

SN8P2949

USER'S MANUAL

Specification V1.4

SONiX 8-Bit Micro-Controller

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

AMENDENT HISTORY

Version	Date	Description
V1.0	2012.02	First issue
V1.1	2012.02	1. Initial value BGM = 1; INRENB = 1 2. Add VLED start recommendation 3. Table of electrical characteristic modification 4. Add note of current consumption from VLED
V1.2	2012.03	1. ISP Voltage is 6.5V 2. Update migration table, pin description table. 3. BGM always set "1". 4. C-Type LCD Pump start procedure must Call Macro "LcdPumpStart". Please reference demo code.
V1.3	2012.04	1. Add 1929 information in the migration table. 2. Modify ADC offset option:(0,-1/4,-1/2,-3/4)*Vref 3. Modify ADC gain option:1x, 2x and 4x 4. Modify cycle description in the Instruction Table
V1.4	2012.11	1. Update ADC ENOB/Noise Table(P.118):Delete ADC x4/x8 Data 2. AVDD, DVDD, AVSS, VSS modified 3. Add INT1/P01 description 4. T0 Note Modified by referring to 1929 5. STKP : A,#00000111b in p.36 6. Modify STKnH: 3bits → 5bits 7. Add: 4, 8, 16, or 32 in p.44 8. Modify application CKT(12.1):VDD and AVDD have individual Capacitors. 9. Add Note: Capacitors connected to power pin as close as possible 10.Add Note: ENOB drop when selecting ADC Offset Function 11. Modify the suggestion programming time to 30us in ISP, and setting in demo code

Table of Content

AMENDENT HISTORY	2
1 PRODUCT OVERVIEW.....	7
1.1 SELECTION TABLE	7
1.2 MIGRATION TABLE	7
1.1 FEATURES	8
1.2 SYSTEM BLOCK DIAGRAM.....	9
1.3 PIN ASSIGNMENT.....	10
1.4 PIN DESCRIPTIONS	11
1.5 PIN CIRCUIT DIAGRAMS.....	12
2 CENTRAL PROCESSOR UNIT (CPU)	13
2.1 MEMORY MAP.....	13
2.1.1 PROGRAM MEMORY (ROM).....	13
2.1.2 RESET VECTOR (0000H).....	14
2.1.3 CODE OPTION TABLE.....	22
2.1.4 DATA MEMORY (RAM)	23
2.1.5 SYSTEM REGISTER.....	24
2.1.6 ACCUMULATOR.....	27
2.1.7 PROGRAM FLAG.....	28
2.1.8 PROGRAM COUNTER.....	29
2.1.9 R REGISTERS.....	33
2.2 ADDRESSING MODE.....	34
2.2.1 IMMEDIATE ADDRESSING MODE.....	34
2.2.2 DIRECTLY ADDRESSING MODE.....	34
2.2.3 INDIRECTLY ADDRESSING MODE.....	34
2.3 STACK OPERATION	35
2.3.1 OVERVIEW.....	35
2.3.2 STACK REGISTERS.....	36
2.3.3 STACK OPERATION EXAMPLE.....	37
3 RESET.....	38
3.1 OVERVIEW	38
3.2 POWER ON RESET	38
3.3 WATCHDOG RESET	39
3.4 BROWN OUT RESET.....	39
3.4.1 BROWN OUT DESCRIPTION.....	39
3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION	40
3.4.3 BROWN OUT RESET IMPROVEMENT	40
4 SYSTEM CLOCK.....	42
4.1 OVERVIEW	42
4.2 CLOCK BLOCK DIAGRAM	42
4.3 OSCM REGISTER	43
4.4 SYSTEM HIGH CLOCK.....	44

4.5	SYSTEM LOW CLOCK.....	44
4.5.1	SYSTEM CLOCK MEASUREMENT.....	45
5	SYSTEM OPERATION MODE.....	46
5.1	OVERVIEW	46
5.2	SYSTEM MODE SWITCHING	47
5.3	WAKEUP	49
5.3.1	OVERVIEW.....	49
5.3.2	WAKEUP TIME.....	49
5.3.3	PIW WAKEUP CONTROL REGISTER.....	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	54
6.5	MULTI-INTERRUPT OPERATION.....	56
7	I/O PORT.....	58
7.1	I/O PORT MODE.....	58
7.2	I/O PULL UP REGISTER.....	59
7.3	I/O PORT DATA REGISTER.....	60
8	TIMERS	61
8.1	WATCHDOG TIMER	61
8.2	TIMER 0 (T0)	63
8.2.1	OVERVIEW.....	63
8.2.2	T0M MODE REGISTER.....	64
8.2.3	T0C COUNTING REGISTER.....	65
8.2.4	T0 TIMER OPERATION SEQUENCE (High_Clk = IHRC).....	66
8.2.5	RTC OPERATION SEQUENCE (High_Clk = "IHRC_RTC" and "TOTB = 1").....	67
8.3	TIMER/COUNTER 0 (TC0)	69
8.3.1	OVERVIEW.....	69
8.3.2	TC0M MODE REGISTER.....	70
8.3.3	TC1X8, TC0X8, TC0GN FLAGS.....	71
8.3.4	TC0C COUNTING REGISTER.....	72
8.3.5	TC0R AUTO-LOAD REGISTER	74
8.3.6	TC0 CLOCK FREQUENCY OUTPUT (BUZZER).....	75
8.3.7	TC0 TIMER OPERATION SEQUENCE.....	76
8.4	TIMER/COUNTER 1 (TC1)	78
8.4.1	OVERVIEW.....	78
8.4.2	TC1M MODE REGISTER.....	79
8.4.3	TC1X8, TC0X8, TC0GN FLAGS.....	80
8.4.4	TC1C COUNTING REGISTER.....	81
8.4.5	TC1R AUTO-LOAD REGISTER	82
8.4.6	TC1 CLOCK FREQUENCY OUTPUT (BUZZER).....	83

8.4.7	TC1 TIMER OPERATION SEQUENCE.....	84
8.5	PWM0 MODE	86
8.5.1	OVERVIEW.....	86
8.5.2	TC0IRQ AND PWM DUTY	87
8.5.3	PWM PROGRAM EXAMPLE.....	87
8.5.4	PWM0 DUTY CHANGING NOTICE	88
8.6	PWM1 MODE	89
8.6.1	OVERVIEW.....	89
8.6.2	TC1IRQ AND PWM DUTY	90
8.6.3	PWM PROGRAM EXAMPLE.....	90
8.6.4	PWM1 DUTY CHANGING NOTICE	91
9	LCD DRIVER.....	92
9.1	LCD TIMING	92
9.2	LCDM1 REGISTER	94
9.3	LCDM2 REGISTER	95
9.4	C-TYPE LCD DRIVER MODE.....	96
9.5	R-TYPE LCD DRIVER MODE.....	98
9.6	LCD RAM LOCATION.....	99
10	IN SYSTEM PROGRAM ROM.....	100
10.1	OVERVIEW	100
10.2	ROMADRH/ROMADRL REGISTER.....	100
10.3	ROMDAH/ROMDAL REGISTERS.....	100
10.4	ROMCNT REGISTERS AND ROMWRT INSTRUCTION.....	101
10.5	ISP ROM ROUTINE EXAMPLE	102
11	REGULATOR, PGIA AND ADC	104
11.1	OVERVIEW	104
11.2	ANALOG INPUT	104
11.3	VOLTAGE REGULATOR.....	105
11.3.1	Voltage Regulator Control Register.....	105
11.4	PGIA -PROGRAMMABLE GAIN INSTRUMENTATION AMPLIFIER	106
11.4.1	AMPM1- Amplifier Mode1 Control Register.....	106
11.4.2	AMPM2- Amplifier Mode2 Control Register.....	109
11.5	TEMPERATURE SENSOR (TS).....	110
11.6	20-BIT ANALOG TO DIGITAL CONVERTER (ADC).....	111
11.6.1	Analog Inputs and Voltage Operation Range	112
11.6.2	Reference Voltage	112
11.6.3	Input Buffer	112
11.6.4	ADC Gain and Offset	112
11.6.5	ADC Output Word Rate	113
11.6.6	ADCM1- ADC Mode1 Register.....	113
11.6.7	ADCM2- ADC Mode2 Register.....	114
11.6.8	ADCM3- ADC Mode3 Register.....	114
11.6.9	ADC Data Register.....	116

11.7	LBTM: LOW BATTERY DETECT.....	123
11.7.1	<i>LBTM: Low Battery Detect Register</i>	<i>123</i>
11.8	CHARGE PUMP AND LED REGULATOR.....	124
11.9	ANALOG SETTING AND APPLICATION.....	126
12	APPLICATION CIRCUIT	127
12.1	SCALE (LOAD CELL) APPLICATION CIRCUIT.....	127
12.2	THERMOMETER APPLICATION CIRCUIT	128
13	INSTRUCTION SET TABLE	129
14	DEVELOPMENT TOOLS	130
14.1	DEVELOPMENT TOOL VERSION	130
14.1.1	ICE (IN CIRCUIT EMULATION)	130
14.1.2	OTP WRITER.....	130
14.1.3	IDE (INTEGRATED DEVELOPMENT ENVIRONMENT).....	130
14.2	OTP PROGRAMMING PIN TO TRANSITION BOARD MAPPING.....	131
14.3	APPENDIX A: EV-KIT BOARD CIRCUIT	132
14.4	SN8P2949 EMULATION	133
14.4.1	<i>INTRODUCTION.....</i>	<i>133</i>
14.4.2	<i>SN8ICE2K_Plus_II Hardware Setting Notice for SN2949 EV-Kit.....</i>	<i>133</i>
14.4.3	<i>SN8P2949 EV-kit Board DESCRIPTION.....</i>	<i>134</i>
14.4.4	<i>EV-kit BOARD SETTING.....</i>	<i>134</i>
14.4.5	<i>Notice for EV-kit Emulation.....</i>	<i>135</i>
15	ELECTRICAL CHARACTERISTIC.....	136
15.1	ABSOLUTE MAXIMUM RATING.....	136
15.2	ELECTRICAL CHARACTERISTIC.....	136
16	PACKAGE INFORMATION	138
16.1	LQFP 80 PIN.....	138
17	MARKING DEFINITION.....	139
17.1	INTRODUCTION	139
17.2	MARKING INDETIFICATION SYSTEM.....	139
17.3	MARKING EXAMPLE.....	140
17.4	DATECODE SYSTEM.....	140

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						
SN8P1919	6K*16	256*8	8	4*32	V	V	V	22	16-bit	V	V	7	LQFP80
SN8P1929	4K*16	256*8	8	4*24	V	V	V	16	16-bit	V	V	6	LQFP80
SN8P2949	8K*16	256*8	8	4*32	V	V	V	20	20-bit	V	V	13	LQFP80

Table 1-1 Selection table of SN8Px9x9

1.2 MIGRATION TABLE

ITEM	SN8P1919	SN8P1929	SN8P2949
Power Supply (VDD)	2.4V ~ 5.5V	2.4 ~ 5.5V	2.4V ~ 3.6V
Operating Current Consumption	More	More	Less
PGIA Gain setting	1x, 12.5x, 50x, 100x, 200x	1x, 12.5x, 50x, 100x, 200x	1x, 12.5x, 50x, 100x, 200x
AVE+ Voltage	3.0V, 2.4V or 1.5V	3.0V, 2.4V or 1.5V	2.0V or 1.5V
AVDDR Voltage	3.8V	3.8V	2.4V
ACM Voltage (Sink Current Only)	1.2V	1.2V	1.0V
AVE / AVDDR Double Current	Yes	Yes	No
Charge pump for Analog Voltage (AVDDCP)	Yes	Yes	No
Internal ADC Reference Voltage V(R+,R-)	0.8V, 0.64V or 0.4V	0.8V, 0.64V or 0.4V	0.3V ~ 0.8V
AO± and X± Pads	Yes	X± Only	No
ADC Gain	No	No	x1, x2 or x4
ADC Offset Function	No	No	Yes (0,-1/4,-1/2,-3/4)*Vref
ADC Resolution	16-Bit	16-Bit	20-Bit
ADC Stable Time	Slow (0.5s)	Slow (0.5s)	Fast (3 x ADC WR)
ADC WR (Output Rate)	1Hz ~ 50Hz	1Hz ~ 50Hz	1Hz ~ 3.9kHz
ADC Fast Measurement	No	No	Yes
ADC Work in Slow/Green Mode with Wake up function	No	No	Yes
Battery Detect Method	By ADC or Comparator	By ADC or Comparator	By ADC or Comparator (LBT:2.2~2.9V)
High Clock	IHRC or 4MHz Crystal	IHRC or 4MHz Crystal	IHRC
LCD Type	R-Type only	R-Type only	R-Type or C-Type
VLCD and VLCD1 Voltage	Can be Different	VLCD Pin Only	VLCD Pin Only
Capacitors Requirement	More	More	Less (Chip Cap.)
LED Backlight Power	AVDDR or AVE+	AVDDR or AVE+	VLED (3.5V)
ISP Function for sensor calibration	No	Yes	Yes
ISP Vpp source and voltage	No	External 12.5V	External 6.5V or Internal generation

Table 1-2 SN8Px9x9 Migration Table

1.1 FEATURES

◆ **Memory configuration**

OTP ROM size: 8K * 16 bits
 RAM size: 256 * 8 bits (bank 0, bank 1)
 8-levels stack buffer
 LCD RAM size: 4*32bits

◆ **I/O pin configuration**

Bi-directional: P0, P1, P2, P5
 Wakeup: P0, P1
 Pull-up resistors: P0, P1, P2, P5
 External interrupt: P00, P01

◆ **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)

◆ **Three 8-Bit Timer**

T0: Basic Timer. Build in 0.5 sec RTC mode
 TC0: Auto-reload Timer/Counter/PWM0/Buzzer
 TC1: Auto-reload Timer/Counter/PWM1/Buzzer

◆ **Programmable Gain Instrumentation Amplifier**

◆ **PGIA Gain Option: 1x/12.5x/50x/100x/200x**

◆ **20-bit Delta-Sigma ADC with 18-bit Noise Free**

ADC Gain selection: 1x, 2x, 4x
 ADC Offset selection: (-1/4, -1/2, -3/4 or 0) x Vref
 ADC Interrupt with Green Mode wakeup function
 Three ADC channel configuration:
 Two Fully Differential Channels
 One Differential and Two Single-ended Channels
 Four Single-ended Channels

◆ **Six Interrupt Sources**

Four Internal Interrupts: T0, TC0, TC1, ADC
 Two External Interrupts: INT0, INT1

◆ **Single Power Supply: 2.4V ~ 3.6V (Analog Part) 2.2V ~ 3.6V (Digital Part)**

◆ **On-chip Watchdog Timer**

◆ **On-chip Regulator (AVDDR) with 2.4V Voltage Output and 5mA Driven Current**

◆ **On-chip Charge Pump Regulator (VLED) with 3.5V Voltage Output and 8mA Driven Current for LED Drive.**

◆ **On Chip Regulator with 2.0V / 1.5V Output Voltage AVE+ Loading Current Consumption will Not Double**

◆ **On-chip 1.2V Band Gap Reference for Battery Monitor**

◆ **Internal LBT 2.2V~2.9V; or External P52 Input LBT**

◆ **Build in ADC reference voltage: internal 0.3V~ 0.8V External reference voltage input V(R+,R-)**

◆ **In-system Programmer ROM: Internal 6.5V Generation for VPP**

◆ **LCD driver:**

1/3 or 1/2 bias voltage.
 4 common * 32 segment
 Both R type and C type LCD
 Multiple C-type LCD Voltage: 2.7 ~3.4V
 R-type optional Bias Resistor: 400k, 200k, 100k, 33k
 C-Type LCD Duty Control Function

◆ **Dual clock system offers four operating modes**

High clock : IHRC 4 MHz only
 Low clock : ILRC 32kHz or 32768 Crystal @IHRC_RTC
 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
 Wakeup by P0/T0/TC0/ADC
 Sleep Mode: Both High and Low Clock Stop

◆ **Package**

Dice /LQFP80

1.2 SYSTEM BLOCK DIAGRAM

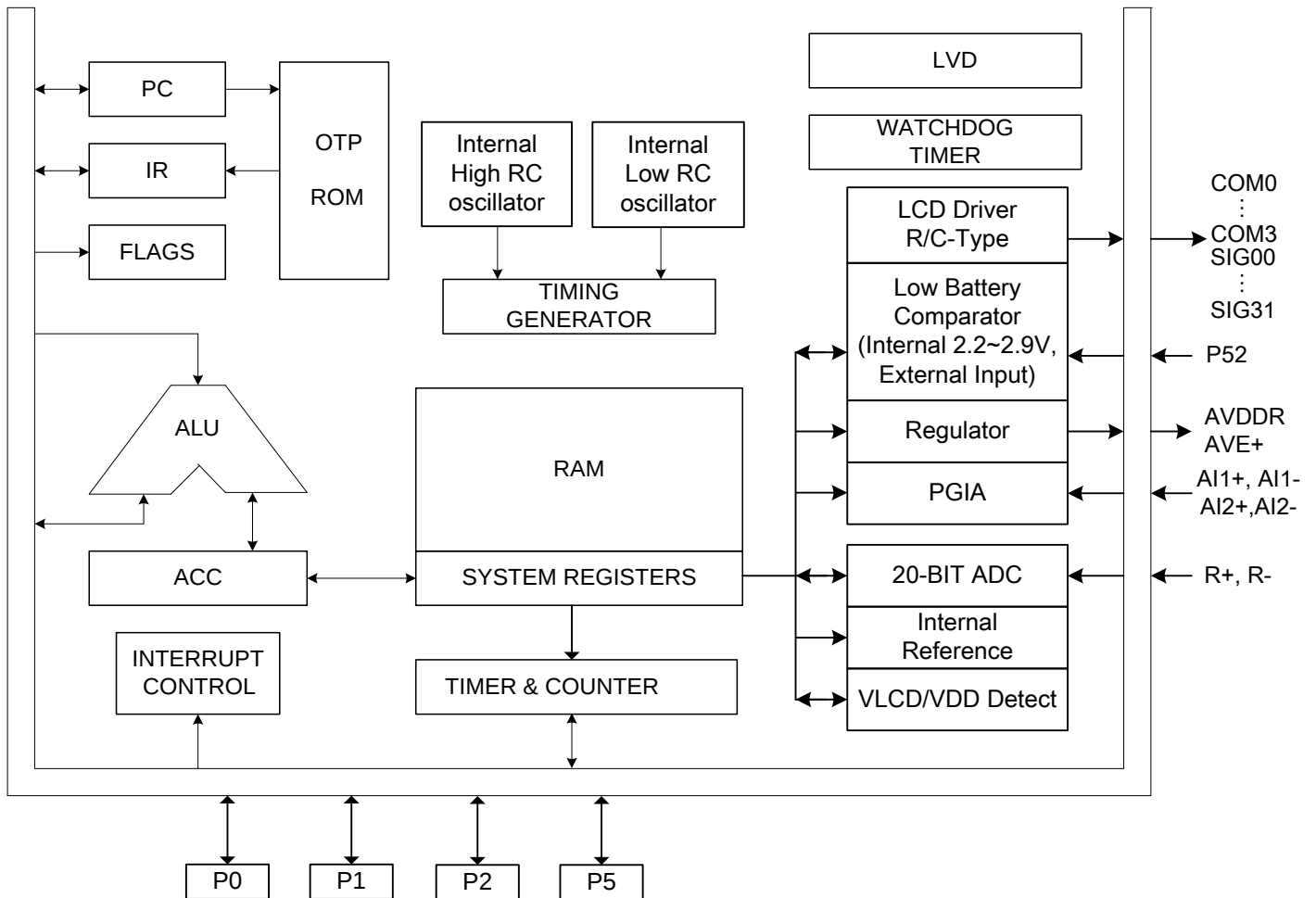
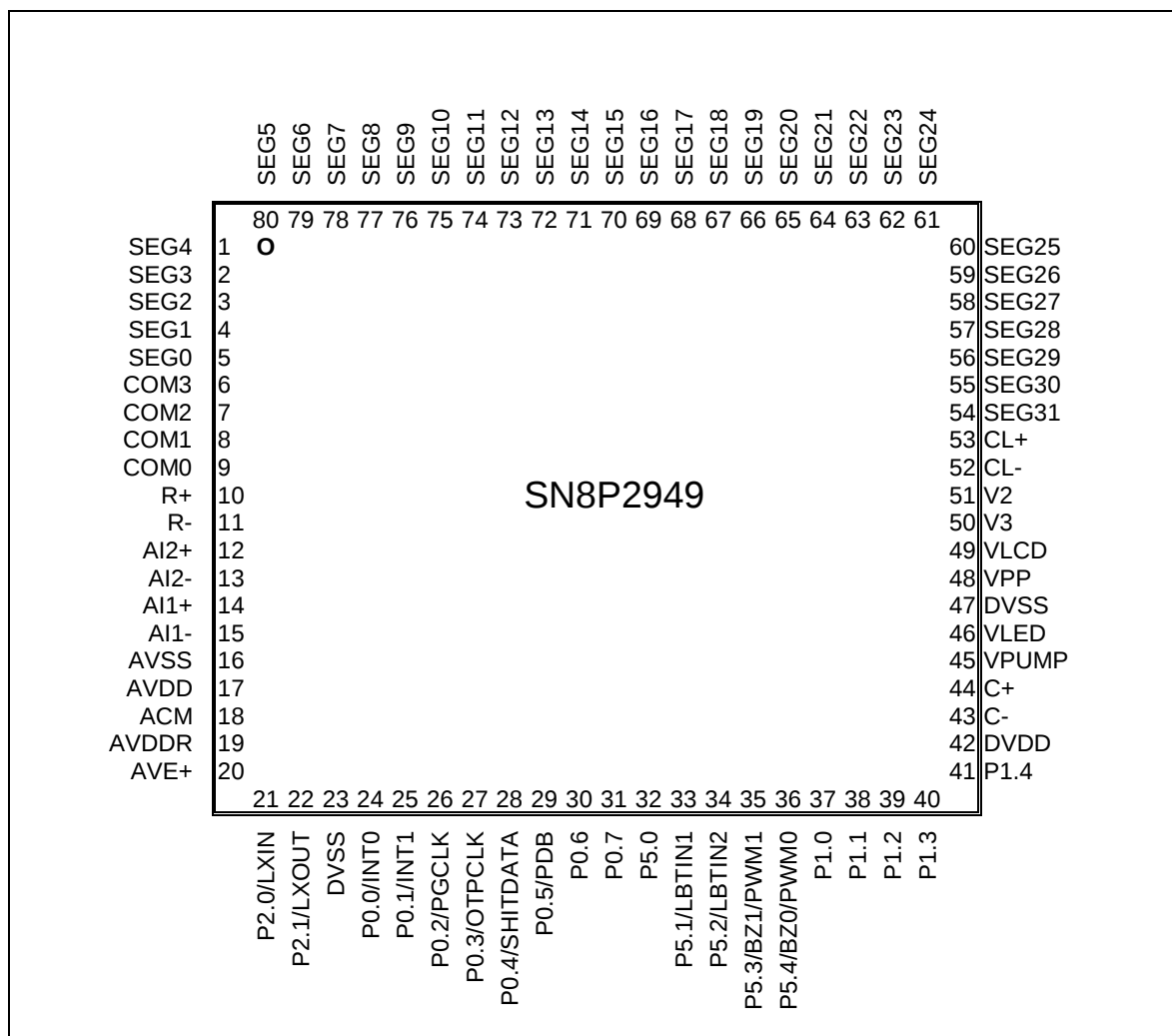


Figure 1-1 Simplified System Block Diagram

1.3 PIN ASSIGNMENT

SN8P2949 LQFP80

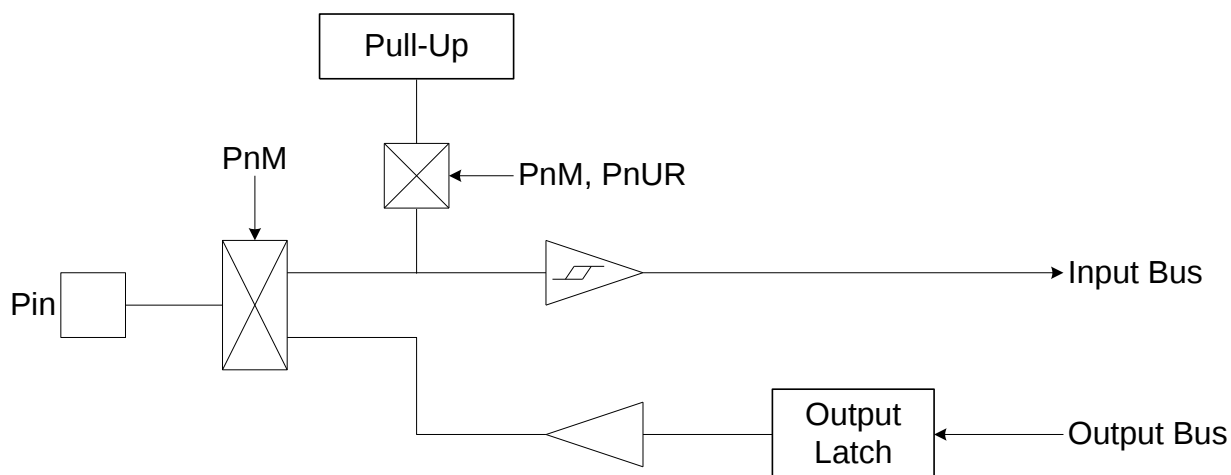


1.4 PIN DESCRIPTIONS

PIN NAME	TYPE	DESCRIPTION
DVDD, DVSS	P	Power supply input pins for digital circuit.
AVDD, AVSS	P	Power supply input pins for analog circuit.
CL+, CL-	P	Charge pump pins for VLCD, connect 0.1uF Cap.
C+, C-	P	Charge pump pins for VLED, connect 1uF Cap.
VPUMP	P	Charge pump output voltage (2 x VDD) for LED regulator, connects 1uF Cap to gnd.
VLED	P	Charge pump regulator output voltage for LED power. Connect 1uF Cap to gnd. VLED = 3.5V, Max output current = 8mA. Operation within VDD form 2.4V~3.6V.
AVDDR	P	Regulator power output pin, Voltage = 2.4V. Connect 1uF Cap to gnd. AVDDR is analog circuit power source.(PGIA, ADC..)
AVE+	P	Regulator output = 2.0/1.5V for Sensor. Maximum output current = 5mA. Connect 1uF Cap to gnd.
ACM	P	ACM Voltage output = 1V. Connect 0.1uF to AVDDR.
R+	AI	ADC positive reference voltage input. R+ input absolutely voltage range is between AVDDR-1V and GND+0.4V. External reference can be applied between R+ and R-.
R-	AI	ADC negative reference voltage input. R- input absolutely voltage range is between AVDDR-1V and GND+0.4V.
AI1+, AI1-	AI	Positive and negative analog input pair of PGIA channels, each input absolutely voltage range is between AVDDR-1V and GND+0.4V.
AI2+, AI2-	AI	Positive and negative analog input pair of PGIA channels, each input absolutely voltage range is between AVDDR-1V and GND+0.4V.
VPP	P	OTP ROM programming pin only. This pin input 6.5V for OTP programming. Keep voltage equal to or lower than VDD when IC normal operation. No reset function.
P0.0 / INT0	I/O	Port 0.0 and share with INT0 trigger pin (Schmitt trigger). Built-in pull-up resistors.
P0.1 / INT1	I/O	Port 0.1 and share with INT1 trigger pin (Schmitt trigger). Built-in pull-up resistors.
P0 [7:0]	I/O	Port0.0~Port 0.7 bi-direction pins. Green mode or sleep mode wakeup pins. Built-in pull-up resistors.
P1 [4:0]	I/O	Port1.0~Port 1.4 bi-direction pins. Built-in pull-up resistors controlled by P1UR. Wakeup controlled by P1W.
P2 [1:0]	I/O	Port2.0~Port 2.1 bi-direction pins. Built-in pull-up resistors controlled by P2UR. Pins share with LXIN/LXOUT.
P5 [4:0]	I/O	Port5.0~Port 5.4 bi-direction pins. Built-in pull-up resistors controlled by P5UR.
COM [3:0]	O	COM0~COM3 LCD driver common port
LBTIN1/LBTIN2	I	LBTIN1 (P51) is internal grounding pin for low battery detection. LBTIN2 (P52) is external input pin for low battery detection. (Comparator input Pin)
SEG0 ~ SEG31	O	LCD driver segment pins
VLCD	P	LCD driver power pin. When R-Type LCD mode selected, VLCD Pin as Input power.
V3	P	2/3 VLCD bias voltage when LCD 1/3 bias selected.
V2	P	1/3 VLCD bias voltage when LCD 1/3 bias selected.

1.5 PIN CIRCUIT DIAGRAMS

Port 0, Port 1, Port2 and Port5 structure:



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

8K words ROM

<i>ROM</i>		
0000H	<i>Reset vector</i>	User reset vector
0001H	<i>General purpose area</i>	Jump to user start address
0002H		Jump to user start address
0003H		Jump to user start address
0004H	<i>Reserved</i>	
0005H		
0006H		
0007H		
0008H	<i>Interrupt vector</i>	User interrupt vector
0009H	<i>General purpose area</i>	User program
.		
000FH		
0010H		
0011H		
.	<i>Reserved</i>	
1FFBH		
1FFCH		
.		
1FFFH		End of user program

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.
                                ...

START:                          ORG      10H
                                ; 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.

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

MY_IRQ:         ; The head of interrupt service routine.
                B0XCH     A, ACCBUF      ; Save ACC in a buffer.
                B0MOV     A, PFLAG
                B0MOV     PFLAGBUF, A    ; Save PFLAG register in a buffer.
                ...
                ...
                B0MOV     A, PFLAGBUF
                B0MOV     PFLAG, A       ; Restore PFLAG register from buffer.
                B0XCH     A, ACCBUF      ; Restore ACC from buffer.
                RETI              ; End of interrupt service routine.
                ...

                ENDP                    ; End of program.
```

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

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

2.1.2.1 LOOK-UP TABLE DESCRIPTION

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

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

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

                ; Increment the index address for next address.
                INCMS      Z          ; Z+1
                JMP        @F         ; Z is not overflow.
                INCMS      Y          ; Z overflow (FFH → 00), → Y=Y+1
                NOP          ;
```

```

@@:          MOVC          ; To lookup data, R = 51H, ACC = 05H.
...          ;
TABLE1:      DW           0035H ; To define a word (16 bits) data.
             DW           5105H
             DW           2012H
             ...

```

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

➤ **Example: INC_YZ macro.**

```

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

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

```

➤ **Example: Modify above example by “INC_YZ” macro.**

```

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

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

```

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

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

```

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

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

        B0BTS1   FC              ; Check the carry flag.
        JMP      GETDATA         ; FC = 0
        INCMS    Y               ; FC = 1. Y+1.
        NOP

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

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

```

2.1.2.2 JUMP TABLE DESCRIPTION

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

When carry flag occurs after executing of “ADD PCL, A”, it will not affect PCH register. Users have to check if the jump table leaps over the ROM page boundary or the listing file generated by SONiX assembly software. If the jump table leaps over the ROM page boundary (e.g. from 0xFFH to 0x00H), move the jump table to the top of next program memory page (0x00H). **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

@@:
MOV      FC
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	IHRC	High Speed Internal 4MHz RC, XIN/XOUT become to P2.0/P2.1 as IO function
	IHRC_RTC	High Speed Internal 4MHz RC, LXIN/LXOUT connect 32768 Crystal
High_Clk_Div	Fhosc/4	High clock Fcpu = IHRC/4 = 1MHz.
	Fhosc/8	High clock Fcpu = IHRC/8 = 500kHz.
	Fhosc/16	High clock Fcpu = IHRC/16 = 250kHz.
	Fhosc/32	High clock Fcpu = IHRC/32 = 125kHz.

- * **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 Fhosc/4.

2.1.4 DATA MEMORY (RAM)

☞ 256 X 8-bit RAM

	Address	RAM Location	
Bank 0	000H	General purpose area	RAM Bank 0
	...		
	...		
	07FH		
Bank 1	080H	System Register	End of Bank 0
	...		
	0FFH		
	100H		
Bank 15	...	General purpose area	RAM Bank 1
	...		
	17FH		
	End of Bank 1		
Bank 15	F00H	LCD RAM Area	RAM Bank 15
	...		
	F1FH		
	End of Bank 15		

2.1.5 SYSTEM REGISTER

2.1.5.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	X	PFLAG	RBANK	-	LCDM1	LCDM2	-	-	-	-	-
9	VREG	AMPM1	AMPM2	ADCM1	ADCM2	ADCM3	LBTM	ADCDH	ADCDL	LEDPCM	-	-	-	-	-	-
A	ROMADRH	ROMADRL	ROMDAH	ROMDAL	ROMCNT	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	P2	-	-	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	P2UR	-	-	P5UR	@HL	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.1.5.2 SYSTEM REGISTER DESCRIPTION

L, H = Working, @LH and ROM addressing register

Y, Z = Working, @YZ and ROM addressing register

PFLAG = ROM Page and Special Flag Register

LCDM1 = LCD Mode Register

AMPM1 = PGIA mode selection register1

ADCM1 = ADC control register1

ADCM3 = ADC control register3

ADCDH = ADC high-byte data buffer

ADCDL = ADC low-byte data buffer

ROMADRH = ISP ROM address

ROMDAH = ISP ROM programming data buffer

ROMCNT = ISP ROM counter

P_NUR = Port N pull-up register

P1W = Port 1 wakeup register

INTEN = Interrupt enable register

WDTR = Watchdog timer register

PCL, PCH = Program counter

T0C = Timer 0 counting register

TC0C = Timer/Counter 0 Counting register

TC1C = Timer/Counter 1 Counting register

@HL= RAM HL indirect addressing index pointer

@YZ= RAM YZ indirect addressing index pointer

STK0~STK7 = Stack 0 ~ stack 7 Buffer

R = Working register and ROM look-up data buffer

X = Working register and ROM look-up data buffer

RBANK = Bank selection register

LCDM2 = LCD mode register

VREG = Voltage Regulators control register

AMPM2 = PGIA mode selection register2

ADCM2 = ADC control register2

LBTM = Low Battery Detect Register

ADCDM = ADC medium-byte data buffer

LEDPCM = Charge Pump LED-regulator control register

ROMADRL = ISP ROM address

ROMDAL = ISP ROM programming data buffer

P_NM = Port N input/output mode register

P_N = Port N data buffer

INTRQ = Interrupt request register

OSCM = Oscillator mode register

TC0R = Timer/Counter 0 auto-reload data buffer

T0M = Timer 0 mode register

TC0M = Timer/Counter 0 mode register

TC1M = Timer/Counter 0 mode register

TC1R = Timer/Counter 1 auto-reload data buffer

STKP= Stack pointer buffer

2.1.5.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Name
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
085H	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0	R/W	X
086H	-	-	-	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	RBNKS3	RBNKS2	RBNKS1	RBNKS0	R/W	RBANK
089H	LCDREF1	LCDREF0	LCDBNK	LCDTYPE	LCDENB	LDCBIAS	LCDRATE	LCDCLK	R/W	LCDM1
08AH	-	BGM	LCDPENB	VPPINTL	VCP3	VCP2	VCP1	VCP0	R/W	LCDM2
090H	BGRENB	ACMSEL	ACMENB	AVESEL	AVENB	AVDDRSEL	AVDDRENB	-	R/W	VREG
091H	CHS3	CHS2	CHS1	CHS0	GS2	GS1	GS0	AMPENB	R/W	AMPM1
092H	INRENB	GX	GR	AMPCKS1	AMPCKS0	PCHPENB	DTENB	DTSEL	R/W	AMPM2
093H	RVS	IRVS3	IRVS2	IRVS1	IRVS0	ADGN1	ADGN0	ADCENB	R/W	ADCM1
094H	-	OSR2	OSR1	OSR0	-	OFSEL1	OFSEL0	DRDY	R/W	ADCM2
095H	-	-	-	ACHPENB	ADCKINV	ADCKS2	ADCKS1	ADCKS0		ADCM3
096H	-	P51IO	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTO	LBTENB	R/W	LBTM
097H	ADCB23	ADCB22	ADCB21	ADCB20	ADCB19	ADCB18	ADCB17	ADCB16	R	ADCDH
098H	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB9	ADCB8	R	ADCDM
099H	ADCB7	ADCB6	ADCB5	ADCB4	ADCB3	ADCB2	ADCB1	ADCB0	R	ADCDL
09AH	-	-	-	VLEDENB	CPCKS2	CPCKS1	CPCKS0	CPRENB	R/W	LEDCPM
0A0H	VPPCHK	-	-	ROMADR12	ROMADR11	ROMADR10	ROMADR9	ROMADR8	R/W	ROMADRH
0A1H	ROMADR7	ROMADR6	ROMADR5	ROMADR4	ROMADR3	ROMADR2	ROMADR1	ROMADR0	R/W	ROMADRL
0A2H	ROMDA15	ROMDA14	ROMDA13	ROMDA12	ROMDA11	ROMDA10	ROMDA9	ROMDA8	R/W	ROMDAH
0A3H	ROMDA7	ROMDA6	ROMDA5	ROMDA4	ROMDA3	ROMDA2	ROMDA1	ROMDA0	R/W	ROMDAL
0A4H	-	-	-	-	-	-	ROMCNT1	ROMCNT0	W	ROMCNT
0B8H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0BFH	-	-	-	-	P01G1	P01G0	P00G1	P00G0	R/W	PEDGE
0C0H	-	-	-	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H	-	-	-	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C2H	-	-	-	-	-	-	P21M	P20M	R/W	P2M
0C5H	-	-	-	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H	ADCIRQ	TC1IRQ	TC0IRQ	T0IRQ	-	-	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	TC1IEN	TC0IEN	T0IEN	-	-	P01IEN	P00IEN	R/W	INTEN
0CAH	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	-	-	-	P14	P13	P12	P11	P10	R/W	P1
0D2H	-	-	-	-	-	-	P21	P20	R/W	P2
0D5H	-	-	-	P54	P53	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0RATE2	T0RATE1	T0RATE0	TC1x8	TC0x8	TC0GN	T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0RATE2	TC0RATE1	TC0RATE0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1RATE2	TC1RATE1	TC1RATE0	TC1CKS	ALOAD1	TC01OUT	PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R	W	P0UR
0E1H	-	-	-	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E2H	-	-	-	-	-	-	P21R	P20R	W	P2UR
0E5H	-	-	-	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	-	-	-	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H

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

*** Note:**

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

2.1.6 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 13-bit binary counter separated into the high-byte 5 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 12.

	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	-	-	-	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
	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

```

END_CLR:

```

CLR      @YZ               ; 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).

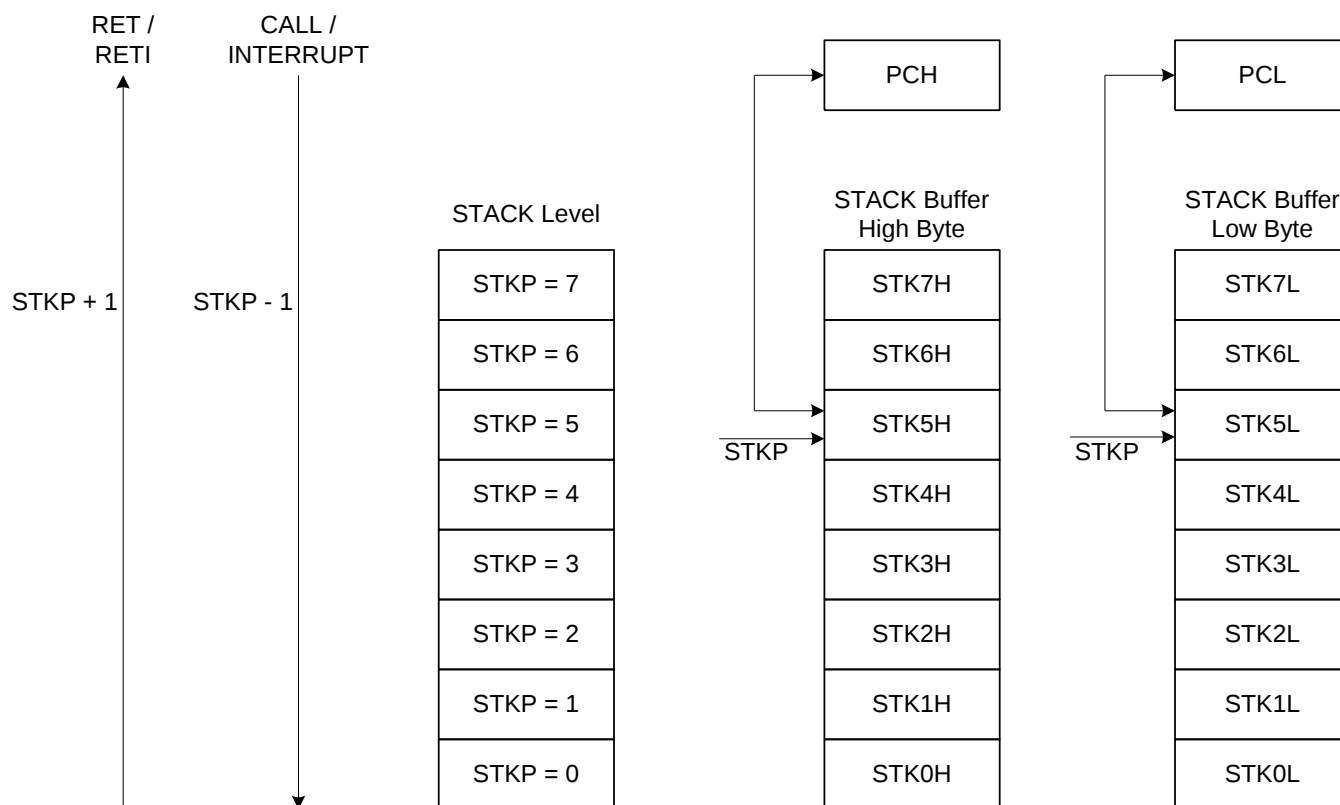
- **Example: Indirectly addressing mode with @YZ register.**

```
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

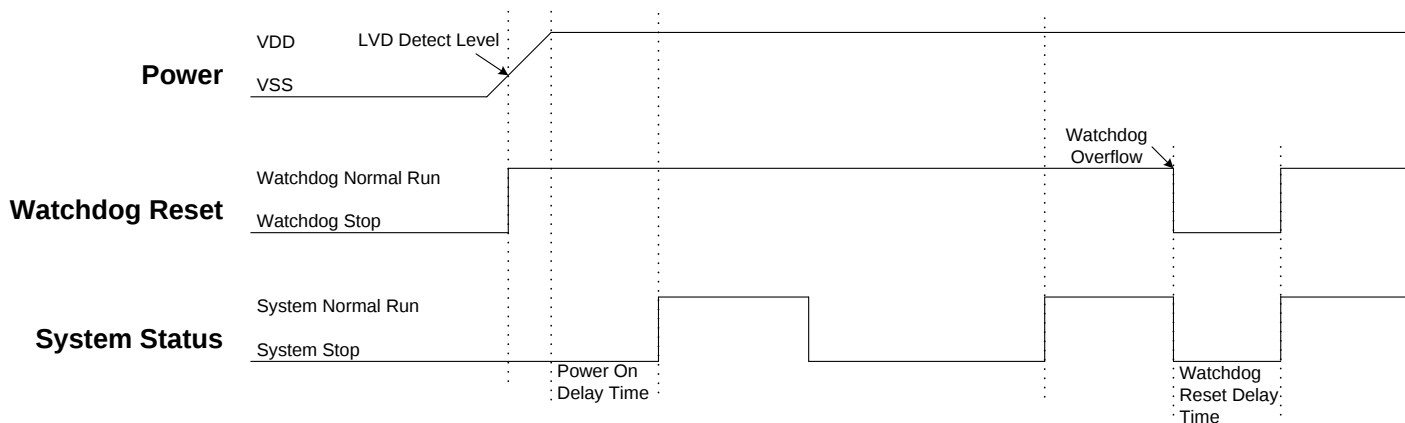
3.1 OVERVIEW

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.

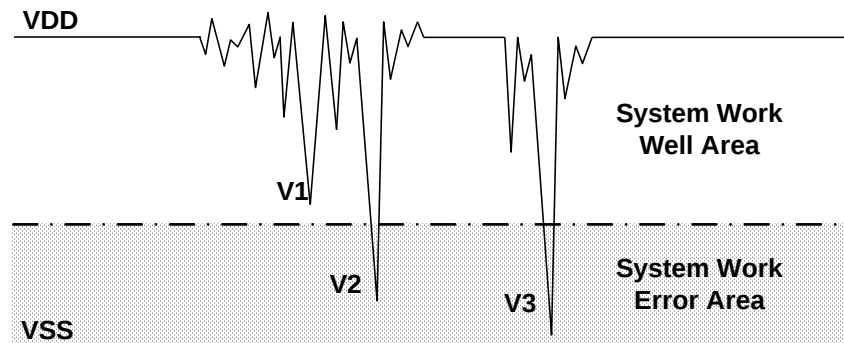
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

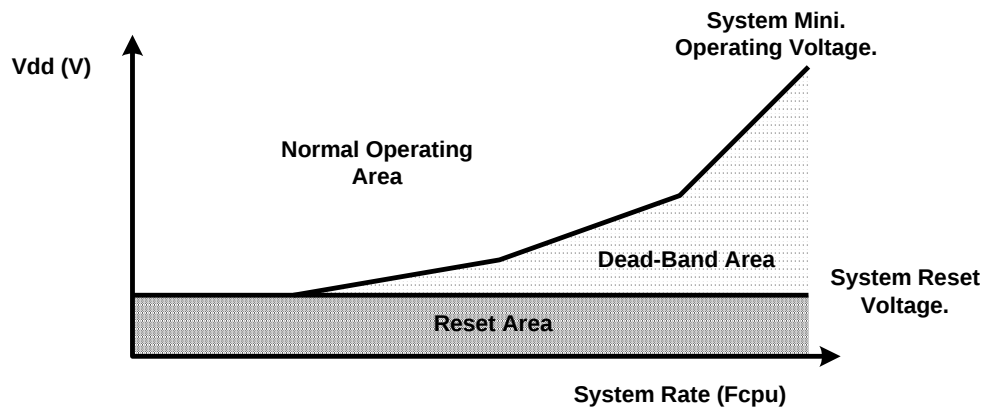
In AC power application, the DC power is regulated from AC power source. This kind of power usually couples with AC noise that makes the DC power dirty. Or the external loading is very heavy, e.g. driving motor. The loading operating

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.



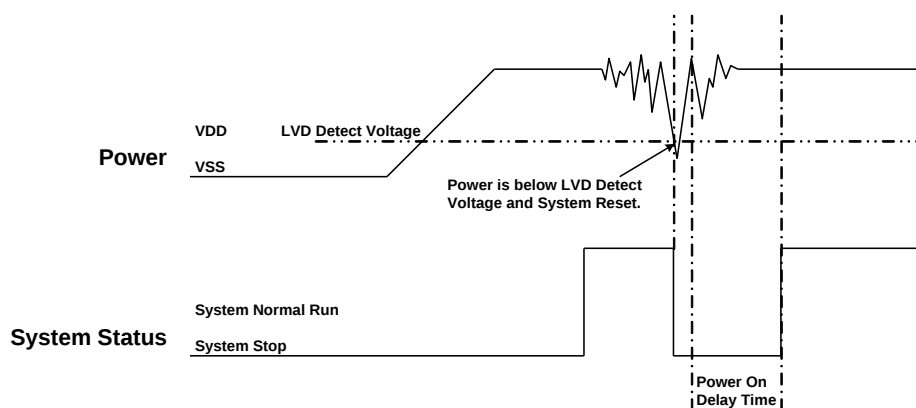
Normally the system operation voltage area is higher than the system reset voltage to VDD, and the reset voltage is decided by LVD detect level. The system minimum operating voltage rises when the system executing rate upper even higher than system reset voltage. The dead-band definition is the system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

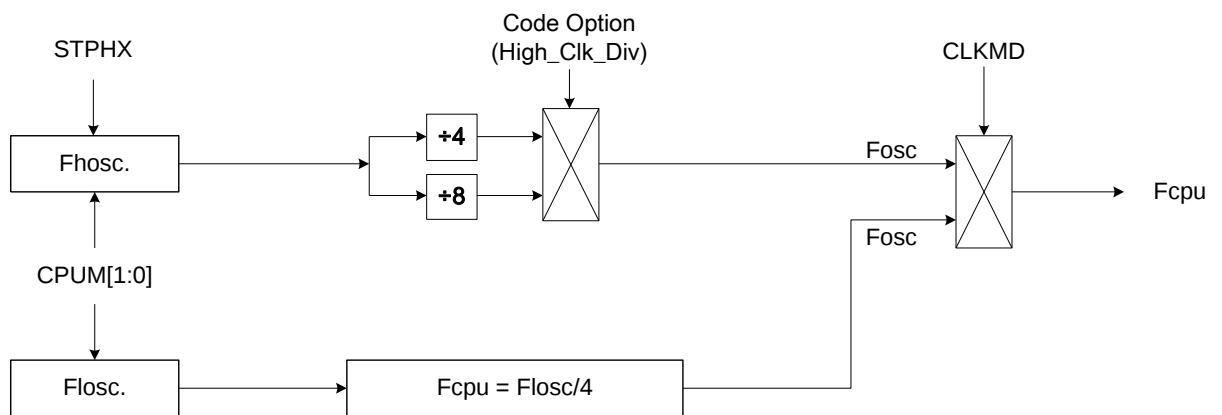
The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is only generated from internal 4MHz high-speed RC oscillator circuit (IHRC 4MHz). The low-speed clock is generated from internal RC oscillator circuit or external 32768Hz Crystal.

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

☞ **Normal Mode (High Clock):** $F_{cpu} = F_{osc} / 4 = 1\text{MHz}$. (Fosc= 4MHz IHRC)
 $F_{cpu} = F_{osc} / 8 = 500\text{ kHz}$. (Fosc= 4MHz IHRC)
 $F_{cpu} = F_{osc} / 16 = 250\text{ kHz}$. (Fosc= 4MHz IHRC)
 $F_{cpu} = F_{osc} / 32 = 125\text{ kHz}$. (Fosc= 4MHz IHRC)

☞ **Slow Mode (Low Clock):** $F_{cpu} = F_{osc}/4$. (Fosc= 32768Hz or ILRC)

4.2 CLOCK BLOCK DIAGRAM



- Fosc: System high clock source is from internal high RC (IHRC).
- Fosc: System low clock source is from internal low RC (ILRC) or external 32k due to Code Option setting.
- 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.

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

- Bit 1 **STPHX**: External high-speed oscillator control bit.
 0 = IHRC free run.
 1 = IHRC free run stop. Internal low-speed RC oscillator is still running.
- Bit 2 **CLKMD**: System high/Low clock mode control bit.
 0 = Normal (dual) mode. System clock is high clock.
 1 = Slow mode. System clock is internal low clock if code option is IHRC. System clock is external 32k if code option is IHRC_RTC.
- Bit[4:3] **CPUM[1:0]**: CPU operating mode control bits.
 00 = normal.
 01 = sleep (power down) mode.
 10 = green mode.
 11 = reserved.

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

```
B0BSET     FSTPHX           ; To stop IHRC only.
```

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

```
B0BSET     FCPUM0           ; To stop IHRC and internal low-speed  

                             ; oscillator called power down mode (sleep mode).
```

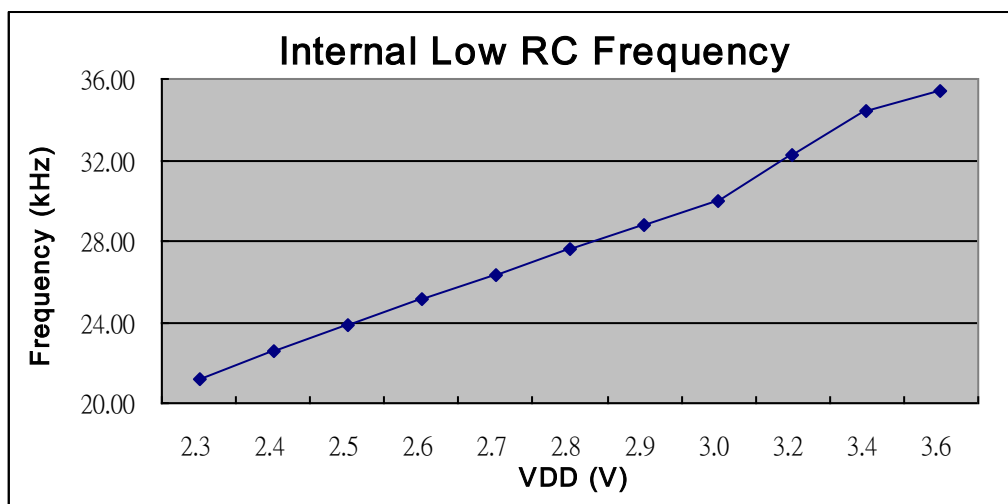
4.4 SYSTEM HIGH CLOCK

The system high clock is only from internal 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.

☞ **Fosc = Internal low RC oscillator (about 32KHz@3.2V).**

☞ **Slow mode Fcpu = Fosc / 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 activates well and system will be reset until watchdog overflow occurring.

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

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

☛ *** Note: The internal low-speed clock can't be turned off individually. It is controlled by CPUM0, CPUM1 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.
```

@@:

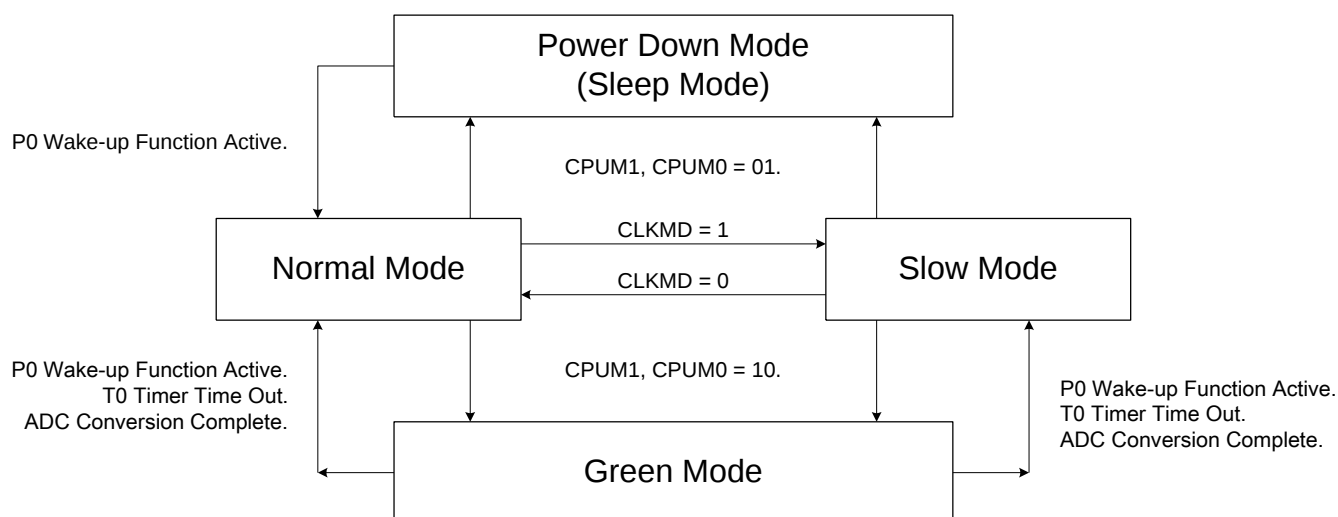
```
B0BSET    P1.0           ; Output Fcpu toggle signal in low-speed clock mode.  
B0BCLR    P1.0           ; Measure the Fcpu frequency by oscilloscope.  
JMP       @B
```

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

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

; Set T0 timer wakeup function.

B0BCLR	FT0IEN	; To disable T0 interrupt service
B0BCLR	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.

$$\text{The Wakeup time} = 1/F_{\text{hosc}} * 64 \text{ (sec)} + \text{high clock start-up time}$$

* **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 change. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.

$$\begin{aligned} \text{The wakeup time} &= 1/F_{\text{hosc}} * 64 = 16\mu\text{s} \quad (F_{\text{hosc}} = 4\text{MHz}) \\ \text{The total wakeup time} &= 16\mu\text{s} + \text{oscillator start-up time (5}\mu\text{s)} \end{aligned}$$

5.3.3 P1W WAKEUP CONTROL REGISTER

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

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

Bit[5:0] **P10W~P14W**: Port 1 wakeup function control bits.

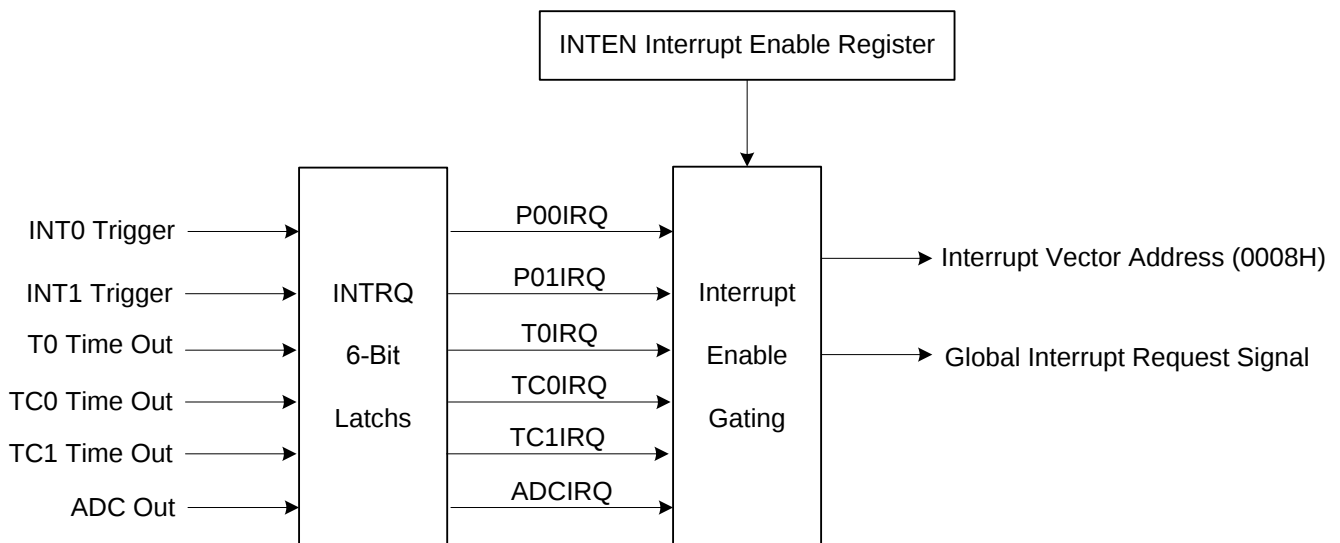
0 = Disable P1n wakeup function.

1 = Enable P1n wakeup function.

6 INTERRUPT

6.1 OVERVIEW

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

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

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

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

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

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

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

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	TC1IRQ	TC0IRQ	T0IRQ	-	-	P01IRQ	P00IRQ
Read/Write	R/W	R/W	R/W	R/W	-	-	R/W	R/W
After reset	0	0	0	0	-	-	0	0

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

Bit 1 **P01IRQ**: External P0.1 interrupt (INT1) request flag.
0 = Non INT1 interrupt request.
1 = INT1 interrupt request.

Bit 4 **T0IRQ**: T0 timer interrupt request flag.

0 = Non T0 interrupt request.
1 = T0 interrupt request.

Bit 5 **TC0IRQ:** TC0 timer interrupt request flag.
 0 = Non TC0 interrupt request.
 1 = TC0 interrupt request.

Bit 6 **TC1IRQ:** TC1 timer interrupt request flag.
 0 = Non TC1 interrupt request.
 1 = TC1 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).**

```
BOBSET      FGIE      ; Enable GIE
```

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

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
```

6.5 MULTI-INTERRUPT OPERATION

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

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

INTP01CHK:          ; Check INT1 interrupt request
    B0BTS1          FP01IEN          ; Check P01IEN
    JMP             INTT0CHK          ; Jump check to next interrupt
    B0BTS0          FP01IRQ          ; Check P01IRQ
    JMP             INTP01

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

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

INTTC1CHK:          ; Check TC1 interrupt request
    B0BTS1          FTC1IEN          ; Check TC1IEN
    JMP             INTTC1CHK        ; Jump check to next interrupt
    B0BTS0          FTC1IRQ          ; Check TC1IRQ
    JMP             INTTC1          ; Jump to TC1 interrupt service routine

INTADCHK:          ; Check ADC interrupt request
    B0BTS1          FADC IEN          ; Check ADC IEN
    JMP             INT_EXIT          ; Jump to exit of IRQ
    B0BTS0          FADC IRQ          ; Check ADC IRQ
    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	-	-	-	P14M	P13M	P12M	P11M	P10M
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2M	-	-	-	-	-	-	P21M	P20M
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	-	P54M	P53M	P52M	P51M	P50M
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

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

- ★ **Note:**
1. Users can program them by bit control instructions (**B0BSET**, **B0BCLR**).
 2. Port 0 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
P0UR	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R
Read/Write	W	W	W	W	W	W	W	W
After Reset	0	0	0	0	0	0	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	-	-	-	P14R	P13R	P12R	P11R	P10R
Read/Write	-	-	-	W	W	W	W	W
After Reset	-	-	-	0	0	0	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2UR	-	-	-	-	-	-	P21R	P20R
Read/Write	-	-	-	-	-	-	W	W
After Reset	-	-	-	-	-	-	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	-	-	-	P54R	P53R	P52R	P51R	P50R
Read/Write	-	-	-	W	W	W	W	W
After Reset	-	-	-	0	0	0	0	0

Bit [7:0] **Pn0R~Pn7R**: I/O port pull-up resistor control bit.

0 = Disable pull-up resistor.

1 = Enable pull-up resistor.

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

➤ Example: I/O Pull up Register

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

7.3 I/O PORT DATA REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	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	-	-	-	P14	P13	P12	P11	P10
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	-	0	0	0	0	0

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2	-	-	-	-	-	-	P21	P20
Read/Write	-	-	-	-	-	-	R/W	R/W
After Reset	-	-	-	-	-	-	0	0

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	-	P54	P53	P52	P51	P50
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	-	0	0	0	0	0

Bit [7:0] **Pn0~Pn7**: I/O port data buffer.

0 = Input low or output low.

1 = Input high or output high.

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

```
B0MOV    A, P0
B0MOV    A, P1
```

```
; Read data from Port 0
; Read data from Port 1
```

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

```
MOV      A, #0FFH
B0MOV    P0, A
B0MOV    P1, A
```

```
; Write data FFH to all Port.
```

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

```
B0BSET   P1.0
B0BCLR   P1.0
```

```
; Set P1.0 to be "1".
; 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.

OCCH	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.
    BOBSET            FWDRST    ; Only one clearing watchdog timer of whole program.
    ...
    CALL        SUB1
    CALL        SUB2
    ...
    ...
    JMP         MAIN

```

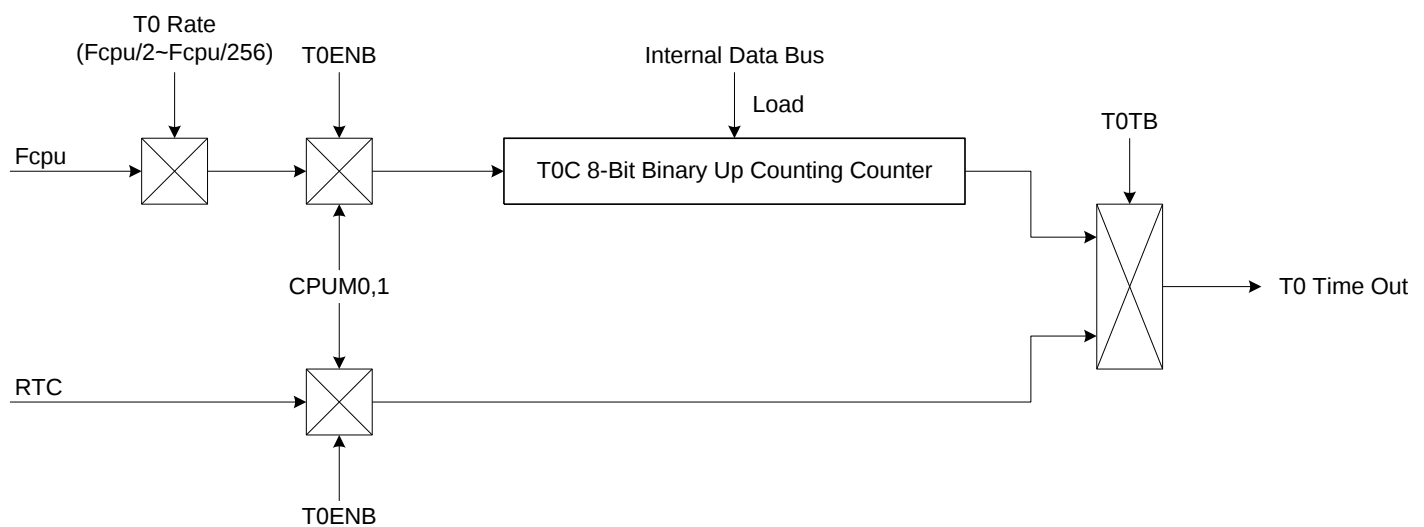
8.2 TIMER 0 (T0)

8.2.1 OVERVIEW

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

The main purposes of the T0 timer are as following.

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



- *** Note:1. In RTC mode, clear T0IRQ must be after 1/2 RTC clock source (32768Hz), or the RTC interval time is error. The delay is about 16us and use T0 interrupt service routine executing time to be the 16us delay time.**
- 2. In RTC mode, the T0 interval time is fixed at 0.5 sec and T0C is 256 counts.
 - 3. T0 timer have T0IRQ clock loss issue in High_Clk code option = "IHRC_RTC" and "T0TB = 1"
- mode. So user have to use main routing polling T0C state by the way, recommend refer to "T0 TIMER WITH RTC FUNCTION OPERATION SEQUENCE" chapter
4. "T0 TIMER WITH RTC FUNCTION OPERATION SEQUENCE" chapter description: main routing
- structure polling T0C value overflow to update RTC time. Main routing interval time as one cycle
 - is not more than 200ms.

8.2.2 T0M MODE REGISTER

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	TC1X8	TC0X8	TC0GN	T0TB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

- Bit 0 **T0TB**: RTC clock source control bit.
0 = Disable RTC (T0 clock source from Fcpu).
1 = Enable RTC.
- Bit 1 **TC0GN**: Enable TC0 Green mode wake up function
0 = Disable.
1 = Enable.
- Bit 2 **TC0X8**: TC0 internal clock source control bit.
0 = TC0 internal clock source is Fcpu. TC0RATE is from Fcpu/2~Fcpu/256.
1 = TC0 internal clock source is Fosc. TC0RATE is from Fosc/1~Fosc/128.
- Bit 3 **TC1X8**: TC1 internal clock source control bit.
0 = TC1 internal clock source is Fcpu. TC1RATE is from Fcpu/2~Fcpu/256.
1 = TC1 internal clock source is Fosc. TC1RATE is from Fosc/1~Fosc/128.
- Bit [6:4] **T0RATE[2:0]**: T0 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
...
110 = fcpu/4.
111 = fcpu/2.
- Bit 7 **T0ENB**: T0 counter control bit.
0 = Disable T0 timer.
1 = Enable T0 timer.

➤
* **Note:** T0RATE is not available in RTC mode. The T0 interval time is fixed at 0.5 sec.

8.2.3 T0C COUNTING REGISTER

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

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

The equation of T0C initial value is as following.

$$T0C \text{ initial value} = 256 - (T0 \text{ interrupt interval time} * \text{input clock})$$

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

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

The basic timer table interval time of T0.

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

- * **Note:** In RTC mode, T0C is 256 counts and generates T0 0.5 sec interval time. Don't change T0C value in RTC mode.

8.2.4 T0 TIMER OPERATION SEQUENCE (High_Clk = IHRC)

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

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

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

☞ **Set T0 timer rate.**

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

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

B0BCLR	FT0TB	; Select T0 Fcpu clock source.
or		
B0BSET	FT0TB	; Select T0 RTC clock source.

☞ **Set T0 interrupt interval time.**

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

☞ **Set T0 timer function mode.**

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

☞ **Enable T0 timer.**

B0BSET	FT0ENB	; Enable T0 timer.
--------	--------	--------------------

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

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

➤ **Declare buffer.**

```
.DATA
        OLDTOC      DS      1
        NEWTOC      DS      1
        T0FLAG      DS      1
        T0IRQFLAG   EQU     T0FLAG.0
```

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

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

➤ **Set T0M register.**

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

➤ **Set T0 clock source from RTC.**

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

➤ **Set T0 interrupt interval time.**

```
        CLR         T0C         ; Clear T0C value.
        CLR         OLDTOC      ; Clear OLDTOC value.
        CLR         NEWTOC      ; Clear NEWTOC value.
        CLR         T0FLAG      ; Clear T0FLAG before execute Main routing.
```

➤ **Disable T0 timer function mode.**

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

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

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

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

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

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

```
CKT_T0CVAL:
        MOV         A, T0C       ; Read T0C value
        MOV         NEWTOC, A    ; Save to NEWTOC
        SUB         A, OLDTOC    ; A sub OLDTOC value.
```

```

                                B0BTS0    FC           ; If FC = 0, borrow
                                JMP        EXIT_CKTT0CVAL: ; If FC = 1, jmp EXIT_CKTT0CVAL

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

```

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

```

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

                                B0BCLR      T0IRQFLAG      ; Clear T0IRQFLAG.
                                CALL        DELAY          ; Call delay time = over 1/32.768ms (for RTC limit).
                                B0BCLR      FT0IRQ         ; Clear FT0IRQ.
                                .
                                CALL        UPDATE_TIME    ; Update time.
                                .

EXIT_CKTT0FLAG:
                                RET                          ; Exit sub-routing.

```

➤ **Into green mode before.**

```

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

```

➤ **Process green mode after wakeup.**

```

INTO_GREENMODE: .
                B0BCLR      FCPUM0
                B0BSET      FCPUM1          ; Into green mode

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

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

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

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

                CALL        UPDATE_TIME    ; Update time.
                .

CKT_OTHER:
                .
                .
                .

```

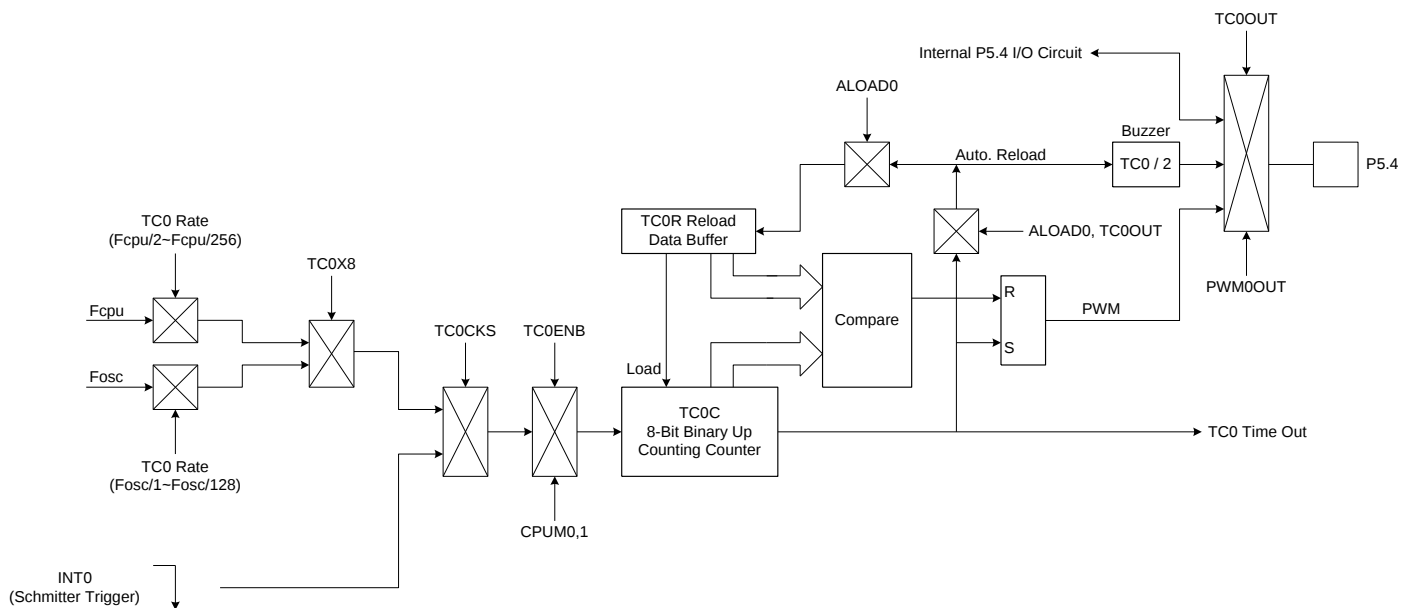
8.3 TIMER/COUNTER 0 (TC0)

8.3.1 OVERVIEW

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

The main purposes of the TC0 timer is as following.

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



8.3.2 TC0M MODE REGISTER

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

- Bit 0 **PWM0OUT**: PWM output control bit.
 0 = Disable PWM output.
 1 = Enable PWM output. PWM duty controlled by TC0OUT, ALOAD0 bits.
- Bit 1 **TC0OUT**: TC0 time out toggle signal output control bit. **Only valid when PWM0OUT = 0.**
 0 = Disable, P5.4 is I/O function.
 1 = Enable, P5.4 is output TC0OUT signal.
- Bit 2 **ALOAD0**: Auto-reload control bit. **Only valid when PWM0OUT = 0.**
 0 = Disable TC0 auto-reload function.
 1 = Enable TC0 auto-reload function.
- Bit 3 **TC0CKS**: TC0 clock source select bit.
 0 = Internal clock (Fcpu or Fosc).
 1 = External clock from P0.0/INT0 pin.
- Bit [6:4] **TC0RATE[2:0]**: TC0 internal clock select bits.

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

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

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

8.3.3 TC1X8, TC0X8, TC0GN FLAGS

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	TC1X8	TC0X8	TC0GN	T0TB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

- Bit 0 **T0TB:** RTC clock source control bit.
0 = Disable RTC (T0 clock source from Fcpu).
1 = Enable RTC.
- Bit 1 **TC0GN:** Enable TC0 Green mode wake up function
0 = Disable.
1 = Enable.
- Bit 2 **TC0X8:** TC0 internal clock source control bit.
0 = TC0 internal clock source is Fcpu. TC0RATE is from Fcpu/2~Fcpu/256.
1 = TC0 internal clock source is Fosc. TC0RATE is from Fosc/1~Fosc/128.
- Bit 3 **TC1X8:** TC1 internal clock source control bit.
0 = TC1 internal clock source is Fcpu. TC1RATE is from Fcpu/2~Fcpu/256.
1 = TC1 internal clock source is Fosc. TC1RATE is from Fosc/1~Fosc/128.
- Bit [6:4] **T0RATE[2:0]:** T0 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
...
110 = fcpu/4.
111 = fcpu/2.
- Bit 7 **T0ENB:** T0 counter control bit.
0 = Disable T0 timer.
1 = Enable T0 timer.



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

8.3.4 TC0C COUNTING REGISTER

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

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

The equation of TC0C initial value is as following.

$$TC0C \text{ initial value} = 256 - (TC0 \text{ interrupt interval time} * \text{input clock})$$

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

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

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

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

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

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

TC0RATE	TC0CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fosc/128	8.192 ms	32 us	1000 ms	3906.25 us
001	Fosc/64	4.096 ms	16 us	500 ms	1953.125 us
010	Fosc/32	2.048 ms	8 us	250 ms	976.563 us
011	Fosc/16	1.024 ms	4 us	125 ms	488.281 us

100	Fosc/8	0.512 ms	2 μ s	62.5 ms	244.141 μ s
101	Fosc/4	0.256 ms	1 μ s	31.25 ms	122.07 μ s
110	Fosc/2	0.128 ms	0.5 μ s	15.625 ms	61.035 μ s
111	Fosc/1	0.064 ms	0.25 μ s	7.813 ms	30.517 μ s

8.3.5 TC0R AUTO-LOAD REGISTER

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

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

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

The equation of TC0R initial value is as following.

$$TC0R \text{ initial value} = N - (TC0 \text{ interrupt interval time} * \text{input clock})$$

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

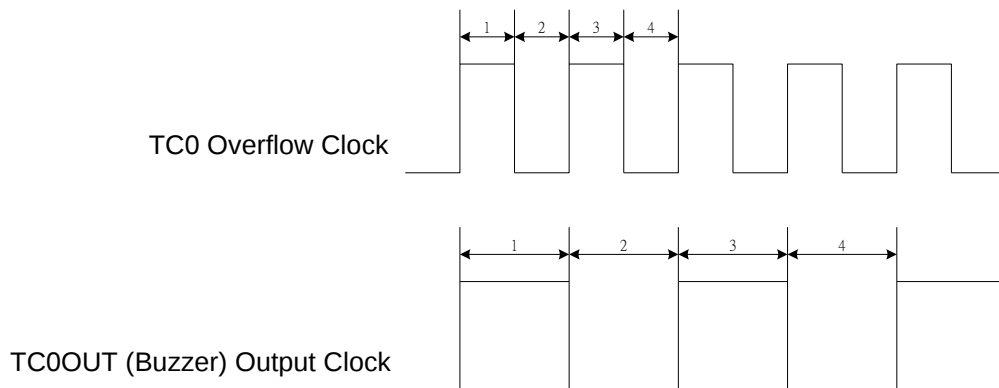
TC0X8	TC0R valid value	TC0R value binary type
0 (Fcpu/2~Fcpu/256)	0x00~0xFF	00000000b~11111111b
1 (Fosc/1~Fosc/128)	0x00~0xFF	00000000b~11111111b

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

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

8.3.6 TC0 CLOCK FREQUENCY OUTPUT (BUZZER)

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



- **Example: Setup TC0OUT output from TC0 to TC0OUT (P5.4). The external high-speed clock is 4MHz. The TC0OUT frequency is 0.5KHz. Because the TC0OUT signal is divided by 2, set the TC0 clock to 1KHz. The TC0 clock source is from external oscillator clock. T0C rate is $F_{cpu}/4$. The $TC0RATE2-TC0RATE1 = 110$. $TC0C = TC0R = 6$.**

```

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

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

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

```

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

8.3.7 TC0 TIMER OPERATION SEQUENCE

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

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

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

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

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

☞ **Set TC0 timer clock source.**

; Select TC0 internal / external clock source.

B0BCLR	FTC0CKS	; Select TC0 internal clock source.
--------	---------	-------------------------------------

or

B0BSET	FTC0CKS	; Select TC0 external clock source.
--------	---------	-------------------------------------

; Select TC0 Fcpu / Fosc internal clock source .

B0BCLR	FTC0X8	; Select TC0 Fcpu internal clock source.
--------	--------	--

or

B0BSET	FTC0X8	; Select TC0 Fosc internal clock source.
--------	--------	--

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

☞ **Set TC0 timer auto-load mode.**

B0BCLR	FALOAD0	; Enable TC0 auto reload function.
--------	---------	------------------------------------

or

B0BSET	FALOAD0	; Disable TC0 auto reload function.
--------	---------	-------------------------------------

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

; Set TC0 interrupt interval time, TC0OUT (Buzzer) frequency or PWM duty.

MOV	A,#7FH	; TC0C and TC0R value is decided by TC0 mode.
B0MOV	TC0C,A	; Set TC0C value.
B0MOV	TC0R,A	; Set TC0R value under auto reload mode or PWM mode.

; In PWM mode, set PWM cycle.

B0BCLR	FALOAD0	; ALOAD0, TC0OUT = 00, PWM cycle boundary is
B0BCLR	FTC0OUT	; 0~255.

or

B0BCLR	FALOAD0	; ALOAD0, TC0OUT = 01, PWM cycle boundary is
B0BSET	FTC0OUT	; 0~63.

or

B0BSET	FALOAD0	; ALOAD0, TC0OUT = 10, PWM cycle boundary is
B0BCLR	FTC0OUT	; 0~31.

or

B0BSET	FALOAD0	; ALOAD0, TC0OUT = 11, PWM cycle boundary is
B0BSET	FTC0OUT	; 0~15.

 Set TC0 timer function mode.

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

 Enable TC0 timer.

	B0BSET	FTC0ENB	; Enable TC0 timer.
--	--------	---------	---------------------

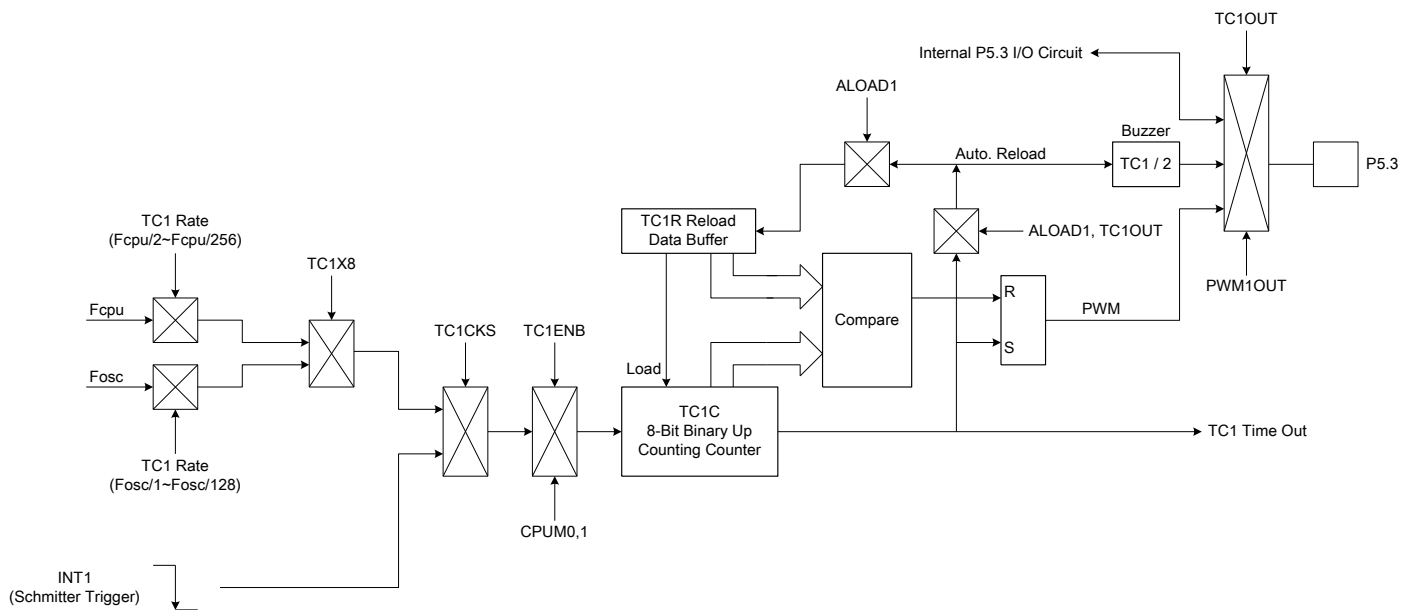
8.4 TIMER/COUNTER 1 (TC1)

8.4.1 OVERVIEW

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

The main purpose of the TC1 timer is as following.

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



8.4.2 TC1M MODE REGISTER

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

- Bit 0 **PWM1OUT:** PWM output control bit.
 0 = Disable PWM output.
 1 = Enable PWM output. PWM duty controlled by TC1OUT, ALOAD1 bits.
- Bit 1 **TC1OUT:** TC1 time out toggle signal output control bit. **Only valid when PWM1OUT = 0.**
 0 = Disable, P5.3 is I/O function.
 1 = Enable, P5.3 is output TC1OUT signal.
- Bit 2 **ALOAD1:** Auto-reload control bit. **Only valid when PWM1OUT = 0.**
 0 = Disable TC1 auto-reload function.
 1 = Enable TC1 auto-reload function.
- Bit 3 **TC1CKS:** TC1 clock source select bit.
 0 = Internal clock (Fcpu or Fosc).
 1 = External clock from P0.1/INT1 pin.
- Bit [6:4] **TC1RATE[2:0]:** TC1 internal clock select bits.

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

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

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

8.4.3 TC1X8, TC0X8, TC0GN FLAGS

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	TC1X8	TC0X8	TC0GN	T0TB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

- Bit 0 **T0TB:** RTC clock source control bit.
0 = Disable RTC (T0 clock source from Fcpu).
1 = Enable RTC.
- Bit 1 **TC0GN:** Enable TC0 Green mode wake up function
0 = Disable.
1 = Enable.
- Bit 2 **TC0X8:** TC0 internal clock source control bit.
0 = TC0 internal clock source is Fcpu. TC0RATE is from Fcpu/2~Fcpu/256.
1 = TC0 internal clock source is Fosc. TC0RATE is from Fosc/1~Fosc/128.
- Bit 3 **TC1X8:** TC1 internal clock source control bit.
0 = TC1 internal clock source is Fcpu. TC1RATE is from Fcpu/2~Fcpu/256.
1 = TC1 internal clock source is Fosc. TC1RATE is from Fosc/1~Fosc/128.
- Bit [6:4] **T0RATE[2:0]:** T0 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
...
110 = fcpu/4.
111 = fcpu/2.
- Bit 7 **T0ENB:** T0 counter control bit.
0 = Disable T0 timer.
1 = Enable T0 timer.



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

8.4.4 TC1C COUNTING REGISTER

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

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

The equation of TC1C initial value is as following.

$$TC1C \text{ initial value} = 256 - (TC1 \text{ interrupt interval time} * \text{input clock})$$

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

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

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

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

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

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

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

111	Fosc/1	0.064 ms	0.25 us	7.813 ms	30.517us
-----	--------	----------	---------	----------	----------

8.4.5 TC1R AUTO-LOAD REGISTER

TC1 timer is with auto-load function controlled by ALOAD1 bit of TC1M. When TC1C overflow occurring, TC1R value will load to TC1C by system. It is easy to generate an accurate time, and users don't reset TC1C during interrupt service routine.

➤ *** Note: Under PWM mode, auto-load is enabled automatically. The ALOAD1 bit is selecting overflow boundary.**

ODEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1R	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The equation of TC1R initial value is as following.

$$TC1R \text{ initial value} = N - (TC1 \text{ interrupt interval time} * \text{input clock})$$

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

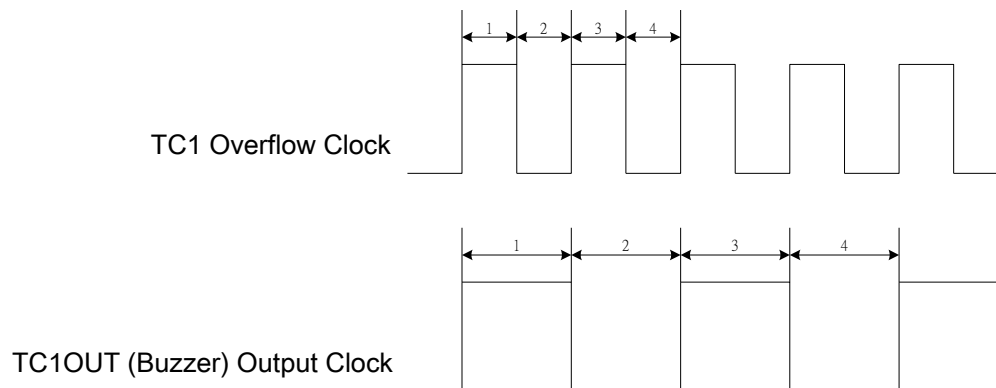
TC1X8	TC1R valid value	TC1R value binary type
0 (Fcpu/2~Fcpu/256)	0x00~0xFF	00000000b~11111111b
1 (Fosc/1~Fosc/128)	0x00~0xFF	00000000b~11111111b

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

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

8.4.6 TC1 CLOCK FREQUENCY OUTPUT (BUZZER)

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



- **Example: Setup TC1OUT output from TC1 to TC1OUT (P5.3). The external high-speed clock is 4MHz. The TC1OUT frequency is 0.5KHz. Because the TC1OUT signal is divided by 2, set the TC1 clock to 1KHz. The TC1 clock source is from external oscillator clock. TC1 rate is $F_{cpu}/4$. The $TC1RATE2-TC1RATE1 = 110$. $TC1C = TC1R = 6$.**

```

MOV      A,#01100000B
B0MOV    TC1M,A           ; Set the TC1 rate to  $F_{cpu}/4$ 

MOV      A,#6
B0MOV    TC1C,A           ; Set the auto-reload reference value
B0MOV    TC1R,A

B0BSET   FTC1OUT          ; Enable TC1 output to P5.3 and disable P5.3 I/O function
B0BSET   FALOAD1          ; Enable TC1 auto-reload function
B0BSET   FTC1ENB          ; Enable TC1 timer

```

➤ *** Note: Buzzer output is enable, and "PWM1OUT" must be "0".**

8.4.7 TC1 TIMER OPERATION SEQUENCE

TC1 timer operation includes timer interrupt, event counter, TC1OUT and PWM. The sequence of setup TC1 timer is as following.

☞ **Stop TC1 timer counting, disable TC1 interrupt function and clear TC1 interrupt request flag.**

B0BCLR	FTC1ENB	; TC1 timer, TC1OUT and PWM stop.
B0BCLR	FTC1IEN	; TC1 interrupt function is disabled.
B0BCLR	FTC1IRQ	; TC1 interrupt request flag is cleared.

☞ **Set TC1 timer rate. (Besides event counter mode.)**

MOV	A, #0xxx0000b	; The TC1 rate control bits exist in bit4~bit6 of TC1M. The
		; value is from x000xxxxb~x111xxxxb.
B0MOV	TC1M,A	; TC1 interrupt function is disabled.

☞ **Set TC1 timer clock source.**

; Select TC1 internal / external clock source.

B0BCLR	FTC1CKS	; Select TC1 internal clock source.
--------	---------	-------------------------------------

or

B0BSET	FTC1CKS	; Select TC1 external clock source.
--------	---------	-------------------------------------

; Select TC1 Fcpu / Fosc internal clock source .

B0BCLR	FTC1X8	; Select TC1 Fcpu internal clock source.
--------	--------	--

or

B0BSET	FTC1X8	; Select TC1 Fosc internal clock source.
--------	--------	--

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

☞ **Set TC1 timer auto-load mode.**

B0BCLR	FALOAD1	; Enable TC1 auto reload function.
--------	---------	------------------------------------

or

B0BSET	FALOAD1	; Disable TC1 auto reload function.
--------	---------	-------------------------------------

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

; Set TC1 interrupt interval time, TC1OUT (Buzzer) frequency or PWM duty.

MOV	A, #7FH	; TC1C and TC1R value is decided by TC1 mode.
B0MOV	TC1C,A	; Set TC1C value.
B0MOV	TC1R,A	; Set TC1R value under auto reload mode or PWM mode.

; In PWM mode, set PWM cycle.

B0BCLR	FALOAD1	; ALOAD1, TC1OUT = 00, PWM cycle boundary is
B0BCLR	FTC1OUT	; 0~255.

or

B0BCLR	FALOAD1	; ALOAD1, TC1OUT = 01, PWM cycle boundary is
B0BSET	FTC1OUT	; 0~63.

or

B0BSET	FALOAD1	; ALOAD1, TC1OUT = 10, PWM cycle boundary is
B0BCLR	FTC1OUT	; 0~31.

or

B0BSET	FALOAD1	; ALOAD1, TC1OUT = 11, PWM cycle boundary is
B0BSET	FTC1OUT	; 0~15.

 Set TC1 timer function mode.

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

 Enable TC1 timer.

	B0BSET	FTC1ENB	; Enable TC1 timer.
--	--------	---------	---------------------

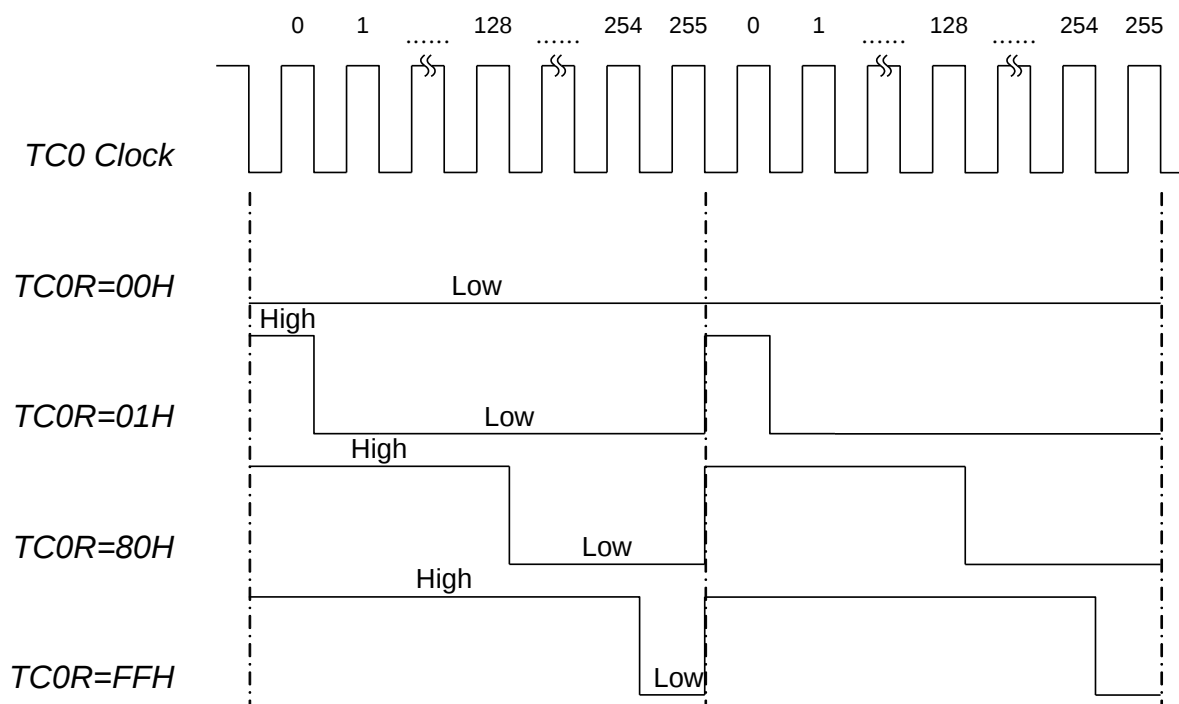
8.5 PWM0 MODE

8.5.1 OVERVIEW

PWM function is generated by TC0 timer counter and output the PWM signal to PWM0OUT pin (P5.4). The 8-bit counter counts modulus 256 bits. The value of the 8-bit counter (TC0C) is compared to the contents of the reference register (TC0R). When the reference register value (TC0R) is equal to the counter value (TC0C), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The ratio (duty) of the PWM0 output is TC0R/256.

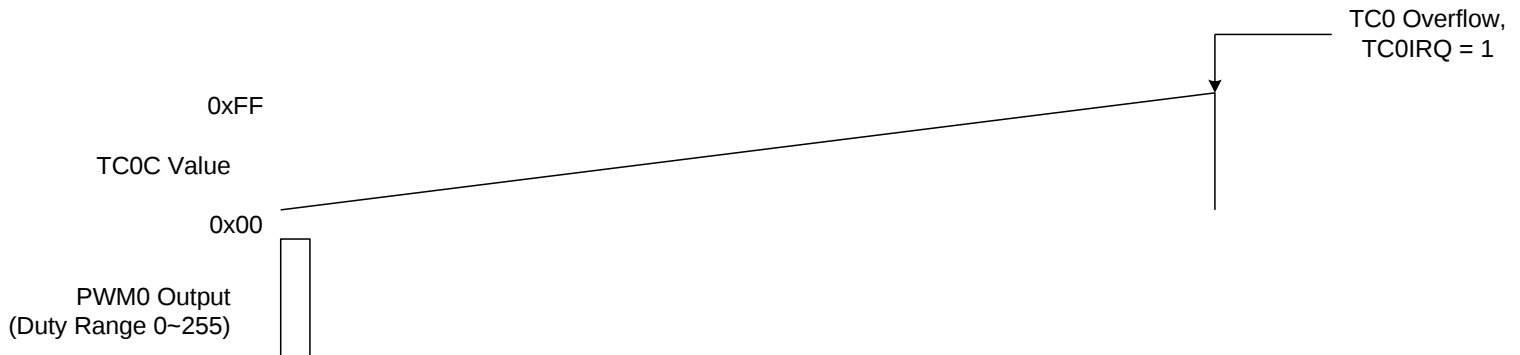
PWM duty range	TC0C valid value	TC0R valid bits value	MAX. PWM Frequency (F _{cpu} = 4MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count

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



8.5.2 TC0IRQ AND PWM DUTY

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



8.5.3 PWM PROGRAM EXAMPLE

- **Example: Setup PWM0 output from TC0 to PWM0OUT (P5.4).** The external high-speed oscillator clock is 4MHz. $F_{cpu} = F_{osc}/4$. The duty of PWM is 30/256. The PWM frequency is about 1KHz. The PWM clock source is from external oscillator clock. TC0 rate is $F_{cpu}/4$. The $TC0RATE2 \sim TC0RATE1 = 110$. $TC0C = TC0R = 30$.

```

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

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

B0BSET   FPWM0OUT         ; Enable PWM0 output to P5.4 and disable P5.4 I/O function
B0BSET   FTC0ENB          ; Enable TC0 timer

```

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

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

```

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

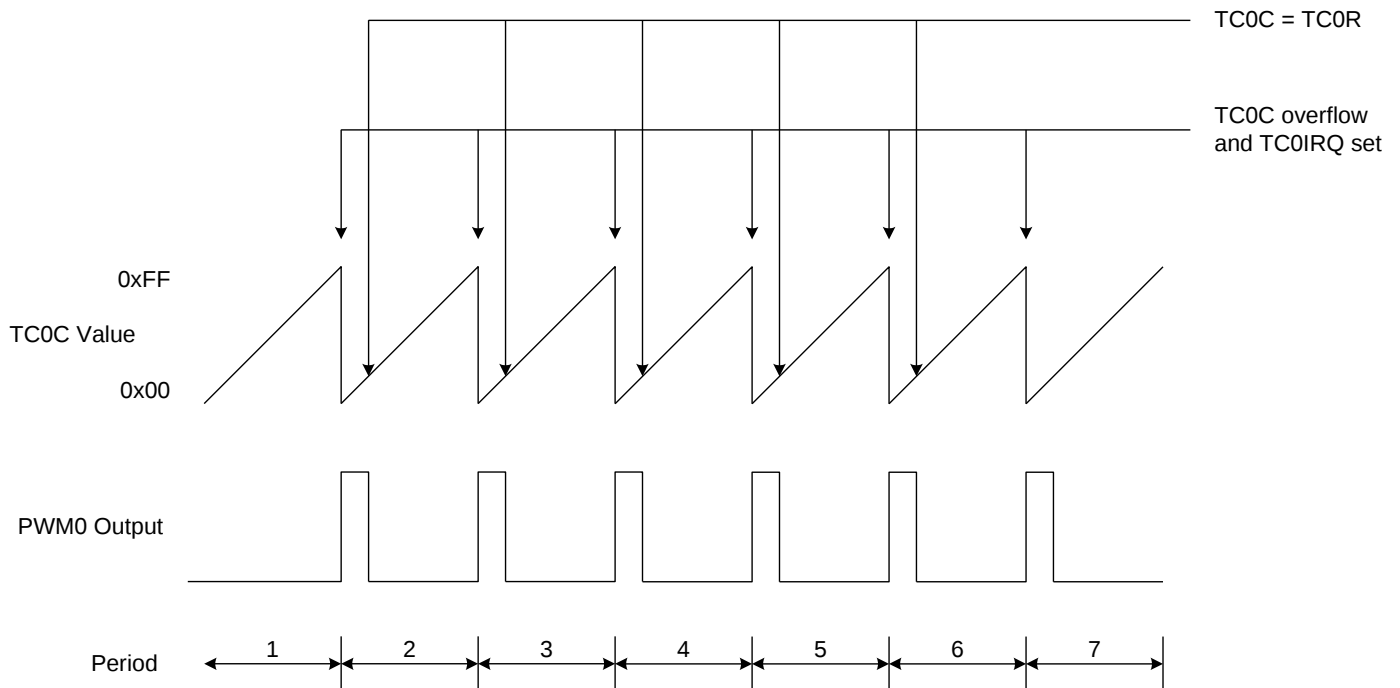
INCMS    BUF0              ; Get the new TC0R value from the BUF0 buffer defined by
NOP                               ; programming.
B0MOV    A, BUF0
B0MOV    TC0R, A

```

➤ **Note: The PWM can work with interrupt request.**

8.5.4 PWM DUTY CHANGING NOTICE

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



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

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

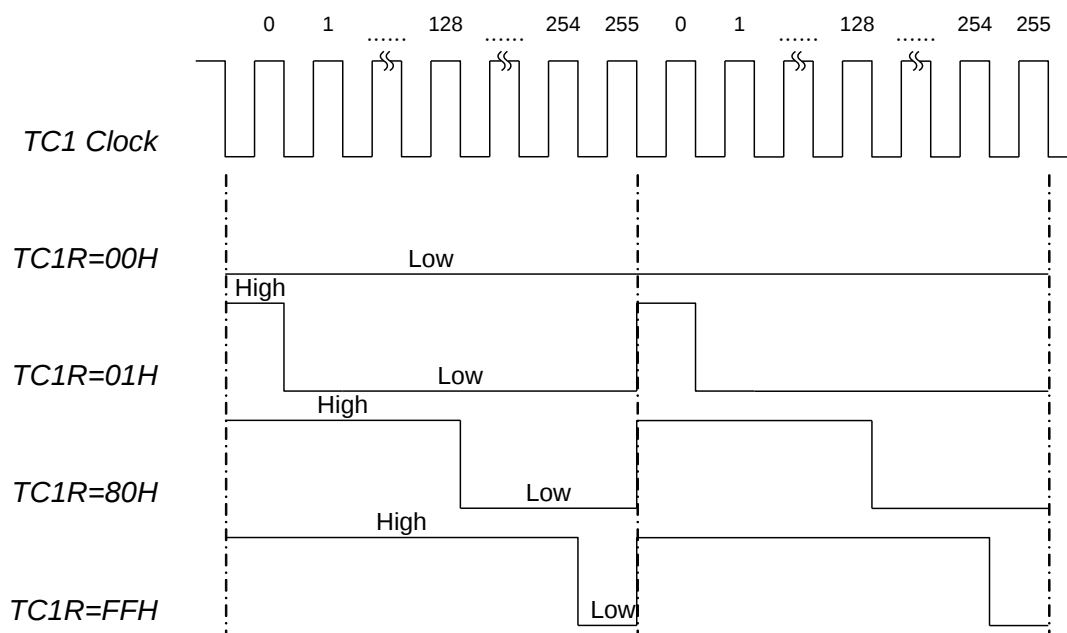
8.6 PWM1 MODE

8.6.1 OVERVIEW

PWM function is generated by TC1 timer counter and output the PWM signal to PWM1OUT pin (P5.3). The 8-bit counter counts modulus 256 bits. The value of the 8-bit counter (TC1C) is compared to the contents of the reference register (TC1R). When the reference register value (TC1R) is equal to the counter value (TC1C), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The ratio (duty) of the PWM1 output is $TC1R/256$,

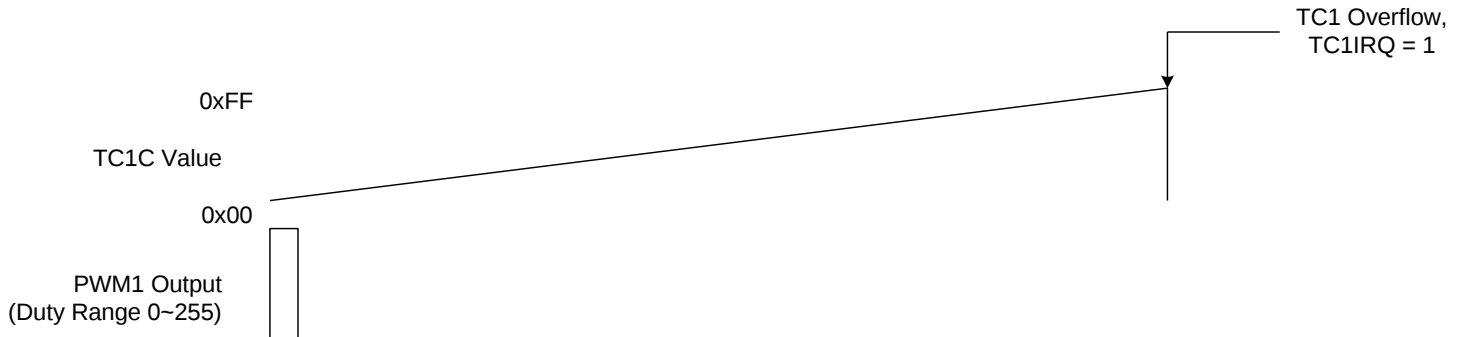
PWM duty range	TC1C valid value	TC1R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count

The Output duty of PWM is with different TC1R. Duty range is from 0/256~255/256.



8.6.2 TC1IRQ AND PWM DUTY

In PWM mode, the frequency of TC1IRQ is depended on PWM duty range. From following diagram, the TC1IRQ frequency is related with PWM duty.



8.6.3 PWM PROGRAM EXAMPLE

- **Example: Setup PWM1 output from TC1 to PWM1OUT (P5.3).** The external high-speed oscillator clock is 4MHz. $F_{cpu} = F_{osc}/4$. The duty of PWM is 30/256. The PWM frequency is about 1KHz. The PWM clock source is from external oscillator clock. TC1 rate is $F_{cpu}/4$. The $TC1RATE2-TC1RATE1 = 110$. $TC1C = TC1R = 30$.

```

MOV      A,#01100000B
B0MOV    TC1M,A           ; Set the TC1 rate to Fcpu/4

MOV      A,#30
B0MOV    TC1C,A           ; Set the PWM duty to 30/256
B0MOV    TC1R,A

B0BSET   FPWM1OUT         ; Enable PWM1 output to P5.3 and disable P5.3 I/O function
B0BSET   FTC1ENB          ; Enable TC1 timer

```

➤ **Note: The TC1R is write-only register. Don't process them using INCMS, DECMS instructions.**

- **Example: Modify TC1R registers' value.**

```

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

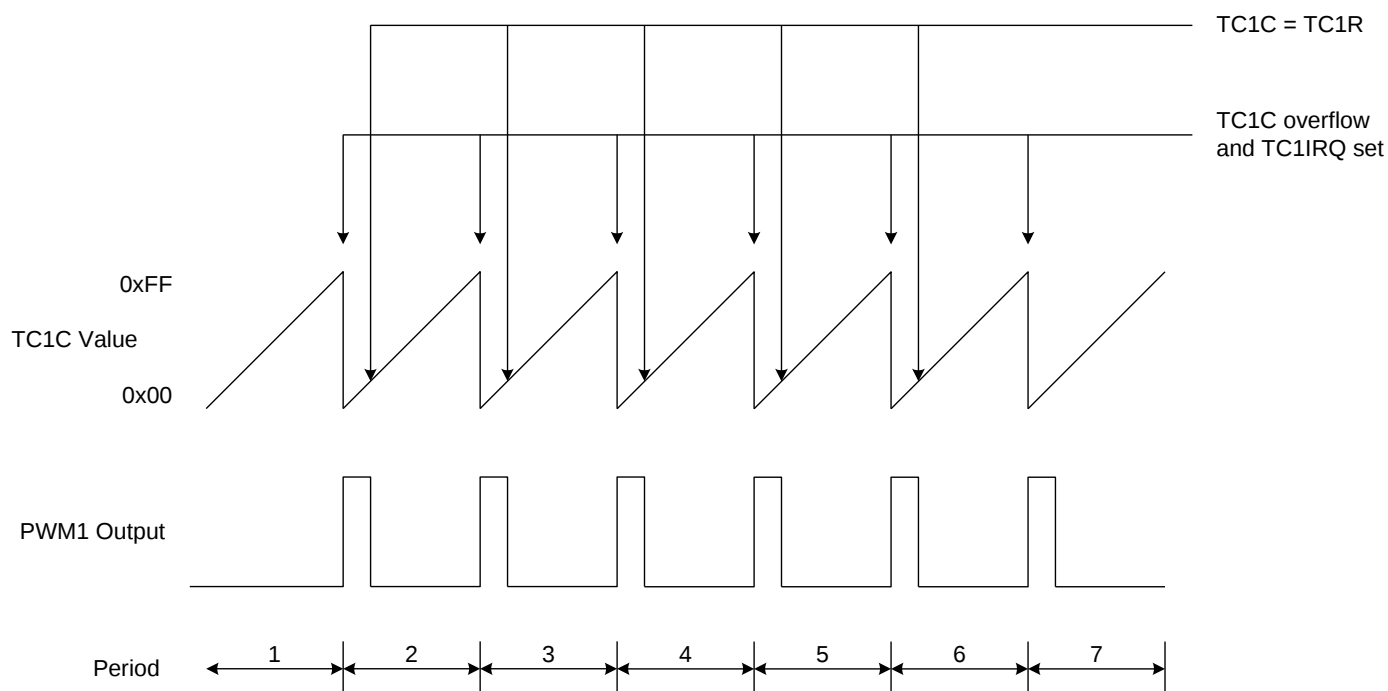
INCMS    BUF0             ; Get the new TC1R value from the BUF0 buffer defined by
NOP                                     ; programming.
B0MOV    A, BUF0
B0MOV    TC1R, A

```

➤ **Note: The PWM can work with interrupt request.**

8.6.4 PWM1 DUTY CHANGING NOTICE

In PWM mode, the system will compare TC1C and TC1R all the time. When $TC1C < TC1R$, the PWM will output logic "High", when $TC1C \geq TC1R$, the PWM will output logic "Low". If TC1C is changed in certain period, the PWM duty will change immediately. If TC1R is fixed all the time, the PWM waveform is also the same.



Above diagram is shown the waveform with fixed TC1R. In every TC1C overflow PWM output "High", when $TC1C \geq TC1R$ PWM output "Low".

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

9 LCD DRIVER

LCD driver includes R-type and C-type structures with 4 common pins and 32 segment pins in the SN8P2949. 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 128 dots LCD driver. LCD power and bias voltage can be adjusted by additional external bias circuit in R-type LCD driver, or adjusted by register setting internal charge pump in C-type LCD driver.

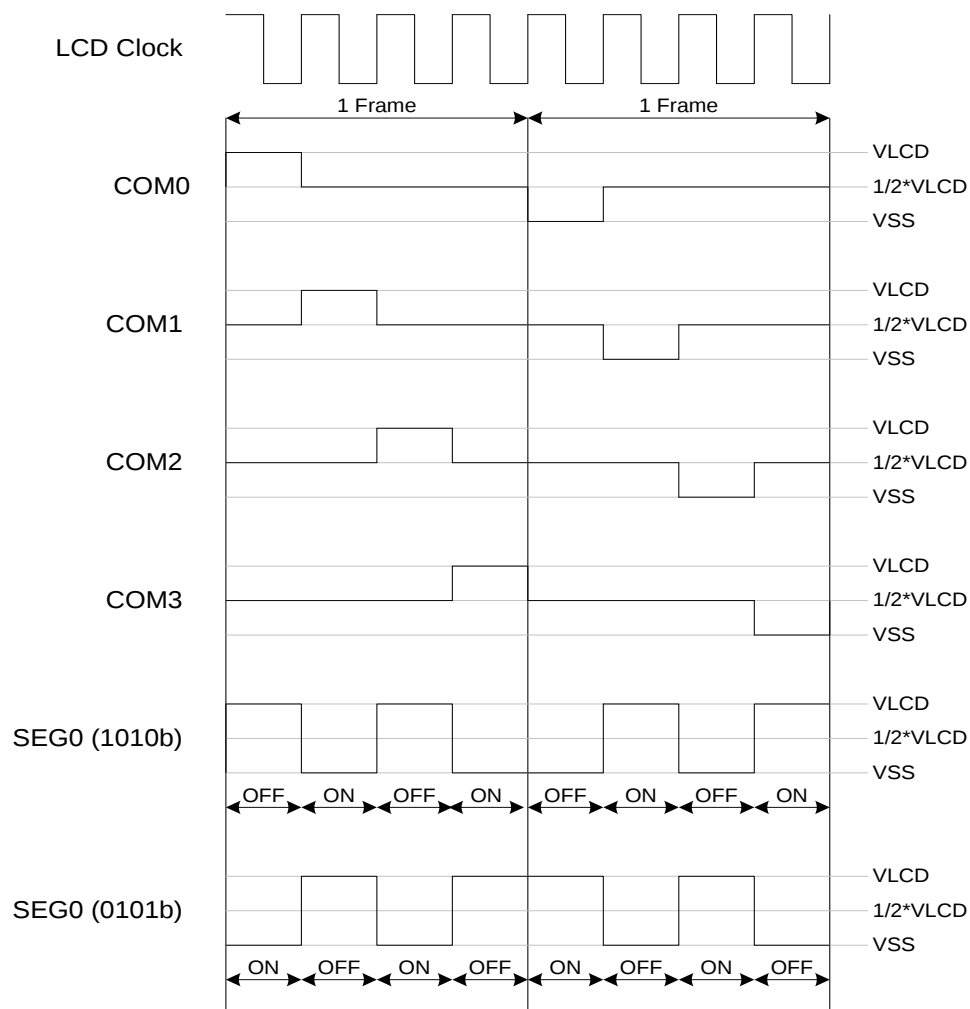
9.1 LCD TIMING

LCD Timing Table

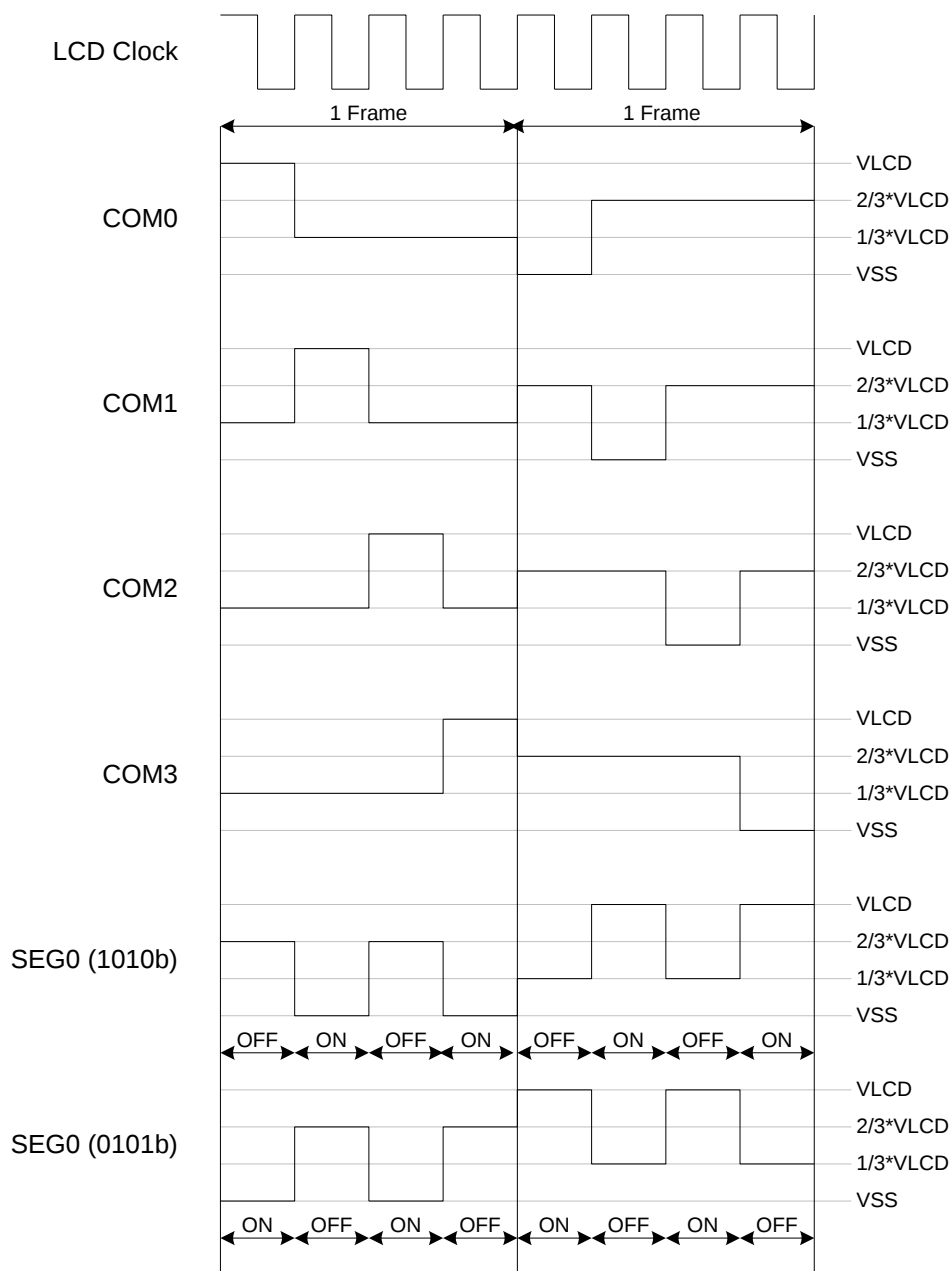
LCDCLK	LCD clock source	LCDRATE	LCD Clock	Frame = LCD clock/4	Note
0	Fhosc	X	$4\text{MHz} / (2^{14}) = 244\text{Hz}$	$244\text{Hz}/4 = 61\text{Hz}$	IHRC= 4MHz
1	Fosc	0	$\text{Fosc} / 128 = 250\text{Hz}$	$256\text{Hz}/4 = 64\text{Hz}$	ILRC = 32kHz@3.3V Or 32768Hz Crystal
1	Fosc	1	$\text{Fosc} / 64 = 500\text{Hz}$	$512\text{Hz}/4 = 128\text{Hz}$	

Note_1: Fosc=ILRC, 32kHz@3.3V (Code Option=IHRC)

Note_2: Fosc=32kHz Crystal (Code Option=IHRC_RTC)



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	LCDREF1	LCDREF0	LCDBNK	LCDDTYPE	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 = $IHRC/(2^{16}) = 61\text{Hz}$.
C-Type CP clock = $IHRC/64 = 62.5\text{kHz}$.
1 = LCD Frame Rate = $ILRC/512 = 64\text{Hz}$ (LCDRATE=0), or $ILRC/256 = 128\text{Hz}$ (LCDRATE=1).
C-Type CP clock = $ILRC = \sim 32\text{kHz}$.
- Bit1 **LCDRATE**: LCD clock rate control when LCDCLK=1 (LCD clock source from ILRC).
0 = LCD Frame Rate = $ILRC / 512 = 64\text{Hz}$.
1 = LCD Frame Rate = $ILRC / 256 = 128\text{Hz}$.
- 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, COM/SEG No output waveform.
1 = Enable, COM/SEN output waveform.
- Bit4 **LCDDTYPE**: LCD R-type or C-type control bit.
0 = R Type
1 = C Type.
- Bit5 **LCDBNK**: LCD blank control bit.
0 = Normal display
1 = All of the LCD dots off.
- Bit[7:6] **LCDREF[1,0]**: R-type LCD Resistor Selection bit for LCD bias voltage division
00 = 400k resistance
01 = 200k resistance
10 = 100k resistance
11 = 33.3k resistance

R-Type and C-Type LCD driver control:

	LCDPENB	LCDDTYPE	LCDENB	BGM	LCDREF[1:0]
R-Type setting	0	0	1	X	Available
C-Type setting	1	1	1	1	Not available
LCD disable setting for power saving (In Green or Sleep mode)	0	X	0	X	X

* **Note_1:** When LCD disable in green or sleep mode, Please Set LCDENB and LCDPENB as "0" for power saving.

9.3 LCDM2 REGISTER

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

VCP[3:0]	1/3 bias Condition			1/2 bias Condition		
	V2	V3	VLCD	V2	V3	VLCD
0000	0.9V	1.8V	2.7V	1.35V	1.35V	2.7V
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	Reserved			Reserved		
1111	-	-	-	2.3V	4.6V	6.9V*

*VLCD output 6.9V for ISP VPP Voltage

Bit4 **VPPINTL:** Internal VPP Control Bit.
0 = VPP no voltage source.
1 = VPP short to VLCD internally for ISP function.
 (C-Type LCD CP output high voltage 6.9V and short to VPP Pin for ISP function.)

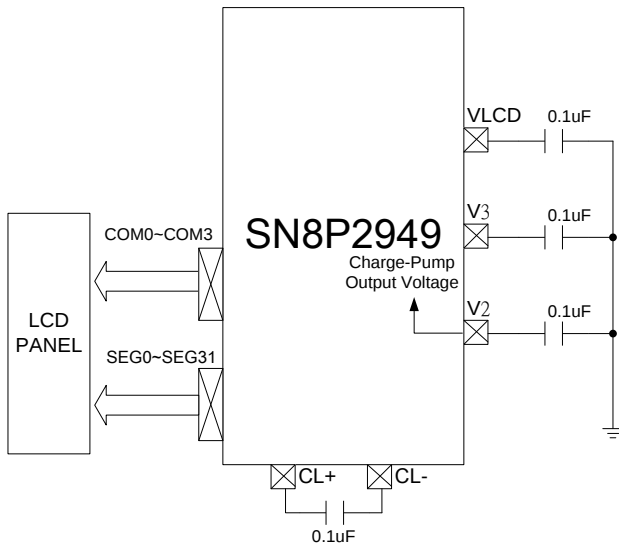
Bit5 **LCDPENB:** C-Type LCD Pump Enable Bit.
0 = Pump disable
1 = Pump enable.

Bit6 **BGM:** Band Gap Selection Bit for LCD Reference Voltage.
0 = Reserved.
1 = System Band Gap Voltage reference. (Always Set "1")

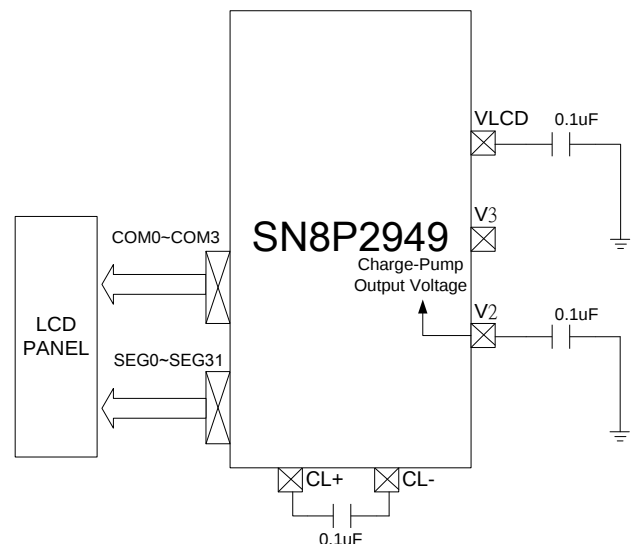
- * **Note_1:** When VCP[3:0]=1111, LCDBIAS=1, LCDTYPE=1 and LCDPENB=1, LCD charge pump is applied to generating VPP voltage for ISP function without additional 6.5V input. Please don't enable LCDENB to protect LCD panel without high voltage present at COM and SEG.
- * **Note_2:** VCP and VPPINTL are 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 6.5V for ISP purpose, VLCD/V3/V2 must connect capacitor to DVSS with 0.1uf individually, and CL+/CL- is also connect 0.1uf.
- * **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

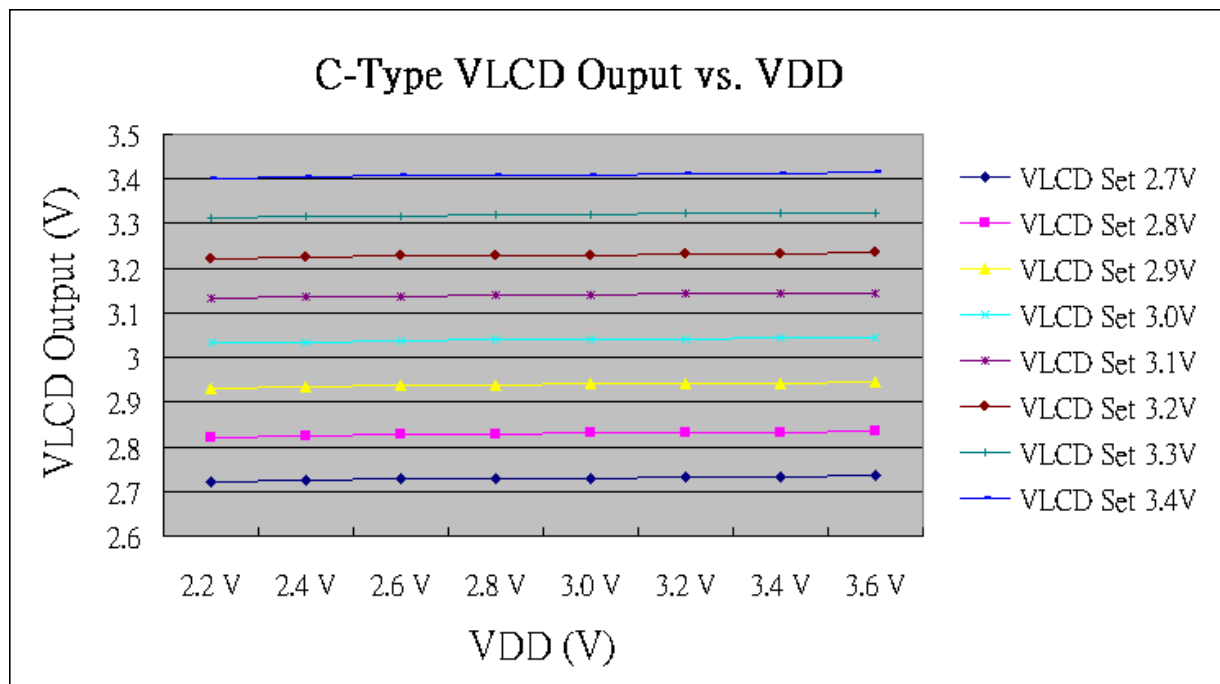
The LCD C-type driver mode is support 1/3 and 1/2 bias LCD panel. The LCD power (VLCD) is supplied by internal LCD charge-pump. The C-Type LCD charge-pump voltage level is following VLCD voltage. V2 is the charge pump source which level is $1/3 \times \text{VLCD}$. V3 is 2 times of V2 by charge pump which level is $2/3 \times \text{VLCD}$. 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 LCD driver mode, VLCD power is not internal connected to VDD.
- * **Note2:** 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.
- * **Note3:** VLCD output voltage can be set from 2.7V to 3.4V and with $\pm 0.2\text{V}$ accuracy.

➤ **Example Code:** Enable C-Type LCD function. (Set VLCD = 3.0V, 1/3 bias mode)

C-Type LCD Setting:

```

B0BCLR    FLCDBIAS    ; Set 1/3 Bias
B0BCLR    FVCP3       ; Set VLCD = 3V
B0BCLR    FVCP2       ; Set VLCD = 3V
B0BSET    FVCP1       ; Set VLCD = 3V
B0BSET    FVCP0       ; Set VLCD = 3V
B0BSET    FBGM        ; Must Set BGM=1.
B0BSET    FLCDTYPE    ; C-Type LCD.
B0BSET    FLCDPENB    ; CP Enable.
LcdPumpStart        ; C-Type pump startup Marco, Must execute it.
                    ; use Marco "LcdPumpStart"
B0BSET    FLCDENB     ; Enable LCD driver. COM/SEG output waveform.
Call      Set_LCD_RAM ; Set LCD RAM to display LCD panel
....

```

- * **Note_1:** The Procedure to execute C-Type LCD driver mode is setting BGM=LCDTYPE=LCDPENB=1, Bias , and VCP[3:0] first, and then must use Macro "**LcdPumpStart**" to execute and check pump startup .
- * **Note_2:** Before use Marco "**LcdPumpStart**", please include 2949_Macro.h file first.

SYM.	PARAMETER	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
I _{LCDC}	C-Type LCD Driver Operation supply current (without LCD Panel)	Temperature: 25°C. VDD = 3V. Clock : ILRC (LCDCLK=1) Current: BGM(1) + Pump + LCD driver	-	120	150	uA
I _{LCDR}	R-Type LCD Driver Operation supply current (without LCD Panel)	Temperature: 25°C. VDD = 3V. (Internal R=400k, 1/3 bias)	-	5	8	
V _{LCD}	VCP[3:0] = 0011	Temperature: 0~70°C. Vdd = 2.2 ~ 3.6V. C-Type LCD	2.8	3	3.2	V
V _{VPPL}	Internal VPP Generation	Temperature: 0~70°C. Vdd = 2.2 ~ 3.6V. charge pump output short to VPP	6.4	6.7	7.0	V

LCD power (VLCD) is source from external power via VLCD pad, not internal short to VDD. In LCD R-type driver, V3 and V2 bias voltage is sourced by internal resistor voltage division. LCD driver electric circuit builds in selective 33.3k, 100k, 200k, 400k of resistor for internal voltage-division. User can connect additional external resistance between VLCD / V3 / V2 for more driving current.

$R = 33.3K$
 $/ 100K$
 $/ 200K$
 $/ 400K$

VLCD
 V3
 V2

$0.1\mu F$
 $0.1\mu F$
 $0.1\mu F$

LCDTYPE, LCDENB

VDD or VLED

VLCD

V3

V2

LCDBIAS=1
(close)

LCDTYPE, LCDENB

R

R

R

0.1uF

0.1uF

VDD or VLED

R= 33.3K
/ 100k
/ 200k
/ 400k

*** Note: In R-Type LCD driver mode, VLCD pin must input voltage power for LCD circuit operation.**

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 31	1FH.0	1FH.1	1FH.2	1FH.3	-	-	-	-

➤ **Example: Enable R-Type 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.
Call	Set_LCD_RAM	; Set LCD RAM to display LCD panel

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	-	-	ROMADR12	ROMADR11	ROMADR10	ROMADR9	ROMADR8
Read/Write	R	-	-	R/W	R/W	R/W	R/W	R/W
After reset	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[12:0] : ISP ROM Programming Address.

ROM Address which will be Programmed

10.3 ROMDAH/ROMDAL 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 30us.

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[12:0]

- * **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
        @B0BTS0_ FVPPCHK        ;Set VPP as VDD voltage.
        JMP      $-1            ;Check VPP Voltage is VDD or not
                                   ;If VPP still reach 6.5V, Keep waiting.

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

        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"
MOV          A, #3              ; Set ISP Program Max. Time 30us
B0MOV        ROMCNT, A

```

;----- 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
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

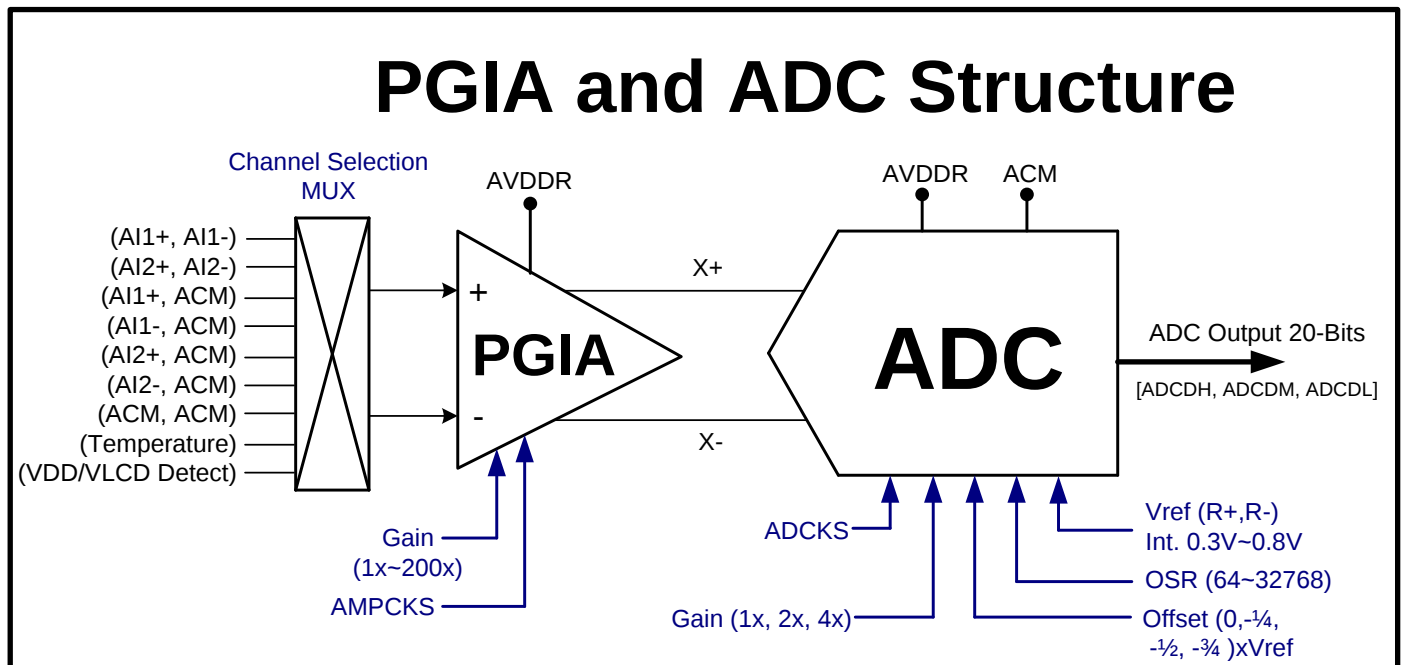
The SN8P2949 has a built-in Voltage Regulator to support a stable voltage 2.4V from pin AVDDR and 1.5V/2.0V from pin AVE+ with maximum 5mA current driving capacity. The AVDDR provides stable voltage for internal circuits (PGIA, ADC) and external sensor (load cell or thermistor). The SN8P2949 series also integrate $\Delta \Sigma$ Analog-to-Digital Converters (ADC) which output 20-bit with 18-bit accuracy. In fast ADC conversion mode, ADC conversion rate can up to 3.9kHz with 12.4 bit resolution (Gain=1, Vref=0.8V). The ADC has internal Gain option with selective range of x1, x2 and x4. The PGIA provides 2 types of input channel modes: (1) Two fully differential inputs [AI1+, AI1-] and [AI2+, AI2-].

(2) four single-ended inputs. This ADC is optimized for measuring low-level unipolar or bipolar signals in weight scale and medical applications. A very low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selectable gains of 1x, 12.5x, 50x, 100x and 200x in the ADC to accommodate these applications.

11.2 ANALOG INPUT

Following diagram illustrates a block diagram of the PGIA and ADC module. The front end consists of a multiplexer for input channel selection, a PGIA (Programmable Gain Instrumentation Amplifier), and the $\Delta \Sigma$ 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 AMPM1 register.



Block Diagram of ADC module

11.3 Voltage Regulator

SN8P2949 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 VREG 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	AVDDRSEL	AVDDRENB	-
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
After Reset	0	1	0	1	0	1	0	-

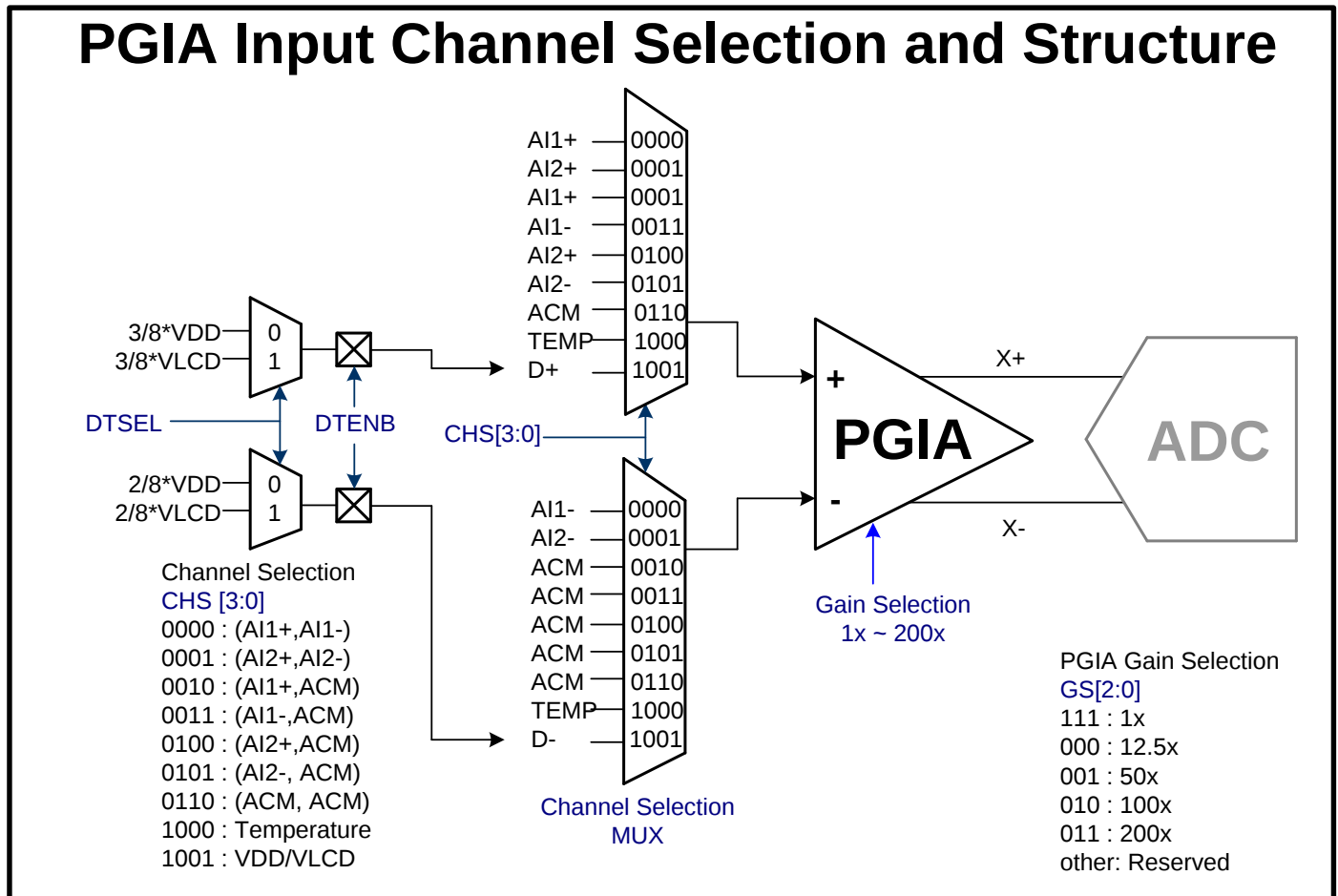
- Bit1: **AVDDRENB**: Regulator (AVDDR) voltage enable control bit.
0 = Disable AVDDR regulator output voltage.
1 = Enable AVDDR regulator output voltage
- Bit2: **AVDDRSEL**: AVDDR voltage selection control bit.
0 = Reserved.
1 = AVDDR output 2.4V
- Bit3: **AVENB**: AVE+ voltage output control bit.
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 voltage
1 = Enable Analog Common Mode voltage
- Bit6: **ACMSEL**: ACM voltage selection bit.
0 = Reserved
1 = Analog Common-mode voltage ACM output 1.0 V
- 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 (FBGENB), before following function accessing: (Reference AMPM1 and AMPM2 register for detail information)**
(1) Regulators of AVDDR, AVE+ and ACM. (2).PGIA Function.
(3) Low Battery Detection Function.
- * **Note 2: PGIA can work in Normal, Slow or Green Mode, when high clock is still running (STPHX=0).**
- * **Note 3: Add 10ms delay time after enabling each regulators, AVDDR/AVE/ACM, to avoid VDD drop in CR2032 battery application.**

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
V _{AVDDR}	Regulator output voltage AVDDR	AVDDR=2.4V, @25°C, Vdd = 2.6~3.6V.	2.25	2.4	2.55	V
		AVDDR=2.4V, @25°C, Vdd < 2.45V	Vdd – 0.05V			
V _{AVE+}	Regulator output voltage AVE+	VACM = 2.0V, @25°C, Vdd = 2.4 ~ 3.6V.	1.85	2.0	2.15	
		VACM = 1.5V, @25°C, Vdd = 2.4 ~ 3.6V.	1.35	1.5	1.65	
V _{ACM}	Analog common voltage	VACM = 0.6V, @25°C, Vdd = 2.4 ~ 3.7V.	0.9	1.0	1.1	

11.4 PGIA -Programmable Gain Instrumentation Amplifier

SN8P2949 includes a low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selection gains of 1x, 12.5x, 50x, 100x, and 200x controlled by register AMPM1. The PGIA also provides two types channel selection mode: (1) Two fully differential inputs (2) four 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	CHS3	CHS2	CHS1	CHS0	GS2	GS1	GS0	AMPENB
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	1	1	1	0

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

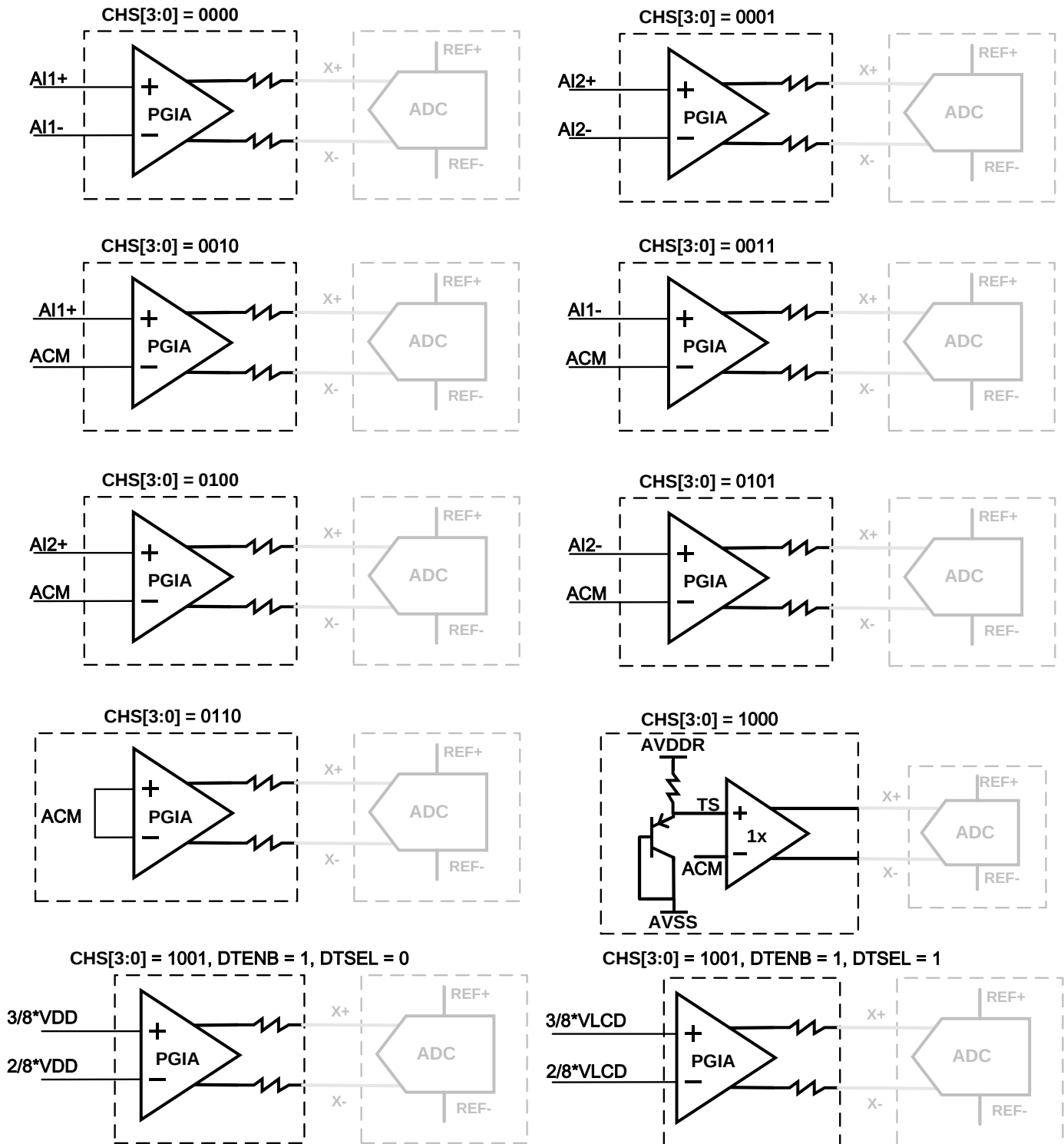
Bit[3:1]: **GS [2:0]**: PGIA Gain Selection control bit.

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

Bit[7:4]: **CHS [3:0]**: PGIA Channel Selection.

PGIA Channel Selection Table:

CHS [3:0]	Selected Channel	ADC Input	Input-Signal Type
0000	AI1+, AI1-	$V (AI1+, AI1-) \times \text{PGIA Gain}$	Differential
0001	AI2+, AI2-	$V (AI2+, AI2-) \times \text{PGIA Gain}$	Differential
0010	AI1+, ACM	$V (AI1+, ACM) \times \text{PGIA Gain}$	Single-ended
0011	AI1-, ACM	$V (AI1-, ACM) \times \text{PGIA Gain}$	Single-ended
0100	AI2+, ACM	$V (AI2+, ACM) \times \text{PGIA Gain}$	Single-ended
0101	AI2-, ACM	$V (AI2-, ACM) \times \text{PGIA Gain}$	Single-ended
0110	ACM, ACM	$V (ACM, ACM) \times \text{PGIA Gain}$	Input-Short
0111	Reserved	-	-
1000	Temperature sensor	$V (VTS, ACM) \times 1$	N/A
1001	Voltage Detection	$VDD (2/8VDD, 3/8VDD) \times \text{PGIA Gain}$ $VLCD (2/8VLDD, 3/8VLDD) \times \text{PGIA Gain}$	Differential



- * Note_1: $V(AI1+, AI1-) = (AI1+ \text{ voltage} - AI1- \text{ voltage})$.
- * Note_2: $V(AI2+, ACM) = (AI2+ \text{ voltage} - ACM \text{ voltage})$.
- * Note_3: The purpose of Input-Short mode is only for PGIA offset testing.
- * Note_4: When PGIA Gain set 1x (GS[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.
- * Note_5: When Input buffer enable (GX=1 or GR=1), the input absolutely voltage range of signal must within **0.4V~1.4V**.

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	INRENB	GX	GR	AMPCKS1	AMPCKS0	PCHPENB	DTENB	DTSEL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	1	0	0	1	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

Bit2: **PCHPENB**: PGIA Chopper Enable bit. **(Always set "1")**
 0 = Disable PGIA Chopper
 1 = Enable PGIA Chopper.

Bit[4:3]: **AMPCKS[1:0]**: PGIA chopper frequency selection. **(Always set AMPCKS[1:0] = 10)**

Bit5: **GR**: R+ R+ Unit Gain Buffer Function control 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-.)

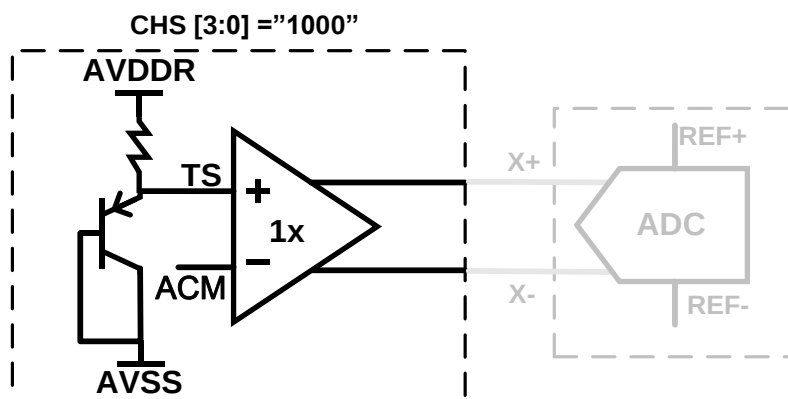
Bit6: **GX**: X+ X+ Unit Gain Buffer Function control bit.
 0 = Disable X+ X- UGB function. (When PGIA Gain set x12.5, x50, x100, or x200)
 1 = Enable X+ X- UGB function. (When PGIA Gain set x1)

Bit7: **INRENB**: **(Always set "1")**

- * **Note_1**: In general application, please set AMPCKS[1:0] = 10.
- * **Note_2**: Set INRENB = 1 for increasing EMS characteristic.
- * **Note_3**: When Input buffer enable (GX=1 or GR=1), the input absolutely voltage range of signal must within **0.4V~1.4V**.

11.5 Temperature Sensor (TS)

In applications, sensor characteristic might change in different temperature also. To get the temperature information, SN8P2949 build in a temperature sensor (TS) for temperature measurement. Select the respective PGIA channel to access the Temperature Sensor ADC output.



- * **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 ACM.
- * **Note 3:** The Temperature Sensor was just a reference data not real air temperature. For precision application, please use external thermistor sensor.

In 25°C, V(TS) will be about 1V typically, and if temperature rise 10°C, V(TS) will increase about 32mV ($V_{TS}=1.032V$), if temperature drop 10°C, V(TS) will decrease about 32mV ($V_{TS}=0.968V$).

Example:

Temperature	V(TS)	(X+) – (X-)	ADC Vref	ADC output (16-Bit)
15°C	0.968V	-0.032V	0.6V	-1747
25°C	1.000V	0V	0.6V	0
35°C	1.032V	0.032V	0.6V	1747

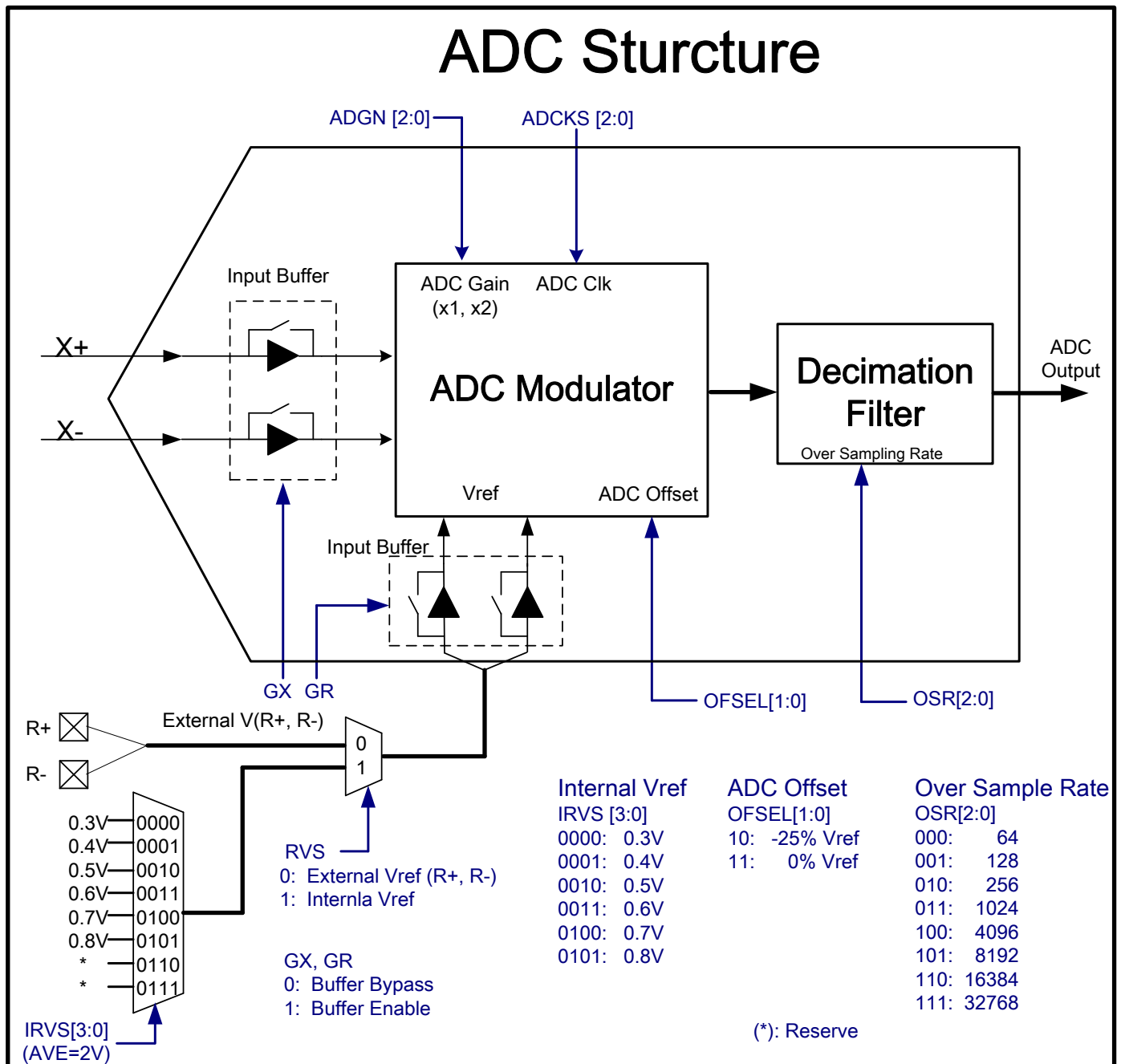
By ADC output of V(TS), can get temperature information and compensation the system.

- * **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:** 3.23mV/°C was typical temperature parameter only sensor, every single chip was different to each other.

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
TR	Temperature Sensor Range	AVDDR=2.4V, VDD = 3V.	-10	-	+70	°C
TS	Temperature Sensor Sensitivity	AVDDR=2.4V, VDD = 3V.	3.52	3.2	2.88	mV/°C
ETS	Temperature Sensor Accuracy	One Temperature point 25°C Calibration.	-10	-	+10	%
		Two Temperature points Calibration.	-1	-	+1	%

11.6 20-Bit Analog to Digital Converter (ADC)

The SN8P2949 integrated a 20-bit $\Delta\Sigma$ Analog-to-Digital Converters (ADC) with decimation filters can be set for variable throughputs range from 1Hz 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, x2 and x4 for additional signal amplification expect PGIA.



11.6.1 Analog Inputs and Voltage Operation Range

There are six analog inputs for ADC and PGIA operation, including pins of AI1+/AI1-, AI2+/AI2-, R+, and R-. The analog inputs of PGIA, AI1+/AI1-, AI2+/AI2-, 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 0.4V 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[3:0]** 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[3:0]** 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, x2 and x4 for additional signal amplification expect PGIA. The ADC Gain setting is controlled by **ADGN [1:0]** 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 **ADCM2**. The following shows ADC output code calculation:

ADC output code (differential mode) :

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

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

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

Voffset: 0, -1/4, -1/2 or -3/4 x Vref

PGIA: 1x ~ 200x

ADC_Gain: 1x, 2x and 4x

Vref Source: Internal Vref or External Vref

Vref Range: 0.3V ~ 0.8V

- * **Note_1:** When ADC Offset Function set -1/4*Vref, the ADC ENOB will drop 0.1~0.3bit, compare with ADC no offset function in Gain=200*1
- * **Note_2:** When ADC Offset Function set -1/2*Vref, the ADC ENOB will drop 0.3~0.5 bit, compare with ADC no offset function in Gain=200*1
- * **Note_3:** When ADC Offset Function set -3/4*Vref, the ADC ENOB will drop 0.6~0.8 bit, compare with ADC no offset function in Gain=200*1

11.6.5 ADC Output Word Rate

Delta-Sigma ADC provides variable output word rate (WR) from 0.95 Hz up to 3.9 kHz, which output word rate is decided by setting bits of **ADCKS[2:0]** 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[2:0]	OSR [2:0]	ADC Clock	WR	ADCKS[2:0]	OSR [2:0]	ADC Clock	WR
000	000	250KHz	3.9 kHz	001	000	125kHz	1.95KHz
000	001		1.95 kHz	001	001		976Hz
000	010		976 Hz	001	010		488Hz
000	011		244 Hz	001	011		122Hz
000	100		61 Hz	001	100		30.5Hz
000	101		30.5 Hz	001	101		15.2Hz
000	110		15.2 Hz	001	110		7.6Hz
000	111		7.6 Hz	001	111		3.8Hz
ADCKS[2:0]	OSR [2:0]	ADC Clock	WR	ADCKS[2:0]	OSR [2:0]	ADC Clock	WR
010	000	62.5kHz	976Hz	011	000	31.25kHz	488Hz
010	001		488Hz	011	001		244Hz
010	010		244Hz	011	010		122Hz
010	011		61Hz	011	011		30.5Hz
010	100		15.2Hz	011	100		7.6Hz
010	101		7.6Hz	011	101		3.8Hz
010	110		3.8Hz	011	110		1.9Hz
010	111		1.9Hz	011	111		0.95Hz

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	IRVS3	IRVS2	IRVS1	IRVS0	ADGN1	ADGN0	ADCENB
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	1	0	1	0	1	0	0	0

Bit0: **ADCENB**: ADC function control bit:
 0 = Disable 20-bit ADC,
 1 = Enable 20-bit ADC

Bit[2:1]: **ADGN[1:0]**: ADC Gain Selection

ADGN[1:0]	ADC Gain
00	x1
01	x2
10	x4
11	Reserved

Bit[6:3]: **IRVS[3:0]**: ADC Internal Reference Voltage Selection.

IRVS[3:0]	Vref source	
	AVE 1.5V	AVE 2.0V
0000	0.225V	0.3V
0001	0.300V	0.4V
0010	0.375V	0.5V
0011	0.450V	0.6V
0100	0.525V	0.7V
0101	0.600V	0.8V
0110~1111	Reserved	Reserved

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 with AVE or AVDDR.

- * **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	-	OSR2	OSR1	OSR0	-	OFSEL1	OFSEL0	DRDY
R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
After Reset	-	1	1	1	-	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[3:1] **OFSEL[1:0]:** ADC Offset selection.

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

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

11.6.8 ADCM3- ADC Mode3 Register

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCM3	-	-	-	ACHPENB	ADCKINV	ADCKS2	ADCKS1	ADCKS0
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	-	1	0	0	0	0

Bit[2:0]: **ADCKS[2:0]:** ADC Clock selection bit.

ADCKS[2:0]	ADC Clock
000	250kHz
001	125KHz
010	62.5KHz
011	31.25KHz
1xx	Reserved

Bit[3]: **ADCKINV**: ADC Clock Inverse control bit. (Please Always set "0")

Bit[4]: **ACHPENB**: ADC Chopper Control bit. (Please Always set "1")

- * *Note 1 ADC Output Word Rate (WR) = ADC Clock / OSR.*
- * *Note 2: Adjust ADC clock (ADCKS[2:0]) 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 data at the 3rd data after ADC enable. The 3rd, 4th, 5th ... are stable data after 1/WR later of each.*

11.6.9 ADC Data Register

097H	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

098H	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

099H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADC DL	ADCB07	ADCB06	ADCB05	ADCB04	ADCB03	ADCB02	ADCB01	ADCB00
R/W	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	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.

ADC DL [7:0]: Output low byte data of ADC conversion word.

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

- * **Note 1:** ADCDH [7:0], ADCDM [7:0] and ADCDL [7:0] are read only registers.
- * **Note 2:** For 16-Bit ADC resolution, please use registers of ADCDH and ADCDM (ADCB23~ADCB08).
For 18-Bit ADC resolution, please use registers of ADCDH, ADCDM and ADCDL.
(ADCB23~ADCB06).
For 20-Bit ADC resolution, please use registers of ADCDH, ADCDM and ADCDL.
(ADCB23~ADCB04).
For 24-Bit ADC resolution, please use registers of ADCDH, ADCDM and ADCDL.
(ADCB23~ADCB00)
- * **Note 3:** The ADC conversion data is combined with ADCDH, ADCDM, ADCDL in 2's compliment with sign bit numerical format, and Bit ADCB23 is the sign bit of ADC data.
ADCB23=0 means data is Positive value, ADCB23=1 means data is Negative value.
- * **Note 4:** The Positive Full-Scale-Output value of ADC conversion is 0x7FFFF.
- * **Note 5:** The Negative Full-Scale-Output value of ADC conversion is 0x80000H.
- * **Note 6:** Because of the ADC design limitation, the ADC Linear range is +131071 ~ -131072 (18-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 SN8P2949 ADC with different output word rate and gain settings. These numbers are typical and generated using a differential input-short condition, and with 1024-data of measurement.

2949 ADC Performance ENOB (Noise Free Bit) vs. Output WR and Gain								
Gain	ADC Output Word Rate							
	3.9 kHz	2.0 kHz	977 Hz	244 Hz	61.0 Hz	30.5 Hz	15.3 Hz	7.6 Hz
1 x 1*	12.4	14.1	14.9	16.1	17.1	17.6	18.1	18.5
1 x 2*	12.3	13.4	14.3	15.4	16.5	17.0	17.6	18.2
12.5 x 1	12.3	13.4	14.2	15.3	16.2	16.8	17.3	17.6
50 x 1	12.1	13.2	13.6	14.8	15.7	16.1	16.7	17.1
100 x 1	11.6	12.5	13.0	14.1	15.0	15.4	16.0	16.4
100 x 2	10.9	11.3	12.0	13.1	14.1	14.6	15.2	15.5
200 x 1	11.0	11.6	12.2	13.2	14.3	14.8	15.2	15.6

*: Buffer off (GX=0, GR=0)

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

- (1). Noise Free Resolution = $\text{Log}_2 (\text{Full Scale Range} / \text{Peak-Peak Noise})$
 where Full Scale Range = $2 \times V_{\text{ref}} / \text{Gain}$ (ex. $V_{\text{ref}}=0.84\text{V}$, Gain=200x)
- (2). Effective Resolution = $\text{Log}_2 (\text{Full Scale Range} / \text{RMS_Noise})$
- (3). RMS Noise = $\sigma \times \text{LSB_Resolution}$
 where $\text{LSB_Resolution} = \text{Full Scale Range} / 2^{\text{Bit}}$, Bit=20
 σ = standard deviation of 1024 ADC output data.
- (4). Peak-Peak Noise = $6.6 \times \text{RMS Noise}$, or code variation range $\times \text{LSB_Resolution}$
 where Code variation range = ADC counts max-min of 1024 data.

2949 ADC Peak to Peak Noise(uV) vs. Output WR and Gain

Gain	ADC Output Word Rate							
	3.9 kHz	2.0 kHz	977 Hz	244 Hz	61.0 Hz	30.5 Hz	15.3 Hz	7.6 Hz
1 x 1	310.8	95.67	54.95	23.92	11.96	8.456	5.980	4.532
1 x 2	166.6	77.71	41.64	19.43	9.063	6.409	4.228	2.790
12.5 x 1	26.65	12.43	7.141	3.332	1.785	1.178	0.833	0.677
50 x 1	7.654	3.571	2.706	1.178	0.631	0.478	0.316	0.239
100 x 1	5.412	2.900	2.051	0.957	0.513	0.389	0.256	0.194
100 x 2	4.396	3.332	2.051	0.957	0.478	0.338	0.223	0.181
200 x 1	4.102	2.706	1.785	0.893	0.416	0.294	0.223	0.169

2949 ADC RMS Noise (uV) with Output WR and Gain

Gain	ADC Output Word Rate							
	3.9 kHz	2.0 kHz	977 Hz	244 Hz	61.0 Hz	30.5 Hz	15.3 Hz	7.6 Hz
1 x 1	47.10	14.50	8.326	3.624	1.812	1.281	0.906	0.687
1 x 2	25.24	11.77	6.310	2.944	1.373	0.971	0.641	0.423
12.5 x 1	4.038	1.884	1.082	0.505	0.271	0.178	0.126	0.103
50 x 1	1.160	0.541	0.410	0.178	0.096	0.072	0.048	0.036
100 x 1	0.820	0.439	0.311	0.145	0.078	0.059	0.039	0.029
100 x 2	0.666	0.505	0.311	0.145	0.072	0.051	0.034	0.027
200 x 1	0.621	0.410	0.271	0.135	0.063	0.045	0.034	0.026

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	FAVDDRSEL	; Set AVDDR output 2.4V
	B0BSET	FAVDDRENB	; Enable AVDDR Voltage
@PGIA_Init:	MOV	A, #00000110B	
	B0MOV	AMPM1, A	; PGIA channel (AI1+, AI1-) and PGIA Gain x 200
	MOV	A, #00010100B	
	B0MOV	AMPM2, A	; Set unit Gain buffer off and PGIA chopper 31.25KHz
	B0BSET	FAMPENB	; Enable PGIA function
			; V (X+, X-) Output = V (AI1+, AI1-) x 200
@ADC_Init:	MOV	A, #10101000B	
	B0MOV	ADCM1, A	; ADC Reference voltage = internal 0.8V, ADC_Gain = 1x.
	MOV	A, #01110110B	
	B0MOV	ADCM2, A	; Set OSR=32768, offset=0V.
	MOV	A, #00010000B	
	B0MOV	ADCM3, A	; Set ADC clock=250kHz
	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	FAVDDRSEL	; Set AVDDR output 2.4V
	B0BSET	FAVDDRENB	; Enable AVDDR Voltage
@VDD_Detection:			
@PGIA_Init:	MOV	A, #10011110B	
	B0MOV	AMPM1, A	; PGIA Voltage Detection and PGIA Gain x 1
	MOV	A, #00010110B	; 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 = 1/8 VDD x 1
@ADC_Init:	MOV	A, #10101000B	
	B0MOV	ADCM1, A	; ADC Reference voltage = internal 0.8V, ADC_Gain = 1x.
	MOV	A, #01110110B	
	B0MOV	ADCM2, A	; Set OSR=32768, offset=0V.
	MOV	A, #00010000B	
	B0MOV	ADCM3, A	; Set ADC clock=250kHz
	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, #10011110B	
	B0MOV	AMPM1, A	; PGIA Voltage Detection and PGIA Gain x 1
	MOV	A, #00010111B	; 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 = 1/8 VLCD x 1
@ADC_Init:	MOV	A, #10101000B	
	B0MOV	ADCM1, A	; ADC Reference voltage = internal 0.8V, ADC_Gain = 1x.
	MOV	A, #01110110B	
	B0MOV	ADCM2, A	; Set OSR=32768, offset=0V.
	MOV	A, #00010000B	
	B0MOV	ADCM3, A	; Set ADC clock=250kHz
	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    FAVDDRSEL   ; Set AVDDR output 2.4V
                B0BSET    FAVDDRENB  ; Enable AVDDR Voltage

@VDD_Detection:
@PGIA_Init:     MOV        A, #00001110B
                B0MOV      AMPM1, A    ; PGIA differential channel (AI1+, AI1-) and PGIA Gain x 1
                MOV        A, #00010100B
                B0MOV      AMPM2, A    ; Set unit Gain buffer off and PGIA chopper 31.25KHz
                B0BSET     FAMPENB     ; Enable PGIA function
                                           ; V (X+, X-) Output = V (AI1+, AI1-) x 1

@ADC_Init:      MOV        A, #10101000B
                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.
                MOV        A, #00010000B
                B0MOV      ADCM3, A    ; Set ADC clock=250kHz
                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    ; Move ADC conversion High byte to Data Buffer
                B0MOV      Data_H_Buf, A
                B0MOV      A, ADCDM    ; Move ADC conversion Medium byte to Data Buffer
                B0MOV      Data_M_Buf, A

```

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 OSR=256. WR=976Hz
MOV              A, #00010000B
B0MOV            ADCM3                ; Set ADC clock=250kHz
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.)
B0BCLR           FADCENB              ; Disable ADC function.
B0BCLR           FAMPENB              ; Disable PGIA function.
B0BCLR           FVLEDENB            ; Disable VLED function.
B0BCLR           FCPRENB              ; Disable LED charge pump
B0BCLR           FLBTENB             ; Disable Low battery detect function.
B0BCLR           FDTENB              ; Disable VDD/VLCD detect function.
B0BCLR           FAVDDRENB           ; Disable AVDDR.
B0BCLR           FAVENB              ; Disable AVE.
B0BCLR           FACMENB             ; Disable ACM.
B0BCLR           FLCDPENB            ; Disable C-Type LCD charge pump.
B0BCLR           FLCDENB             ; Disable LCD display.
B0BCLR           FBGRENB             ; Disable Band Gap Voltage.
B0BSET           FLCKMD              ; Into slow mode.
B0BSET           FSTPHX              ; Stop High clock (IHRC).
MOV              A, #11000000
B0MOV            T0M, A                ; T0 overflow and green mode wakeup
GreenMode        ; System into Green mode. (Macro)

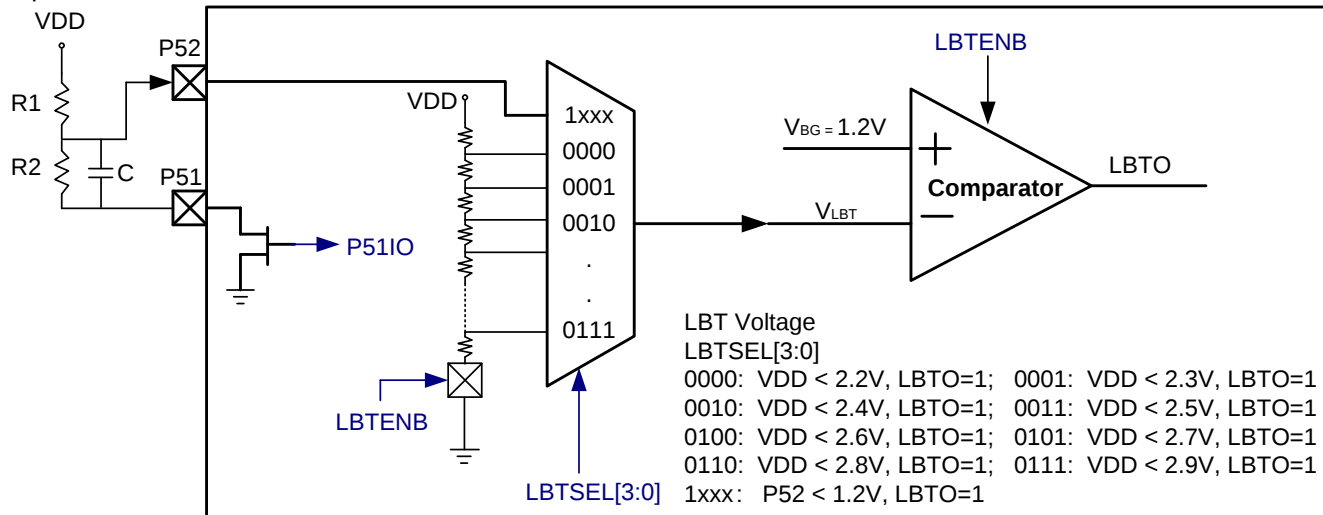
Sleep_Mode:     // Wakeup by P0 level change.
B0BCLR           FADCENB              ; Disable ADC function.
B0BCLR           FAMPENB              ; Disable PGIA function.
B0BCLR           FVLEDENB            ; Disable VLED function.
B0BCLR           FCPRENB              ; Disable LED charge pump
B0BCLR           FLBTENB             ; Disable Low battery detect function.
B0BCLR           FDTENB              ; Disable VDD/VLCD detect function.
B0BCLR           FAVDDRENB           ; Disable AVDDR.
B0BCLR           FAVENB              ; Disable AVE.
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

SN8P2949 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 eight internal levels, every 0.1V one level from 2.2V~2.9V. The Function can be set for low battery detect, or via divide VDD voltage and connect to P52. 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	-	P51IO	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTO	LBTENB
R/W	-	R/W	R/W	R/W	R/W	R/W	R	R/W
After Reset	-	0	0	0	0	0	0	0

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[5:2]: **LBTSEL[3:0]**: Low Battery Detect threshold voltage selection bit.

LBTENB	LBTSEL [3:0]	LBTO = 1	Note
0	-	-	LBT Function disable
1	0000	VDD < 2.2V	Internal Input
1	0001	VDD < 2.3V	Internal Input
1	0010	VDD < 2.4V	Internal Input
1	0011	VDD < 2.5V	Internal Input
1	0100	VDD < 2.6V	Internal Input
1	0101	VDD < 2.7V	Internal Input
1	0110	VDD < 2.8V	Internal Input
1	0111	VDD < 2.9V	Internal Input
1	1xxx	P52 < 1.2V, LBTO=1	External Input

Bit6: **P51IO**: Port 5.1 Input/LBT function control bit.
0 = Set P51 as Input Port,
1 = Set P51 as LBT function, P51 connect to ground internally.

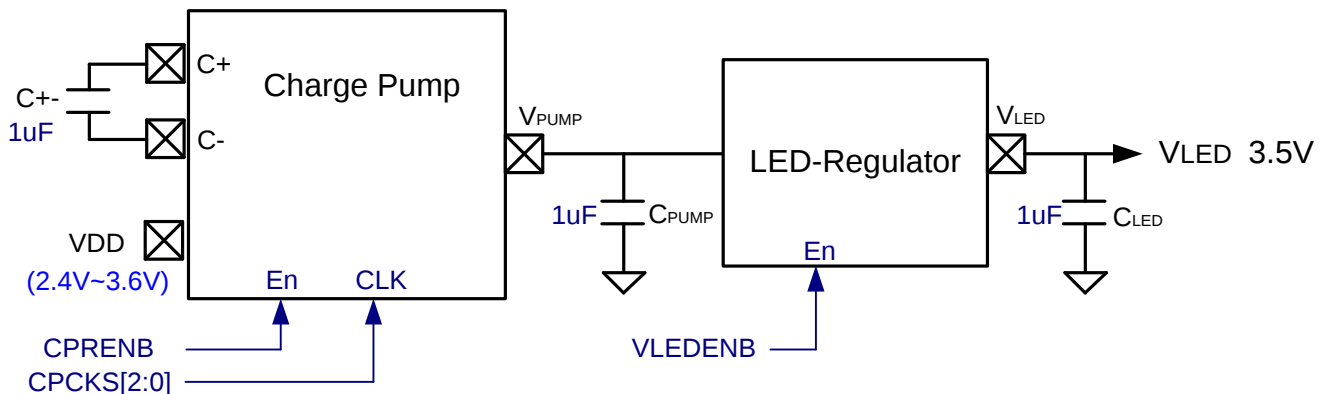
External Input (P52) LBT functions:

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 row every certain period, ex. 20 ms or more to make sure the Low Battery signal is stable.
- * **Note_2:** LBT external input P52 and P51IO function is available in ICE emulation.
- * **Note_3:** IO input voltage must keep lower than VDD.

11.8 Charge Pump and LED Regulator

SN8P2949 provided fixed Regulator output voltage, which is for LED driving purpose. LED-Regulator output voltage is 3.5V from VDD 3.6V to 2.4V. Its maximum driving current is over 8mA when DVDD is 2.4V. The diagram below shows the charge pump and LED Regular system.



- Charge Pump output voltage $V_{pump} = 2 \times VDD$, VDD range from 2.4V~3.6V
- Capacitors requirement: $C_{pump} = C_{+-} = C_{LED} = 1\mu F$
- LED-Regulator output voltage 3.5V.
- LED-Regulator maximum driving current is 8mA at VDD = 2.4V.

CPM register initial value = xxx0 0000

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LEDCPM	-	-	-	VLEDENB	CPCKS2	CPCKS1	CPCKS0	CPRENB
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	-	0	0	0	0	0

Bit0 **CPRENB:** LED Charge Pump Enable bit for VLED power source.
0 = Disable charge pump.
1 = Enable charge pump.

Bit[3:1] **CPCKS[2:0]:** Charge Pump Clock Selection Bits.

CPCKS[2:0]	Charge pump clock
0xx	Reserved
100	15.625 KHz
101	31.25 KHz
110	62.5 KHz
111	125 KHz

Bit3 **LEDENB:** VLED Output 3.5V Enable Bit.
0 = Disable VLCD output.
1 = Enable charge pump. Only available in normal mode

- * **Note 1: VLED Start recommendation : Enable CPRENB about 5ms before enable LEDENB**
- * **Note 2: Charge Pump LED-Regular function is available in normal mode and slow mode (STPHX = 0).**
- * **Note 3: Current consumption from VLED will time 2 when charge pump was enabled.**
- * **Note 4: When CPRENB = 0, Vpump = VDD.**
- * **Note 5: If VLED output current is over 8mA, it is strongly recommended to select the 125kHz**

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
VLED	VLED Output Voltage	VDD = 3V.	3.3	3.5	3.7	V
ILED	VLED Output Current Capacity T	VDD = 2.4V, Pump Clock = 125KHz.	8	-	-	mA

11.9 Analog Setting and Application

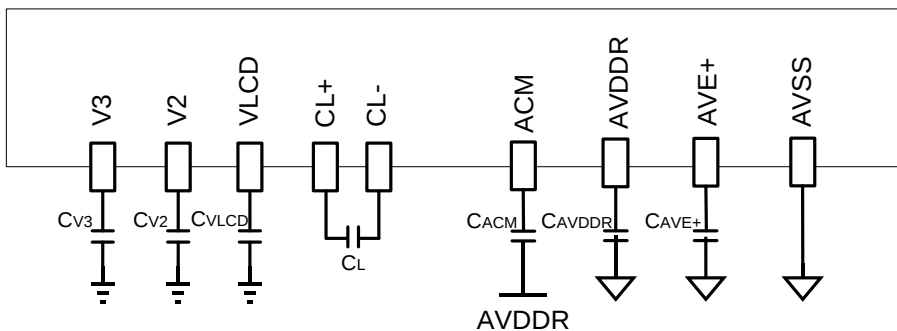
The most applications of SN8P2949 were for DC measurement ex. Weight scale, Pressure measure. Following table indicate different applications setting which MCU power source came from CR2032 battery, AA/AAA dry battery or external Regulator.

Capacitor Table:

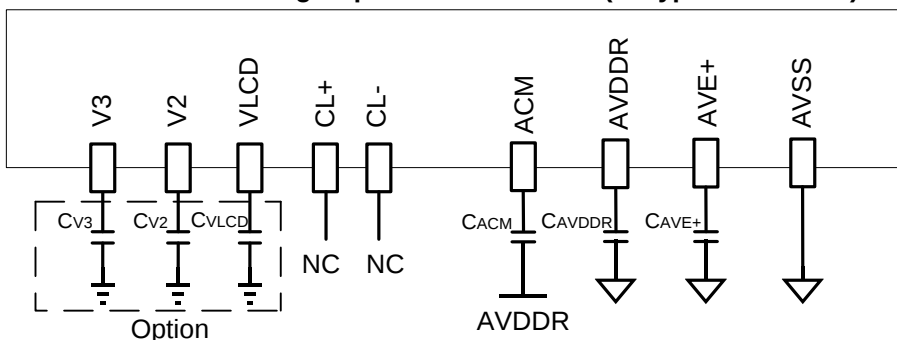
Power type	AI+	AI-	R+/ R-	ACM	AVDDR	AVE+	CL+/CL-	C+/C-	VPUMP	VLED	AVDD	DVDD
	C _{AI+}	C _{AI-}	C _R	C _{ACM}	C _{AVDDR}	C _{AVE+}	C _L	C _C	C _{pump}	C _{LED}	C _{AVDD}	C _{DVDD}
CR2032 (2.4~3V)	0.01uF	0.01uF	0.1uF	0.1uF	0.47uF	0.47uF	0.1uF	1uF	1uF	1uF	10uF, 0.1uF	10uF, 0.1uF
AA/AAA Bat.(2.4~3V)	0.01uF	0.01uF	0.1uF	0.1uF	1uF	1uF	0.1uF	1uF	1uF	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)



13 INSTRUCTION SET TABLE

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

Note: 1. The "M" is memory including system registers and user defined memory.

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

3. If "M" is system registers (80h ~ FFh of bank 0) then "N" = 0, otherwise "N" = 1

14 Development Tools

14.1 Development Tool Version

14.1.1 ICE (In circuit emulation)

- **SN8ICE2K Plus II:** Full function emulates SN8P2949

- * **SN8ICE2K ICE emulation notice**
 - Operation voltage of ICE: 3.3V.
 - Recommend maximum emulation speed at 3.3V: 1 MIPS (e.g. $F_{cpu} = F_{hosc}/4$).
 - Use SN8P2949 EV-KIT to emulation Analog Function.
 - Note: S8ICE1K doesn't support SN8P2949 emulation.

14.1.2 OTP Writer

MP Pro Writer : ON/OFF line operation to support SN8P2949 mass production.

- * **Note:** MP111 Writer doesn't support SN8P2949 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 MP111 Writer doesn't support SN8P2949

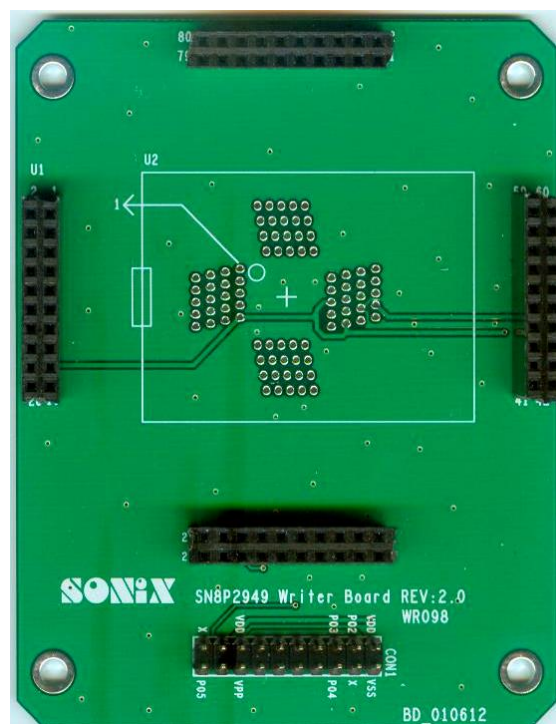
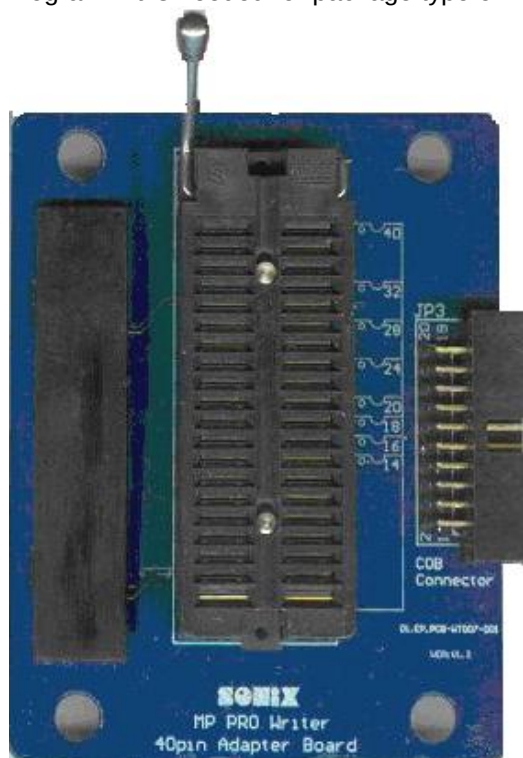
14.2 OTP Programming Pin to Transition Board Mapping

SN8P2949 COB Programming Pin Mapping:

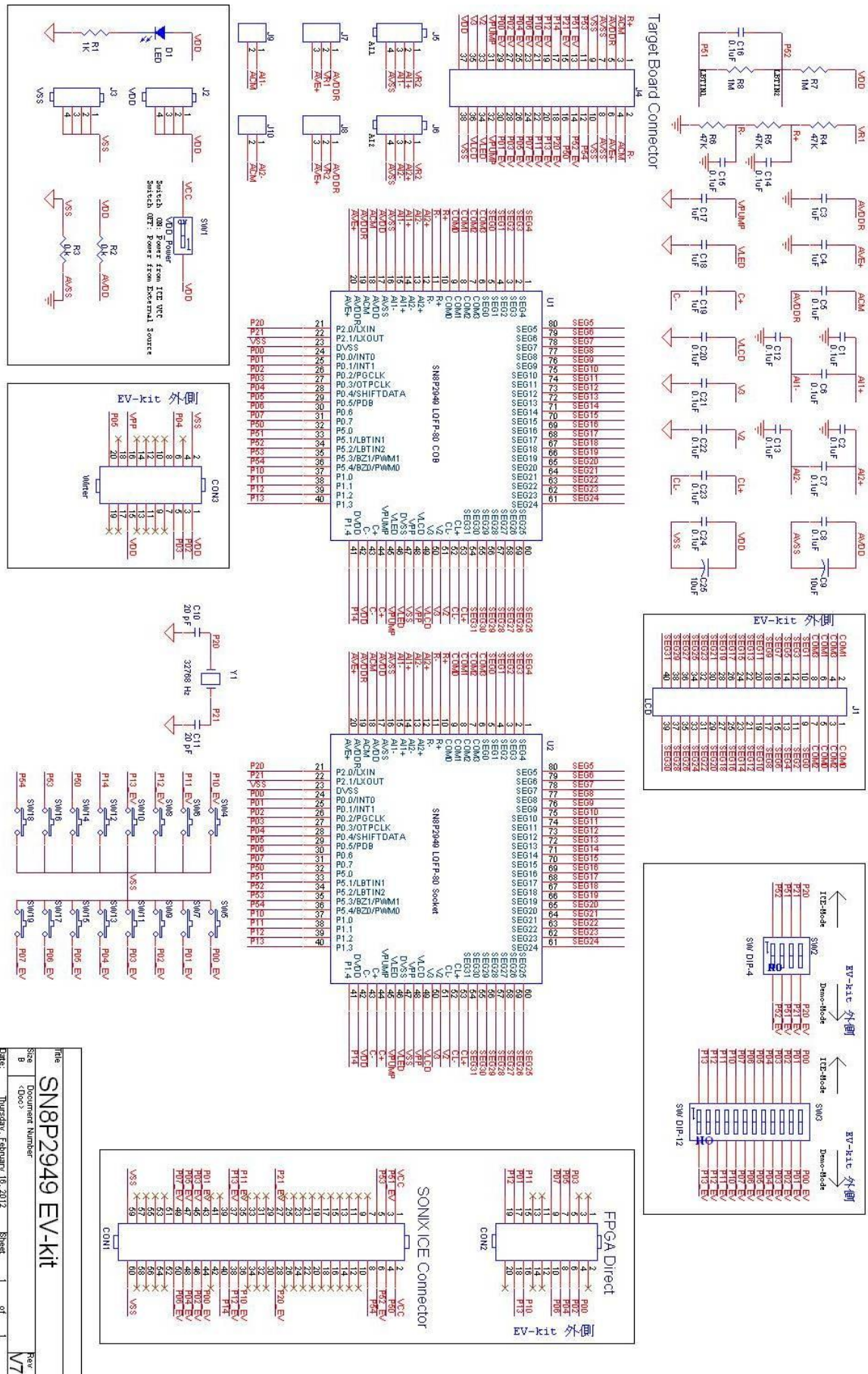
OTP Programming Pin of SN8P2949			
MP PRO Writer		SN8P2949	
JP3 of 40 PIN Adapter Board		Pin Assignment	
Number	Pin Name	Pad Name	LQFP80 PIN Number
1	VDD	DVDD/AVDD/ VLCD	42/17/49
2	GND	DVSS/AVSS	47/16
3	CLK / PGCLK	P0.2	26
4	CE	-	-
5	PGM / OTPCLK	P0.3	27
6	OE / ShiftData	P0.4	28
7	D1	-	-
8	D0	-	-
9	D3	-	-
10	D2	-	-
11	D5	-	-
12	D4	-	-
13	D7	-	-
14	D6	-	-
15	VDD	DVDD/AVDD / VLCD	42/17/49
16	VPP	VPP	48
17	HLS	-	-
18	RST	-	-
19	-	-	-
20	ALSB/PDB	P0.5	29

SN8P2949 Package Type Programming with 40 PIN adapter board and with writer board:

1. 40 PIN adapter board connect to MP PRO Writer.
2. Connect CON1 of SN8P2949 Writer Board to the adapter board (JP3).
3. Program Kit is needed for package type of LQFP-80 IC.



14.3 APPENDIX A: EV-KIT BOARD CIRCUIT



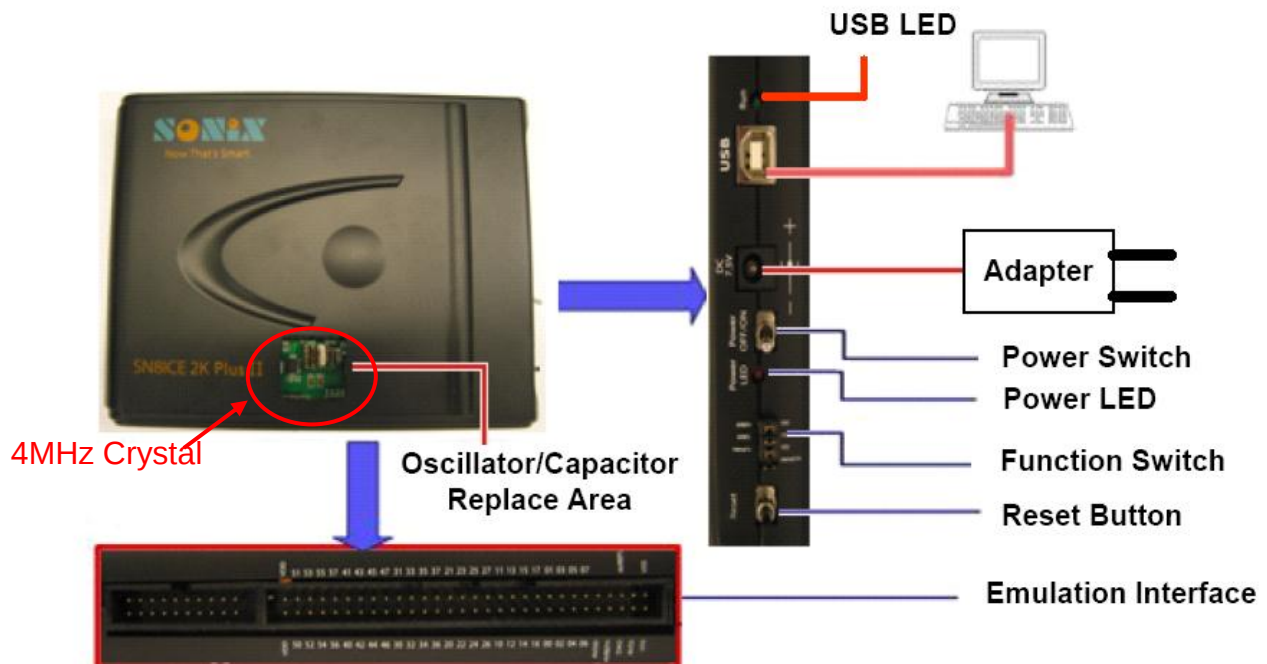
14.4 SN8P2949 Emulation

14.4.1 INTRODUCTION

Sonix provides a complete EV-KIT for SN8P2949 emulation, which includes an ICE SN8ICE2K_Plus_II, a SN8P2949 EV Board, Sonix Assembler and Compiler. 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 use 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 SN2949 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 SN8P2949 emulation.
3. Detail setting reference [SN8ICE2K Plus II User's Manual](#).



14.4.3 SN8P2949 EV-kit Board DESCRIPTION

Sonix provides SN8P2949 EV board for all functions emulation shown in FIG.1

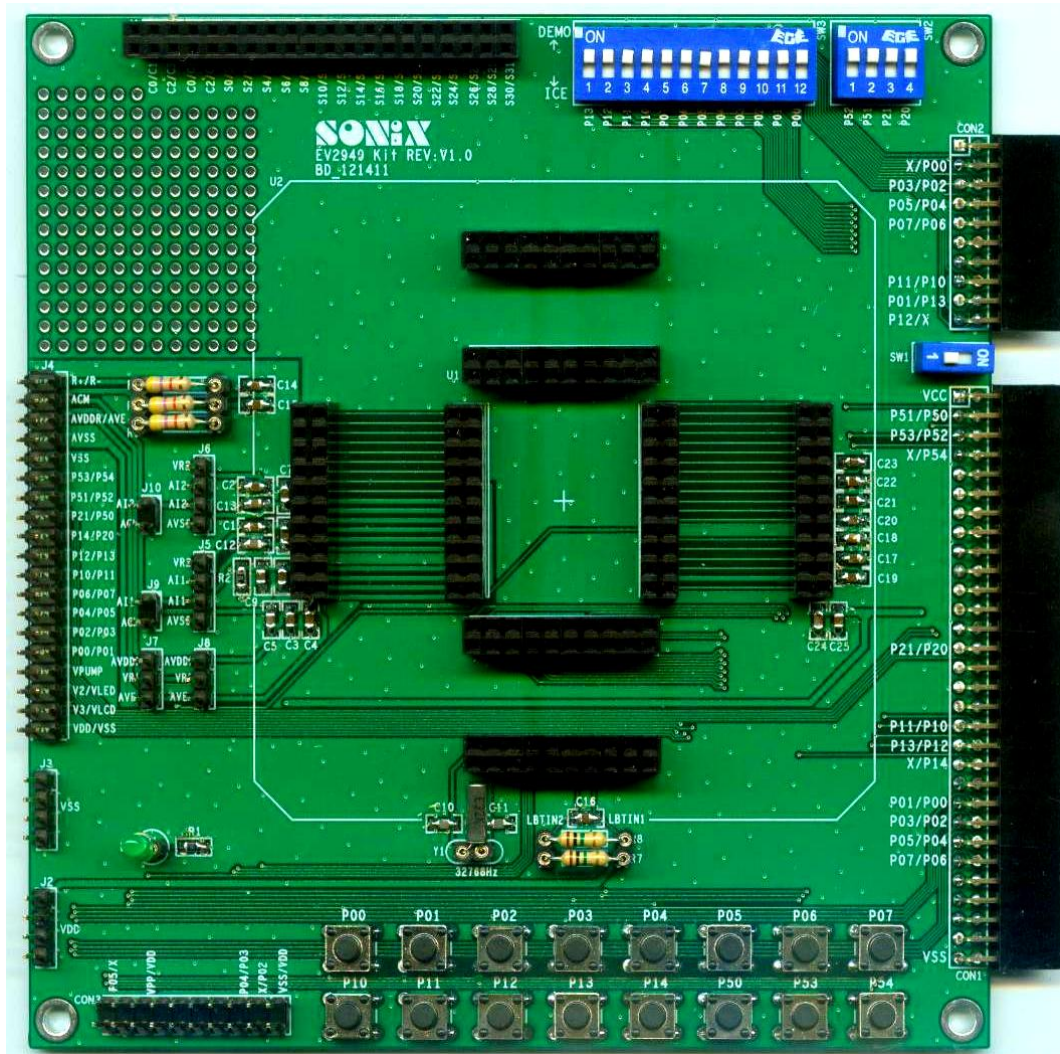


FIG.1 SN8P2949 EV-kit board

14.4.4 EV-kit BOARD SETTING

1. U1/U2 : SN8P2949 LQFP80 Socket
2. CON1/CON2 : connecting with the ICE PORT
3. SW1: 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 J2/J3 input.
4. SW2/SW3: Switch to "ICE" position when EV-Board connect to ICE. When separate ICE and EV board, switch to "DEMO" position , or else the relative IO port won't work.
5. J1 : LCD COM/SEG connect Pin.
6. J4: Target board connector.
7. J5/J6 : Analog Differential (AI1+, AI1-), (AI2+, AI2-) input Pin.
8. J7: Selection of external reference voltage V(R+, R-) power from AVDDDR or AVE+.
9. J8 : Selection of Load cell power from AVDDDR or AVE+.
10. J9/J10: Analog Single-End (AI1-/ACM), (AI2-/ACM) input Pin.

11. R4/R5/R6: Resistors for ADC external reference voltage.
12. D1 : Power indicator
13. R7/R8 : Resistors for low battery detect. R7 is connected with VDD and LBTIN2. R8 is connected with LBTIN2 and LBTIN1.

14.4.5 Notice for EV-kit Emulation

1. ICE VDD must switch to 3.3V. *VDD 5V is not available for SN8P2949 EV-kit*
2. SW2/SW3 must switch to "ICE" position.
3. Low Battery Detect (LBT) function can support both internal LBT emulation and external P52 Input.
4. The Level of internal LBT emulation function on the EV-kit Board ranges from 2.3V to 2.9V
5. Before using Internal LBT emulation, SW1 should be open, and the reference voltage is from J2
6. Node LBTIN2 is the external input of LBT. Node LBTIN1 can be I/O or internally grounded determined by setting flag P51IO. Before using external LBT emulation, SW1 should be also open.
7. *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})	$V_{SS} - 0.2V \sim V_{DD} + 0.2V$
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} , $V_{DD} = 3.0V$, $F_{HOSC} = IHRC(4MHz)$, $F_{CPU} = 1MHz$, ambient temperature is 25 °C 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	
VDD rise rate	VPOR	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	ILEKG	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
			Vdd=3V, Analog Parts ON	-	1.5	3	mA
			Vdd=3V, Analog Parts ON + LED driver	-	1.7	3.4	mA
		Slow Mode	Vdd=3V, IHRC On Analog Parts On, C-Type LCD On	-	1	2	mA
			Vdd=3V, IHRC OFF Analog Parts OFF, C-Type LCD On	-	150	300	uA
			Vdd=3V, HRC OFF Analog Parts OFF, R-LCD 400k On	-	10	20	uA
			Vdd=3V, IHRC OFF Analog Parts OFF, LCD OFF	-	5	10	uA
			Vdd=3V, IHRC OFF, IHRC_RTC Mode Analog Parts OFF, R-LCD 400k On	-	15	30	uA
			Vdd=3V, IHRC OFF, IHRC_RTC Mode Analog Parts OFF, LCD OFF	-	10	20	uA
		Green Mode	Vdd=3V, IHRC On Analog Parts On, C-Type LCD On	-	1	2	mA
			Vdd=3V, IHRC On Analog Parts OFF, C-Type LCD On	-	150	300	uA
			Vdd=3V, IHRC OFF, Analog Parts OFF, R-LCD 400k On	-	8	15	uA
			Vdd=3V, IHRC OFF, Analog Parts OFF, LCD OFF	-	3	6	uA
			Vdd=3V, IHRC OFF, IHRC_RTC Mode Analog Parts OFF, R-LCD 400k On	-	10	20	uA
			Vdd=3V, IHRC OFF, IHRC_RTC Mode Analog Parts OFF, LCD OFF	-	6	12	uA
		Sleep Mode	Vdd= 3V	-	1	2	uA
LVD detect level	VLVD	Internal POR detect level	1.7	1.9	2.2	uA	
Internal High Clock Freq.	FIHRC	Internal High RC Oscillator Frequency (Vdd = 2.4V ~ 3.6V, Temperature: 0℃~ 70℃)	3.6	4	4.4	MHz	

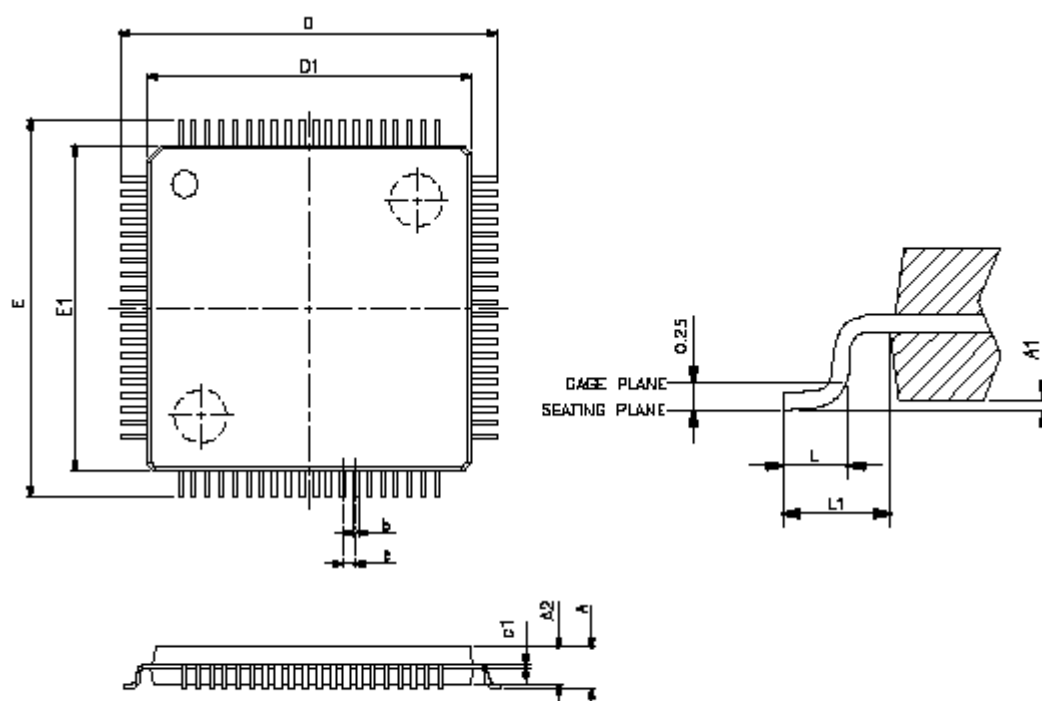
➤ **Note:** Analog Parts including Regulator, PGIA and ADC.

(All of voltages refer to $V_{DD}=3V$ $F_{H_{OSC}} = 1HRC$ (4MHz), ambient temperature is 25 °C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Analog to Digital Converter						
Operating current	I _{DD_ADC}	Run mode @ 2.4V	-	200	300	uA
Power down current	I _{PDN}	Stop mode @ 2.4V	-	0.1	-	μA
Conversion rate (Word Rate, WR)	F _{WR}	ADC Clock=250KHz, OSR=32768	-	7.6	-	Sps
		ADC Clock=250KHz, OSR=64	-	3.9	-	kSps
Reference Voltage Input absolutely Voltage	V _{R+}	GR=1, R+, R- Input absolutely Voltage	0.4	-	1.4	V
		GR=0, R+, R- Input absolutely Voltage	0.4	-	1.7	V
Reference Voltage Range	V _{ref}	ADC Reference voltage range. (R+) - (R-)	0.3	-	0.8	V
ADC Input signal absolutely Voltage	V _{AIN}	Input Buffer on (GX=1), AVDDR=2.4V	0.4	-	1.4	V
		Input Buffer off (GX=0) , AVDDR=2.4V	0.4	-	1.7	V
Integral non-linearity	INL	PGIA x 200, ADC Input Range ± 0.9xV _{ref}	-	-	0.01	%FSR
No missing code	NMC	ADC range ± 0.9 x V _{ref}	20	-	-	bit
ADC Noise free bits	NFB	Gain=1, V _{ref} :0.8V, OSR:32768, Input-short Buffer Off. (GR=GX=0)	-	18.4	-	bit
		Gain=200, V _{ref} :0.8V, OSR:32768, Input-short	-	16	-	bit
Input Refer Noise (Peak to Peak)	V _{Np-p}	Gain=1, WR=7.6Hz.	-	4.5	-	uV
		Gain=200, WR=7.6Hz.	-	0.17	-	uV
Unit Gain Buffer Current	I _{GX}	ADC signal input buffer operation current.	-	80	100	uA
	I _{GR}	ADC reference input buffer operation current.	-	80	100	uA
Unit Gain Buffer Input/Output Range	V _{GX}	Absolutely voltage range. (AVDDR 2.4V)	0.4	-	1.4	V
	V _{GR}	Min = 0 + 0.4V, Max = AVDDR-1V.	0.4	-	1.4	V
Temperature sensor Range	T _R	Temperature Sensor Operation Range	-10	-	+70	°C
Temperature Sensitivity	T _S	Temperature Sensor Sensitivity.	3.52	3.2	2.88	mV/°C
Temperature Sensor Accuracy	E _{Ts}	One Temperature point Calibration.	-10	-	+10	%
		Two Temperature points Calibration.	-1	-	+1	%
PGIA						
PGIA Current consumption	I _{DD_PGIA}	Run mode @ 2.4V	-	250	400	uA
Power down current	I _{PDN}	Stop mode @ 2.4V	-	-	0.1	uA
PGIA Gain Range	Gain	VDD = 2.4V, PGIA x 200	190	200	220	Gain
PGIA Input Range	V _{opin}	AI+, AI- signal input range. (AVDDR = 2.4V)	0.4	-	1.4	V
PGIA Output Range	V _{opout}	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/°C
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
Regulator output current capacity	I _{VA+}	AVDDR, AVE output current ability	-	-	5	mA
Quiescent current	I _{QI}	ACM + AVDDR + AVE	-	130	150	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, 400k bias resistor, No panel		3	5	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°C	-50	-	50	mV
VLED Driver (Pump + LDO)						
VLED Output Voltage	V _{LED}	VDD=3V,	3.3	3.5	3.7	V
VLED Output Current Capacity	I _{LED}	VDD=2.4V. Pump Clock = 125KHz	8			mA

16 PACKAGE INFORMATION

16.1 LQFP 80 PIN



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

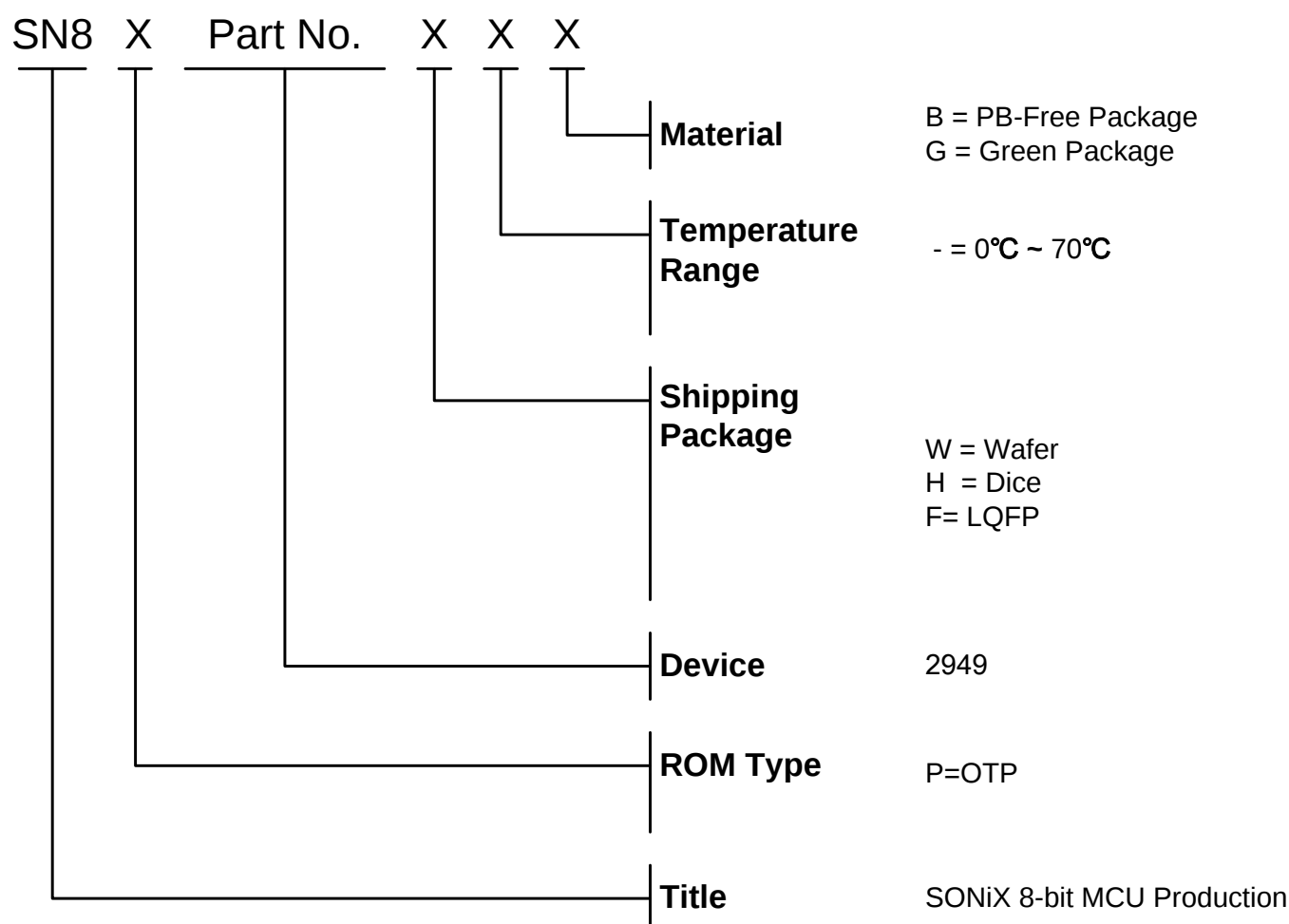
SYMBOLS	MIN.	MAX.
A	--	1.6
A1	0.05	0.15
A2	1.35	1.45
a1	0.09	0.16
D	12 BSC	
D1	10 BSC	
E	12 BSC	
E1	10 BSC	
a	0.4 BSC	
b	0.17	0.27
L	0.45	0.75
L1	1 REF	

17 Marking Definition

17.1 INTRODUCTION

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

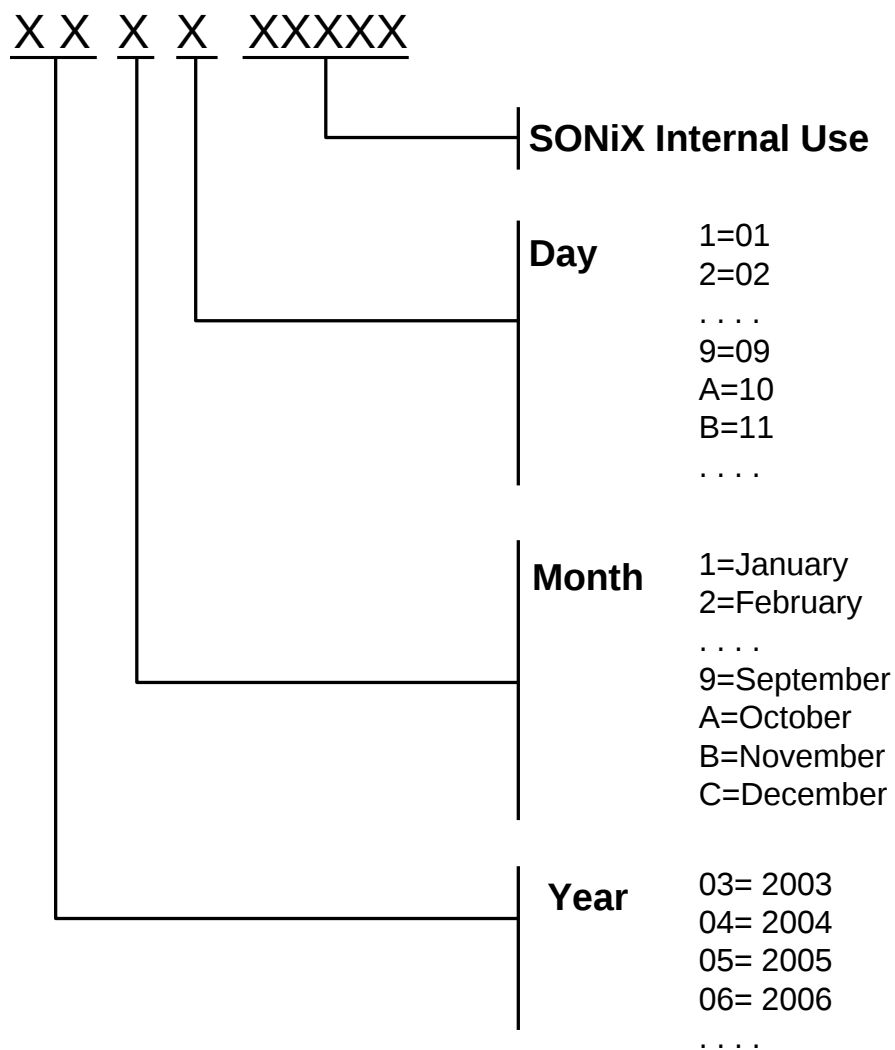
17.2 MARKING INDETIFICATION SYSTEM



17.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P2949FG	OTP	2949	LQFP80	0°C~70°C	Green Package

17.4 DATECODE SYSTEM



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

Main Office:

Address: 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