SN8P2318 Series

USER'S MANUAL

Version 1.0

SN8P2317
SN8P2318

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 us as components in systems intended, for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SONIX product could create a situation where personal injury or death may occur. Should Buyer purchase or use SONIX products for any such unintended or unauthorized application. Buyer shall indemnify and hold SONIX and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that SONIX was negligent regarding the design or manufacture of the part.

AMENDENT HISTORY

Version	Date	Description
VER 0.1	Mar. 2010	First issue.
VER 0.2	July. 2010	Add SN8P2317 48 pin item.
VER 1.0	May. 2011	1. Modify XIN Pin Description : In PLL_16M mode, XIN connects a decouple 0.1uF capacitor to VDD. 2. Add High_Clk = PLL_16M limited. 3. SN8P2318 support to industry -40~+85℃ temperature range. 4. Modify “DEVELOPMENT TOOL” description. 5. Modify “Chapter 9.4 RFCM REGISTER” RFCH[2:1] >> RFCH[2:0]. 6. Modify “Chapter 9.5 RFC OPERATION EXPLAME” T1 rate = Fcpu/256 >> Fcpu/128. 7. Modify “CHARACTERISTIC GRAPHS” Chapter

Table of Content

AMENDENT HISTORY 2
1 PRODUCT OVERVIEW 7
1.1 FEATURES 7
1.2 SYSTEM BLOCK DIAGRAM 8
1.3 PIN ASSIGNMENT 8
1.4 PIN DESCRIPTIONS 10
1.5 PIN CIRCUIT DIAGRAMS 11
2 CENTRAL PROCESSOR UNIT (CPU) 13
2.1 PROGRAM MEMORY (ROM) 13
2.1.1 RESET VECTOR (0000H) 14
2.1.2 INTERRUPT VECTOR (0008H) 15
2.1.3 LOOK-UP TABLE DESCRIPTION 17
2.1.4 JUMP TABLE DESCRIPTION 19
2.1.5 CHECKSUM CALCULATION 21
2.2 DATA MEMORY (RAM) 22
2.2.1 SYSTEM REGISTER 22
2.2.1.1 SYSTEM REGISTER TABLE 22
2.2.1.2 SYSTEM REGISTER DESCRIPTION 22
2.2.1.3 BIT DEFINITION of SYSTEM REGISTER 23
2.2.2 ACCUMULATOR 25
2.2.3 PROGRAM FLAG 26
2.2.4 PROGRAM COUNTER 27
2.2.5 H, L REGISTERS 30
2.2.6 Y, Z REGISTERS 31
2.2.7 R REGISTER 31
2.3 ADDRESSING MODE 32
2.3.1 IMMEDIATE ADDRESSING MODE 32
2.3.2 DIRECTLY ADDRESSING MODE 32
2.3.3 INDIRECTLY ADDRESSING MODE 32
2.4 STACK OPERATION 33
2.4.1 OVERVIEW 33
2.4.2 STACK REGISTERS 34
2.4.3 STACK OPERATION EXAMPLE 35
2.5 CODE OPTION TABLE 36
2.5.1 High_Clk code option 36
2.5.2 Low_Clk code option 36
2.5.3 Fcpu code option 36
2.5.4 Reset_Pin code option 37
2.5.5 Security code option 37
2.5.6 Noise Filter code option 37
3 RESET 38
3.1 OVERVIEW 38
3.2 POWER ON RESET 39
3.3 WATCHDOG RESET 39
3.4 BROWN OUT RESET 39
3.5 THE SYSTEM OPERATING VOLTAGE 40
3.6 LOW VOLTAGE DETECTOR (LVD) 40
3.7 BROWN OUT RESET IMPROVEMENT 42

3.8 EXTERNAL RESET 43
3.9 EXTERNAL RESET CIRCUIT 43
3.9.1 Simply RC Reset Circuit 43
3.9.2 Diode & RC Reset Circuit 44
3.9.3 Zener Diode Reset Circuit 44
3.9.4 Voltage Bias Reset Circuit 45
3.9.5 External Reset IC 45
4 SYSTEM CLOCK 46
4.1 OVERVIEW 46
4.2 FCPU (INSTRUCTION CYCLE) 46
4.3 SYSTEM HIGH-SPEED CLOCK 46
4.3.1 HIGH_CLK CODE OPTION 47
4.3.2 INTERNAL HIGH-SPEED OSCILLATOR 47
4.3.3 EXTERNAL HIGH-SPEED OSCILLATOR 47
4.4 SYSTEM LOW-SPEED CLOCK 48
4.5 OSCM REGISTER 49
4.6 SYSTEM CLOCK MEASUREMENT 49
4.7 SYSTEM CLOCK TIMING 50
5 SYSTEM OPERATION MODE 53
5.1 OVERVIEW 53
5.2 NORMAL MODE 54
5.3 SLOW MODE 54
5.4 POWER DOWN MDOE 54
5.5 GREEN MODE 55
5.6 OPERATING MODE CONTROL MACRO 56
5.7 WAKEUP 57
5.7.1 OVERVIEW 57
5.7.2 WAKEUP TIME 57
5.7.3 P1W WAKEUP CONTROL REGISTER 59
6 INTERRUPT 60
6.1 OVERVIEW 60
6.2 INTEN INTERRUPT ENABLE REGISTER 61
6.3 INTRQ INTERRUPT REQUEST REGISTER 62
6.4 GIE GLOBAL INTERRUPT OPERATION 63
6.5 PUSH, POP ROUTINE 64
6.6 EXTERNAL INTERRUPT OPERATION (INT0) 65
6.7 INT1 (P0.1) INTERRUPT OPERATION 66
6.8 T0 INTERRUPT OPERATION 67
6.9 TC0 INTERRUPT OPERATION 68
6.10 T1 INTERRUPT OPERATION 69
6.11 MULTI-INTERRUPT OPERATION 70
7 I/O PORT 71
7.1 OVERVIEW 71
7.2 I/O PORT MODE 72
7.3 I/O PULL UP REGISTER 74
7.4 I/O PORT DATA REGISTER 75
8 TIMERS 76
8.1 WATCHDOG TIMER 76
8.2 T0 8-BIT BASIC TIMER 78
8.2.1 OVERVIEW 78
8.2.2 T0 Timer Operation 79
8.2.3 T0M MODE REGISTER 80

8.2.4 T0C COUNTING REGISTER 80
8.2.5 T0 TIMER OPERATION EXPLAME 81
8.3 TC0 8-BIT TIMER/COUNTER 82
8.3.1 OVERVIEW 82
8.3.2 TC0 TIMER OPERATION 83
8.3.3 TC0M MODE REGISTER 84
8.3.4 TC0C COUNTING REGISTER 84
8.3.5 TC0R AUTO-RELOAD REGISTER 85
8.3.6 TC0D PWM DUTY REGISTER 85
8.3.7 TC0 EVENT COUNTER 86
8.3.8 PULSE WIDTH MODULATION (PWM) 86
8.3.9 TC0 TIMER OPERATION EXPLAME 87
8.4 T1 16-BIT TIMER WITH CAPTURE TIMER FUNCTION 89
8.4.1 OVERVIEW 89
8.4.2 T1 TIMER OPERATION 90
8.4.3 T1M MODE REGISTER 91
8.4.4 T1CH, T1CL 16-bit COUNTING REGISTERS 92
8.4.5 T1 CPATURE TIMER OPERATION 93
8.4.6 CAPTURE TIMER CONTROL REGISTERS 94
8.4.7 10-bit Event Counter Function 94
8.4.8 T1VCH, T1VCL 10-bit EVENT COUNTER REGISTERS 95
8.4.9 T1 TIMER OPERATION EXPLAME 96
9 RESISTANCE TO FREQURNCY CONVERTER (RFC) 98
9.1 OVERVIEW 98
9.2 RFC APPLICATION CIRCUIT 99
9.3 RFC OPERATION 99
9.4 RFCM REGISTER 100
9.5 RFC OPERATION EXPLAME 101
10 4X32 LCD DRIVER 102
10.1 OVERVIEW 102
10.2 LCD REGISTERS 102
10.3 C-TYPE LCD MODE 103
10.4 R-TYPE LCD MODE 104
10.5 LCD RAM MAP 105
10.6 LCD WAVEFORM 106
11 INSTRUCTION TABLE 107
12 ELECTRICAL CHARACTERISTIC 108
12.1 ABSOLUTE MAXIMUM RATING 108
12.2 ELECTRICAL CHARACTERISTIC 108
12.3 CHARACTERISTIC GRAPHS 109
13 DEVELOPMENT TOOL 110
13.1 SN8P2318 EV-KIT 110
13.2 ICE AND EV-KIT APPLICATION NOTIC 111
14 OTP PROGRAMMING PIN 112
14.1 WRITER TRANSITION BOARD SOCKET PIN ASSIGNMENT 112
14.2 SN8P2317 / SN8P2318 OTP PROGRAMMING TOOL 113
14.3 PROGRAMMING PIN MAPPING: 114
15 MARKING DEFINITION 115
15.1 INTRODUCTION 115
15.2 MARKING INDETIFICATION SYSTEM 115
15.3 MARKING EXAMPLE 115
15.4 DATECODE SYSTEM 116

16 PACKAGE INFORMATION 117
16.1 LQFP 64 PIN 117
16.2 LQFP 48 PIN 118

1 PRODUCT OVERVIEW

1.1 FEATURES

- **Memory configuration**
  ROM size: 4K * 16 bits.
  RAM size: 128 * 8 bits.

- **8 levels stack buffer.**
- **5 interrupt sources**
  3 internal interrupts: T0, TC0, T1
  2 external interrupt: INT0, INT1

- **I/O pin configuration**
  Bi-directional: P0, P1, P5.
  Bi-directional and shared with SEG pins: P2, P3.
  Input only shared with reset pin: P0.3.
  Wakeup: P0, P1 level change.
  Pull-up resisters: P0, P1, P2, P3, P5.
  External interrupt pin:
  P0.0 trigger edge controlled by PEDGE.
  P0.1 level change trigger.
  Event counter input:
  TC0 event counter input pin: P0.0
  T1 event counter input pin: P0.2
  RFC channel pins: P1.0~P1.4.

- **3-Level LVD: 2.0V/2.4V/3.6V**
- **Powerful instructions**
  Instruction's length is one word.
  Most of instructions are one cycle only.
  All ROM area JMP/CALL instruction.
  All ROM area lookup table function (MOVC).

- **Fcpu (Instruction cycle)**
  Fcpu = Fosc/1, Fpsc/2, Fosc/4, Fosc/8, Fosc/16.

- **One 8-bit basic timer. (T0).**
- **One 8-bit timer/counter (TC0) with duty/cycle programmable PWM and IR carry signal.**
- **One 16-bit timer/counter (T1) with auto-reload/ counter/capture timer supports for RFC function.**
- **5-Channel RFC (Resistor to Frequency Converter)**
- **4*32 LCD Driver (128 dots) supports internal C type and external R type bias.**
- **On chip watchdog timer and clock source is Internal low clock RC type (16KHz @3V, 32KHz @5V).**

- **4 system clocks**
  External high clock: RC type up to 10 MHz
  External high clock: Crystal type up to 16 MHz
  Internal high clock: PLL 16MHz from 32K X'tal.
  Internal low clock: RC type 16KHz(3V), 32KHz(5V)

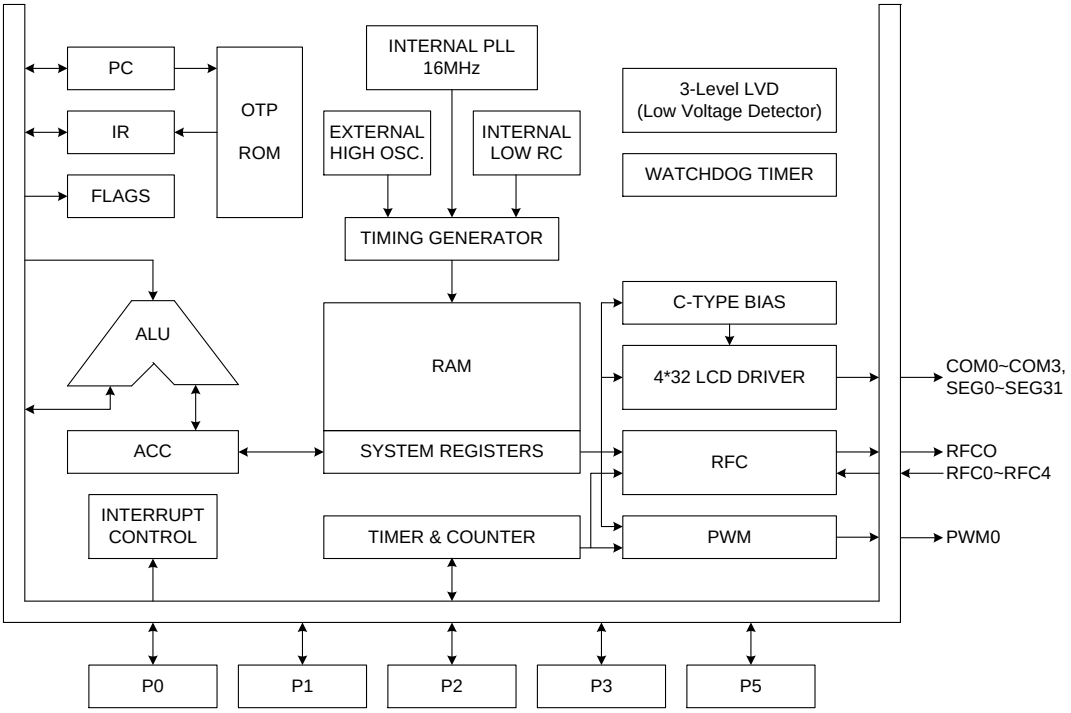
- **4 operating modes**
  Normal mode: Both high and low clock active
  Slow mode: Low clock only.
  Sleep mode: Both high and low clock stop
  Green mode: Periodical wakeup by timer

- **Package (Chip form support)**
  LQFP 48 pin
  LQFP 64 pin

☞ **Features Selection Table**

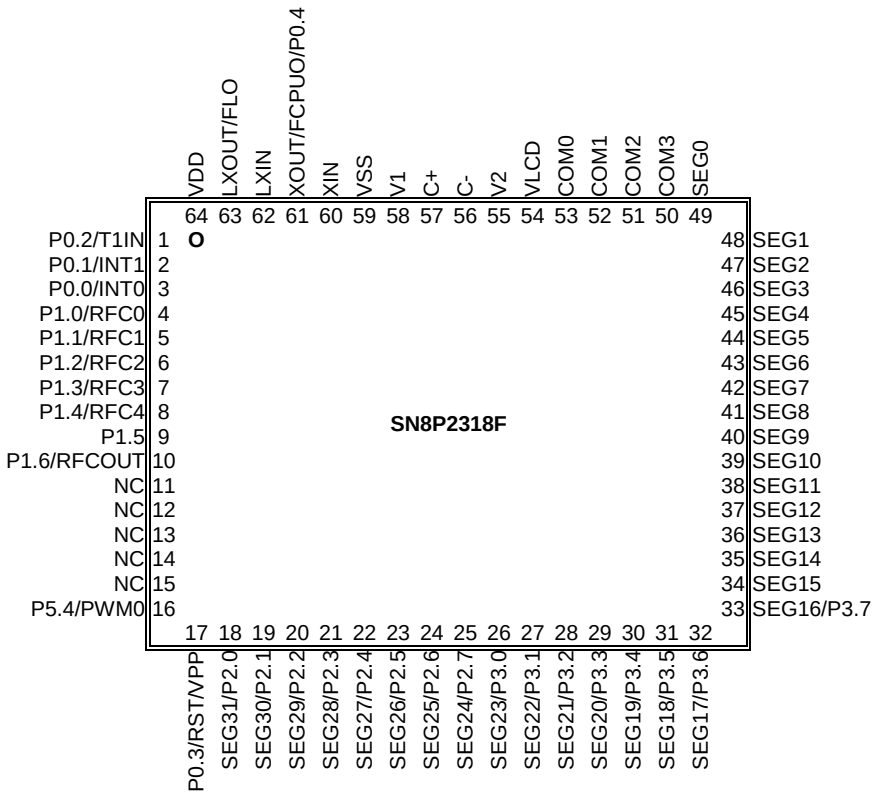
CHIP	ROM	RAM	Stack	Timer			PWM	RFC	Ext. Int	I/O	LCD Driver			Package
				T0	TC0	T1					Dot	C-type	R-type	
SN8P2308	4K*16	128*8	8	V	V	1	1	2-ch	1	25	4*32	V	V	LQFP64
SN8P2318	4K*16	128*8	8	V	V	1	1	5-ch	2	29	4*32	V	V	LQFP64
SN8P2317	4K*16	128*8	8	V	V	1	1	5-ch	2	25	4*21	V	V	LQFP48

1.2 SYSTEM BLOCK DIAGRAM

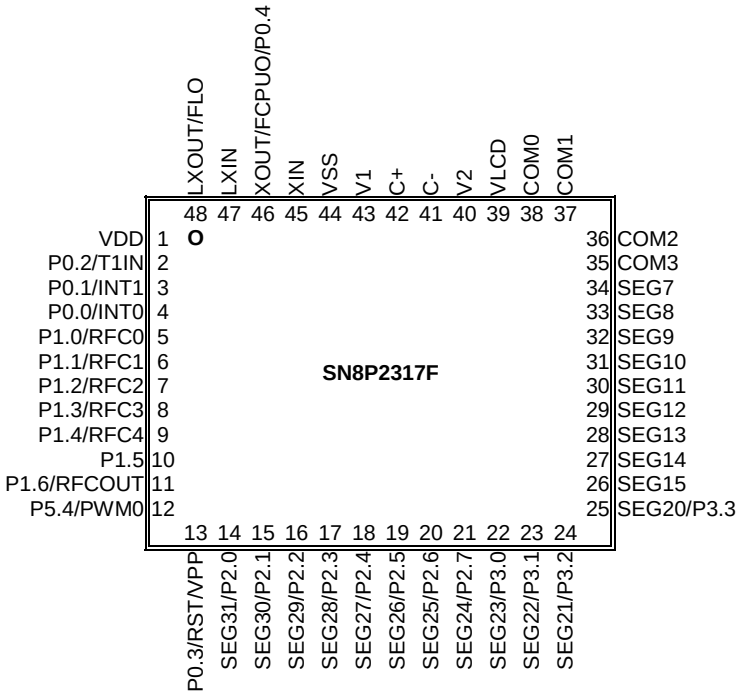


1.3 PIN ASSIGNMENT

SN8P2318F (LQFP 64 pin)



SN8P2317F (LQFP 48 pin)

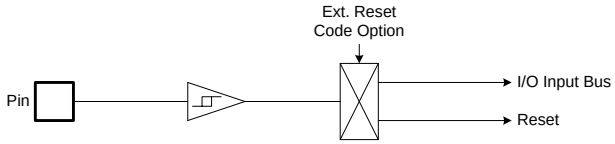


1.4 PIN DESCRIPTIONS

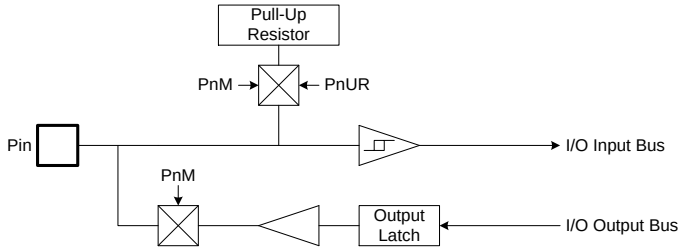
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital and analog circuit.
VLCD	P	LCD power source.
C+, C-	P	LCD C-type charge pump output pins. LCD C-type: Connected a 0.1uF capacitor. LCD R-type: Low status.
V1, V2	P	LCD bias voltage. 1/2 bias: V1 = V2. 1/3 bias: V1 = 1/3*Vdd, V2 = 2/3*Vdd. LCD R-type: Connect external resistors to decide bias voltage and current.
P0.3/RST/VPP	I, P	RST: System external reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”.
		VPP: OTP 12.3V power input pin in programming mode.
		P0.3: Input only pin with Schmitt trigger structure and no pull-up resistor. Level change wake-up.
XIN	I/O	XIN: Oscillator input pin while external oscillator enable (crystal and RC). In PLL_16M mode, XIN connects a decouple 0.1uF capacitor to VDD.
XOUT/P0.4/FCPUO	I/O	XOUT: Oscillator output pin while external crystal enable.
		P0.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		FCPUO: Fcpu clock signal output pin as High_Clk code option=RC.
LXIN	I	LXIN: Low-speed oscillator input pin.
LXOUT/FLO	O	LXOUT: Low-speed oscillator output pin.
		FLO: Low-speed RC type oscillator Flosc freq output pin as Low_Clk code option=RC.
P0.0/INT0	I/O	P0.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		INT0: External interrupt 0 input pin.
		TC0 event counter input pin.
P0.1/INT1	I/O	P0.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		INT1: External interrupt 1 input pin.
P0.2/T1IN	I/O	P0.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		T1 event counter input pin.
P1.0/RFC0	I/O	P1.0: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		RFC0: RFC channel 0.
P1.1/RFC1	I/O	P1.1: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		RFC1: RFC channel 1.
P1.2/RFC2	I/O	P1.2: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		RFC2: RFC channel 2.
P1.3/RFC3	I/O	P1.3: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		RFC3: RFC channel 3.
P1.4/RFC4	I/O	P1.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		RFC4: RFC channel 4.
P1.5	I/O	P1.5: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
P1.6/RFCOUT	I/O	P1.6: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
		RFCOUT: RFC oscillating signal output pin.
P5.4/ PWM0	I/O	P5.4: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters.
		PWM0: Programmable PWM output pin from TC0.
P2[7:0]/ SEG[31:24]	I/O	P2[7:0]: Bi-direction pin. No Schmitt trigger structure. Built-in pull-up resisters.
		SEG[31:24]: LCD segment output pins.
P3[7:0]/ SEG[23:16]	I/O	P3[7:0]: Bi-direction pin. No Schmitt trigger structure. Built-in pull-up resisters.
		SEG[23:16]: LCD segment output pins.
SEG[15:0]	O	SEG[15:0]: LCD segment output pins.
COM[3:0]	O	COM[3:0]: LCD common 0~3 output pins.

1.5 PIN CIRCUIT DIAGRAMS

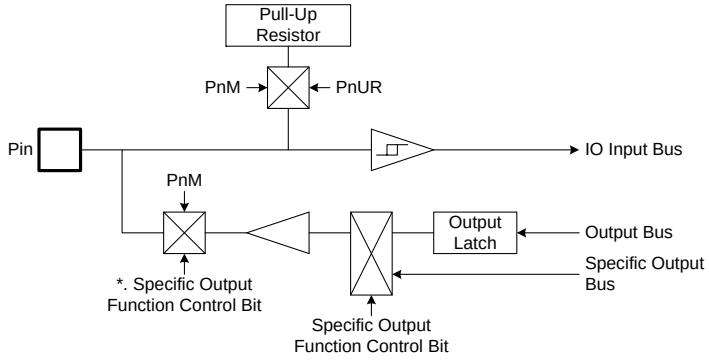
- **Reset shared pin structure:**



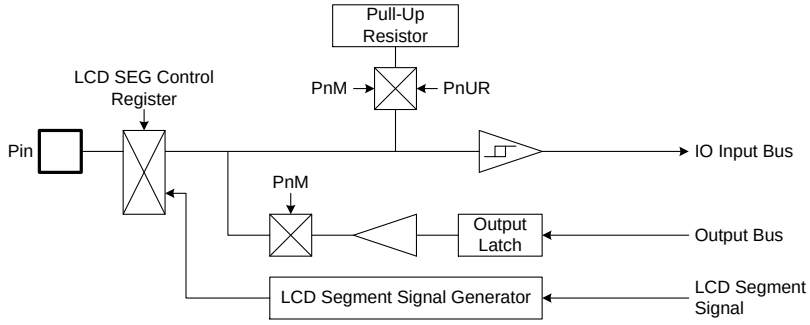
- **GPIO pin structure:**



- **Specific Output Function (e.g. PWM, RFC…) shared pin structure:**



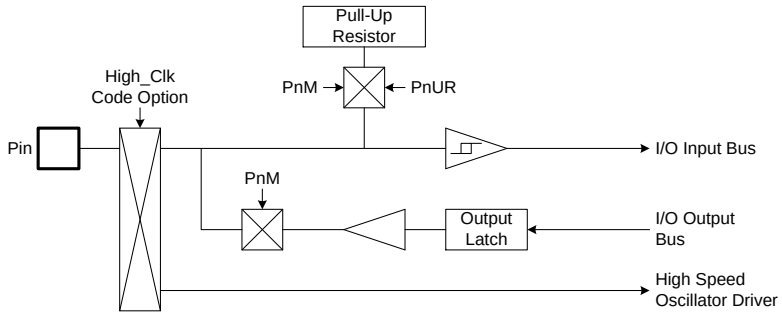
- **LCD SEG shared pin structure:**



- **Oscillator shared pin structure:**

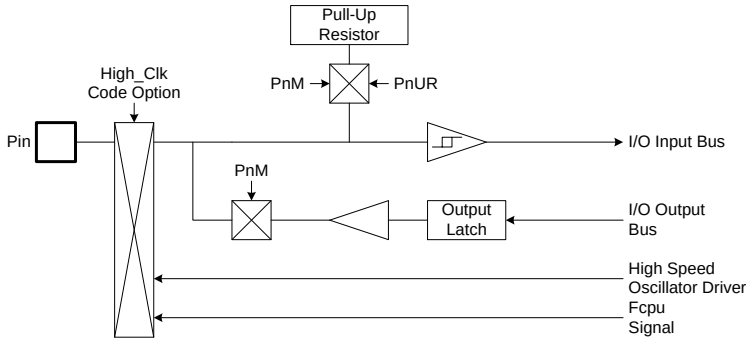
XIN:

Pin Name	Oscillator Code Option	Description
XIN	RC, 4M X'tal, 12M X'tal, PLL_16M	Oscillator input pin.



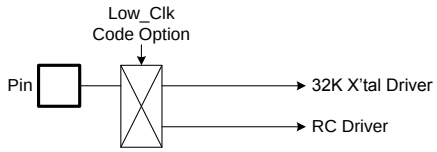
XOUT/FCPUO/P0.4:

Pin Name	Oscillator Code Option	Description
XOUT	4M X'tal, 12M X'tal	Oscillator output pin.
FCPUO	RC	Fcpu signal output pin to measure RC frequency for adjusting RC parameters.
P0.4	PLL_16M	GPIO mode.



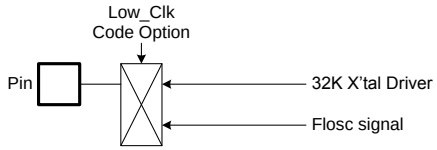
LXIN:

Pin Name	Oscillator Code Option	Description
LXIN	RC, 32K X'tal	Oscillator input pin.



LXOUT/FLO:

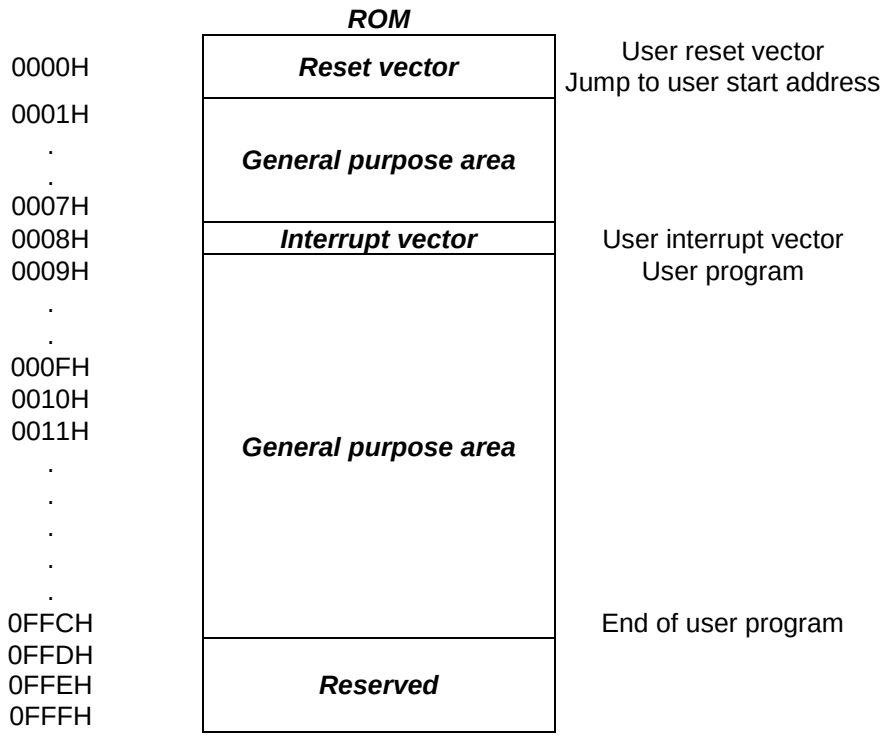
Pin Name	Oscillator Code Option	Description
LXOUT	32K X'tal	Oscillator output pin.
FLO	RC	Flosc signal output pin to measure RC frequency for adjusting RC parameters.



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 PROGRAM MEMORY (ROM)

☞ ***4K words ROM***



The ROM includes Reset vector, Interrupt vector, General purpose area and Reserved area. The Reset vector is program beginning address. The Interrupt vector is the head of interrupt service routine when any interrupt occurring. The General purpose area is main program area including main loop, sub-routines and data table.

2.1.1 RESET VECTOR (0000H)

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

- ☞ **Power On Reset (NT0=1, NPD=0).**
- ☞ **Watchdog Reset (NT0=0, NPD=0).**
- ☞ **External Reset (NT0=1, NPD=1).**

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. It is easy to know reset status from NT0, NPD flags of PFLAG register. The following example shows the way to define the reset vector in the program memory.

➢ **Example: Defining Reset Vector**

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

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


                ENDP                ; End of program
```

2.1.2 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: "PUSH", "POP" instructions save and load ACC/PFLAG without (NT0, NPD). PUSH/POP buffer is a unique buffer and only one level.***

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

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

                ORG     8               ; Interrupt vector.
                PUSH                    ; Save ACC and PFLAG register to buffers.
                …
                …
                POP                     ; Load ACC and PFLAG register from buffers.
                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.**

```
.CODE
        ORG     0           ; 0000H
        JMP     START       ; Jump to user program address.
        …
        ORG     8           ; Interrupt vector.
        JMP     MY_IRQ      ; 0008H, Jump to interrupt service routine address.

        ORG     10H
START:                      ; 0010H, The head of user program.
        …                   ; User program.
        …
        …
        JMP     START       ; End of user program.
        …
MY_IRQ:                     ;The head of interrupt service routine.
        PUSH                ; Save ACC and PFLAG register to buffers.
        …
        …
        POP                 ; Load ACC and PFLAG register from buffers.
        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.***

2.1.3 LOOK-UP TABLE DESCRIPTION

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

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

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

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

➢ **Example: INC_YZ macro.**

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

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

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

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

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

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

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

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

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

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

2.1.4 JUMP TABLE DESCRIPTION

The jump table operation is one of multi-address jumping function. Add low-byte program counter (PCL) and ACC value to get one new PCL. If PCL is overflow after PCL+ACC, PCH adds one automatically. 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).

✸ ***Note: PCH only support PC up counting result and doesn't support PC down counting. When PCL is carry after PCL+ACC, PCH adds one automatically. If PCL borrow after PCL–ACC, PCH keeps value and not change.***

➢ **Example: Jump table.**

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

        B0ADD   PCL, A          ; PCL = PCL + ACC, PCH + 1 when PCL overflow occurs.
        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
```

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

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

```
@JMP_A  MACRO   VAL
        IF      (($+1) !& 0XFF00) !!= (($+(VAL)) !& 0XFF00)
        JMP     ($ | 0XFF)
        ORG     ($ | 0XFF)
        ENDIF
        B0ADD   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
```

2.1.5 CHECKSUM CALCULATION

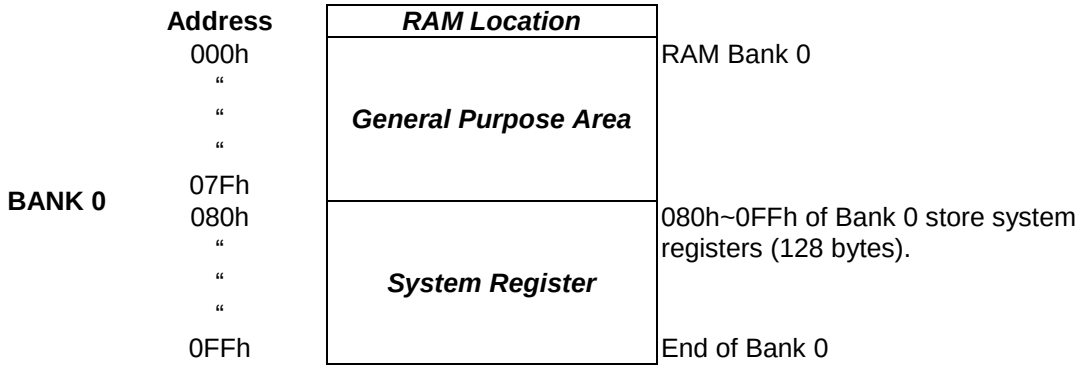
The last ROM address are reserved area. User should avoid these addresses (last address) when calculate the Checksum value.

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

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

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

2.2 DATA MEMORY (RAM)

☞ ***128 X 8-bit RAM***



The 128-byte general purpose RAM is in Bank 0. Sonix provides “Bank 0” type instructions (e.g. b0mov, b0add, b0bts1, b0bset…) to control Bank 0 RAM in non-zero RAM bank condition directly.

2.2.1 SYSTEM REGISTER

2.2.1.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	-	PFLAG	RBANK	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	T1M	T1CL	T1CH	T1VCL	T1VCH	T1CKSM	RFCM	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M	P3M	-	P5M	-	-	INTRQ	INTEN	OSCM	LCDM	WDTR	TC0R	PCL	PCH
D	P0	P1	P2	P3	-	P5	-	-	T0M	T0C	TC0M	TC0C	-	-	-	STKP
E	P0UR	P1UR	P2UR	P3UR	-	P5UR	@HL	@YZ	TC0D	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 SYSTEM REGISTER DESCRIPTION

H, L = Working, @HL addressing register.
R = Working register and ROM look-up data buffer.
RBANK = RAM bank control register.
T1CH,L = T1 counting register.
T1CKSM = T1 capture timer control register.
INTRQ = Interrupt request register.
OSCM = Oscillator mode register.
WDTR = Watchdog timer clear register.
P1W = P1 wake-up control register.
PnM = Port n input/output mode register.
PnUR = Port n pull-up resister control register.
T0C = T0 counting register.
TC0C = TC0 counting register.
TC0D = TC0 duty control register.
@YZ = RAM YZ indirect addressing index pointer.
STK0~STK7 = Stack 0 ~ stack 7 buffer.
Y, Z = Working, @YZ and ROM addressing register.
PFLAG = Special flag register.
T1M = T1 mode register.
T1VCH,L = T1 event counter counting register.
RFCM = RFC mode register.
INTEN = Interrupt enable register.
LCDM = LCD mode register.
PEDGE = P0.0 edge direction register.
PCH, PCL = Program counter.
Pn = Port n data buffer.
T0M = T0 mode register.
TC0M = TC0 mode register.
TC0R = TC0 auto-reload data buffer.
@HL = RAM HL indirect addressing index pointer.
STKP = Stack pointer buffer.

2.2.1.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD	LVD36	LVD24		C	DC	Z	R/W	PFLAG
087H								RBNKS0	R/W	RBANK
0A0H	T1ENB	T1rate2	T1rate1	T1rate0	T1CKS				R/W	T1M
0A1H	T1CL7	T1CL6	T1CL5	T1CL4	T1CL3	T1CL2	T1CL1	T1CL0	R/W	T1CL
0A2H	T1CH7	T1CH6	T1CH5	T1CH4	T1CH3	T1CH2	T1CH1	T1CH0	R/W	T1CH
0A3H	T1VCL7	T1VCL6	T1VCL5	T1VCL4	T1VCL3	T1VCL2	T1VCL1	T1VCL0	R/W	T1VCL
0A4H							T1VCH1	T1VCH0	R/W	T1VCH
0A5H	CPTVC				CPTCKS	CPTStart	CPTG1	CPTG0	R/W	T1CKSM
0A6H	RFCENB		**0**	**1**	RFCOUT	RFCH2	RFCH1	RFCH0	R/W	RFCM
0B8H				P04M		P02M	P01M	P00M	R/W	P0M
0BFH				P00G1	P00G0				R/W	PEDGE
0C0H		P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H		P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M
0C3H	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M	R/W	P3M
0C5H				P54M					R/W	P5M
0C8H		T1IRQ	TC0IRQ	T0IRQ			P01IRQ	P00IRQ	R/W	INTRQ
0C9H		T1IEN	TC0IEN	T0IEN			P01IEN	P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CBH	CPCK1	CPCK0	VLCDCP	PSEG2	PSEG1	PSEG0	BIAS	LCDENB	R/W	LCDM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH					PC11	PC10	PC9	PC8	R/W	PCH
0D0H				P04	P03	P02	P01	P00	R/W	P0
0D1H		P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2
0D3H	P37	P36	P35	P34	P33	P32	P31	P30	R/W	P3
0D5H				P54					R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0				T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS1	TC0CKS0		PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H				P04R		P02R	P01R	P00R	W	P0UR
0E1H		P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2UR
0E3H	P37R	P36R	P35R	P34R	P33R	P32R	P31R	P30R	W	P3UR
0E5H				P54R					W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E8H	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0	R/W	TC0D
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H					S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H					S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H					S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H					S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H					S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH					S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H

0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH					S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH					S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ ***Note:***

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

2.2.2 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
; or
                B0MOV       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. "PUSH", "POP" save and load ACC, PFLAG data into buffers.

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

```
INT_SERVICE:
                PUSH                ; Save ACC and PFLAG to buffers.
                …
                …
                POP                 ; Load ACC and PFLAG from buffers.

                RETI                ; Exit interrupt service vector
```

2.2.3 PROGRAM FLAG

The PFLAG register contains the arithmetic status of ALU operation, system reset status and LVD detecting status. NT0, NPD bits indicate system reset status including power on reset, LVD reset, reset by external pin active and watchdog reset. C, DC, Z bits indicate the result status of ALU operation. LVD24, LVD36 bits indicate LVD detecting power voltage status.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

Bit [7:6] **NT0, NPD:** Reset status flag.

NT0	NPD	Reset Status
0	0	Watch-dog time out
0	1	Reserved
1	0	Reset by LVD
1	1	Reset by external Reset Pin

Bit 5 **LVD36:** LVD 3.6V operating flag and only support LVD code option is LVD_H.
0 = Inactive (VDD > 3.6V).
1 = Active (VDD ≦ 3.6V).

Bit 4 **LVD24:** LVD 2.4V operating flag and only support LVD code option is LVD_M.
0 = Inactive (VDD > 2.4V).
1 = Active (VDD ≦ 2.4V).

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.***

2.2.4 PROGRAM COUNTER

The program counter (PC) is a 12-bit binary counter separated into the high-byte 4 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 11.

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

☞ **ONE ADDRESS SKIPPING**

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

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

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

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

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

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

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

INCS instruction:

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

INCMS instruction:

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

If the destination decreased by 1, which results underflow of 0x01 to 0x00, the PC will add 2 steps to skip next instruction.

DECS instruction:

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

DECMS instruction:

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

☞ **MULTI-ADDRESS JUMPING**

Users can jump around the multi-address by either JMP instruction or ADD M, A instruction (M = PCL) to activate multi-address jumping function. Program Counter supports **"ADD M,A"**, **"ADC M,A"** and **"B0ADD M,A"** instructions for carry to PCH when PCL overflow automatically. For jump table or others applications, users can calculate PC value by the three instructions and don't care PCL overflow problem.

✸ ***Note: PCH only support PC up counting result and doesn't support PC down counting. When PCL is carry after PCL+ACC, PCH adds one automatically. If PCL borrow after PCL–ACC, PCH keeps value and not change.***

➢ **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
                …
                …
```

2.2.5 H, L REGISTERS

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

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

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

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

➢ **Example: If want to read a data from RAM address 20H of bank_0, it can use indirectly addressing mode to access data as following.**

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

➢ **Example: Clear general-purpose data memory area of bank 0 using @HL register.**

```
        CLR     H           ; H = 0, bank 0
        B0MOV   L, #07FH    ; L = 7FH, the last address of the data memory area
CLR_HL_BUF:
        CLR     @HL         ; Clear @HL to be zero
        DECMS   L           ; L – 1, if L = 0, finish the routine
        JMP     CLR_HL_BUF  ; Not zero

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

2.2.6 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_CLR:                        ; End of clear general purpose data memory area of bank 0
        …
```

2.2.7 R REGISTER

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.***

2.3 ADDRESSING MODE

2.3.1 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.***

2.3.2 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.
```

2.3.3 INDIRECTLY ADDRESSING MODE

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

- **Example: Indirectly addressing mode with @HL register**

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

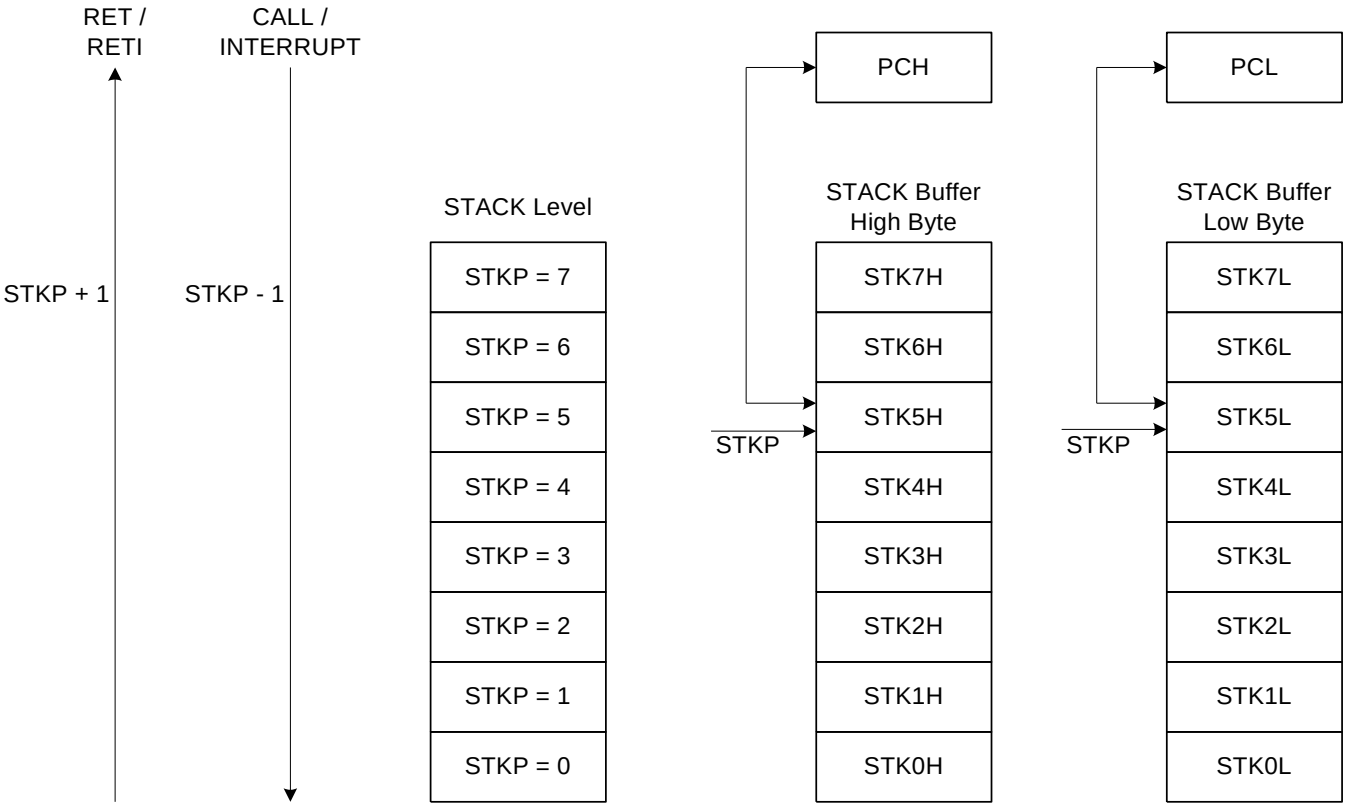
- **Example: Indirectly addressing mode with @YZ register**

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

2.4 STACK OPERATION

2.4.1 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.



2.4.2 STACK REGISTERS

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

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

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

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

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

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

➢ **Example: Stack pointer (STKP) reset, we strongly recommended to clear the stack pointers 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	-	-	-	SnPC12	SnPC11	SnPC10	SnPC9	SnPC8
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	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)

2.4.3 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	-

2.5 CODE OPTION TABLE

The code option is the system hardware configurations including oscillator type, watchdog timer operation, Noise Filter option, LVD option, reset pin option and OTP ROM security control. The code option items are as following table:

Code Option	Content	Function Description
High_Clk	PLL_16M	High speed internal 16MHz PLL. External low-speed 32K oscillator free runs. XOUT pin is bi-direction GPIO mode. XIN pin connects a 0.1uF decouple capacitor to VDD.
	RC	Low cost RC for external high clock oscillator. XIN pin is connected to RC oscillator. XOUT pin is Fcpu signal output.
	12M X'tal	High speed crystal /resonator (e.g. 12MHz) for external high clock oscillator.
	4M X'tal	Standard crystal /resonator (e.g. 4M) for external high clock oscillator.
Low_Clk	32K X'tal	Low frequency, power saving crystal (e.g. 32.768KHz) for external low clock oscillator. LXIN/LXOUT pins drive external 32768Hz low speed crystal/ceramic oscillator.
	RC	LXIN pin drives external RC oscillator. LXOUT pin outputs Flosc clock.
Fcpu	Fhosc/1	Instruction cycle is 1 oscillator clocks.
	Fhosc/2	Instruction cycle is 2 oscillator clocks.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
	Fhosc/16	Instruction cycle is 16 oscillator clocks.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
Reset_Pin	Reset	Enable External reset pin.
	P03	Enable P0.3 input only without pull-up resister.
Noise_Filter	Enable	Enable Noise_Filter.
	Disable	Disable Noise_Filter.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
LVD	LVD_L	LVD will reset chip if VDD is below 2.0V
	LVD_M	LVD will reset chip if VDD is below 2.0V Enable LVD24 bit of PFLAG register for 2.4V low voltage indicator.
	LVD_H	LVD will reset chip if VDD is below 2.4V Enable LVD36 bit of PFLAG register for 3.6V low voltage indicator.
	LVD_MAX	LVD will reset chip if VDD is below 3.6V

2.5.1 High_Clk code option

High_Clk code option control system high speed oscillator type including PLL_16M, RC, 4M X'tal and 12M X'tal. In PLL_16M mode, LXIN/LXOUT connects 32KHz crystal or RC oscillator, XIN connects 0.1uF decouple capacitor, and XOUT pin is Fcpu signal output.

2.5.2 Low_Clk code option

Low_Clk code option control system low speed oscillator type including 32K X'tal and RC. In RC mode, LXIN connects RC oscillator, and LXOUT pin outputs Flosc signal for measuring RC frequency.

2.5.3 Fcpu code option

Fcpu means instruction cycle of normal mode (high clock). In slow mode, the system clock source is external low speed 32KHz oscillator connected to LXIN/LXOUT pins. The Fcpu of slow mode isn't controlled by Fcpu code option and fixed Flosc/4.

2.5.4 Reset_Pin code option

The reset pin is shared with general input only pin controlled by code option.

- **Reset:** The reset pin is external reset function. When falling edge trigger occurring, the system will be reset.
- **P03:** Set reset pin to general input only pin (P0.3). The external reset function is disabled and the pin is input pin.

2.5.5 Security code option

Security code option is OTP ROM protection. When enable security code option, the ROM code is secured and not dumped complete ROM contents.

2.5.6 Noise Filter code option

Noise Filter code option is a power noise filter manner to reduce noisy effect of system clock. If noise filter enable, In high noisy environment, enable noise filter, enable watchdog timer and select a good LVD level can make whole system work well and avoid error event occurrence.

✷ ***Note: When code option High_Clk = PLL_16M / Low_Clk = 32K X'tal and watchdog = Enable / Always_On, the system is unused with sleep mode (support green mode).***

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 (only supports external reset pin enable situation)

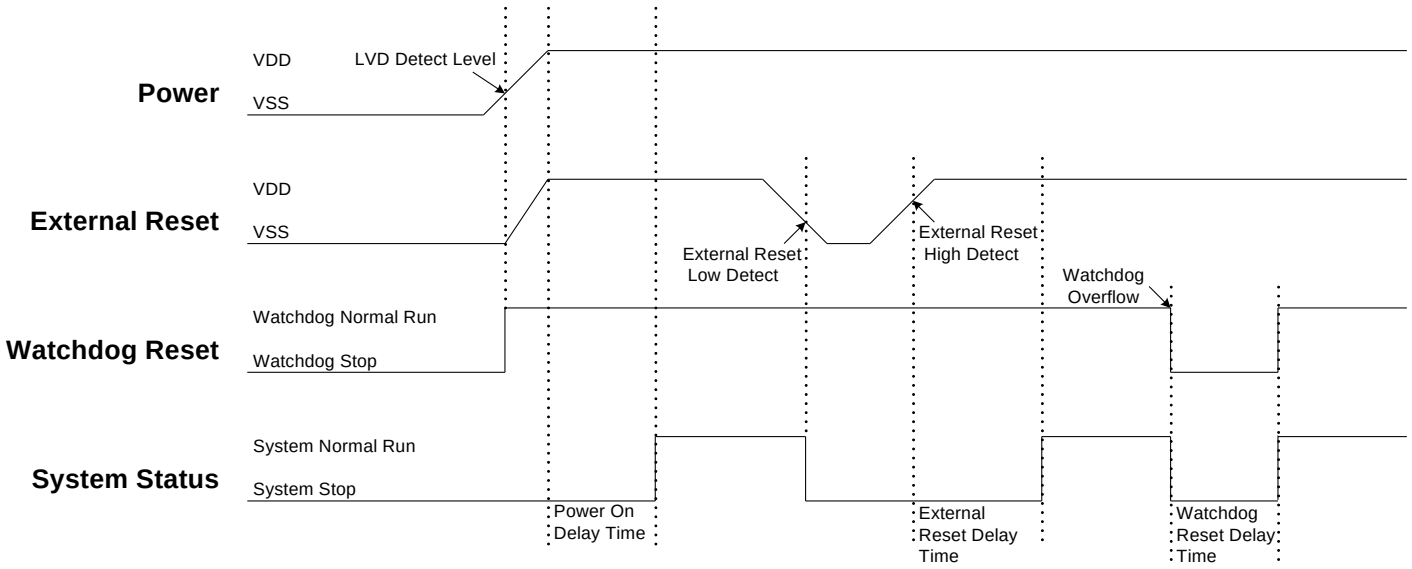
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status and go to different paths by program.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

Bit [7:6] **NT0, NPD:** Reset status flag.

NT0	NPD	Condition	Description
0	0	Watchdog reset	Watchdog timer overflow.
0	1	Reserved	-
1	0	Power on reset and LVD reset.	Power voltage is lower than LVD detecting level.
1	1	External reset	External reset pin detect low level status.

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 (only external reset pin enable):** 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.

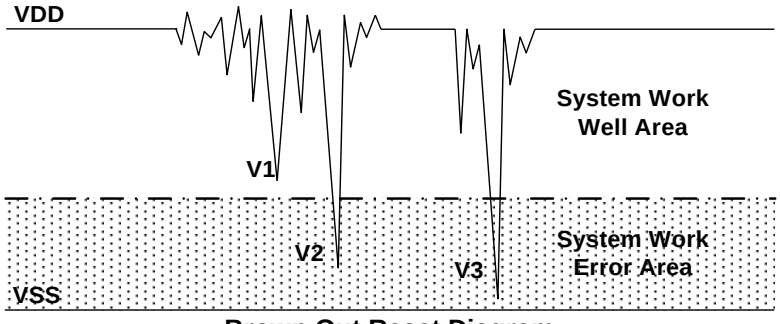
Watchdog timer application note is as following.

- 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

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

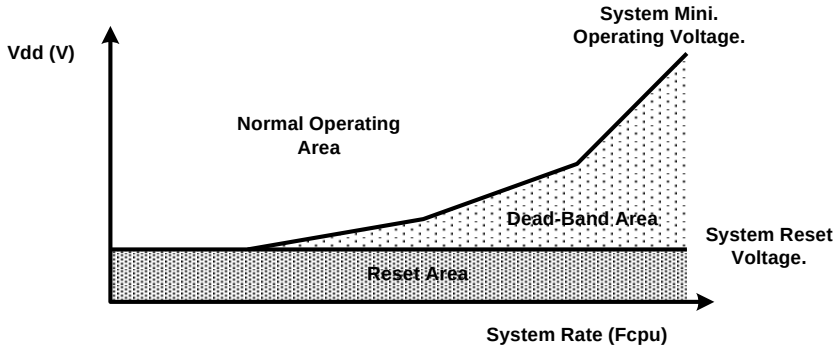
The power dropping might through the voltage range that's the system dead-band. The dead-band means the power range can't offer the system minimum operation power requirement. The above diagram is a typical brown out reset diagram. There is a serious noise under the VDD, and VDD voltage drops very deep. 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.

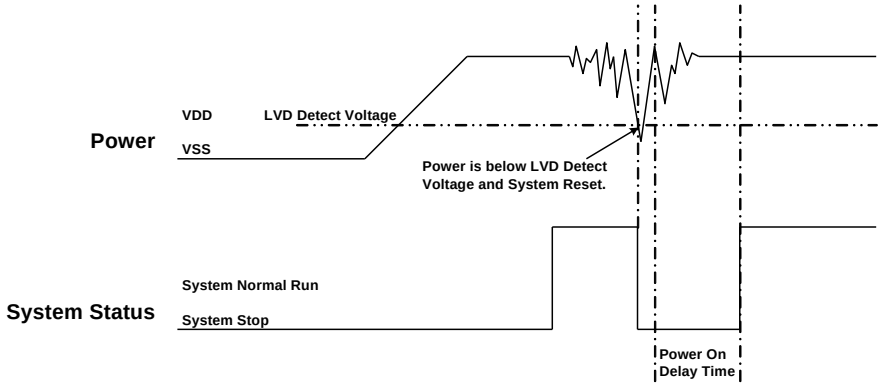
3.5 THE SYSTEM OPERATING VOLTAGE

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.

3.6 LOW VOLTAGE DETECTOR (LVD)



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.

The LVD is three levels design (2.0V/2.4V/3.6V) and controlled by LVD code option. The 2.0V LVD is always enable for power on reset and Brown Out reset. The 2.4V LVD includes LVD reset function and flag function to indicate VDD status function. The 3.6V includes flag function to indicate VDD status. LVD flag function can be an **easy low battery detector**. LVD24, LVD36 flags indicate VDD voltage level. For low battery detect application, only checking LVD24, LVD36 status to be battery status. This is a cheap and easy solution.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

Bit 5 **LVD36:** LVD 3.6V operating flag and only support LVD code option is LVD_H.
0 = Inactive (VDD > 3.6V).
1 = Active (VDD ≦ 3.6V).

Bit 4 **LVD24:** LVD 2.4V operating flag and only support LVD code option is LVD_M.
0 = Inactive (VDD > 2.4V).
1 = Active (VDD ≦ 2.4V).

LVD	LVD Code Option			
	LVD_L	LVD_M	LVD_H	LVD_MAX
2.0V Reset	Available	Available	Available	Available
2.4V Flag	-	Available	-	-
2.4V Reset	-	-	Available	-
3.6V Flag	-	-	Available	-
3.6V Reset	-	-	-	Available

LVD_L
If VDD < 2.0V, system will be reset.
Disable LVD24 and LVD36 bit of PFLAG register.
LVD_M
If VDD < 2.0V, system will be reset.
Enable LVD24 bit of PFLAG register. If VDD > 2.4V, LVD24 is "0". If VDD ≦2.4V, LVD24 flag is "1".
Disable LVD36 bit of PFLAG register.
LVD_H
If VDD < 2.4V, system will be reset.
Enable LVD24 bit of PFLAG register. If VDD > 2.4V, LVD24 is "0". If VDD ≦2.4V, LVD24 flag is "1".
Enable LVD36 bit of PFLAG register. If VDD > 3.6V, LVD36 is "0". If VDD ≦3.6V, LVD36 flag is "1".
LVD_MAX
If VDD < 3.6V, system will be reset.

✸ ***Note:***
1. ***After any LVD reset, LVD24, LVD36 flags are cleared.***
2. ***The voltage level of LVD 2.4V or 3.6V is for design reference only. Don't use the LVD indicator as precision VDD measurement.***

3.7 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.***

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. Watchdog timer application note is as following.

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.

3.8 EXTERNAL RESET

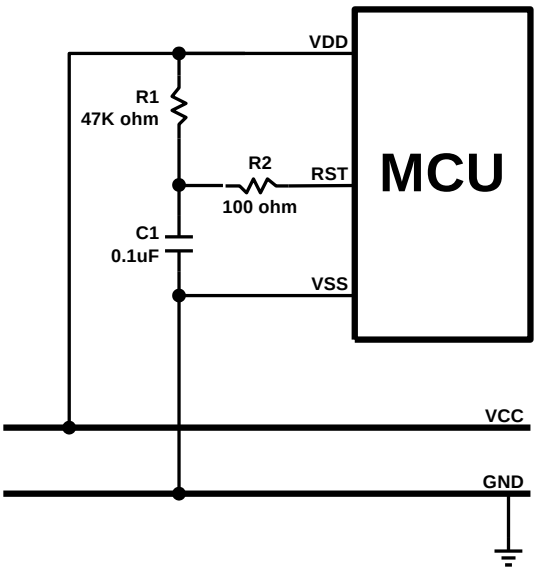
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the low voltage and the system is reset. The external reset operation actives in power on and normal running mode. During system power-up, the external reset pin must be high level input, or the system keeps in reset status. External reset sequence is as following.

- **External reset (only external reset pin enable):** 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…

3.9 EXTERNAL RESET CIRCUIT

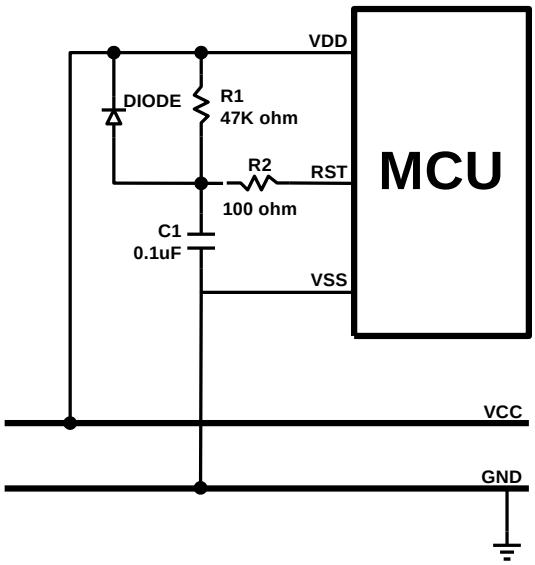
3.9.1 Simply RC Reset Circuit



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.***

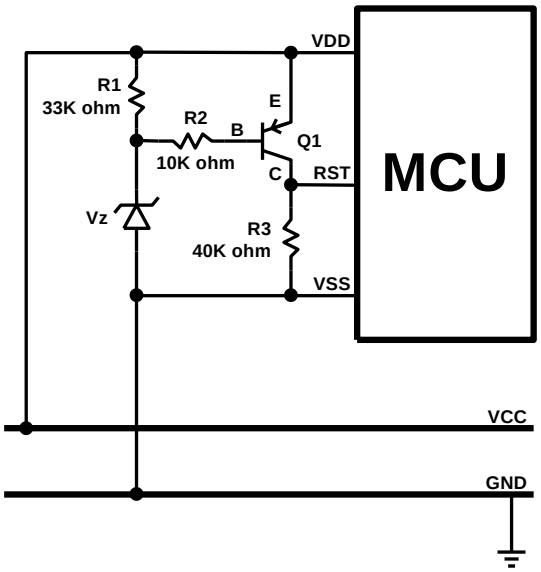
3.9.2 Diode & RC Reset Circuit



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.

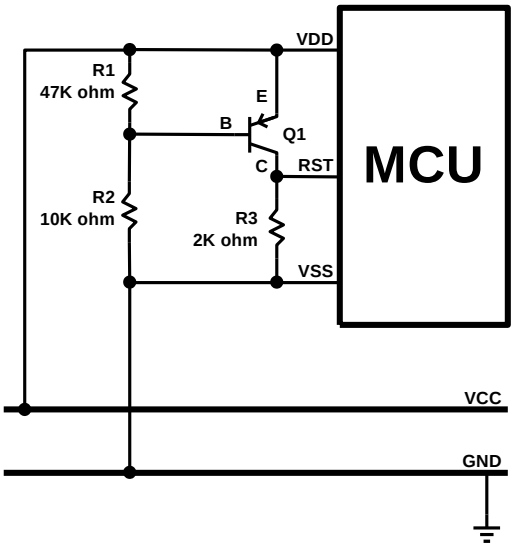
✷ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.9.3 Zener Diode Reset Circuit



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

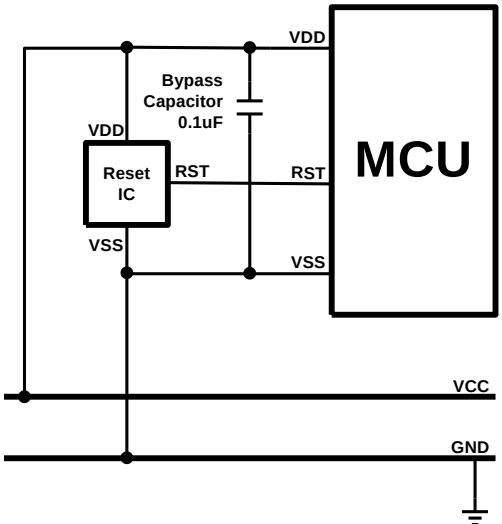
3.9.4 Voltage Bias Reset Circuit



The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to "0.7V x (R1 + R2) / R1", the C terminal of the PNP transistor outputs high voltage and MCU operates normally. When VDD is below "0.7V x (R1 + R2) / R1", the C terminal of the PNP transistor outputs low voltage and MCU is in reset mode.
Decide the reset detect voltage by R1, R2 resistances. Select the right R1, R2 value to conform the application. In the circuit diagram condition, the MCU's reset pin level varies with VDD voltage variation, and the differential voltage is 0.7V. If the VDD drops and the voltage lower than reset pin detect level, the system would be reset. If want to make the reset active earlier, set the R2 > R1 and the cap between VDD and C terminal voltage is larger than 0.7V. The external reset circuit is with a stable current through R1 and R2. For power consumption issue application, e.g. DC power system, the current must be considered to whole system power consumption.

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

3.9.5 External Reset IC



The external reset circuit also use external reset IC to enhance MCU reset performance. This is a high cost and good effect solution. By different application and system requirement to select suitable reset IC. The reset circuit can improve all power variation.

4 SYSTEM CLOCK

4.1 OVERVIEW

The micro-controller is a dual clock system including high-speed and low-speed clocks. The high-speed clock includes internal high-speed oscillator and external oscillators selected by “High_Clk” code option. The low-speed clock is external low-speed oscillator. Both high-speed clock and external low-speed clock can be system clock source through a divider to decide the system clock rate.

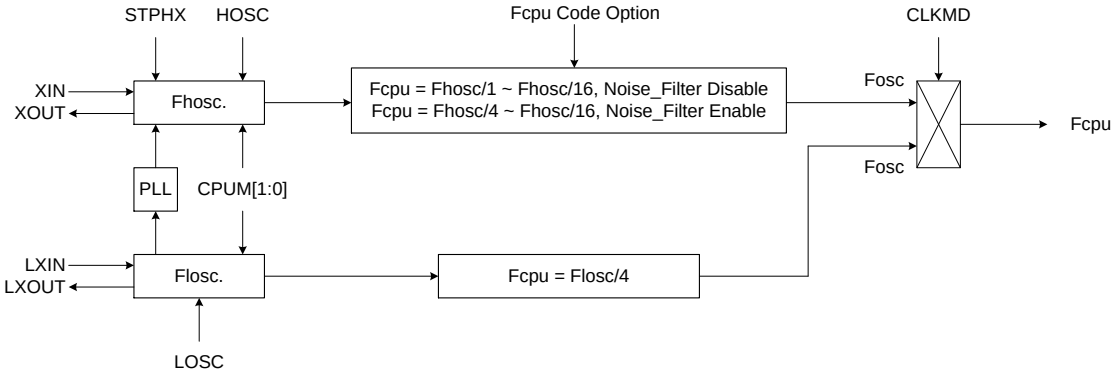
- **High-speed oscillator**

Internal high-speed oscillator is 16MHz PLL type called “**PLL_16M**”.
External high-speed oscillator includes crystal/ceramic (4MHz, 12MHz) and RC type.

- **Low-speed oscillator**

External low-speed oscillator includes crystal/ceramic (32KHz) and RC type.

- **System clock block diagram**



- HOSC: High_Clk code option.
- LOSC: Low_Clk code option.
- Fhosc: External high-speed clock / Internal PLL clock.
- Flosc: External low-speed clock.
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.2 FCPU (INSTRUCTION CYCLE)

The system clock rate is instruction cycle called “**Fcpu**” which is divided from the system clock source and decides the system operating rate. Fcpu rate is selected by Fcpu code option and the range is **Fhosc/1~Fhosc/16** under system normal mode. If the system high clock source is external 4MHz crystal, and the Fcpu code option is Fhosc/4, the Fcpu frequency is 4MHz/4 = 1MHz. Under system slow mode, the Fcpu is fixed Flosc/4, 32KHz/4=8KHz.

4.3 SYSTEM HIGH-SPEED CLOCK

The system high-speed clock has internal and external two-type. The external high-speed clock includes 4MHz, 12MHz, crystal/ceramic and RC type. The internal high-speed clock is internal PLL 16MHz oscillator. These high-speed oscillators are selected by “**High_Clk**” code option.

4.3.1 HIGH_CLK CODE OPTION

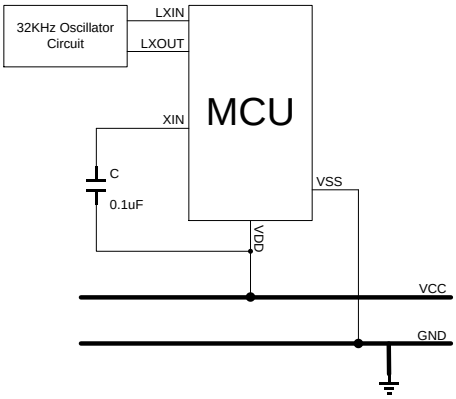
For difference clock functions, Sonix provides multi-type system high clock options controlled by “High_Clk” code option. The High_Clk code option defines the system oscillator types including PLL_16M, RC, 12M X’tal and 4M X’tal. These oscillator options support different bandwidth oscillator.

- **PLL_16M:** The system high-speed clock source is internal PLL high-speed 16MHz type oscillator. The PLL is pumped from external 32KHz low-speed oscillator (LXINT/XOUT).
- **RC:** The system high-speed clock source is external low cost RC type oscillator. The RC oscillator circuit only connects to XIN pin, and the XOUT pin is Fcpu signal output.
- **12M X’tal:** The system high-speed clock source is external high-speed crystal/ceramic. The oscillator bandwidth is 10MHz~16MHz.
- **4M X’tal:** The system high-speed clock source is external high-speed crystal/resonator. The oscillator bandwidth is 1MHz~10MHz.

4.3.2 INTERNAL HIGH-SPEED OSCILLATOR

The internal high-speed oscillator is 16MHz PLL type. The accuracy is ±2% under commercial condition. When the “High_Clk” code option is “PLL_16M”, the internal high-speed oscillator is enabled.

- **PLL_16M:** The system high-speed clock is internal 16MHz oscillator RC type. LXIN/LXOUT pins connects external low-speed 32KHz oscillator. XIN pin connects a 0.1uF decouple capacitor to VDD. XOUT pin is GPIO structure.
- **INTERNAL HIGH-SPEED OSCILLATOR APPLICATION CIRCUIT**

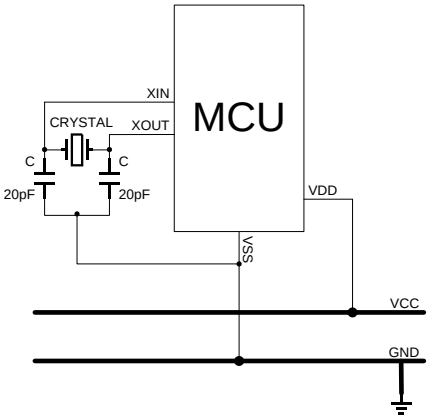


4.3.3 EXTERNAL HIGH-SPEED OSCILLATOR

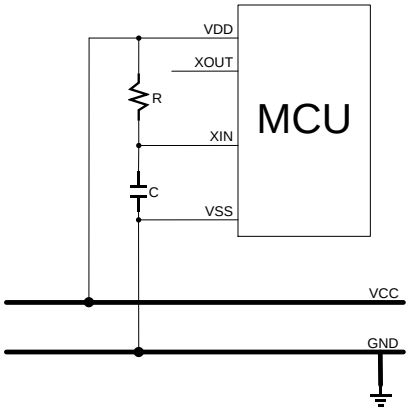
The external high-speed oscillator includes 4MHz, 12MHz and RC type controlled by “High_Clk” code option. The 4MHz and 12MHz oscillators support crystal and ceramic types connected to XIN/XOUT pins with 20pF capacitors to ground. The RC type is a low cost RC circuit only connected to XIN pin. The capacitance is not below 100pF, and use the resistance to decide the frequency.

- **EXTERNAL OSCILLATOR APPLICATION CIRCUIT**

CRYSTAL/CERAMIC:



RC Type:



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

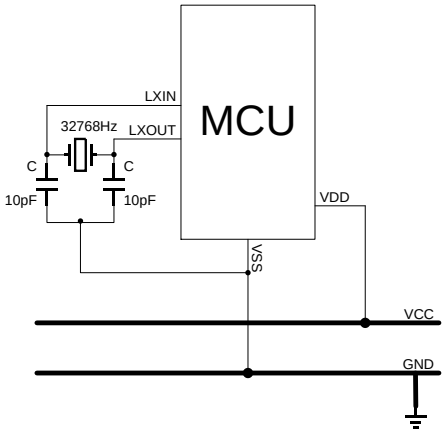
4.4 SYSTEM LOW-SPEED CLOCK

The external low-speed oscillator includes 32KHz and RC type controlled by "Low_Clk" code option. The 32KHz oscillator supports crystal and ceramic types connected to LXIN/LXOUT pins with 10pF capacitors to ground. The RC type is a low cost RC structure, builds in R (resistor), and connects external C (capacitor) with LXIN pin. The capacitance is 27pF @3V and 39pF @5V at 32KHz. LXOUT pin outputs Flosc = 32KHz signal for adjust the capacitance by RC type.

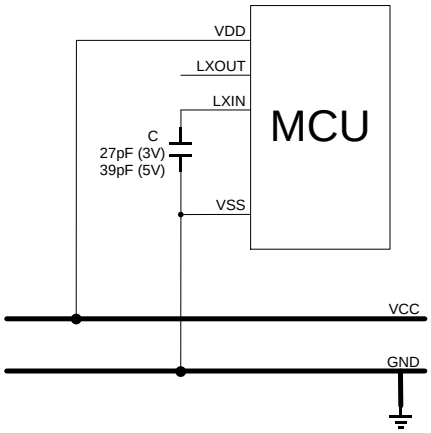
- **32K X'tal:** The system low-speed clock source is external low-speed 32768Hz crystal.
- **RC:** The system low-speed clock source is external low cost RC type oscillator. The RC oscillator circuit only connects one capacitor to LXIN pin, and the LXOUT pin outputs 32KHz signal.

- **EXTERNAL OSCILLATOR APPLICATION CIRCUIT**

32KHz CRYSTAL/CERAMIC:



RC Type:



✸ ***Note: Connect the Crystal/Ceramic and C as near as possible to the LXIN/LXOUT/VSS pins of micro-controller. Connect the C as near as possible to the VSS pin of micro-controller.***

4.5 OSCM REGISTER

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

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

Bit 1 **STPHX:** System high-speed oscillator (external high-speed oscillator and internal PLL) control bit.
0 = System high-speed oscillator free run.
1 = Disable system high-speed oscillator. System low-speed oscillator keeps running.

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

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

"STPHX" bit controls internal high speed PLL oscillator and external oscillator operations. When "STPHX=0", the external oscillator or internal high speed PLL oscillator active. When "STPHX=1", the external oscillator or internal high speed PLL oscillator are disabled. The STPHX function is depend on different high clock options to do different controls.

- **PLL_16M: "STPHX=1" disables internal high speed PLL oscillator.**
- **RC, 4M, 12M: "STPHX=1" disables external oscillator.**

4.6 SYSTEM CLOCK MEASUREMENT

The frequency of RC type oscillator can be measured from clock output pin for adjusting R and C parameters.

- In high-speed RC mode (High_Clk code option = RC), the system clock signal outputs from "FCPUO" pin. The clock rate is instruction cycle through system pre-scaler.
- In low-speed RC mode (Low_Clk code option = RC), the low-speed oscillator's frequency (Flosc) signal outputs from "FLO" pin.
- The system clock also can be measured by software.

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

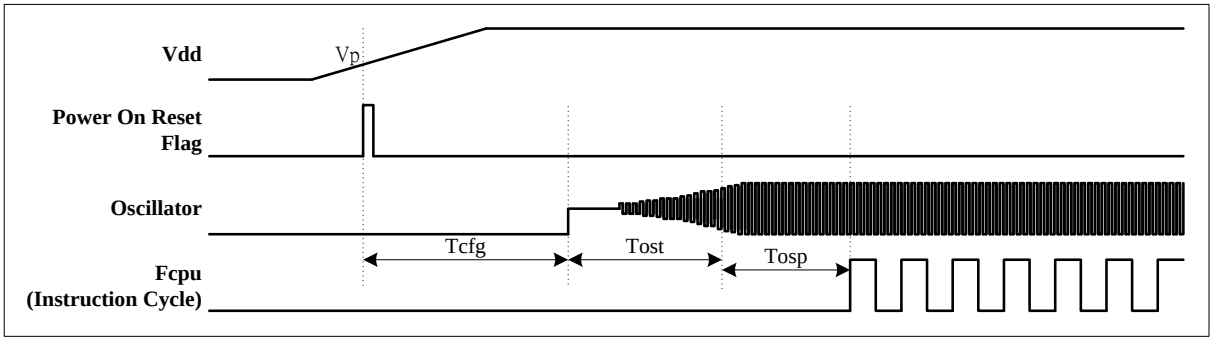
```
        B0BSET    P0M.0       ; Set P0.0 to be output mode for outputting Fcpu toggle signal.
@@:
        B0BSET    P0.0        ; Output Fcpu toggle signal in low-speed clock mode.
        B0BCLR    P0.0        ; Measure the Fcpu frequency by oscilloscope.
        JMP       @B
```

✸ ***Note: Do not measure the RC frequency directly from XIN pin of external high-speed RC mode and LXIN pin of external low-speed RC mode; the probe impendence will affect the RC frequency.***

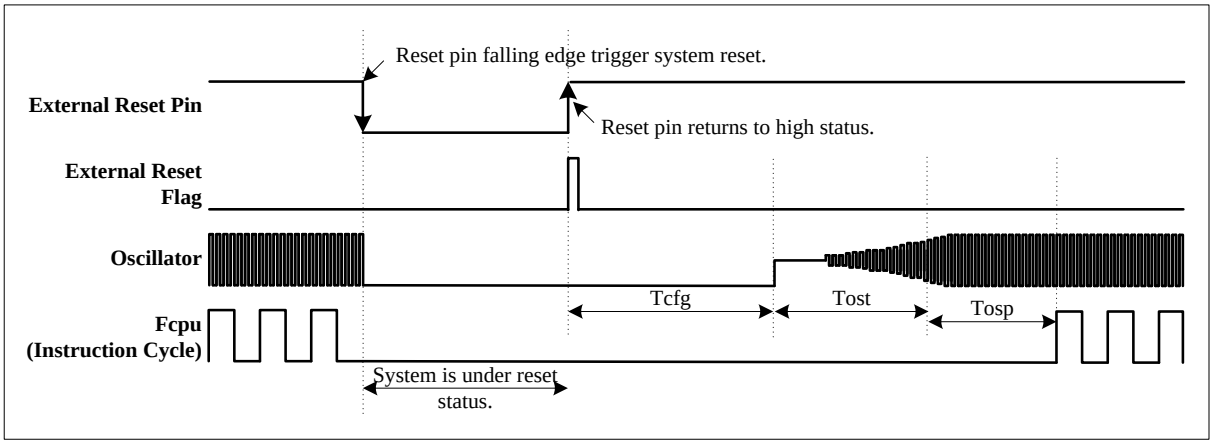
4.7 SYSTEM CLOCK TIMING

Parameter	Symbol	Description	Typical
Hardware configuration time	Tcfg	2048*F_{ILRC}. (The hardware configuration time clock source is internal low-speed RC oscillator whose frequency is 16KHz @3V, 32KHz @5V)	64ms @ F_{ILRC} = 32KHz 128ms @ F_{ILRC} = 16KHz
Oscillator start up time	Tost	The start-up time is depended on oscillator's material, factory and architecture. The RC type oscillator's start-up time is very short and ignored.	-
Oscillator warm-up time	Tosp	Oscillator warm-up time of reset condition (Power on reset, LVD reset, watchdog reset, external reset pin active and wake-up from power down mode): **External high-speed crystal 4M/12M modes: 2048*F_{hosc} (Reset modes).** **External high-speed crystal 4M/12M modes: 2560*F_{hosc} (Wake-up modes)** External high-speed RC modes: 32*F_{hosc} External low-speed 32KHz crystal mode: (2^{14} +256)*F_{losc} External low-speed RC modes: 32*F_{losc} Internal PLL 16MHz oscillator warm-up time is external low-speed oscillator warm-up time + 256 low-speed clock. ex. PLL 16MHz in external low-speed crystal mode: (2^{14} +256)*F_{losc} PLL 16MHz in external low-speed RC mode: 288*F_{losc}	**High-speed oscillator:** 8us @ F_{hosc} = 4MHz RC 128us @ F_{hosc} = 16MHz X'tal (Reset modes) 512us @ F_{hosc} = 4MHz X'tal (Reset modes) 160us @ F_{hosc} = 16MHz X'tal (Wake-up modes) 640us @ F_{hosc} = 4MHz X'tal (Wake-up modes) **Low-speed oscillator:** 0.52sec @ F_{losc} = 32KHz X'tal 1ms @ F_{losc} = 32KHz RC **PLL 16MHz:** 0.52sec @ F_{losc} = 32KHz X'tal 9ms @ F_{losc} = 32KHz RC

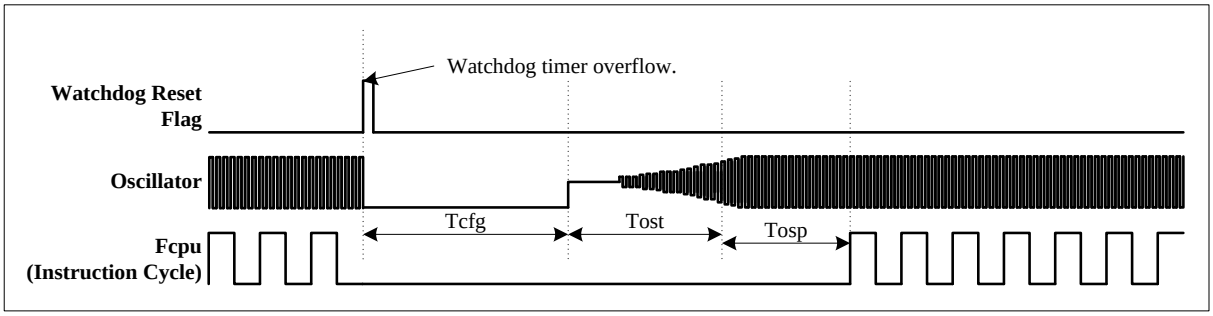
- **Power On Reset Timing**



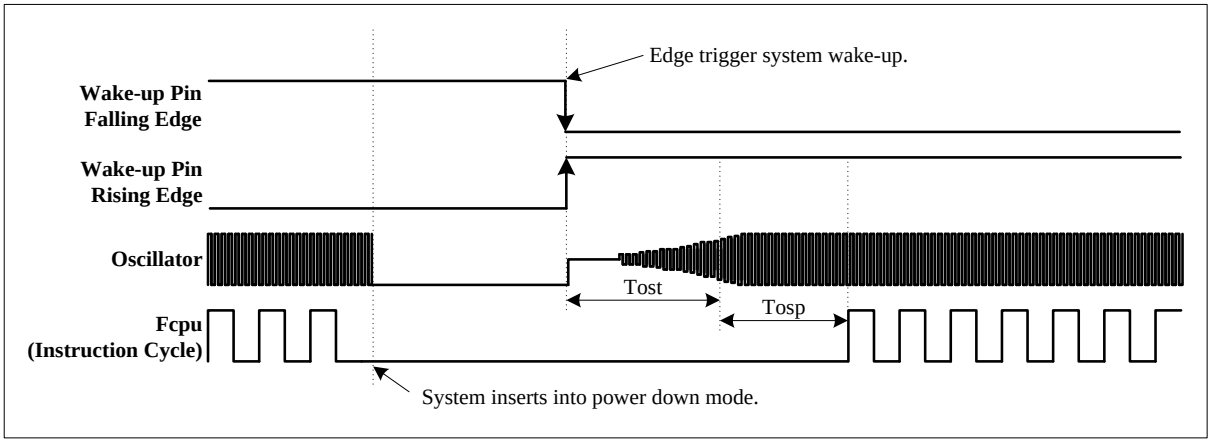
- **External Reset Pin Reset Timing**



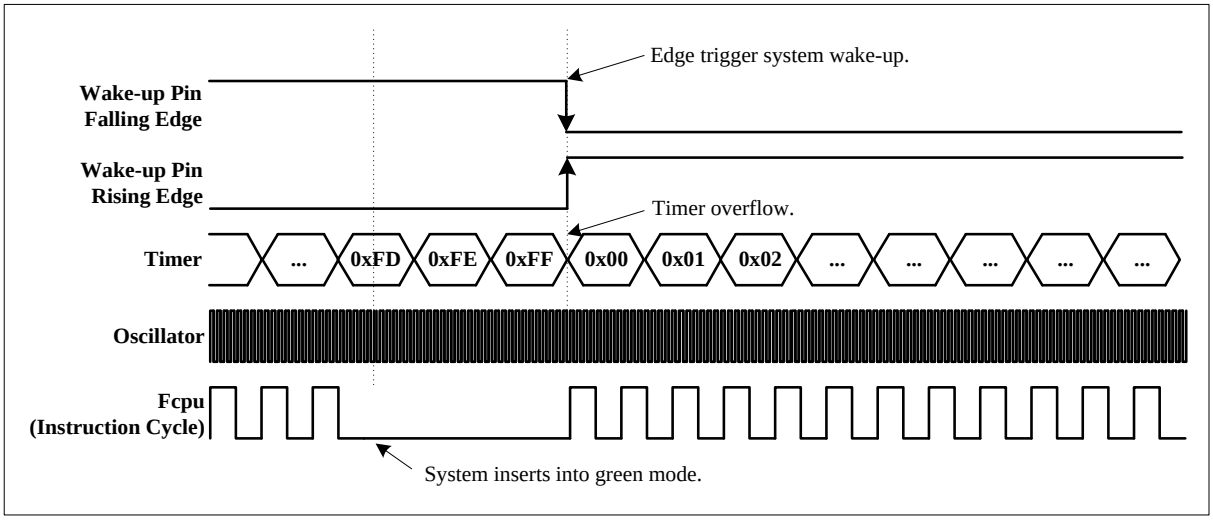
Watchdog Reset Timing



Power Down Mode Wake-up Timing

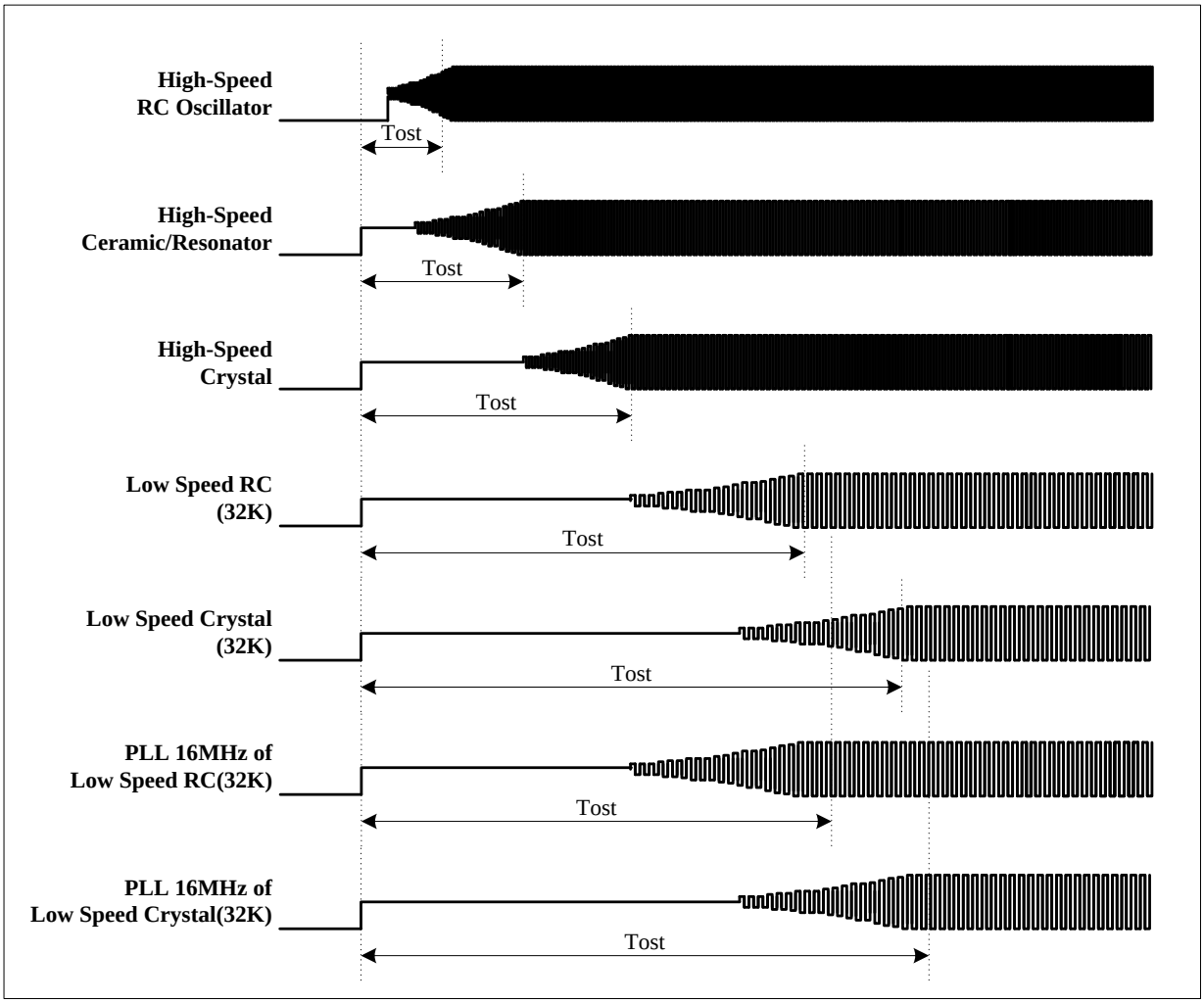


Green Mode Wake-up Timing



Oscillator Start-up Time

The start-up time is depended on oscillator's material, factory and architecture. The RC type oscillator's start-up time is very short and ignored.



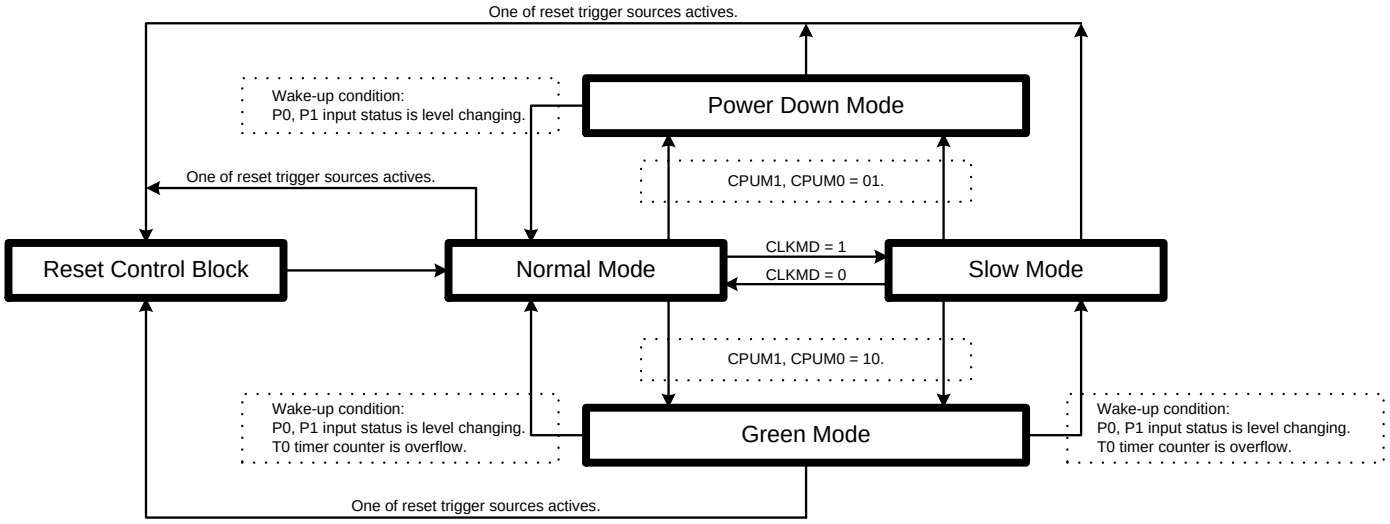
5 SYSTEM OPERATION MODE

5.1 OVERVIEW

The chip builds in four operating mode for difference clock rate and power saving reason. These modes control oscillators, op-code operation and analog peripheral devices' operation.

- Normal mode: System high-speed operating mode.
- Slow mode: System low-speed operating mode.
- Power down mode: System power saving mode (Sleep mode).
- Green mode: System idle mode.

Operating Mode Control Block



Operating Mode Clock Control Table

Operating Mode	Normal Mode	Slow Mode	Green Mode	Power Down Mode
EHOSC/IHOSC	Running	By STPHX	By STPHX	Stop
ELOSC	Running	Running	Running	Stop
CPU instruction	Executing	Executing	Stop	Stop
T0 timer	By T0ENB	By T0ENB	By T0ENB	Inactive
TC0 timer	By TC0ENB	By TC0ENB	By TC0ENB PWM active	Inactive
T1 timer	By T1ENB	By T1ENB	Inactive	Inactive
LCD Driver	By LCDENB	By LCDENB	By LCDENB	Inactive
RFC function	By RFCENB	By RFCENB	By RFCENB	Inactive
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option
Internal interrupt	All active	All active	T0	All inactive
External interrupt	All active	All active	All active	All inactive
Wakeup source	-	-	P0, P1, T0, Reset	P0, P1, Reset

- **EHOSC:** External high-speed oscillator (XIN/XOUT).
- **IHOSC:** Internal high-speed PLL oscillator.
- **ELOSC:** External low-speed oscillator (LXIN/LXOUT).

5.2 NORMAL MODE

The Normal Mode is system high clock operating mode. The system clock source is from high-speed oscillator. The program is executed. After power on and any reset trigger released, the system inserts into normal mode to execute program. When the system is wake-up from power down mode, the system also inserts into normal mode. In normal mode, both of high-speed oscillator and low-speed oscillator active, and the power consumption is largest of all operating modes.

- The program is executed, and full functions are controllable.
- The system rate is high-speed.
- The high-speed oscillator and low-speed oscillator active.
- Normal mode can be switched to other operating modes through OSCM register.
- Power down mode is wake-up to normal mode.
- Slow mode is switched to normal mode.
- Green mode from normal mode is wake-up to normal mode.

5.3 SLOW MODE

The slow mode is system low clock operating mode. The system clock source is from external low-speed oscillator including crystal type and RC type. The slow mode is controlled by CLKMD bit of OSCM register. When CLKMD=0, the system is in normal mode. When CLKMD=1, the system inserts into slow mode. The high-speed oscillator won't be disabled automatically after switching to slow mode, and must be disabled by SPTHX bit to reduce power consumption. In slow mode, the system rate is fixed Flosc/4 (Flosc is external low-speed crystal type and RC type oscillator frequency).

- The program is executed, and full functions are controllable.
- The system rate is low speed (Flosc/4).
- The external low-speed oscillator actives, and the high-speed oscillator is controlled by STPHX=1. In slow mode, to stop high speed oscillator is strongly recommendation.
- Slow mode can be switched to other operating modes through OSCM register.
- Power down mode from slow mode is wake-up to normal mode.
- Normal mode is switched to slow mode.
- Green mode from slow mode is wake-up to slow mode.

5.4 POWER DOWN MDOE

The power down mode is the system idle status. No program execution and oscillator operation. Whole chip is under low power consumption status under 1uA. The power down mode is waked up by P0, P1 hardware level change trigger. P1 wake-up function is controlled by P1W register. Any operating modes into power down mode, the system is waked up to normal mode. Inserting power down mode is controlled by CPUM0 bit of OSCM register. When CPUM0=1, the system inserts into power down mode. After system wake-up from power down mode, the CPUM0 bit is disabled (zero status) automatically.

- The program stops executing, and full functions are disabled.
- All oscillators including external high-speed oscillator, internal high-speed PLL oscillator and external low-speed oscillator stop.
- The power consumption is under 1uA.
- The system inserts into normal mode after wake-up from power down mode.
- The power down mode wake-up source is P0 and P1 level change trigger.

✷ ***Note: If the system is in normal mode, to set STPHX=1 to disable the high clock oscillator. The system is under no system clock condition. This condition makes the system stay as power down mode, and can be wake-up by P0, P1 level change trigger.***

5.5 GREEN MODE

The green mode is another system idle status not like power down mode. In power down mode, all functions and hardware devices are disabled. But in green mode, the system clock source keeps running, so the power consumption of green mode is larger than power down mode. In green mode, the program isn't executed, but the timer with wake-up function actives as enabled, and the timer clock source is the non-stop system clock. The green mode has 2 wake-up sources. One is the P0, P1 level change trigger wake-up. The other one is internal timer with wake-up function occurring overflow. That's mean users can setup one period to timer, and the system is waked up until the time out. Inserting green mode is controlled by CPUM1 bit of OSCM register. When CPUM1=1, the system inserts into green mode. After system wake-up from green mode, the CPUM1 bit is disabled (zero status) automatically.

- The program stops executing, and full functions are disabled.
- Only the timer with wake-up function actives.
- The oscillator to be the system clock source keeps running, and the other oscillators operation is depend on system operation mode configuration.
- If inserting green mode from normal mode, the system insets to normal mode after wake-up.
- If inserting green mode from slow mode, the system insets to slow mode after wake-up.
- The green mode wake-up sources are P0, P1 level change trigger and unique time overflow.
- LCD, PWM and RFC functions active in green mode, but the timer can't wake-up the system as overflow.

✸ ***Note: Sonix provides "GreenMode" macro to control green mode operation. It is necessary to use "GreenMode" macro to control system inserting green mode.***
The macro includes three instructions. Please take care the macro length as using BRANCH type instructions, e.g. bts0, bts1, b0bts0, b0bts1, ins, incms, decs, decms, cmprs, jmp, or the routine would be error.

5.6 OPERATING MODE CONTROL MACRO

Sonix provides operating mode control macros to switch system operating mode easily.

Macro	Length	Description
SleepMode	1-word	The system insets into Sleep Mode (Power Down Mode).
GreenMode	3-word	The system inserts into Green Mode.
SlowMode	2-word	The system inserts into Slow Mode and stops high speed oscillator.
Slow2Normal	5-word	The system returns to Normal Mode from Slow Mode. The macro includes operating mode switch, enable high speed oscillator, high speed oscillator warm-up delay time.

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

```
SleepMode                   ; Declare “SleepMode” macro directly.
```

➢ **Example: Switch normal mode to slow mode.**

```
SlowMode                    ; Declare “SlowMode” macro directly.
```

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

```
Slow2Normal                 ; Declare “Slow2Normal” macro directly.
```

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

```
GreenMode                   ; Declare “GreenMode” macro directly.
```

➢ **Example: Switch normal/slow mode to green mode and enable T0 wake-up 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
        GreenMode               ; Declare “GreenMode” macro directly.
```

➢ **Example: Switch normal/slow mode to green mode and enable T0 wake-up function with RTC.**

```
        CLR         T0C         ; Clear T0 counter.
        B0BSET      FT0TB       ; Enable T0 RTC function.
        B0BSET      FT0ENB      ; To enable T0 timer.

; Go into green mode
        GreenMode               ; Declare “GreenMode” macro directly.
```

5.7 WAKEUP

5.7.1 OVERVIEW

Under power down mode (sleep mode) or green mode, program doesn't execute. The wakeup trigger can wake the system up to normal mode or slow mode. The wakeup trigger sources are external trigger (P0/P1 level change) and internal trigger (T0 timer overflow).

- Power down mode is waked up to normal mode. The wakeup trigger is only external trigger (P0/P1 level change)
- Green mode is waked up to last mode (normal mode or slow mode). The wakeup triggers are external trigger (P0/P1 level change) and internal trigger (T0 timer overflow).

5.7.2 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 a period for stabling the oscillator circuit. After the wakeup time, the system goes into the normal mode.

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

The wake-up time of the external high-speed (12M_X'tal, 4M_C'tal) crystal type oscillator is as the following.

The Wakeup time = 1/Fhosc * 2560 (sec) + high clock start-up time

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

The wakeup time = 1/Fhosc * 2560 = 0.64 ms (4MHz crystal)
The total wakeup time = 0.64 ms + oscillator start-up time

The wake-up time of the external high/low speed RC type oscillator is as the following.

The Wakeup time = 1/Fosc * 32 (sec) + clock start-up time

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

The wakeup time = 1/Fhosc * 32 = 8 us (4MHz RC)
The total wakeup time = 8 us + oscillator start-up time

The wakeup time = 1/Fhosc * 32 = 1 ms (32KHz RC)
The total wakeup time = 1 ms + oscillator start-up time

The wake-up time of the external low-speed crystal type oscillator is as the following.

The Wakeup time (32K_X'tal) = 1/Flosc * (2^{14}+256) + low clock start-up time

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

The wakeup time = 1/Flosc * (2^{14} +256) = 0.52 sec (32KHz crystal)
The total wakeup time = 0.52 sec + oscillator start-up time

The wake-up time of the internal high-speed PLL 16MHz oscillator is as the following.

The Wakeup time of 32KHz RC type oscillator mode = 1/Flosc * 288 (sec) + clock start-up time

The Wakeup time of 32KHz crystal type oscillator mode = 1/Flosc * (2^{14}+256) (sec) + clock start-up time

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

The wakeup time = 1/Flosc * 288 = 9 ms (32KHz RC)
The total wakeup time = 9 ms + oscillator start-up time

The wakeup time = 1/Flosc * (2^{14}+256) = 0.52 sec (32KHz crystal)
The total wakeup time = 0.52 sec + oscillator start-up time

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

5.7.3 P1W WAKEUP CONTROL REGISTER

Under power down mode (sleep mode) and green mode, the I/O ports with wakeup function are able to wake the system up to normal mode. The wake-up trigger edge is level changing. When wake-up pin occurs rising edge or falling edge, the system is waked up by the trigger edge. The Port 0 and Port 1 have wakeup function. Port 0 wakeup function always enables, but the Port 1 is controlled by the P1W register.

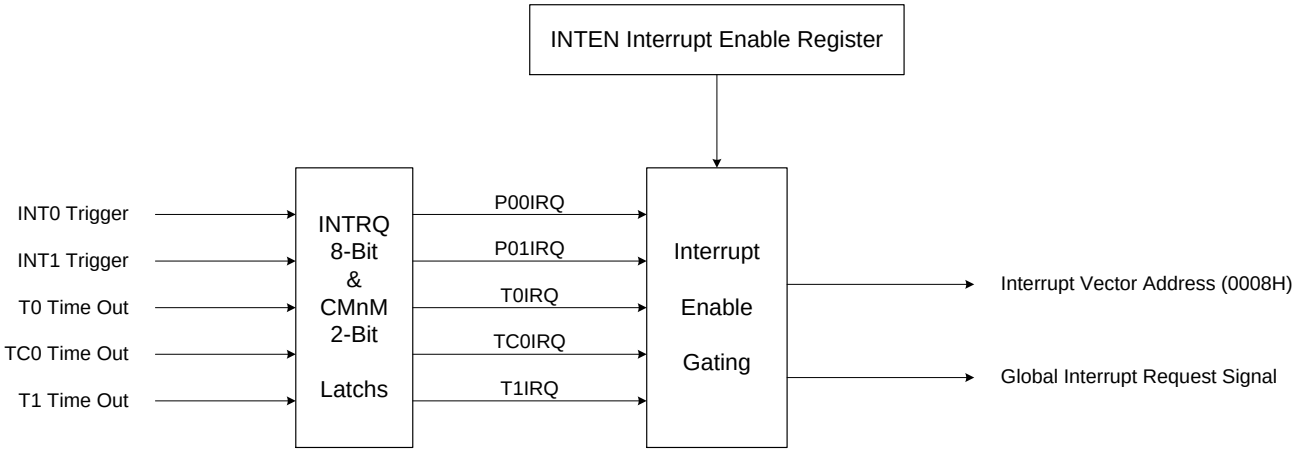
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	-	P16W	P15W	P14W	P13W	P12W	P11W	P10W
Read/Write	-	W	W	W	W	W	W	W
After reset	-	0	0	0	0	0	0	0

Bit[6:0] **P10W~P16W:** Port 1 wakeup function control bits.
0 = Disable P1n wakeup function.
1 = Enable P1n wakeup function.

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 5 interrupt sources, including 3 internal interrupt (T0/TC0/T1) and 2 external interrupt (INT0/INT1). The external interrupt can wake-up the chip while the system is switched from power down mode to high-speed normal mode, and interrupt request is latched until return to normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to “0” for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to “1” to accept the next interrupts’ request. The interrupt request signals are stored in INTRQ register.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	-	T1IEN	TC0IEN	T0IEN	-	-	P01IEN	P00IEN
Read/Write	-	R/W	R/W	R/W	-	-	R/W	R/W
After reset	-	0	0	0	-	-	0	0

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

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

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

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

Bit 6 **T1IEN:** T1 timer interrupt control bit.
0 = Disable T1 interrupt function.
1 = Enable T1 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	-	T1IRQ	TC0IRQ	T0IRQ	-	-	P01IRQ	P00IRQ
Read/Write	-	R/W	R/W	R/W	-	-	R/W	R/W
After reset	-	0	0	0	-	-	0	0

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

Bit 1 **P01IRQ:** External P0.1 interrupt (INT1) request flag.
0 = None INT1 interrupt request.
1 = INT1 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.

Bit 6 **T1IRQ:** T1 timer interrupt request flag.
0 = None T1 interrupt request.
1 = T1 interrupt request.

6.4 GIE GLOBAL INTERRUPT OPERATION

GIE is the global interrupt control bit. All interrupts start work after the GIE = 1 It is necessary for interrupt service request. One of the interrupt requests occurs, and the program counter (PC) points to the interrupt vector (ORG 8) and the stack add 1 level.

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

Bit 7 **GIE:** Global interrupt control bit.
0 = Disable global interrupt.
1 = Enable global interrupt.

Example: Set global interrupt control bit (GIE).

```
B0BSET      FGIE            ; Enable GIE
```

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

6.5 PUSH, POP ROUTINE

When any interrupt occurs, system will jump to ORG 8 and execute interrupt service routine. It is necessary to save ACC, PFLAG data. The chip includes “PUSH”, “POP” for in/out interrupt service routine. The two instructions save and load **ACC**, **PFLAG** data into buffers and avoid main routine error after interrupt service routine finishing.

✷ ***Note: ”PUSH”, “POP” instructions save and load ACC/PFLAG without (NT0, NPD). PUSH/POP buffer is an unique buffer and only one level.***

➢ **Example: Store ACC and PAFLG data by PUSH, POP instructions when interrupt service routine executed.**

```
                ORG     0
                JMP     START

                ORG     8
                JMP     INT_SERVICE

                ORG     10H
START:
                …

INT_SERVICE:
                PUSH                    ; Save ACC and PFLAG to buffers.
                …
                …
                POP                     ; Load ACC and PFLAG from buffers.

                RETI                    ; Exit interrupt service vector
                …
                ENDP
```

6.6 EXTERNAL INTERRUPT OPERATION (INT0)

INT0 is external interrupt trigger source and builds in edge trigger configuration function. When the external edge trigger occurs, the external interrupt request flag will be set to “1” no matter the external interrupt control bit enabled or disable. When external interrupt control bit is enabled and external interrupt edge trigger is occurring, the program counter will jump to the interrupt vector (ORG 8) and execute interrupt service routine.

The external interrupt builds in wake-up latch function. That means when the system is triggered wake-up from power down mode, the wake-up source is external interrupt source (P0.0), and the trigger edge direction matches interrupt edge configuration, the trigger edge will be latched, and the system executes interrupt service routine fist after wake-up.

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

Bit[4:3] **P00G[1:0]:** INT0 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

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

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

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

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

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

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

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

6.7 INT1 (P0.1) INTERRUPT OPERATION

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

If the interrupt trigger direction is identical with wake-up trigger direction, the INT1 interrupt request flag (INT1IRQ) is latched while system wake-up from power down mode or green mode by P0.1 wake-up trigger. System inserts to interrupt vector (ORG 8) after wake-up immediately.

✸ ***Note: INT1 interrupt request can be latched by P0.1 wake-up trigger.***

✸ ***Note: The interrupt trigger direction of P0.1 is falling edge.***

➢ **Example: INT1 interrupt request setup.**

```
        B0BSET      FP01IEN     ; Enable INT1 interrupt service
        B0BCLR      FP01IRQ     ; Clear INT1 interrupt request flag
        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: INT1 interrupt service routine.**

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

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

        B0BTS1      FP01IRQ     ; Check P01IRQ
        JMP         EXIT_INT    ; P01IRQ = 0, exit interrupt vector

        B0BCLR      FP01IRQ     ; Reset P01IRQ
        …                       ; INT1 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.8 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. Fcpu = 4MHz / 4.**

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

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

        B0BSET      FGIE        ; Enable GIE
```

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

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

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

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

        B0BCLR      FT0IRQ      ; Reset T0IRQ
        MOV         A, #64H
        B0MOV       T0C, A      ; Reset T0C.
        …                       ; T0 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.


        RETI                    ; Exit interrupt vector
```

6.9 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 TC0IEN is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: TC0 interrupt request setup. Fcpu = 16MHz / 16.**

```
        B0BCLR      FTC0IEN         ; Disable TC0 interrupt service
        B0BCLR      FTC0ENB         ; Disable TC0 timer
        MOV         A, #20H         ;
        B0MOV       TC0M, A         ; Set TC0 clock = Fcpu / 64
        MOV         A, #64H         ; Set TC0C initial value = 64H
        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, #64H
        B0MOV       TC0C, A         ; Reset TC0C.
        …                           ; TC0 interrupt service routine
        …
EXIT_INT:
        …                           ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

6.10 T1 INTERRUPT OPERATION

When the T1C (T1CH, T1CL) counter occurs overflow, the T1IRQ will be set to “1” however the T1IEN is enable or disable. If the T1IEN = 1, the trigger event will make the T1IRQ to be “1” and the system enter interrupt vector. If the T1IEN = 0, the trigger event will make the T1IRQ to be “1” but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

➢ **Example: T1 interrupt request setup.**

```
        B0BCLR      FT1IEN      ; Disable T1 interrupt service
        B0BCLR      FT1ENB      ; Disable T1 timer
        MOV         A, #20H     ;
        B0MOV       T1M, A      ; Set T1 clock = Fcpu / 32.
        CLR         T1CH
        CLR         T1CL

        B0BSET      FT1IEN      ; Enable T1 interrupt service
        B0BCLR      FT1IRQ      ; Clear T1 interrupt request flag
        B0BSET      FT1ENB      ; Enable T1 timer

        B0BSET      FGIE        ; Enable GIE
```

Example: T1 interrupt service routine.

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

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

        B0BTS1      FT1IRQ      ; Check T1IRQ
        JMP         EXIT_INT    ; T1IRQ = 0, exit interrupt vector

        B0BCLR      FT1IRQ      ; Reset T1IRQ
        B0MOV       A, T1CH
        B0MOV       T1CHBUF, A
        B0MOV       A, T1CL
        B0MOV       T1CLBUF, A  ; Save pulse width.
        CLR         T1CH
        CLR         T1CL
        …                       ; T1 interrupt service routine
        …
EXIT_INT:
        POP                     ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.11 MULTI-INTERRUPT OPERATION

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

Interrupt Name	*Trigger Event Description*
P00IRQ	P0.0 trigger controlled by PEDGE
P01IRQ	P0.1 falling edge trigger
T0IRQ	T0C overflow
TC0IRQ	TC0C overflow
T1IRQ	T1C (T1CH, T1CL) overflow

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

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

```
                ORG         8               ; Interrupt vector
                JMP         INT_SERVICE
INT_SERVICE:
                …                           ; Push routine to save ACC and PFLAG to buffers.
INTP00CHK:                                  ; Check INT0 interrupt request
                B0BTS1      FP00IEN         ; Check P00IEN
                JMP         INTT0CHK        ; Jump check to next interrupt
                B0BTS0      FP00IRQ         ; Check P00IRQ
                JMP         INTP00          ; Jump to INT0 interrupt service routine
INTP01CHK:                                  ; Check INT1 interrupt request
                B0BTS1      FP01IEN         ; Check P01IEN
                JMP         INTT0CHK        ; Jump check to next interrupt
                B0BTS0      FP01IRQ         ; Check P01IRQ
                JMP         INTP01          ; Jump to INT1 interrupt service routine
INTT0CHK:                                   ; Check T0 interrupt request
                B0BTS1      FT0IEN          ; Check T0IEN
                JMP         INTTC0CHK       ; Jump check to next interrupt
                B0BTS0      FT0IRQ          ; Check T0IRQ
                JMP         INTT0           ; Jump to T0 interrupt service routine
INTTC0CHK:                                  ; Check TC0 interrupt request
                B0BTS1      FTC0IEN         ; Check TC0IEN
                JMP         INTT1CHK        ; Jump check to next interrupt
                B0BTS0      FTC0IRQ         ; Check TC0IRQ
                JMP         INTTC0          ; Jump to TC0 interrupt service routine
INTT1CHK:                                   ; Check T1 interrupt request
                B0BTS1      FT1IEN          ; Check T1IEN
                JMP         …               ; Jump check to next interrupt
                B0BTS0      FT1IRQ          ; Check T1IRQ
                JMP         INTT1           ; Jump to T1 interrupt service routine
                …
                …
INT_EXIT:
                …                           ; Pop routine to load ACC and PFLAG from buffers.
                RETI                        ; Exit interrupt vector
```

7 I/O PORT

7.1 OVERVIEW

The micro-controller builds in 29 pin I/O. Most of the I/O pins are mixed with analog pins and special function pins. The I/O shared pin list is as following.

I/O Pin		Shared Pin		Shared Pin Control Condition
Name	Type	Name	Type	
P0.0	I/O	INT0	DC	P00IEN=1
P0.1	I/O	INT1	DC	P01IEN=1
P0.2	I/O	T1IN	DC	T1CKS=1
P0.3	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P0.4	I/O	XOUT	AC	High_Clk code option = 4M, 12M
		FCPUO	DC	High_Clk code option = RC
P1.0	I/O	RFC0	AC	RFCENB=1, RFCH[2:0]=000b
P1.1	I/O	RFC1	AC	RFCENB=1, RFCH[2:0]=001b
P1.2	I/O	RFC2	AC	RFCENB=1, RFCH[2:0]=010b
P1.3	I/O	RFC3	AC	RFCENB=1, RFCH[2:0]=011b
P1.4	I/O	RFC4	AC	RFCENB=1, RFCH[2:0]=100b
P1.6	I/O	RFCOUT	DC	RFCENB=1, RFCOUT=1
P5.4	I/O	PWM0	DC	TC0ENB=1, PWMOUT=1
P2[3:0]	I/O	SEG[28:31]	DC	PSEG[2:0]=000b
P2[7:4]	I/O	SEG[24:27]	DC	PSEG[2:0]=000b/001b
P3[3:0]	I/O	SEG[20:23]	DC	PSEG[2:0]=000b/001b/010b
P3[7:4]	I/O	SEG[16:19]	DC	PSEG[2:0]=000b/001b/010b/011b

*** DC: Digital Characteristic. AC: Analog Characteristic. HV: High Voltage Characteristic.**

7.2 I/O PORT MODE

The port direction is programmed by PnM register. When the bit of PnM register is “0”, the pin is input mode. When the bit of PnM register is “1”, the pin is output mode.

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	-	-	-	P04M	-	P02M	P01M	P00M
Read/Write	-	-	-	R/W	-	R/W	R/W	R/W
After reset	-	-	-	0	-	0	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	-	P16M	P15M	P14M	P13M	P12M	P11M	P10M
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

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

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

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	-	P54M	-	-	-	-
Read/Write	-	-	-	R/W	-	-	-	-
After reset	-	-	-	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. ***P0.3 input only pin, and the P0M.3 is undefined.***

➢ **Example: I/O mode selection.**

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

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

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

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

➢ **Example: LCD shared pin control methods.**

```
;Enable P2.0~P2.3 GPIO function.
        B0BCLR      FPSEG2      ; Set PSEG[2:0]=001b
        B0BCLR      FPSEG1
        B0BSET      FPSEG0

        MOV         A, #0X0F    ; Set P2.0~P2.3 to output mode.
        B0MOV       P2M, A
        …
        B0BCLR      P2M.0       ; Set P2.0 to input mode.

;Enable P2.0~P2.7 GPIO function.
        B0BCLR      FPSEG2      ; Set PSEG[2:0]=010b
        B0BSET      FPSEG1
        B0BCLR      FPSEG0

        MOV         A, #0XFF    ; Set P2.0~P2.7 to output mode.
        B0MOV       P2M, A
        …
        B0BCLR      P2M.0       ; Set P2.0 to input mode.

;Enable P2.0~P2.7 and P3.0~P3.3 GPIO function.
        B0BCLR      FPSEG2      ; Set PSEG[2:0]=011b
        B0BSET      FPSEG1
        B0BSET      FPSEG0

        MOV         A, #0XFF    ; Set P2.0~P2.7 to output mode.
        B0MOV       P2M, A
        MOV         A, #0X0F    ; Set P3.0~P3.3 to output mode.
        B0MOV       P3M, A
        …
        B0BCLR      P2M.0       ; Set P2.0 to input mode.
        B0BCLR      P3M.0       ; Set P3.0 to input mode.

;Enable P2.0~P2.7 and P3.0~P3.7 GPIO function.
        B0BSET      FPSEG2      ; Set PSEG[2:0]=100b
        B0BCLR      FPSEG1
        B0BCLR      FPSEG0

        MOV         A, #0XFF    ; Set P2.0~P2.7 and P3.0~P3.7 to output mode.
        B0MOV       P2M, A
        B0MOV       P3M, A
        …
        B0BCLR      P2M.0       ; Set P2.0 to input mode.
        B0BCLR      P3M.0       ; Set P3.0 to input mode.

;Disable P2.0~P2.7 and P3.0~P3.7 GPIO function.
        B0BCLR      FPSEG2      ; Set PSEG[2:0]=000b
        B0BCLR      FPSEG1
        B0BCLR      FPSEG0
```

7.3 I/O PULL UP REGISTER

The I/O pins build in internal pull-up resistors and only support I/O input mode. The port internal pull-up resistor is programmed by PnUR register. When the bit of PnUR register is "0", the I/O pin's pull-up is disabled. When the bit of PnUR register is "1", the I/O pin's pull-up is enabled.

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	-	-	P04R	-	P02R	P01R	P00R
Read/Write	-	-	-	W	-	W	W	W
After reset	-	-	-	0	-	0	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	-	P16R	P15R	P14R	P14R	P12R	P11R	P10R
Read/Write	-	W	W	W	W	W	W	W
After reset	-	0	0	0	0	0	0	0

0E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2UR	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0E3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3UR	P37R	P36R	P35R	P34R	P33R	P32R	P31R	P30R
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	-	-	-	P54R	-	-	-	-
Read/Write	-	-	-	W	-	-	-	-
After reset	-	-	-	0	-	-	-	-

✸ ***Note: P0.3 is input only pin and without pull-up resister. The P0UR.3 is undefined.***

➢ **Example: I/O Pull up Register**

```
MOV     A, #0FFH        ; Enable Port0, 1 Pull-up register,
B0MOV   P0UR, A         ;
B0MOV   P1UR,A
```

7.4 I/O PORT DATA REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	-	-	-	P04	P03	P02	P01	P00
Read/Write	-	-	-	R/W	R	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	-	P16	P15	P14	P13	P12	P11	P10
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

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

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

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	-	P54	-	-	-	-
Read/Write	-	-	-	R/W	-	-	-	-
After reset	-	-	-	0	-	-	-	-

✸ ***Note: The P03 keeps "1" when external reset enable by code option.***

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

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

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

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

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

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

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

8 TIMERS

8.1 WATCHDOG TIMER

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

Watchdog overflow time = 8192 / Internal Low-Speed oscillator (sec).

VDD	*Internal Low RC Freq.*	*Watchdog Overflow Time*
3V	16KHz	512ms
5V	32KHz	256ms

The watchdog timer has three operating options controlled “WatchDog” code option.

- **Disable:** Disable watchdog timer function.
- **Enable:** Enable watchdog timer function. Watchdog timer actives in normal mode and slow mode. In power down mode and green mode, the watchdog timer stops.
- **Always_On:** Enable watchdog timer function. The watchdog timer actives and not stop in power down mode and green mode.

In high noisy environment, the “Always_On” option of watchdog operations is the strongly recommendation to make the system reset under error situations and re-start again.

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

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

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

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

➢ **Example: Clear watchdog timer by “@RST_WDT” macro of Sonix IDE.**

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

Watchdog timer application note is as following.

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

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

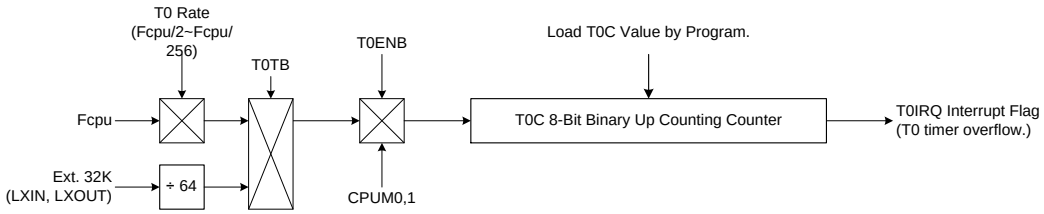
```
Main:
                …                               ; Check I/O.
                …                               ; Check RAM
Err:            JMP $                           ; I/O or RAM error. Program jump here and don't
                                                ; clear watchdog. Wait watchdog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

Correct:                                        ; I/O and RAM are correct. Clear watchdog timer and
                                                ; execute program.
                MOV         A, #5AH             ; Clear the watchdog timer.
                B0MOV       WDTR, A
                …
                CALL        SUB1
                CALL        SUB2
                …
                …
                …
                JMP         MAIN
```

8.2 T0 8-BIT BASIC TIMER

8.2.1 OVERVIEW

The T0 timer is an 8-bit binary up timer with basic timer function. The basic timer function supports flag indicator (T0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T0M, T0C registers and supports RTC function. The T0 builds in green mode wake-up function. When T0 timer overflow occurs under green mode, the system will be waked-up to last operating mode.

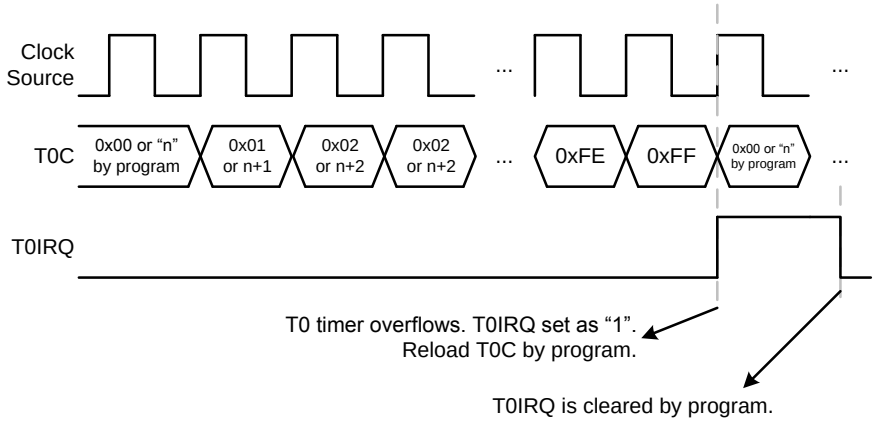
- **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** T0 timer function supports interrupt function. When T0 timer occurs overflow, the T0IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **RTC function:** T0 supports RTC function. The RTC clock source is from external low speed 32K oscillator (LXIN/LXOUT) when T0TB=1.
- **Green mode function:** T0 timer keeps running in green mode and wakes up system when T0 timer overflows.



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

8.2.2 T0 Timer Operation

T0 timer is controlled by T0ENB bit. When T0ENB=0, T0 timer stops. When T0ENB=1, T0 timer starts to count. T0C increases “1” by timer clock source. When T0 overflow event occurs, T0IRQ flag is set as ”1” to indicate overflow and cleared by program. The overflow condition is T0C count from full scale (0xFF) to zero scale (0x00). T0 doesn’t build in double buffer, so load T0C by program when T0 timer overflows to fix the correct interval time. If T0 timer interrupt function is enabled (T0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 8) and executes interrupt service routine after T0 overflow occurrence. Clear T0IRQ by program is necessary in interrupt procedure. T0 timer can works in normal mode, slow mode and green mode. In green mode, T0 keeps counting, set T0IRQ and wakes up system when T0 timer overflows.



T0 clock source is Fcpu (instruction cycle) through T0rate[2:0] pre-scaler to decide Fcpu/2~Fcpu/256. T0 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

T0rate[2:0]	T0 Clock	T0 Interval Time					
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4		IHRC_RTC mode	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)	max. (sec)	Unit (ms)
000b	Fcpu/256	16.384	64	65.536	256	-	-
001b	Fcpu/128	8.192	32	32.768	128	-	-
010b	Fcpu/64	4.096	16	16.384	64	-	-
011b	Fcpu/32	2.048	8	8.192	32	-	-
100b	Fcpu/16	1.024	4	4.096	16	-	-
101b	Fcpu/8	0.512	2	2.048	8	-	-
110b	Fcpu/4	0.256	1	1.024	4	-	-
111b	Fcpu/2	0.128	0.5	0.512	2	-	-
-	32768Hz/64	-	-	-	-	0.5	1.953

8.2.3 T0M MODE REGISTER

T0M is T0 timer mode control register to configure T0 operating mode including T0 pre-scaler, clock source…These configurations must be setup completely before enabling T0 timer.

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB
Read/Write	R/W	R/W	R/W	R/W	-	-	-	R/W
After reset	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 [6:4] **T0RATE[2:0]:** T0 timer clock source select bits.
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 **T0ENB:** T0 counter control bit.
0 = Disable T0 timer.
1 = Enable T0 timer.

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

8.2.4 T0C COUNTING REGISTER

T0C is T0 8-bit counter. When T0C overflow occurs, the T0IRQ flag is set as "1" and cleared by program. The T0C decides T0 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T0C register, and then enable T0 timer to make sure the first cycle correct. After one T0 overflow occurs, the T0C register is loaded a correct value by program.

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

The equation of T0C initial value is as following.

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

✸ **Example: To calculation T0C to obtain 10ms T0 interval time. T0 clock source is Fcpu = 4MHz/4 = 1MHz. Select T0RATE=001 (Fcpu/128).**
T0 interval time = 10ms. T0 clock rate = 4MHz/4/128

*T0C initial value = 256 - (T0 interval time * input clock)*
*= 256 - (10ms * 4MHz / 4 / 128)*
*= 256 - (10^{-2} * 4 * 10^{6} / 4 / 128)*
= B2H

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

8.2.5 T0 TIMER OPERATION EXPLAME

- **T0 TIMER CONFIGURATION:**

```
; Reset T0 timer.
        MOV     A, #0x00        ; Clear T0M register.
        B0MOV   T0M, A

; Set T0 clock source and T0 rate.
        MOV     A, #0nnn00000b
        B0MOV   T0M, A

; Set T0C register for T0 Interval time.
        MOV     A, #value
        B0MOV   T0C, A

; Clear T0IRQ
        B0BCLR  FT0IRQ

; Enable T0 timer and interrupt function.
        B0BSET  FT0IEN          ; Enable T0 interrupt function.
        B0BSET  FT0ENB          ; Enable T0 timer.
```

- **T0 works in RTC mode:**

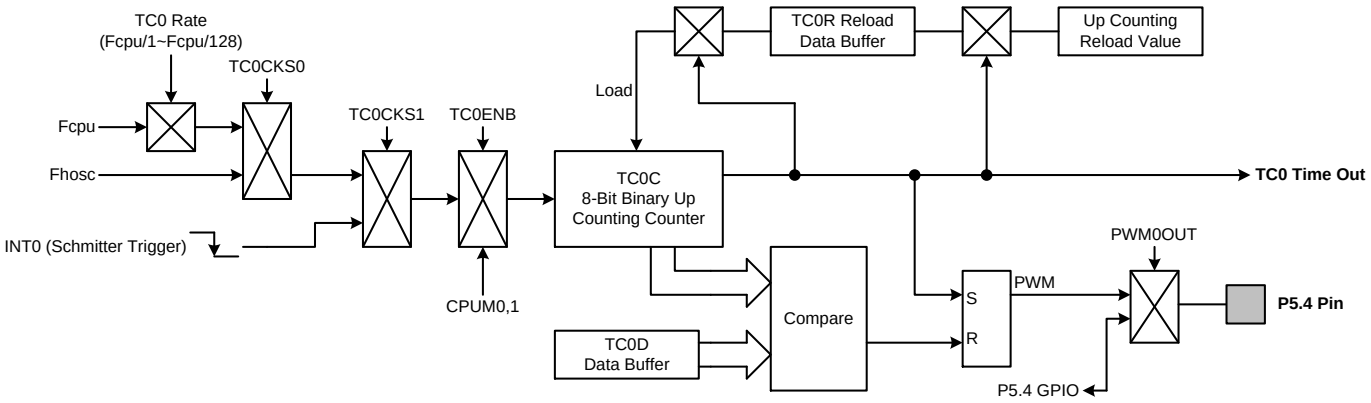
```
; Reset T0 timer.
        MOV     A, #0x00        ; Clear T0M register.
        B0MOV   T0M, A

; Set T0 RTC function.
        B0BSET  FT0TB

; Clear T0C.
        CLR     T0C

; Clear T0IRQ
        B0BCLR  FT0IRQ

; Enable T0 timer and interrupt function.
        B0BSET  FT0IEN          ; Enable T0 interrupt function.
        B0BSET  FT0ENB          ; Enable T0 timer.
```

8.3 TC0 8-BIT TIMER/COUNTER

8.3.1 OVERVIEW

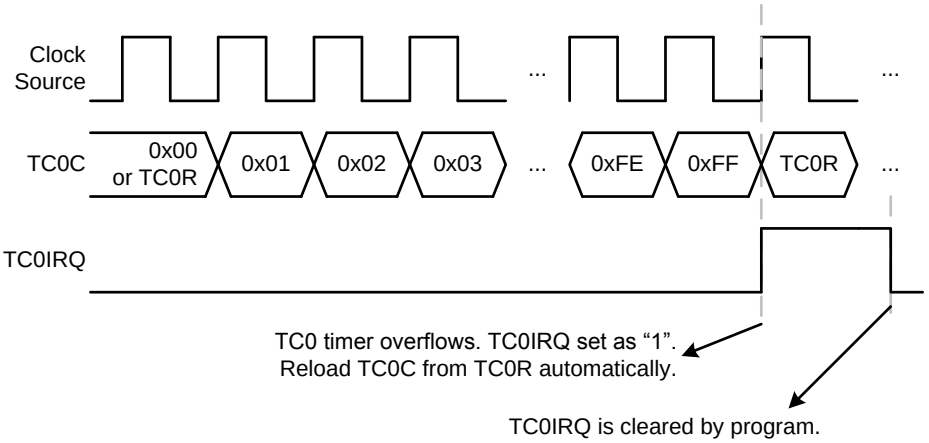
The TC0 timer is an 8-bit binary up timer with basic timer, event counter and PWM functions. The basic timer function supports flag indicator (TC0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TC0M, TC0C, TC0R registers. The event counter is changing TC0 clock source from system clock (Fcpu/Fhosc) to external clock like signal (e.g. continuous pulse, R/C type oscillating signal…). TC0 becomes a counter to count external clock number to implement measure application. TC0 also builds in duty/cycle programmable PWM. The PWM cycle and resolution are controlled by TC0 timer clock rate, TC0R and TC0D registers, so the PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster…The main purposes of the TC0 timer are as following.

- **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **Interrupt function:** TC0 timer function supports interrupt function. When TC0 timer occurs overflow, the TC0IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Event Counter:** The event counter function counts the external clock counts.
- **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by TC0R and TC0D registers.
- **Green mode function:** All TC0 functions (timer, PWM, event counter, auto-reload) keep running in green mode and no wake-up function.



8.3.2 TC0 TIMER OPERATION

TC0 timer is controlled by TC0ENB bit. When TC0ENB=0, TC0 timer stops. When TC0ENB=1, TC0 timer starts to count. Before enabling TC0 timer, setup TC0 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…TC0C increases "1" by timer clock source. When TC0 overflow event occurs, TC0IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is TC0C count from full scale (0xFF) to zero scale (0x00). In difference function modes, TC0C value relates to operation. If TC0C value changing effects operation, the transition of operations would make timer function error. So TC0 builds in double buffer to avoid these situations happen. The double buffer concept is to flash TC0C during TC0 counting, to set the new value to TC0R (reload buffer), and the new value will be loaded from TC0R to TC0C after TC0 overflow occurrence automatically. In the next cycle, the TC0 timer runs under new conditions, and no any transitions occur. The auto-reload function is no any control interface and always actives as TC0 enables. If TC0 timer interrupt function is enabled (TC0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 8) and executes interrupt service routine after TC0 overflow occurrence. Clear TC0IRQ by program is necessary in interrupt procedure. TC0 timer can works in normal mode, slow mode and green mode. But in green mode, TC0 keep counting, set TC0IRQ and outputs PWM, but can't wake-up system.



TC0 provides different clock sources to implement different applications and configurations. TC0 clock source includes Fcpu (instruction cycle), Fhosc (high speed oscillator) and external input pin (P0.0) controlled by TC0CKS[1:0] bits. TC0CKS0 bit selects the clock source is from Fcpu or Fhosc. If TC0CKS0=0, TC0 clock source is Fcpu through TC0rate[2:0] pre-scaler to decide Fcpu/1~Fcpu/128. If TC0CKS0=1, TC0 clock source is Fhosc without any divider. TC0CKS1 bit controls the clock source is external input pin or controlled by TC0CKS0 bit. If TC0CKS1=0, TC0 clock source is selected by TC0CKS0 bit. If TC0CKS1=1, TC0 clock source is external input pin that means to enable event counter function. TC0rate[2:0] pre-scaler is unless when TC0CKS0=1 or TC0CKS1=1 conditions. TC0 length is 8-bit (256 steps), and the one count period is each cycle of input clock.

TC0CKS0	TC0rate[2:0]	TC0 Clock	TC0 Interval Time	
			Fhosc=16MHz, Fcpu=Fhosc/2	
			max. (ms)	Unit (us)
0	000b	Fcpu/128	4.096	16
0	001b	Fcpu/64	2.048	8
0	010b	Fcpu/32	1.024	4
0	011b	Fcpu/16	0.512	2
0	100b	Fcpu/8	0.256	1
0	101b	Fcpu/4	0.128	0.5
0	110b	Fcpu/2	0.064	0.25
0	111b	Fcpu/1	0.032	0.125
1	useless	Fhosc	0.016	0.0625

8.3.3 TC0M MODE REGISTER

TC0M is TC0 timer mode control register to configure TC0 operating mode including TC0 pre-scaler, clock source, PWM function…These configurations must be setup completely before enabling TC0 timer.

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS1	TC0CKS0	-	PWM0OUT
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	-	R/W
After reset	0	0	0	0	0	0	-	0

Bit 0 **PWM0OUT:** PWM output control bit.
0 = Disable PWM output function, and P5.4 is GPIO mode.
1 = Enable PWM output function, and P5.4 outputs PWM signal.

Bit 2 **TC0CKS0:** TC0 clock source select bit.
0 = Fcpu.
1 = Fhosc. **TC0rate[2:0] bits are useless.**

Bit 3 **TC0CKS1:** TC0 clock source select bit.
0 = Internal clock (Fcpu and Fhosc controlled by TC0CKS0 bit).
1 = External input pin (P0.0/INT0) and enable event counter function. **TC0rate[2:0] bits are useless.**

Bit [6:4] **TC0RATE[2:0]:** TC0 timer clock source select bits.
000 = Fcpu/128, 001 = Fcpu/64, 010 = Fcpu/32, 011 = Fcpu/16, 100 = Fcpu/8, 101 = Fcpu/4, 110 = Fcpu/2, 111 = Fcpu/1.

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

8.3.4 TC0C COUNTING REGISTER

TC0C is TC0 8-bit counter. When TC0C overflow occurs, the TC0IRQ flag is set as “1” and cleared by program. The TC0C decides TC0 interval time through below equation to calculate a correct value. It is necessary to write the correct value to TC0C register and TC0R register first time, and then enable TC0 timer to make sure the fist cycle correct. After one TC0 overflow occurs, the TC0C register is loaded a correct value from TC0R register automatically, not program.

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

The equation of TC0C initial value is as following.

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

8.3.5 TC0R AUTO-RELOAD REGISTER

TC0 timer builds in auto-reload function, and TC0R register stores reload data. When TC0C overflow occurs, TC0C register is loaded data from TC0R register automatically. Under TC0 timer counting status, to modify TC0 interval time is to modify TC0R register, not TC0C register. New TC0C data of TC0 interval time will be updated after TC0 timer overflow occurrence, TC0R loads new value to TC0C register. But at the first time to setup TC0M, TC0C and TC0R must be set the same value before enabling TC0 timer. TC0 is double buffer design. If new TC0R value is set by program, the new value is stored in 1st buffer. Until TC0 overflow occurs, the new value moves to real TC0R buffer. This way can avoid any transitional condition to effect the correctness of TC0 interval time and PWM output signal.

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

The equation of TC0R initial value is as following.

TC0R initial value = 256 - (TC0 interrupt interval time * TC0 clock rate)

☞ **Example: To calculation TC0C and TC0R value to obtain 10ms TC0 interval time. TC0 clock source is Fcpu = 16MHz/16 = 1MHz. Select TC0RATE=000 (Fcpu/128).**
TC0 interval time = 10ms. TC0 clock rate = 16MHz/16/128

*TC0C/TC0R initial value = 256 - (TC0 interval time * input clock)*
*= 256 - (10ms * 16MHz / 16 / 128)*
*= 256 - (10^{-2} * 16 * 10^{6} / 16 / 128)*
= B2H

8.3.6 TC0D PWM DUTY REGISTER

TC0D register's purpose is to decide PWM duty. In PWM mode, TC0R controls PWM's cycle, and TC0D controls the duty of PWM. The operation is base on timer counter value. When TC0C = TC0D, the PWM high duty finished and exchange to low level. It is easy to configure TC0D to choose the right PWM's duty for application.

0E8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0D	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0
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 TC0D initial value is as following.

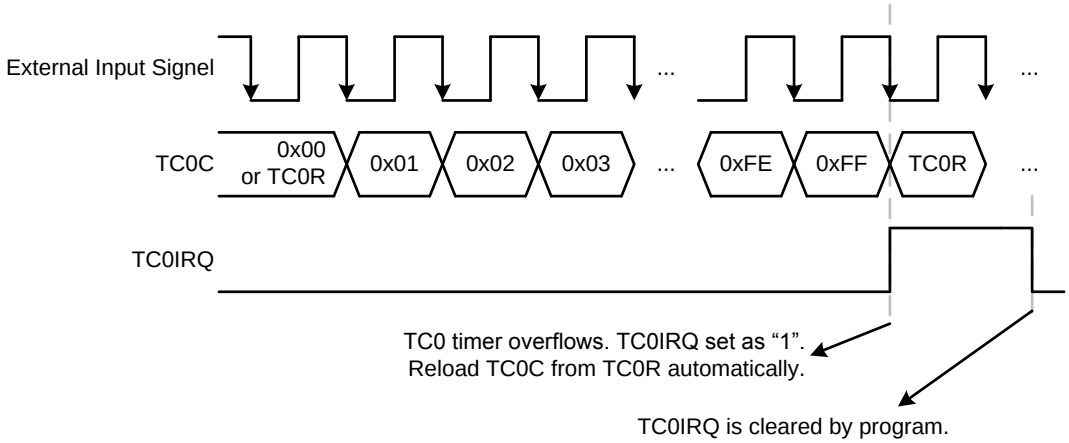
TC0D initial value = TC0R + (PWM high pulse width period / TC0 clock rate)

☞ **Example: To calculate TC0D value to obtain 1/3 duty PWM signal. The TC0 clock source is Fcpu = 16MHz/16= 1MHz. Select TC0RATE=000 (Fcpu/128).**
TC0R = B2H. TC0 interval time = 10ms. So the PWM cycle is 100Hz. In 1/3 duty condition, the high pulse width is about 3.33ms.

TC0D initial value = B2H + (PWM high pulse width period / TC0 clock rate)
*= B2H + (3.33ms * 16MHz / 16 / 128)*
= B2H + 1AH
= CCH

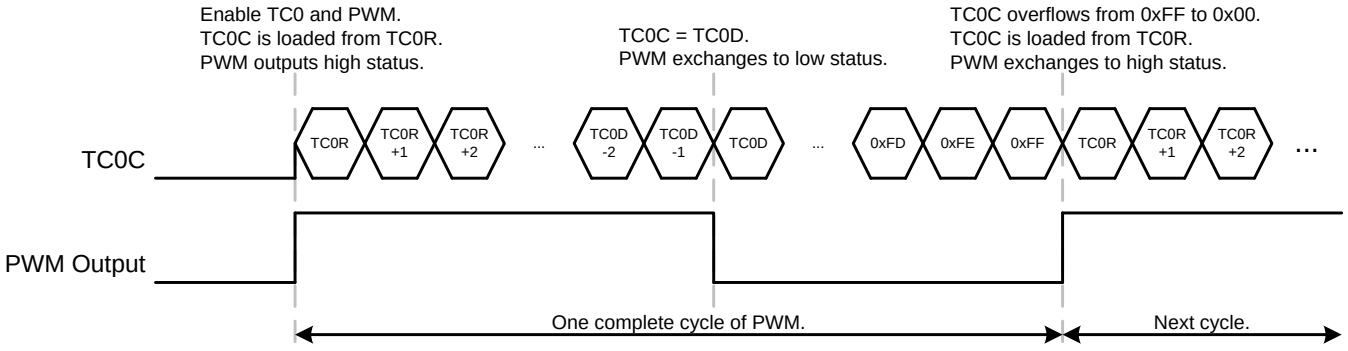
8.3.7 TC0 EVENT COUNTER

TC0 event counter is set the TC0 clock source from external input pin (P0.0). When TC0CKS1=1, TC0 clock source is switch to external input pin (P0.0). TC0 event counter trigger direction is falling edge. When one falling edge occurs, TC0C will up one count. When TC0C counts from 0xFF to 0x00, TC0 triggers overflow event. The external event counter input pin's wake-up function of GPIO mode is disabled when TC0 event counter function enabled to avoid event counter signal trigger system wake-up and not keep in power saving mode. The external event counter input pin's external interrupt function is also disabled when TC0 event counter function enabled, and the P00IRQ bit keeps "0" status. The event counter usually is used to measure external continuous signal rate, e.g. continuous pulse, R/C type oscillating signal…These signal phase don't synchronize with MCU's main clock. Use TC0 event to measure it and calculate the signal rate in program for different applications.

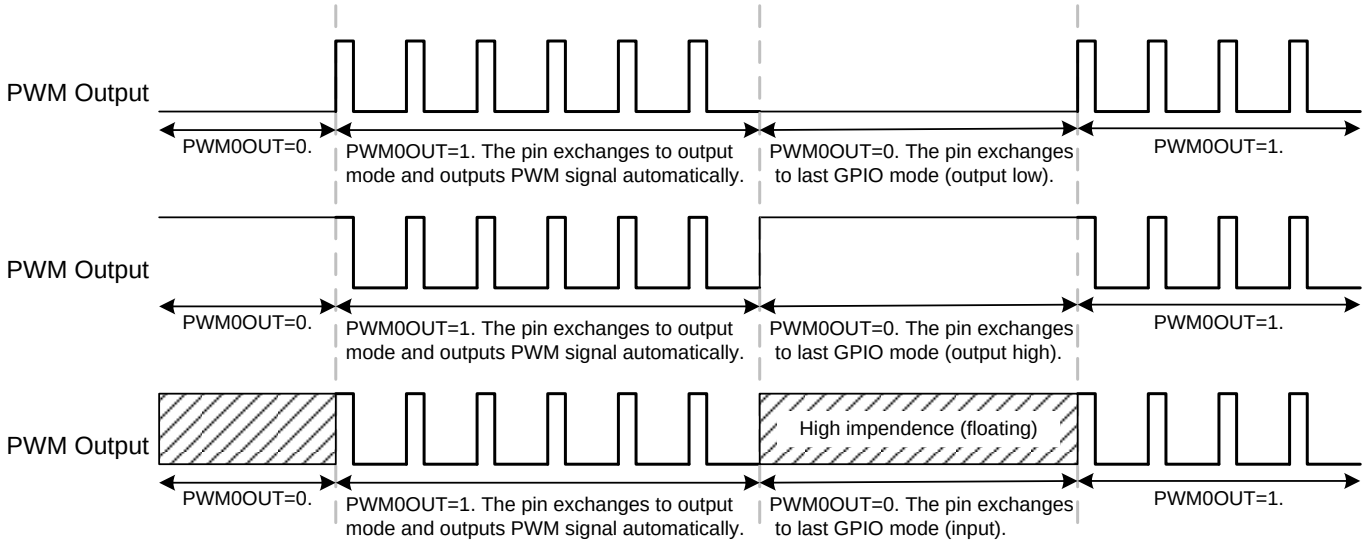


8.3.8 PULSE WIDTH MODULATION (PWM)

The PWM is duty/cycle programmable design to offer various PWM signals. When TC0 timer enables and PWM0OUT bit sets as "1" (enable PWM output), the PWM output pin (P5.4) outputs PWM signal. One cycle of PWM signal is high pulse first, and then low pulse outputs. TC0R register controls the cycle of PWM, and TC0D decides the duty (high pulse width length) of PWM. TC0C initial value is TC0R reloaded when TC0 timer enables and TC0 timer overflows. When TC0C count is equal to TC0D, the PWM high pulse finishes and exchanges to low level. When TC0 overflows (TC0C counts from 0xFF to 0x00), one complete PWM cycle finishes. The PWM exchanges to high level for next cycle. The PWM is auto-reload design to load TC0C from TC0R automatically when TC0 overflows and the end of PWM's cycle, to keeps PWM continuity. If modify the PWM cycle by program as PWM outputting, the new cycle occurs at next cycle when TC0C loaded from TC0R.



The resolution of PWM is decided by TC0R. TC0R range is from 0x00~0xFF. If TC0R = 0x00, PWM's resolution is 1/256. If TC0R = 0x80, PWM's resolution is 1/128. TC0D controls the high pulse width of PWM for PWM's duty. When TC0C = TC0D, PWM output exchanges to low status. TC0D must be greater than TC0R, or the PWM signal keeps low status. When PWM outputs, TC0IRQ still actives as TC0 overflows, and TC0 interrupt function actives as TC0IEN = 1. But strongly recommend be careful to use PWM and TC0 timer together, and make sure both functions work well.
The PWM output pin is shared with GPIO and switch to output PWM signal as PWM0OUT=1 automatically. If PWM0OUT bit is cleared to disable PWM, the output pin exchanges to last GPIO mode automatically. It easily to implement carry signal on/off operation, not to control TC0ENB bit.



8.3.9 TC0 TIMER OPERATION EXPLAME

- **TC0 TIMER CONFIGURATION:**

```
; Reset TC0 timer.
        CLR     TC0M            ; Clear TC0M register.

; Set TC0 clock source and TC0 rate.
        MOV     A, #0nnn0n00b
        B0MOV   TC0M, A

; Set TC0C and TC0R register for TC0 Interval time.
        MOV     A, #value       ; TC0C must be equal to TC0R.
        B0MOV   TC0C, A
        B0MOV   TC0R, A

; Clear TC0IRQ
        B0BCLR  FTC0IRQ

; Enable TC0 timer and interrupt function.
        B0BSET  FTC0IEN         ; Enable TC0 interrupt function.
        B0BSET  FTC0ENB         ; Enable TC0 timer.
```

- **TC0 EVENT COUNTER CONFIGURATION:**

```
; Reset TC0 timer.
        CLR         TC0M            ; Clear TC0M register.

; Enable TC0 event counter.
        B0BSET      FTC0CKS1        ; Set TC0 clock source from external input pin (P0.0).

; Set TC0C and TC0R register for TC0 Interval time.
        MOV         A, #value       ; TC0C must be equal to TC0R.
        B0MOV       TC0C, A
        B0MOV       TC0R, A

; Clear TC0IRQ
        B0BCLR      FTC0IRQ

; Enable TC0 timer and interrupt function.
        B0BSET      FTC0IEN         ; Enable TC0 interrupt function.
        B0BSET      FTC0ENB         ; Enable TC0 timer.
```

- **TC0 PWM CONFIGURATION:**

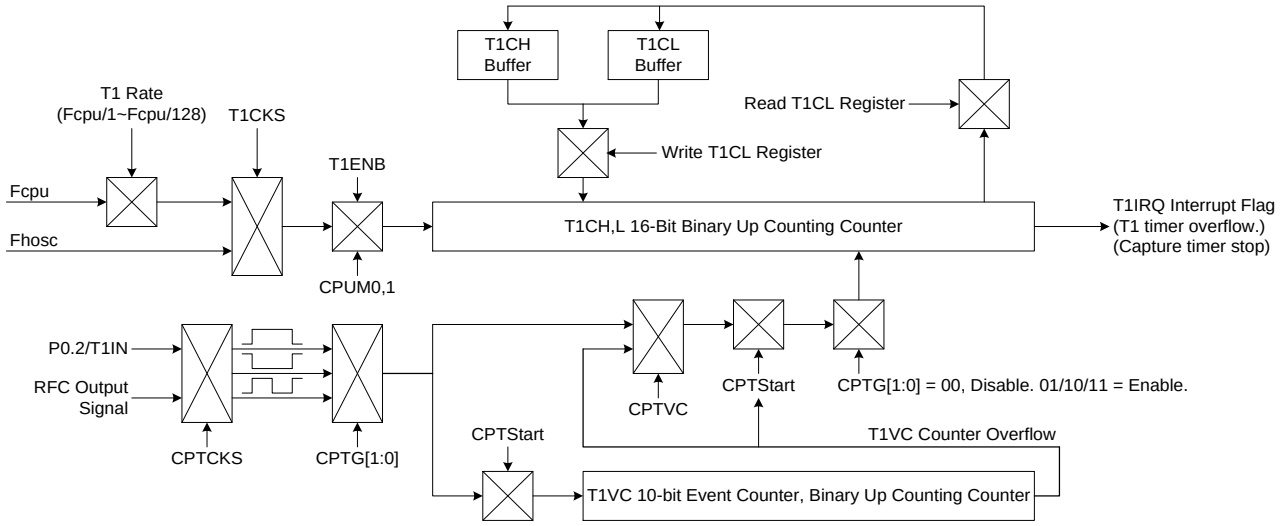
```
; Reset TC0 timer.
        CLR         TC0M            ; Clear TC0M register.

; Set TC0 clock source and TC0 rate.
        MOV         A, #0nnn0n00b
        B0MOV       TC0M, A

; Set TC0C and TC0R register for PWM cycle.
        MOV         A, #value1      ; TC0C must be equal to TC0R.
        B0MOV       TC0C, A
        B0MOV       TC0R, A

; Set TC0D register for PWM duty.
        MOV         A, #value2      ; TC0D must be greater than TC0R.
        B0MOV       TC0D, A

; Enable PWM and TC0 timer.
        B0BSET      FTC0ENB         ; Enable TC0 timer.
        B0BSET      FPWM0OUT        ; Enable PWM.
```

8.4 T1 16-BIT TIMER WITH CAPTURE TIMER FUNCTION

8.4.1 OVERVIEW

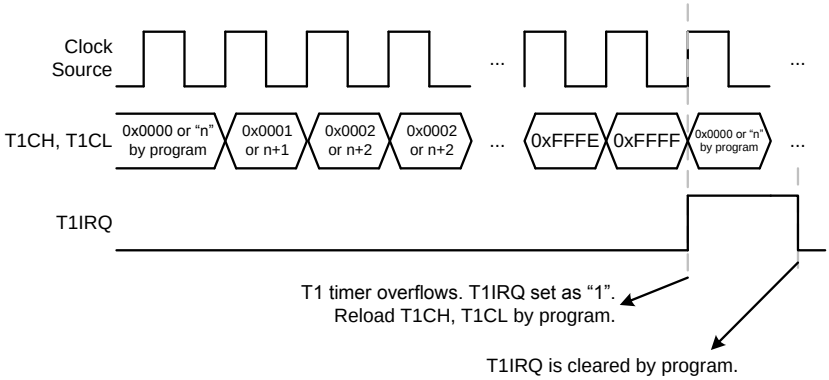
The T1 timer is a 16-bit binary up timer with basic timer and capture timer functions. The basic timer function supports flag indicator (T1IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T1M, T1CH/T1CL 16-bit counter registers. The capture timer supports high pulse width measurement, low pulse width measurement, cycle measurement and continuous duration from P0.2/T1IN pin. T1 becomes a timer meter to count external signal time parameters to implement measure application. The main purposes of the T1 timer are as following.

- **16-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **16-bit capture timer:** Measure the input signal pulse width and cycle depend on the T1 clock time base to decide the capture timer's resolution. The capture timer builds in programmable trigger edge selection to decide the start-stop trigger event.
- **10-bit event counter:** The 10-bit event counter to detect event source for accumulative capture timer function. The event counter is up counting design. When the counter is overflow, the T1 stops counting and the T1 counter buffers records the period of event counter duration.
- **Interrupt function:** T1 timer function and capture timer function support interrupt function. When T1 timer occurs overflow or capture timer stops counting, the T1IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Green mode function:** All T1 functions (timer, event counter, capture timer, auto-reload) keeps running in green mode, but no wake-up function.



8.4.2 T1 TIMER OPERATION

T1 timer is controlled by T1ENB bit. When T1ENB=0, T1 timer stops. When T1ENB=1, T1 timer starts to count. Before enabling T1 timer, setup T1 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…T1 16-bit counter (T1CH, T1CL) increases "1" by timer clock source. When T1 overflow event occurs, T1IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is T1CH, T1CL count from full scale (0xFFFF) to zero scale (0x0000). T1 doesn't build in double buffer, so load T1CH, T1CL by program when T1 timer overflows to fix the correct interval time. If T1 timer interrupt function is enabled (T1IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 000FH) and executes interrupt service routine after T1 overflow occurrence. Clear T1IRQ by program is necessary in interrupt procedure. T1 timer can works in normal mode, slow mode and green mode.



T1 provides different clock sources to implement different applications and configurations. T1 clock source includes Fcpu (instruction cycle) and Fhosc (high speed oscillator) controlled by T1CKS bit. T1CKS bit selects the clock source is from Fcpu or Fhosc. If T1CKS=0, T1 clock source is Fcpu through T1rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. If T1CKS=1, T1 clock source is Fhosc. T1 length is 16-bit (65536 steps), and the one count period is each cycle of input clock.

T1CKS	T1rate[2:0]	T1 Clock	T1 Interval Time			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	2097.152	32	8388.608	128
0	001b	Fcpu/64	1048.576	16	4194.304	64
0	010b	Fcpu/32	524.288	8	2097.152	32
0	011b	Fcpu/16	262.144	4	1048.576	16
0	100b	Fcpu/8	131.072	2	524.288	8
0	101b	Fcpu/4	65.536	1	262.144	4
0	110b	Fcpu/2	32.768	0.5	131.072	2
0	111b	Fcpu/1	16.384	0.25	65.536	1
1	-	Fhosc/1	4.096	0.0625	16.384	0.25

8.4.3 T1M MODE REGISTER

T1M is T1 timer mode control register to configure T1 operating mode including T1 pre-scalar, clock source, capture parameters…These configurations must be setup completely before enabling T1 timer.

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1rate2	T1rate1	T1rate0	T1CKS			
Read/Write	R/W	R/W	R/W	R/W	R/W			
After reset	0	0	0	0	0			

Bit 7 **T1ENB:** T1 counter control bit.
0 = Disable T1 timer.
1 = Enable T1 timer.

Bit [6:4] **T1RATE[2:0]:** T1 timer clock source select bits. **If T1CKS0=1, the T1RATE[2:0] control is "Ignored".**
000 = Fcpu/128, 001 = Fcpu/64, 010 = Fcpu/32, 011 = Fcpu/16, 100 = Fcpu/8, 101 = Fcpu/4,
110 = Fcpu/2,111 = Fcpu/1.

Bit 3 **T1CKS:** T1 clock source control bit.
0 = Fcpu divided by T1rate[2:0].
1 = Fhosc.

8.4.4 T1CH, T1CL 16-bit COUNTING REGISTERS

T1 counter is 16-bit counter combined with T1CH and T1CL registers. When T1 timer overflow occurs, the T1IRQ flag is set as "1" and cleared by program. The T1CH, T1CL decide T1 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T1CH and T1CL registers, and then enable T1 timer to make sure the fist cycle correct. After one T1 overflow occurs, the T1CH and T1CL registers are loaded correct values by program.

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CL	T1CL7	T1CL6	T1CL5	T1CL4	T1CL3	T1CL2	T1CL1	T1CL0
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

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CH	T1CH7	T1CH6	T1CH5	T1CH4	T1CH3	T1CH2	T1CH1	T1CH0
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 T1 timer counter length is 16-bit and points to T1CH and T1CL registers. The timer counter is double buffer design. The core bus is 8-bit, so access 16-bit data needs a latch flag to avoid the transient status affect the 16-bit data mistake occurrence. Under write mode, the write T1CH is the latch control flag. Under read mode, the read T1CL is the latch control flag. So, write T1 16-bit counter is to write T1CH first, and then write T1CL. The 16-bit data is written to 16-bit counter buffer after executing writing T1CL. Read T1 16-bit counter is to read T1CL first, and then read T1CH. The 16-bit data is dumped to T1CH, T1CL registers after executing reading T1CL.

- **Read T1 counter buffer sequence is to read T1CL first, and then read T1CH.**
- **Write T1 counter buffer sequence is to write T1CH first, and then write T1CL.**

The equation of T1 16-bit counter (T1CH, T1CL) initial value is as following.

T1CH, T1CL initial value = 65536 - (T1 interrupt interval time * T1 clock rate)

- **Example: To calculation T1CH and T1CL values to obtain 500ms T1 interval time. T1 clock source is Fcpu = 16MHz/16 = 1MHz. Select T1RATE=000 (Fcpu/128).**
  T1 interval time = 500ms. T1 clock rate = 16MHz/16/128

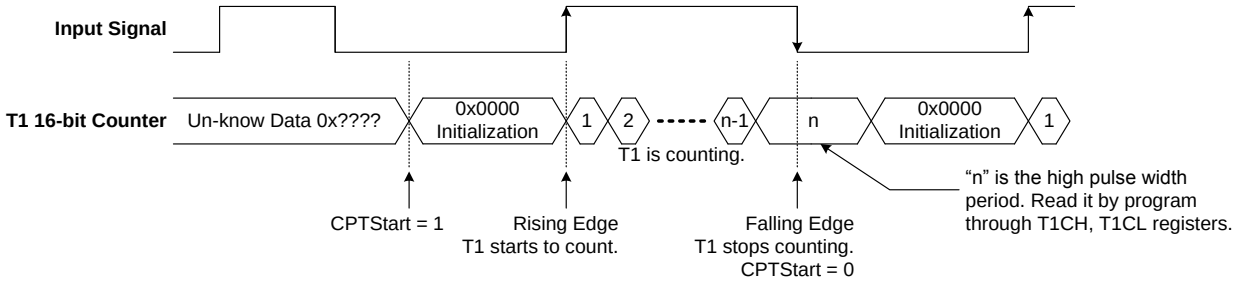
*T1 16-bit counter initial value = 65536 - (T1 interval time * input clock)*
*= 65536 - (500ms * 16MHz / 16 / 128)*
*= 65536 - (500*10^{-3} * 16 * 10^{6} / 16 / 128)*
= F0BDH ***(T1CH = F0H, T1CL = BDH)***

8.4.5 T1 CPATURE TIMER OPERATION

The 16-bit capture timer purpose is to measure input signal pulse width and cycle. The measure is through T1 timer by trigger selection. The capture timer is controlled by CPTG[1:0] bits. When CPTG[1:0] = 00, the capture timer is disabled. When CPTG[1:0] = 01/10/11, the capture timer is enabled, but the T1ENB must be enabled. The capture timer can measure input high pulse width, input low pulse width and the cycle of input signal controlled by CPTG[1:0]. CPTG[1:0] = 01, measure input high pulse width. CPTG[1:0] = 10, measure input low pulse width. CPTG[1:0] = 11, measure the cycle of input signal. The CPTG[1:0] only selects the capture timer function, not to execute the capture timer. CPTStart bit is to execute capture timer. When CPTStart is set as "1", the capture timer waits the right trigger edge to active 16-bit counter. The trigger edge finds, and the 16-bit counter starts to count which clock source is T1. When the second right edge finds, the 16-counter stops, CPTStart is cleared and the T1IRQ actives. Before setting CPTStart bit, the T1 16-bit counter is cleared for capture timer initialization.

- **High Pulse Width Measurement**

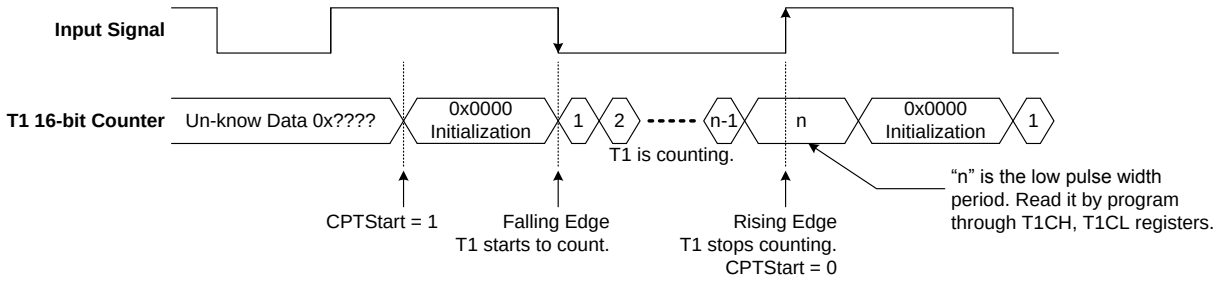
T1ENB = 1. CPTG[1:0] = 01.



The high pulse width measurement is using rising edge to start T1 16-bit counter and falling edge to stop T1 16-bit counter. If set CPTStart bit at high pulse duration, the capture timer will measure next high pulse until the rising edge occurrence.

- **Low Pulse Width Measurement**

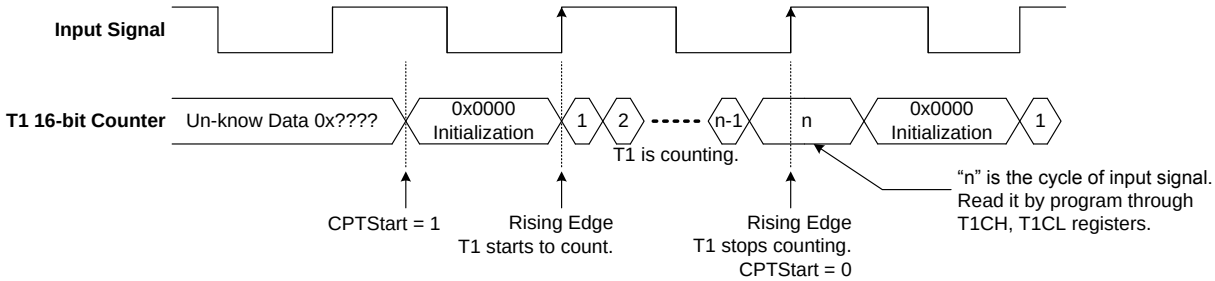
T1ENB = 1. CPTG[1:0] = 10.



The low pulse width measurement is using falling edge to start T1 16-bit counter and rising edge to stop T1 16-bit counter. If set CPTStart bit at low pulse duration, the capture timer will measure next low pulse until the falling edge occurrence.

- **Input Cycle Measurement**

T1ENB = 1. CPTG[1:0] = 11.



The cycle measurement is using rising edge to start and stop T1 16-bit counter. If set CPTStart bit at high or low pulse duration, the capture timer will measure next cycle until the rising edge occurrence.

8.4.6 CAPTURE TIMER CONTROL REGISTERS

A5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CKSM	CPTVC				CPTCKS	CPTStart	CPTG1	CPTG0
Read/Write	R/W				R/W	R/W	R/W	R/W
After Reset	0				0	0	0	0

Bit 7 **CPTVC:** Event counter function control bit.
0 = Disable event counter function.
1 = Enable event counter function.

Bit 3 **CPTCKS:** Capture timer clock source control bit.
0 = External input pin, P0.2/T1IN.
1 = RFC output terminal.

Bit 2 **CPTStart:** Capture timer counter control bit.
0 = Process end.
1 = Start to count and processing.

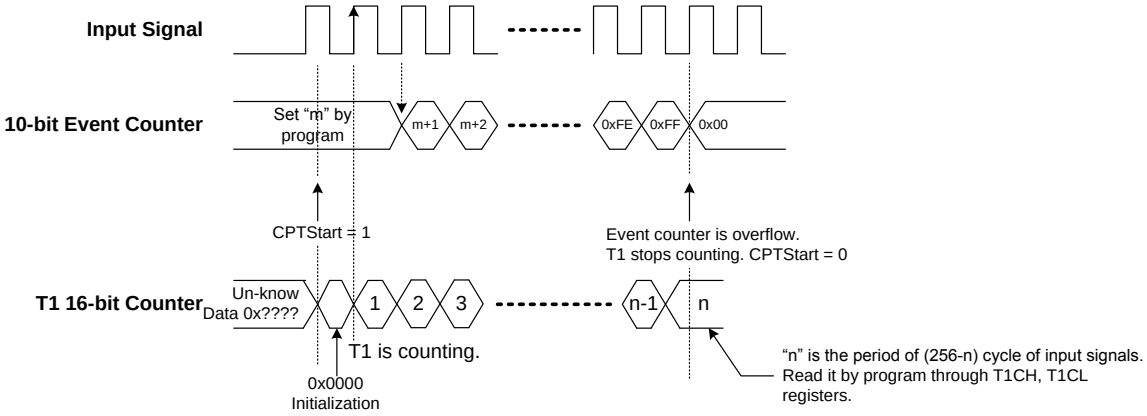
Bit [1:0] **CPTG[1:0]:** Capture timer function control bit.
00 = Disable capture timer function.
01 = High pulse width measurement.
10 = Low pulse width measurement.
11 = Cycle measurement.

8.4.7 10-bit Event Counter Function

The 10-bit event timer purpose is to measure the period of a continuous input signal. The measure is through T1 timer by trigger selection. The event counter is controlled by CPTVC bit. When CPTVC = 0, the event counter is disabled. When CPTVC = 1, the event counter is enabled, but the T1ENB must be enabled. The event counter trigger edge is rising edge and must be set as CPTG[1:0] = 01 by program. The trigger edge finds, and the 10-bit event counter and T1 16-bit timer start to count. When T1 event counter overflows, T1 event counter and 16-bit timer stops counting and the T1IRQ actives. Before execute T1 event counter, the T1 16-bit counter must be cleared for initialization.

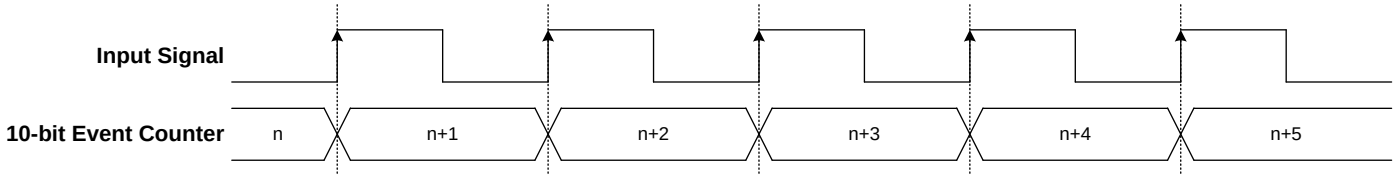
- ***Note: The event counter trigger edge is rising edge and must be set as CPTG[1:0] = 01 by program.***

- **Event Counter Operation**



- **Event Counter Trigger Format**

The capture timer input source has high pulse, low pulse and cycle signal. In event counter function, the trigger format is not like capture timer and fixed rising edge format. Use the start trigger signal of capture timer to be the event counter trigger source:



8.4.8 T1VCH, T1VCL 10-bit EVENT COUNTER REGISTERS

T1 event counter is 10-bit counter combined with T1VCH and T1VCL registers. When T1 event counter overflow occurs, the T1IRQ flag is set as "1" and cleared by program. The T1VCH, T1VCL decide T1 event counter number.

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1VCL	T1VCL7	T1VCL6	T1VCL5	T1VCL4	T1VCL3	T1VCL2	T1VCL1	T1VCL0
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

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1VCH	-	-	-	-	-	-	T1VCH1	T1VCH0
Read/Write	-	-	-	-	-	-	R/W	R/W
After Reset	-	-	-	-	-	-	0	0

The T1 event counter length is 10-bit and points to T1VCH and T1VCL registers. The core bus is 8-bit, so access 10-bit data needs a latch flag to avoid the transient status affect the 10-bit data mistake occurrence. Under write mode, the write T1VCH is the latch control flag. Under read mode, the read T1VCL is the latch control flag. So, write T1 10-bit event counter is to write T1VCH first, and then write T1VCL. The 10-bit data is written to 10-bit counter buffer after executing writing T1VCL. Read T1 10-bit event counter is to read T1VCL first, and then read T1VCH. The 10-bit data is dumped to T1VCH, T1VCL registers after executing reading T1VCL.

- **Read T1 event counter buffer sequence is to read T1VCL first, and then read T1VCH.**
- **Write T1 event counter buffer sequence is to write T1VCH first, and then write T1VCL.**

8.4.9 T1 TIMER OPERATION EXPLAME

- **T1 TIMER CONFIGURATION:**

```
; Reset T1 timer.
        MOV     A, #0x00        ; Clear T1M register.
        B0MOV   T1M, A

; Set T1 clock rate.
        MOV     A, #0nnn0000b   ; T1rate[2:0] bits.
        B0MOV   T1M, A

; Set T1CH, T1CL registers for T1 Interval time.
        MOV     A, #value1      ; Set high byte first.
        B0MOV   T1CH, A
        MOV     A, #value2      ; Set low byte.
        B0MOV   T1CL, A

; Clear T1IRQ
        B0BCLR  FT1IRQ

; Enable T1 timer and interrupt function.
        B0BSET  FT1IEN          ; Enable T1 interrupt function.
        B0BSET  FT1ENB          ; Enable T1 timer.
```

- **T1 CAPTURE TIMER FOR SINGLE CYCLE MEASUREMENT CONFIGURATION:**

```
; Reset T1 timer.
        MOV     A, #0x00        ; Clear T1M register.
        B0MOV   T1M, A

; Set T1 clock rate, select input source, and select/enable T1 capture timer.
        MOV     A, #0nnnm000b   ; “nnn” is T1rate[2:0] for T1 clock rate selection.
        B0MOV   T1M, A          ; “m” is T1 clock source control bit.
        MOV     A, #000000mmb   ; “mm” is CPTG[1:0] for T1 capture timer function selection.
        B0MOV   T1CKSM, A       ; CPTG[1:0] = 01b, high pulse width measurement.
                                ; CPTG[1:0] = 10b, low pulse width measurement.
                                ; CPTG[1:0] = 11b, cycle measurement.


; Clear T1CH, T1CL.
        CLR     T1CH            ; Clear high byte first.
        CLR     T1CL            ; Clear low byte.

; Clear T1IRQ
        B0BCLR  FT1IRQ

; Enable T1 timer, interrupt function and T1 capture timer function.
        B0BSET  FT1IEN          ; Enable T1 interrupt function.
        B0BSET  FT1ENB          ; Enable T1 timer.

; Set capture timer start bit.
        B0BSET  FCPTStart
```

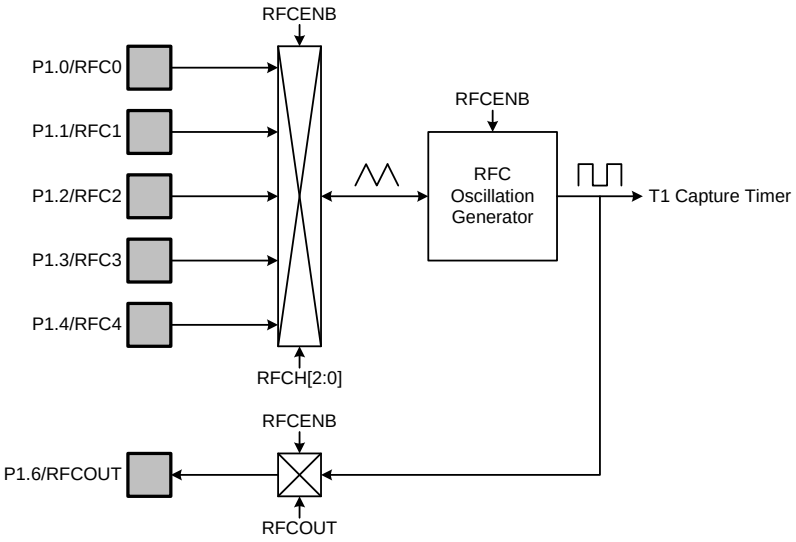
- **T1 EVENT COUNTER FOR CONTINUOUS SIGNAL MEASUREMENT CONFIGURATION:**

```
; Reset T1 timer.
        CLR     T1M             ; Clear T1M register.

; Set T1 clock rate and select/enable T1 capture timer.
        MOV     A, #0nnnm000b   ; "nnn" is T1rate[2:0] for T1 clock rate selection.
        B0MOV   T1M, A          ; "m" is T1 clock source control bit.
        MOV     A, #00000001b   ; Set capture timer function.
        B0MOV   T1CKSM, A       ; CPTG[1:0] must be set as "01".
                                ; "High pulse width measurement"
; Clear T1CH, T1CL.
        CLR     T1CH            ; Clear high byte first.
        CLR     T1CL            ; Clear low byte.

; Set T1VCH, T1VCL 10-bit capture timer for continuous signal measurement.
        MOV     A, #value1      ; Set high nibble first.
        B0MOV   T1VCH, A
        MOV     A, #value2      ; Set low byte.
        B0MOV   T1VCL, A

; Clear T1IRQ
        B0BCLR  FT1IRQ

; Enable T1 timer, interrupt function and T1 capture timer function.
        B0BSET  FT1IEN          ; Enable T1 interrupt function.
        B0BSET  FT1ENB          ; Enable T1 timer.
        B0BSET  FCPTVC          ; Enable T1 event counter function.
; Set capture timer start bit.
        B0BSET  FCPTStart
```

9 RESISTANCE TO FREQURNCY CONVERTER (RFC)

9.1 OVERVIEW

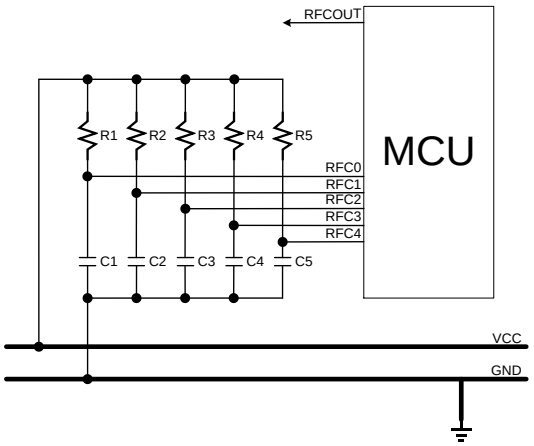
The MCU builds in resistance to frequency converter (RFC) CR type oscillation converter. The RFC conversion circuit is connecting a resistive sensor, reference resistance and a capacitor. Resistance value of the resistive sensor or reference resistance connected to the RFC channel input is converted into frequency by CR oscillator and the number of clocks is counted in the built-in measurement timer/counter (T1). Reading the value of T1 measurement obtains the digitally converting value detected by the resistance. Various sensor circuits such as temperature measurement using a thermistor can easily realize using the RFC. The RFC includes 5 channels. One channel is used reference channel, and the other channels are sensor input channels. For different resistive sensor, the RFC clock frequency is different. T1 provides pulse width measurement and input frequency measurement functions to measure high/low speed RFC clock. RFC pin is shared with port 1. When RFCENB = 1, P1.0~P1.4 are selected to RFC0~RFC4 channel through RFCH[2:0] bits. P1.6 is shared with RFCOUT pin controlled by RFCOUT bit of RFCM register. Each of RFC channels is connected to T1capture timer function to measure RFC frequency and does RFC oscillation. These RFC pins are shared with GPIO controlled by RFCM register. RFC function actives under green mode, but not support wake-up function.



- **RFC Channel:** RFC channels are shared with GPIO controlled by RFCENB = 1 and RFCH[2:0] selected. RFCENB=1 is necessary, or the RFC channel selected by RFCH[2:0] is GPIO mode.
- **RFCOUT pin:** RFC output pin is shared with GPIO controlled by RFCOUT bit. RFC output pin outputs RFC oscillating signal through RFC oscillator generator processing. The signal also inputs to T1 capture timer.

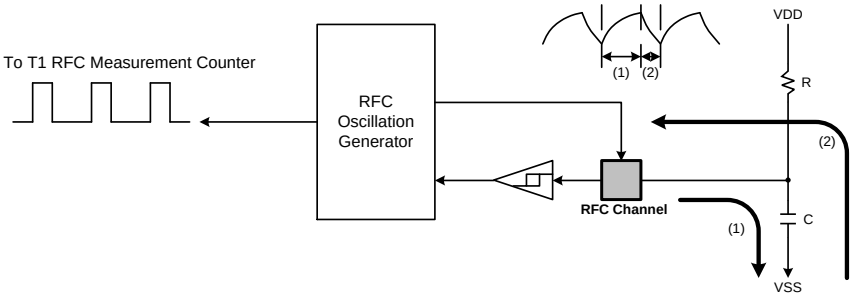
9.2 RFC APPLICATION CIRCUIT

RFC application is configured a resistive device and a capacitor. Normally, the resistor is a sensor type device and the capacitor is a static device. The connection diagram is as following. The resistor connects from Vdd to RFC channel, and the capacitor connects from Vss to RFC channel. When RFC function actives, the internal RFC oscillation generator makes the RFC channel oscillating. The frequency depends on R and C value.



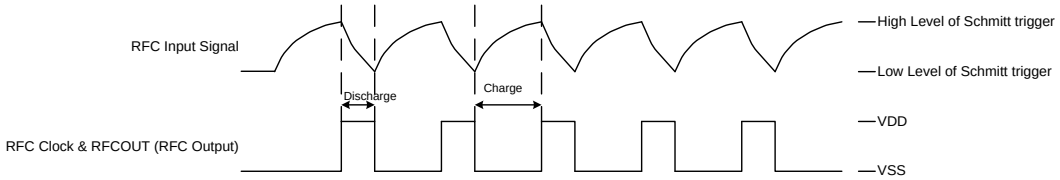
These RC circuits are switched by RFCH[2:0] bits of RFCM register. If one RFC input is selected, the other pins switch to GPIO mode. Recommend set RFC0~RFC4 (P1.0~P1.4) as input floating. RFC digitally converting clock is output from RFCOUT pin (P1.6) controlled by RFCOUT bit of RFC register.

9.3 RFC OPERATION



RFC operation is base on charge and discharge capacitor to obtain the converting clock. The RFCENB=1 turns on RFC function. RFC channel charges the external RC circuit until the voltage level higher than ViH of RFC cahnnel. Then RFC channel exchanges to discharge operation until voltage level under ViL. The charge/discharge operation obtains the RFC oscillation.

Loop (1) of above figure is charge mode. The capacitor is charged by RFC channel, and the voltage level increases. RFC channel detects the voltage level to Schmitt trigger high-level voltage, and then RFC channel switches to discharge mode. Loop (2) is discharge mode. RFC channel discharges and the voltage level starts to fall down. RFC channel detects the voltage level to Schmitt trigger low-level voltage, and then RFC channel switches to charge mode. The result of charge and discharge switching is converted by RC oscillation control circuit and generate digitally clock.



The reference capacitor value is steady. The RFC clock frequency value depends on resistance value and determines charge and discharge rate. Using the RFC clocks of reference resistance and resistive sensor can measure the sensor resistance by the frequency of RFC converting result. The RFC clock can be T1 clock source to measure RFC converting value by frequency measurement and pulse width measurement. The RFC clock also can output to RFCOUT pin (P1.6) when RFCOUT =1.

9.4 RFCM REGISTER

0A6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RFCM	RFCENB	-	**0**	**1**	RFCOUT	RFCH2	RFCH1	RFCH0
Read/Write	R/W	-	**W**	**W**	R/W	R/W	R/W	R/W
After reset	0	-	**0**	**0**	0	0	0	0

Bit 7 **RFCENB:** RFC function control bit.
0 = Disable RFC function.
1 = Enable RFC function.

Bit 5 **RFCM.5 must be set as "0" by program.**
Bit 4 **RFCM.4 must be set as "1" by program.**

Bit 3 **RFCOUT:** RFC output control bit.
0 = Disable RFC output, P1.6 is general purpose I/O.
1 = Enable RFC output, P1.6 is RFCOUT pin.

Bit[2:0] **RFCH[2:0]:** RFC input channels select bit.
000 = Select RFC0 channel. Disable P1.0 GPIO function. P1.1, P1.2, P1.3, P1.4 are GPIO mode.
001 = Select RFC1 channel. Disable P1.1 GPIO function. P1.0, P1.2, P1.3, P1.4 are GPIO mode.
010 = Select RFC2 channel. Disable P1.2 GPIO function. P1.0, P1.1, P1.3, P1.4 are GPIO mode.
011 = Select RFC3 channel. Disable P1.3 GPIO function. P1.0, P1.1, P1.2, P1.4 are GPIO mode.
100 = Select RFC4 channel. Disable P1.4 GPIO function. P1.0, P1.1, P1.2, P1.3 are GPIO mode.
101~111 = Reserved. P1.0, P1.1, P1.2, P1.3, P1.4 are GPIO mode.

✸ ***Note:***

1. ***Before RFC enable, P1.0~P1.4 must be set as input mode without pull-up resistor first by program.***
2. ***RFCM.4 must be set as "1" by program through MOV/B0MOV instruction because the bit is write only type.***
3. ***RFCM.5 must be set as "0" by program through MOV/B0MOV instruction because the bit is write only type.***
4. ***We strongly recommend controlling RFCM register through MOV/B0MOV instructions.***

9.5 RFC OPERATION EXPLAME

➢ **Example: Measure RFC signal through T1 event counter function.**

```
; Set T1 timer mode.
        CLR     T1M             ; T1 clock is Fcpu and rate is Fcpu/128.
        MOV     A, #0           ; Clear T1 counter buffers.
        B0MOV   T1CH, A
        B0MOV   T1CL, A

; Set T1 event counter mode.
        MOV     A, #10001000B   ; Configure and enable T1 capture timer.
        B0MOV   T1CKSM, A
        MOV     A, #0           ; Set T1 event counter buffers.
        B0MOV   T1VCH, A
        MOV     A, #0xFF
        B0MOV   T1VCL, A

        B0BCLR  FT1IRQ          ; Clear T1 interrupt request flag.

; Set RFC.
        CLR     P1UR            ; Disable P1 pull-up resistor.
        MOV     A, #11100000B
        AND     P1M, A          ; Set P1.0~P1.4 to input mode.
        MOV     A, #10010000B   ; Enable RFC and select RFC channel 0.
        B0MOV   RFCM, A

        B0BSET  FT1ENB          ; Enable T1 timer.
        B0BSET  FCPTStart       ; Start to measure RFC frequency.

; Check T1IRQ=1.
Chk_T1IRQ:
        B0BTS1  FT1IRQ          ; Check T1IRQ=1.
        JMP     Chk_T1IRQ
        B0MOV   A, T1CL         ; The end of RFC measurement.
        …
        B0MOV   A, T1CH
        …
```

10 4x32 LCD DRIVER

10.1 OVERVIEW

The LCD driver density is 4x32 (4 commons and 32 segments, 128 dots) and builds in C-type and R-type structure. The LCD supports 1/4 duty and 1/2, 1/3 bias LCD panel. The LCD frame rate is 64Hz and clock source is external 32768Hz oscillator crystal or RC type. The C-type only supports 1/3 bias LCD structure. R-type is using external bias circuit to adjust LCD power and bias voltage. There are 16-pin GPIO shared with SEGs controlled by register. For difference density LCD panel, users can decides more GPIO pins for application.

10.2 LCD REGISTERS

0CBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM	CPCK1	CPCK0	VLCDCP	PSEG2	PSEG1	PSEG0	BIAS	LCDENB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:6] **CPCK[1:0]:** VLCD charge-pump clock selection.

CPCK[1:0]	Charge-pump Clock
00	32KHz
01	16KHz
10	4KHz
11	1KHz

✸ ***Note: In general speaking, 1KHz charge-pump clock is enough for most application. Lower charge-pump clock frequency can save more power consumption. Higher charge-pump clock frequency can provide stronger VLCD driving capability.***

Bit 5 **VLCDCP:** VLCD charge-pump control bit.
0 = Disable. LCD is R-type.
1 = Enable. LCD is C-type.

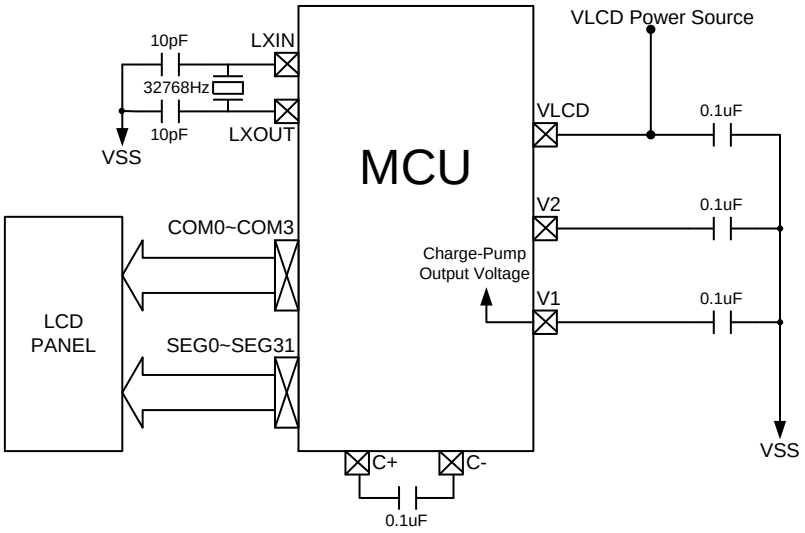
Bit [4:2] **PSEG2:** LCD shared pin control bit.
000 = Disable all LCD shared pins' GPIO function. The SEG pin number is 32.
001 = Enable P2.0~P2.3 (SEG28~SEG31) GPIO function. The SEG pin number is 28.
010 = Enable P2.0~P2.7 (SEG24~SEG31) GPIO function. The SEG pin number is 24.
011 = Enable P2.0~P2.7 and P3.0~P3.3 (SEG20~31) GPIO function. The SEG pin number is 20.
100 = Enable P2.0~P2.7 and P3.0~P3.7 (SEG16~31) GPIO function. The SEG pin number is 16.
101~111 = Reserved.

Bit 1 **BIAS:** LCD bias control bit.
0 = 1/3 bias (C type and R type).
1 = 1/2 bias (R type only).

Bit 0 **LCDENB:** LCD control bit.
0 = Disable.
1 = Enable.

10.3 C-TYPE LCD MODE

In C-type mode only support 1/3 bias LCD panel. The LCD power (VLCD) is supplied by internal charge-pump. The power consumption of C-type mode is less than R-type, because no external DC bias circuit expenses power current. The charge-pump voltage level is following VLCD voltage. V1 is the charge pump source which level is 1/3*VLCD. V2 is 2 times of V1 by charge pump which level is 2/3*VLCD. In C-type LCD mode, the VLCDCP bit of VLCD register must be “1”.



Basic C type LCD Application Circuit

- LCD C-type mode only supports 1/3 bias.
- In C-type mode, connect a 0.1uF capacitor between C+ and C- pins.
- The 0.1uF capacitors of VLCD/V1/V2 pins are necessary for power stable.
- V1 voltage is charge-pump source voltage and 1/3*VLCD.
- V2 voltage is 2*V1 = 2/3*VLCD.

✸ ***Note: The VLCD voltage source is from external power source depended on LCD panel power level, and VLCD voltage level can’t be larger than MCU’s Vdd.***

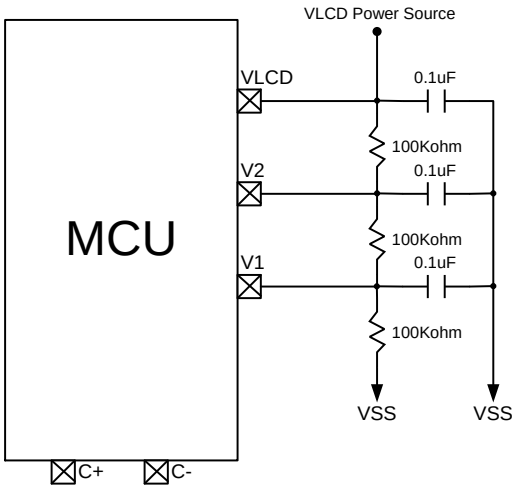
➢ **Example : The configuration of C-type LCD mode.**

```
; Set C-type LCD.
        MOV     A, #nn0mmm00B   ; “nn” selects charge pump clock rate.
        B0MOV   LCDM            ; “mmm” controls P2/P3 LCD shared pins.
                                ; BIAS = 0 for 1/3 bias structure.
; Enable charge pump.
        B0BSET  FVLCDCP         ; Enable LCD charge-pump and LCD is switched to
                                ; C-type mode.
; Enable LCD driver.
        B0BSET  FLCDENB         ; Enable LCD driver..

; LCD picture process.
        ...
        ...
```

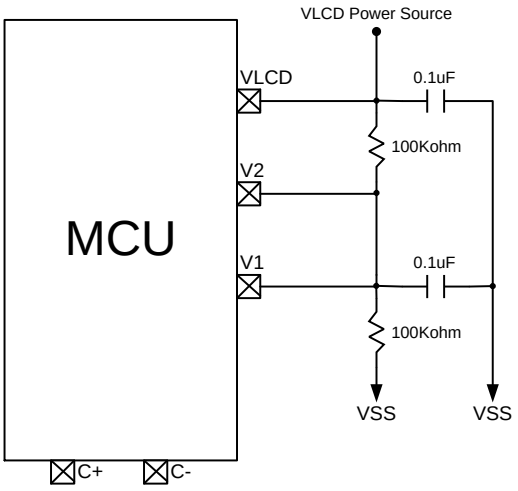
10.4 R-TYPE LCD MODE

In R-type mode, LCD power (VLCD) is connected with internal VDD. V1, V2 bias voltage is controlled by external resistor. The bias resistors determine LCD V1, V2 bias voltage and LCD driver current. Too much current makes LCD panel to bring remnant images. In normal condition, the suggestion of the external bias resistor's value is 100Kohm. In R-type LCD mode, the VLCDCP bit of VLCD register must be "0".

LCD Bias	VLCD	V2	V1
1/3 Bias (Bias=0)	VDD	2/3*VDD	1/3*VDD
1/2 Bias (Bias=1)	VDD	1/2*VDD	1/2*VDD



1/3 bias, 1/4 duty, R-type LCD Circuit.



1/2 bias, 1/4 duty, R-type LCD Circuit.

- In R-type mode, C+/C- don't connect any devices.
- 1/2 bias, V2 connects to V1.
- The 0.1uF capacitors of VLCD/V1/V2 pins are necessary for power stable.

✷ ***Note: Note: The VLCD voltage source is from external power source depended on LCD panel power level, and VLCD voltage level can't be larger than MCU's Vdd.***

➢ **Example : The configuration of R-type LCD mode.**

```
; Set C-type LCD.
        MOV     A, #000mmmn0B   ; "n" selects bias.
        B0MOV   LCDM            ; "mmm" controls P2/P3 LCD shared pins.

; Enable LCD driver.
        B0BSET  FLCDENB         ; Enable LCD driver..

; LCD picture process.
        ...
        ...
```

10.5 LCD RAM MAP

LCD dots are controlled by LCD RAM in Bank15. Program the LCD RAM data is using index pointer in bank 0 or directly addressing in bank 15. The LCD RAM placement is by LCD segments. One segment address includes four common bits data. COM0~COM3 is in low-nibble of one LCD RAM (bit0~bit3). The high-nibble of one LCD RAM is useless. The LCD RAM map is as following.

RAM bank 15 address vs. Common/Segment location.

RAM	Bit	0	1	2	3	4	5	6	7
Address	LCD	COM0	COM1	COM2	COM3	-	-	-	-
00h	SEG0	00h.0	00h.1	00h.2	00h.3	-	-	-	-
01h	SEG1	01h.0	01h.1	01h.2	01h.3	-	-	-	-
02h	SEG2	02h.0	02h.1	02h.2	02h.3	-	-	-	-
03h	SEG3	03h.0	03h.1	03h.2	03h.3	-	-	-	-
.	.	.	.	.	.	-	-	-	-
.	.	.	.	.	.	-	-	-	-
.	.	.	.	.	.	-	-	-	-
0Ch	SEG12	0Ch.0	0Ch.1	0Ch.2	0Ch.3	-	-	-	-
0Dh	SEG13	0Dh.0	0Dh.1	0Dh.2	0Dh.3	-	-	-	-
0Eh	SEG14	0Eh.0	0Eh.1	0Eh.2	0Eh.3	-	-	-	-
0Fh	SEG15	0Fh.0	0Fh.1	0Fh.2	0Fh.3	-	-	-	-
10h	SEG16	10h.0	10h.1	10h.2	10h.3	-	-	-	-
.	.	.	.	.	.	-	-	-	-
.	.	.	.	.	.	-	-	-	-
.	.	.	.	.	.	-	-	-	-
1Bh	SEG27	1Bh.0	1Bh.1	1Bh.2	1Bh.3	-	-	-	-
1Ch	SEG28	1Ch.0	1Ch.1	1Ch.2	1Ch.3	-	-	-	-
1Dh	SEG29	1Dh.0	1Dh.1	1Dh.2	1Dh.3	-	-	-	-
1Eh	SEG30	1Eh.0	1Eh.1	1Eh.2	1Eh.3	-	-	-	-
1Fh	SEG31	1Fh.0	1Fh.1	1Fh.2	1Fh.3	-	-	-	-

➢ **Example : Set LCD RAM data by index pointer (@YZ) in bank 0.**

```
B0MOV     Y, #0FH          ; Set @YZ point to LCD RAM address 0x1500.
CLR       Z

MOV       A, #00001010B    ; Set COM0=0, COM1=1, OCM2=0,COM3=1 of SEG 0.
B0MOV     @YZ, A

INCMS     Z                ; Point to next segment address.
…
…
```

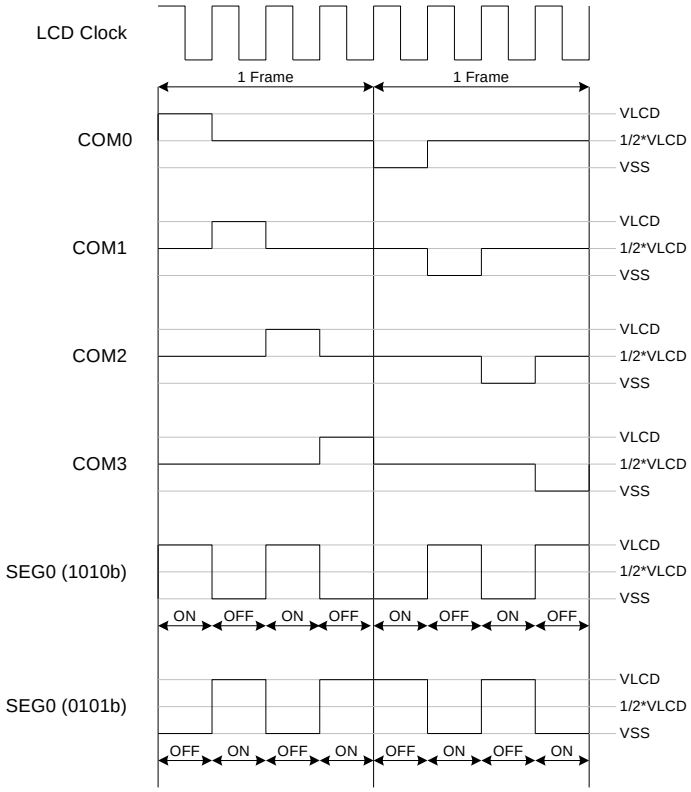
➢ **Example : Set LCD RAM data by directly addressing in bank 15.**

```
MOV       A, #01           ; Switch to RAM bank 15.
B0MOV     RBANK, A

MOV       A, #00001010B    ; Set COM0=0, COM1=1, OCM2=0,COM3=1 of SEG 0.
MOV       00H, A

BCLR      01H.0            ; Clear COM0=0 of SEG 1.
BSET      01H.1            ; Clear COM1=1 of SEG 1.
…

MOV       A, #0            ; Switch to RAM bank 0.
B0MOV     RBANK, A
```

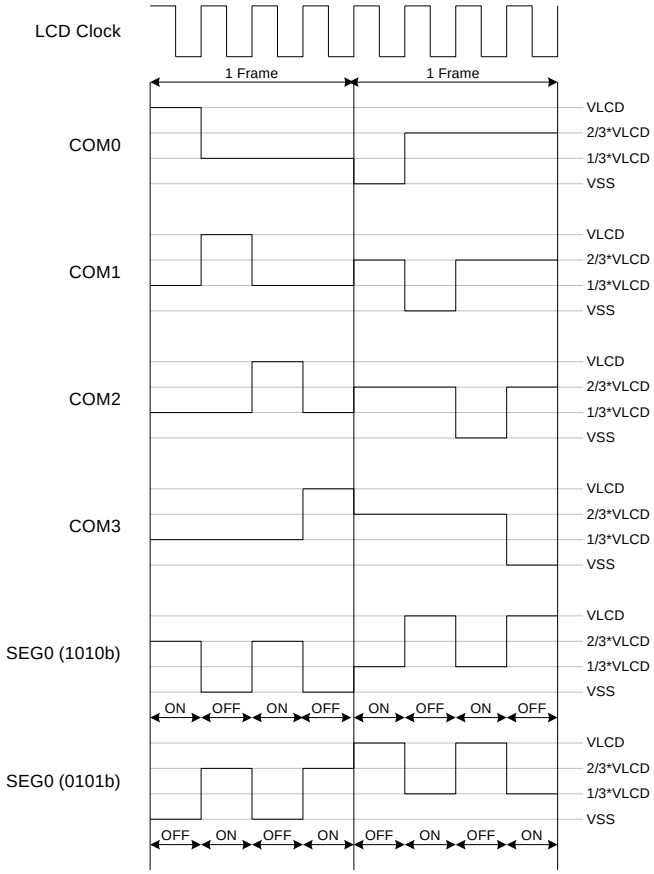
✸ ***Note: Access RAM data of bank 0 (system registers and user define RAM 0x0000~0x007F) is using "B0xxx" instructions.***

10.6 LCD WAVEFORM

1/2 bias, 1/4 duty



1/3 bias, 1/4 duty



11 INSTRUCTION TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
	MOV	A,M	A ← M	-	-	√	1
M	MOV	M,A	M ← A	-	-	-	1
O	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
V	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
E	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I, “M” only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z…)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
A	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1+N
R	ADD	A,M	A ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
I	ADD	M,A	M ← A + M, if occur carry, then C=1, else C=0	√	√	√	1+N
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
H	ADD	A,I	A ← A + I, if occur carry, then C=1, else C=0	√	√	√	1
M	SBC	A,M	A ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
E	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
T	SUB	A,M	A ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
I	SUB	M,A	M ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1+N
C	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
	DAA		To adjust ACC’s data format from HEX to DEC.	√	-	-	1
	AND	A,M	A ← A and M	-	-	√	1
L	AND	M,A	M ← A and M	-	-	√	1+N
O	AND	A,I	A ← A and I	-	-	√	1
G	OR	A,M	A ← A or M	-	-	√	1
I	OR	M,A	M ← A or M	-	-	√	1+N
C	OR	A,I	A ← A or I	-	-	√	1
	XOR	A,M	A ← A xor M	-	-	√	1
	XOR	M,A	M ← A xor M	-	-	√	1+N
	XOR	A,I	A ← A xor I	-	-	√	1
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
R	RRC	M	A ← RRC M	√	-	-	1
O	RRCM	M	M ← RRC M	√	-	-	1+N
C	RLC	M	A ← RLC M	√	-	-	1
E	RLCM	M	M ← RLC M	√	-	-	1+N
S	CLR	M	M ← 0	-	-	-	1
S	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
B	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
R	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1+ S
A	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
N	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1+ S
C	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1+N+S
H	BTS0	M.b	If M.b = 0, then skip next instruction	-	-	-	1 + S
	BTS1	M.b	If M.b = 1, then skip next instruction	-	-	-	1 + S
	B0BTS0	M.b	If M(bank 0).b = 0, then skip next instruction	-	-	-	1 + S
	B0BTS1	M.b	If M(bank 0).b = 1, then skip next instruction	-	-	-	1 + S
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
M	RET		PC ← Stack	-	-	-	2
I	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
S	PUSH		To push ACC and PFLAG (except NT0, NPD bit) into buffers.	-	-	-	1
C	POP		To pop ACC and PFLAG (except NT0, NPD bit) from buffers.	√	√	√	1
	NOP		No operation	-	-	-	1

Note: ***1. “M” is system register or RAM. If “M” is system registers then “N” = 0, otherwise “N” = 1.***
2. If branch condition is true then “S = 1”, otherwise “S = 0”.

12 ELECTRICAL CHARACTERISTIC

12.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2318F, SN8P2317F..... 0°C ~ + 70°C
SN8P2318FD, SN8P2317FD..... -40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

12.2 ELECTRICAL CHARACTERISTIC

- **DC CHARACTERISTIC**

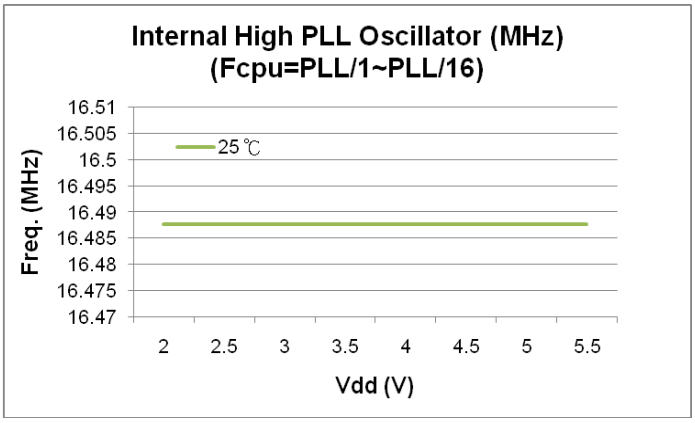
(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

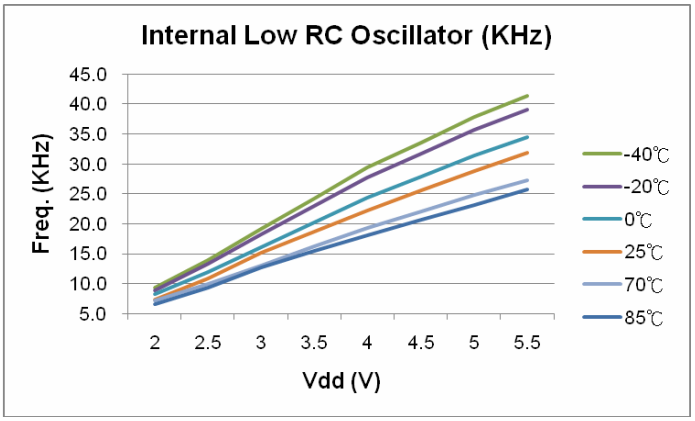
PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode. Fcpu = 1MHz		2.2	-	5.5	V
		Normal mode. Fcpu = 4MHz		2.4	-	5.5	
RAM Data Retention voltage	Vdr			1.5	-	-	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.8Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	150	
I/O output source current	IoH	Vop = Vdd – 0.5V		8	-	-	mA
sink current	IoL	Vop = Vss + 0.5V		8	-	-	
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current	Idd1	Run Mode (LCD disable)	Vdd= 3V, PLL/4 = 4MHz	-	2.5	-	mA
			Vdd= 5V, PLL/4 = 4MHz	-	4.5	-	mA
			Vdd= 3V, PLL/16 = 1MHz	-	1.5	-	mA
			Vdd= 5V, PLL/16 = 1MHz	-	3	-	mA
			Vdd= 3V, 16M X'tal/4 = 4MHz	-	2.5	-	mA
			Vdd= 5V, 16M X'tal/4 = 4MHz	-	5	-	mA
			Vdd= 3V, 4M X'tal/4 = 1MHz	-	1	-	mA
			Vdd= 5V, 4M X'tal/4 = 1MHz	-	2.5	-	mA
	Idd2	Slow Mode (LCD disable)	Vdd= 3V, Ext. 32K/4	-	5	-	uA
			Vdd= 5V, Ext. 32K/4	-	15	-	uA
	Idd3	Sleep Mode (LCD disable)	Vdd= 5V/3V	-	-	2	uA
	Idd4	Green Mode (LCD disable)	Vdd= 3V, PLL	-	0.6	-	mA
			Vdd= 5V, PLL	-	0.9	-	mA
			Vdd= 3V, 16M X'tal	-	0.6	-	mA
			Vdd= 5V, 16M X'tal	-	1.6	-	mA
			Vdd= 3V, 4M X'tal	-	0.2	-	mA
			Vdd= 5V, 4M X'tal	-	0.5	-	mA
			Vdd= 3V, Ext. 32KHz X'tal	-	2.5	-	uA
			Vdd= 5V, Ext. 32KHz X'tal	-	9	-	uA
	Idd5	Green Mode (LCD enable, no panel).	Vdd= 3V, Ext. 32KHz X'tal	-	5.5	-	uA
			Vdd= 5V, Ext. 32KHz X'tal	-	15	-	uA
Internal PLL Oscillator	Fpll	PLL 16MHz	*25℃*, Vdd=2.2V~ 5.5V Fcpu=Fhosc/1	16.11	16.78	17.45	MHz
			25℃, Vdd=2.2V~ 5.5V Fcpu=Fhosc/2~Fhosc/16	16.44	16.78	17.12	MHz
LVD Voltage	Vdet0	Low voltage reset level. *25℃*		1.9	2.0	2.1	V
	Vdet1	Low voltage reset/indicator level. *25℃*		2.3	2.4	2.5	V
	Vdet2	Low voltage reset/indicator level. *25℃*		3.5	3.6	3.7	V

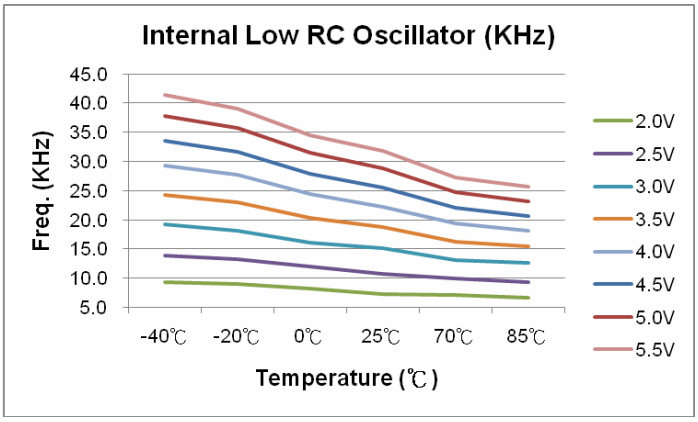
" *" These parameters are for design reference, not tested.

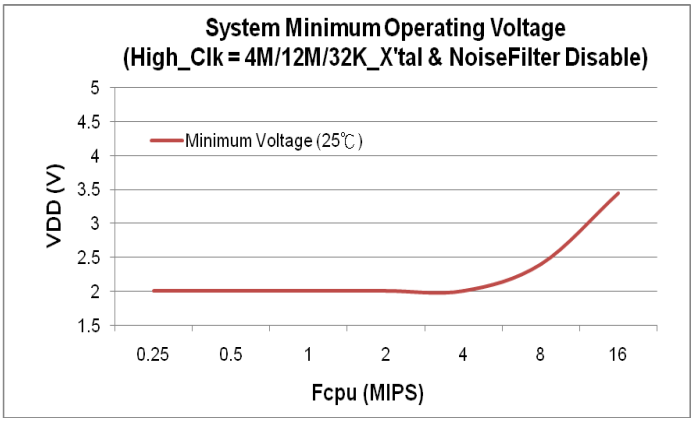
12.3 CHARACTERISTIC GRAPHS

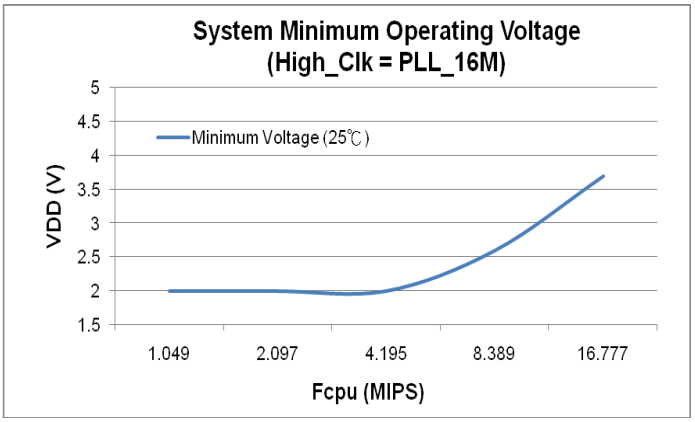
The Graphs in this section are for design guidance, not tested or guaranteed. In some graphs, the data presented are outside specified operating range. This is for information only and devices are guaranteed to operate properly only within the specified range (-40℃~+85℃ curves are for design reference).

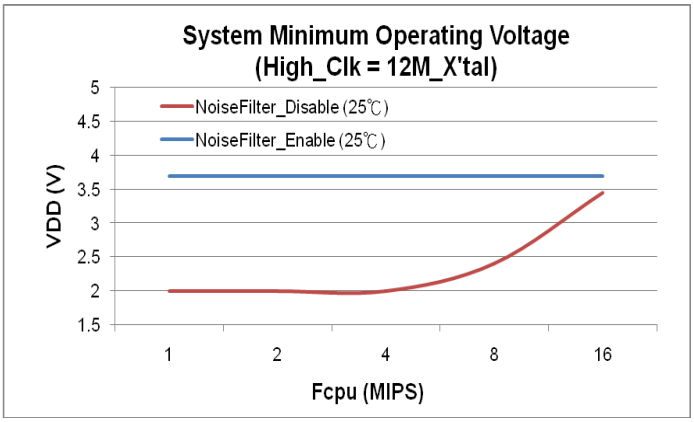












13 DEVELOPMENT TOOL

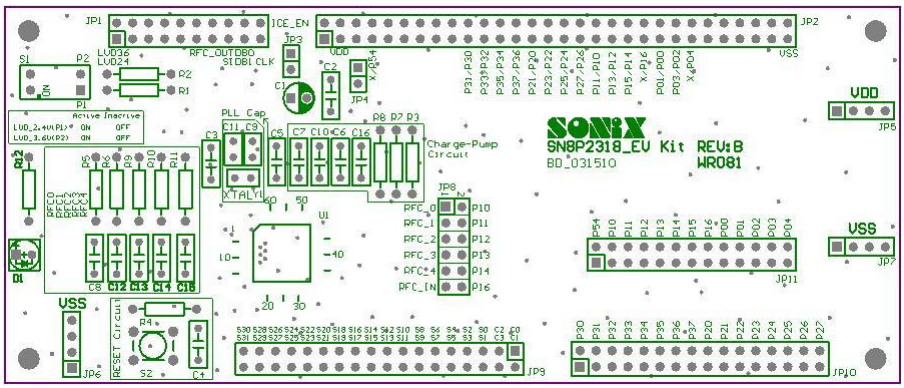
SONIX provides ICE (in circuit emulation), IDE (Integrated Development Environment) and EV-kit for SN8P2318 development. ICE and EV-kit are external hardware devices, and IDE is a friendly user interface for firmware development and emulation. These development tools' version is as following.

- **ICE: SN8ICE2K Plus II. (Please install 16MHz crystal in ICE to implement PLL_16M emulation.).**
- **ICE emulation speed maximum: 8 MIPS @ 5V (e.g. 16Mhz crystal, Fcpu = Fosc/2).**
- **EV-kit: SN8P2318_EV kit Rev. B.**
- **IDE: SONiX IDE M2IDE_V129 and later version.**
- **Writer: MPIII writer.**
- **Writer transition board: SN8P2318**

13.1 SN8P2318 EV-KIT

SONIX provides SN8P2318 MCU which includes LCD and RFC functions. These functions aren't built in SN8ICE2K Plus 2. To emulate the functions must be through SN8P2318 real chip. The real chip provides an EV-KIT to achieve LCD and RFC functions emulations. For SN8P2318 ICE emulation, the EV-Kit includes LCD/RFC/ LVD2.4V/3.6V and switch circuits.

SN8P2318 EV-kit PCB Outline:

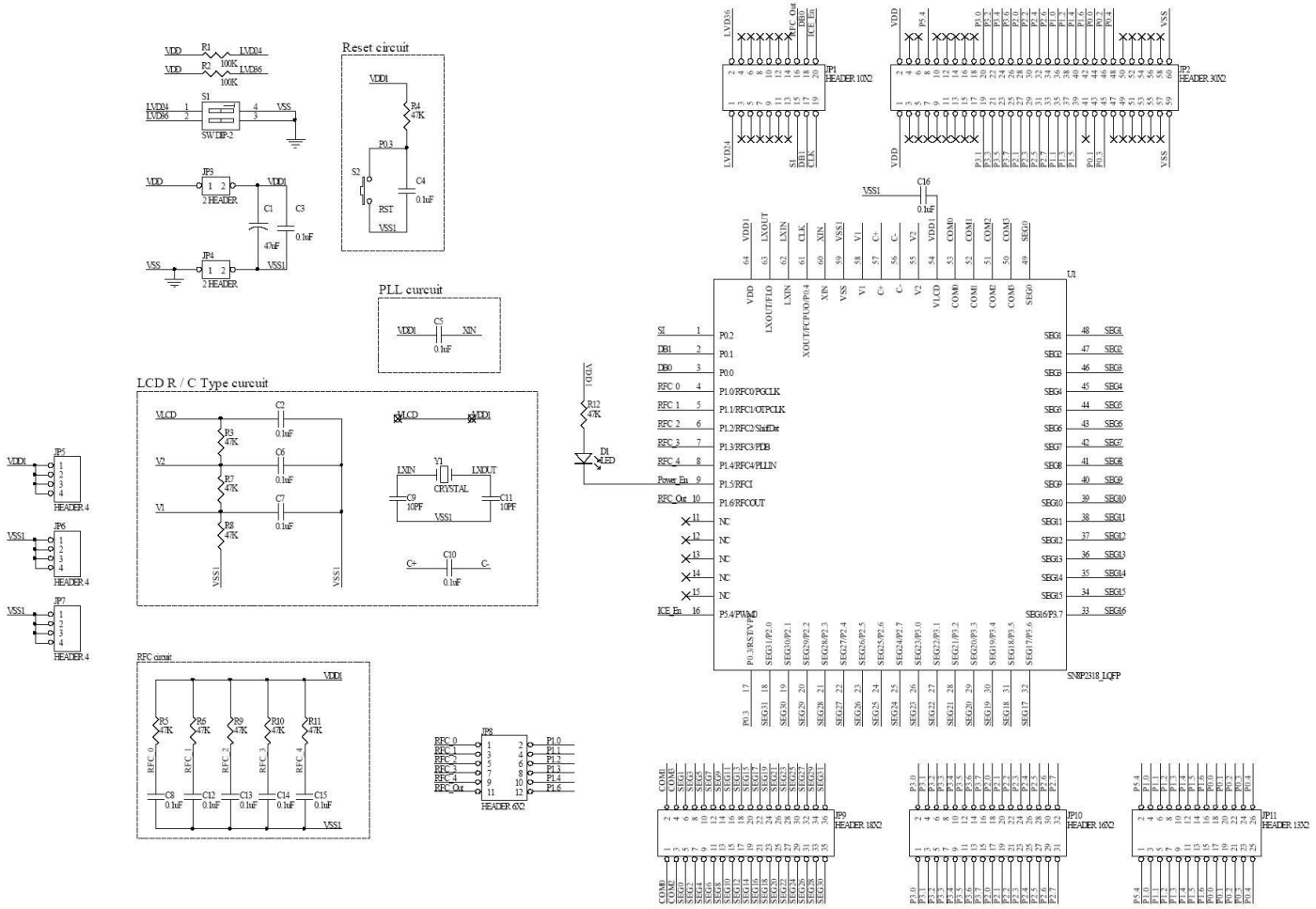


- JP2: Connect to SN8ICE2K Plus 2 CON1 (includes GPIO, EV-KIT control signal, and the others).
- JP1: Connect to SN8ICE2K Plus 2 JP3 (EV-KIT communication bus with ICE, control signal, and the others).
- S1: LVD24V/LVD36V control switch. To emulate LVD2.4V flag/reset function and LVD3.6V/flag function

Switch No.	ON	OFF
LVD24 (1)	LVD 2.4V Active	LVD 2.4V Inactive
LVD36 (2)	LVD 3.6V Active	LVD 3.6V Inactive

- U1: SN8P2318 EV-chip.
- S2: SN8P2318 EV-chip reset key. If EV-KIT active fail, press S2 to reset EV-KIT Real Chip (U1).
- JP8: Keep JP8 as open stauts.
- JP9: LCD connector.
- JP10, JP11: GPIO connector.
- R5, C8: RFC channel 0.
- R6, C12: RFC channel 1.
- R9, C13: RFC channel 2.
- R10, C14: RFC channel 3.
- R11, C15: RFC channel 4.
- C7, C10, C6, C16: LCD decouple capacitors connected 0.1uF.
- R8, R7, R3: R-type LCD resistors.
- C11, C9, Y1: SN8P2318 EV-chip 32KHz crystal circuit.

SN8P2318 EV-kit schematic:

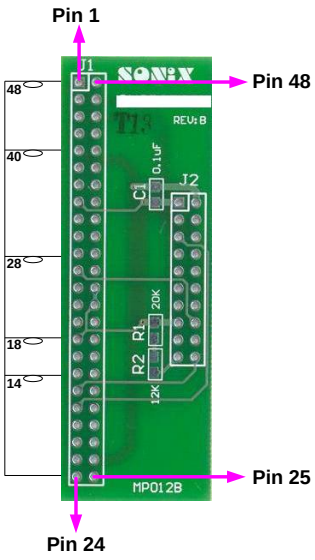


13.2 ICE AND EV-KIT APPLICATION NOTIC

1. SN8ICE2K Plus 2 power switch must be turned off before you connect the SN8P2318 EV-KIT to SN8ICE2K Plus 2.
2. Connect EV-KIT's JP1/JP2 to ICE's JP3/CON1.
3. Turn on SN8ICE2K Plus 2 power switch to start emulation.
4. If the power indicator (LED D1) doesn't light, the EV-kit occurs some mistakes. Please contact SONIX's agent for maintain service.
5. **It is necessary to connect 16MHz crystal in ICE for IHRC_16M mode emulation. SN8ICE2K Plus 2 doesn't support over 8-mips instruction cycle, but real chip does**
6. JP11's P10~P14 and P16 only support GPIO function, not RFC function.
7. JP10 only supports P2 / P3 GPIO function only, not LCD function.
8. JP9 only supports LCD function to plug LCD panel.

14 OTP PROGRAMMING PIN

14.1 WRITER TRANSITION BOARD SOCKET PIN ASSIGNMENT



JP3 (Mapping to 48-pin text tool)

DIP 1	1	48	DIP48
DIP 2	2	47	DIP47
DIP 3	3	46	DIP46
DIP 4	4	45	DIP45
DIP 5	5	44	DIP44
DIP 6	6	43	DIP43
DIP 7	7	42	DIP42
DIP 8	8	41	DIP41
DIP 9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP37
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

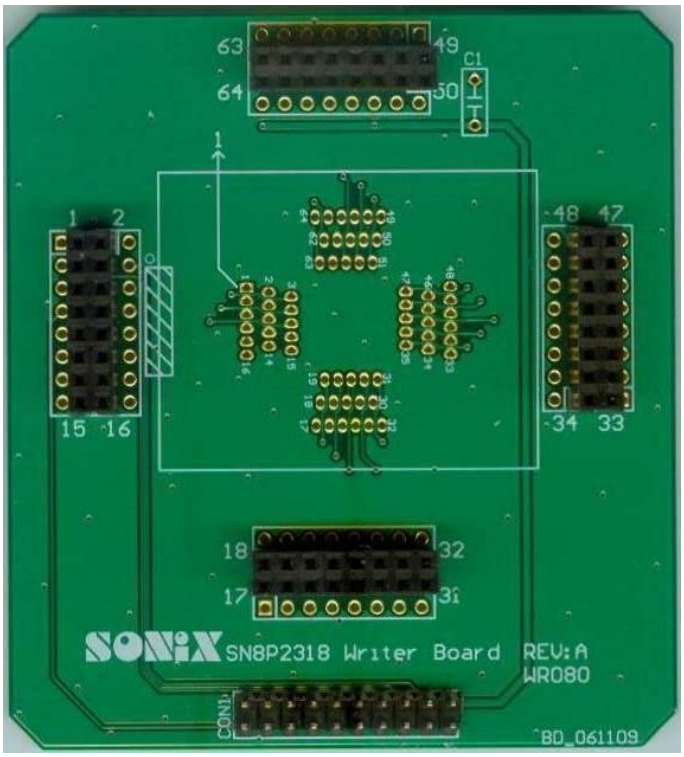
Writer JP1/JP2

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

JP1 for Writer transition board
JP2 for dice and >48 pin package

14.2 SN8P2317 / SN8P2318 OTP PROGRAMMING TOOL

- WR080 is writer board for SN8P2318 LQFP64 OTP programming socket connection.
- MP125 is transition board for SN8P2317 LQFP48 OTP programming socket.



WR080 (Writer Board)

14.3 PROGRAMMING PIN MAPPING:

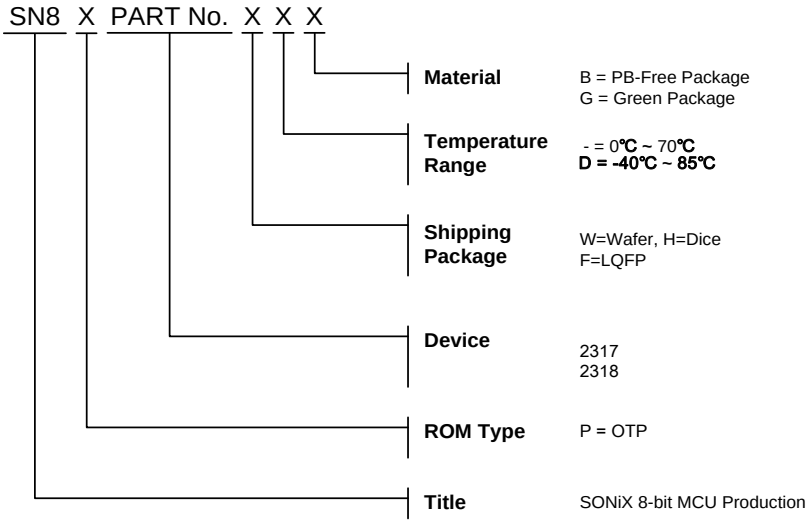
Programming Pin Information of SN8P2318 series							
Chip Name		SN8P2317F(LQFP48)			SN8P2318F(LQFP64)		
Writer Connector		IC and JP3 48-pin text tool Pin Assignment			IC and JP1 20-pin text tool Pin Assignment		
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	-
1	VDD	1	VDD	-	64	VDD	-
2	GND	44	VSS	-	59	VSS	-
3	CLK	5	P1.0	-	4	P1.0	-
4	CE	-	-	-	-	-	-
5	PGM	6	P1.1	-	5	P1.1	-
6	OE	7	P1.2	-	6	P1.2	-
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	13	RST	-	17	RST	-
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	8	P1.3	-	7	P1.3	-

15 Marking Definition

15.1 INTRODUCTION

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

15.2 MARKING INDETIFICATION SYSTEM



15.3 MARKING EXAMPLE

- **Wafer, Dice:**

Name	ROM Type	Device	Package	Temperature	Material
S8P2318W	OTP	2318	Wafer	0℃~70℃	-
SN8P2318H	OTP	2318	Dice	0℃~70℃	-

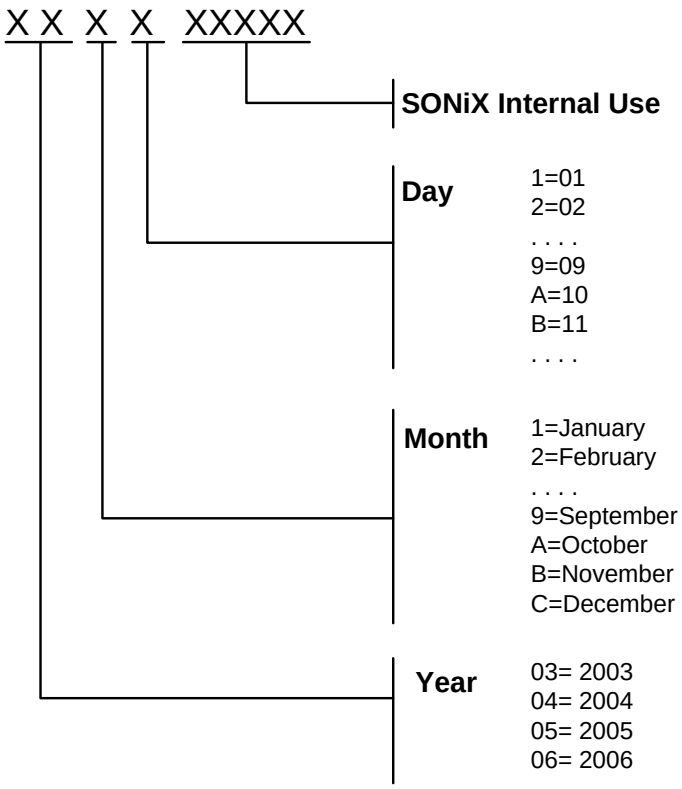
- **Green Package:**

Name	ROM Type	Device	Package	Temperature	Material
SN8P2317FG	OTP	2318	LQFP48	0℃~70℃	Green Package
SN8P2318FG	OTP	2318	LQFP64	0℃~70℃	Green Package
SN8P2317FDG	OTP	2318	LQFP48	-40℃~85℃	Green Package
SN8P2318FDG	OTP	2318	LQFP64	-40℃~85℃	Green Package

- **PB-Free Package:**

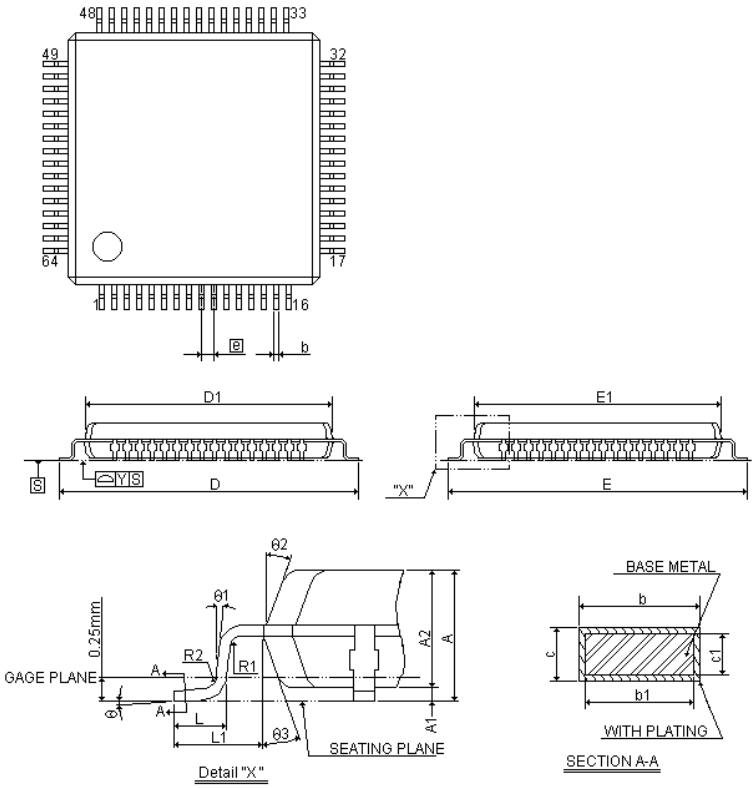
Name	ROM Type	Device	Package	Temperature	Material
SN8P2317FB	OTP	2318	LQFP48	0℃~70℃	PB-Free Package
SN8P2318FB	OTP	2318	LQFP64	0℃~70℃	PB-Free Package
SN8P2317FDB	OTP	2318	LQFP48	-40℃~85℃	PB-Free Package
SN8P2318FDB	OTP	2318	LQFP64	-40℃~85℃	PB-Free Package

15.4 DATECODE SYSTEM



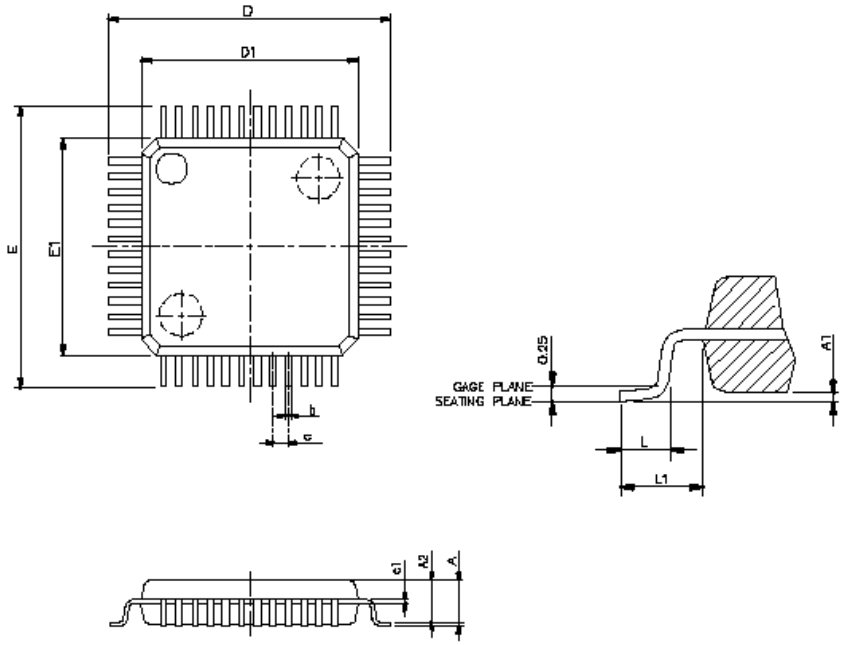
16 PACKAGE INFORMATION

16.1 LQFP 64 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.063	-	-	1.600
A1	0.002	0.055	0.006	0.050	1.400	0.150
A2	0.054	0.009	0.057	1.360	0.220	1.450
b	0.007	0.009	0.011	0.170	0.220	0.270
b1	0.007	-	0.009	0.170	-	0.230
C	0.004	-	0.008	0.090	-	0.200
C1	0.004	-	0.006	0.090	-	0.160
D	0.472			12.000		
D1	0.394			10.000		
E	0.472			12.000		
E1	0.394			10.000		
[e]	0.020			0.500		
L	0.018	0.024	0.030	0.450	0.600	0.750
L1	0.039	-	-	1.000	-	-
R1	0.003	-	-	0.080	-	-
R2	0.003	-	0.008	0.080	-	0.200
Y	-	-	0.030	-	-	0.750
θ°	0°	3.5°	7°	0°	3.5°	7°
θ1°	0°	-	-	0°	-	-
θ2°	11°	12°	13°	11°	12°	13°
θ3°	11°	12°	13°	11°	12°	13°

16.2 LQFP 48 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.06	-	-	1.6
A1	0.00	-	0.01	0.05	-	0.15
A2	0.05	-	0.06	1.35	-	1.45
c1	0.00	-	0.01	0.09	-	0.16
D	0.35 BSC			9.00 BSC		
D1	0.27 BSC			7.00 BSC		
E	0.35 BSC			9.00 BSC		
E1	0.27 BSC			7.00 BSC		
e	0.02 BSC			0.5 BSC		
B	0.01	-	0.01	0.17	-	0.27
L	0.02	-	0.03	0.45	-	0.75
L1	0.04 REF			1 REF		

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 us 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: 10F-1, NO.36, Taiyuan Street, Chupei City, Hsinchu, Taiwan R.O.C.
Tel: 886-3-560 0888
Fax: 886-3-560 0889

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:
Unit 1519, Chevalier Commercial Centre, NO.8 Wang Hoi Road, Kowloon Bay, Hong Kong.
Tel: 852-2723-8086
Fax: 852-2723-9179

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