

SN8P1937

USER'S MANUAL

Specification Version 1.3

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.

AMENDMENT HISTORY

| Version | Date | Description |
|---------|-----------|---|
| VER 0.1 | SEP. 2009 | First issue Brief version. |
| VER 1.0 | Nov. 2009 | <ol style="list-style-type: none">1. Modified TC0 description2. Modified of LCD Timing Table and C-Type LCD description.3. Cancel ADCDLL and CMPTEST registers.4. Modified Capacitor Table.5. AVE+ 2V and ACM_0.6V functions canceled.6. Modified ADC full scale calculation.7. Modified X+ voltage limitation.8. Modified Supply Voltage 2.4~5.5V9. Modified X+X-, AI+AI- input/output range 0.4~1.05V |
| VER 1.1 | Dec. 2009 | <ol style="list-style-type: none">1. Update Slow Mode current spec.2. Add C-Type VLCD output voltage chart |
| VER 1.2 | Dec. 2009 | <ol style="list-style-type: none">1. Update AVDDR Spec.2. Update Idd10, Idd11 and LVD Spec. of electrical characteristic.3. Modified C-Type VLCD output with $\pm 0.2V$ accuracy. |
| VER 1.3 | May. 2010 | <ol style="list-style-type: none">1. Error correction: Int_16k_RC \rightarrow Int_32k_RC.2. Error correction: FALOAD description in chapter 8.3.73. Add ILRC Frequency chart and description.4. BGM must be set 1 for C-Type LCD application. |

Table of Content

| | |
|---|-----------|
| AMENDMENT HISTORY..... | 2 |
| 1 PRODUCT OVERVIEW..... | 8 |
| 1.1 SELECTION TABLE..... | 8 |
| 1.2 MIGRATION TABLE..... | 8 |
| 1.3 FEATURES..... | 9 |
| 1.4 SYSTEM BLOCK DIAGRAM..... | 10 |
| 1.5 PIN ASSIGNMENT..... | 11 |
| 1.6 PIN DESCRIPTIONS..... | 12 |
| 1.7 PIN CIRCUIT DIAGRAMS..... | 13 |
| 2 CENTRAL PROCESSOR UNIT (CPU)..... | 14 |
| 2.1 MEMORY MAP..... | 14 |
| 2.1.1 PROGRAM MEMORY (ROM)..... | 14 |
| 2.1.2 LOOK-UP TABLE DESCRIPTION..... | 18 |
| 2.1.3 CODE OPTION TABLE..... | 24 |
| 2.1.4 DATA MEMORY (RAM)..... | 25 |
| 2.1.5 SYSTEM REGISTER..... | 26 |
| 2.1.6 Y, Z REGISTERS..... | 34 |
| 2.1.7 R REGISTERS..... | 35 |
| 2.2 ADDRESSING MODE..... | 36 |
| 2.2.1 IMMEDIATE ADDRESSING MODE..... | 36 |
| 2.2.2 DIRECTLY ADDRESSING MODE..... | 36 |
| 2.2.3 INDIRECTLY ADDRESSING MODE..... | 36 |
| 2.3 STACK OPERATION..... | 37 |
| 2.3.1 OVERVIEW..... | 37 |
| 2.3.2 STACK REGISTERS..... | 38 |
| 2.3.3 STACK OPERATION EXAMPLE..... | 39 |
| 3 RESET..... | 40 |
| 3.1 OVERVIEW..... | 40 |
| 3.2 POWER ON RESET..... | 41 |
| 3.3 WATCHDOG RESET..... | 41 |
| 3.4 BROWN OUT RESET..... | 42 |
| 3.4.1 BROWN OUT DESCRIPTION..... | 42 |
| 3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION..... | 43 |
| 3.4.3 BROWN OUT RESET IMPROVEMENT..... | 43 |
| 3.5 EXTERNAL RESET..... | 45 |

| | | |
|----------|--|-----------|
| 3.6 | EXTERNAL RESET CIRCUIT | 45 |
| 3.6.1 | Simply RC Reset Circuit | 45 |
| 3.6.2 | Diode & RC Reset Circuit | 46 |
| 3.6.3 | Zener Diode Reset Circuit | 46 |
| 3.6.4 | Voltage Bias Reset Circuit | 47 |
| 3.6.5 | External Reset IC | 47 |
| 4 | SYSTEM CLOCK | 48 |
| 4.1 | OVERVIEW | 48 |
| 4.2 | CLOCK BLOCK DIAGRAM | 48 |
| 4.3 | OSCM REGISTER | 49 |
| 4.4 | SYSTEM HIGH CLOCK | 50 |
| 4.4.1 | INTERNAL HIGH RC | 50 |
| 4.4.2 | EXTERNAL HIGH CLOCK | 50 |
| 4.5 | SYSTEM LOW CLOCK | 52 |
| 4.5.2 | SYSTEM CLOCK MEASUREMENT | 54 |
| 5 | SYSTEM OPERATION MODE | 55 |
| 5.1 | OVERVIEW | 55 |
| 5.2 | SYSTEM MODE SWITCHING | 56 |
| 5.3 | WAKEUP | 58 |
| 5.3.1 | OVERVIEW | 58 |
| 5.3.2 | WAKEUP TIME | 58 |
| 6 | INTERRUPT | 59 |
| 6.1 | OVERVIEW | 59 |
| 6.2 | INTEN INTERRUPT ENABLE REGISTER | 59 |
| 6.3 | INTRQ INTERRUPT REQUEST REGISTER | 60 |
| 6.4 | GIE GLOBAL INTERRUPT OPERATION | 60 |
| 6.5 | PUSH, POP ROUTINE | 61 |
| 6.6 | INT0 (P0.0) INTERRUPT OPERATION | 62 |
| 6.7 | T0 INTERRUPT OPERATION | 63 |
| 6.8 | TC0 INTERRUPT OPERATION | 65 |
| 6.9 | MULTI-INTERRUPT OPERATION | 65 |
| 7 | I/O PORT | 67 |
| 7.1 | I/O PORT MODE | 67 |
| 7.2 | I/O PULL UP REGISTER | 68 |
| 7.3 | I/O PORT DATA REGISTER | 69 |
| 8 | TIMERS | 70 |

| | | |
|-----------|---|-----------|
| 8.1 | WATCHDOG TIMER (WDT) | 70 |
| 8.2 | TIMER 0 (T0) | 72 |
| 8.2.1 | OVERVIEW | 72 |
| 8.2.2 | T0M MODE REGISTER | 73 |
| 8.2.3 | T0C COUNTING REGISTER | 74 |
| 8.2.4 | T0 TIMER OPERATION SEQUENCE | 75 |
| 8.3 | TIMER/COUNTER 0 (TC0) | 76 |
| 8.3.1 | OVERVIEW | 76 |
| 8.3.2 | TC0M MODE REGISTER | 77 |
| 8.3.3 | TC0GN FLAGS | 78 |
| 8.3.4 | TC0C COUNTING REGISTER | 79 |
| 8.3.5 | TC0R AUTO-LOAD REGISTER | 80 |
| 8.3.6 | TC0 CLOCK FREQUENCY OUTPUT (BUZZER) | 81 |
| 8.3.7 | TC0 TIMER OPERATION SEQUENCE | 82 |
| 9 | LCD DRIVER | 83 |
| 9.1 | LCD TIMING | 83 |
| 9.2 | LCDM1 REGISTER | 85 |
| 9.3 | LCDM2 REGISTER | 86 |
| 9.4 | C-TYPE LCD DRIVER MODE | 87 |
| 9.5 | R-TYPE LCD DRIVER MODE | 88 |
| 9.6 | LCD RAM LOCATION | 90 |
| 10 | IN SYSTEM PROGRAM ROM | 91 |
| 10.1 | OVERVIEW | 91 |
| 10.2 | ROMADRH/ROMADRL REGISTER | 91 |
| 10.3 | ROMDAH/ROMADL REGISTERS | 91 |
| 10.4 | ROMCNT REGISTERS AND ROMWRT INSTRUCTION | 92 |
| 10.5 | ISP ROM ROUTINE EXAMPLE | 93 |
| 11 | REGULATOR, PGIA AND ADC | 94 |
| 11.1 | OVERVIEW | 94 |
| 11.2 | ANALOG INPUT | 94 |
| 11.3 | VOLTAGE REGULATOR | 95 |
| 11.3.1 | CPM-Charge Pump Mode Register | 95 |
| 11.4 | PGIA -PROGRAMMABLE GAIN INSTRUMENTATION AMPLIFIER | 96 |
| 11.4.1 | AMPM- Amplifier Mode Register | 96 |
| 11.4.2 | AMPCKS- PGIA CLOCK SELECTION | 97 |
| 11.4.3 | AMPCHS-PGIA CHANNEL SELECTION | 98 |
| 11.4.4 | Temperature Sensor (TS) | 99 |
| 11.5 | 16-BIT ADC | 102 |

| | | |
|-----------|---|------------|
| 11.5.1 | ADCM- ADC Mode Register | 102 |
| 11.5.2 | ADCKS- ADC Clock Register | 105 |
| 11.5.3 | ADC Data Register..... | 106 |
| 11.5.4 | DFM-ADC Digital Filter Mode Register | 107 |
| 11.5.5 | LBTM: Low Battery Detect Register | 111 |
| 11.5.6 | Analog Setting and Application..... | 112 |
| 12 | APPLICATION CIRCUIT | 114 |
| 12.1 | SCALE (LOAD CELL) APPLICATION CIRCUIT | 114 |
| 12.2 | THERMOMETER APPLICATION CIRCUIT | 115 |
| 13 | INSTRUCTION SET TABLE..... | 116 |
| 14 | DEVELOPMENT TOOLS | 117 |
| 14.1 | DEVELOPMENT TOOL VERSION | 117 |
| 14.1.1 | ICE (In circuit emulation)..... | 117 |
| 14.1.2 | OTP Writer | 117 |
| 14.1.3 | IDE (Integrated Development Environment) | 117 |
| 14.2 | OTP PROGRAMMING PIN TO TRANSITION BOARD MAPPING | 118 |
| 14.2.1 | The pin assignment of Easy and MP EZ Writer transition board socket:..... | 118 |
| 14.2.2 | The pin assignment of Writer V3.0 transition board socket:..... | 118 |
| 14.2.3 | SN8P1937 Series Programming Pin Mapping: | 119 |
| 14.3 | SN8P1937 EV-KIT..... | 120 |
| 14.3.1 | INTRODUCTION..... | 120 |
| 14.3.2 | PCB DESCRIPTION | 120 |
| 14.3.3 | EV BOARD SETTING | 121 |
| 14.3.4 | SN8P1937 EV BOARD CONNECT WITH ICE..... | 122 |
| 14.3.5 | STAND ALONG EV BOARD..... | 122 |
| 14.3.6 | SONIX ASSEMBLER..... | 122 |
| 14.3.7 | SYSTEM REQUIREMENT | 122 |
| 14.3.8 | NOTE FOR SOFTWARE INSTALLATION | 123 |
| 14.3.9 | EXAMPLE: PROGRAM CODE STRUCTURE | 123 |
| 14.3.10 | OTP WRITE-IN STEP | 125 |
| 14.3.11 | SN8P1937 PROGRAMMING BOARD CONNECT TO MPIII WRITER..... | 126 |
| | APPENDIX A: EV-KIT BOARD CIRCUIT | 127 |
| 15 | ELECTRICAL CHARACTERISTIC | 128 |
| 15.1 | ABSOLUTE MAXIMUM RATING | 128 |
| 15.2 | ELECTRICAL CHARACTERISTIC..... | 128 |
| 16 | PACKAGE INFORMATION | 131 |

| | | |
|-----------|------------------------------------|------------|
| 16.1 | LQFP 64 PIN | 131 |
| 17 | MARKING DEFINITION..... | 133 |
| 17.1 | INTRODUCTION..... | 133 |
| 17.2 | MARKING INDETIFICATION SYSTEM..... | 133 |
| 17.3 | MARKING EXAMPLE | 134 |
| 17.4 | DATECODE SYSTEM..... | 134 |

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 | | | | | | |
| SN8P1907 | 2K*16 | 128*8 | 8 | 4*12 | V | - | - | 11 | 16-bit | - | - | 5 | SSOP48 |
| SN8P1917 | 2K*16 | 128*8 | 8 | 4*12 | V | - | - | 13 | 16-bit | - | - | 5 | SSOP48 |
| SN8P1927 | 2K*16 | 128*8 | 8 | 4*12 | V | - | - | 13 | 16-bit | - | - | 5 | SSOP48/LQFP48 |
| SN8P1937 | 2K*16 | 128*8 | 8 | 4*12 | V | V | - | 13 | 16-bit | 1 | V | 5 | LQFP64 |

Table 1-1 Selection table of SN8P19x7 serial

1.2 MIGRATION TABLE

Migration SN8P1937 to SN8P1927/SN8P1917 Series

| Item | SN8P1917 | SN8P1927 | SN8P1937 |
|-------------------------------------|------------------------------|--------------------------------|--|
| PGIA Gain setting | 1x, 12.5x, 50x, 100x, 200x | 1x, 12.5x, 50x, 100x, 200x | 1x, 16x, 32x, 64x, 128x (ADC Gain option ~ 4x) |
| PGIA Temperature Drift | Good | Good | Good |
| AVE+ Voltage | 3.0V or 1.5V | 2.0V or 1.5V | 1.5V |
| AVE+ current loading when CPR ON | Double Current Consumption | NOT Double Current Consumption | NOT Double Current Consumption |
| ADC Reference Voltage V(R+, R-) | 0.8V or 0.4V | 0.4V | 0.6V or 0.3V |
| Battery Detect Method | By Comparator or By ADC | By Comparator or By ADC | By Comparator or By ADC |
| Temperature Sensor | Build In | Build In | Build In |
| ACM Voltage | Not Change with Sink current | Not Change with Sink current | Not Change with Sink current |
| Charge pump clock frequency (CPCKS) | 4-Bit Selection | 4-Bit Selection | No |
| Chopper clock frequency (AMPCKS) | 3-Bit Selection | 3-Bit Selection | 3-Bit Selection |
| AVDDR/AVE+ working in slow mode | Yes | Yes | Yes |
| Operating Current Consumption | Less | Less | Least |
| Slow mode Current Consumption | Less | Less | Least |
| LCD Bias Voltage | 1/3 or 1/2 Bias | 1/3 or 1/2 Bias | 1/3 or 1/2 Bias |
| LCD Type | R type only | R type only | R type/ C type |
| VLCD Voltage | Fixed | Fixed | Adjustable |
| Internal 16M RC Oscillator | Yes | Yes | Yes |
| P2 [1:0] I/O | Available when Fosc=IHRC | Available when Fosc=IHRC | Available when Fosc=IHRC |
| OTP Programming Method | Serial Method | Serial Method | Serial Method |
| In-System Programmer ROM | No | Yes | Yes |

Table 1-2 SN8P1927 Migration Table

1.3 FEATURES

Memory configuration

OTP ROM size: 2K * 16 bits
RAM size: 128 * 8 bits (bank 0)
8-levels stack buffer
LCD RAM size: 4*12bits

◆ **I/O pin configuration**

Input only: P0
Bi-directional: P1, P2, P5
Wakeup: P0, P1
Pull-up resistors: P0, P1, P2, P5
External interrupt: P0

◆ **Powerful instructions**

Four clocks per instruction cycle
All instructions are one word length
Most of instructions are 1 cycle only
Maximum instruction cycle is "2"
JMP instruction jumps to all ROM area
All ROM area look-up table function (MOVC)

Two 8-bit Timer/Counter

T0: Basic Timer with 0.5sec RTC
TC0: Auto-Reload timer/Counter/Buzzer/PWM output

◆ **Programmable Gain Instrumentation Amplifier**

◆ **PGIA Gain option: 1x/16x/32x/64x/128x**

◆ **16-bit Delta-Sigma ADC with 14-bit noise free**

ADC Gain selection: 1x, 2x, 4x
Two ADC channel configuration:
One fully differential channel
Two single channels

Two interrupt sources

Two internal interrupts: T0, TC0
One external interrupts: INT0

- ◆ **Single power supply: 2.4V ~5.5V**
- ◆ **On-chip watchdog timer**
- ◆ **On-chip Regulator (AVDDR) with 2.4V voltage output and 10mA driven current**
- ◆ **On chip regulator with 1.5V output voltage**
AVE+ Loading current consumption will Not double
- ◆ **On-chip 1.2V Band gap reference for battery monitor**
- ◆ **On chip Voltage Comparator**
- ◆ **Build in ADC reference voltage $V(R+,R-)= 0.3V/0.6V$**
- ◆ **One set Buzzer/ PWM**
- ◆ **Build in 0.5 sec RTC mode**
- ◆ **In-system Programmer ROM**

◆ **LCD driver:**

1/3 or 1/2 bias voltage.
4 common * 12 segment
Both R type and C type LCD
Multiple C-type LCD Voltage: 2.7 ~3.4V

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

Internal high clock: RC type up to 16 MHz
External high clock: Crystal type up to 8 MHz
Normal mode: Both high and low clock active
Slow mode: Low clock only
Green mode: Low clock active and optional high clock
Sleep mode: Both high and low clock stop

◆ **Package**

Dice / LQFP64

1.4 SYSTEM BLOCK DIAGRAM

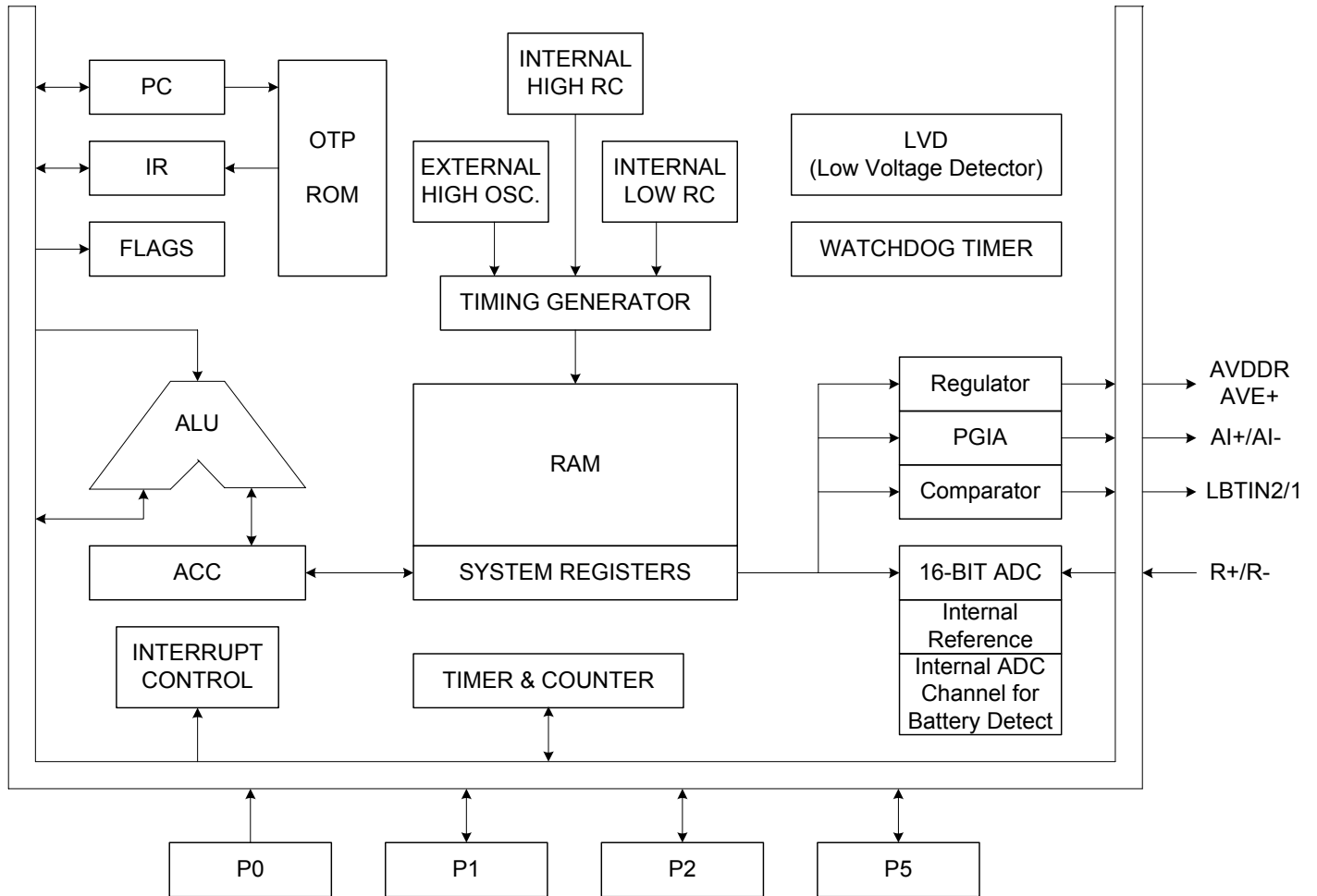
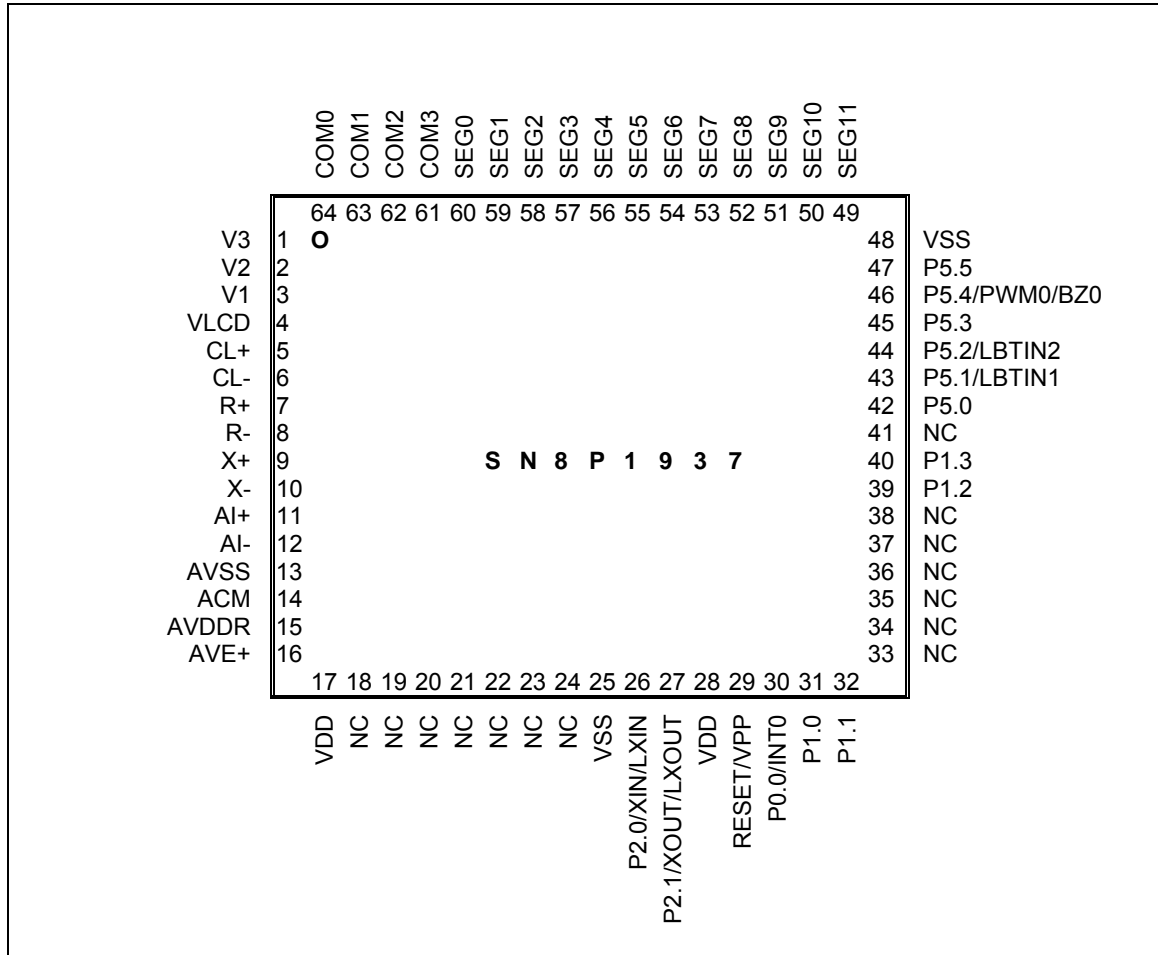


Figure 1-1 Simplified system block diagram

1.5 PIN ASSIGNMENT

SN8P1937 LQFP64

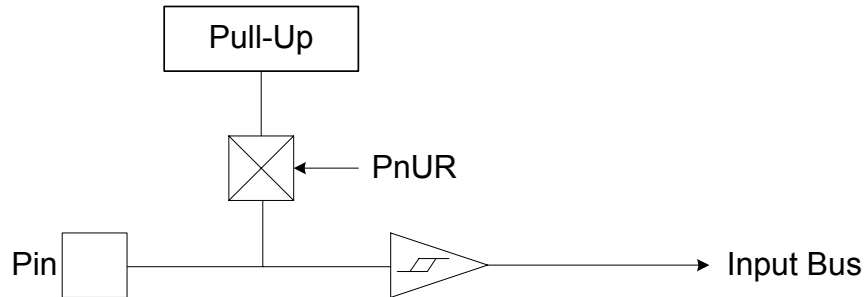


1.6 PIN DESCRIPTIONS

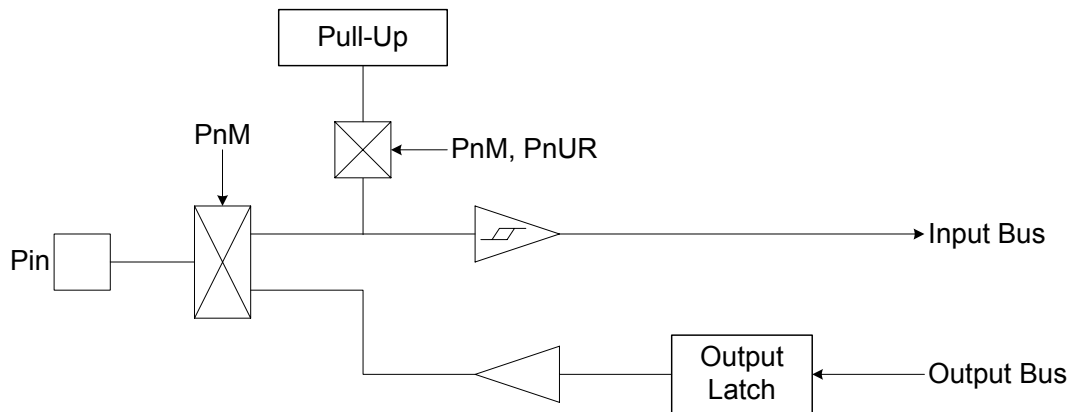
| PIN NAME | TYPE | DESCRIPTION |
|-------------------|-------------|--|
| VDD, VSS, AVSS | P | Power supply input pins for digital / Analog circuit |
| AVDDR | P | Regulator power output pin, Voltage = 2.4V |
| AVE+ | P | Regulator output = 1.5V for Sensor. Maximum output current = 10mA |
| ACM | P | ACM Voltage output = 0.4V |
| R+ | AI | ADC positive reference input |
| R- | AI | ADC negative reference input |
| X+ | AI | ADC positive differential input |
| X- | AI | ADC negative differential input |
| AI+ | AI | Positive analog input channel |
| AI- | AI | Negative analog input channel |
| VPP/ RST | P, I | OTP ROM programming pin System reset input pin. Schmitt trigger structure, active "low", normal stay to "high" |
| XIN / LXIN / P20 | I, O | External High clock oscillator pins. (4M Crystal code option) External Low clock 32768Hz oscillator pins. (IHRC_RTC code option) Share with P2.0 I/O |
| XOUT / LXOUT/ P21 | I, O | External High clock oscillator pins. (4M Crystal code option) External Low clock 32768Hz oscillator pins. (IHRC_RTC code option) Share with P2.1 I/O |
| P0.0 / INT0 | I | Port 0.0 and share with INT0 trigger pin (Schmitt trigger) / Built-in pull-up resistors. |
| P1 [3:0] | I/O | Port 1.0~Port 1.3 bi-direction pins / wakeup pins/ Built-in pull-up resistors |
| P2 [1:0] | I/O | Port 2.0~Port 2.1 bi-direction pins / Built-in pull-up resistors. Share with XIN/LXIN, XOUT/LXOUT |
| P5 [5:0] | I/O | Port 5.0~Port 5.5 bi-direction pins / Built-in pull-up resistors P5.4 shared with BZ0/PWM0 |
| LBTIN1 | I | Low Battery detect Input pins shared with P5.1 |
| LBTIN2 | I | Low Battery detect Input pins shared with P5.2 |
| COM [3:0] | O | COM0~COM3 LCD driver common port |
| SEG0 ~ SEG11 | O | LCD driver segment pins |
| CL+, CL- | P | C-Type LCD charge pump capacitor |
| VLCD | P | LCD driver power pin (no additional connect to external power) |
| V3 | P | LCD bias voltage |
| V2 | P | LCD bias voltage |
| V1 | P | LCD bias current active/ inactive control pad |
| | | |

1.7 PIN CIRCUIT DIAGRAMS

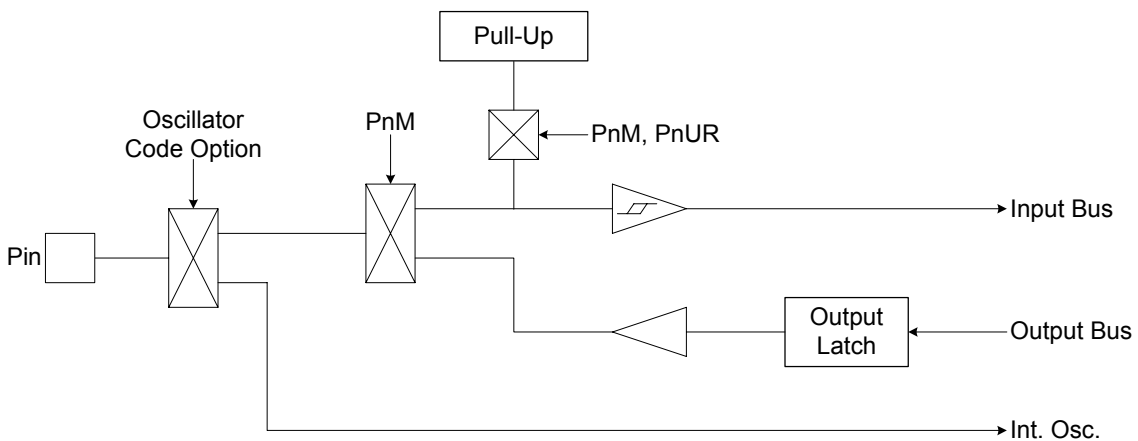
Port 0 structure:



Port1 and Port5 structure:



Port2 structure:



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ 2K words ROM

| ROM | | |
|------------|-----------------------------|----------------------------|
| 0000H | Reset vector | User reset vector |
| 0001H | General purpose area | Jump to user start address |
| 0002H | | Jump to user start address |
| 0003H | | Jump to user start address |
| 0004H | Reserved | |
| 0005H | | |
| 0006H | | |
| 0007H | | |
| 0008H | Interrupt vector | User interrupt vector |
| 0009H | General purpose area | User program |
| . | | |
| 000FH | | |
| 0010H | | |
| 0011H | | |
| . | | |
| 7FEH | | |
| 7FFH | Reserved | End of user program |

2.1.1.1 RESET VECTOR (0000H)

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

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

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

➤ Example: Defining Reset Vector

```

                ORG      0                ; 0000H
                JMP      START           ; Jump to user program address.
                ...
START:       ORG      10H                ; 0010H, The head of user program.
                ...                ; User program
                ...
                ENDP                ; End of program
```

2.1.1.2 INTERRUPT VECTOR (0008H)

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

* **Note: Users have to save and load ACC and PFLAG register by program as interrupt occurrence.**

➤ **Example: Defining Interrupt Vector. The interrupt service routine is following ORG 8.**

```
.DATA          ACCBUF   DS  1          ; Define ACCBUF for store ACC data.
               PFLAGBUF DS  1          ; Define PFLAGBUF for store PFLAG data.

.CODE

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

               ORG      8              ; Interrupt vector.
               BOXCH   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.
               BOXCH   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 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: ;
        ; 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
    ...

```

2.1.2.1 JUMP TABLE DESCRIPTION

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

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

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

➤ **Example: Jump table.**

```

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

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

```

In following example, the jump table starts at 0x00FD. When execute B0ADD PCL, A. If ACC = 0 or 1, the jump table points to the right address. If the ACC is larger than 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)
             ENDF
             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
```


2.1.2.2 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     C
B0BSET  FC                ; Clear C flag
ADD     DATA1, A         ; Add A to Data1
MOV     A, R
ADC     DATA2, A         ; Add R to Data2
JMP     END_CHECK        ; Check if the YZ address = the end of code

AAA:
INCMS   Z                 ; Z=Z+1
JMP     @B                ; If Z != 00H calculate to next address
JMP     Y_ADD_1          ; If Z = 00H increase Y

END_CHECK:
MOV     A, END_ADDR1
CMPRS  A, Z               ; Check if Z = low end address
JMP     AAA              ; If Not jump to checksum calculate
MOV     A, END_ADDR2
CMPRS  A, Y               ; If Yes, check if Y = middle end address
JMP     AAA              ; If Not jump to checksum calculate
JMP     CHECKSUM_END     ; If Yes checksum calculated is done.

Y_ADD_1:
INCMS   Y                 ; Increase Y
NOP
JMP     @B                ; Jump to checksum calculate

CHECKSUM_END:
...
...
END_USER_CODE:          ; Label of program end

```

2.1.3 CODE OPTION TABLE

| Code Option | Content | Function Description |
|-------------|-----------|---|
| High_Clk | IHRC | High speed internal 16MHz RC. XIN / XOUT become to P2.0 / P2.1 bi-direction I/O pins |
| | IHRC_RTC | High speed internal 16MHz RC and external 32768Hz crystal |
| | 4M X'tal | Standard crystal /resonator (e.g. 4M) for external high clock oscillator |
| Watch_Dog | Enable | Enable Watchdog function |
| | Disable | Disable Watchdog function |
| Security | Enable | Enable ROM code Security function |
| | Disable | Disable ROM code Security function |
| INT_32K_RC | Always_ON | Force Watch Dog Timer clock source come from INT 32K RC Also INT 32K RC never stop both in power down and green mode that means Watch Dog Timer will always enable both in power down and green mode |
| | By_CPUM | Enable or Disable internal 32K(@ 3V) RC clock by CPUM register |
| Low Power | Enable | Enable Low Power function to save Operating current |
| | Disable | Disable Low Power function |

- * **Note1: In high noisy environment, set Watch_Dog as "Enable" and INT_32K_RC as "Always_ON" is strongly recommended.**
- * **Note2: Fcpu code option is only available for High Clock. Fcpu of slow mode is Ffosc/4.**
- * **Note3: In high noisy environment, disable "Low Power" is strongly recommended.**
- * **Note4: The side effect is to increase the lowest valid working voltage level if enable "Low Power" code option.**
- * **Note4: Enable "Low Power" option will reduce operating current except in slow mode.**

2.1.4 DATA MEMORY (RAM)

128 X 8-bit RAM

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

2.1.5 SYSTEM REGISTER

2.1.5.1 SYSTEM REGISTER TABLE

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---------|---------|--------|--------|--------|-------|-------|-------|--------|--------|-------|---------|-------|-------|-------|-------|
| 8 | - | - | R | Z | Y | - | PFLAG | RBANK | - | LCDM1 | LCDM2 | - | - | - | - | - |
| 9 | AMPM | AMPCHS | AMPCKS | ADCM | ADCKS | CPM | CCKS | DFM | ADC DL | ADC DH | LBTM | CPMTEST | - | - | - | - |
| A | ROMADRH | ROMADRL | ROMDAH | ROMDAL | ROMCNT | - | - | - | - | - | - | - | - | - | - | - |
| B | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | PEDGE |
| C | - | P1M | P2M | - | - | - | - | - | INTRQ | INTEN | OSCM | - | - | - | PCL | PCH |
| D | P0 | P1 | P2 | - | P4 | P5 | - | - | T0M | T0C | TC0M | TC0C | - | - | - | STKP |
| E | - | P1UR | P2UR | - | - | - | - | @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

| | |
|---|---|
| <p>Y, Z = Working, @YZ and ROM addressing register</p> <p>PFLAG = ROM page and special flag register</p> <p>AMPM = PGIA mode register</p> <p>AMPCKS = PGIA clock selection</p> <p>ADCKS = ADC clock selection</p> <p>CCKS = Charge pump clock selection</p> <p>ADC DL = ADC low-byte data buffer</p> <p>P_NM = Port N input/output mode register</p> <p>P_N = Port N data buffer</p> <p>INTEN = Interrupt enable register</p> <p>LCDM1 = LCD mode register</p> <p>LCDM2 = LCD mode register</p> <p>T0M = Timer 0 mode register</p> <p>T0C = Timer 0 counting register</p> <p>TC0M = Timer/Counter 0 mode register</p> <p>TC0C = Timer/Counter 0 counting register</p> <p>LBTM = Low Battery Detect Register</p> | <p>R = Working register and ROM look-up data buffer</p> <p>AMPCHS = PGIA channel selection</p> <p>ADCM = ADC's mode register</p> <p>CPM = Charge pump mode</p> <p>DFM = Decimation filter mode</p> <p>ADC DH = ADC high-byte data buffer</p> <p>P_NUR = Port N pull-up register</p> <p>INTRQ = Interrupt request register</p> <p>OSCM = Oscillator mode register</p> <p>PCH, PCL = Program counter</p> <p>STK0~STK7 = Stack 0 ~ stack 7 buffer</p> <p>@YZ = RAM YZ indirect addressing index pointer</p> <p>STKP = Stack pointer buffer</p> <p>ROMADRH/L = ISP ROM Address</p> <p>ROMDAH/L = ISP ROM Data</p> <p>ROMCNT = ISP ROM Counter</p> |
|---|---|

2.1.5.3 BIT DEFINITION of SYSTEM REGISTER

| Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | R/W | Name |
|---------|---------|----------|----------|----------|---------|----------|---------|---------|-----|---------|
| 082H | RBIT7 | RBIT6 | RBIT5 | RBIT4 | RBIT3 | RBIT2 | RBIT1 | RBIT0 | R/W | R |
| 083H | ZBIT7 | ZBIT6 | ZBIT5 | ZBIT4 | ZBIT3 | ZBIT2 | ZBIT1 | ZBIT0 | R/W | Z |
| 084H | YBIT7 | YBIT6 | YBIT5 | YBIT4 | YBIT3 | YBIT2 | YBIT1 | YBIT0 | R/W | Y |
| 086H | - | - | - | - | - | C | DC | Z | R/W | PFLAG |
| 087H | - | - | - | - | RBNKS3 | RBNKS2 | RBNKS1 | RBNKS0 | R/W | RBANK |
| 089H | LCDREF1 | LCDREF0 | LCDBNK | LCDTYPE | LCDENB | LCDBIAS | LCDRATE | LCDCLK | R/W | LCDM1 |
| 08AH | - | - | BGM | LCDCPK1 | LCDCPK0 | VCP2 | VCP1 | VCP0 | R/W | LCDM2 |
| 090H | CHPENB | BGRENB | FDS1 | FDS0 | GS2 | GS1 | GS0 | AMPENB | R/W | AMPM |
| 091H | - | - | - | - | - | CHS2 | CHS1 | CHS0 | R/W | AMPCHS |
| 092H | - | - | - | - | - | AMPCKS2 | AMPCKS1 | AMPCKS0 | R/W | AMPCKS |
| 093H | ADG2 | ADG1 | ADG0 | RVS | IRVS | DTENB | DISEL | ADCENB | R/W | ADCM |
| 094H | ADCKS7 | ADCKS6 | ADCKS5 | ADCKS4 | ADCKS3 | ADCKS2 | ADCKS1 | ADCKS0 | W | ADCKS |
| 095H | ACMENB | AVDDRENB | AVESEL | AVENB | ACMSEL | - | - | - | R/W | CPM |
| 097H | OS2 | OS1 | OS0 | ADCHPENB | - | - | - | DRDY | R/W | DFM |
| 098H | ADCB7 | ADCB6 | ADCB5 | ADCB4 | ADCB3 | ADCB2 | ADCB1 | ADCB0 | R | ADCDL |
| 099H | ADCB15 | ADCB14 | ADCB13 | ADCB12 | ADCB11 | ADCB10 | ADCB9 | ADCB8 | R | ADCDH |
| 09AH | - | - | - | - | - | LBTO | P51IO | LBTENB | R/W | LBTM |
| 0A0H | VPPCHK | - | - | - | - | ROMADR10 | ROMADR9 | ROMADR8 | R/W | ROMADRH |
| 0A1H | ROMADR7 | ROMADR6 | ROMADR5 | ROMADR4 | ROMADR3 | ROMADR2 | ROMADR1 | ROMADR0 | R/W | ROMADRL |
| 0A2H | ROMDA15 | ROMDA14 | ROMDA13 | ROMDA12 | ROMDA11 | ROMDA10 | ROMDA9 | ROMDA8 | R/W | ROMDAH |
| 0A3H | ROMDA7 | ROMDA6 | ROMDA5 | ROMDA4 | ROMDA3 | ROMDA2 | ROMDA1 | ROMDA0 | R/W | ROMDAL |
| 0A4H | ROMCNT7 | ROMCNT6 | ROMCNT5 | ROMCNT4 | ROMCNT3 | ROMCNT2 | ROMCNT1 | ROMCNT0 | W | ROMCNT |
| 0BFH | PEDGEN | - | - | P00G1 | P00G0 | - | - | - | R/W | PEDGE |
| 0C1H | - | - | - | - | P13M | P12M | P11M | P10M | R/W | P1M |
| 0C2H | - | - | - | - | - | - | P21M | P20M | R/W | P2M |
| 0C5H | - | - | P55M | P54M | P53M | P52M | P51M | P50M | R/W | P5M |
| 0C8H | - | - | TC0IRQ | T0IRQ | - | - | - | P00IRQ | R/W | INTRQ |
| 0C9H | - | - | TC0IEN | T0IEN | - | - | - | P00IEN | R/W | INTEN |
| 0CAH | WTCKS | WDRST | WDRATE | CPUM1 | CPUM0 | CLKMD | STPHX | - | R/W | OSCM |
| 0CDH | TC0R7 | TC0R6 | TC0R5 | TC0R4 | TC0R3 | TC0R2 | TC0R1 | TC0R0 | W | TC0R |
| 0CEH | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 | R/W | PCL |
| 0CFH | - | - | - | - | - | PC10 | PC9 | PC8 | R/W | PCH |
| 0D0H | - | - | - | - | - | - | - | P00 | R | P0 |
| 0D1H | - | - | - | - | P13 | P12 | P11 | P10 | R/W | P1 |
| 0D2H | - | - | - | - | - | - | P21 | P20 | R/W | P2 |
| 0D5H | - | - | P55 | P54 | P53 | P52 | P51 | P50 | R/W | P5 |
| 0D8H | T0EN | T0RATE2 | T0RATE1 | T0RATE0 | - | - | TC0GN | T0TB | R/W | T0M |
| 0D9H | T0C7 | T0C6 | T0C5 | T0C4 | T0C3 | T0C2 | T0C1 | T0C0 | R/W | T0C |
| 0DAH | TC0EN | TC0RATE2 | TC0RATE1 | TC0RATE0 | TC0CKS | ALOAD0 | TC0OUT | PWM0OUT | R/W | TC0M |
| 0DBH | TC0C7 | TC0C6 | TC0C5 | TC0C4 | TC0C3 | TC0C2 | TC0C1 | TC0C0 | R/W | TC0C |
| 0DFH | GIE | - | - | - | - | STKPB2 | STKPB1 | STKPB0 | R/W | STKP |
| 0E1H | - | - | - | - | P13R | P12R | P11R | P10R | W | P1UR |
| 0E2H | - | - | - | - | - | - | P21R | P20R | W | P2UR |
| 0E5H | - | - | P55R | P54R | P53R | P52R | P51R | P50R | W | P5UR |
| 0E7H | @YZ7 | @YZ6 | @YZ5 | @YZ4 | @YZ3 | @YZ2 | @YZ1 | @YZ0 | R/W | @YZ |
| 0F0H | S7PC7 | S7PC6 | S7PC5 | S7PC4 | S7PC3 | S7PC2 | S7PC1 | S7PC0 | R/W | STK7L |
| 0F1H | - | - | - | - | - | S7PC10 | S7PC9 | S7PC8 | R/W | STK7H |
| 0F2H | S6PC7 | S6PC6 | S6PC5 | S6PC4 | S6PC3 | S6PC2 | S6PC1 | S6PC0 | R/W | STK6L |
| 0F3H | - | - | - | - | - | S6PC10 | S6PC9 | S6PC8 | R/W | STK6H |
| 0F4H | S5PC7 | S5PC6 | S5PC5 | S5PC4 | S5PC3 | S5PC2 | S5PC1 | S5PC0 | R/W | STK5L |
| 0F5H | - | - | - | - | - | S5PC10 | S5PC9 | S5PC8 | R/W | STK5H |
| 0F6H | S4PC7 | S4PC6 | S4PC5 | S4PC4 | S4PC3 | S4PC2 | S4PC1 | S4PC0 | R/W | STK4L |
| 0F7H | - | - | - | - | - | S4PC10 | S4PC9 | S4PC8 | R/W | STK4H |
| 0F8H | S3PC7 | S3PC6 | S3PC5 | S3PC4 | S3PC3 | S3PC2 | S3PC1 | S3PC0 | R/W | STK3L |
| 0F9H | - | - | - | - | - | S3PC10 | S3PC9 | S3PC8 | R/W | STK3H |
| 0FAH | S2PC7 | S2PC6 | S2PC5 | S2PC4 | S2PC3 | S2PC2 | S2PC1 | S2PC0 | R/W | STK2L |
| 0FBH | - | - | - | - | - | S2PC10 | S2PC9 | S2PC8 | R/W | STK2H |
| 0FCH | S1PC7 | S1PC6 | S1PC5 | S1PC4 | S1PC3 | S1PC2 | S1PC1 | S1PC0 | R/W | STK1L |
| 0FDH | - | - | - | - | - | S1PC10 | S1PC9 | S1PC8 | R/W | STK1H |

| | | | | | | | | | | |
|------|-------|-------|-------|-------|-------|--------|-------|-------|-----|-------|
| 0FEH | S0PC7 | S0PC6 | S0PC5 | S0PC4 | S0PC3 | S0PC2 | S0PC1 | S0PC0 | R/W | STK0L |
| 0FFH | - | - | - | | | S0PC10 | S0PC9 | S0PC8 | R/W | STK0H |

* **Note:**

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

2.1.5.4 ACCUMULATOR

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

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

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

; Read ACC data and store in BUF data memory

```
MOV     BUF, A
```

; Write a immediate data into ACC

```
MOV     A, #0FH
```

; Write ACC data from BUF data memory

```
MOV     A, BUF
```

The system doesn't store ACC and PFLAG value when interrupt executed. ACC and PFLAG data must be saved to other data memories 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    ; Save PFLAG register in a buffer.
           B0MOV    PFLAGBUF, A
           ...
           ...
           B0MOV    A, PFLAGBUF  ; Restore PFLAG register from buffer.
           B0MOV    PFLAG, A
           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.**

2.1.5.5 PROGRAM FLAG

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

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

- Bit 2 **C:** Carry flag
 1 = Addition with carry, subtraction without borrowing, rotation with shifting out logic "1", comparison result ≥ 0 .
 0 = Addition without carry, subtraction with borrowing signal, rotation with shifting out logic "0", comparison result < 0 .
- Bit 1 **DC:** Decimal carry flag
 1 = Addition with carry from low nibble, subtraction without borrow from high nibble.
 0 = Addition without carry from low nibble, subtraction with borrow from high nibble.
- Bit 0 **Z:** Zero flag
 1 = The result of an arithmetic/logic/branch operation is zero.
 0 = The result of an arithmetic/logic/branch operation is not zero.

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

2.1.5.6 PROGRAM COUNTER

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

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

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

☞ ONE ADDRESS SKIPPING

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

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

```

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

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

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

```

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

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

INCS instruction:

INCS BUF0
JMP C0STEP ; Jump to C0STEP if ACC is not zero.

...

...

C0STEP: NOP

INCMS instruction:

INCMS BUF0
JMP C0STEP ; Jump to C0STEP if BUF0 is not zero.

...

...

C0STEP: NOP

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

DECS instruction:

DECS BUF0
JMP C0STEP ; Jump to C0STEP if ACC is not zero.

...

...

C0STEP: NOP

DECMS instruction:

DECMS BUF0
JMP C0STEP ; Jump to C0STEP if BUF0 is not zero.

...

...

C0STEP: NOP

☞ MULTI-ADDRESS JUMPING

Users can jump around the multi-address by either JMP instruction or ADD M, A instruction (M = PCL) to activate multi-address jumping function. Program counter can't carry to PCH when PCL overflow automatically after executing addition instructions. Users have to take care program counter result and adjust PCH value by program. For jump table or others applications, users have to calculate PC value to avoid PCL overflow making PC error and program executing error.

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

➤ Example: If PC = 0323H (PCH = 03H, PCL = 23H)

```
; PC = 0323H
      MOV      A, #28H
      B0MOV   PCL, A          ; Jump to address 0328H
      ...
```

```
; PC = 0328H
      MOV      A, #00H
      B0MOV   PCL, A          ; Jump to address 0300H
      ...
```

➤ Example: If PC = 0323H (PCH = 03H, PCL = 23H)

```
; PC = 0323H
      B0ADD   PCL, A          ; PCL = PCL + ACC, the PCH cannot be changed.
      JMP    A0POINT         ; If ACC = 0, jump to A0POINT
      JMP    A1POINT         ; ACC = 1, jump to A1POINT
      JMP    A2POINT         ; ACC = 2, jump to A2POINT
      JMP    A3POINT         ; ACC = 3, jump to A3POINT
      ...
      ...
```

2.1.6 Y, Z REGISTERS

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

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

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

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

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

```

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

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

```

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

CLR_YZ_BUF:

```

CLR      @YZ               ; Clear @YZ to be zero
    
```

```

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

```

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

2.1.7 R REGISTERS

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

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

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

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

2.2 ADDRESSING MODE

2.2.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.2.2 DIRECTLY ADDRESSING MODE

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

- **Example: Move 0x12 RAM location data into ACC.**

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

- **Example: Move ACC data into 0x12 RAM location.**

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

2.2.3 INDIRECTLY ADDRESSING MODE

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

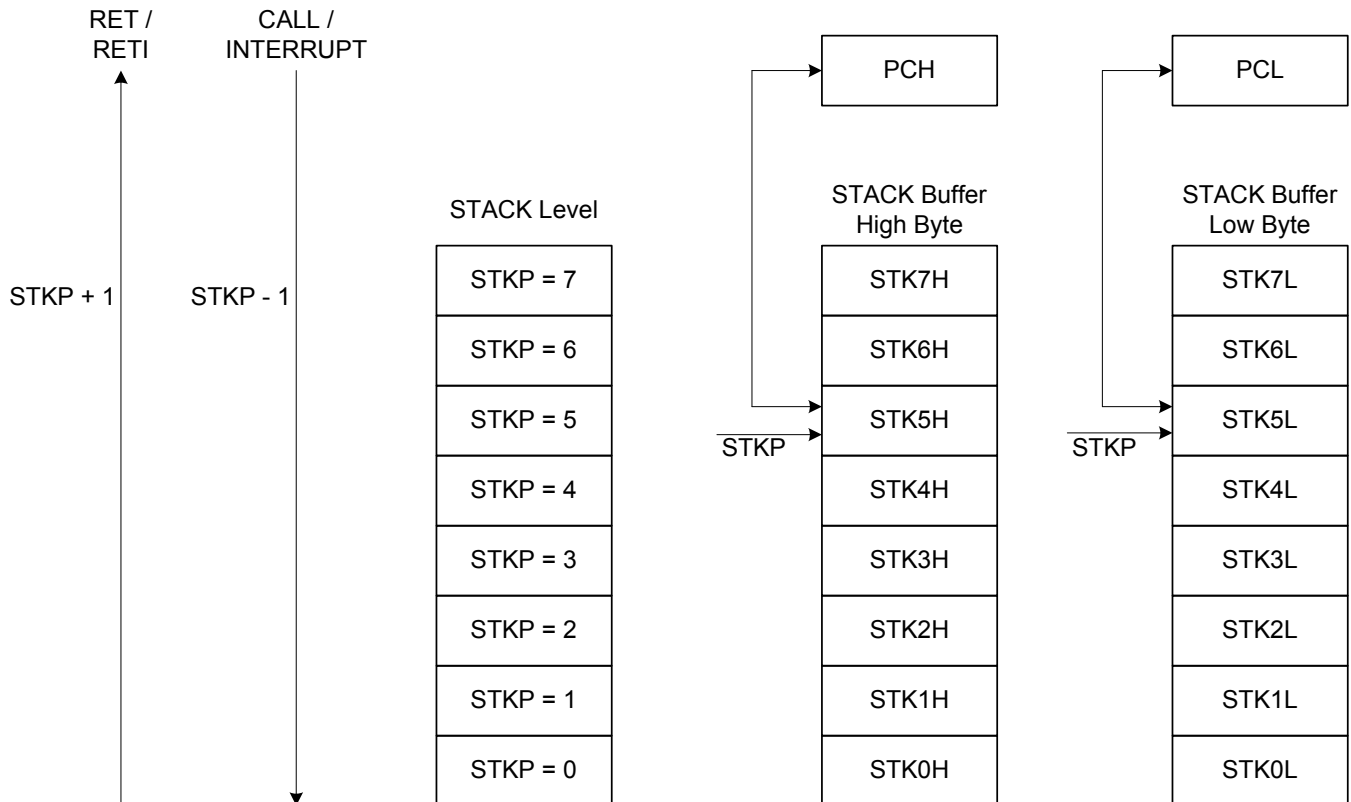
- **Example: Indirectly addressing mode with @YZ register.**

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

2.3 STACK OPERATION

2.3.1 OVERVIEW

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



2.3.2 STACK REGISTERS

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

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

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

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

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

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

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

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

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

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

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

2.3.3 STACK OPERATION EXAMPLE

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

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

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

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

3 RESET

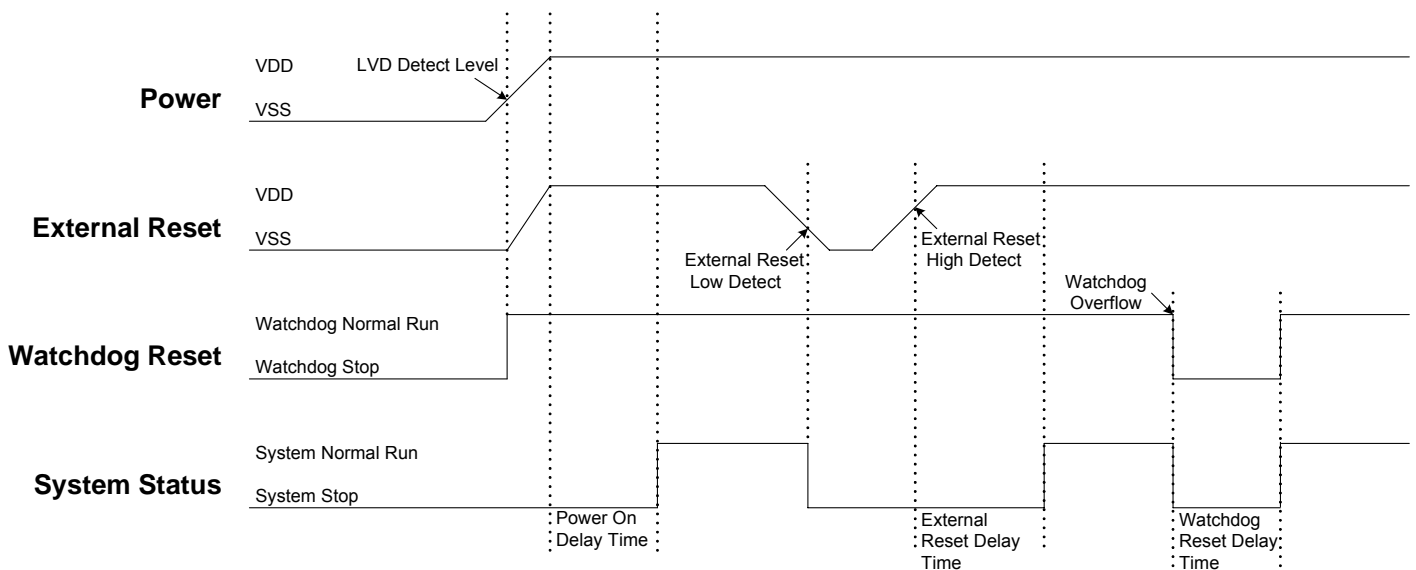
3.1 OVERVIEW

The system would be reset in three conditions as following.

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

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

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



3.2 POWER ON RESET

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

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

3.3 WATCHDOG RESET

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

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

Watchdog timer application note is as following.

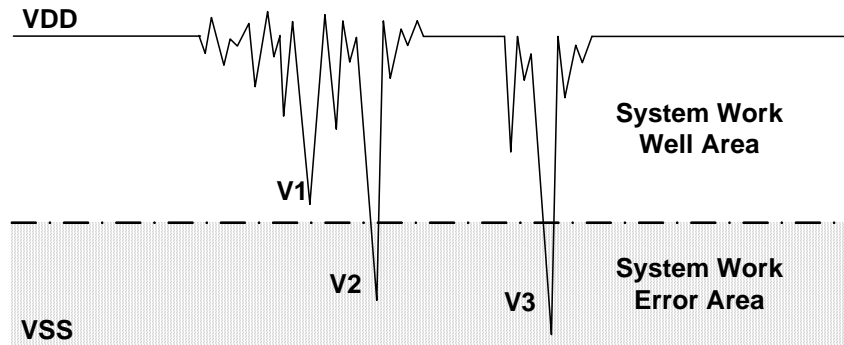
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

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

3.4 BROWN OUT RESET

3.4.1 BROWN OUT DESCRIPTION

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



Brown Out Reset Diagram

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

DC application:

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

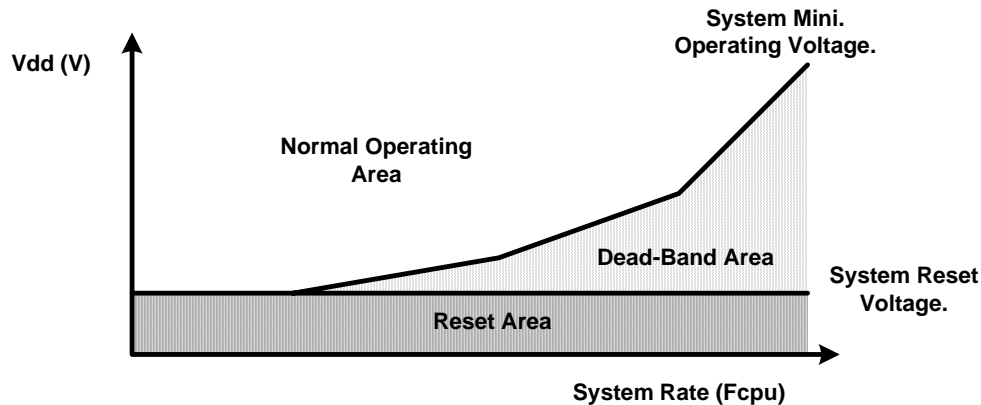
AC application:

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

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

3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION

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



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

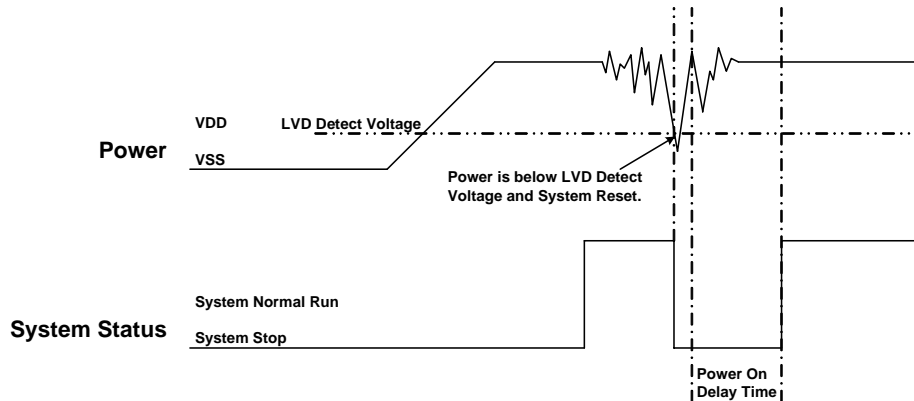
3.4.3 BROWN OUT RESET IMPROVEMENT

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

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

*** Note:**

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

LVD reset:

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

Watchdog reset:

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

If the system reset by watchdog and the power is still in dead-band, the system reset sequence won't be successful and the system stays in reset status until the power return to normal range.

Reduce the system executing rate:

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

External reset circuit:

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

3.5 EXTERNAL RESET

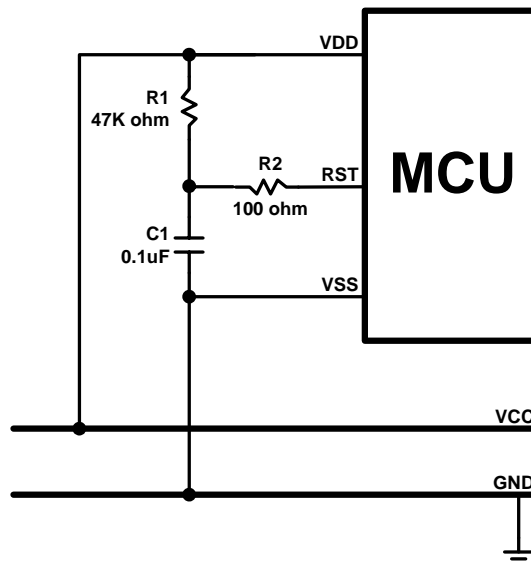
External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the low voltage and the system is reset. The external reset operation activates in power on and normal running mode. During system power-up, the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

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

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

3.6 EXTERNAL RESET CIRCUIT

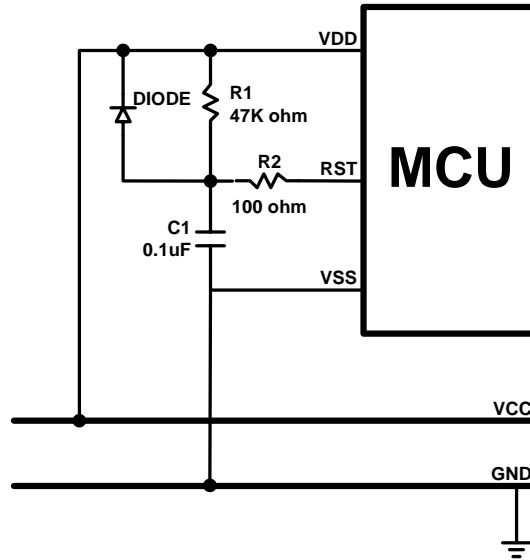
3.6.1 Simply RC Reset Circuit



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

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

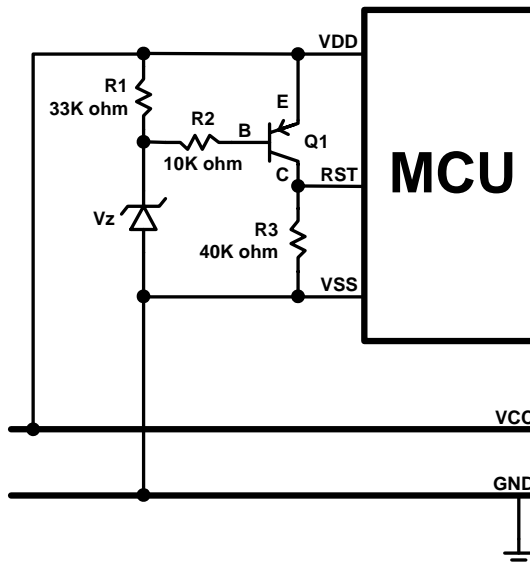
3.6.2 Diode & RC Reset Circuit



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

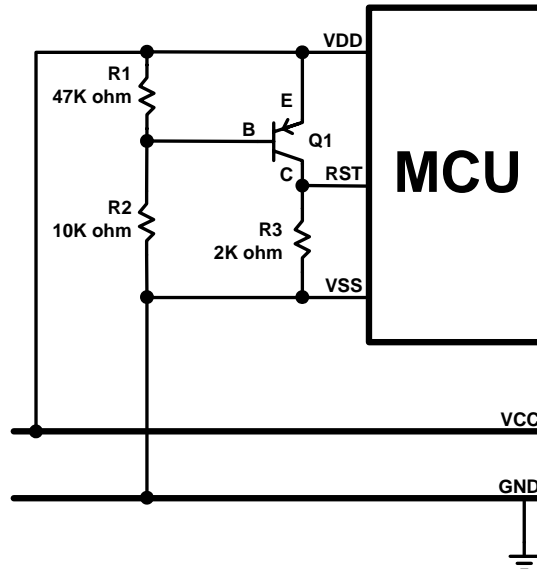
* **Note:** The R2 100 ohm resistor of “Simply reset circuit” and “Diode & RC reset circuit” is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).

3.6.3 Zener Diode Reset Circuit



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

3.6.4 Voltage Bias Reset Circuit

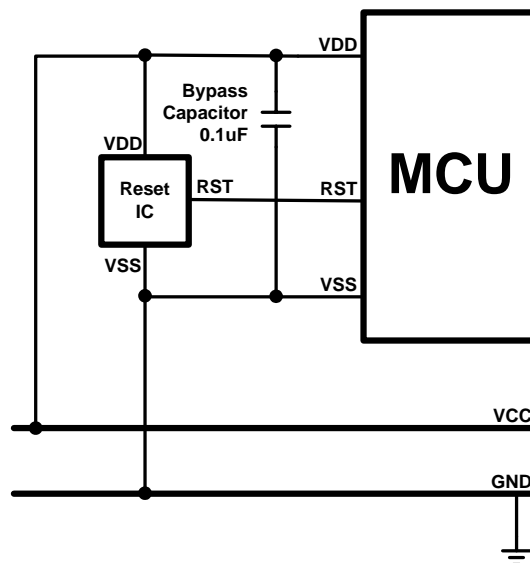


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to $0.7V \times (R1 + R2)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

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

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

3.6.5 External Reset IC



4 SYSTEM CLOCK

4.1 OVERVIEW

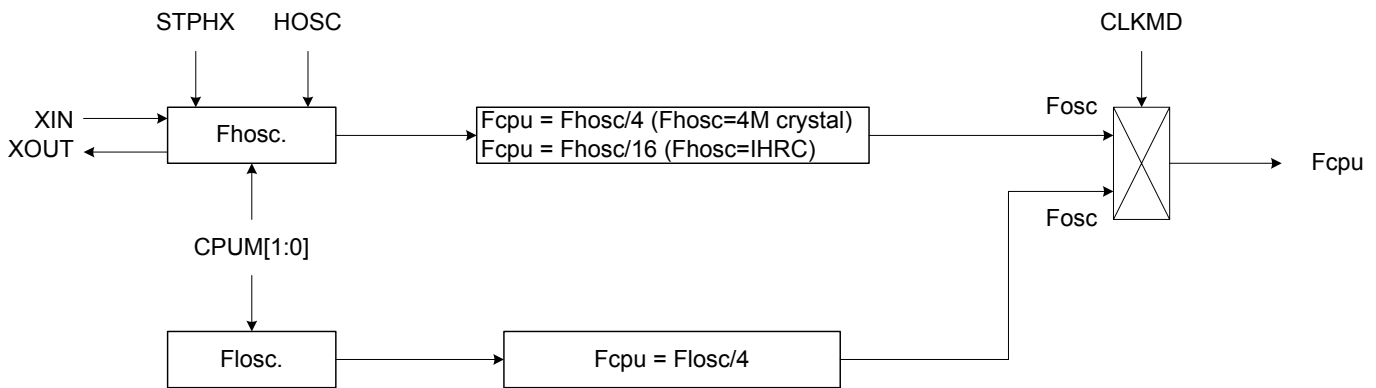
The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit or on-chip 16MHz high-speed RC oscillator circuit (IHRC 16MHz). The low-speed clock is generated from LXIN/LXOUT by 32768 crystal or RC oscillator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

Both the high-speed clock and the low-speed clock can be system clock (F_{osc}). The system clock in slow mode is divided by 4 to be the instruction cycle (F_{cpu}).

☞ **Normal Mode (High Clock):** $F_{cpu} = F_{osc} / 4$, ($F_{osc}= 4M/8M$ crystal)
 $F_{cpu} = F_{osc} / 16$, ($F_{osc}=IHRC$)

☞ **Slow Mode (Low Clock):** $F_{cpu} = F_{osc}/4$.

4.2 CLOCK BLOCK DIAGRAM



- HOSC: High_Clk code option.
- Fosc: System high clock source is from external crystal or internal high RC.
- Fosc: (1).System low clock source is from internal low RC. (code option = 4M, IHRC)
(2).System low clock source is from external 32768Hz crystal when system in IHRC_RTC mode.
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.4 SYSTEM HIGH CLOCK

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

| High_Clk Code Option | Description |
|----------------------|---|
| IHRC | The high clock is internal 16MHz oscillator RC type. XIN and XOUT pins are general purpose I/O pins. |
| IHRC_RTC | The high clock is internal 16MHz oscillator RC type. XIN/XOUT pins are connected with external 32768Hz crystal/ceramic oscillator for RTC clock source. |
| 4M | The high clock is external oscillator. The typical frequency is 4MHz. |

4.4.1 INTERNAL HIGH RC

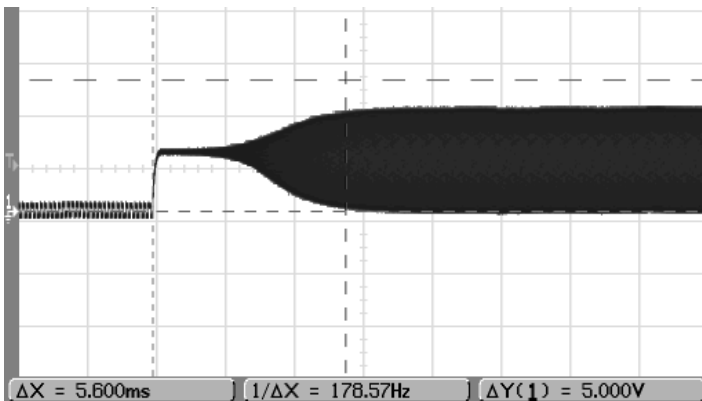
The chip is built-in RC type internal high clock (16MHz) controlled by "IHRC_16M" code options. In "IHRC_16M" mode, the system clock is from internal 16MHz RC type oscillator and XIN / XOUT pins are general-purpose I/O pins.

- **IHRC:** High clock is internal 16MHz oscillator RC type. XIN/XOUT pins are general purpose I/O pins.
- **IHRC_RTC:** High clock is internal 16MHz oscillator RC type. XIN/XOUT pins are connected with external 32768Hz crystal/ceramic oscillator for RTC clock source.

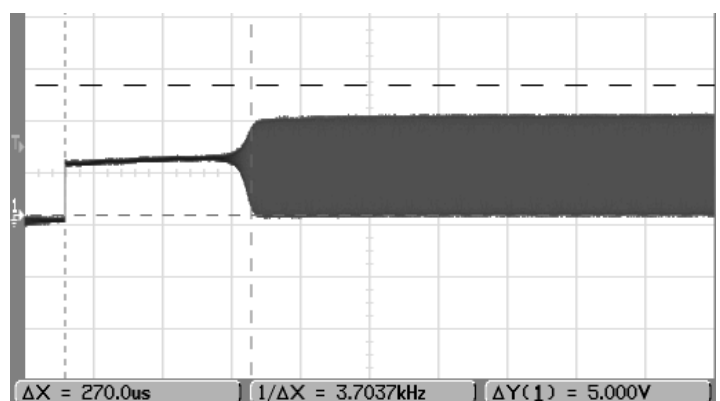
4.4.2 EXTERNAL HIGH CLOCK

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

4MHz Crystal

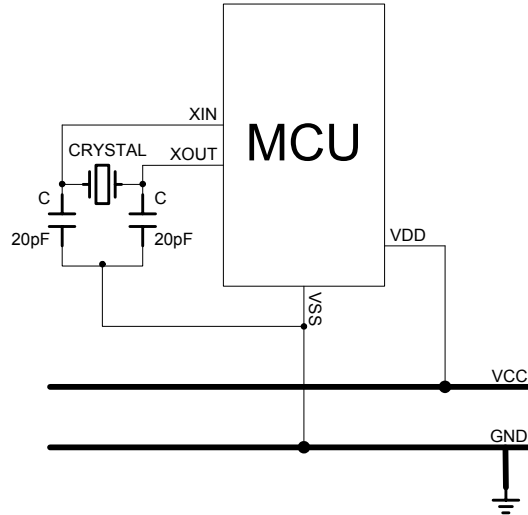


4MHz Ceramic



4.4.2.1 CRYSTAL/CERAMIC

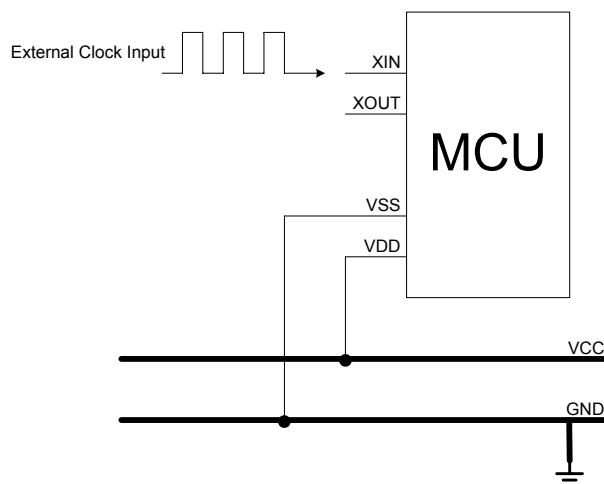
Crystal/Ceramic devices are driven by XIN, XOUT pins. For high/normal/low frequency, the driving currents are different. High_Clk code option supports different frequencies. 12M option is for high speed (ex. 12MHz). 4M option is for normal speed (ex. 4MHz).



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

4.4.2.2 EXTERNAL CLOCK SIGNAL

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



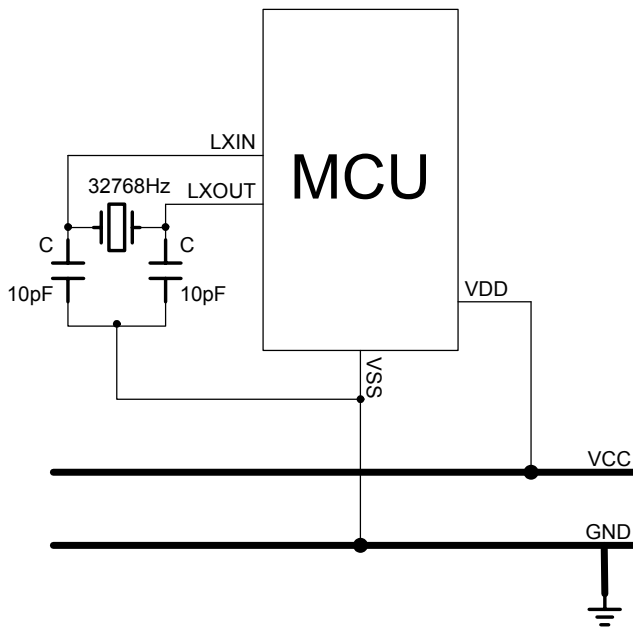
* **Note:** The GND of external oscillator circuit must be as near as possible to VSS pin of micro-controller.

4.5 SYSTEM LOW CLOCK

The system low clock source is from internal Low RC (ILRC) oscillator or external 32768Hz crystal. When IHRC_RTC code option selected, the system low clock is from external 32768Hz crystal. In the high clock mode of 4MHz and IHRC, system low clock is from ILRC.

4.5.1.1 CRYSTAL

Crystal devices are driven by LXIN, LXOUT pins. The 32768Hz crystal and 10uF capacitor must be as near as possible to MCU.

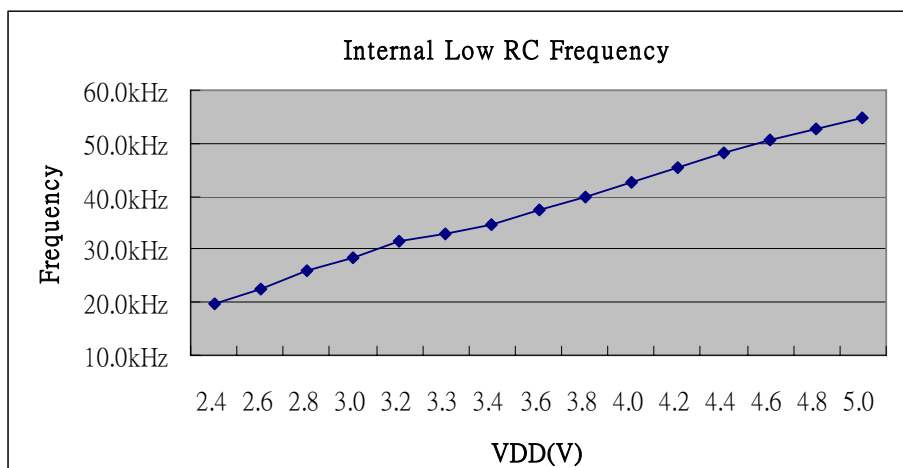


☞ $F_{osc} = 32768\text{Hz (Crystal)}$

☞ $\text{Slow mode } F_{cpu} = F_{osc} / 4$

4.5.1.2 Internal Low RC Type (ILRC)

The system low clock source is the internal low RC 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 3.3V. The relation between the RC frequency and voltage is as the following figure.



The ILRC clock supports watchdog clock source and system slow mode controlled by CLKMD.

☞ **$F_{osc} = ILRC$**

☞ **$Slow\ mode\ F_{cpu} = F_{osc} / 4$**

In power down mode the external low clock will be Stop.

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

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

* **Note: The external low-speed clock can't be turned off individually. It is controlled by CPUM0, CPUM1 bits of OSCM register.**

4.5.2 SYSTEM CLOCK MEASUREMENT

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

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

```
B0BSET    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
```

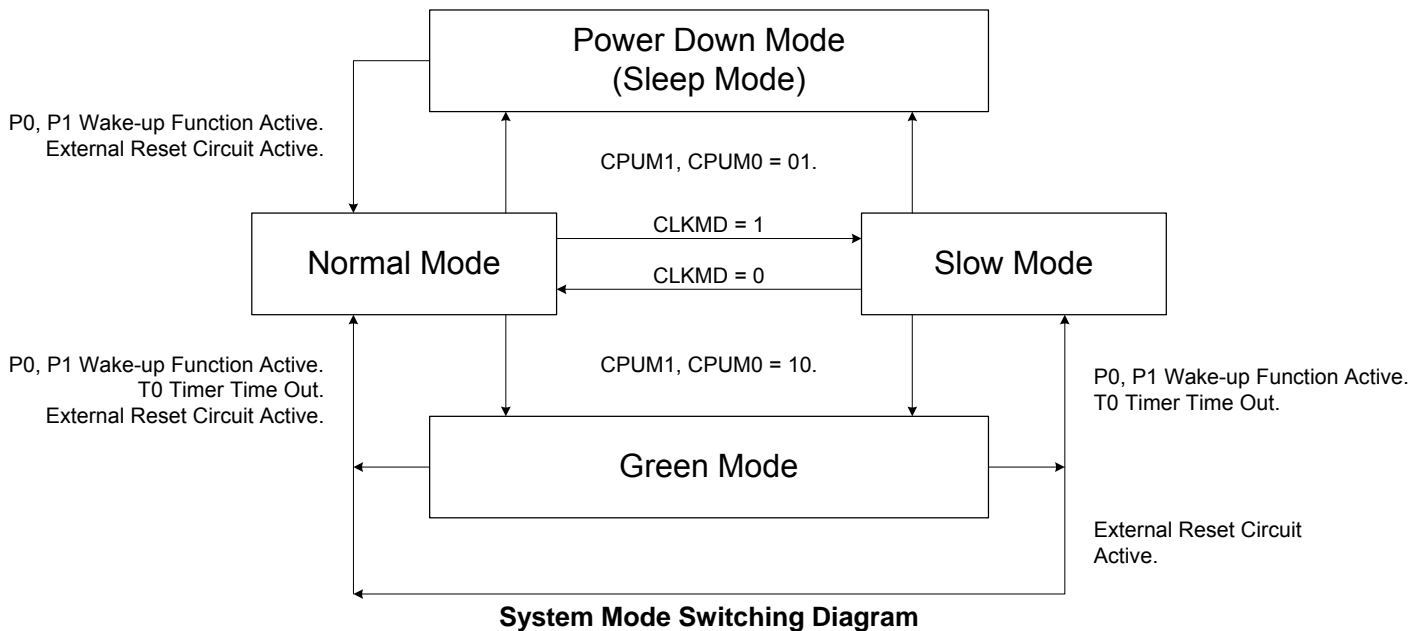
* **Note: Do not measure the RC frequency directly from XIN; the probe impedance will affect the RC frequency.**

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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



Operating mode description

| MODE | NORMAL | SLOW | GREEN | POWER DOWN (SLEEP) | REMARK |
|--------------------|--------------------------|--------------------------|--------------------------|--------------------------|----------------------------------|
| Fosc | 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 |
| TC0 timer | *Active | *Active | By TC0GN | Inactive | * Active if TC0ENB=1 |
| Watchdog timer | By Watch_Dog Code option | By Watch_Dog Code option | By Watch_Dog Code option | By Watch_Dog Code option | Refer to code option description |
| Internal interrupt | All active | All active | T0, TC0 | All inactive | TC0GN = 1 |
| External interrupt | All active | All active | All active | All inactive | |
| Wakeup source | - | - | P0, P1, T0, TC0 Reset | P0, P1, Reset | |

5.2 SYSTEM MODE SWITCHING

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

```
BOBSET          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.**

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

- **Example: Switch slow mode to normal mode (The external high-speed oscillator is still running)**

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

- **Example: Switch slow mode to normal mode (The external high-speed oscillator stops)**

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

```
BOBCLR          FSTPHX          ; Turn on the external high-speed oscillator.
@@:             BOMOV          Z, #54          ; If VDD = 5V, internal RC=32KHz (typical) will delay
                DECMS          Z              ; 0.125ms X 162 = 20.25ms for external clock stable
                JMP            @B
BOBCLR          FCLKMD          ; Change the system back to the normal mode
```

- **Example: Switch normal/slow mode to green mode.**

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

* **Note: If T0/TC0 timer wakeup function is disabled in the green mode, only the wakeup pin and reset pin can wakeup the system backs to the previous operation mode.**

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

; Set T0 timer wakeup function.

| | | |
|---------------|---------------|---|
| B0BCLR | FT0IEN | ; To disable T0 interrupt service |
| B0BCLR | FT0ENB | ; To disable T0 timer |
| MOV | A,#20H | ; |
| B0MOV | T0M,A | ; To set T0 clock = Fcpu / 64 |
| MOV | A,#74H | ; |
| B0MOV | T0C,A | ; To set T0C initial value = 74H (To set T0 interval = 10 ms) |
| B0BCLR | FT0IEN | ; To disable T0 interrupt service |
| B0BCLR | FT0IRQ | ; To clear T0 interrupt request |
| B0BSET | FT0ENB | ; To enable T0 timer |

; Go into green mode

| | | |
|--------|--------|--------------------|
| B0BCLR | FCPUM0 | ;To set CPUMx = 10 |
| B0BSET | FCPUM1 | |

* **Note: During the green mode with T0 wake-up function, the wakeup pins, reset pin and T0 can wakeup the system back to the last mode. T0 wake-up period is controlled by program and T0ENB must be set.**

5.3 WAKEUP

5.3.1 OVERVIEW

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

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

5.3.2 WAKEUP TIME

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

The value of the wakeup time is as the following.

$$\text{The Wakeup time} = 1/F_{osc} * 2048 \text{ (sec)} + \text{high clock start-up time}$$

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

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

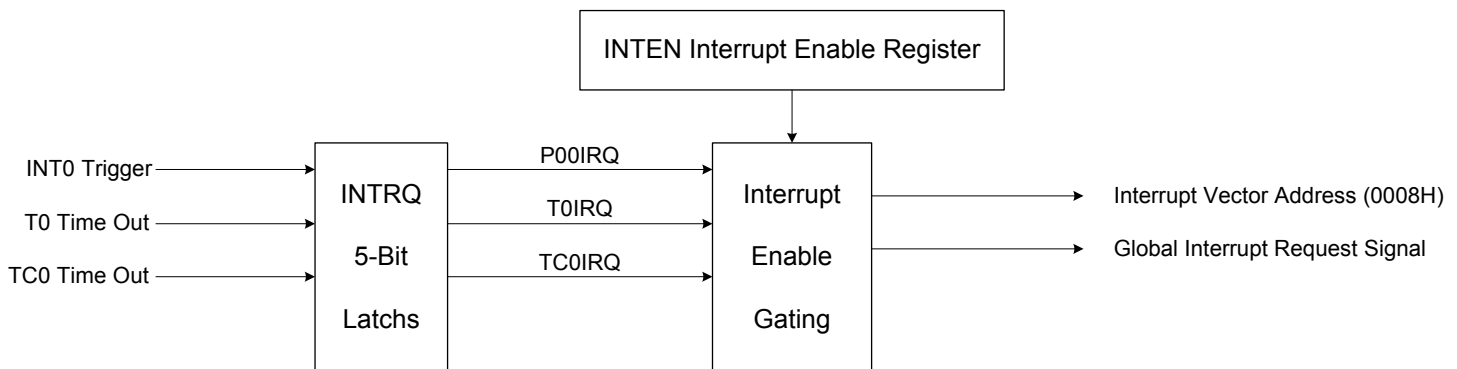
$$\text{The wakeup time} = 1/F_{osc} * 2048 = 0.512 \text{ ms} \quad (F_{osc} = 4\text{MHz})$$

$$\text{The total wakeup time} = 0.512 \text{ ms} + \text{oscillator start-up time}$$

6 INTERRUPT

6.1 OVERVIEW

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



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

6.2 INTEN INTERRUPT ENABLE REGISTER

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

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

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

Bit 4 **TOIEN:** 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.

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

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

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

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

6.5 PUSH, POP ROUTINE

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

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

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

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

.CODE

               ORG      0
               JMP      START

               ORG      8
               JMP      INT_SERVICE

START:         ORG      10H
               ...

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

               RETI                    ; Exit interrupt service vector
               ...
               ENDP
```

6.6 INTO (P0.0) INTERRUPT OPERATION

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

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

| 0BFH | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------------|--------|-------|-------|-------|-------|-------|-------|-------|
| PEDGE | PEDGEN | - | - | P00G1 | P00G0 | - | - | - |
| | R/W | - | - | R/W | R/W | - | - | - |

Bit7 **PEDGEN**: Interrupt and wakeup trigger edge control bit.
 0 = Disable edge trigger function.
 Port 0: Low-level wakeup trigger and falling edge interrupt trigger.
 Port 1: Low-level wakeup trigger.
 (In Green mode wakeup, P0 and P1 falling edge interrupt trigger only)

1 = Enable edge trigger function.
 P0.0: Both Wakeup and interrupt trigger are controlled by P00G1 and P00G0 bits.
 Port 1: Wakeup trigger is Level change (falling or rising edge).

Bit[4:3] **P00G[1:0]**: Port 0.0 edge select bits.
 00 = reserved,
 01 = falling edge
 10 = rising edge,
 11 = rising/falling bi-direction.

➤ **Example: Setup INTO interrupt request and bi-direction edge trigger.**

```

MOV      A, #98H
BOMOV   PEDGE, A      ; Set INTO interrupt trigger as bi-direction edge.

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

➤ **Example: INTO interrupt service routine.**

```

ORG      8             ; Interrupt vector
INT_SERVICE:
JMP     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
...     ; INTO interrupt service routine
...

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

RETI                    ; Exit interrupt vector
  
```

6.7 T0 INTERRUPT OPERATION

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

➤ Example: T0 interrupt request setup.

```

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

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

B0BSET    FGIE      ; Enable GIE

```

Example: T0 interrupt service routine as no RTC function.

```

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

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

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

RETI     ; Exit interrupt vector

```

- * **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, don't reset T0C in interrupt service routine.

Example: T0 interrupt service routine with RTC function.

```

ORG          8          ; Interrupt vector
INT_SERVICE:
  JMP          INT_SERVICE

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

  > 16us {
    B0BTS1    FT0IRQ    ; Check T0IRQ
    JMP      EXIT_INT  ; T0IRQ = 0, exit interrupt vector

    ...
    ; T0 interrupt service routine
    ...
    ; The time must be longer than 16us.
    B0BCLR    FT0IRQ    ; Reset T0IRQ
  }

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

  RETI        ; Exit interrupt vector

```


6.8 TC0 INTERRUPT OPERATION

When the TC0C counter overflows, the TC0IRQ will be set to "1" no matter the TC0IEN is enable or disable. If the TC0IEN and the trigger event TC0IRQ is set to be "1". As the result, the system will execute the interrupt vector. If the TC0IEN = 0, the trigger event TC0IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the TC0IRQ is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

➤ Example: TC0 interrupt request setup.

```

B0BCLR    FTC0IEN    ; Disable TC0 interrupt service
B0BCLR    FTC0ENB    ; Disable TC0 timer
MOV       A, #20H    ;
B0MOV     TC0M, A     ; Set TC0 clock = Fcpu / 64
MOV       A, #74H    ; Set TC0C initial value = 74H
B0MOV     TC0C, A     ; Set TC0 interval = 10 ms

B0BSET    FTC0IEN    ; Enable TC0 interrupt service
B0BCLR    FTC0IRQ    ; Clear TC0 interrupt request flag
B0BSET    FTC0ENB    ; Enable TC0 timer

B0BSET    FGIE       ; Enable GIE

```

➤ Example: TC0 interrupt service routine.

```

ORG       8           ; Interrupt vector
JMP      INT_SERVICE

INT_SERVICE:

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

B0BTS1   FTC0IRQ    ; Check TC0IRQ
JMP      EXIT_INT    ; TC0IRQ = 0, exit interrupt vector

B0BCLR   FTC0IRQ    ; Reset TC0IRQ
MOV      A, #74H    ;
B0MOV    TC0C, A     ; Reset TC0C.
...           ; TC0 interrupt service routine
...

EXIT_INT:

...           ; Pop routine to load ACC and PFLAG from buffers.

RETI      ; Exit interrupt vector

```

6.9 MULTI-INTERRUPT OPERATION

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

| Interrupt Name | Trigger Event Description |
|-----------------------|----------------------------------|
| P00IRQ | P0.0 trigger controlled by PEDGE |
| T0IRQ | T0C overflow |
| TC0IRQ | TC0C overflow |

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

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

```

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

INTP00CHK:
                B0BTS1        FP00IEN    ; Check INT0 interrupt request
                JMP            INTT0CHK    ; Check P00IEN
                B0BTS0        FP00IRQ    ; Jump check to next interrupt
                JMP            INTP00     ; Check P00IRQ
                B0BTS1        FT0IEN     ; Jump to INT0 interrupt service routine
                JMP            INTT0CHK    ; Check T0 interrupt request
INTT0CHK:
                B0BTS1        FT0IEN     ; Check T0IEN
                JMP            INTTC0CHK   ; Jump check to next interrupt
                B0BTS0        FT0IRQ     ; Check T0IRQ
                JMP            INTT0     ; Jump to T0 interrupt service routine
                B0BTS1        FTC0IEN    ; Check TC0 interrupt request
INTTC0CHK:
                JMP            INT_EXIT    ; Check TC0IEN
                B0BTS0        FTC0IRQ    ; Jump to exit of IRQ
                JMP            INTTC0     ; Check TC0IRQ
                B0BTS1        FTC0IEN    ; Jump to TC0 interrupt service routine
INT_EXIT:
                ...                ; Pop routine to load ACC and PFLAG from buffers.

                RETI                ; Exit interrupt vector

```

7 I/O PORT

7.1 I/O PORT MODE

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

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

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

| 0C5H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| P5M | - | - | P55M | P54M | P53M | P52M | P51M | P50M |
| Read/Write | - | - | R/W | R/W | R/W | R/W | R/W | R/W |
| After reset | - | - | 0 | 0 | 0 | 0 | 0 | 0 |

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

- * **Note:**
1. Users can program them by bit control instructions (*B0BSET*, *B0BCLR*).
 2. Port 0 is Input only port.
 3. Port 2 is shared with *XIN/LXIN* and *XOUT/LXOUT*.

➤ Example: I/O mode selecting

```

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

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

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

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

7.2 I/O PULL UP REGISTER

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

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

| 0E5H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| P5UR | - | - | P55R | P54R | P53R | P52R | P51R | P50R |
| Read/Write | - | - | W | W | W | W | W | W |
| After reset | - | - | 0 | 0 | 0 | 0 | 0 | 0 |

* **Note: Pull up Resistance of Port 0 is always existence.**

➤ Example: I/O Pull up Register

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

7.3 I/O PORT DATA REGISTER

| 0D0H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| P0 | - | - | - | - | - | - | - | P00 |
| Read/Write | - | - | - | - | - | - | - | R/W |
| After reset | - | - | - | - | - | - | - | 0 |

| 0D1H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| P1 | - | - | - | - | P13 | P12 | P11 | P10 |
| Read/Write | - | - | - | - | R/W | R/W | R/W | R/W |
| After reset | - | - | - | - | 0 | 0 | 0 | 0 |

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

| 0D5H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| P5 | - | - | P55 | P54 | P53 | P52 | P51 | P50 |
| Read/Write | - | - | R/W | R/W | R/W | R/W | R/W | R/W |
| After reset | - | - | 0 | 0 | 0 | 0 | 0 | 0 |

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

```

B0MOV    A, P0           ; Read data from Port 0
B0MOV    A, P1           ; Read data from Port 1
B0MOV    A, P5           ; Read data from Port 5
    
```

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

```

MOV      A, #0FFH       ; Write data FFH to all Port.
B0MOV    P1, A
B0MOV    P5, A
    
```

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

```

B0BSET   P1.0           ; Set P1.0 to be "1".
B0BCLR   P1.0           ; Set P1.0 to be "0".
    
```

8 TIMERS

8.1 WATCHDOG TIMER (WDT)

The watchdog timer (WDT) is a binary up counter designed for monitoring program execution. If the program goes into the unknown status by noise interference, WDT overflow signal raises and resets MCU. The instruction that clears the watchdog timer (" B0BSET FWDRST ") should be executed within a certain period. If an instruction that clears the watchdog timer is not executed within the period and the watchdog timer overflows, reset signal is generated and system is restarted.

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

Bit5 **WDRATE**: Watchdog timer rate select bit.

$$0 = F_{CPU} \div 2^{14}$$

$$1 = F_{CPU} \div 2^8$$

Bit6 **WDRST**: Watchdog timer reset bit.

0 = No reset
1 = clear the watchdog timer's counter.
(The detail information is in watchdog timer chapter.)

Bit7 **WTCKS**: Watchdog clock source select bit.

0 = F_{CPU}
1 = internal RC low clock.

Watchdog timer overflow table.

| WTCKS | WTRATE | CLKMD | Watchdog Timer Overflow Time |
|-------|--------|-------|---|
| 0 | 0 | 0 | $1 / (f_{cpu} \div 2^{14} \div 16) = 293 \text{ ms}$, Fosc=3.58MHz |
| 0 | 1 | 0 | $1 / (f_{cpu} \div 2^8 \div 16) = 500 \text{ ms}$, Fosc=32768Hz |
| 0 | 0 | 1 | $1 / (f_{cpu} \div 2^{14} \div 16) = 32.7\text{s}$, Fosc=32KHz@3V |
| 0 | 1 | 1 | $1 / (f_{cpu} \div 2^8 \div 16) = 0.5\text{s}$, Fosc=32KHz@3V |
| 1 | - | - | $1 / (16\text{K} \div 512 \div 16) \sim 0.5\text{s @}3\text{V}$ |

* **Note:** The watchdog timer can be enabled or disabled by the code option.

Watchdog timer application note is as following.

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

Main:

```
... ; Check I/O.
... ; Check RAM
```

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

Correct:

```
BOBSET FWDRST ; I/O and RAM are correct. Clear watchdog timer and
; execute program.
; Only one clearing watchdog timer of whole program.
```

```
...
CALL SUB1
CALL SUB2
...
...
...
JMP MAIN
```

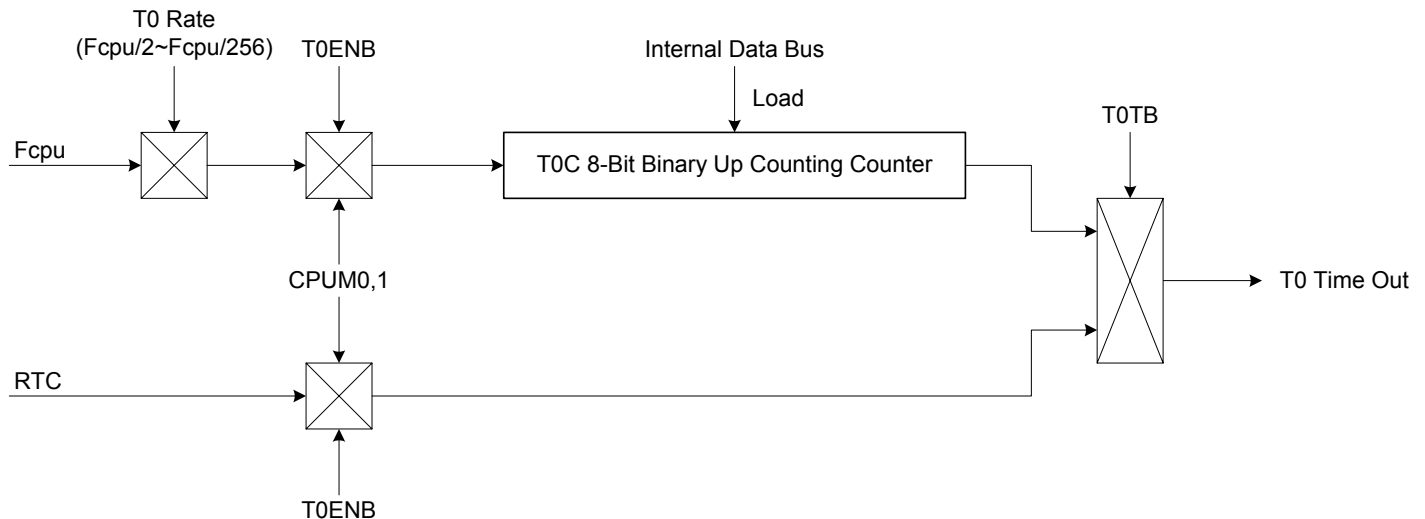
8.2 TIMER 0 (T0)

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 T0TB=1.**
- ☞ **Green mode wakeup function:** T0 can be green mode wake-up time as T0ENB = 1. System will be wake-up by T0 time out.



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

8.2.2 T0M MODE REGISTER

| 0D8H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|---------|---------|---------|-------|-------|-------|-------|
| T0M | T0ENB | T0rate2 | T0rate1 | T0rate0 | - | - | TC0GN | T0TB |
| Read/Write | R/W | R/W | R/W | R/W | - | - | R/W | R/W |
| After reset | 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 [6:4] **TORATE[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:** TORATE 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.

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

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

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

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: T0C is not available in RTC mode. The T0 interval time is fixed at 0.5 sec.**

8.2.4 T0 TIMER OPERATION SEQUENCE

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

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

```

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

```

☞ **Set T0 timer rate.**

```

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

```

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

```

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

```

☞ **Set T0 interrupt interval time.**

```

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

```

☞ **Set T0 timer function mode.**

```

B0BSET    FT0IEN    ; Enable T0 interrupt function.

```

☞ **Enable T0 timer.**

```

B0BSET    FT0ENB    ; Enable T0 timer.

```


8.3.2 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] | TC0 Clock |
|---------------|------------|
| 000 | Fcpu / 256 |
| 001 | Fcpu / 128 |
| 010 | Fcpu / 64 |
| 011 | Fcpu / 32 |
| 100 | Fcpu / 16 |
| 101 | Fcpu / 8 |
| 110 | Fcpu / 4 |
| 111 | Fcpu / 2 |

- 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 TC0GN FLAGS

| 0D8H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|---------|---------|---------|-------|-------|-------|-------|
| T0M | TOENB | T0rate2 | T0rate1 | T0rate0 | - | - | TC0GN | T0TB |
| Read/Write | R/W | R/W | R/W | R/W | - | - | R/W | R/W |
| After reset | 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 [6:4] **T0RATE[2:0]**: T0 internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
...
110 = fcpu/4.
111 = fcpu/2.

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

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

8.3.4 TC0C COUNTING REGISTER

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

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

The equation of TC0C initial value is as following.

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

- **Example:** To set 10ms interval time for TC0 interrupt. TC0 clock source is Fcpu (TC0KS=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.

| 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 |

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.

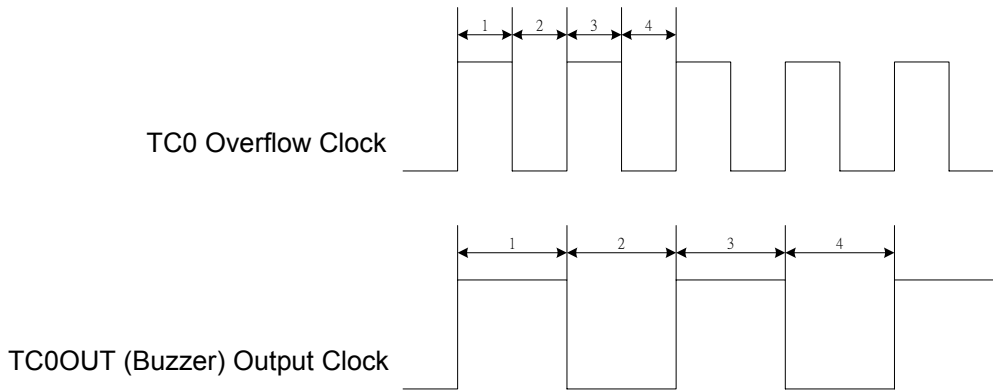
$$TC0R \text{ initial value} = 256 - (TC0 \text{ interrupt interval time} * \text{input clock})$$

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

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

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 $TC0RATE2-TC0RATE1 = 110$. $TC0C = TC0R = 131$.**

```

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  FALOAD0          ; 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 auto-load mode.**

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

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

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

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

☞ **Set TC0 timer function mode.**

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

☞ **Enable TC0 timer.**

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

9 LCD DRIVER

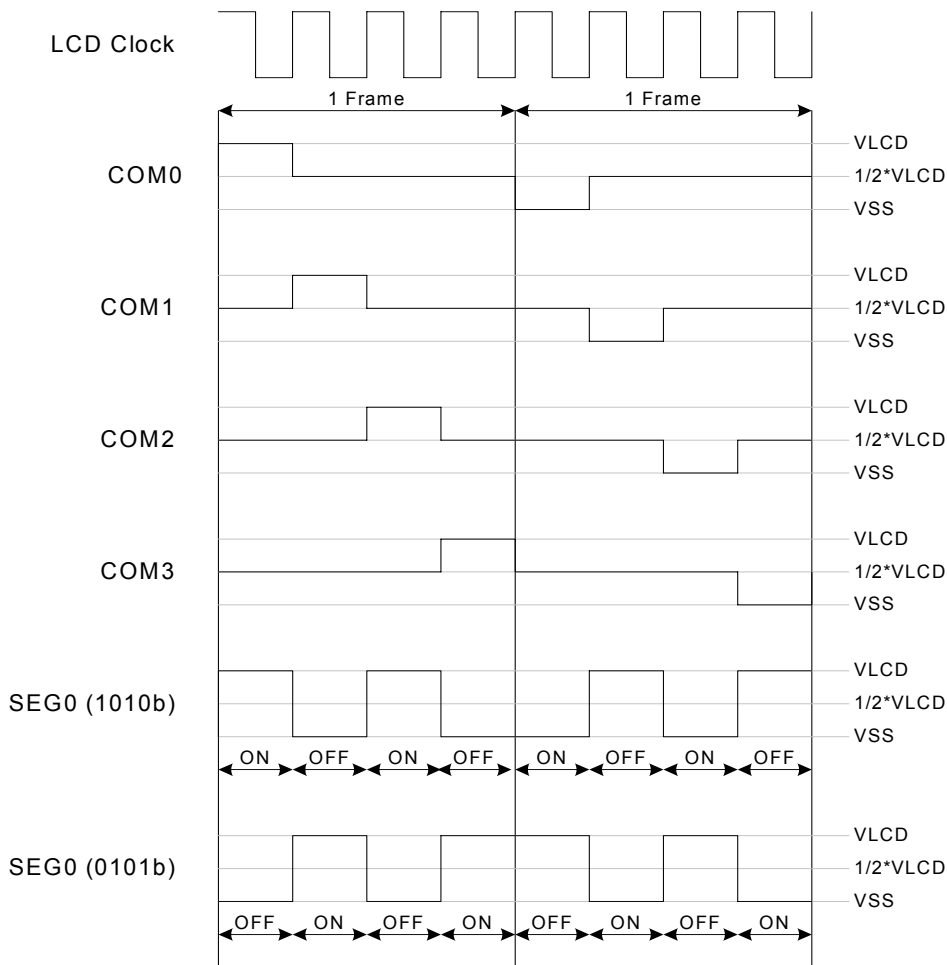
LCD driver includes R-type and C-type structures with 4 common pins and 12segment pins in the SN8P1937. The LCD scan timing is 1/4 duty with 1/2 bias or 1/3 bias structure, all support in R-type and C-type mode to yield 48 dots LCD driver. LCD power and bias voltage can be adjusted by internal / external bias circuit in R-type LCD driver, and adjusted by setting internal charge pump in C-type LCD driver.

9.1 LCD TIMING

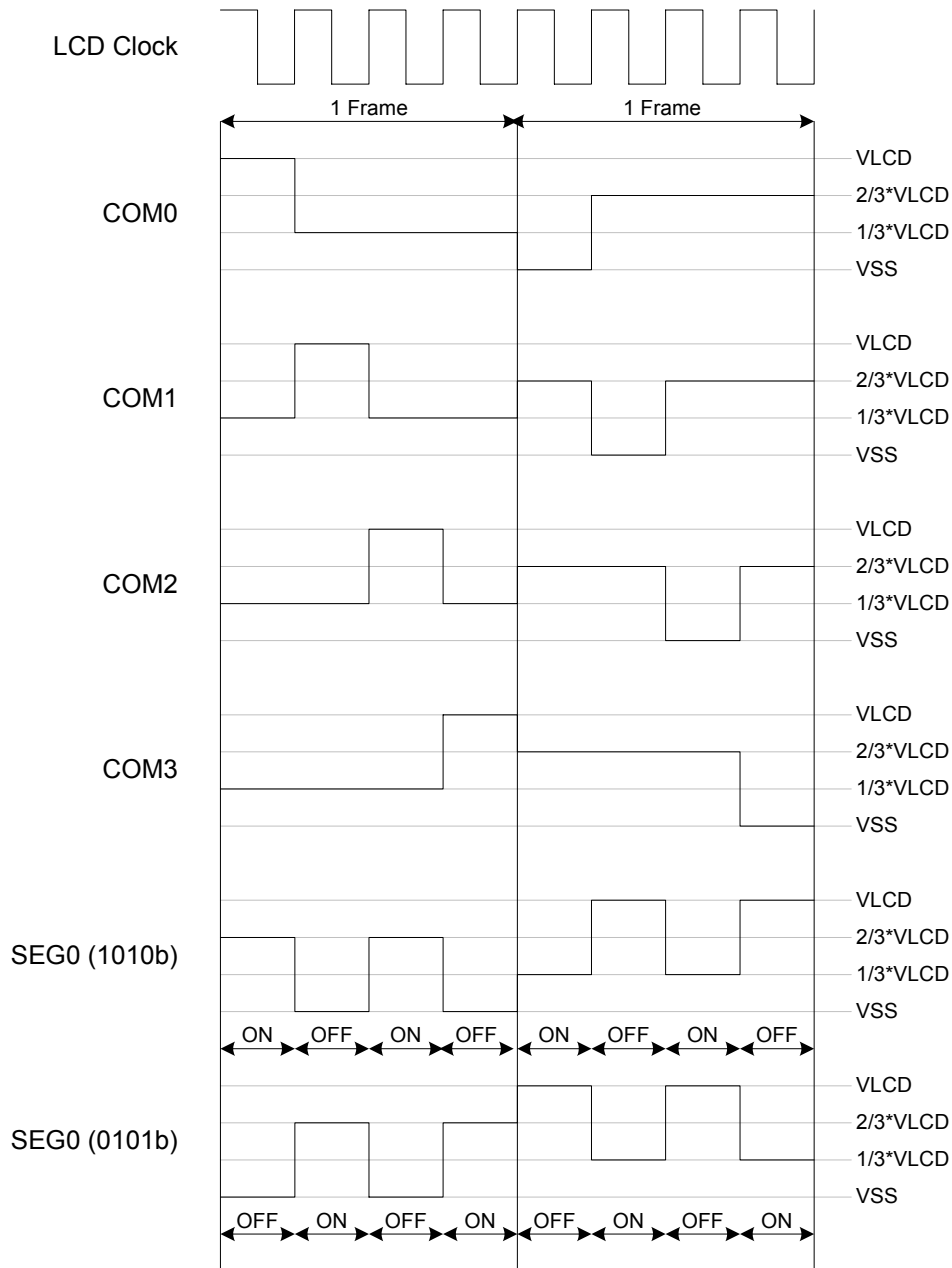
LCD Timing Table

| Code Option "High_Clk = IHRC or 4MHz" | | | | | |
|---------------------------------------|------------------|---------|---------------|-------------|-----------------------------|
| LCDCLK | LCD clock source | LCDRATE | LCD Clock | | Frame = LCD clock/4 |
| 0 | Fhosc (4M X'tal) | X | $4M/2^{14} =$ | 244.14Hz@4M | $244.14/4 = 61.03\text{Hz}$ |
| 0 | Fhosc (IHRC) | X | $4M/2^{14} =$ | 244.14Hz@4M | $244.14/4 = 61.03\text{Hz}$ |
| 1 | Fosc | 0 | $32K/64 =$ | 500Hz@3V | $500/4 = 125\text{Hz}$ |
| 1 | Fosc | 1 | $32K/32 =$ | 1000Hz@3V | $1000/4 = 250\text{Hz}$ |
| 1 | Fosc | 0 | $64K/64 =$ | 1000Hz@5V | $1000/4 = 250\text{Hz}$ |
| 1 | Fosc | 1 | $64K/32 =$ | 2000Hz@5V | $2000/4 = 500\text{Hz}$ |

Note: When system in "IHRC_RTC" mode, LCD frame rate is always fixed in 64Hz. Bit LCDCLK is set only for C-Type Charge pump clock.



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 clock = External Clock / 2¹⁴, Frame rate = LCD clock / 4
 Ex: High clock = 4M, LCD clock = 244.14 Hz , Frame rate = 244.14/4 = 61.03 Hz.
 High clock = 3.58M, LCD clock = 218.51 Hz , Frame rate = 218.51/4 = 54.62 Hz.
1 = LCD clock = Internal RC / 32(LCDRATE=1) or Internal RC = 64 (LCDRATE=0),
 Frame rate = LCD clock/4.
- Bit1 **LCDRATE:** LCD clock rate control when LCD CLK=1.
0 = LCD clock= Internal RC/ 64.
1 = LCD clock= Internal RC/ 32.
- Bit2 **LCDBIAS:** LCD Bias Selection Bit.
0 = LCD Bias is 1/3 Bias.
1 = LCD Bias is 1/2 Bias.
- Bit3 **LCDENB:** LCD driver enable control bit.
0 = Disable.
1 = Enable.
- Bit4 **LCDDTYPE:** R-Type / C-Type LCD driver control bit.
0 = R-Type.
1 = C-Type.
- Bit5 **LCDBNK:** LCD blank control bit.
0 = Normal display.
1 = All of the LCD dots off.
- Bit[7:6] **LCDREF[1:0]:** Resistance selection for LCD Bias Voltage-division in R-Type LCD driver.

| LCDREF[1:0] | Resistance |
|-------------|------------|
| 00 | 400k |
| 01 | 200k |
| 10 | 100k |
| 11 | 50k |

R-Type and C-Type LCD driver control:

| LCDENB | LCDDTYPE | R-Type Driver | C-Type Driver | Note |
|--------|----------|---------------|---------------|---------------------|
| 0 | 0 | Disable | Disable | LCD Charge Pump off |
| 0 | 1 | Disable | Disable | LCD Charge Pump on |
| 1 | 0 | Enable | Disable | LCD Charge Pump off |
| 1 | 1 | Disable | Enable | LCD Charge Pump on |

9.3 LCDM2 REGISTER

| 08AH | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------------|-------|-------|-------|---------|---------|-------|-------|-------|
| LCDM2 | - | - | BGM | LCDCPK1 | LCDCPK0 | VCP2 | VCP1 | VCP0 |
| R/W | - | - | R/W | R/W | R/W | R/W | R/W | R/W |
| After Reset | - | - | 0 | 0 | 0 | 0 | 1 | 1 |

Bit[0:2] **VCP[0:2]:** C-Type LCD charge pump output voltage.
(When LCDENB = 1 and LCDTYPE = 1, LCD charge pump start pumping)

| VCP[2:0] | In 1/3 bias Condition | | | | In 1/2 bias Condition | | | |
|----------|-----------------------|--------|--------|------|-----------------------|--------|--------|------|
| | V1 | V2 | V3 | VLCD | V1 | V2 | V3 | VLCD |
| 000 | 0V | 0.900V | 1.800V | 2.7V | 0V | 1.350V | 1.350V | 2.7V |
| 001 | 0V | 0.933V | 1.866V | 2.8V | 0V | 1.400V | 1.400V | 2.8V |
| 010 | 0V | 0.966V | 1.933V | 2.9V | 0V | 1.450V | 1.450V | 2.9V |
| 011 | 0V | 1.000V | 2.000V | 3.0V | 0V | 1.500V | 1.500V | 3.0V |
| 100 | 0V | 1.033V | 2.066V | 3.1V | 0V | 1.550V | 1.550V | 3.1V |
| 101 | 0V | 1.066V | 2.133V | 3.2V | 0V | 1.600V | 1.600V | 3.2V |
| 110 | 0V | 1.100V | 2.200V | 3.3V | 0V | 1.650V | 1.650V | 3.3V |
| 111 | 0V | 1.133V | 2.266V | 3.4V | 0V | 1.700V | 1.700V | 3.4V |

Bit[4:3] **LCDCPK[1:0]:** LCD charge pump clock selection.

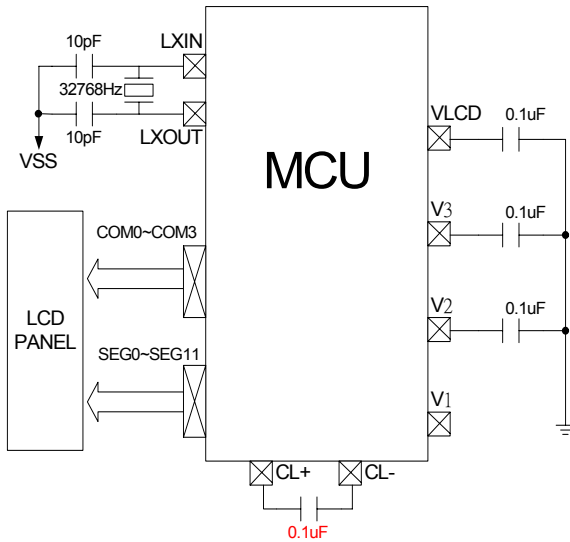
| LCDCPK [1:0] | LCDCCLK | Charge-pump Clock Source |
|--------------|---------|------------------------------|
| 00 | 1 | Fosc / 1 = 32kHz |
| 01 | 1 | Fosc / 2 = 16kHz |
| 10 | 1 | Fosc / 8 = 4kHz |
| 11 | 1 | Fosc / 32 = 1kHz |
| 00 | 0 | Fhosc / 64 = 62.5kHz |
| 01 | 0 | Fhosc / 128 = 31.2kHz |
| 10 | 0 | Fhosc / 512 = 7.8kHz |
| 11 | 0 | Fhosc / 2048 = 1.9kHz |

Bit5 **BGM:** Band Gap mode selection.
0 = not allowed.
1 = High precision Band Gap mode for accurate VLCD application.
(BGM must be set "1")

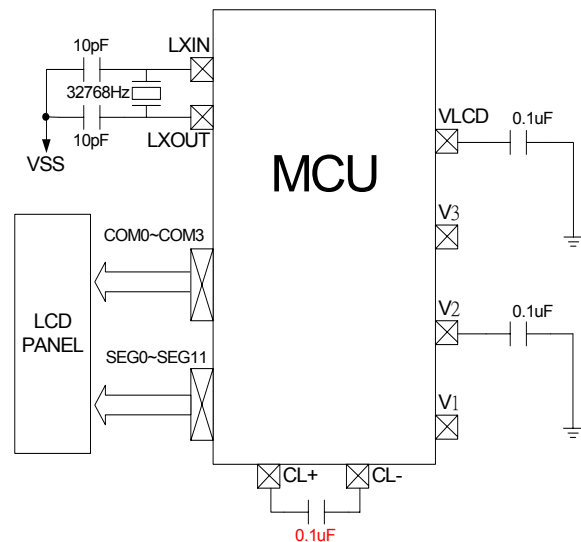
- * **Note1:** In IHRC-RTC mode, Fosc is from 32768 Hz crystal; in other mode, Fosc is from ILRC. (ILRC: 32kHz@3V, 64kHz@5V)
- * **Note2:** In R-Type LCD mode, three external pads of V1/V2/V3 are available for fine tune the LCD bias voltage and current.
- * **Note3:** in C-Type LCD mode, three external pads of VLCD/V2/V3 must connect 0.1uf to Gnd for voltage pumping.
- * **Note4:** BGM must be set "1" for all C-Type LCD application.
- * **Note5:** Before into Power down mode, please clear bits of LCDENB, LCDTYPE and BGRENB for function disable power saving.

9.4 C-TYPE LCD DRIVER MODE

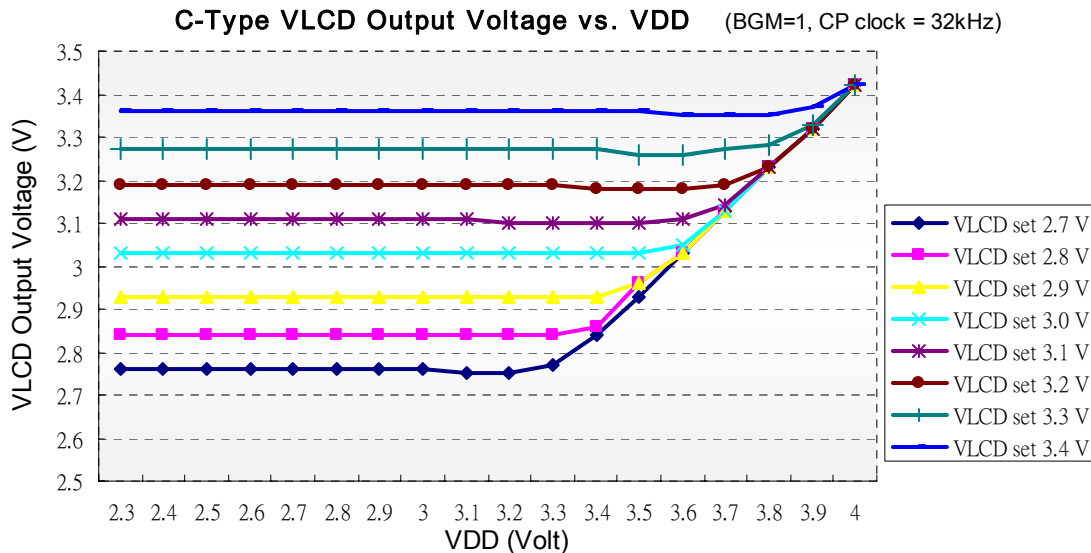
The LCD C-type driver mode is support 1/3 and 1/2 bias LCD panel. The LCD power (VLCD) is supplied by internal LCD charge-pump. The power consumption of C-type mode is less than R-type, because no additional DC bias circuit expenses power current. The charge-pump voltage level is following VLCD voltage. V2 is the charge pump source which level is 1/3*VLCD. V3 is 2 times of V2 by charge pump which level is 2/3*VLCD. In C-type LCD mode, the LCDTYPE bit of LCDM1 register must be "1". The following are shown the 1/3 and 1/2 bias C-Type LCD application circuit and VLCD output voltage chart.



1/3 Basic C-type LCD Application Circuit



1/2 Basic C-type LCD Application Circuit

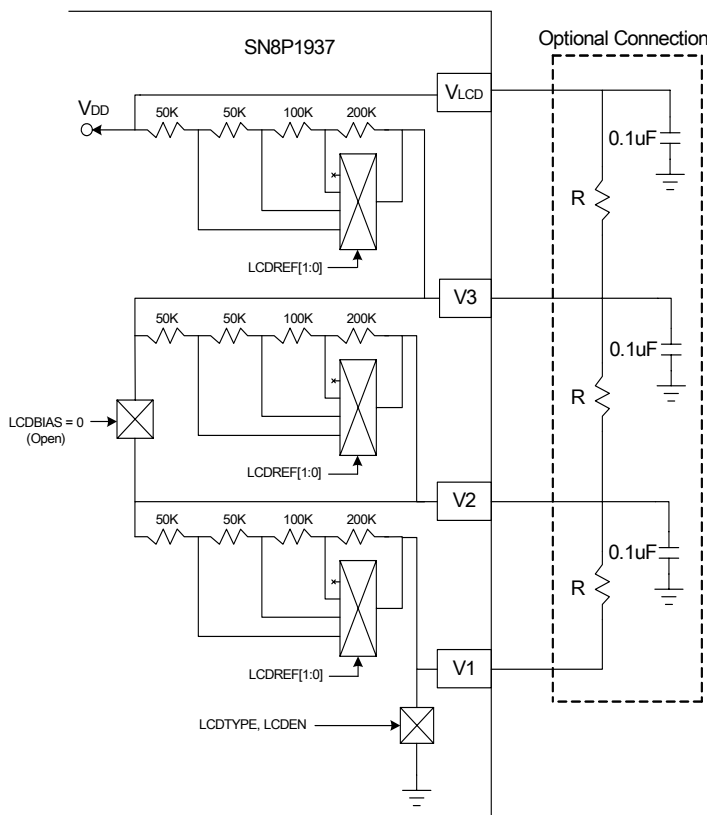


- * **Note1:** In C-type mode, connect a 0.1uF capacitor between CL+ and CL- pins. The 0.1uF capacitor also connect pin VLCD/V3/V2 to VSS.
- * **Note2:** Increasing the LCD charge pump clock via setting LCDCPK [1:0] for larger LCD panel size application.
- * **Note3:** VLCD 3.0V setting in C-Type LCD mode is applied to VDD < 3.5V power supply.
- * **Note4:** VLCD output voltage can be set from 2.7V to 3.4V and with ±0.2V accuracy.

9.5 R-TYPE LCD DRIVER MODE

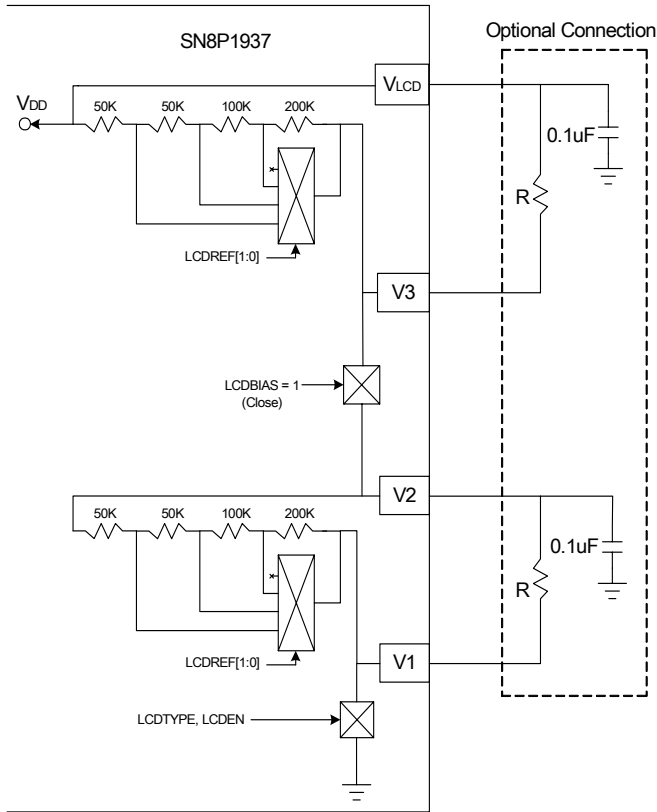
In LCD R-type driver, LCD power (VLCD) is auto connected to VDD via internal circuit. V3 and V2 bias voltage source from internal voltage division by selective resistors. The selective resistors are provided in difference resistance value for bias voltage division to adjusting LCD driving current and fit the difference LCD size application, which can be configured as the resistance value of 400k, 200k, 100k, and 50k, controlled by bit LCDREF [1:0]. Moreover, additional external resistors can be connected in parallel type with internal selective resistor via pin VLCD, V3, V2, and V1, to increase more driving current for large LCD application purpose. The following diagram shows the connection of 1/4 duty with 1/3 bias and 1/2 bias.

1/4 duty with 1/3 bias:



$$\text{LCD current consumption} = \frac{\text{VLCD}}{\left(\frac{50\text{k} \times R}{50\text{k} + R}\right) \times 3}, \quad \text{where LCDREF} = [11] \text{ and external parallel } R \text{ connected.}$$

1/4 duty with 1/2 bias:



LCD current consumption = $\frac{VLCD}{\left(\frac{50k \times R}{50k + R}\right) \times 2}$, where LCDREF= [11] and external parallel R connected.

- * **Note1:** LCDREF [1:0] function is disabled in C-Type LCD Driver mode.
- * **Note2:** In R-Type LCD driver mode, VLCD power is auto connected to VDD via internal circuit. Pin VLCD do not connect to any power source.
- * **Note3:** External resistor is optional connected for LCD size.

9.6 LCD RAM LOCATION

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

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

➤ **Example: Enable LCD function.**

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

```
BOBSET          FLCDENB          ; LCD driver.
```

10 IN SYSTEM PROGRAM ROM

10.1 OVERVIEW

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

10.2 ROMADRH/ROMADRL REGISTER

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

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

VPPCHK: VPP pin Programming Voltage. Check

0 = VPP's Voltage NOT reached 12.5V. Can't program ISP ROM

1 = VPP's Voltage reached 12.5V. Can program ISP ROM

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

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

10.3 ROMDAH/ROMADL REGISTERS

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

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

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

10.4 ROMCNT REGISTERS and ROMWRT INSTRUCTION

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

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

| Fcpu | ROMCNT | Programming Time |
|-------|--------|------------------|
| 1MIPs | 6 | 1ms |

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

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

10.5 ISP ROM ROUTINE EXAMPLE

➤ Example :

```

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

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

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

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

```

11 Regulator, PGIA and ADC

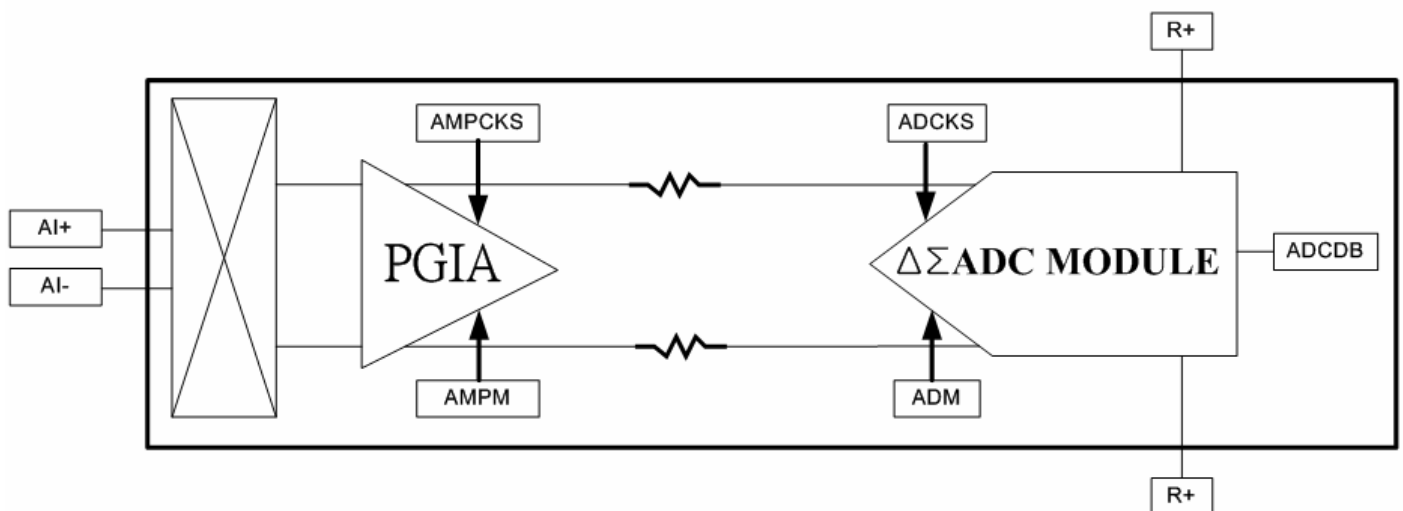
11.1 OVERVIEW

The SN8P1937 has a built-in Voltage Regulator to support a stable voltage 2.4V from pin AVDDR and 1.5V from pin AVE+ with maximum 10mA current driving capacity. The AVDDR provides stable voltage for internal circuits (PGIA, ADC) and external sensor (load cell or thermistor). The SN8P1937 series also integrated $\Delta \Sigma$ Analog-to-Digital Converters (ADC) to achieve 16-bit performance and up to 65536-step resolution. The ADC has 1 different input channel modes: (1) One fully differential input. (2) Two single-ended inputs. This ADC is optimized for measuring low-level unipolar or bipolar signals in weight scale and medical applications. A very low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selectable gains of 1x, 16x, 32x, 64x, and 128x in the ADC to accommodate these applications.

11.2 ANALOG INPUT

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

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



Block Diagram of ADC module

11.3 Voltage Regulator

SN8P1937 is built in voltage regulators, which can provide a stable 2.4V (pin AVDDR) and 1.5V (pin AVE+) with maximum 10mA current driving capacity. Register CPM can enable or disable AVDDR, AVE+ and ACM output voltage. Because the power of PGIA and ADC are came from AVDDR, turn on AVDDR (AVDDRENB = 1) first before enabling PGIA and ADC. The AVDDR voltage was regulated from VDD.

11.3.1 CPM-Charge Pump Mode Register

| 095H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|--------|----------|--------|-------|--------|-------|-------|-------|
| CPM | ACMENB | AVDDRENB | AVESEL | AVENB | ACMSEL | - | - | - |
| R/W | R/W | R/W | R/W | R/W | R/W | - | - | - |
| After Reset | 0 | 0 | 0 | 0 | 0 | - | - | - |

Bit3: **ACMSEL**: ACM Voltage selection bit.

- 0 = ACM output 0.4V.
- 1 = Reserve.

Bit4: **AVENB**: AVE+ voltage output control bit.

- 0 = Disable AVE+ output Voltage.
- 1 = Enable AVE+ output Voltage.

Bit5: **AVESEL**: AVE+ voltage selection control bit.

- 0 = AVE+ output 1.5V.
- 1 = Reserve.

Bit6: **AVDDRENB**: Regulator (AVDDR) voltage Enable control bit.

- 0 = Disable Regulator and AVDDR Output voltage 2.4V.
- 1 = Enable Regulator and AVDDR Output voltage 2.4V.

Bit7: **ACMENB**: Analog Common Mode (ACM) voltage Enable control bit.

- 0 = Disable Analog Common Mode and ACM Output voltage 0.4V.
- 1 = Enable Analog Common Mode and ACM Output voltage 0.4V.

- * **Note1: Band Gap Reference voltage must be enable (FBRGEN), before following function accessing: (Reference AMPM register for detail information)**
 - (1) Regulators of AVDDR, AVE+ and ACM.
 - (2) PGIA Function.
 - (3) ADC Function.
 - (4) Low Battery Detection Function.
- * **Note2: PGIA and ADC can work in slow mode, but AMPCKS register value must be reassigned.**
- * **Note3: All current consumptions from AVE+ (ex. Load cell) or AVDDR will NOT double.**

11.4 PGIA -Programmable Gain Instrumentation Amplifier

SN8P1937 includes a low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selection gains of 1x, 16x, 32x, 64x, and 128x by register AMPM. The PGIA also provides two types channel selection mode: (1) One fully differential input (2) Two single-ended inputs, it was defined by register AMPCHS.

11.4.1 AMPM- Amplifier Mode Register

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

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

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

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

Bit[5:4] **FDS [1:0]**: Chopper Low frequency setting.
11 = for all applications.

Bit6: **BGRENB**: Band Gap Reference voltage enable control bit.
0 = Disable Band Gap Reference Voltage
1 = Enable Band Gap Reference Voltage

Bit7: **CHPENB**: PGIA Chopper function enable control bit.
0 = Disable PGIA chopper.
1 = Enable PGIA chopper.

- * **Note: When selected gain is 1x, PGIA can be disabled (AMPENB=0) for power saving.**
- * **Note: Set FDS [1:0] = "11" for all applications.**

11.4.2 AMPCKS- PGIA CLOCK SELECTION

| 092H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|---------|---------|---------|
| AMPCKS | - | - | - | - | - | AMPCKS1 | AMPCKS1 | AMPCKS0 |
| R/W | - | - | - | - | - | W | W | W |
| After Reset | - | - | - | - | - | 0 | 0 | 0 |

Bit[2:0] **AMPCKS[2:0]**: register sets the PGIA Chopper working clock. The suggestion Chopper clock is 31.25K Hz.@ 4MHz.

PGIA Clock= Fcpu / 32 / (2^AMPCKS)

Refer to the following table for AMPCKS [2:0] register value setting in different Fosc frequency.

| AMPCKS2 | AMCKS1 | AMPCKS0 | High Clock | | | |
|---------|--------|---------|------------|--------|---------|--------|
| | | | 2M | 3.58M | 4M/IHRC | 8M |
| 0 | 0 | 0 | 15.62K | 27.96K | 31.25K | 62.5K |
| 0 | 0 | 1 | 7.81K | 13.98K | 15.62K | 31.25K |
| 0 | 1 | 0 | 3.90K | 6.99K | 7.812K | 15.62K |
| 0 | 1 | 1 | 1.95K | 3.49K | 3.90K | 7.81K |
| 1 | 0 | 0 | 976Hz | 1.748K | 1.95K | 3.90K |
| 1 | 0 | 1 | 488Hz | 874Hz | 976Hz | 1.95K |
| 1 | 1 | 0 | 244Hz | 437Hz | 488Hz | 976Hz |
| 1 | 1 | 1 | 122Hz | 218Hz | 244Hz | 488Hz |

* **Note: In general application, set PGIA Chopper working clock of 31.25K Hz, but set clock to 250Hz when system clock is 32768Hz crystal or in Internal Low clock mode.**

11.4.3 AMPCHS-PGIA CHANNEL SELECTION

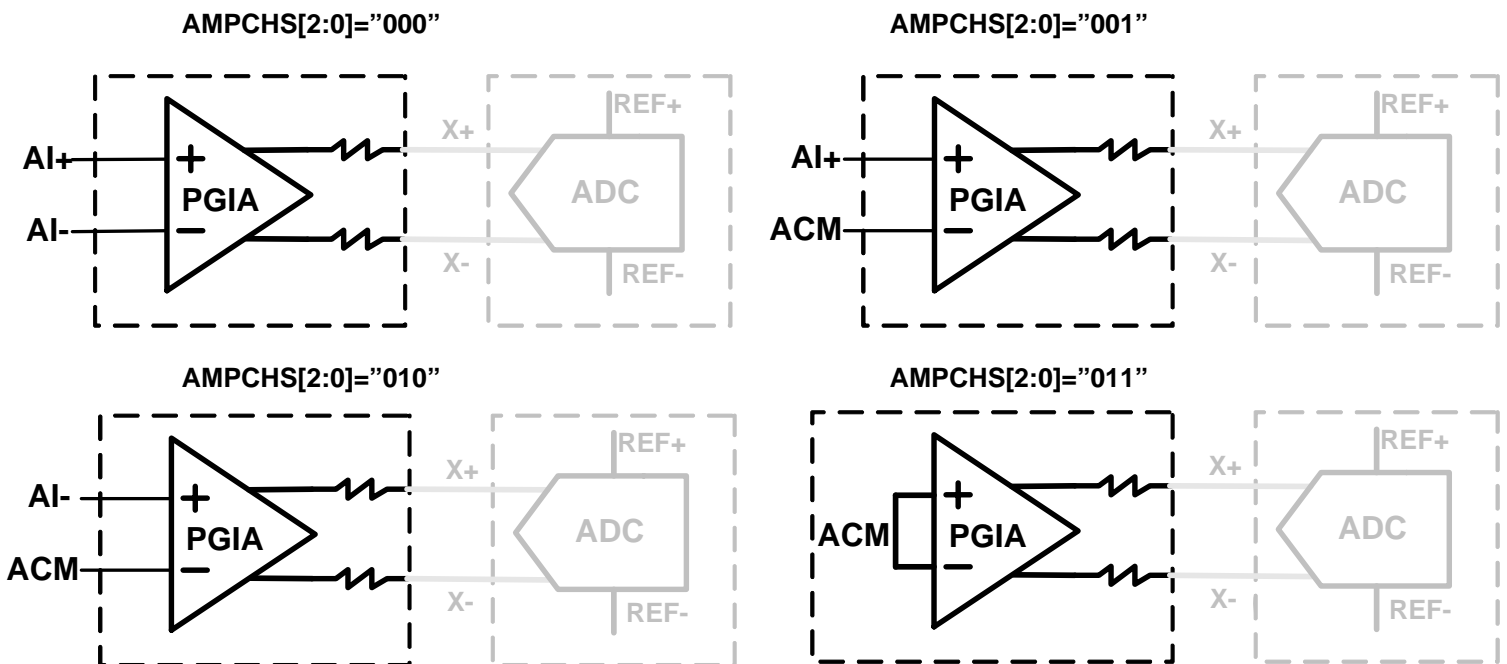
| 091H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| AMPCHS | - | - | - | - | - | CHS2 | CHS1 | CHS0 |
| R/W | - | - | - | - | - | R/W | R/W | R/W |
| After Reset | - | - | - | - | - | 0 | 0 | 0 |

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

PGIA Channel Selection Table:

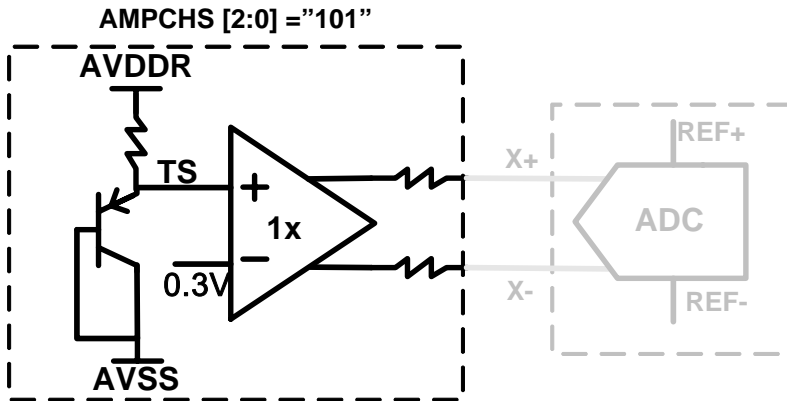
| CHS [2:0] | Selected Channel | V (X+, X-) Output | Input-Signal Type |
|-----------|--------------------|--|-------------------|
| 000 | AI+, AI- | $V (AI+, AI-) \times PGIA \text{ Gain}$ | Differential |
| 001 | AI+, ACM | $V (AI+, ACM) \times PGIA \text{ Gain}$ | Single-ended |
| 010 | AI-, ACM | $V (AI-, ACM) \times PGIA \text{ Gain}$ | Single-ended |
| 011 | ACM, ACM | $V (ACM, ACM) \times PGIA \text{ Gain}$ | Input-Short |
| 100 | Reserved | - | - |
| 101 | Temperature Sensor | $V (V_{TS}, 0.6V) \times 1$ | N/A |
| 110 | Voltage Detection | $V (3/16V_{DD}, 2/16V_{DD}) \times PGIA \text{ Gain}$ $V (3/16V_{LDD}, 2/16V_{LDD}) \times PGIA \text{ Gain}$ | Differential |
| 110,111 | Reserved | - | - |

- * **Note 1:** $V (AI+, AI-) = (AI+ \text{ voltage} - AI- \text{ voltage})$.
- * **Note 2:** $V (AI-, ACM) = (AI- \text{ voltage} - ACM \text{ voltage})$.
- * **Note 3:** The purpose of Input-Short mode is only for PGIA offset testing.



11.4.4 Temperature Sensor (TS)

In applications, sensor characteristic might change in different temperature also. To get the temperature information, SN8P1937 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 0.3V.
- * **Note 3:** The Temperature Sensor was just a reference data not real air temperature. For precision application, please use external thermistor sensor.

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

Example:

| Temperature | V(TS) | V(REF+,REF-) | ADC output |
|-------------|--------|--------------|------------|
| 15 | 0.769V | 0.6V | 25601 |
| 25 | 0.754V | 0.6V | 24782 |
| 35 | 0.738V | 0.6V | 23908 |

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

- * **Note 1:** The V(TS) voltage and temperature curve of each chip might different. Calibration in room temperature is necessary when application temperature sensor.
- * **Note 2:** 1.5mV/C was typical temperature parameter only sensor, every single chip was different to each other.

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

```

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

@ACM_Enable:
XB0BCLR      FACMSEL      ; Set ACM output 0.4V.
XB0BSET      FACMENB      ; Enable ACM Voltage=0.4v

@AVDDR_Enable:
XB0BSET      FAVDDRENB    ; Enable AVDDR Voltage=2.4V

@AVE_Enable:
XB0BCLR      FAVESEL      ; Set AVE+ output 1.5V
XB0BSET      FAVENB       ; Enable AVE+ Voltage

@PGIA_Init:
MOV          A, #11110110B
XB0MOV       AMPM, A      ; Enable Band Gap, Set :FDS="11" and PGIA Gain=128
MOV          A, #00000000B
XB0MOV       AMPCKS, A    ; Set AMPCKS="000" for PGIA working clock=31.2k@4MHz
MOV          A, #00h
XB0MOV       AMPCHS, A    ; Selected PGIA differential input channel= AI+, AI-

@PGIA_Enable:
XB0BSET      FAMPENB      ; Enable PGIA function
...          ; V (X+, X-) Output = V (AI+, AI-) x 128

```

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

Example: PGIA channel change:

```

@PGIA_Init:
MOV      A, #11110110B
XB0MOV  AMPM, A      ; Enable Band Gap, Set :FDS="11" and PGIA Gain=128
MOV      A, #00000000B
XB0MOV  AMPCKS, A    ; Set AMPCKS="000" for PGIA working clock=31.2K@4MHz
MOV      A, #00h
XB0MOV  AMPCHS, A    ; Selected PGIA differential input channel= AI+, AI-

@PGIA_Enable:
XB0BSET FAMPENB      ; Enable PGIA function
...      ; V (X+, X-) Output = V (AI+, AI-) x 128

@PGIA_single-end:
MOV      A, #11110111B ;Don't Disable PGIA when change PGIA Channel.
XB0MOV  AMPM, A      ; Enable Band Gap, Set :FDS="11" and PGIA Gain=128
MOV      A, #00000001B
XB0MOV  AMPCHS, A    ; Selected PGIA as Single-end channel
...      ; V (X+, X-) Output = V(AI+,ACM) x 128

@PGIA_TS:
MOV      A, #11111111B ;Don't Disable PGIA when change PGIA Channel.
XB0MOV  AMPM, A      ; Enable Band Gap, Set :FDS="11" and PGIA Gain=1x
MOV      A, #00000101B
XB0MOV  AMPCHS, A    ; Selected PGIA as Temperature Sensor channel
...      ; V (X+, X-) Output = V (TS, 0.3) x 1

```

11.5 16-Bit ADC

11.5.1 ADCM- ADC Mode Register

| 093H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| ADCM | ADG2 | ADG1 | ADG0 | RVS | IRVS | DTENB | DTSEL | ADCEN |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| After Reset | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

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

Bit1: **DTSEL**: Voltage Detection Source Selection bit.
0 = Selection VDD as Voltage Detection source.
1 = Selection VLCD as Voltage Detection source.

Bit2: **DTENB**: Detection Function Enable bit.
0 = Selection ADC as normal operation from X+, X-.
1 = Enable ADC as VDD or VLCD voltage detect function.

Bit3: **IRVS**: Internal Reference Voltage Selection.
0 = Internal Reference Voltage V(REF+,REF-) is AVE+/5 (When AVE+=1.5V, V(REF+,REF-)=0.3V).
1 = Internal Reference Voltage V(REF+,REF-) is AVE+/2.5. (When AVE+=1.5V, V(REF+,REF-)=0.6V).

Bit4: **RVS**: ADC Reference Voltage Selection bit.
0 = Selection ADC Reference voltage from External reference R+, R-.
1 = Selection ADC Reference voltage from Internal reference.

Bit[7:5]: **ADG[2:0]**: ADC Gain Selection Table in following table.

| ADG[2:0] | ADC Gain |
|----------|----------|
| 0xx | 1x |
| 100 | 2x |
| 101 | 4x |
| 110, 111 | Reserve |

ADC Reference Voltage Configuration Table:

| RVS | IRVS | DTENB | DTSEL | ADC Reference Voltage | | AD Channel Input | | Note |
|-----|------|-------|-------|-----------------------|------|------------------|-----------|---|
| | | | | REF+ | REF- | ADCIN+ | ADCIN- | |
| 0 | X | 0 | x | R+ | R- | X+ | X- | External Ref. Voltage |
| 1 | 0 | 0 | x | 0.6V | 0.3V | | | V (X+, X-) < 0.3V, (AVE+=1.5V) |
| 1 | 1 | 0 | x | 0.9V | 0.3V | | | V (X+, X-) < 0.6V, (AVE+=1.5V) |
| 0 | X | 1 | 0 | R+ | R- | VDD*3/16 | VDD*2/16 | ADC input = 1/16 VDD For Battery monitor. (AVE+=1.5V) |
| 1 | 0 | 1 | 0 | 0.6V | 0.3V | | | |
| 1 | 1 | 1 | 0 | 0.9V | 0.3V | | | |
| 0 | x | 1 | 1 | R+ | R- | VLCD*3/16 | VLCD*2/16 | ADC input = 1/16 VLCD For VLCD monitor |
| 1 | 0 | 1 | 1 | 0.6V | 0.3V | | | |
| 1 | 1 | 1 | 1 | 0.9V | 0.3V | | | |

- * **Note 1:** The ADC conversion data is combined with ADCDH and ADCDL register in 2's compliment with sign bit numerical format, and Bit ADCB15 is the sign bit of ADC data. Refer to following formula to calculate ADC conversion data value.
- * **Note 2:** The Internal Reference Voltage is divided from AVE+.

$(ADCIN+) > (ADCIN-) \Rightarrow ADCConversionData =$

$$+ \frac{[(ADCIN+) - (ADCIN-)]}{(REF+) - (REF-)} \times ADC_Gain \times 32768 \times \left(\frac{OSR - 1}{OSR} \right)^2$$

$(ADCIN+) < (ADCIN-) \Rightarrow ADCConversionData = (ADCIN+) > (ADCIN-) \Rightarrow ADCConversionData =$

$$- \frac{[(ADCIN+) - (ADCIN-)]}{(REF+) - (REF-)} \times ADC_Gain \times 32768 \times \left(\frac{OSR - 1}{OSR} \right)^2$$

Note 1: OSR value reference OSR Table (ADCHPENB=0) in chapter 11.5.4.

Note 2: ex: OSR [2:0] =111, OSR value=4096, ADC full counts = 32768x [(4096-1)/4096]^2 = 32752.

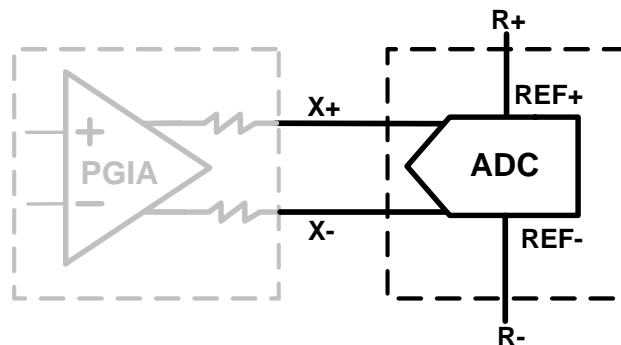
Note 3: Following Table shows ADC full counts vs. OSR setting.

| OSR[2:0] | ADC Full Counts | OSR[2:0] | ADC Full Counts |
|----------|-----------------|----------|-----------------|
| 000 | 30752 | 100 | 32640 |
| 001 | 31752 | 101 | 32704 |
| 010 | 32258 | 110 | 32736 |
| 011 | 32513 | 111 | 32752 |

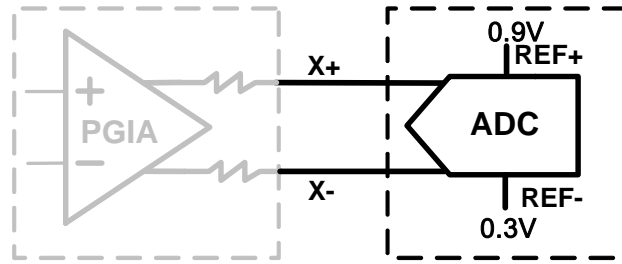
External and Internal Reference Circuit Table:

| External Ref. Circuit | Internal Reference Circuit | |
|-----------------------|----------------------------|-------------------|
| RVS=0 | RVS=1 | |
| | IRVS=1, AVE+=1.5V | IRVS=0, AVE+=1.5V |
| | | |

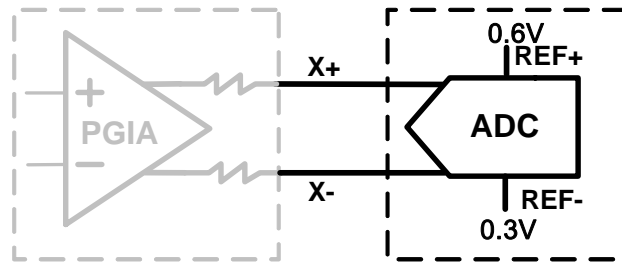
$ADCM=\#xxx0x0xxB$, $V(REF+, REF-) = V(R+, R-)$, ADC Reference Voltage from External R+,R-.



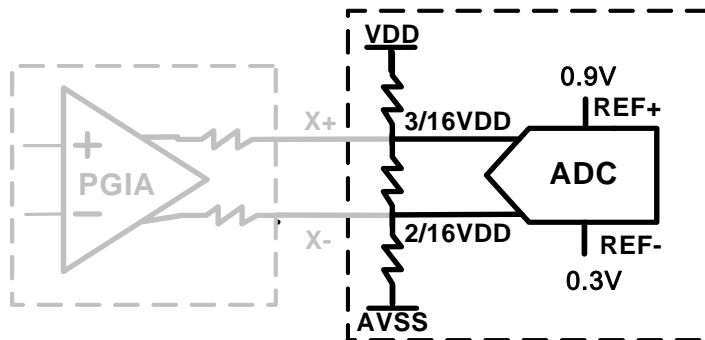
ADCM=#xxx1100xB, $V(\text{REF+}, \text{REF-}) = V(0.9\text{V}, 0.3\text{V}) = 0.6\text{V}$, ADC Reference Voltage from Internal 0.9V and 0.3V.



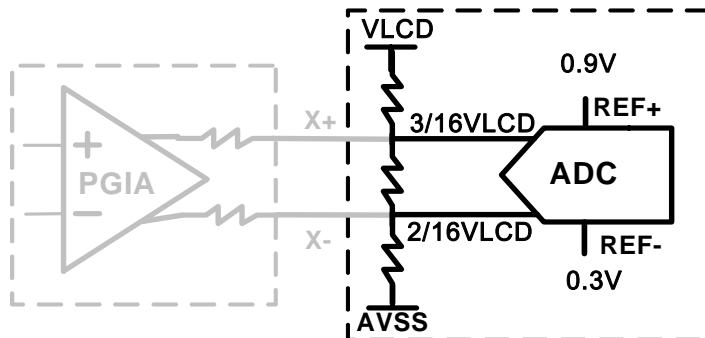
ADCM=#xxx1000xB, $V(\text{REF+}, \text{REF-}) = V(0.6\text{V}, 0.3\text{V}) = 0.3\text{V}$, ADC Reference Voltage from Internal 0.6V and 0.3V.



ADCM=#xxx1110xB, $V(\text{REF+}, \text{REF-}) = V(0.9\text{V}, 0.3\text{V}) = 0.6\text{V}$, ADC Reference Voltage from Internal 0.9V and 0.3V, and ADC output is Voltage measurement result.



ADCM=#xxx1111xB, $V(\text{REF+}, \text{REF-}) = V(0.9\text{V}, 0.3\text{V}) = 0.6\text{V}$, ADC Reference Voltage from Internal 0.9V and 0.3V, and ADC output is Voltage measurement result.



11.5.2 ADCKS- ADC Clock Register

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

Bit[7:0]: **ADCKS[7:0]**: Register sets the ADC working clock, the suggestion of ADC clock is 100kHz.

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

$$\text{ADC Clock} = (\text{Fosc} / (256 - \text{ADCKS} [7:0])) / 2$$

| ADCKS [7:0] | F _{osc} | ADC Working Clock |
|-------------|------------------|--------------------------------------|
| 246 | 4M | $(4\text{M} / 10) / 2 = 200\text{K}$ |
| 236 | 4M | $(4\text{M} / 20) / 2 = 100\text{K}$ |
| 243 | 4M | $(4\text{M} / 13) / 2 = 154\text{K}$ |
| 231 | 4M | $(4\text{M} / 25) / 2 = 80\text{K}$ |

| ADCKS [7:0] | F _{osc} | ADC Working Clock |
|-------------|------------------|--------------------------------------|
| 236 | 8M | $(8\text{M} / 20) / 2 = 200\text{K}$ |
| 216 | 8M | $(8\text{M} / 40) / 2 = 100\text{K}$ |
| 231 | 8M | $(8\text{M} / 25) / 2 = 160\text{K}$ |
| 206 | 8M | $(8\text{M} / 50) / 2 = 80\text{K}$ |

* **Note 1:** In general application, ADC working clock is 100K Hz.

11.5.3 ADC Data Register

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

| 098H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| ADC DL | ADCB7 | ADCB6 | ADCB5 | ADCB4 | ADCB3 | ADCB2 | ADCB1 | ADCB0 |
| R/W | R | R | R | R | R | R | R | R |
| After Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ADCDH [7:0]: Output high 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 |
|--|---------------|
| 0x7FF0 | 32752 |
| ... | ... |
| 0x4000 | 16384 |
| ... | ... |
| 0x1000 | 4096 |
| ... | ... |
| 0x0002 | 2 |
| 0x0001 | 1 |
| 0x0000 | 0 |
| 0xFFFF | -1 |
| 0xFFFE | -2 |
| ... | ... |
| 0xF000 | -4096 |
| ... | ... |
| 0xC000 | -16384 |
| ... | ... |
| 0x8010 | -32752 |

- * **Note 1:** ADC DL [7:0] and ADC DH [7:0] are both read only registers.
- * **Note 2:** The ADC conversion data is combined with ADC DH, ADC DL in 2's compliment with sign bit numerical format, and Bit ADC B15 is the sign bit of ADC data.
ADCB15=0 means data is Positive value, ADC B15=1 means data is Negative value.
- * **Note 3:** The Positive Full-Scale-Output value of ADC conversion is 0x7FFF.
- * **Note 4:** The Negative Full-Scale-Output value of ADC conversion is 0x8000H.
- * **Note 5:** Because of the ADC design limitation, the ADC Linear range is +29491 ~ -29491 (decimal). The MAX ADC output must keep inside this range.

11.5.4 DFM-ADC Digital Filter Mode Register

| 097H | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|----------|-------|-------|-------|-------|
| DFM | OSR2 | OSR1 | OSR0 | ADCHPENB | - | - | - | DRDY |
| R/W | R/W | R/W | R/W | R/W | - | - | - | R/W |
| After Reset | 0 | 0 | 0 | 1 | - | - | - | 0 |

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

Bit4: **ADCHPENB:** ADC Chopper Control Bit.
1 = ADC chopper enable.
0 = ADC chopper disable.

Bit[7:5]: **OSR[2:0]:** ADC Over Sample Rate Selection Bit.

OSR Table:

| OSR [2:0] | ADCHPENB = 1 | ADCHPENB = 0 |
|-----------|--------------|--------------|
| | OSR | |
| 000 | 128 | 32 |
| 001 | 256 | 64 |
| 010 | 512 | 128 |
| 011 | 1024 | 256 |
| 100 | 2048 | 512 |
| 101 | 4096 | 1024 |
| 110 | 8192 | 2048 |
| 111 | 16384 | 4096 |

ADC Output Word Rate = ADC clock / OSR / ADC_Gain

- EX_1: ADC clock = 100k, OSR = 16384, ADC_Gain = 1. (ADCHPENB = 1)
ADC Output Word Rate = $100k/16384/1 = 6.1\text{Hz}$.
- EX_2: ADC clock = 100k, OSR = 16384, ADC_Gain = 2. (ADCHPENB = 1)
ADC Output Word Rate = $100k/16384/2 = 3\text{Hz}$
- EX_3: ADC clock = 250k, OSR = 64, ADC_Gain = 2. (ADCHPENB = 0)
ADC Output Word Rate = $250k/64/2 = 1.95\text{ kHz}$. (Fast ADC conversion rate example for Auto step on application)

- * **Note 1:** Adjust ADC clock (ADCKS) and OSR can get suitable ADC output word rate.
- * **Note 2:** For High resolution application, OSR set maximum value of 16384 recommended.
- * **Note 3:** Clear Bit DRDY after got ADC data or this bit will keep high all the time.
- * **Note 4:** ADCHPENB = 1 for Normal ADC conversion; ADCHPENB = 0 for Fast ADC conversion application.
- * **Note 5:** The Noise Free resolution of Fast ADC conversion is down to 10-bit.

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

```

@CPREG_Init:  XB0BSET      FBGRENB      ;Enable Band Gap Reference voltage

@ACM_Enable:  XB0BCLR      FACMSEL      ; Set ACM output 0.4V
               XB0BSET      FACMENB      ; Enable ACM Voltage

@AVE_Enable:  XB0BCLR      FAVESEL      ; Set AVE+ output 1.5V
               XB0BSET      FAVENB       ; Enable AVE+ Voltage

@AVDDR_Enable: XB0BSET      FAVDDRENB     ; Enable AVDDR Voltage=2.4V

@PGIA_Init:   MOV          A, #11110110B
               XB0MOV      AMPM, A        ; Enable Band Gap, Set :FDS="11" and PGIA Gain=128
               MOV        A, #00000000B
               XB0MOV      AMPCKS, A      ; Set AMPCKS = "000" for PGIA working clock = 31.25KHz
               MOV        A, #00h
               XB0MOV      AMPCHS, A     ; Selected PGIA differential input channel= AI+, AI-
               XB0BSET      FAMPENB      ; Enable PGIA function
               ; V (X+, X-) Output = V (AI+, AI-) x 128

@ADC_Init:    MOV          A, #10011000B
               XB0MOV      ADCM, A        ; ADC Reference voltage = 0.6V, ADC_Gain = 2x.
               MOV        A, #246
               XB0MOV      ADCKS, A      ; Set ADCKS=246 for ADC working clock=200K @ 4MHz
               MOV        A, #11110000B  ; Set ADC OSR = 16384.
               XB0MOV      DFM, A        ; ADC conversion rate =200k / 16384 / 2 = 6.1 Hz
               Call        Wait_300uS    ; Wait 300us for Regulators and analog function stable
               XB0BSET      FADCENB     ; Enable ADC function

@ADC_Wait:    XB0BTS1     FDRDY        ; Check ADC output new data or not
               JMP        @ADC_Wait     ; Wait for Bit DRDY = 1
               ; Output ADC conversion word

@ADC_Read:    XB0BCLR     FDRDY
               XB0MOV      A, ADCDH     ; Move ADC conversion High byte to Data Buffer
               B0MOV      Data_H_Buf, A
               XB0MOV      A, ADCDL
               B0MOV      Data_L_Buf, A  ; Move ADC conversion Low 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:    XB0BSET    FBGRENB    ;Enable Band Gap Reference voltage

@ACM_Enable:   XB0BCLR    FACMSEL    ; Set ACM output 0.4V
                XB0BSET    FACMENB      ; Enable ACM Voltage

@AVE_Enable:   XB0BCLR    FAVESEL     ; Set AVE+ output 1.5V
                XB0BSET    FAVENB      ; Enable AVE+ Voltage

@AVDDR_Enable: XB0BSET    FAVDDRENB   ; Enable AVDDR Voltage=2.4V

@PGIA_Init:    MOV        A, #1111110B    ; Enable Band Gap, Set :FDS="11" and PGIA Gain=128
                XB0MOV    AMPM, A
                MOV      A, #0000000B
                XB0MOV    AMPCKS, A    ; Set AMPCKS = "000" for PGIA working clock = 31.25KHz
                MOV      A, #00000110B
                XB0MOV    AMPCHS, A    ; Selected PGIA Voltage detection channel = VDD or VLCD.
                XB0BSET    FAMPENB     ; Enable PGIA function

@VDD_Detection: MOV      A, #00011100B   ; ADC internal Vref, ADC Gain = 1, Enable Voltage detect.
                XBMOV    ADCM, A        ; VDD detection function enable
                MOV      A, #0236
                XB0MOV    ADCKS, A      ; Set ADCKS=236 for ADC working clock=100K @ 4MHz
                MOV      A, #11110000B  ; Set ADC OSR = 16384
                XB0MOV    DFM, A        ; ADC conversion rate =100k / 16384 / 1 = 6.1 Hz
                Call     Wait_300uS     ; Wait 300us for Regulators and analog function stable
                XB0BSET    FADCENB     ; Enable ADC function

@ADC_Wait:     XB0BTS1   FDRDY      ; Check ADC output new data or not
                JMP      @ADC_Wait    ; Wait for Bit DRDY = 1
@ADC_Read:     XB0BCLR   FDRDY      ; Output ADC conversion word
                XB0MOV   A, ADCDH     ; Move ADC conversion High byte to Data Buffer
                B0MOV    Data_H_Buf, A
                XB0MOV   A, ADCDL     ; Move ADC conversion Low byte to Data Buffer
                B0MOV    Data_L_Buf, A
                ...
                ...
                ...
@VLCD_Detection: MOV      A, #00011110B   ; ADC internal Vref, ADC Gain = 1, Enable Voltage detect.
                XBMOV    ADCM, A        ; VDD detection function enable
                MOV      A, #0236
                XB0MOV    ADCKS, A      ; Set ADCKS=236 for ADC working clock=100K @ 4MHz
                MOV      A, #11110000B  ; Set ADC OSR = 16384
                XB0MOV    DFM, A        ; ADC conversion rate =100k / 16384 / 1 = 6.1 Hz
                Call     Wait_300uS     ; Wait 300us for Regulators and analog function stable
                XB0BSET    FADCENB     ; Enable ADC function

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

```

Example: Fast ADC conversion Rate setting

```

@ADC_Init:
MOV          A, #10011000B      ; ADC internal Vref, ADC Gain = 2
XBMOV       ADCM, A
MOV          A, #0248
XB0MOV      ADCKS, A           ; Set ADCKS=248 for ADC working clock=250K @ 4MHz
MOV          A, #11100000B      ; Set ADC OSR = 64, ADCHPENB =0,
XB0MOV      DFM, A            ; ADC conversion rate =250k / 64 / 2 = 1.95k Hz
Call        Wait_300uS         ; Wait 300us for Regulators and analog function stable
XB0BSET     FADCENB           ; Enable ADC function

@ADC_Wait:
XB0BTS1    FDRDY              ; Check ADC output new data or not
JMP        @ADC_Wait          ; Wait for Bit DRDY = 1

@ADC_Read:
XB0BCLR    FDRDY              ; Output ADC conversion word
XB0MOV     A, ADCDH
B0MOV     Data_H_Buf, A        ; Move ADC conversion High byte to Data Buffer
XB0MOV     A, ADCDL
B0MOV     Data_L_Buf, A        ; Move ADC conversion Low byte to Data Buffer
...
...
JMP        @ADC_Wait:         ; The Second ADC data available for application.

```

- * **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 300us for all functions stable.**
- * **Note 3: In fast ADC conversion application, the second ADC data will available for application.**
- * **Note 4: For increasing fast ADC conversion accuracy, recommend averaging several times of ADC row Data.**

11.5.5 LBTM: Low Battery Detect Register

SN8P1937 provided two different way to measure Power Voltage. One is from ADC reference voltage selection. It will be more precise but take more time and a little bit complex. The another way is using build in Voltage Comparator, divide power voltage and connect to P5.1, bit LBTO will output the P5.2 voltage Higher or Lower than Band Gap Reference Voltage (1.2V).

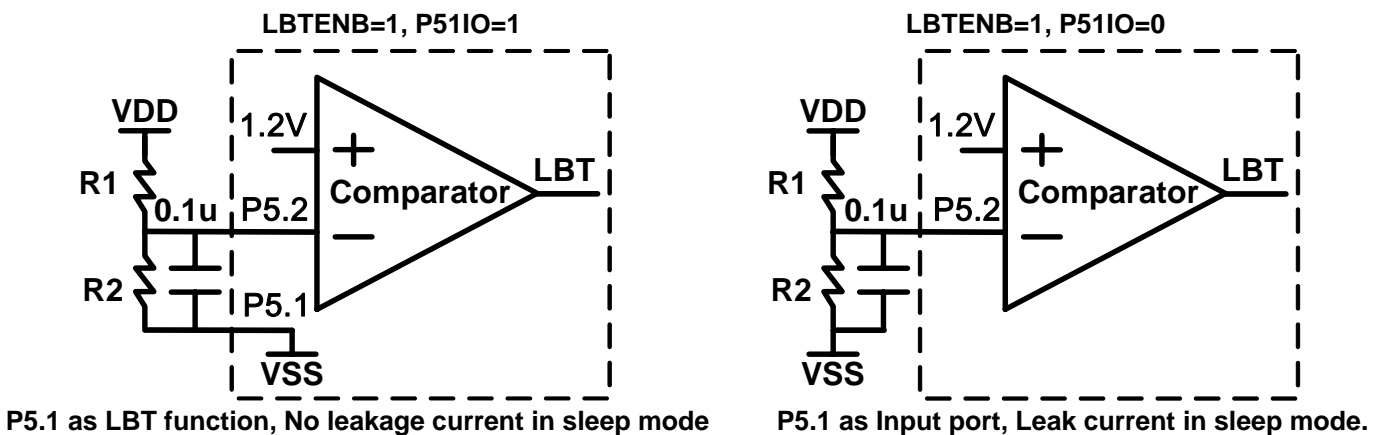
| 09AH | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------------------|-------|-------|-------|-------|-------|-------------|--------------|---------------|
| LBTM | - | - | - | - | - | LBTO | P51IO | LBTENB |
| R/W | - | - | - | - | - | R | R/W | R/W |
| After Reset | - | - | - | - | - | 0 | 0 | 0 |

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

- Bit1: **P51IO**: Port 5.1 Input/LBT function control bit.
0 = Set P51 as Input Port,
1 = Set P51 as LBT function

- Bit2: **LBTO**: Low Battery Detect Output Bit.
0 = P5.2/LBT voltage Higher than Band Gap Reference Voltage 1.2V.
1 = P5.2/LBT voltage Lower than Band Gap Reference Voltage 1.2V.

There are two circuit connections for LBT application, one is using P5.2 and P5.1, which can avoid power consumption in sleep mode, another is using P5.2 only. The second way will leak a small current in power down mode but can use P5.1 for Input application. These two circuits are following:



| Low Battery Voltage | R1 | R2 | LBTO=1 |
|---------------------|--------|--------|----------|
| 2.4V | 1MΩ | 1MΩ | VDD<2.4V |
| 3.6V | 1.33MΩ | 0.66MΩ | VDD<3.6V |
| 4.8V | 1.5MΩ | 0.5MΩ | VDD<4.8V |

* **Note: Get LBTO = 1 more 10 times in a raw every certain period, ex. 20 ms or more to make sure the Low Battery signal is stable.**

11.5.6 Analog Setting and Application

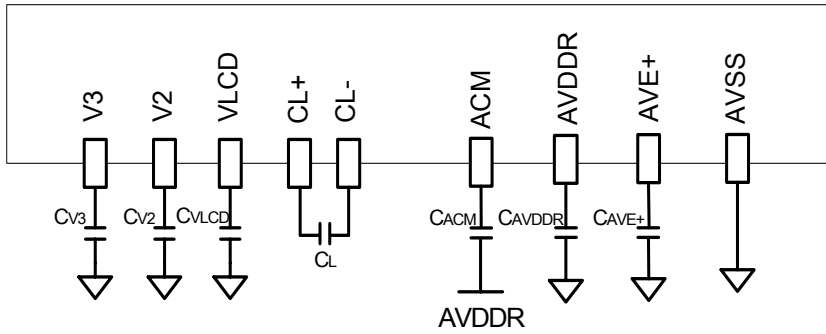
The most applications of SN8P1937 were for DC measurement ex. Weight scale, Pressure measure. Following table indicate different applications setting which MCU power source came from CR2032 battery, AA/AAA dry battery or external Regulator.

Capacitor Table:

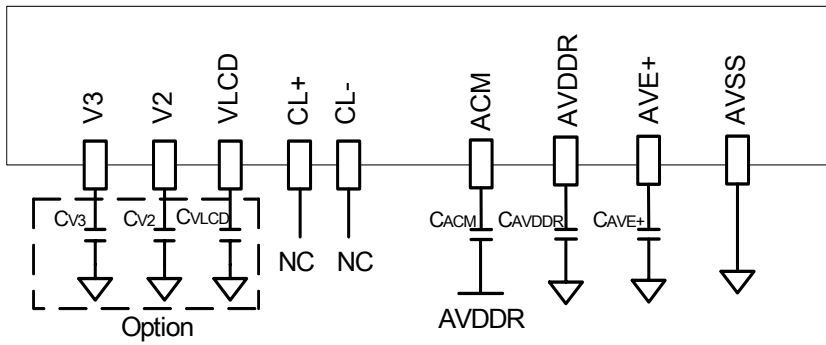
| Power type | AI+ | AI- | R+/R- | ACM | AVDDR | AVE+ | CL+/CL- | VDD (Pin17) | VDD (Pin28) |
|---------------------|------------------|------------------|----------------|------------------|--------------------|-------------------|----------------|-------------------|-------------------|
| | C _{AI+} | C _{AI-} | C _R | C _{ACM} | C _{AVDDR} | C _{AVE+} | C _L | C _{AVDD} | C _{DVDD} |
| CR2032 (2.4~3V) | 0.01uF | 0.01uF | 0.1uF | 0.1uF | 0.47uF | 0.47uF | 0.1uF | 10uF | 0.1uF |
| CR2032 ((4.4~6V)) | 0.1uF | 0.1uF | 0.1uF | 0.1uF | 0.47uF | 0.47uF | 0.1uF | 10uF | 0.1uF |
| AA/AAA Bat.(2.4~3V) | 0.1uF | 0.1uF | 0.1uF | 0.1uF | 0.47uF | 0.47uF | 0.1uF | 10uF | 0.1uF |
| AA/AAA Bat.(4.4~6V) | 0.1uF | 0.1uF | 0.1uF | 0.1uF | 0.47uF | 0.47uF | 0.1uF | 10uF | 0.1uF |
| External 5V Reg. | 0.1uF | 0.1uF | 0.1uF | 0.1uF | 0.47uF | 0.47uF | 0.1uF | 10uF | 0.1uF |

* *Note1: In R-Type LCD Driver mode, C_L is not connected to MCU.*

VDD=2.4V ~ 5.5V Analog Capacitor Connection (C-Type LCD Driver)

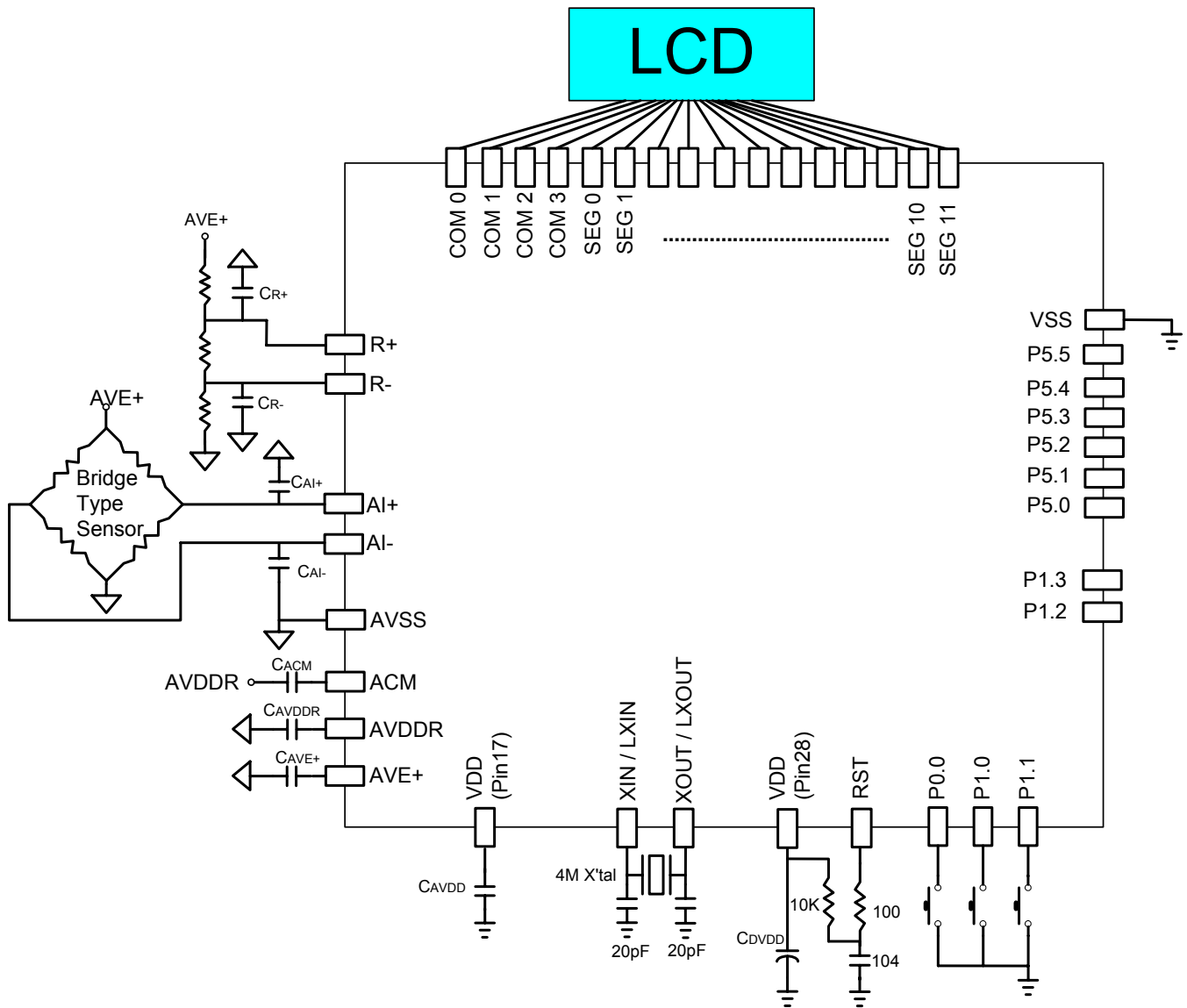


VDD=2.4V ~ 5.5V Analog Capacitor Connection (R-Type LCD Driver)



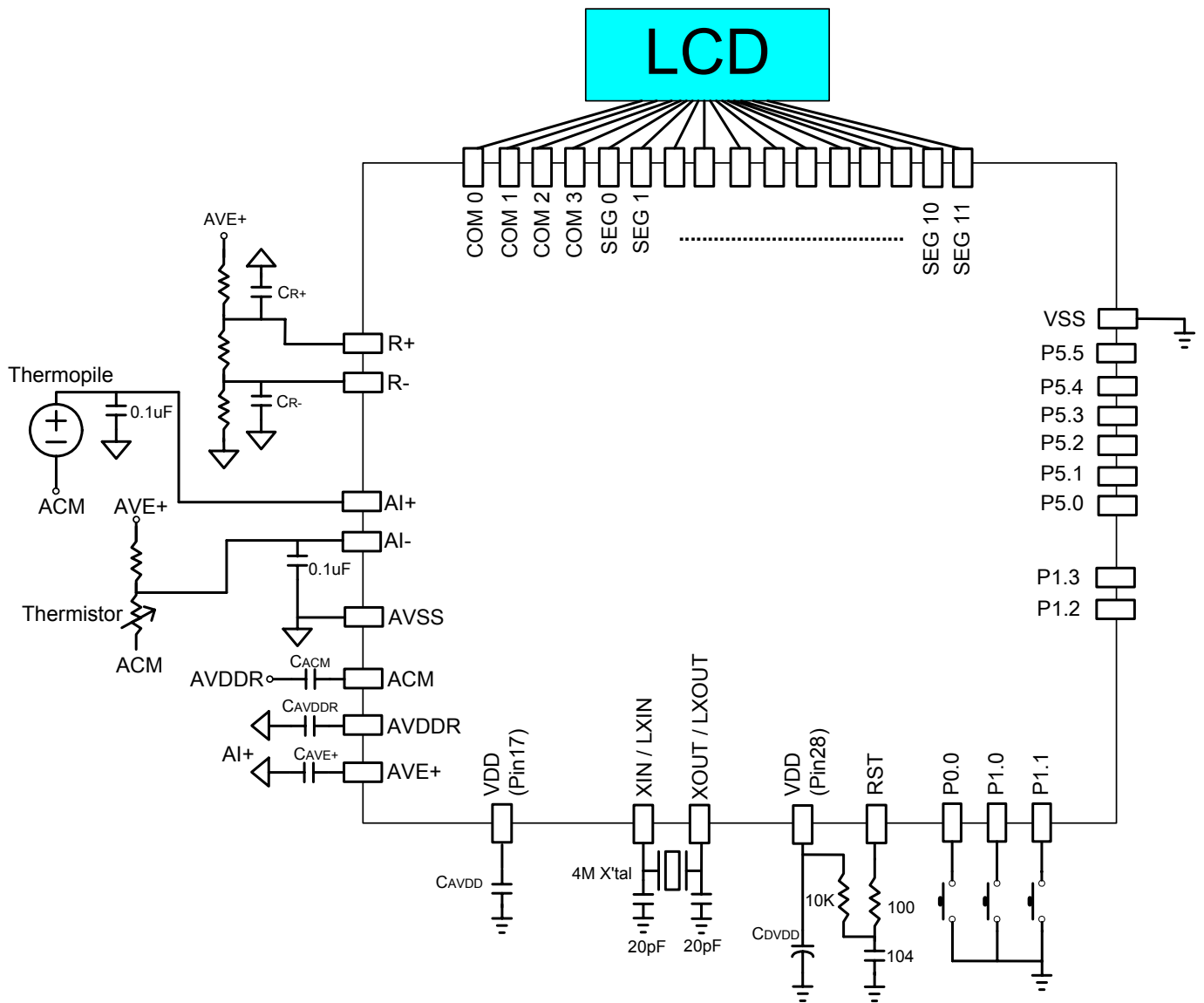
12 APPLICATION CIRCUIT

12.1 Scale (Load Cell) Application Circuit



* Note : Please refer 11.5.7 for capacitor setting.

12.2 Thermometer Application Circuit



* Note : Please refer 11.5.7 for capacitor setting.

13 INSTRUCTION SET TABLE

| Field | Mnemonic | Description | C | DC | Z | Cycle |
|---------------------|----------------------------|--|---|----|---|-------|
| MOV | MOV A,M | $A \leftarrow M$ | - | - | √ | 1 |
| | MOV M,A | $M \leftarrow A$ | - | - | - | 1 |
| | B0MOV A,M | $A \leftarrow M$ (bank 0) | - | - | √ | 1 |
| | B0MOV M,A | M (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 |
| | BOXCH A,M | $A \leftrightarrow M$ (bank 0) | - | - | - | 1 |
| MOVC | | R, $A \leftarrow ROM [Y,Z]$ | - | - | - | 2 |
| A R I T H M E T I 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 |
| | 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 |
| | B0ADD M,A | M (bank 0) $\leftarrow M$ (bank 0) + A, if occur carry, then C=1, else C=0 | √ | √ | √ | 1 |
| | 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 |
| | 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 |
| C | 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 |
| L O G I C | AND A,M | $A \leftarrow A$ and M | - | - | √ | 1 |
| | AND M,A | $M \leftarrow A$ and M | - | - | √ | 1 |
| | AND A,I | $A \leftarrow A$ and I | - | - | √ | 1 |
| | OR A,M | $A \leftarrow A$ or M | - | - | √ | 1 |
| | OR M,A | $M \leftarrow A$ or M | - | - | √ | 1 |
| | OR A,I | $A \leftarrow A$ or I | - | - | √ | 1 |
| | XOR A,M | $A \leftarrow A$ xor M | - | - | √ | 1 |
| | XOR M,A | $M \leftarrow A$ xor M | - | - | √ | 1 |
| | XOR A,I | $A \leftarrow A$ xor I | - | - | √ | 1 |
| P R O C E S S I N G | 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 |
| | RRC M | $A \leftarrow RRC M$ | √ | - | - | 1 |
| | RRCM M | $M \leftarrow RRC M$ | √ | - | - | 1 |
| | RLC M | $A \leftarrow RLC M$ | √ | - | - | 1 |
| | RLCM M | $M \leftarrow RLC M$ | √ | - | - | 1 |
| | CLR M | $M \leftarrow 0$ | - | - | - | 1 |
| | BCLR M.b | $M.b \leftarrow 0$ | - | - | - | 1 |
| | BSET M.b | $M.b \leftarrow 1$ | - | - | - | 1 |
| | BOBCLR M.b | $M(bank 0).b \leftarrow 0$ | - | - | - | 1 |
| BOBSET M.b | $M(bank 0).b \leftarrow 1$ | - | - | - | 1 | |
| B R A N C H I N G | 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 + 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 + 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 |
| | BOBTS0 M.b | If M (bank 0).b = 0, then skip next instruction | - | - | - | 1 + S |
| | BOBTS1 M.b | If M (bank 0).b = 1, then skip next instruction | - | - | - | 1 + S |
| | JMP d | PC15/14 \leftarrow RomPages1/0, PC13~PC0 \leftarrow d | - | - | - | 2 |
| | CALL d | Stack \leftarrow PC15~PC0, PC15/14 \leftarrow RomPages1/0, PC13~PC0 \leftarrow d | - | - | - | 2 |
| M I S C | RET | PC \leftarrow Stack | - | - | - | 2 |
| | RETI | PC \leftarrow Stack, and to enable global interrupt | - | - | - | 2 |
| | NOP | No operation | - | - | - | 1 |

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

14 Development Tools

14.1 Development Tool Version

14.1.1 ICE (In circuit emulation)

- SN8ICE 1K: (S8KD-2) Full function emulates SN8P1937 series

- * SN8ICE1K ICE emulation notice
 - Operation voltage of ICE: 3.0V~5.0V.
 - Recommend maximum emulation speed at 5V: 4 MIPS (e.g. 16MHZ crystal and $F_{cpu} = F_{osc}/4$).
 - Use SN8P1937 EV-KIT to emulation Analog Function.

- * Note: S8ICE2K doesn't support SN8P1937 serial emulation.

14.1.2 OTP Writer

MPIII Writer : ON/OFF line operation to support SN8P1937 mass production.

- * Note: Writer 3.0 doesn't support SN8P1937 OTP programming.

14.1.3 IDE (Integrated Development Environment)

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

- For SN8ICE 1K: SN8IDE 1.99Z05 or later
- Easy Writer and MP-Easy Writer doesn't support SN8P1937.
- M2IDE V1.0X doesn't support SN8P1937

14.2 OTP Programming Pin to Transition Board Mapping

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

| Easy Writer JP1/JP2 | | | | Easy Writer JP3 (Mapping to 48-pin text tool) | | | |
|---------------------|----|----|------------|---|----|----|-------|
| VSS | 2 | 1 | VDD | DIP1 | 1 | 48 | DIP48 |
| CE | 4 | 3 | CLK/PGCLK | DIP2 | 2 | 47 | DIP47 |
| OE/ShiftDat | 6 | 5 | PGM/OTPCLK | DIP3 | 3 | 46 | DIP46 |
| D0 | 8 | 7 | D1 | DIP4 | 4 | 45 | DIP45 |
| D2 | 10 | 9 | D3 | DIP5 | 5 | 44 | DIP44 |
| D4 | 12 | 11 | D5 | DIP6 | 6 | 43 | DIP43 |
| D6 | 14 | 13 | D7 | DIP7 | 7 | 42 | DIP42 |
| VPP | 16 | 15 | VDD | DIP8 | 8 | 41 | DIP41 |
| RST | 18 | 17 | HLS | DIP9 | 9 | 40 | DIP40 |
| ALSB/PDB | 20 | 19 | - | DIP10 | 10 | 39 | DIP39 |
| | | | | DIP11 | 11 | 38 | DIP38 |
| | | | | DIP12 | 12 | 37 | DIP38 |
| | | | | DIP13 | 13 | 36 | DIP36 |
| | | | | DIP14 | 14 | 35 | DIP35 |
| | | | | DIP15 | 15 | 34 | DIP34 |
| | | | | DIP16 | 16 | 33 | DIP33 |
| | | | | DIP17 | 17 | 32 | DIP32 |
| | | | | DIP18 | 18 | 31 | DIP31 |
| | | | | DIP19 | 19 | 30 | DIP30 |
| | | | | DIP20 | 20 | 29 | DIP29 |
| | | | | DIP21 | 21 | 28 | DIP28 |
| | | | | DIP22 | 22 | 27 | DIP27 |
| | | | | DIP23 | 23 | 26 | DIP26 |
| | | | | DIP24 | 24 | 25 | DIP25 |

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

JP3 for MP transition board

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

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

Writer V3.0 JP1 Pin Assignment

14.2.3 SN8P1937 Series Programming Pin Mapping:

| OTP Programming Pin of SN8P1937 Series | | | |
|---|------------|------------------------------------|------------|
| Chip Name | | SN8P1937 | |
| Easy, MP-EZ Writer And Writer V3.0 | | OTP IC / JP3 Pin Assignment | |
| Number | Pin | Number | Pin |
| 1 | VDD | 17,28 | VDD |
| 2 | GND | 13,25,48 | VSS |
| 3 | CLK | 31 | P1.0 |
| 4 | CE | - | - |
| 5 | PGM | 32 | P1.1 |
| 6 | OE | 39 | P1.2 |
| 7 | D1 | - | - |
| 8 | D0 | - | - |
| 9 | D3 | - | - |
| 10 | D2 | - | - |
| 11 | D5 | - | - |
| 12 | D4 | - | - |
| 13 | D7 | - | - |
| 14 | D6 | - | - |
| 15 | VDD | 17,28 | VDD |
| 16 | VPP | 29 | RST |
| 17 | HLS | - | - |
| 18 | RST | - | - |
| 19 | - | - | - |
| 20 | ALSB/PDB | 40 | P1.3 |

14.3 SN8P1937 EV-KIT

14.3.1 INTRODUCTION

Sonix provides a complete EV-KIT for SN8P1937, which includes an ICE with S8KD chip, SN8P1937 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. Also, the system has built-in 5.0V power supply.

14.3.2 PCB DESCRIPTION

Sonix provides SN8P1937 EV board for all functions emulation shown in FIG.1

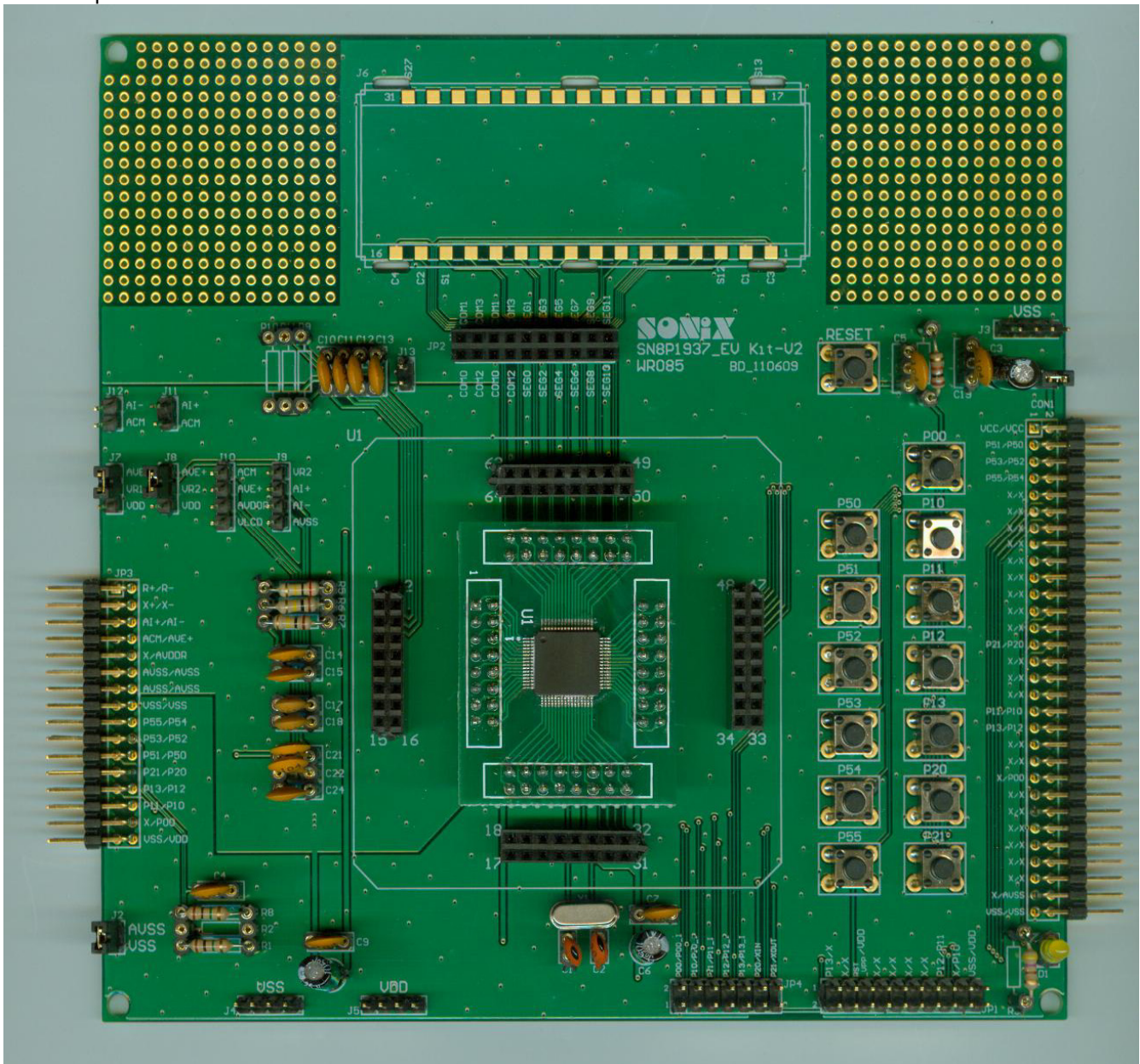


FIG.1 SN8P1937 EV board

14.3.3 EV BOARD SETTING

1. CON1 : connecting with the ICE PORT
2. J1 : Switch it “SHORT” position when using the power from ICE, and switch it “OPEN” when using connecting the power from other source.
3. D1 : Power indicator
4. RESET : RESET KEY
5. JP1 : OTP write-in PORT
6. JP2 : LCD PORT
7. JP3: Target board connector.
select different voltage setting for VLCD.
8. JP4 : When separate ICE and Ev board, switch JP4 to “SHORT” , or else the relative IO port won't work.
9. J2: AVSS and VSS “SHORT” Pin.
10. J6: LCD COM/SEG connect Pin.
11. J7: Selection of external reference voltage V(R+, R-) power from VDD or AVE+.
12. J8: Selection of Load cell power from VDD or AVE+.
13. J9: Analog Differential (AI+, AI-) input Pin.
14. J10: Analog Voltage output Pin (ACM/AVE+/AVDDR/VLCD).
15. J11/J12: Analog Single-End AI+/ACM and AI-/ACM input Pin.
16. R5/R6/R7: Resistors for ADC external reference voltage.
17. R1/R2/R8: LBT function resistance.
18. R9/R10/R11: R-Type LCD external resistors for voltage division. User can add resistance between VLCD / V3 / V2 / V1 for more driving current.
19. C10/C11/C12/C13: C-Type LCD external capacitors.

| | EV board connects ICE | EV board connects WRITER | Only EV board |
|----------------|-----------------------------|-----------------------------------|--|
| JP1 | Disconnect | Write-in SN8P1937 | Disconnect |
| JP2(LCD) | No Function | - | Depends on actual situation |
| JP3 | Depends on actual situation | Disconnect | Depends on actual situation |
| JP4 | Disconnect all | Disconnect all | Connect all |
| J6/7/8/9/10/11 | Depends on actual situation | - | Depends on actual situation |
| J1 | Depends on actual situation | Short | Short |
| J2 | Short | Short | Short |
| U1 | SN8P1937 EV-Link IC only | Blank SN8P1937 IC for programming | SN8P1937 IC with system application code |
| R1/R2/R8(LBT) | Depends on actual situation | - | Depends on actual situation |
| C10~C13 | - | - | C-Type LCD Driver |

14.3.4 SN8P1937 EV BOARD CONNECT WITH ICE

1. EV Board U1 requires IC board embedded with connection program when attach to each other.
2. When power source is supply from the ICE , J1 should be in "SHORT".
3. When connecting to the develop system, the JP4 should be "OPEN". Or else the system will not be able to reset successfully.
4. Use the same oscillator for both EV Board and ICE.
5. When simulate both EV Board and developer system , J2 needed to be in "SHORT"(Digital GND and Analog GND both connected).
6. R1/R2/R8 are for LBT function. Connect the resistance for this function.
7. J7: ADC external reference voltage source.
8. J8: Load cell power source.

14.3.5 STAND ALONG EV BOARD

1. Separate EV board and ICE.
2. Attached the IC board with pre-write-in program code to EV Board U1.
3. When using the stand along EV Board JP4 should all be in "SHORT" , in order to prevent malfunction for relative IO ports.
4. Confirm J2's connectivity status , J2 should be in "SHORT"(Digital GND and Analog GND both connected).
5. C10~C13 are used for C-type LCD driver.

14.3.6 SONIX ASSEMBLER

SONIX provides SN8ASM Assembler program, and all sort developer related SN8P1937 software, and as well as complier. The combination of the SONIX ICE and SN8P1937 Emulation Board are used for the purpose of actual circuitry verification and testing, thus to save the costs and time for development.

14.3.7 SYSTEM REQUIREMENT

Operating system:

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

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

Files Description:

| | |
|------------------|---|
| SN8IDE_ xxxx.EXE | Assembler software package , xxx means version. |
| SN8ASMxxxx.EXE | Main execution program. xxxx mean version. |
| MACRO1.H | Reference macro one |
| MACRO2.H | Reference macro two |

| | |
|------------------|---|
| MACRO3.H | Reference macro three |
| SN8P1937.INC | Define SN8P1937 all function |
| 1937Ev.H | Constant and Macro definition for SN8P1937 Ev Kit emulation code |
| 1937Ev.ASM | SN8P1937 EV Kit interface subroutine. User must include this file to communication with Ev. Kit |
| 1937_EV_Demo.ASM | SN8P1937 Demo code |

14.3.8 NOTE FOR SOFTWARE INSTALLATION

1. Check if the SN8P1937.INC has been included in the designated folder for SONiX Assembler ◦ (Default directory is C:\Sonix\Sn8IDE_xxx\use_inc2)
2. Check if the main program #include 1937Ev.H and 1937Ev.ASM, please refer to 1937_TEMPLATE.ASM for detailed instruction ◦
3. The first line of the main program are listed below :
ICE_Mode EQU 1
CHIP SN8P1937
4. When done with the programming, users must set the first line in the main program to real chip mode for OTP programming, example is as follows :
ICE_Mode EQU 0
CHIP SN8P1937

Warning:

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

14.3.9 EXAMPLE: PROGRAM CODE STRUCTURE

```

*****
;
; FILENAME : 1937_Demo.ASM
; AUTHOR : SONiX
; PURPOSE : Demo Code for SN8P1937
*****
;* (c) Copyright 2009, SONiX TECHNOLOGY CO., LTD.
*****
;
ICE_Mode EQU 1 ; 1 for ICE , 0 for real chip
;ICE_Mode EQU 0
CHIP SN8P1937 ; Select the CHIP
;-----

```

```

; Include Files
;-----
.nolist           ; do not list the macro file
    INCLUDESTD MACRO1.H
    INCLUDESTD MACRO2.H
    INCLUDESTD MACRO3.H
    INCLUDE 1937Ev.h      ; for ICE linking emulation board
.list           ; Enable the listing function
;-----
; Constants Definition
;-----
;   ONE EQU 1
;-----
; Variables Definition
;-----
.DATA
    org 0h           ; Bank 0 data section start from RAM address 0x000
Wk00B0 DS 1         ;Temporary buffer for main loop
lwk00B0 DS 1         ;Temporary buffer for ISR
AccBuf DS 1         ; Accumulater buffer
PflagBuf DS 1       ;PFLAG buffer
;-----
; Bit Flag Definition
;-----
Wk00B0_0 EQU Wk00B0.0 ;Bit 0 of Wk00B0
lwk00B0_1 EQU lwk00B0.1 ;Bit 1 of lwk00
;-----
; Code section
;-----
.CODE
    ORG 0           ;Code section start
    jmp Reset       ;Reset vector
                    ;Address 4 to 7 are reserved

    ORG 8
    Jmp Isr         ;Interrupt vector
    ORG 10h
;-----
; Program reset section
;-----
Reset:
    mov A,#07Fh     ;Initial stack pointer and
    b0mov STKP,A    ;disable global interrupt

```

```

b0mov    PFLAG,#00h    ;pflag = x,x,x,x,x,c,dc,z
b0mov    RBANK,#00h    ;Set initial RAM bank in bank 0
call     ClrRAM        ;Clear RAM
call     Syslnit       ;System initial
;-----
INIT_1937Ev          ; for ICE linking emulation board
;-----
b0bclr   FGIE          ;Enable global interrupt
;-----
; Main routine
;-----
Main:
b0bset   FWDRST        ;Clear watchdog timer
mov      a, #11010000B;
XB0MOV   CPM,a         ;using XB0MOV command for CPM setting
Call     @Delay_300us ;Delay 300us for Analog voltage stable.
jmp      Main
;-----
INCLUDE 1937Ev.asm   ; SN8P1937 Ev. Kit interface code
;-----
ENDP

```

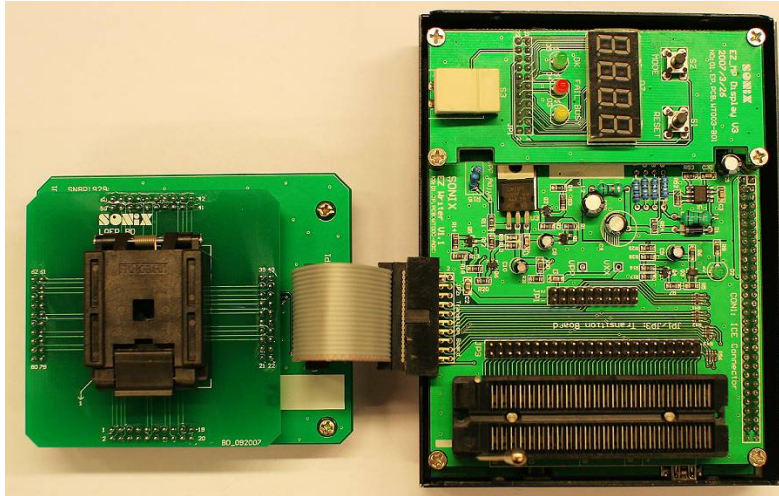
Please be aware of the position of the listed file names marked in red.

14.3.10 OTP WRITE-IN STEP

Using SN8P1937 EV-board to program is shown in following step:

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

14.3.11 SN8P1937 PROGRAMMING BOARD CONNECT TO MPIII WRITER



15 ELECTRICAL CHARACTERISTIC

15.1 ABSOLUTE MAXIMUM RATING

| | |
|--|------------------------------------|
| Supply voltage (V_{DD})..... | - 0.3V ~ 6.0V |
| Input in voltage (V_{IN})..... | $V_{SS} - 0.2V \sim V_{DD} + 0.2V$ |
| Operating ambient temperature (T_{OPR})..... | $0^{\circ}C \sim + 70^{\circ}C$ |
| Storage ambient temperature (T_{STOR})..... | $-40^{\circ}C \sim + 125^{\circ}C$ |

15.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to V_{SS} , $V_{DD} = 5.0V, F_{OSC} = 4MHz, F_{cpu} = 1MHz$, ambient temperature is $25^{\circ}C$ unless otherwise note.)

| PARAMETER | SYM. | DESCRIPTION | MIN. | TYP. | MAX. | UNIT | |
|---|---|---|---------------------|------|--------|-------|----|
| Operating voltage | Vdd | Normal mode, Vpp = Vdd | 2.4 | 5.0 | 5.5 | V | |
| RAM Data Retention voltage | Vdr | | - | 1.5 | - | V | |
| 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 | |
| | ViL2 | Reset pin | Vss | - | 0.2Vdd | V | |
| Input High Voltage | ViH1 | All input pins | 0.7Vdd | - | Vdd | V | |
| | ViH2 | Reset pin | 0.9Vdd | - | Vdd | V | |
| Reset pin leakage current | Ilekg | Vin = Vdd | - | - | 5 | uA | |
| I/O port pull-up resistor | Rup | Vin = Vss , Vdd = 3V | 100 | 200 | 300 | KΩ | |
| | | Vin = Vss , Vdd = 5V | 50 | 100 | 180 | KΩ | |
| I/O port input leakage current | Ilekg | Pull-up resistor disable, Vin = Vdd | - | - | 2 | uA | |
| I/O Port source current sink current | IoH | Vop = Vdd - 0.5V | 8 | 12 | - | mA | |
| | IoL | Vop = Vss + 0.5V | 8 | 15 | - | | |
| INTn trigger pulse width | Tint0 | INT0 interrupt request pulse width | 2/fcpu | - | - | cycle | |
| Supply Current | Idd1 | normal Mode (Low Power Disable, Analog Parts OFF) | Vdd= 5V 4MHz | - | 1.5 | 3 | mA |
| | | | Vdd= 5V IHRC | - | 1.5 | 3 | |
| | | | Vdd= 3V 4MHz | - | 0.6 | 1.2 | |
| | | | Vdd= 3V IHRC | - | 0.8 | 1.2 | |
| | Idd2 | normal Mode (Low Power Enable, Analog Parts OFF) | Vdd= 5V 4MHz | - | 1 | 2 | mA |
| | | | Vdd= 5V IHRC | - | 1 | 2 | |
| | | | Vdd= 3V 4MHz | - | 0.5 | 1 | |
| | | | Vdd= 3V IHRC | - | 0.7 | 1.4 | |
| | Idd3 | normal Mode (Low Power Disable, Analog Parts ON) | Vdd= 5V 4MHz | - | 2 | 4 | mA |
| | | | Vdd= 5V IHRC | - | 2 | 4 | |
| | | | Vdd= 3V 4MHz | - | 1.2 | 2.4 | |
| | | | Vdd= 3V IHRC | - | 1.2 | 2.4 | |
| | Idd4 | normal Mode (Low Power Enable, Analog Parts ON) | Vdd= 5V 4MHz | - | 1.6 | 3.2 | mA |
| | | | Vdd= 5V IHRC | - | 1.6 | 3.2 | |
| | | | Vdd= 3V 4MHz | - | 1.2 | 2.4 | |
| | | | Vdd= 3V IHRC | - | 1.2 | 2.4 | |
| | Idd5 | Slow mode (Stop High Clock, LCD OFF) | Vdd= 5V Ext.32768Hz | - | 20 | 40 | uA |
| | | | Vdd= 5V | - | 12 | 24 | |
| | | | Vdd= 3V Ext.32768Hz | - | 7 | 14 | |
| | | | Vdd= 3V | - | 4 | 8 | |
| | Idd6 | Slow mode (Stop High Clock, R-LCD ON 200K) | Vdd= 5V Ext.32768Hz | - | 28 | 56 | uA |
| | | | Vdd= 5V | - | 15 | 30 | |
| | | | Vdd= 3V Ext.32768Hz | - | 16 | 32 | |
| | | | Vdd= 3V | - | 12 | 24 | |
| Idd7 | Slow mode (Stop High Clock, C-LCD) CP clock =32k , LCD no loading | Vdd= 5V Ext.32768Hz | - | 45 | 90 | uA | |
| | | Vdd= 5V | - | 40 | 80 | | |
| | | Vdd= 3V Ext.32768Hz | - | 36 | 72 | | |
| | | Vdd= 3V | - | 28 | 56 | | |
| Idd8 | | Green mode | Vdd= 5V Ext.32768Hz | - | 15 | 30 | uA |

| | | | | | | | | |
|---------------------------|-------------------|---------------------------------------|--|---------------------|------|-----|-----|----|
| | | | Vdd= 5V | - | 6 | 12 | uA | |
| | | | Vdd= 3V Ext.32768Hz | - | 5 | 10 | uA | |
| | | | Vdd= 3V | - | 2 | 4 | uA | |
| | Idd9 | | Green mode *Stop High Clock *R-Type LCD ON 200K | Vdd= 5V Ext.32768Hz | - | 25 | 50 | uA |
| | | | | Vdd= 5V | - | 15 | 30 | uA |
| | | | | Vdd= 3V Ext.32768Hz | - | 10 | 10 | uA |
| | | | | Vdd= 3V | - | 7 | 14 | uA |
| | Idd10 | | Green mode *Stop High Clock *C-Type LCD ON, BGM = 0. *LCDCPK=32k,no LCD loading, C+ = 0.1uF | Vdd= 3V Ext.32768Hz | - | 35 | 70 | uA |
| | | | | Vdd= 3V | - | 30 | 30 | uA |
| | Idd11 | | Green mode *High Clock Non-stop *C-Type LCD ON, BGM = 1. *LCDCPK=32k,no LCD loading, C+ = 0.1u | Vdd= 3V Ext.32768Hz | - | 200 | 400 | uA |
| | | | | Vdd= 3V | - | 200 | 400 | uA |
| | Idd13 | | Sleep Mode | Vdd= 5V | - | 1 | 2 | uA |
| Vdd= 3V | | | | - | 0.7 | 1.5 | uA | |
| LVD Detect Level | V _{LVD} | Internal POR detect level | 0°C~70°C | 1.7 | 2.07 | 2.3 | V | |
| Internal High Clock Freq. | F _{IHRC} | Internal High RC Oscillator Frequency | | 14 | 16 | 18 | MHz | |

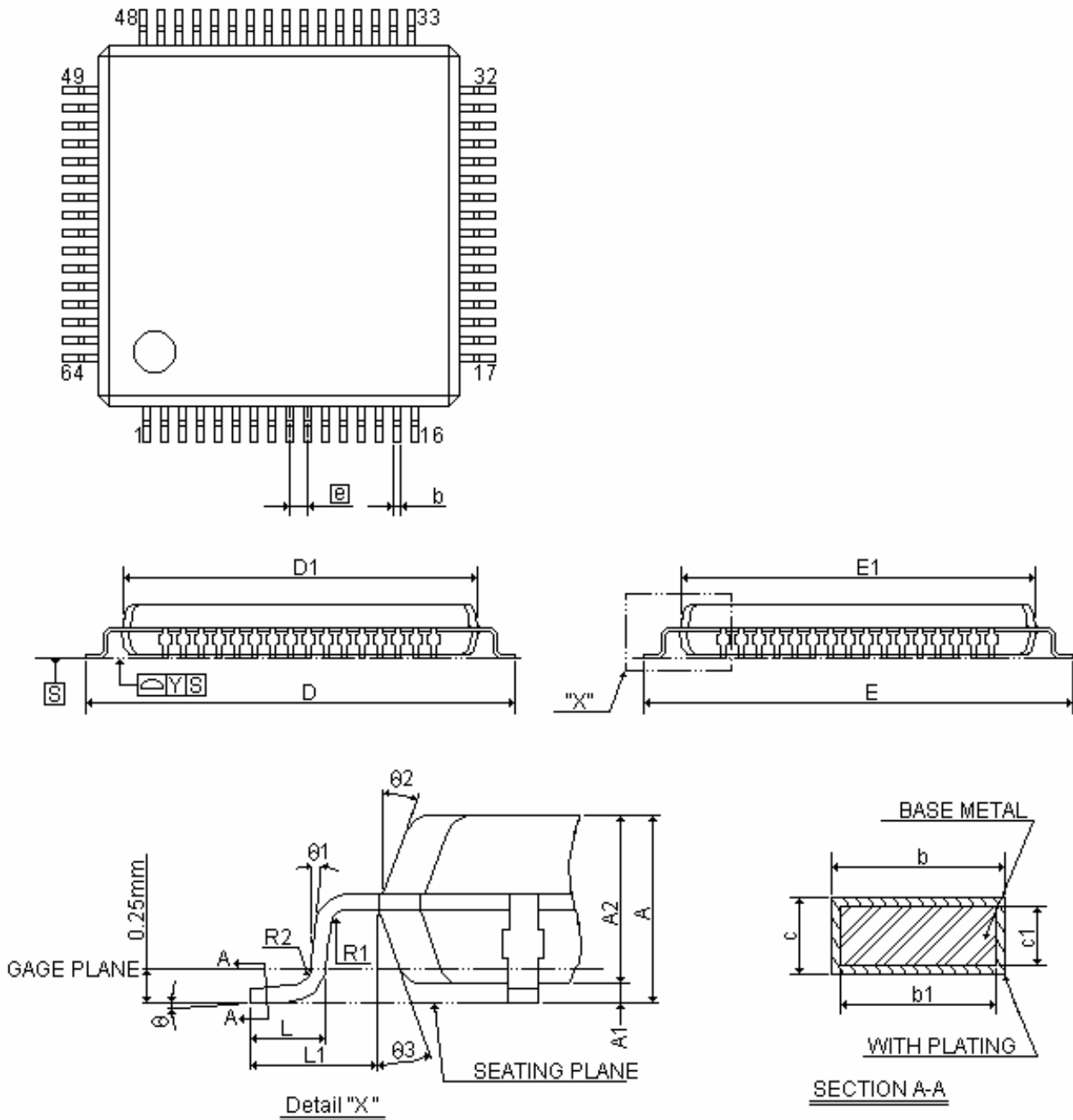
➤ **Note: Analog Parts including Regulator, PGIA and ADC.**

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

| PARAMETER | SYM. | DESCRIPTION | MIN. | TYP. | MAX. | UNIT |
|---|----------------------|-------------------------------------|------|------|------|--------|
| Analog to Digital Converter | | | | | | |
| Operating current | I _{DD_ADC} | Run mode @ 2.4V | | 250 | 300 | uA |
| Power down current | I _{PDN} | Stop mode @ 2.4V | | 0.1 | 1 | μA |
| Conversion rate | F _{SMP} | ENOB 14-bit | | | 25 | Sps |
| | | ENOB 10-bit | | | 1000 | Sps |
| Reference Voltage Input Voltage | V _{ref} | R+, R- Input Range (External Ref.) | 0.3 | | 0.9 | V |
| | | R+, R- Input Range (Internal Ref.) | 0.3 | | 0.9 | V |
| Differential non-linearity | DNL | ADC range ± 29491 | | ±0.5 | ±0.5 | LSB |
| Integral non-linearity | INL | ADC range ± 29491 | | ±1 | ±4 | LSB |
| No missing code | NMC | ADC range ± 29491 | 16 | | | bit |
| Noise free code | NFC | ADC range ± 29491 | | 14 | 16 | bit |
| Effective number of bits | ENOB | ADC range ± 29491 | | 14 | 16 | bit |
| ADC Input range | V _{AIN} | | 0.4 | | 1.05 | V |
| Temperature Sensor inaccuracy | E _{TS} | Inaccuracy range vs. real Temp. | | ±8 | | °C |
| PGIA | | | | | | |
| Current consumption | I _{DD_PGIA} | Run mode @ 2.4V | | 200 | 250 | uA |
| Power down current | I _{PDN} | Stop mode @ 2.4V | | | 0.1 | uA |
| Input offset voltage | V _{os} | | | 25 | 50 | uV |
| Bandwidth | BW | | | | 100 | Hz |
| PGIA Gain Range (Gain=128x) | GR | VDD = 2.4V | 110 | 128 | 150 | |
| PGIA Input Range | V _{opin} | AI+,AI- input range. (AVDDR = 2.4V) | 0.4 | | 1.05 | V |
| PGIA Output Range | V _{opout} | X+,X- output range. (AVDDR = 2.4V) | 0.4 | | 1.05 | V |
| Band gap Reference (Refer to ACM) | | | | | | |
| Band gap Reference Voltage | V _{BG} | | 1.18 | 1.23 | 1.28 | V |
| Reference Voltage Temperature Coefficient | T _{ACM} | | | 50* | | PPM/°C |
| Operating current | I _{BG} | Run mode @ 2.4V | | 200 | 250 | uA |
| Regulator | | | | | | |
| Regulator output voltage AVDDR | V _{AVDDR} | | 2.25 | 2.4 | 2.5 | V |
| Regulator output voltage AVE+ | V _{AVE+} | AVE+ set as 1.5V | 1.4 | 1.5 | 1.6 | V |
| Analog common voltage | V _{ACM} | V _{ACM} = 0.4 | 0.35 | 0.4 | 0.45 | V |
| Regulator output current capacity | I _{VA+} | | 10 | | | mA |
| Quiescent current | I _{QI} | ACM + AVDDR + AVE | | 80 | 100 | uA |
| V _{ACM} driving capacity | I _{SRC} | | 10 | - | - | μA |
| V _{ACM} sinking capacity | I _{SNK} | | 1 | - | - | mA |

16 PACKAGE INFORMATION

16.1 LQFP 64 PIN



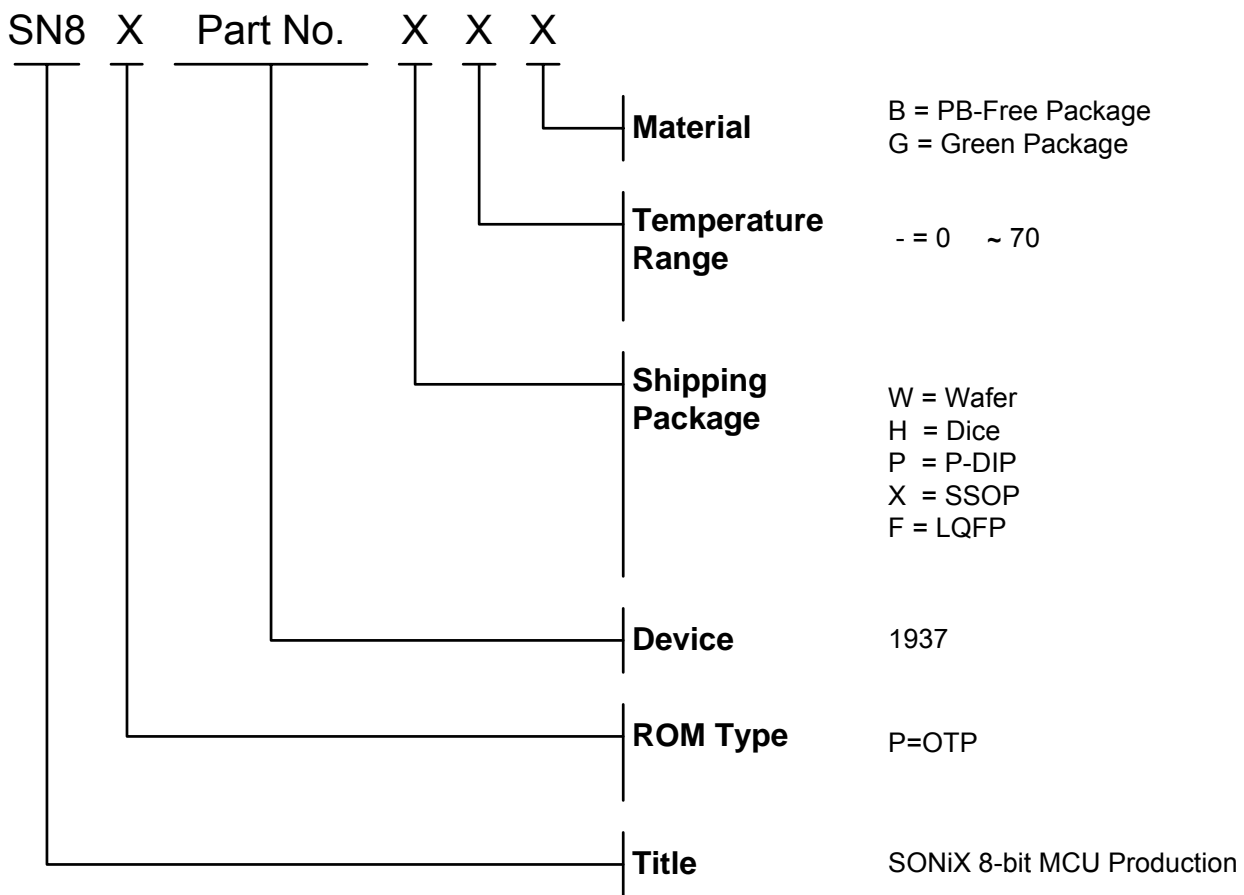
| SYMBLE | DIMENSION (MM) | | | DIMENSION (MIL) | | |
|------------|----------------|-------|-------|-----------------|------|------|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | | | 1.60 | | | 63 |
| A1 | 0.05 | 1.40 | 0.15 | 2 | 55 | 6 |
| A2 | 1.36 | 0.22 | 1.45 | 35 | 9 | 57 |
| b | 0.17 | 0.22 | 0.27 | 7 | 8 | 11 |
| b1 | 0.17 | | 0.23 | 7 | | 12 |
| c | 0.09 | | 0.20 | 4 | | 8 |
| c1 | 0.09 | | 0.16 | 4 | | 6 |
| D | 11.75 | 12.00 | 12.25 | 463 | 473 | 483 |
| D1 | 9.95 | 10.00 | 10.05 | 392 | 394 | 396 |
| E | 11.75 | 12.00 | 12.25 | 463 | 473 | 483 |
| E1 | 9.95 | 10.00 | 10.05 | 392 | 394 | 396 |
| [e] | | 0.50 | | | 20 | |
| L | 0.45 | 0.60 | 0.75 | 18 | 24 | 30 |
| L1 | 0.9 | 1 | 1.1 | | 39 | |
| R1 | 0.08 | | | 3 | | |
| R2 | 0.08 | | 0.20 | 3 | | 8 |
| Y | | | 0.075 | | | 3 |
| θ | 0° | 3.5° | 7° | 0° | 3.5° | 7° |
| θ 1 | 0° | | | 0° | | |
| θ 2 | 11° | 12° | 13° | 11° | 12° | 13° |
| θ 3 | 11° | 12° | 13° | 11° | 12° | 13° |

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.

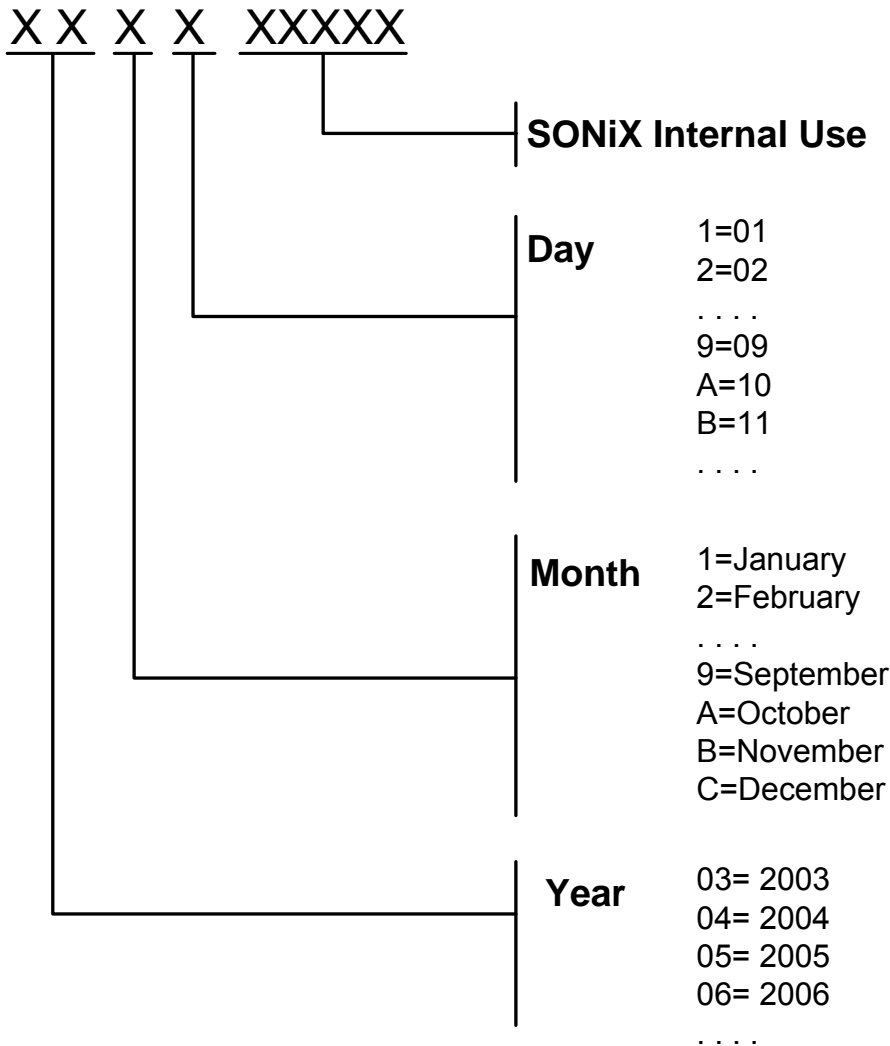
17.2 MARKING INDETIFICATION SYSTEM



17.3 MARKING EXAMPLE

| Name | ROM Type | Device | Package | Temperature | Material |
|------------|----------|--------|---------|-------------|-----------------|
| SN8P1937FB | OTP | 1937 | LQFP | 0°C~70°C | PB-Free Package |
| SN8P1937FG | OTP | 1937 | LQFP | 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