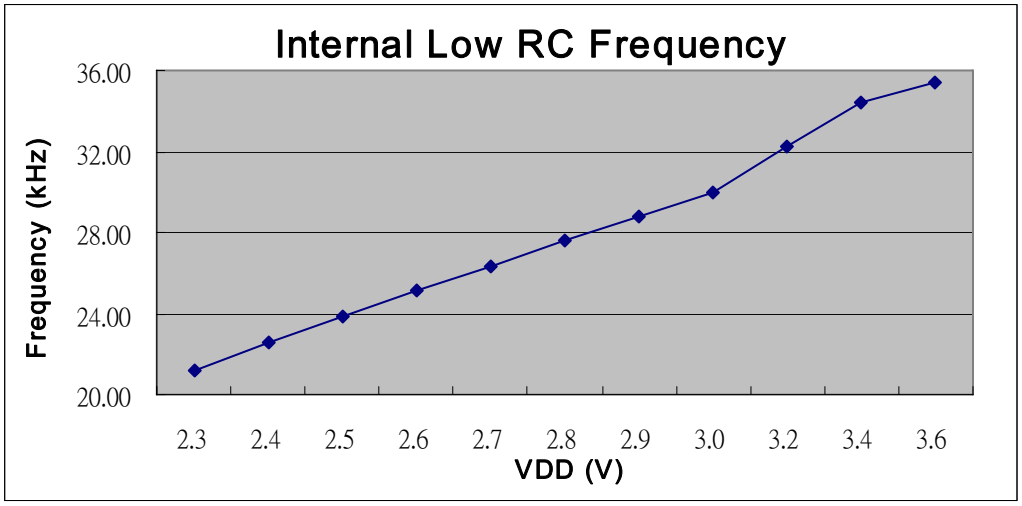
4.4 SYSTEM HIGH CLOCK

The system high clock is only from internal 4MHz oscillator RC type (IHRC). The system clock in normal mode is divided by 4 or 8, 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 3.2V. 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@3.2V).***

☞ ***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 oscillator 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.**

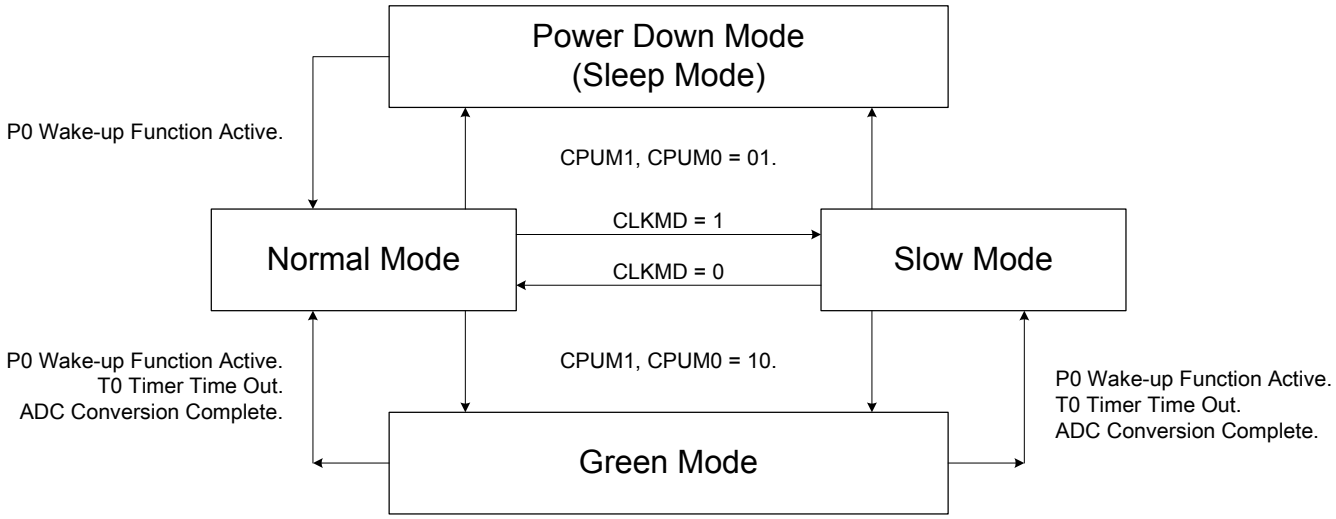
```
        B0BSET    P1M.0       ; Set P1.0 to be output mode for outputting Fcpu toggle signal.
@@:
        B0BSET    P1.0        ; Output Fcpu toggle signal in low-speed clock mode.
        B0BCLR    P1.0        ; Measure the Fcpu frequency by oscilloscope.
        JMP       @B
```

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 T0ENB=1
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, ADC	T0, *ADC	T0, *ADC	All inactive	*Active if high clock still running. (STPHX=0)
External interrupt	P00	P00	P00	P00	
Wakeup source	-	-	P0, T0, *ADC	P0	*Active if high clock still running. (STPHX=0)

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_1: 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.***

✸ ***Note_2: ADC***

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

```
; Set T0 timer wakeup function.
        B0BCLR      FT0IEN          ; To disable T0 interrupt service
        B0BCLR      FT0ENB          ; To disable T0 timer
        MOV         A,#20H          ;
        B0MOV       T0M,A           ; To set T0 clock = Fcpu / 64
        MOV         A,#74H
        B0MOV       T0C,A           ; To set T0C initial value = 74H (To set T0 interval = 10 ms)

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

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

✸ ***Note: During the green mode with T0 wake-up function, the wakeup pins and T0 can wakeup the system back to the last mode. T0 wake-up period is controlled by program and T0ENB 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 ADCENB=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.***

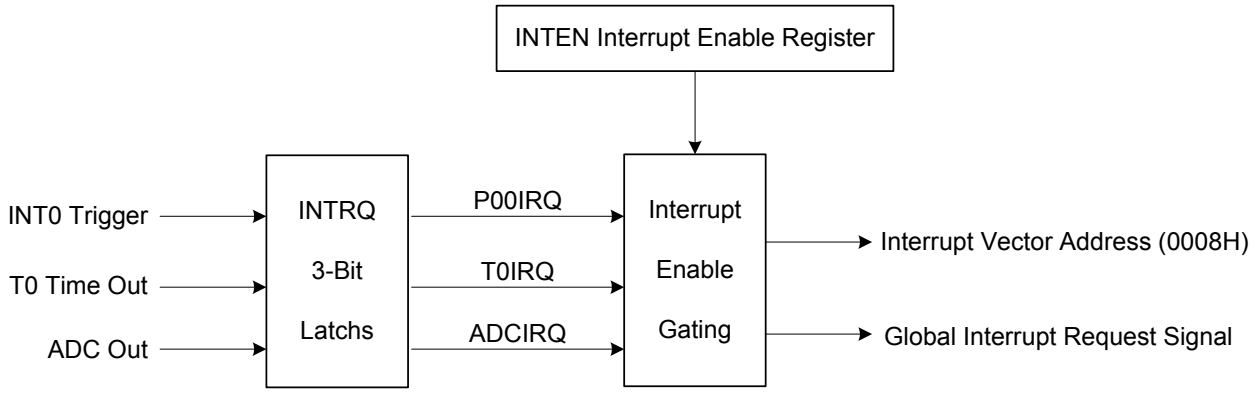
■ **Example: The system is waked up from power down (sleep mode) by P0 level chande. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.**

The wakeup time = 1/Fhosc * 64 = 16us (Fhosc = 4MHz)
The total wakeup time = 16us + oscillator start-up time (5us)

6 INTERRUPT

6.1 OVERVIEW

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



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

6.2 INTEN INTERRUPT ENABLE REGISTER

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

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

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

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

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	-	-	T0IRQ	-	-	-	P00IRQ
Read/Write	R/W	-	-	R/W	-	-	-	R/W
After reset	0	-	-	0	-	-	-	0

Bit 0 **P00IRQ:** External P0.0 interrupt (INT0) request flag.
0 = Non INT0 interrupt request.
1 = INT0 interrupt request.

Bit 4 **T0IRQ:** T0 timer interrupt request flag.
0 = Non T0 interrupt request.
1 = T0 interrupt request.

Bit 7 **ADCIRQ:** ADC interrupt request flag.
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.

➢ **Example: Set global interrupt control bit (GIE).**

```
B0BSET      FGIE        ; Enable GIE
```

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

6.5 PUSH, POP ROUTINE

When any interrupt occurs, system will jump to ORG 8 and execute interrupt service routine. It is necessary to save ACC, PFLAG data. The chip doesn't have any special instructions to process ACC, PFLAG registers when into interrupt service routine. Users have to save ACC, PFLAG by program, Using "B0XCH" to save/load ACC buffer, "B0MOV" to save/load PFLAG and avoid main routine error after interrupt service routine finishing.

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

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

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

.CODE
            ORG       0
            JMP       START

            ORG       8
            JMP       INT_SERVICE

            ORG       10H
START:
            …

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

            RETI                    ; Exit interrupt service vector
            …
            ENDP
```

6.6 INT0 (P0.0) INTERRUPT OPERATION

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

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

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

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

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

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

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

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

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

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

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

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

        RETI                    ; Exit interrupt vector
```

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
T0IRQ	T0C overflow
ADCIRQ	ADC converting end.

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

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

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

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

INTP00CHK:                      ; Check INT0 interrupt request
        B0BTS1  FP00IEN         ; Check P00IEN
        JMP     INTT0CHK        ; Jump check to next interrupt
        B0BTS0  FP00IRQ         ; Check P00IRQ
        JMP     INTP00
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
INTADCHK:                       ; Check ADC interrupt request
        B0BTS1  FADCIEN         ; Check ADCIEN
        JMP     INT_EXIT        ; Jump to exit of IRQ
        B0BTS0  FADCIRQ         ; Check ADCIRQ
        JMP     INTADC          ; Jump to ADC interrupt service routine
INT_EXIT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

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

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

✷ ***Note:***
1. ***Users can program them by bit control instructions (B0BSET, B0BCLR).***
2. ***Port 0 and Port 1 are bi-direction I/O port..***

➢ **Example: I/O mode selecting**

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

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

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

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

7.2 I/O PULL UP REGISTER

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

✷ ***Note: Pull up Resistance of Port 0 is always existence when Port 0 set as input mode.***

➢ **Example: I/O Pull up Register**

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

7.3 I/O PORT DATA REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	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

- **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”.
```

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

✷ ***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
```

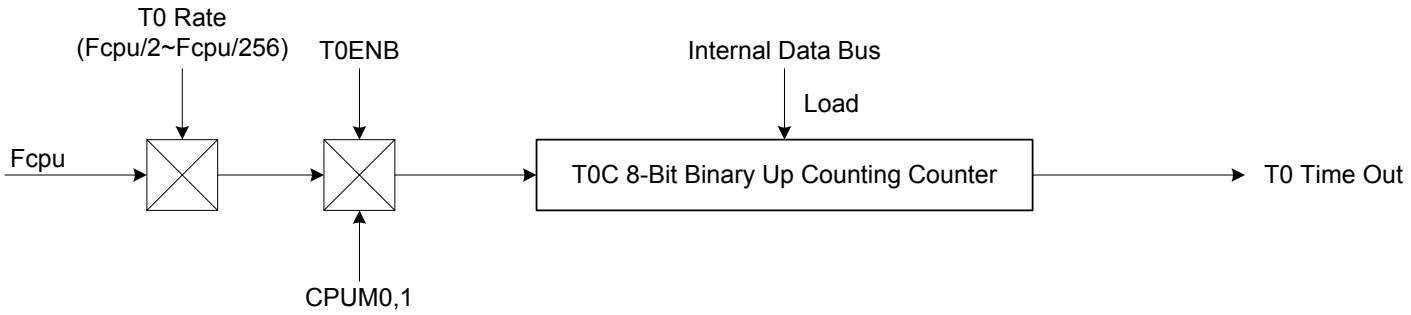
8.2 TIMER 0 (T0)

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



8.2.2 T0M MODE REGISTER

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

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

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

8.2.3 T0C COUNTING REGISTER

T0C is an 8-bit counter register for T0 interval time control.

0D9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0C	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of T0C initial value is as following.

T0C initial value = 256 - (T0 interrupt interval time * input clock)

- **Example: To set 10ms interval time for T0 interrupt. High clock is 4MHz IHRC. Fcpu=Fosc/4. Select T0RATE=010 (Fcpu/64).**

*T0C initial value = 256 - (T0 interrupt interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 64)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 64)*
= 100
= 64H

The basic timer table interval time of T0.

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

8.2.4 T0 TIMER OPERATION SEQUENCE

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

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

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

☞ **Set T0 timer rate.**

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

☞ **Set T0 interrupt interval time.**

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

☞ **Set T0 timer function mode.**

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

☞ **Enable T0 timer.**

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

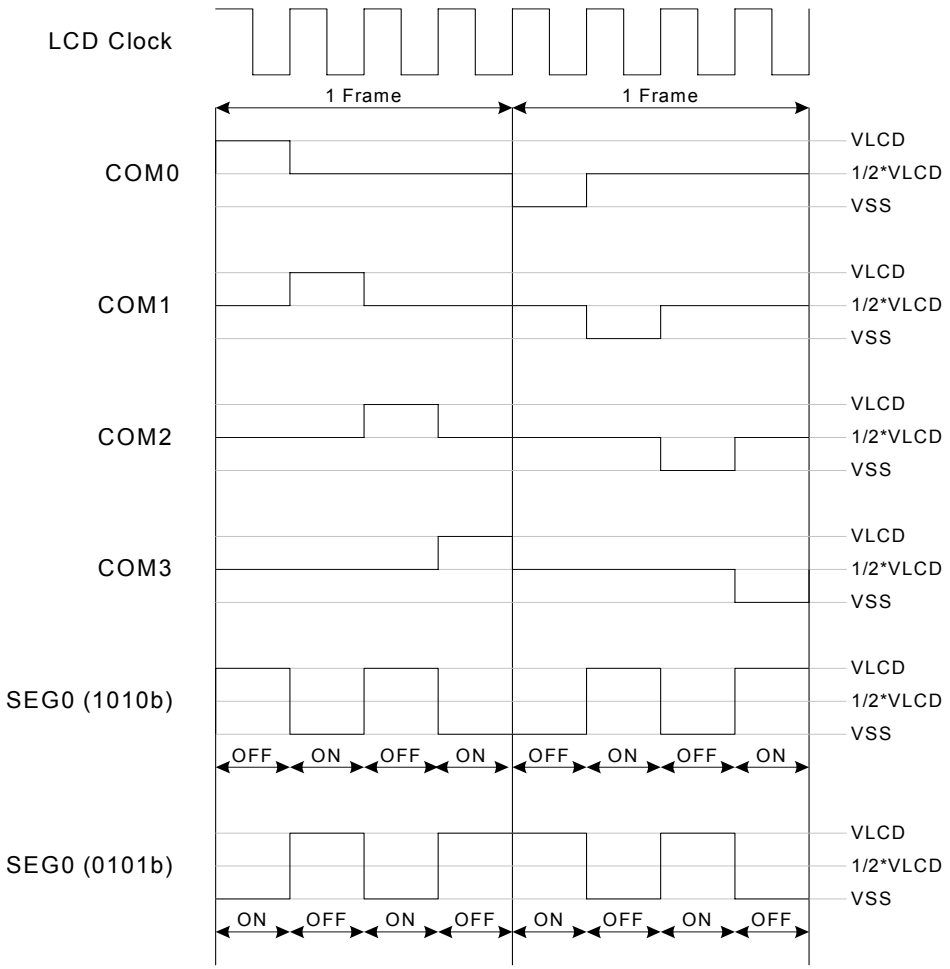
9 LCD DRIVER

LCD driver includes R-type and C-type structures with 4 common pins and 12segment pins in the SN8P2947. 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 48 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.

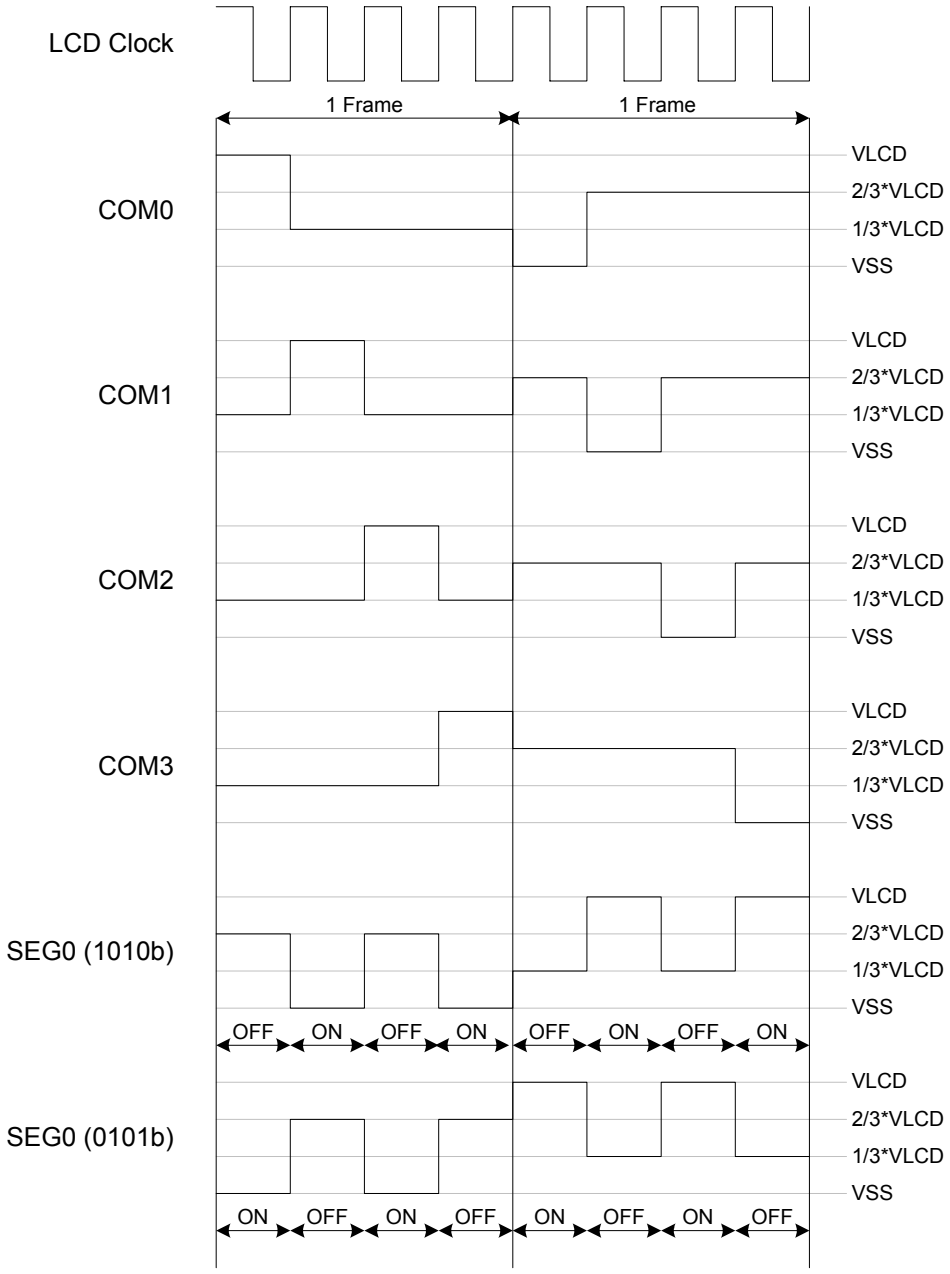
9.1 LCD TIMING

LCD Timing Table

LCDCLK	Clock Source	LCDRATE	LCD Frame Rate	C-Type CP clock	Note
0	IHRC	X	IHRC / (2^16) = 61Hz	IHRC/ 64 = 62.5kHz	IHRC= 4MHz
1	ILRC	0	ILRC / 512 = 64Hz@3V	ILRC = ~32kHz@3.3V	ILRC = ~32kHz@3.3V
1	ILRC	1	ILRC / 256 = 128Hz@3V	ILRC = ~32kHz@3.3V	**CP** = Charge Pump.



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



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

9.2 LCDM1 REGISTER

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

Bit0 **LCDCLK:** LCD clock source selection control bit.
0 : LCD Frame Rate = IHRC/(2^16)= 61Hz.
C-Type CP clock = IHRC/64 = 62.5kHz.
1 : LCD Frame Rate = ILRC/512 = 64Hz (LCDRATE=0), or ILRC/256 =128Hz (LCDRATE=1).
C-Type CP clock = ILRC = ~32kHz.

Bit1 **LCDRATE:** LCD clock rate control when LCDCLK=1 (LCD clock source from ILRC).
0 = LCD Frame Rate = ILRC / 512 = 64Hz.
1 = LCD Frame Rate = ILRC / 256 = 128Hz.

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.

Bit4 **LCDTYPE:** R-Type / C-Type LCD driver control bit.
0 = R-Type.
1 = C-Type.

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

Bit6 **LCDPENB:** C-Type LCD charge pump enable bit.
0 = C-Type LCD charge pump disable.
1 = C-Type LCD charge pump enable.

Bit7 **LCDREF:** R-Type LCD resistor selection bit for LCD bias voltage division
0 = 100k bias resistor. (Reserved option, VLCD/V3/V2 must connect 0.1uF capacitors to ground.)
1 = 33.3k bias resistor.
(Please Set LCDREF= 1 in R-Type LCD drive mode.)

R-Type and C-Type LCD driver control:

	LCDPENB	LCDTYPE	LCDENB
R-Type setting	0	0	1
C-Type setting	1	1	1
LCD disable setting for power saving (In Green or Sleep mode)	0	X	0

✸ ***Note1: LCD disable in green or sleep mode, LCDENB and LCDPENB are set "0" for power saving.)***
✸ ***Note2: In C-Type LCD start-up procedure, we recommend to set LCDPENB=LCDTYPE=1 with delay 5ms before LCDENB set "1".***

9.3 LCDM2 REGISTER

08AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM2	-	-	-	VPPINTL	VCP3	VCP2	VCP1	VCP0
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	-	0	0	0	1	1

Bit[0:3] **VCP[3:0]:** C-Type VLCD output voltage selection bits.
(When LCDEPNB = LCDTYPE = 1, LCD charge pump start pumping)

	LCD 1/3 bias Condition			**LCD 1/2 bias Condition**		
VCP[3:0]	**V2**	**V3**	**VLCD**	**V2**	**V3**	**VLCD**
0000	0.73V	1.46V	2.2V	1.10V	1.10V	2.2V
0001	0.93V	1.86V	2.8V	1.40V	1.40V	2.8V
0010	0.96V	1.93V	2.9V	1.45V	1.45V	2.9V
0011	1.00V	2.00V	3.0V	1.50V	1.50V	3.0V
0100	1.03V	2.06V	3.1V	1.55V	1.55V	3.1V
0101	1.06V	2.13V	3.2V	1.60V	1.60V	3.2V
0110	1.10V	2.20V	3.3V	1.65V	1.65V	3.3V
0111	1.13V	2.26V	3.4V	1.70V	1.70V	3.4V
1000~1110	Reserve			Reserve		
1111	2.1V	4.4V	4.6V	2.3V	4.6V	6.9V*

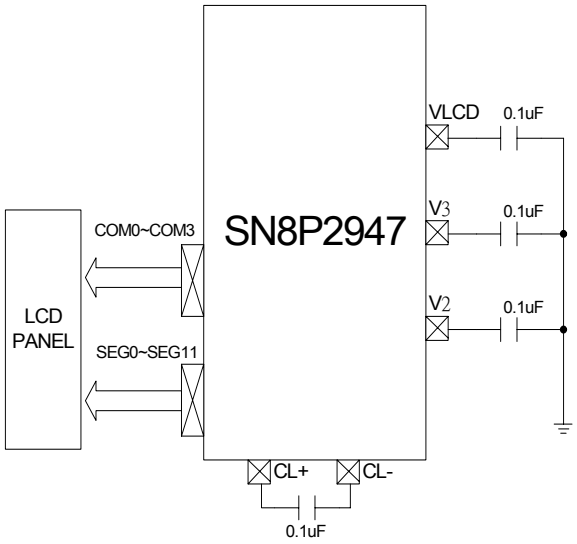
*: VLCD output 6.9V for ISP VPP voltage.

Bit[4] **VPPINTL:** Internal VPP control bits.
0 = VPP no voltage source.
1 = VPP short to VLCD internally for ISP function.
(C-Type LCD CP output high voltage and short to VPP Pin for ISP function.)

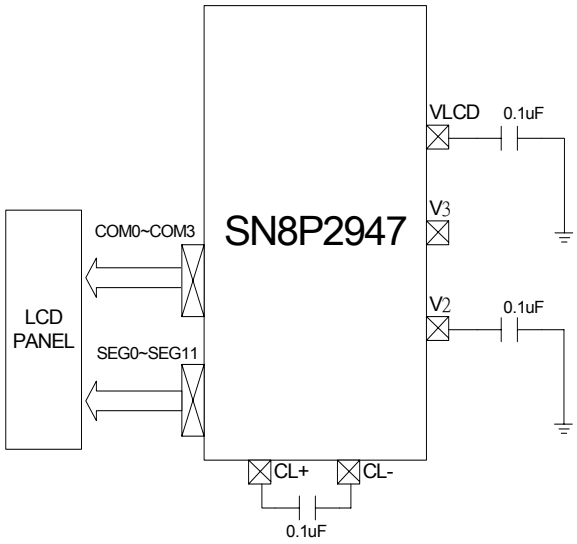
- ***Note_1: When VCP[3:0]=1111, LCDTYPE=1 and LCDPENB=1, LCD charge pump is applied to generating VPP voltage for ISP function without additional 6.5V input. Please don't enable LCDENB to protect LCD panel without high voltage present at COM and SEG.***
- ***Note_2:VCP and VPPINTL are controlled for ISP function without external VPP voltage 6.5V. Please reference chapter "IN-System Program ROM".***
- ***Note_3: if VLCD set high voltage output 6.9V for ISP purpose, VLCD/V3/V2 must connect capacitor to DVSS with 0.1uf individually.***
- ***Note_4: Macro "RomwrtVpp" instruction cover procedures of internal VPP generation and ROMWRT instruction for ISP function without external 6.5V requirement.***

9.4 C-TYPE LCD DRIVER MODE

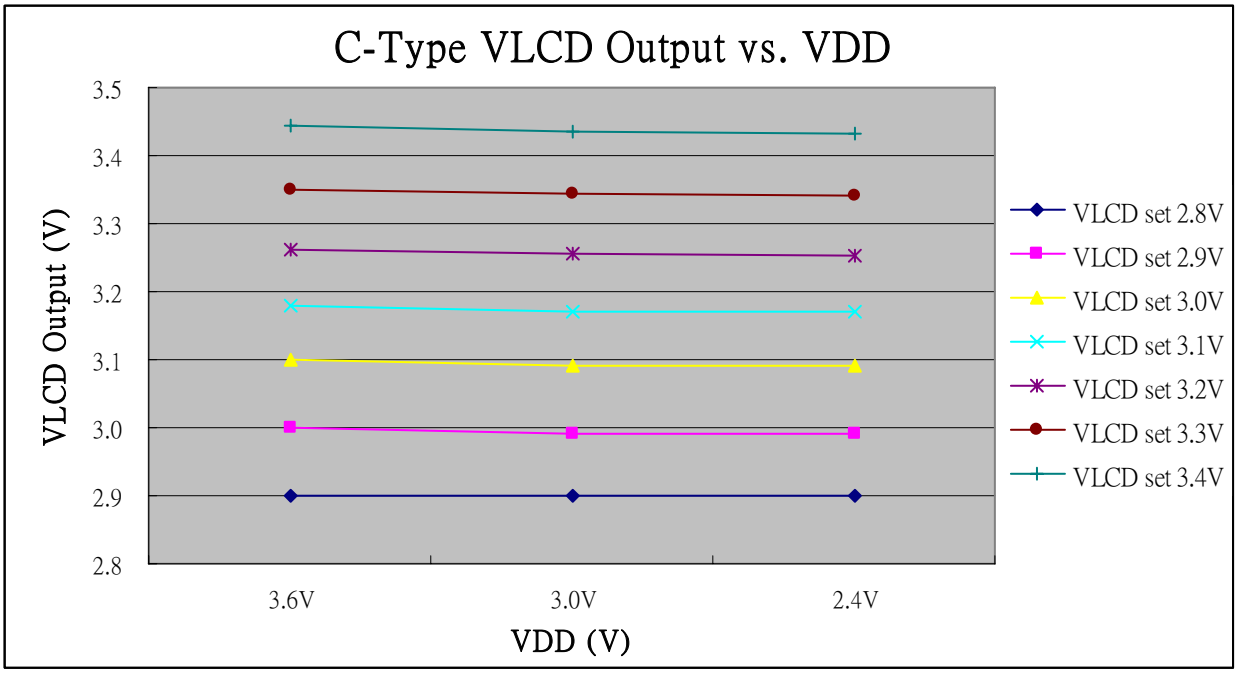
The LCD C-type driver mode is support 1/3 and 1/2 bias LCD panel. The LCD power (VLCD) is supplied by internal LCD charge-pump. The C-Type LCD charge-pump voltage level is following VLCD voltage. V2 is the charge pump source which level is 1/3*VLCD. V3 is 2 times of V2 by charge pump which level is 2/3*VLCD. In C-type LCD mode, the LCDTYPE and LCDPENB bit of LCDM1 register must be “1”. The following are shown the 1/3 and 1/2 bias C-Type LCD application circuit and VLCD output voltage chart.



1/3 Basic C-type LCD Application Circuit



1/2 Basic C-type LCD Application Circuit

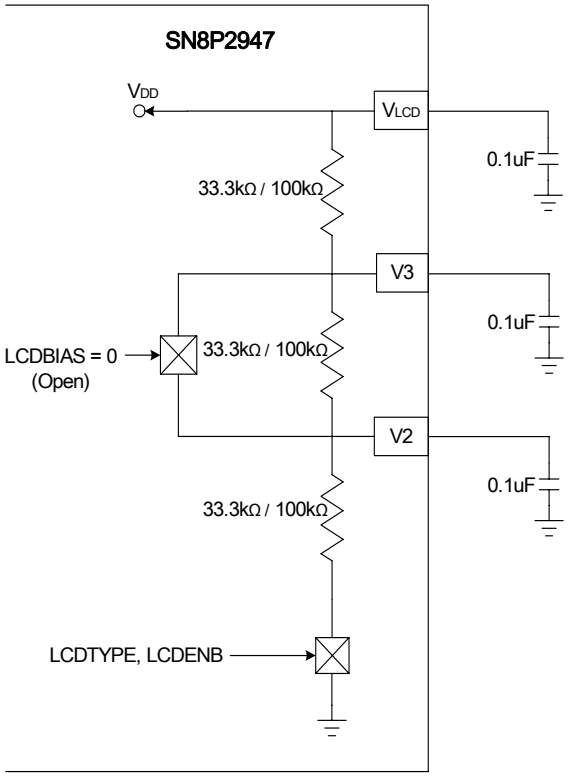


- ***Note1: In C-type mode, connect a 0.1uF capacitor between CL+ and CL- pins. The 0.1uF capacitor also connect pin VLCD/V3/V2 to VSS.***
- ***Note2: VLCD output voltage can be set from 2.8V to 3.4V and with ±0.2V accuracy.***

9.5 R-TYPE LCD DRIVER MODE

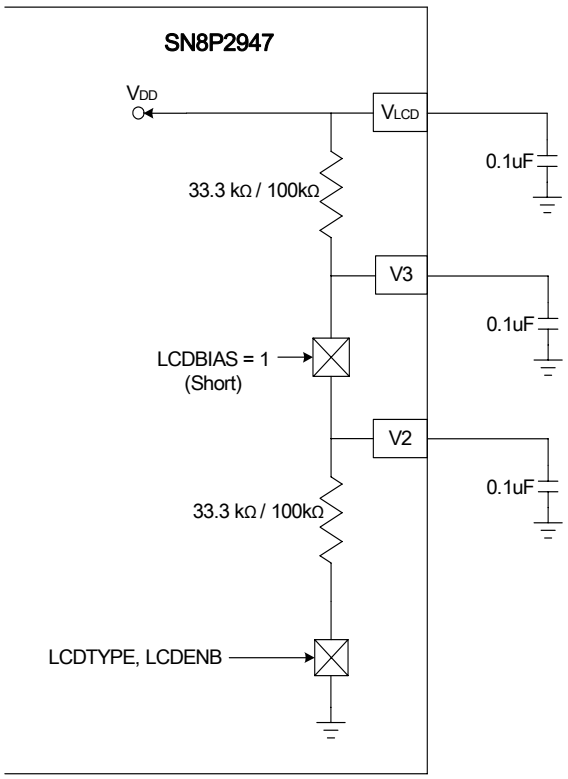
In LCD R-type driver, LCD power (VLCD) is auto connected to VDD via internal circuit. V3 and V2 bias voltage source from internal voltage division by resistors. The following diagram shows the connection of 1/4 duty with 1/3 bias and 1/2 bias.

1/4 duty with 1/3 bias:



R-Type LCD current consumption = $\frac{VLCD}{33.3k \times 3}$ or $\frac{VLCD}{100k \times 3}$

1/4 duty with 1/2 bias:



$$\text{LCD current consumption} = \frac{VLCD}{33.3k \times 2} \text{ or } \frac{VLCD}{100k \times 2}$$

- ***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.***
- ***Note_2: In R-Type LCD driver mode, please set LCDREF = 1.***

9.6 LCD RAM LOCATION

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

	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
	COM0	COM1	COM2	COM3	-	-	-	-
SEG 0	00H.0	00H.1	00H.2	00H.3	-	-	-	-
SEG 1	01H.0	01H.1	01H.2	01H.3	-	-	-	-
SEG 2	02H.0	02H.1	02H.2	02H.3	-	-	-	-
SEG 3	03H.0	03H.1	03H.2	03H.3	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
SEG 11	11H.0	11H.1	11H.2	11H.3	-	-	-	-

➢ **Example: Enable LCD function.**

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

R-Type LCD Setting:

```
B0BCLR      FLCDTYPE    ; R-Type LCD.
B0BCLR      FLCDPENB    ; CP disable.
B0BSET      FLCDENB     ; Enable LCD driver.
```

C-Type LCD Setting:

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

10 IN SYSTEM PROGRAM ROM

10.1 OVERVIEW

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

10.2 ROMADRH/ROMADRL REGISTER

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

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

VPPCHK: VPP pin Programming Voltage. Check
0 = VPP's Voltage NOT reached 6.5V. Can't program ISP ROM
1 = VPP's Voltage reached 6.5V. Can program ISP ROM

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

10.3 ROMDAH/ROMADL REGISTERS

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

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

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

10.4 ROMCNT REGISTERS and ROMWRT INSTRUCTION

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMCNT	-	-	-	-	-	-	ROMCNT1	ROMCNT0
Read/Write	-	-	-	-	-	-	W	W
After reset	-	-	-	-	-	-	0	0

Bit[7:0] ROMCNT**[1:0]:** ISP ROM Programming Time Counter
The ISP ROM Programming Time was controlled by ROMCNT [1:0]
The Suggestion Programming is 120us.

Fcpu	ROMCNT [1:0]	Programming Time
1 or 0.5 MIP	00	240us
1 or 0.5 MIP	01	120us
1 or 0.5 MIP	10	60us
1 or 0.5 MIP	11	30us

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

- ***Note1: Please Keep VDD=3V when accessing ISP ROM.***
- ***Note2: After access ROMWRT, at least 3 NOP instruction delay is necessary.***
- ***Note3: Please executing ISP function in room temperature(25°C)***
- ***Note4: Using Macro "RomwrtVpp" to execute ISP function without external VPP 6.5V input. VPP high voltage is generated from IC internally. In this condition, CL+-/VLCD/V3/V2 must connect 0.1uf capacitor individually.***
- ***Note5: Interrupt Function must be disabled before ISP execution.***

10.5 ISP ROM ROUTINE EXAMPLE

➢ **Example : ISP Example with External VPP 6.5V input:**

```
; Reserved ISP ROM Area as 0xFFFF
ORG             0100H
@CALDATA:
                DW              0xFFFF
                ……..
                ……..
; Program Data 0xAA55 into address @CALDATA
                MOV             A, #@CALDATA$L
                B0MOV           ROMADRL, A      ;Move Low Byte Address to ROMADRL
                MOV             A, #@CALDATA$H
                B0MOV           ROMADRH, A      ;Move Low Byte Address to ROMADRH
                MOV             A, #0X55
                B0MOV           ROMDAL, A       ;Move Low Byte Data to ROMDAL
                MOV             A, #0XAA
                B0MOV           ROMDAH, A       ;Move Low Byte Data to ROMADRH
;VPP Voltage Check
                @B0BTS1_ FVPPCHK                ;Check VPP Voltage is 6.5V or not
                JMP             $-1             ;If VPP not reach 6.5V, Keep waiting.

;Set programming counter and Accessing ISP ROM
@ROM_WRT:       MOV             A,#1            ;Set Programming Counter
                B0MOV           ROMCNT,A
                B0BCLR          FGIE            ;Interrupt disable before ISP execution.
                ROMWRT                          ;Programming ISP ROM
                NOP                             ;NOP Delay
                NOP                             ;NOP Delay
                NOP                             ;NOP Delay
                B0BSET          FGIE            ;Interrupt enable if necessary.
;VPP Voltage Check                               ;Set VPP as VDD voltage.
                @B0BTS0_ FVPPCHK                ;Check VPP Voltage is VDD or not
                JMP             $-1             ;If VPP still reach 6.5V, Keep waiting.

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

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

                ……..
```

➢ **Example : ISP Example with Internal VPP generation:**

- Calibration data in RAM “Cal_Data[8]”. (8-Bytes)
- ISP ROM Address from 0x0700H to 0x0703H (4-words)
- Using Macro “RomwrtVpp”.

```
ISP_Internal:
            MOV         A, #07H
            B0MOV       ROMADRH, A          ; Initial ISP ROM Address "0x0700H"
            CLR         ROMADRL             ; Initial ISP ROM Address "0x0700H"
            CLR         ROMCNT              ; Set ISP Program Max. Time 240us

;------ Load ISP Data address from RAM "ISP_Data" ----------
            MOV         A, #Cal_Data$L      ; Calibration Data in RAM "Cal_Data[8]"
            B0MOV       Z, A
            CLR         Y

;------ fetch Cal_data into [ROMDAH,ROMDAL] --------------------
@@:         B0MOV       A, @YZ
            B0MOV       ROMDAH, A
            INCMS       Z
            B0MOV       A, @YZ
            B0MOV       ROMDAL, A

;------ Store Register Y and Z ----------------------------------------
            B0MOV       A, Z
            B0MOV       Z_buffer, A
            B0MOV       A, Y
            B0MOV       Y_buffer, a

;------ [ ISP ROM Write Command ] ------------------------------------
            B0BCLR      FGIE                ; Disable Interrupt.
            RomwrtVpp                       ; ISP ROM Write Macro Instruction
            B0BSET      FGIE                ; Enable Interrupt if necessary.

;------ Restore Register Y and Z --------------------------------------
            B0MOV       A, Z_buffer
            B0MOV       Z, A
            B0MOV       A, Y_buffer
            B0MOV       Y, A

            INCMS       ROMADRL             ; Next ISP ROM Address
            MOV         A, #4               ; ISP Address [0x0700H, 0x0701H, 0x0702H, 0x0703H]
            CMPRS       A, ROMADRL
            JMP         @b

;------ [ ISP End ] ---------------------------------------------------------
            Call        ISP_ROM_Check       ; Check ISP ROM Data correct or not
```

11 Regulator, PGIA and ADC

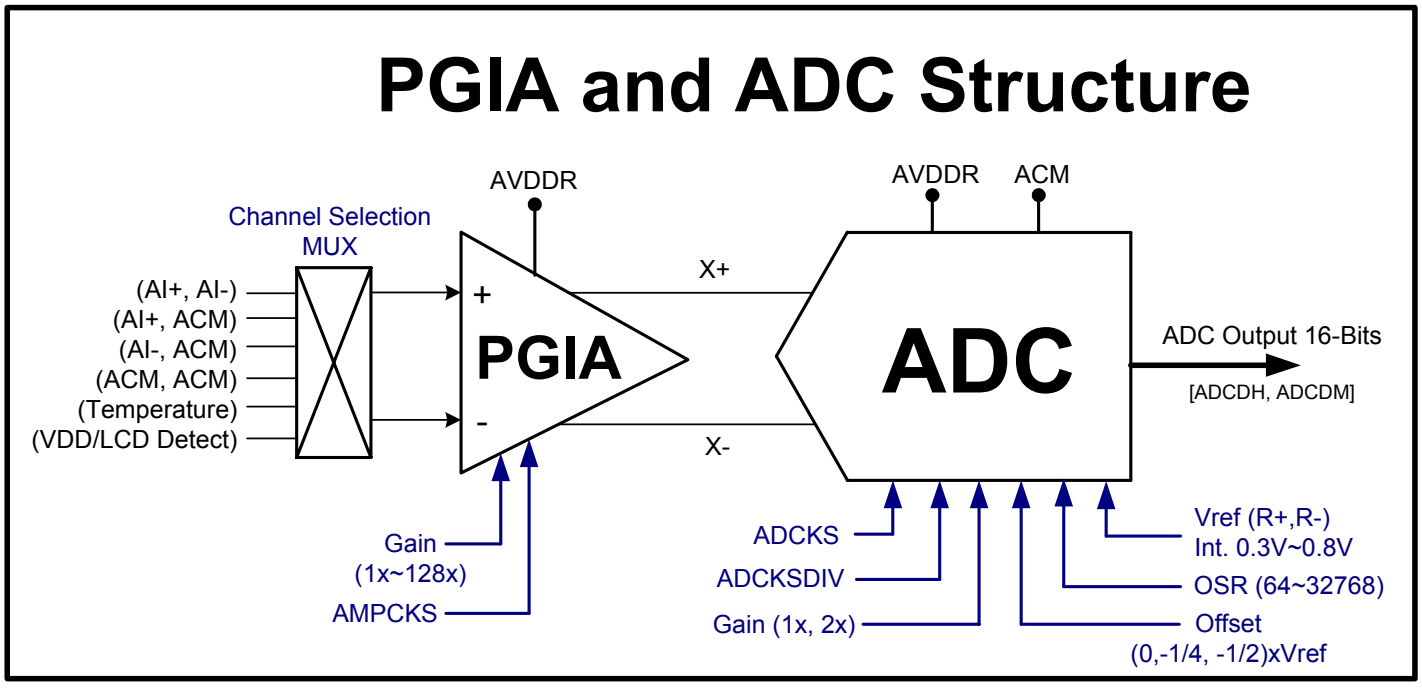
11.1 OVERVIEW

The SN8P2947 has a built-in Voltage Regulator to support a stable voltage 2.4V from pin AVDDR and 1.5V/2.0V from pin AVE+ with maximum 5mA current driving capacity. The AVDDR provides stable voltage for internal circuits (PGIA, ADC) and external sensor (load cell or thermistor). The SN8P2947 series also integrated Δ Σ Analog-to-Digital Converters (ADC) to achieve 16-bit performance and up to 2^16-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 2 types of input channel modes: (1) One fully differential input. (2) Two single-ended inputs. This ADC is optimized for measuring low-level unipolar or bipolar signals in weight scale and medical applications. A very low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selectable gains of 1x, 16x, 32x, 64x, and 128x in the ADC to accommodate these applications.

11.2 ANALOG INPUT

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

To obtain maximum range of ADC output, the ADC maximum input signal voltage should be close to but can't over the reference voltage V(R+, R-), Choosing a suitable reference voltage and a suitable gain of PGIA can reach this purpose. The relative control bits are RVS and IRVS bits (Reference Voltage Selection) in ADCM1 register and GS[2:0] bits (Gain Selection) in AMPM register.



Block Diagram of ADC module

11.3 Voltage Regulator

SN8P2947 is built in voltage regulators, which can provide a stable 2.4V (pin AVDDR) and 1.5V/2.0V (pin AVE+) with maximum 5mA current driving capacity. Register CPM can enable or disable AVDDR, AVE+ and ACM 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.

11.3.1 Voltage Regulator Control Register

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VREG	BGRENB	ACMSEL	ACMENB	AVESEL	AVENB	AVDDRENB	-	-
R/W	R/W	R/W	R/W	R/W	R/W	R/W	-	-
After Reset	0	1	0	1	0	0	-	-

Bit2: **AVDDRENB:** Regulator (AVDDR) voltage enable control bit.
0 = Disable AVDDR regulator output voltage.
1 = Enable AVDDR regulator output voltage

Bit3: **AVENB:** AVE+ voltage output control bit.
0 = Disable AVE+ output voltage.
1 = Enable AVE+ output voltage.

Bit4: **AVESEL**: AVE+ voltage selection control bit.
0 = AVE+ output 1.5V.
1 = AVE+ output 2.0V.

Bit5: **ACMENB:** Analog Common Mode (ACM) voltage enable control bit.
0 = Disable Analog Common Mode and ACM output voltage.
1 = Enable Analog Common Mode and ACM output voltage.

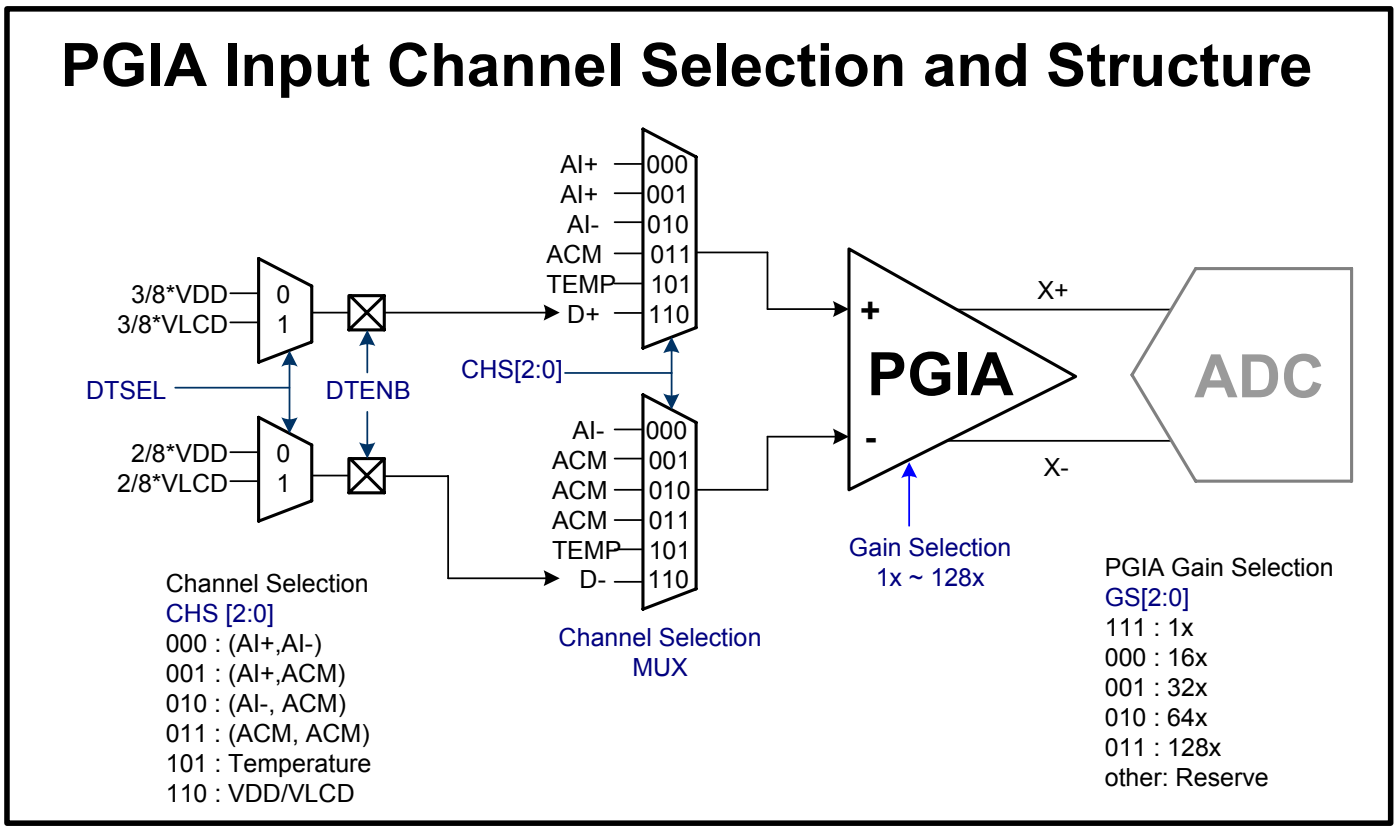
Bit6: **ACMSEL:** ACM Voltage selection bit.
0 = Analog Common-mode voltage ACM output 0.75V.
1 = Analog Common-mode voltage ACM output 1V.

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

- ***Note_1: Band Gap Reference voltage must be enable (FBRGENB), before following function accessing:***
  (Reference AMPM1 and AMPM2 register for detail information)
  (1) Regulators of AVDDR, AVE+ and ACM.
  (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/ACM, to avoid VDD drop in CR2032 battery application.***

11.4 PGIA -Programmable Gain Instrumentation Amplifier

SN8P2947 includes a low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selection gains of 1x, 16x, 32x, 64x, and 128x controlled by register AMPM1. The PGIA also provides two types channel selection mode: (1) One fully differential input (2) Two single-ended inputs, it was defined by register AMPM1.



11.4.1 AMPM1- Amplifier Mode1 Control Register

091H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPM1	CHS2	CHS1	CHS0	-	GS2	GS1	GS0	AMPENB
R/W	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
After Reset	1	0	1	-	1	1	1	0

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

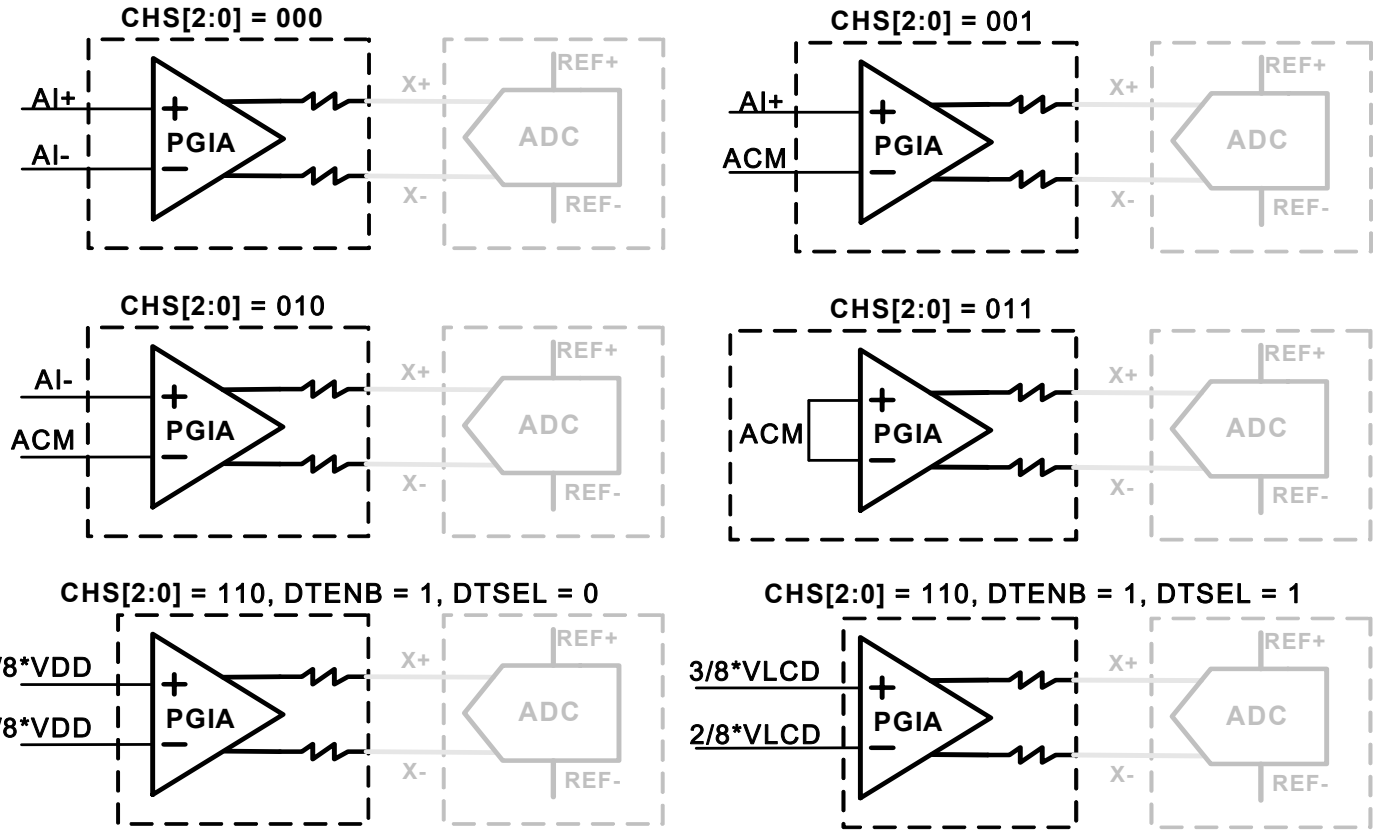
Bit[3:1]: **GS [2:0]:** PGIA Gain Selection control bit.

GS [2:0]	PGIA Gain
000	16
001	32
010	64
011	128
100,101,110	Reserved
111	1

Bit[7:5]: **CHS [2:0]:** PGIA Channel Selection.

PGIA Channel Selection Table:

CHS [2:0]	Selected Channel	ADC Input	Input-Signal Type
000	AI+, AI-	V (AI+, AI-) × PGIA Gain	Differential
001	AI+, ACM	V (AI+, ACM) × PGIA Gain	Single-ended
010	AI-, ACM	V (AI-, ACM) × PGIA Gain	Single-ended
011	ACM, ACM	V (ACM, ACM) × PGIA Gain	Input-Short
101	Temperature sensor	V(VTS, VREF-)	N/A
110	Voltage Detection	1/8VDD × PGIA Gain 1/8VLCD, × PGIA Gain	Differential
100,111	Reserved	-	-



- ***Note_1: V (AI+, AI-) = (AI+ voltage − AI- voltage).***
- ***Note_2: V (AI-, ACM) = (AI- voltage − ACM voltage).***
- ***Note_3: The purpose of Input-Short mode is only for PGIA offset testing.***
- ***Note_4: When PGIA Gain set 1x (GS0[2:0]=111) application, the AI+/AI- signal will bypass PGIA and input ADC directly. PGIA can be disabled (AMPENB=0) for power saving, and input buffer of ADC must be enabled (GX=1) for input high impedance characteristic of ADC.***

11.4.2 AMPM2- Amplifier Mode2 Control Register

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPM2	UGBH	GX	GR	AMPCKS	-	-	DTENB	DTSEL
R/W	R/W	R/W	R/W	R/W	-	-	R/W	R/W
After Reset	0	0	0	1	-	-	0	0

Bit0: **DTSEL**: VDD/VLCD voltage detect function control bit.
0 = Select VDD voltage detect function.
1 = Select VLCD voltage detect function.

Bit1: **DTENB**: VDD/VLCD voltage detect function enable bit.
0 = Disable VDD/VLCD voltage detect function.
1 = Enable VDD/VLCD voltage detect function.

Bit4: **AMPCKS**: PGIA Chopper frequency selection bit.
0 = 15.6 kHz or 10.4kHz.
1 = 31.25 kHz or 20.8kHz. (For ADC high conversion rate application.)

Bit5: **GR**: R+ R+ Unit Gain Buffer enable bit.
0 = Disable R+ R- UGB function. (When ADC Vref set from internal.)
1 = Enable R+ R- UGB function. (When ADC Vref set from external pin R+ and R-.)

Bit6: **GX**: X+ X+ Unit Gain Buffer enable bit.
0 = Disable X+ X- UGB function. (when PGIA Gain set x16, x32, x64, or x128.)
1 = Enable X+ X- UGB function. (when PGIA Gain set x1.)

Bit7: **UGBH:** Unit Gain Buffer Enhance control bit.
0 = Disable UGB Enhance function.
1 = Enable UGB Enhance function.
(Please set “0”, function not available.)

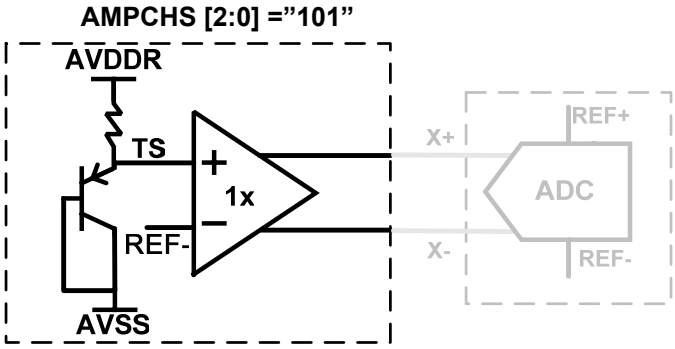
ADC clock and PGIA chopper clock selection table:

AMPCKS	ADCKS	ADCKSDIV	ADC Clock	Chopper clock
0	0	0	250 kHz	15.625kHz
0	0	1	125 kHz	
0	1	0	166 kHz	10.4kHz
0	1	1	83 kHz	
1	0	0	250 kHz	31.25kHz
1	0	1	125 kHz	
1	1	0	166 kHz	20.83kHz
1	1	1	83 kHz	

✷ ***Note: In general application, please set AMPCKS = 1.***

11.5 Temperature Sensor (TS)

In applications, sensor characteristic might change in different temperature also. To get the temperature information, SN8P2947 build in a temperature senor (TS) for temperature measurement. Select the respective PGIA channel to access the Temperature Sensor ADC output.



REF- Voltage table:

RVS	IRVS[2:0]	Vref (AVE 2V)	REF-	
			AVE=2V	AVE=1.5V
1	000	0.3V	0.85V	0.64V
1	001	0.4V	0.80V	0.60V
1	010	0.5V	0.75V	0.56V
1	011	0.6V	0.70V	0.53V
1	100	0.7V	0.65V	0.49V
1	101	0.8V	0.60V	0.45V
1	110*	Reserve		
1	111*			
0	xxx	(R+) - (R-)	R-	

- ***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 REF-.***
- ***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 0.72 typically, and if temperature rise 10℃, V(TS) will decrease about 17.5mV, if temperature drop 10℃ , V(TS) will increase about 17.5mV.

Example:

Temperature	V(TS)	Vref = (REF+ - REF-)	REF-	ADC output
15℃	0.740V	0.6V	0.7V	2184
25℃	0.723V	0.6V	0.7V	1256
35℃	0.706V	0.6V	0.7V	327

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

- ***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: -1.75mV/C was typical temperature parameter only sensor, every single chip was different to each other.***

Example: PGIA setting (Fhosc = 4MHz IHRC)

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

@ACM_Enable:
            B0BSET      FACMSEL         ; Set ACM output 1V.
            B0BSET      FACMENB         ; Enable ACM Voltage=1v

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

@AVE_Enable:
            B0BSET      FAVESEL         ; Set AVE+ output 2.0V
            B0BSET      FAVENB          ; Enable AVE+ Voltage

@PGIA_Setting:
            MOV         A, #00000110B
            B0MOV       AMPM1, A        ; V (X+, X-) Output = V (AI+, AI-) x 128
            MOV         A, #00010000B
            B0MOV       AMPM2, A        ; PGIA chopper Freq. 31.25kHz.

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

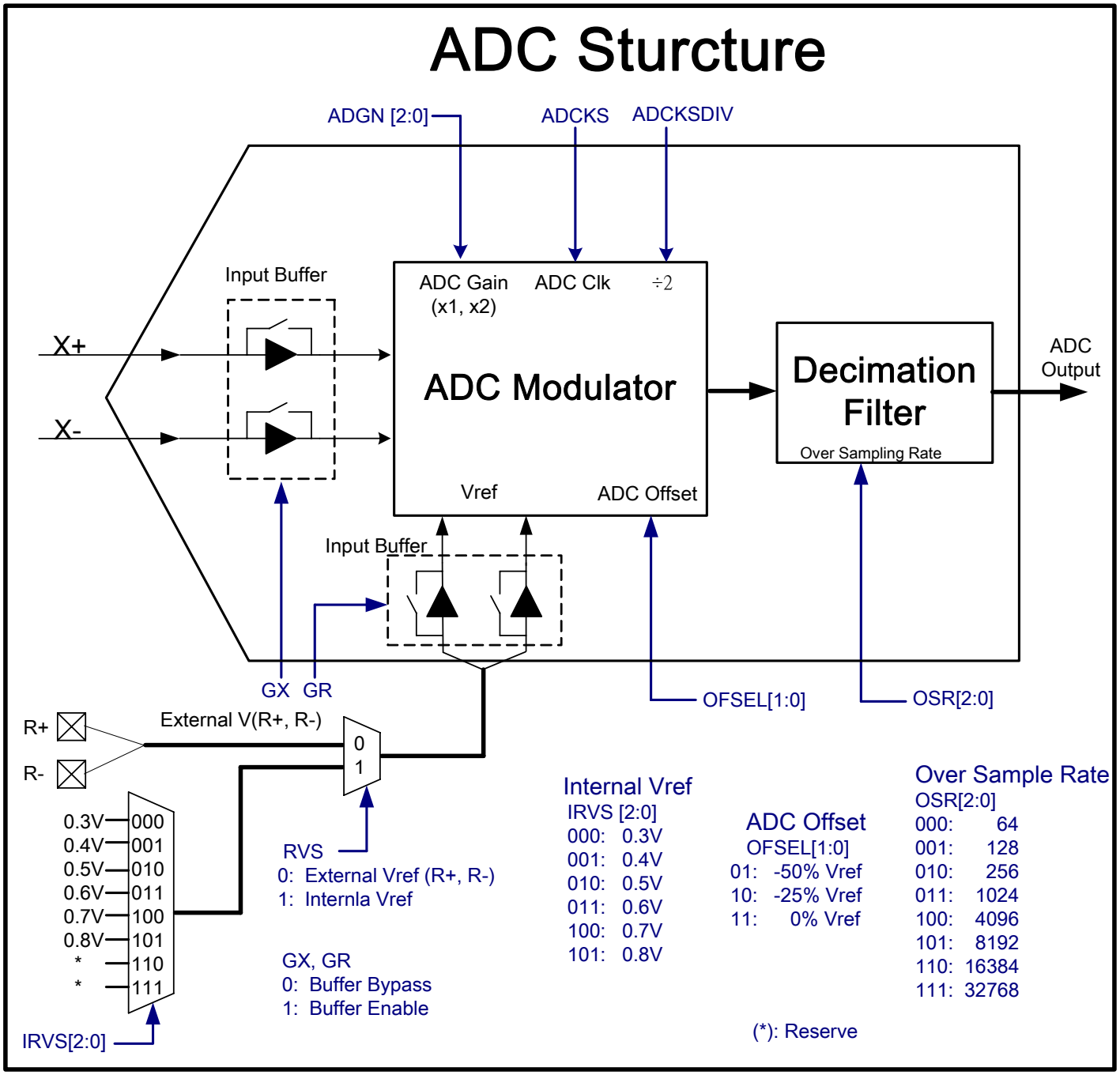
- ***Note_1: Enable AVDDR Regulator before PGIA working***

Example: PGIA channel change:

```
@PGIA_Setting:
            MOV         A, #00000110B
            B0MOV       AMPM1, A        ; V (X+, X-) Output = V (AI+, AI-) x 128
            MOV         A, #00010000B
            B0MOV       AMPM2, A        ; PGIA chopper Freq. 31.25kHz.

@PGIA_Enable:   B0BSET      FAMPENB     ; Enable PGIA function
            …
            …
@PGIA_single-end:
            MOV         A, #00100111B   ; Don't Disable PGIA when change PGIA Channel.
            B0MOV       AMPM1, A        ; Selected PGIA as Single-end channel
            …                           ; V (X+, X-) Output = V(AI+,ACM) x 128
            …
@PGIA_VDD:
            MOV         A, #11001111B   ; Don't Disable PGIA when change PGIA Channel. PGIA Gain 1x
            B0MOV       AMPM1, A        ; Selected PGIA as Voltage detection channel
            B0BCLR      FDTSEL          ; Select VDD Voltage Detect function.
            B0BSET      FDTENB          ; Enable Voltage Detect function.
            …                           ; V (X+, X-) Output = 1/8*VDD x 1
```

11.6 16-Bit Analog to Digital Converter (ADC)

The SN8P2947 integrated a 16-bit ΔΣ Analog-to-Digital Converters (ADC) with decimation filters can be set for variable throughputs range from 2.5Hz up to 3.9 kHz. A reference voltage (Vref) is built in internal with selective range from 0.3V to 0.8V in AVE=2V 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.



11.6.1 Analog Inputs and Voltage Operation Range

There are four analog inputs for ADC and PGIA operation, including pins of AI+, AI-, R+, and R-. The analog inputs of PGIA, AI+/AI-, are connected to external sensor's output signal, which can be configured as differential mode (AI+ to AI-) or single-end (AI± to ACM). 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 0.3V to 1.4V. Moreover, the output signals of PGIA, X+/X-, must also remain within the absolute voltage range.

11.6.2 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 **RVS** and **IRVS[2:0]** bits in register **ADCM1**. When **RVS** bit is set to '1', the ADC uses a internal Vref source which can be selected value from 0.3V~0.8 with 0.1V step via setting **IRVS[2:0]** bits. When **RVS** bit is cleared to '0', the Vref is from external and the value is decided by differential voltage between Pins of **R+** and **R-**. Detail setting reference register **ADCM1**.

11.6.3 Input Buffer

Input Buffers are included ADC signal input buffer and ADC external reference input buffer R+/R-, which provide a high impedance of analog input, to minimized the input current of ADC for sensitive measurement and to avoid loading effect. When PGIA set 1x of application, the sensor output signal is bypass PGIA and direct connected to ADC's input. In that case, Input buffer function must be enabled by setting GX bit as "1". If external Vref is selected for ADC, input buffer R+/R- also must be enabled by setting GR bit as "1".

11.6.4 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:

ADC output code (differential mode)：

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

Voffset: 0, -1/4 x Vref, and -1/2 x Vref
PGIA: 1x ~ 128x
ADC_Gain: 1x and 2x
Vref Source: Internal Vref or External Vref
Vref Range: 0.3V ~ 0.8V

- ***Note_1: When ADC offset function set -1/2*Vref, the ADC ENOB will drop 0.3~0.5 Bit, compare with ADC No offset condition.***
- ***Note_2: When ADC offset function set -1/4*Vref, the ADC ENOB will drop 0.1~0.3 Bit, compare with ADC No offset condition.***

11.6.5 Output Word Rate

Delta-Sigma ADC provides variable output word rate (WR) from 2.5 Hz up to 3.9 kHz, which output word rate is decided by setting bits of **ADCKS, ADCKSDIV** 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	ADCKSDIV	OSR [2:0]	ADC clock	WR	ADCKS	ADCKSDIV	OSR [2:0]	ADC clock	WR
0	0	000	250KHz	3.9 kHz	1	0	000	166KHz	2.6KHz
0	0	001		1.95 kHz	1	0	001		1.3KHz
0	0	010		976 Hz	1	0	010		651Hz
0	0	011		244 Hz	1	0	011		163Hz
0	0	100		61 Hz	1	0	100		40.7Hz
0	0	101		30.5 Hz	1	0	101		20.3Hz
0	0	110		15.2 Hz	1	0	110		10.2Hz
0	0	111		7.6 Hz	1	0	111		5.1Hz

ADCKS	ADCKSDIV	OSR [2:0]	ADC clock	WR	ADCKS	ADCKSDIV	OSR [2:0]	ADC clock	WR
0	1	000	125KHz	1.95KHz	1	1	000	83KHz	1.3KHz
0	1	001		976Hz	1	1	001		651Hz
0	1	010		488Hz	1	1	010		325Hz
0	1	011		122Hz	1	1	011		81.4Hz
0	1	100		30.5Hz	1	1	100		20.3Hz
0	1	101		15.3Hz	1	1	101		10.2Hz
0	1	110		7.6Hz	1	1	110		5.1Hz
0	1	111		3.8Hz	1	1	111		2.5Hz

11.6.6 ADCM1- ADC Mode1 Register

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCM1	RVS	IRVS2	IRVS1	IRVS0	ADGN2	ADGN1	ADGN0	ADCENB
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	1	1	0	1	0	0	1	0

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

Bit[3:1] ADGN[2:0]**: ADC Gain Selection bits**
001 = ADC Gain x 1.
010 = ADC Gain x 2.
Others: Reserve.

Bit[6:4]: **IRVS[2:0]:** ADC Internal Reference Voltage Selection.

RVS	IRVS[2:0]	ADC Vref	
		AVE= 2V	AVE=1.5V
1	000	0.3V	0.225V
1	001	0.4V	0.300V
1	010	0.5V	0.375V
1	011	0.6V	0.450V
1	100	0.7V	0.525V
1	101	0.8V	0.600V
1	110	Reserve	
1	111		
0	xxx	(R+) - (R-)	

Bit7: **RVS:** ADC Reference Voltage Internal/External Selection bit.
0 = Selection ADC Reference voltage from **External** reference R+, R-.
1 = Selection ADC Reference voltage from **Internal** reference.

- ***Note_1: the Operation range of ADC Reference Voltage (Vref) is from 0.3V to 0.8V.***
- ***Note_2: Vref(Int.) means ADC reference voltage form internal setting; Vref(Ext.) means ADC reference voltage from external (R+ and R- input).***

11.6.7 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	ADCKSDIV	OFSEL1	OFSEL0	DRDY
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	1	1	1	0	1	1	0

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

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

➢

OFSEL[1:0]	ADC Offset
01	-1/2 x Vref
10	-1/4 x Vref
11	No Offset
Others	Reserve

Bit[3] **ADCKSDIV:** ADC clock division bit.
0 = Original ADC clock rate.
1 = ADC clock rate divide by 2.

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

Bit7: **ADCKS:** ADC Clock source selection Bit.
"0" = ADC clock source set 250 kHz.
"1" = ADC clock source set 166 kHz.

ADCKS	ADCKSDIV	ADC Clock Rate
0	**0**	250 KHz
0	**1**	125 KHz
1	**0**	166 KHz
1	**1**	83KHz

- ***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.***

11.6.8 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	-	-	-	-
R/W	R	R	R	R	-	-	-	-
After Reset	0	0	0	0	-	-	-	-

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*
0x7FFFH	32767
…	…
0x4000H	16384
…	…
0x1000H	4096
…	…
0x0002H	2
0x0001H	1
0x0000H	0
0xFFFFH	-1
0xFFFEH	-2
…	…
0xF000H	-40966
…	…
0xC000H	-16384
…	…
0x8000H	-32768

- ***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 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 0x7FFF.***
- ***Note 5: The Negative Full-Scale-Output value of ADC conversion is 0x8000H.***
- ***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 SN8P2947 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 0.8V 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	32768	7.6Hz	16.4	19.1	18.5 uV	2.80 uV
16 x 1	32768	7.6Hz	16	18.7	1.53 uV	0.231 uV
32 x 1	32768	7.6Hz	15.9	18.6	0.818 uV	0.124uV
64 x 1	32768	7.6Hz	15.6	18.3	0.503 uV	0.076 uV
128 x 1	32768	7.6Hz	15	17.7	0.381 uV	0.058 uV
128 x 2	32768	7.6Hz	14.1	16.8	0.356 uV	0.054 uV
128 x 1	16384	15.3Hz	14.6	17.3	0.503 uV	0.076 uV
128 x 1	8192	31Hz	14.1	16.8	0.712 uV	0.108 uV
128 x 1	4096	61Hz	13.6	16.3	1.01 uV	0.153 uV
128 x 1	1024	244Hz	12.6	15.3	2.01 uV	0.305 uV
128 x 1	256	977 Hz	11.5	14.2	4.32 uV	0.654 uV
128 x 1	128	2.0 kHz	10.8	13.5	7.01 uV	1.06 uV
128 x 1	64	3.9 kHz	9.5	12.2	17.26 uV	2.61 uV

All Test condition: ADC 250kHz, Input-Short, Vref=0.8V, 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=0.8V, 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 4MHz)

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

@ACM_Enable:      B0BSET      FACMSEL          ; Set ACM output 1V
                  B0BSET      FACMENB          ; Enable ACM Voltage

@AVE_Enable:      B0BSET      FAVESEL          ; Set AVE+ output 2.0V
                  B0BSET      FAVENB           ; Enable AVE+ Voltage

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

@PGIA_Init:       MOV         A, #00000110B
                  B0MOV       AMPM1, A         ; PGIA differential channel (AI+, AI-) and PGIA Gain x 128
                  MOV         A, #00010000B
                  B0MOV       AMPM2, A         ; Set unit Gain buffer off and PGIA chopper 31.25KHz
                  B0BSET      FAMPENB          ; Enable PGIA function
                                               ; V (X+, X-) Output = V (AI+, AI-) x 128

@ADC_Init:        MOV         A, #11010010B
                  B0MOV       ADCM1, A         ; ADC Reference voltage = internal 0.8V, 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/VLCD Voltage Detection:

```
@CPREG_Init:    B0BSET    FBGRENB     ;Enable Band Gap Reference voltage
@ACM_Enable:    B0BSET    FACMSEL     ; Set ACM output 1V
                B0BSET    FACMENB     ; Enable ACM Voltage
@AVE_Enable:    B0BSET    FAVESEL     ; Set AVE+ output 2.0V
                B0BSET    FAVENB      ; Enable AVE+ Voltage
@AVDDR_Enable:  B0BSET    FAVDDRENB   ; Enable AVDDR Voltage=2.4V

@VDD_Detection:
@PGIA_Init:     MOV       A, #00001000B
                B0MOV     AMPM1, A    ; PGIA differential channel (AI+, AI-) and PGIA Gain x 1
                MOV       A, #00010010B ; Set VDD detection function
                B0MOV     AMPM2, A    ; Set unit Gain buffer off and PGIA chopper 31.25KHz
                B0BSET    FAMPENB     ; Enable PGIA function
                                      ; V (X+, X-) Output = V (AI+, AI-) x 1
@ADC_Init:      MOV       A, #11010010B
                B0MOV     ADCM1, A    ; ADC Reference voltage = internal 0.8V, 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
                …
                …
                …
@VLCD_Detection:
@PGIA_Init:     MOV       A, #00001000B
                B0MOV     AMPM1, A    ; PGIA differential channel (AI+, AI-) and PGIA Gain x 1
                MOV       A, #00010011B ; Set VLCD detection function
                B0MOV     AMPM2, A    ; Set unit Gain buffer off and PGIA chopper 31.25KHz
                B0BSET    FAMPENB     ; Enable PGIA function
                                      ; V (X+, X-) Output = V (AI+, AI-) x 1
@ADC_Init:      MOV       A, #11010010B
                B0MOV     ADCM1, A    ; ADC Reference voltage = internal 0.8V, 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
@ACM_Enable:    B0BSET      FACMSEL         ; Set ACM output 1V
                B0BSET      FACMENB         ; Enable ACM Voltage
@AVE_Enable:    B0BSET      FAVESEL         ; Set AVE+ output 2.0V
                B0BSET      FAVENB          ; Enable AVE+ Voltage
@AVDDR_Enable:  B0BSET      FAVDDRENB       ; Enable AVDDR Voltage=2.4V

@VDD_Detection:
@PGIA_Init:     MOV         A, #00001000B
                B0MOV       AMPM1, A        ; PGIA differential channel (AI+, AI-) and PGIA Gain x 1
                MOV         A, #00010000B   ; Set VDD detection function
                B0MOV       AMPM2, A        ; Set unit Gain buffer off and PGIA chopper 31.25KHz
                B0BSET      FAMPENB         ; Enable PGIA function
                                            ; V (X+, X-) Output = V (AI+, AI-) x 1
@ADC_Init:      MOV         A, #11010010B
                B0MOV       ADCM1, A        ; ADC Reference voltage = internal 0.8V, 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

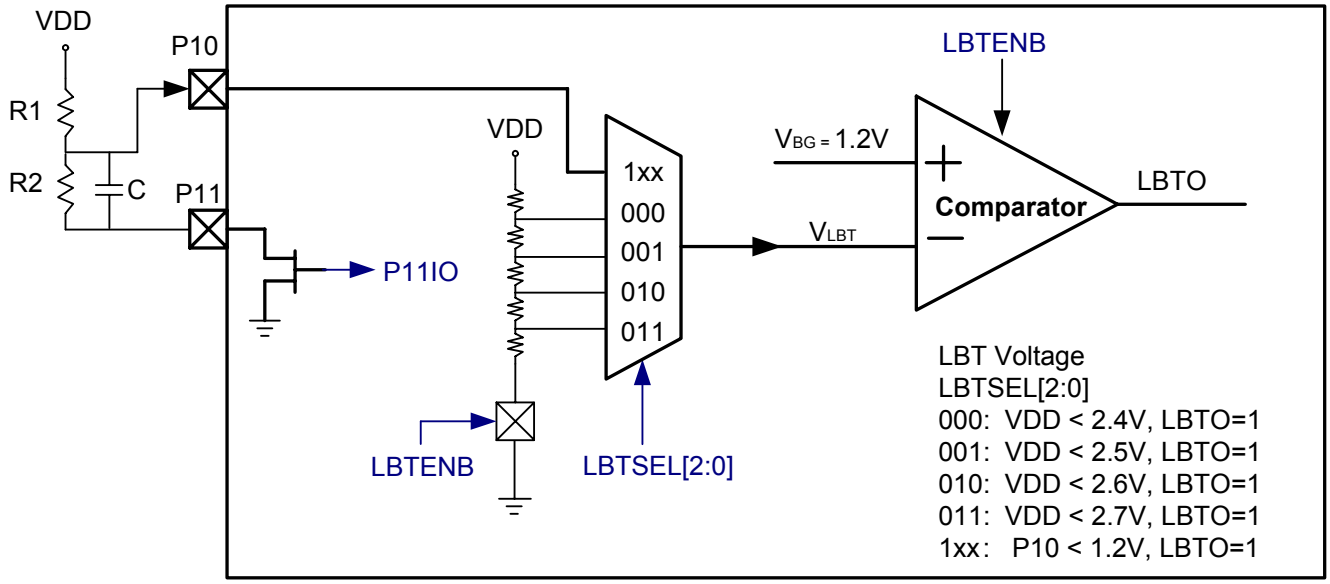
```
Green_Mode_1:   // Wakeup By ADC ready, T0 overflow and P0 level change.
                MOV         A, #11110000
                B0MOV       T0M, A          ; T0 overflow and wakeup every 512us. (Fcpu = 1MIP)
                MOV         A, #00100110B
                B0MOV       ADCM2, A        ; 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      FDTENB          ; Disable VDD/VLCD detect function.
                B0BCLR      FAMPENB         ; Disable PGIA function.
                B0BCLR      FADCENB         ; Disable ADC function.
                B0BCLR      FAVENB          ; Disable AVE.
                B0BCLR      FAVDDRENB       ; Disable AVDDR.
                B0BCLR      FACMENB         ; Disable ACM.
                B0BCLR      FLCDPENB        ; Disable C-Type LCD charge pump.
                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      FDTENB      ; Disable VDD/VLCD detect function.
                B0BCLR      FAMPENB     ; Disable PGIA function.
                B0BCLR      FADCENB     ; Disable ADC function.
                B0BCLR      FAVENB      ; Disable AVE.
                B0BCLR      FAVDDRENB   ; Disable AVDDR.
                B0BCLR      FACMENB     ; Disable ACM.
                B0BCLR      FLCDPENB    ; Disable C-Type LCD charge pump.
                B0BCLR      FLCDENB     ; Disable LCD display.
                B0BCLR      FBGRENB     ; Disable Band Gap Voltage.
                SleepMode               ; System into Sleep mode. (Macro)
```

- ***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.***

11.7 LBTM: Low Battery Detect

SN8P2947 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, 2.4V, 2.5V, 2.6V, 2.7V, 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.



11.7.1 LBTM: Low Battery Detect Register

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

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

Bit1: **LBTO:** Low Battery Detect Output Bit.
0 = LBT voltage (V_{LBT}) Higher than Band Gap Reference Voltage 1.2V.
1 = LBT voltage (V_{LBT}) Lower than Band Gap Reference Voltage 1.2V.

Bit[4:2]: **LBTSEL[2:0]:** Low Battery Detect threshold voltage selection bit.

LBTENB	LBTSEL [2:0]	LBTO = 1	Note
0	-	-	LBT Function disable
1	000	VDD < 2.4V	Internal Input
1	001	VDD < 2.5V	Internal Input
1	010	VDD < 2.6V	Internal Input
1	011	VDD < 2.7V	Internal Input
1	1xx	P10 < 1.2V, LBTO=1	External Input

Bit5: **P11IO:** Port 1.1 Input/LBT function control bit.
0 = Set P11 as Input Port,
1 = Set P11 as LBT function, P11 connect to ground internally.

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

Low Battery Voltage	R1	R2	LBTO=1
2.3V	470kΩ	530kΩ	VDD<2.3V
2.8V	620kΩ	470kΩ	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: LBT external input P10 and P11IO function is not available in ICE emulation.***
- ***Note_3: IO input voltage must keep lower than VDD.***

11.8 Analog Setting and Application

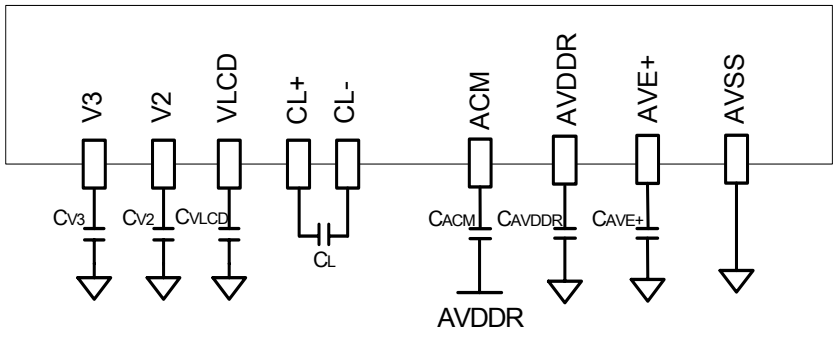
The most applications of SN8P2947 were for DC measurement ex. Weight scale, Pressure measure. Following table indicate different applications setting which MCU power source came from CR2032 battery, AA/AAA dry battery or external Regulator.

Capacitor Table:

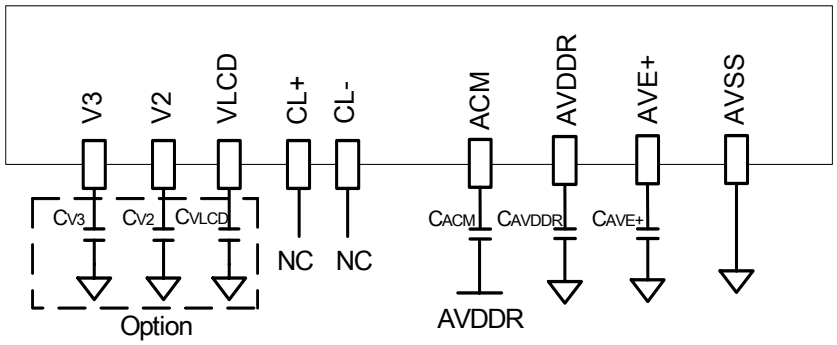
Power type	AI+	AI-	R+ / R-	ACM	AVDDR	AVE+	CL+/CL-	AVDD	DVDD
	C_{AI+}	C_{AI-}	C_R	C_{ACM}	C_{AVDDR}	C_{AVE+}	C_L	C_{AVDD}	C_{DVDD}
CR2032 (2.4~3V)	0.01uF	0.01uF	0.1uF	0.1uF	0.47uF	0.47uF	0.1uF	10uF, 0.1uF	10uF, 0.1uF
AA/AAA Bat.(2.4~3V)	0.01uF	0.01uF	0.1uF	0.1uF	1uF	1uF	0.1uF	1uF, 0.1uF	1uF, 0.1uF

- ***Note_1: In R-Type LCD Driver mode, C_L is not connected to MCU.***
- ***Note_2: In CR2032 battery application, 0.47uf capacitors are applied to AVE and AVDDR for lower VDD drop when regulator turn on at low battery situation.***
- ***Note_3: If AVE+ and AVDDR connect 0.47uf capacitors, the maximum output current of AVE+ will be limited 3mA maximally.***
- ***Note_4: 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.***

VDD=2.4V ~ 3.6V Analog Capacitor Connection (C-Type LCD Driver)

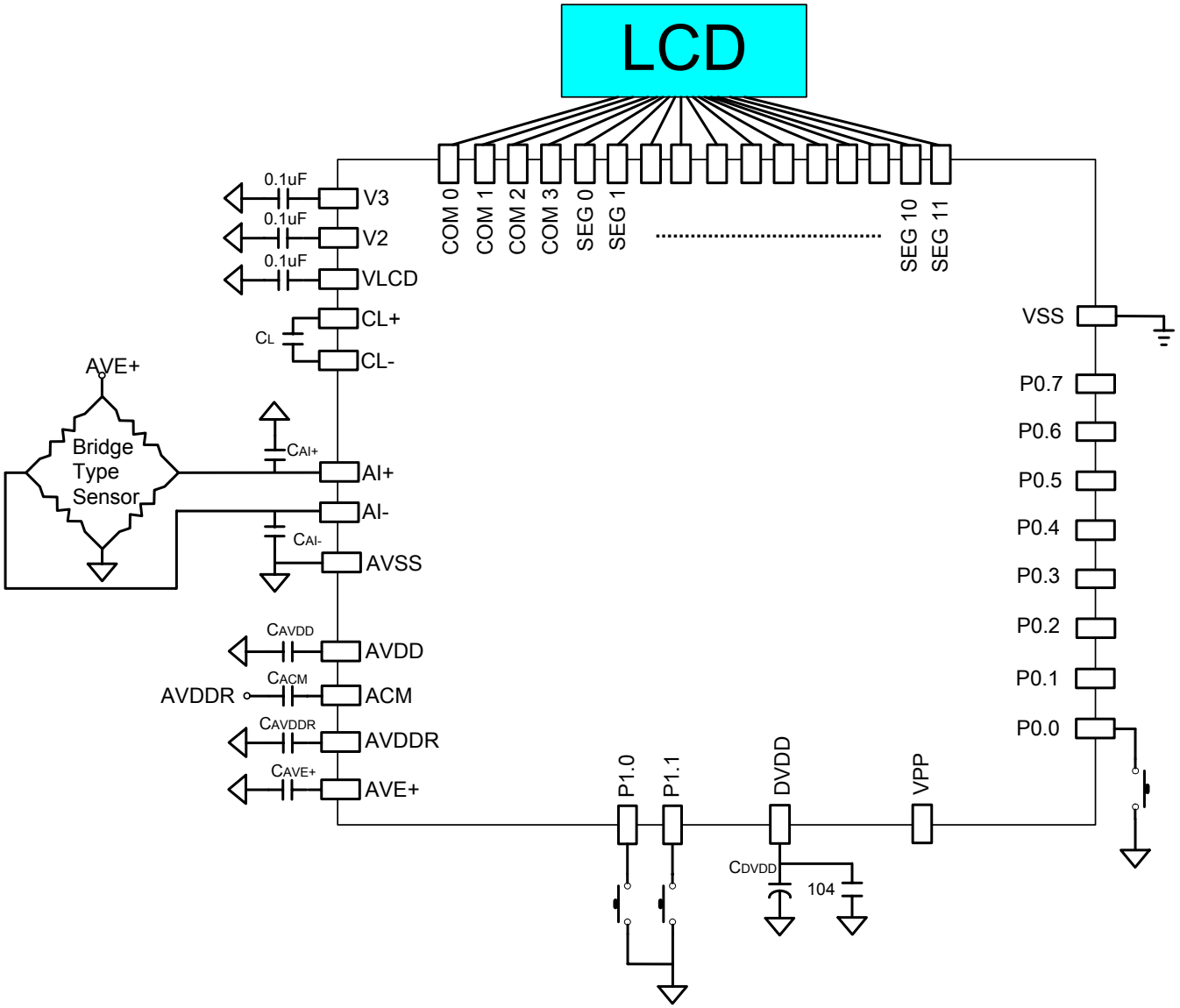


VDD=2.4V ~ 3.6V Analog Capacitor Connection (R-Type LCD Driver)

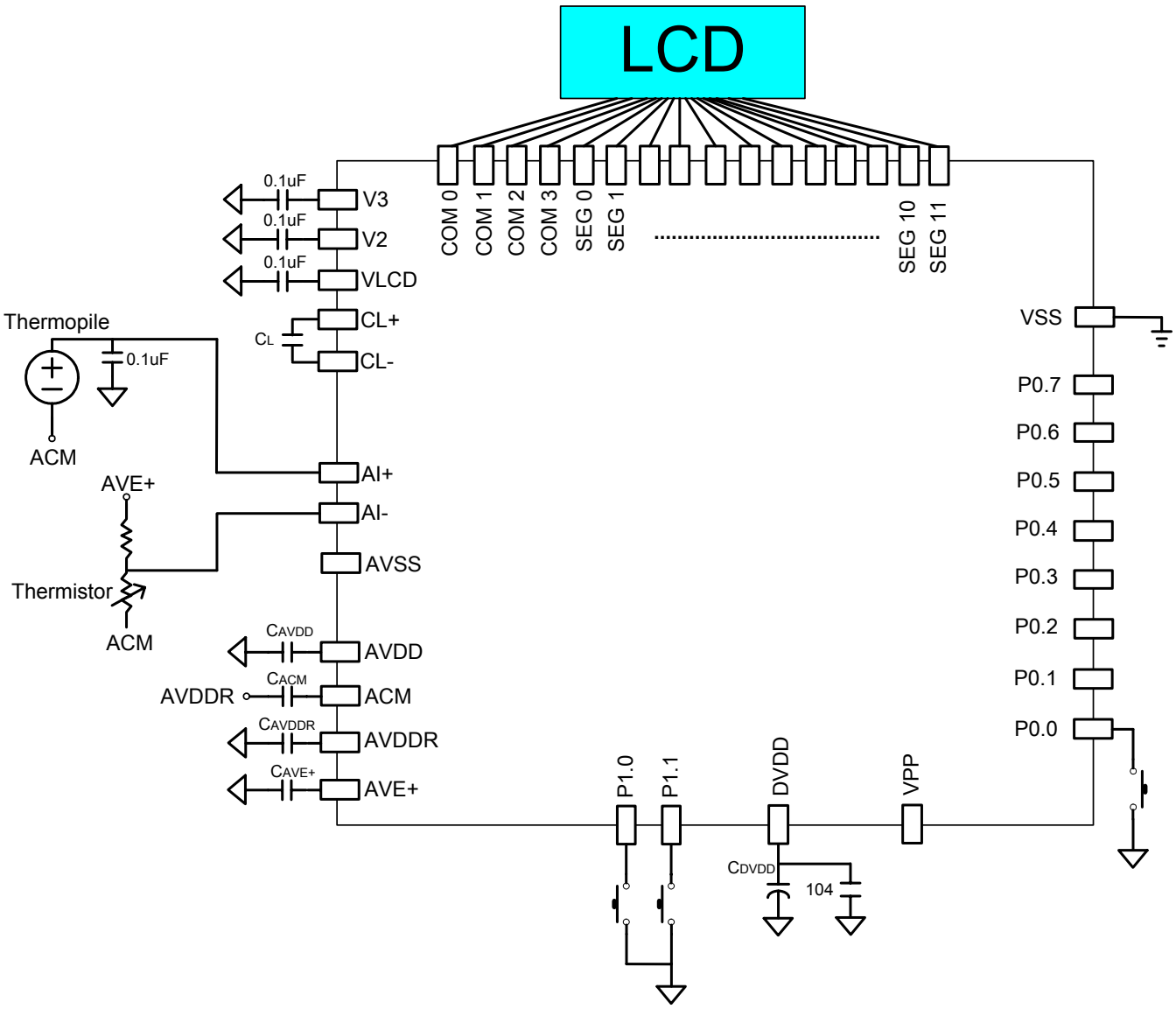


12 APPLICATION CIRCUIT

12.1 Scale (Load Cell) Application Circuit



12.2 Thermometer Application Circuit



13 INSTRUCTION SET TABLE

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

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

14 Development Tools

14.1 Development Tool Version

14.1.1 ICE (In circuit emulation)

- **SN8ICE2K Plus II:** Full function emulates SN8P2947 series

✷ **SN8ICE2K ICE emulation notice**
- **Operation voltage of ICE: 3.3V.**
- **Recommend maximum emulation speed at 3.3V: 1 MIPS (e.g. Fcpu = Fhosc/4).**
- **Use SN8P2947 EV-KIT to emulation Analog Function.**
- **Note: S8ICE1K doesn't support SN8P2947 serial emulation.**

14.1.2 OTP Writer

MP Pro Writer：ON/OFF line operation to support SN8P2947 mass production.

✷ **Note: MPIII Writer doesn't support SN8P2947 OTP programming.**

14.1.3 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 SN8P2947

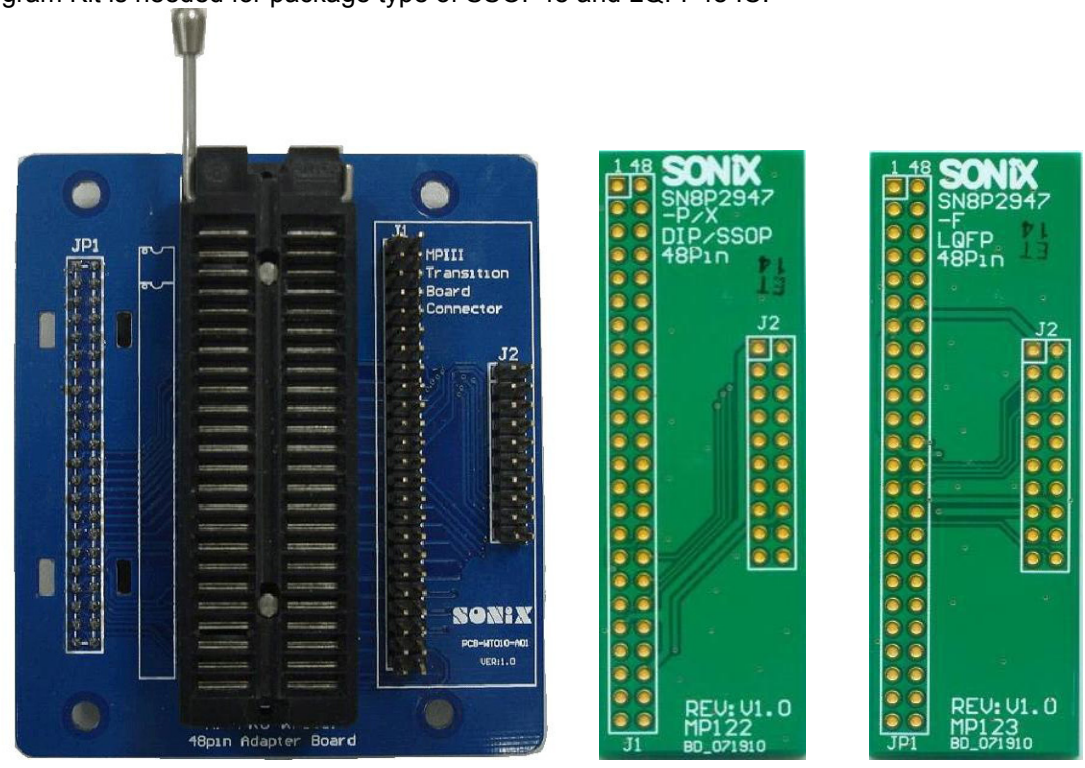
14.2 OTP Programming Pin to Transition Board Mapping

SN8P2947 COB Programming Pin Mapping:

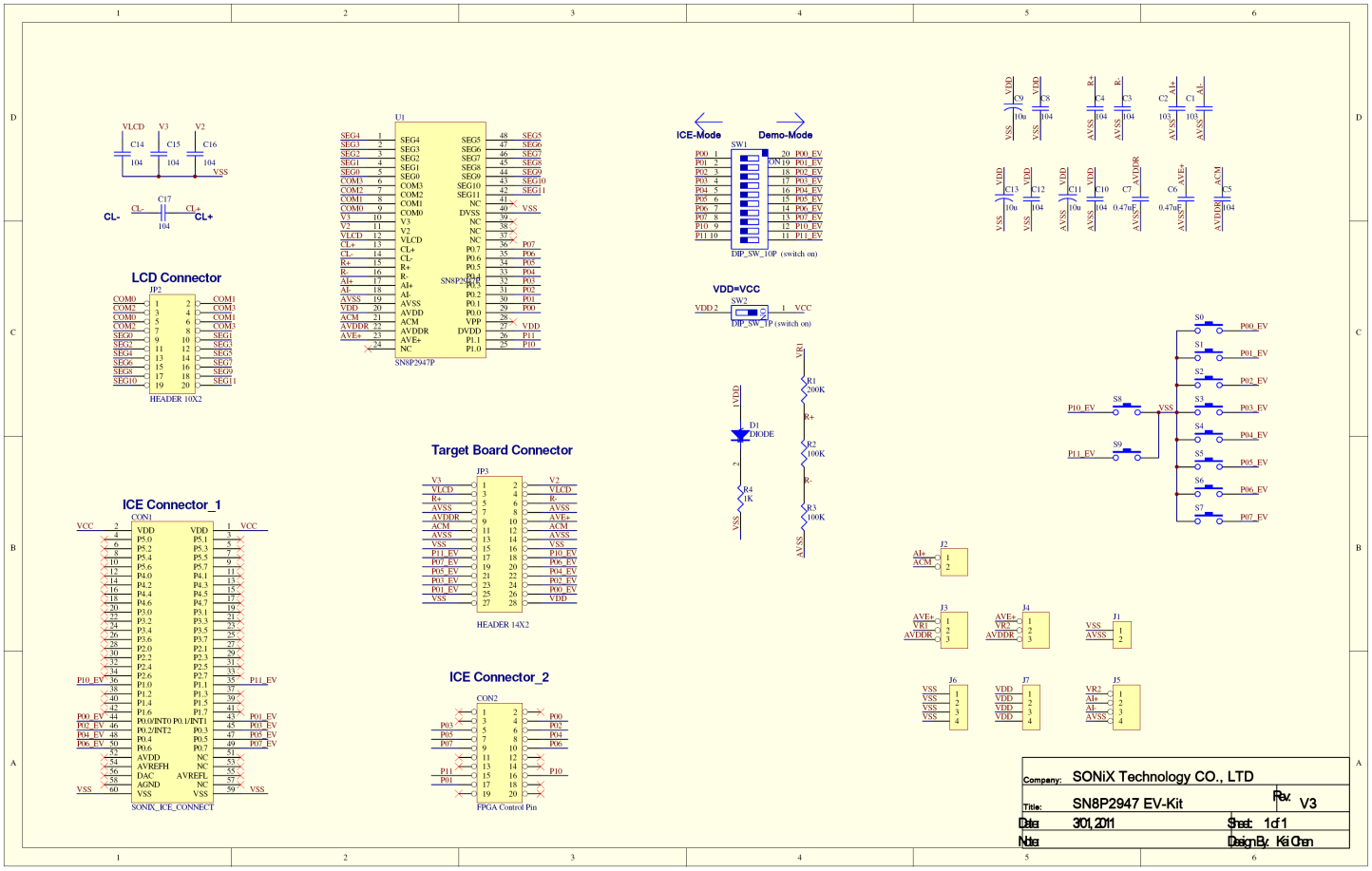
OTP Programming Pin of SN8P2947 Series				
MP PRO Writer		SN8P2947		
JP3 of 40 PIN Adapter Board		Pin Assignment		
Number	Pin Name	Pad Name	DIP48/SSOP48 PIN Number	LQFP48 PIN Number
1	VDD	DVDD/AVDD/ VLCD	27/20/12	23/13/5
2	GND	DVSS/AVSS	40/19	33/12
3	CLK / PGCLK	P0.1	30	26
4	CE	-	-	-
5	PGM / OTPCLK	P0.2	31	27
6	OE / ShiftData	P0.3	32	28
7	D1	-	-	-
8	D0	-	-	-
9	D3	-	-	-
10	D2	-	-	-
11	D5	-	-	-
12	D4	-	-	-
13	D7	-	-	-
14	D6	-	-	-
15	VDD	DVDD/AVDD / VLCD	27/20/12	23/13/5
16	VPP	VPP	28	24
17	HLS	-	-	-
18	RST	-	-	-
19	-	-	-	-
20	ALSB/PDB	P0.4	33	29

SN8P2947 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, MP122 or MP123, on the adapter board (J1/J2).
3. Program Kit is needed for package type of SSOP48 and LQFP48 IC.



14.3 APPENDIX A: EV-KIT BOARD CIRCUIT



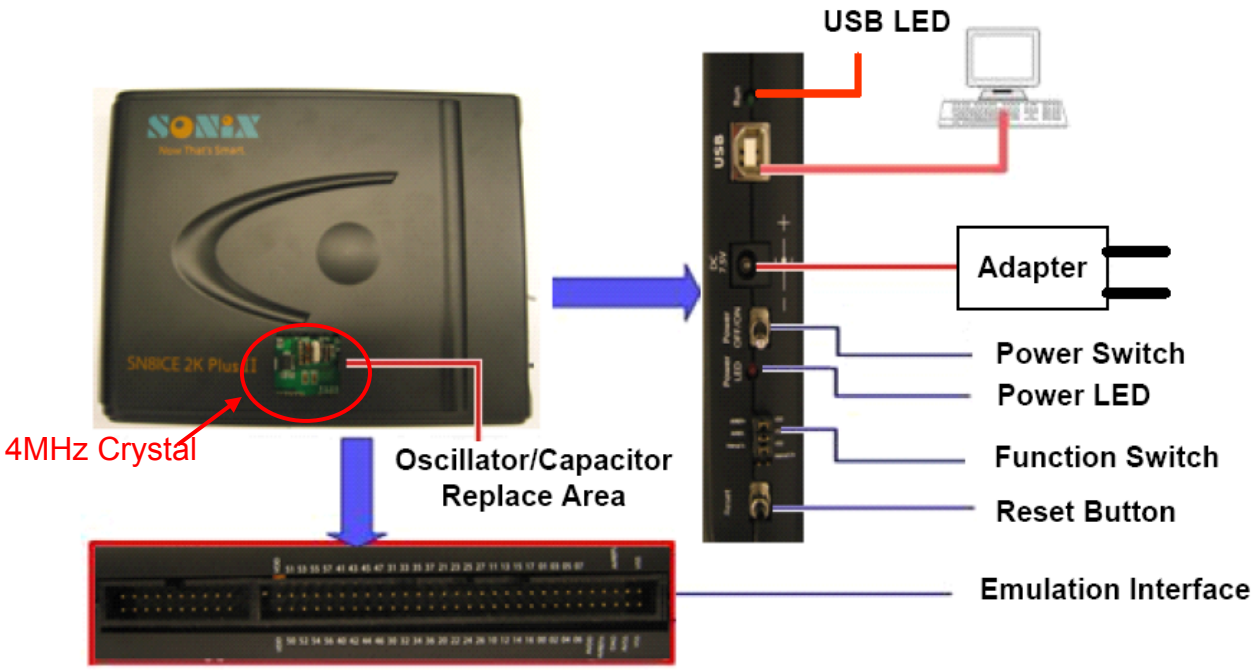
14.4 SN8P2947 Emulation

14.4.1 INTRODUCTION

Sonix provides a complete EV-KIT for SN8P2947 emulation, which includes an ICE SN8ICE2K_Plus_II, a SN8P2947 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.

14.4.2 SN8ICE2K_Plus_II Hardware Setting Notice for SN2947 EV-Kit

1. Chosen 4MHz 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 SN8P2947 emulation.
3. Detail setting reference SN8ICE2K Plus II User's Manual.



14.4.3 SN8P2947 EV Board DESCRIPTION

Sonix provides SN8P2947 EV board for all functions emulation shown in FIG.1

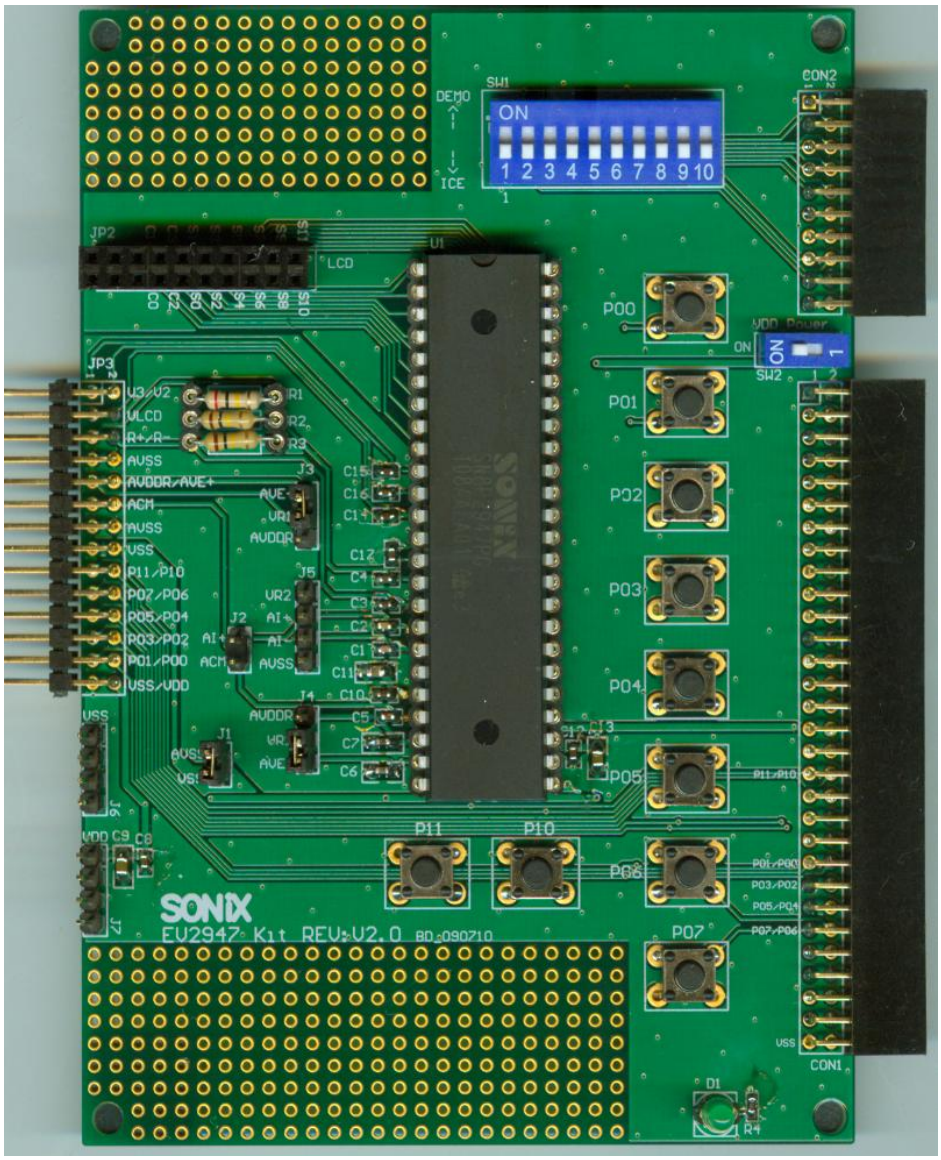


FIG.1 SN8P2947 EV board

14.4.4 EV BOARD SETTING

1. CON1/CON2 : connecting with the ICE PORT
2. SW2: 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 J6/J7 input.
3. D1 : Power indicator
4. JP2 : LCD COM/SEG connect Pin.
5. JP3: Target board connector.
6. SW1: Switch to “ICE” position when EV-Board connect to ICE. When separate ICE and Ev board, switch SW1 to “DEMO” position，or else the relative IO port won’t work.
7. J1: AVSS and VSS “SHORT” Pin.
8. J2: Analog Single-End AI-/ACM input Pin.

9. J3: Selection of external reference voltage V(R+, R-) power from AVDDR or AVE+.
10. J4: Selection of Load cell power from VDD or AVE+.
11. J5: Analog Differential (AI+, AI-) input Pin.
12. R5/R6/R7: Resistors for ADC external reference voltage.
13. C14/C15/C16/C17: C-Type LCD external capacitors.

14.4.5 Notice for EV Emulation

1. ICE VDD must switch to 3.3V. VDD 5V is not available for SN8P2947 EV-Kit
2. SW1 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...)

15 ELECTRICAL CHARACTERISTIC

15.1 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

15.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to V_{SS}, V_{DD} = 3.0V,F_{OSC} = IHRC(4MHz),Fcpu=1MHZ, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd		2.4	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	IoH	Vop = Vdd - 0.5V		4	8	-	mA
sink current	IoL	Vop = Vss + 0.5V		4	8	-	
INT0 trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
	Idd1	**Normal Mode**	Vdd=3V, Analog Parts OFF	-	0.5	1	mA
	Idd2		Vdd=3V, Analog Parts ON	-	1.2	2.4	mA
	Idd3	**Slow Mode**	Vdd=3V, High Clock On, Analog Parts On, C-Type LCD On	-	1	2	mA
	Idd4		Vdd=3V, High Clock On, Analog Parts OFF, C-Type LCD On	-	150	300	uA
	Idd5		Vdd=3V, High Clock OFF, Analog Parts OFF, R-LCD 300k On	-	15	30	uA
	Idd6		Vdd=3V, High Clock OFF, Analog Parts OFF, LCD OFF	-	4	8	uA
	Idd7	**Green Mode**	Vdd=3V, High Clock On, Analog Parts On, C-Type LCD On	-	1	2	mA
	Idd8		Vdd=3V, High Clock On, Analog Parts OFF, C-Type LCD On	-	150	300	uA
	Idd9		Vdd=3V, High Clock OFF, Analog Parts OFF, R-LCD 300k On	-	14	28	uA
	Idd10		Vdd=3V, High Clock OFF, Analog Parts OFF, LCD OFF	-	3	6	uA
	Idd11	**Sleep Mode**	Vdd= 3V	-	1	2	uA
LVD detect level	V_{LVD}	Internal POR detect level		1.7	1.9	2.2	uA
Internal High Clock Freq.	F_{IHRC}	Internal High RC Oscillator Frequency (Vdd = 2.4V ~ 3.6V, Temperature: 0℃~ 70℃)		3.6	4	4.4	MHz

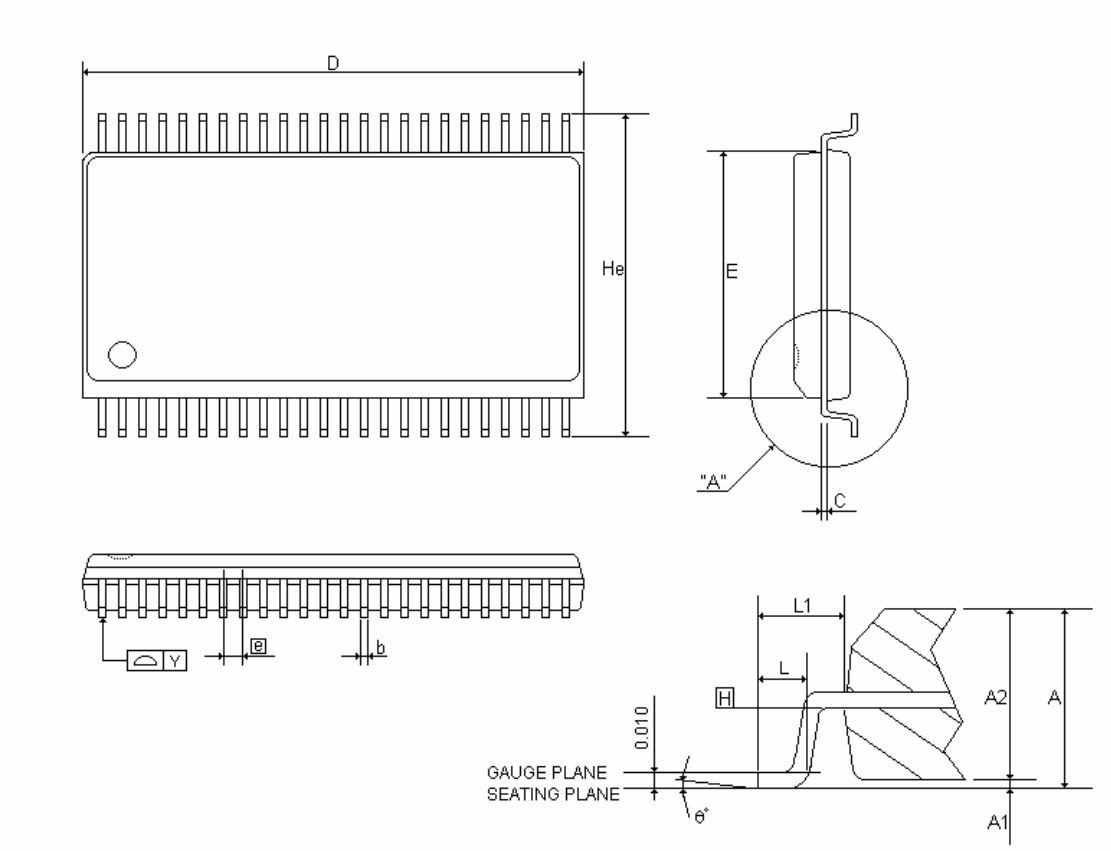
➢ **Note: Analog Parts including Regulator, PGIA and ADC.**

(All of voltages refer to Vdd=3V F_{OSC} = IHRC (4MHz), 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	-	200	300	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=333KHz, OSR=64	-	5.2	-	kSps
Reference Voltage Input absolutely Voltage	V_{AIN}	R+, R- Input Range (ADC External Vref.)	0.4	-	1.4	V
		ADC Internal Vref	0.4	-	1.4	V
Reference Voltage Range	Vref	ADC Reference voltage range	0.3	-	0.8	V
Integral non-linearity	INL	PGIAx128, ADC Input Range ±0.9xVref	-	-	0.01	%FSR
No missing code	NMC	ADC range ±29491. (0.9 x Vref)	20	-	-	bit
ADC Noise free bits	NFB	Gain:1, Vref:0.8V, OSR:32768, Input-short	-	16.4	-	bit
		Gain:128, Vref:0.8V, OSR:32768, Input-short	-	15	-	bit
Effective number of bits	ENOB	Gain=128, Vref=0.8V, OSR=32768, Input-short	-	17.5	-	bit
		Gain=1, Vref=0.8V, OSR:32768, Input-short	-	19	-	bit
ADC Input range	V_{AIN}	ADC input signal, signal after PGIA application	0.4	-	1.4	V
Unit Gain Buffer	GX	ADC signal input buffer	-	50	75	uA
	GR	ADC reference input buffer	-	50	75	uA
Temperature sensor inaccuracy	E_{TS}	Inaccuracy range vs. real Temp.	-	±8	-	℃
PGIA						
PGIA Current consumption	I_{DD_PGIA}	Run mode @ 2.4V	-	120	200	uA
Power down current	I_{PDN}	Stop mode @ 2.4V	-	-	0.1	uA
Input offset voltage	Vos		-100	-	25	uV
Bandwidth	BW		-	-	2	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 (Refer to ACM)						
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	-	160	200	uA
Regulator						
Regulator output voltage AVDDR	V_{AVDDR}		2.25	2.4	2.55	V
Regulator output voltage AVE+	V_{AVE+}	AVE+ set as 2.0V	1.85	2.0	2.15	V
Analog common voltage	V_{ACM}	V_{ACM} = 1V	0.9	1	1.1	V
		V_{ACM} = 0.75V	0.65	0.75	0.85	V
Regulator output current capacity	I_{VA+}	AVDDR, AVE output current ability	-	-	5	mA
Quiescent current	I_{QI}	ACM + AVDDR + AVE	-	80	100	uA
V_{ACM} driving capacity	I_{SRC}		-	-	10	μA
V_{ACM} sinking capacity	I_{SNK}		-	-	1	mA
LCD Driver						
R-Type LCD Operation Current	I_{RLCD}	VDD:3V, 1/3 bias, 100k bias resistor, No panel		10	15	uA
		VDD:3V, 1/3 bias, 33k bias resistor, No panel		30	45	uA
C-Type LCD Operation Current	I_{CLCD}	1/3 bias, LCD Charge pump + Bandgap current		160	200	uA
C-Type VLCD output Voltage	V_{LCD}	VLCD set 3V,	2.85	3.05	3.25	V
VLCD Variation vs. VDD and Temp		VDD: 2.4~3.6V. Temp.: -10 ~ 50℃	-50	-	50	mV

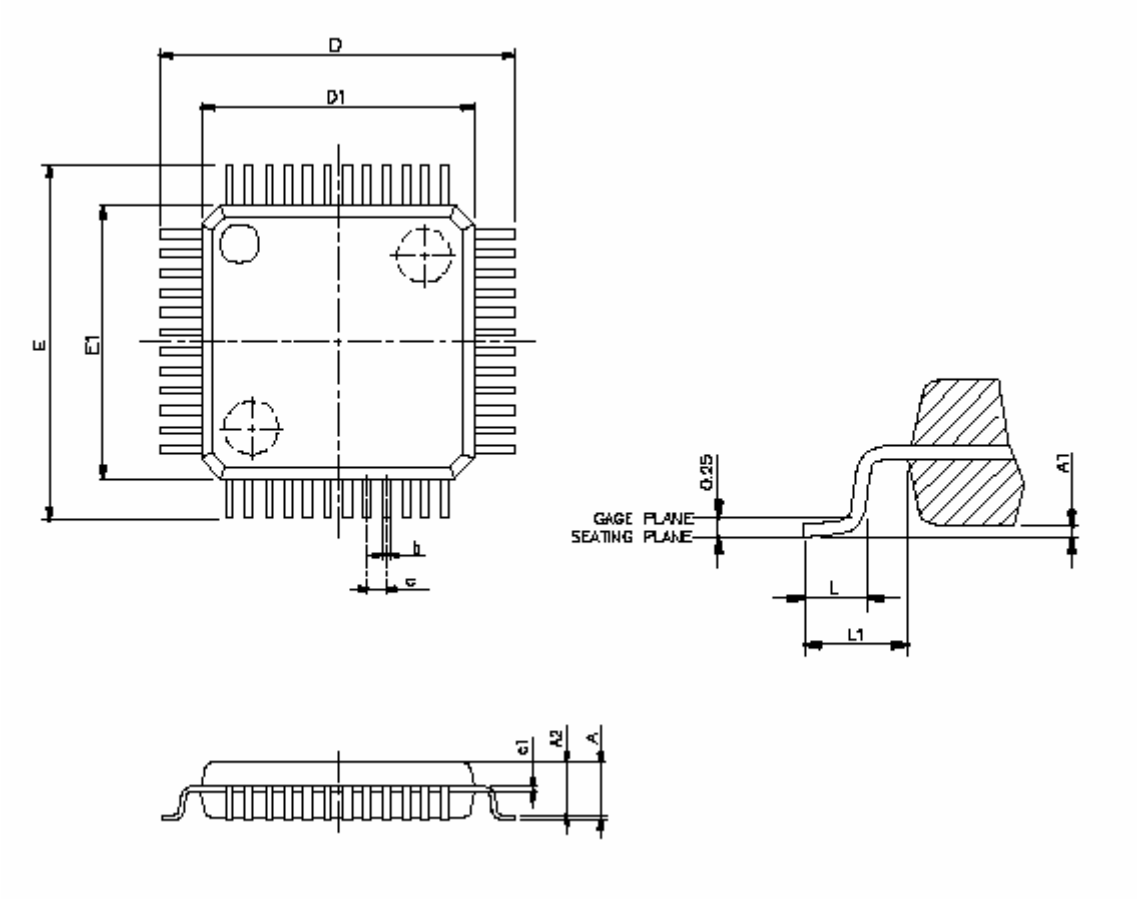
16 PACKAGE INFORMATION

16.1 SSOP 48 PIN



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

16.2 LQFP 48 PIN



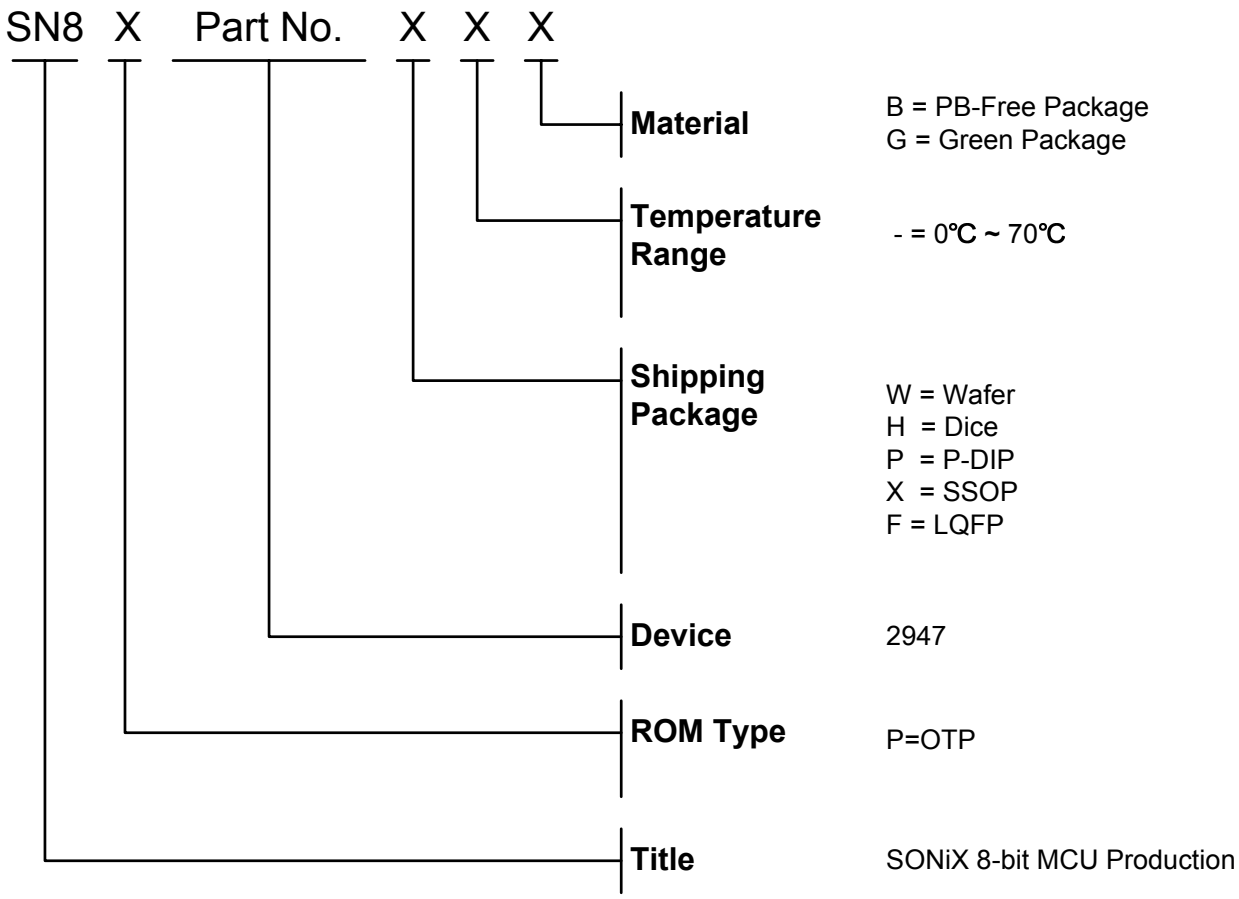
SYMBOLS	MIN	NOR	MAX
	(mm)		
A	-	-	1.6
A1	0.05	-	0.15
A2	1.35	-	1.45
c1	0.09	-	0.16
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
e	0.5 BSC		
B	0.17	-	0.27
L	0.45	-	0.75
L1	1 REF		

17 Marking Definition

17.1 INTRODUCTION

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

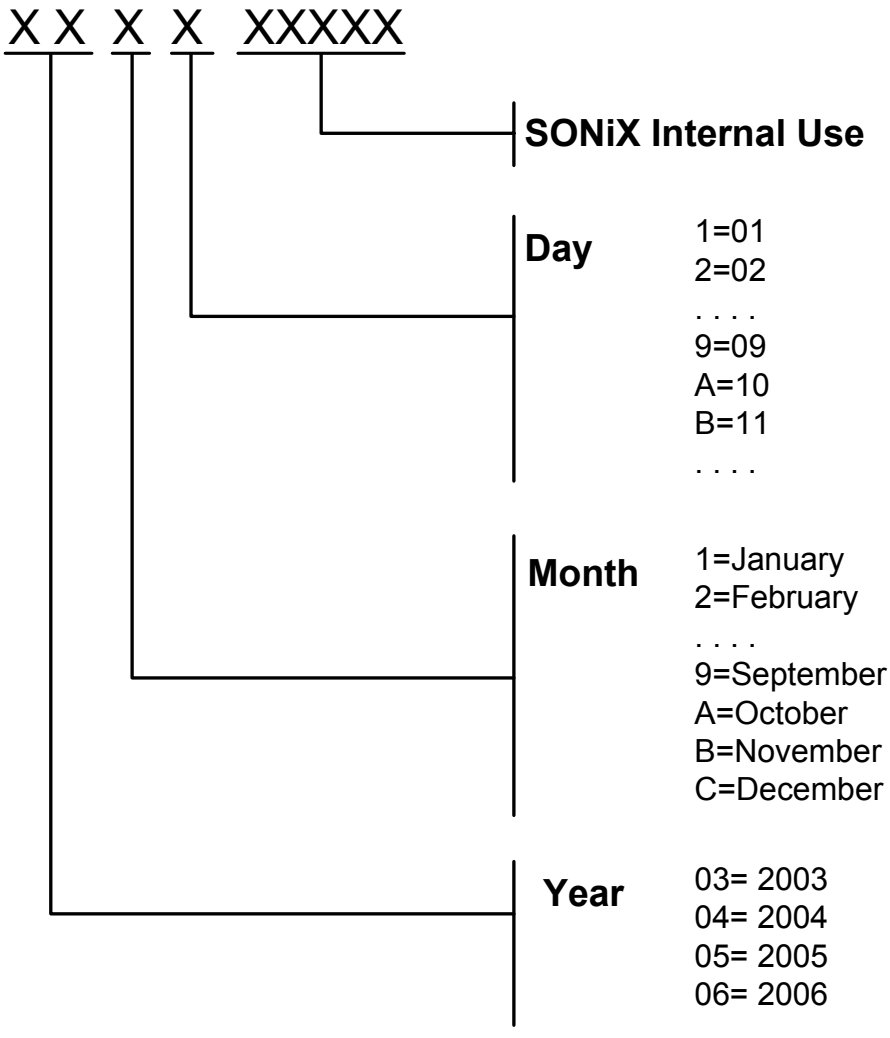
17.2 MARKING INDETIFICATION SYSTEM



17.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P2947PG	OTP	2947	DIP48	0℃~70℃	Green Package
SN8P2947XG	OTP	2947	SSOP48	0℃~70℃	Green Package
SN8P2947FG	OTP	2947	LQFP48	0℃~70℃	Green Package

17.4 DATECODE SYSTEM



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

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

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

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