SN8P2308

USER'S MANUAL

Specification Version 1.2

SONiX 8-Bit Micro-Controller

AMENDMENT HISTORY

Version	Date	Description
VER 0.1	Jan. 2005	First issue.
VER 0.2	Jan. 2005	1. Modify RFC, LCD and T1 timer descriptions. 2. Add development tool chapter.
VER 0.3	Jan. 2005	1. Add RFC to feature selections table. 2. TCL doesn't support "CLR" instruction. Modify T1 and RFC description about T1CL clearing. 3. Modify P.106 T0ENB, T0IRQ, T0IEN to FT0ENB, FT0IRQ, FT0IEN. 4. Fcpu of slow mode is Flosc/1. 5. Add RFC first pulse width measurement example. 6. Modify P1W register of system register table to write only. 7. Correct P.96 typing error: Change "PCH" to "T1CH" and "PCL" to "T1CL" in Note. 8. Change part number to SN8P2308Q and SN8P2308QD in electrical characteristic chapter. 9. Modify supply current values of electrical characteristic table.
VER 0.4	Mar. 2005	1. Modify Z flag description. 2. Correct the LCDENB bit definition. Set "0" to disable LCD driver. 3. LCD driver In C type mode only support 1/3 bias LCD panel. 4. In SN8P2308 EV KIT section: a. Add RFC circuit schematic description b. Add more description about LCD emulation
VER 1.0	May 2005	1. Modify P121 "COM0~COM4" to "COM0~COM3". 2. Modify P77 TC0X8. 3. Modify P11、P70：After power on or reset, P3 default state as input LOW. 4. ADD BROWN-OUT circuit.
VER 1.1	Dec. 2005	1. Modify Topr value. 2. Modify Brown-Out Reset description 3. Remove High clock 32K mode. 4. Modify M2IDE 1.07 5. Add T0 RTC description 6. Remove power consumption(Pc) 7. Add Fcpu limitation by Noise filter enable. 8. Modify ELECTRICAL CHARACTERISTIC.
VER 1.2	Jan. 2007	1. Add Marking Definition. 2. Modify ELECTRICAL CHARACTERISTIC. 3. Modify RFC register to RFCM.

Table of Content

1 PRODUCT OVERVIEW

1.1 FEATURES

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

- **8 levels stack buffer.**

- **I/O pin configuration**
  Bi-directional: P0, P1, P5.
  Bi-directional and shared with SEG pins: P3.
  Input only: P0.2 shared with reset pin.
  Wakeup: P0, P1 level change.
  Pull-up resisters: P0, P1, P3, P5
  External interrupt trigger edge:
  P0.0 controlled by PEDGE.
  Event counter input:
  P0.0 is TC0 event counter input.
  P0.1 is T1 event counter input.
  RFC shared pins:
  RFC0 (P5.0): RFC channel 0 input pin.
  RFC1 (P5.1): RFC channel 1 input pin.
  RFC2 (P5.2): RFC channel 2 input pin.
  RFCI (P5.3): RFC feedback input pin.
  RFCOUT (P5.5): RFC output pin.

- **3-Level LVD**
  For system reset or brief VDD indicator.

- **Powerful instructions**
  Instruction cycle controlled by code option.
  Instruction's length is one word.
  Most of instructions are one cycle only.
  Maximum instruction cycle is two.
  All ROM area JMP and CALL instruction.
  All ROM area CALL address instruction.
  All ROM area lookup table function (MOVC)

- **4*32 LCD Driver (128 dots)**
  Support C type or R type bias

- **RFC (Resistor to Frequency Converter)**
  For 2-channel RC type AD converter

- **Four interrupt sources**
  Three internal interrupts: T0, TC0, T1.
  One external interrupts: INT0.

- **Two 8-bit Timer/Counter**
  T0: Basic timer.
  TC0: Auto-reload timer/counter/PWM0/Buzzer output.

- **One 16-bit timer counters.**
  T1: Timer/counter/RFC measurement and output.

- **On chip watchdog timer and clock source is internal low clock RC type (16KHz @3V, 32KHz @5V).**

- **Dual system clocks**
  External high clock: RC type up to 10MHz.
  External high clock: Crystal type up to 16MHz.
  External low clock: 32KHz crystal/RC type

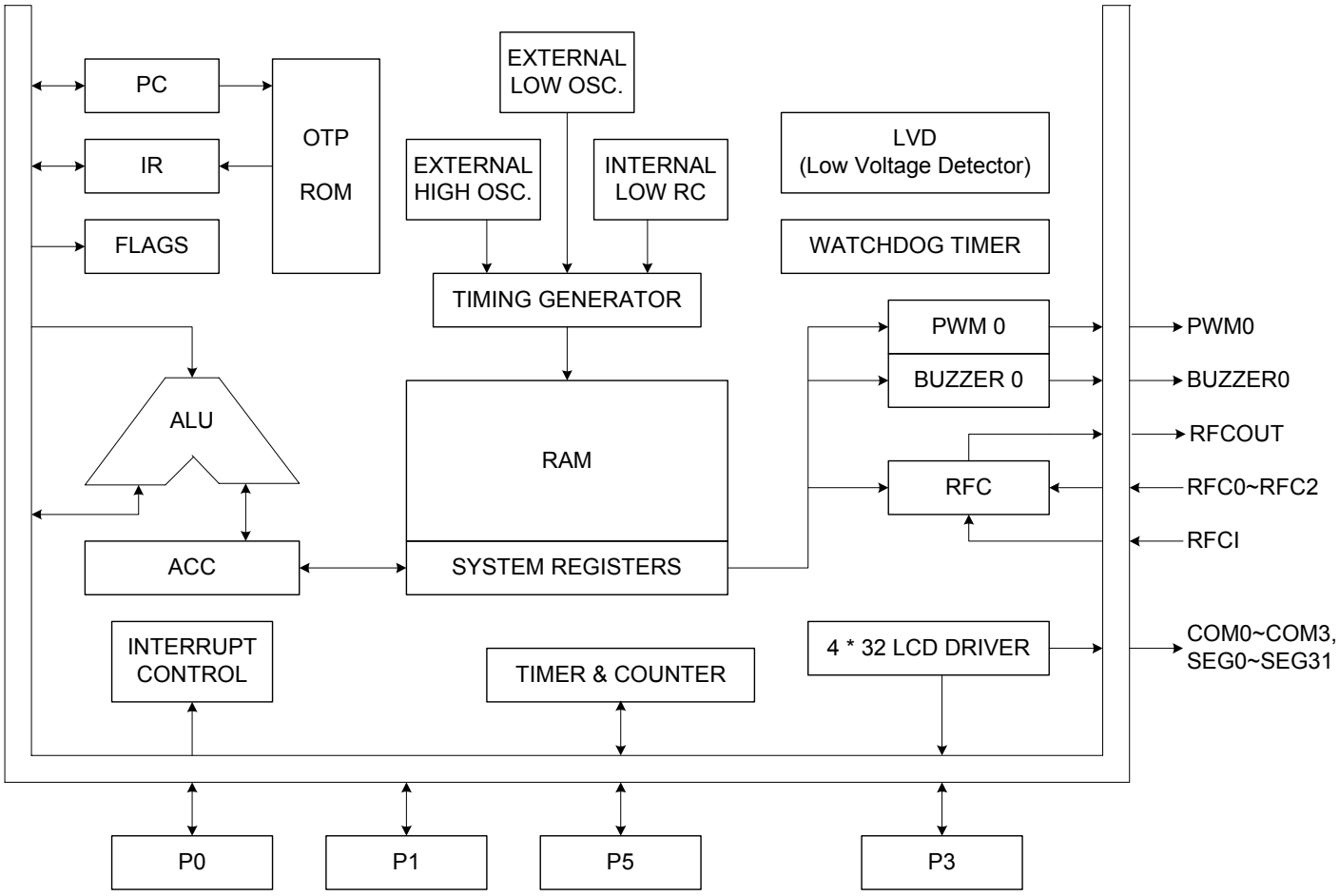
- **Four operation modes**
  Normal mode: Both high and low clock active.
  Slow mode: 32KHz low clock active.
  Sleep mode: Both high and low clock stop.
  Green mode: Periodical wake-up by T0 timer.

- **Package (Chip form support)**
  LQFP 64 pins.

☞ **Features Selection Table**

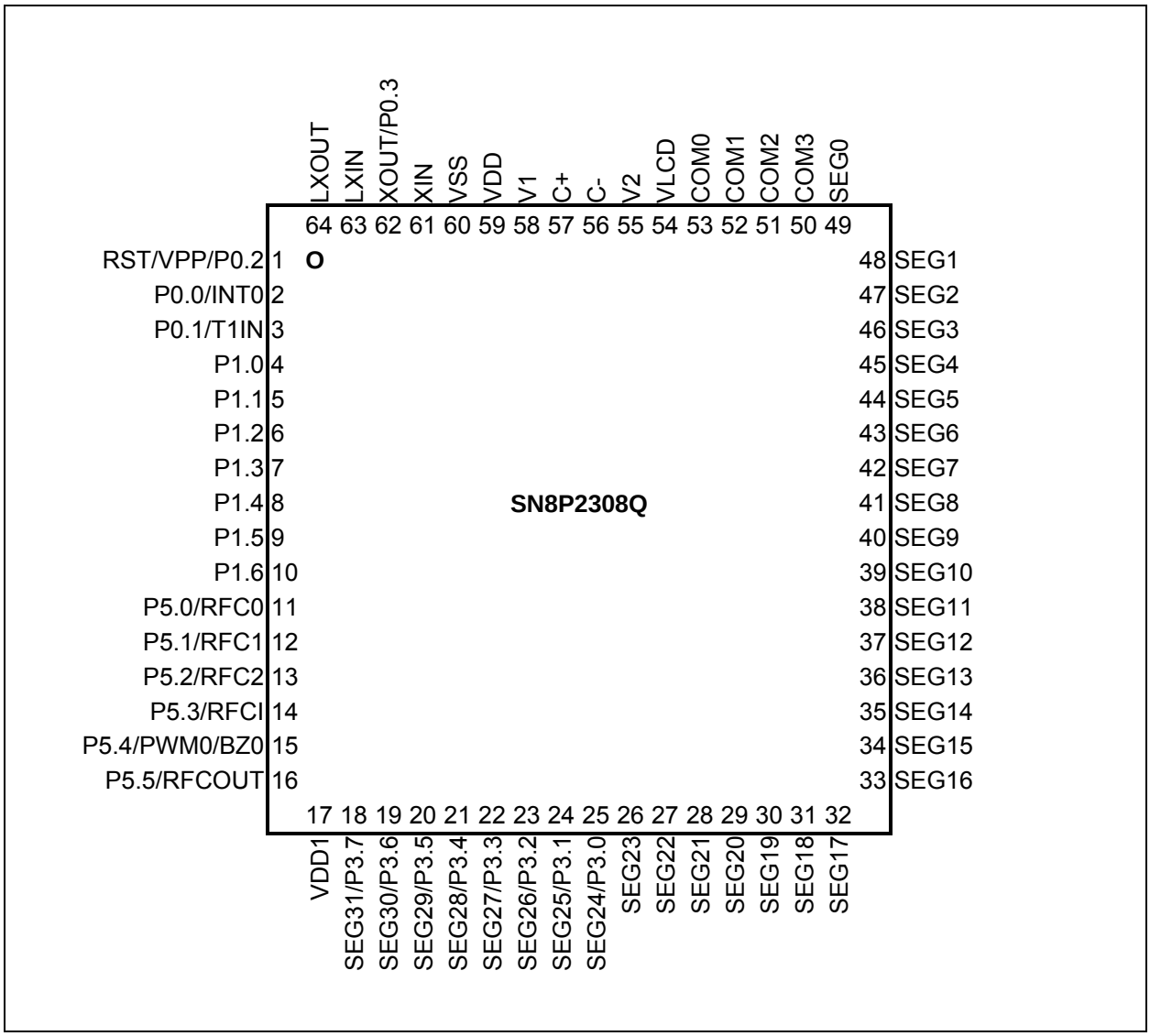
CHIP	ROM	RAM	Stack	Timer			I/O	RFC	Green Mode	PWM	Wakeup Pin No.	LCD Driver	Package
				TC0	T0	T1				Buzzer			
SN8P2308	4K*16	128	8	V	V	V	25	2-ch	V	1	11	4*32	LQFP 64

1.2 SYSTEM BLOCK DIAGRAM



1.3 PIN ASSIGNMENT

SN8P2308Q (LQFP 64 pins)

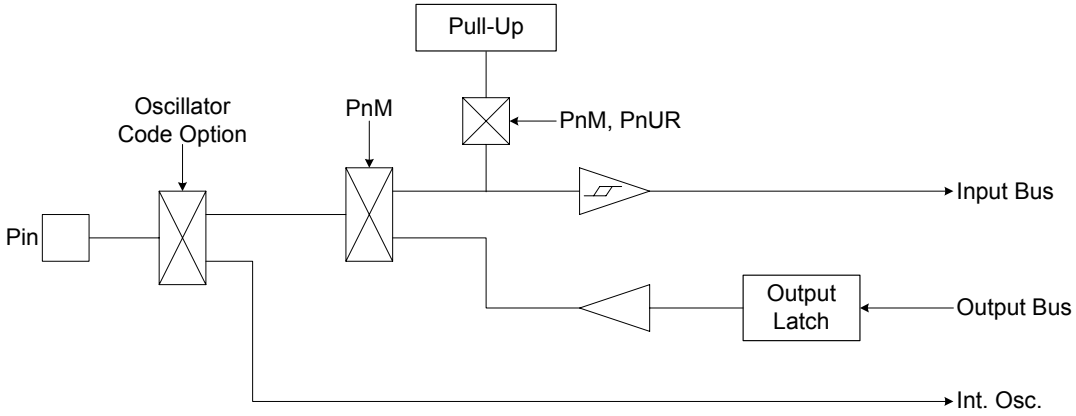


1.4 PIN DESCRIPTIONS

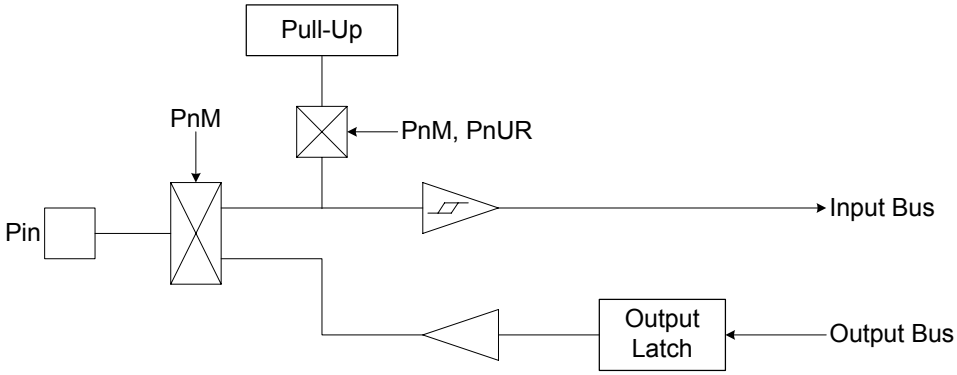
PIN NAME	TYPE	DESCRIPTION
VDD, VSS	P	Power supply input pins for digital circuit.
VDD1	P	Connect to VDD when P3.0~P3.7 I/O enable. Connect to VLCD when SEG24~SEG31 enable.
VLCD	P	LCD power supply
V1, V2	P	LVD bias voltage In LCD R type mode: Connect the external resistors to adjust VLCD voltage.
C1, C2	I	Connect the capacitors for internal VLCD charge-pump.
RST/VPP/P0.2	I, P	P0.2: Input only pin (Schmitt trigger) if disable external reset function. **P0.2 without build-in pull-up resister** Built- in wake-up function. RST: System reset input pin. Schmitt trigger structure, low active, normal operation must stay to “high”. VPP: OTP programming pin.
XIN	I	Oscillator input pin while external oscillator enable (crystal and RC).
XOUT/P0.3	I/O	Port 0.3 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resistor. Built- in wake-up function. XOUT: Oscillator output pin while external crystal enable.
LXIN	I	External low oscillator input pin (32768Hz crystal or RC).
LXOUT	O	External low oscillator output pin.
P0.0/INT0	I/O	Port 0.0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Built- in wake-up function. INT0 trigger pin (Schmitt trigger). TC0 event counter clock input pin.
P0.1/T1IN	I/O	Port 0.1 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Built- in wake-up function. T1 event counter clock input pin.
P1 [6:0]	I/O	Port 1 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Built- in wake-up function.
P3 [7:0]	I/O	Port 3 bi-direction pin in P3SEG=1. Schmitt trigger structure as input mode. Built-in pull-up resisters. After power on or reset,P3 default state as input LOW.
P5.0/ RFC0	I/O	Port 5.0 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. RFC0 is RFC channel 0 input pin.
P5.1/RFC1	I/O	Port 5.1 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. RFC1 is RFC channel 1 input pin.
P5.2/RFC2	I/O	Port 5.2 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. RFC2 is RFC channel 2 input pin.
P5.3/RFCI	I/O	Port 5.3 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. RFCI is RFC feedback input pin.
P5.4/BZ0/PWM0	I/O	Port 5.4 bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. TC1 ÷ 2 signal output pin for buzzer or PWM output pin.
P5.5/RFCOUT	I/O	Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. RFCOUT is RFC output pin.
COM0~COM3	O	LCD driver’s COM pin
SEG0~SEG31	O	LCD driver’s SEG pin

1.5 PIN CIRCUIT DIAGRAMS

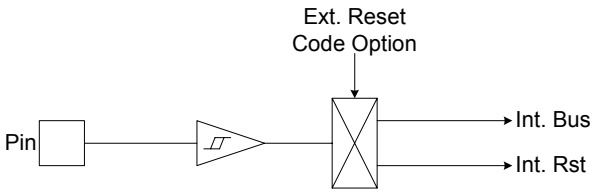
Port 0.3 structure:



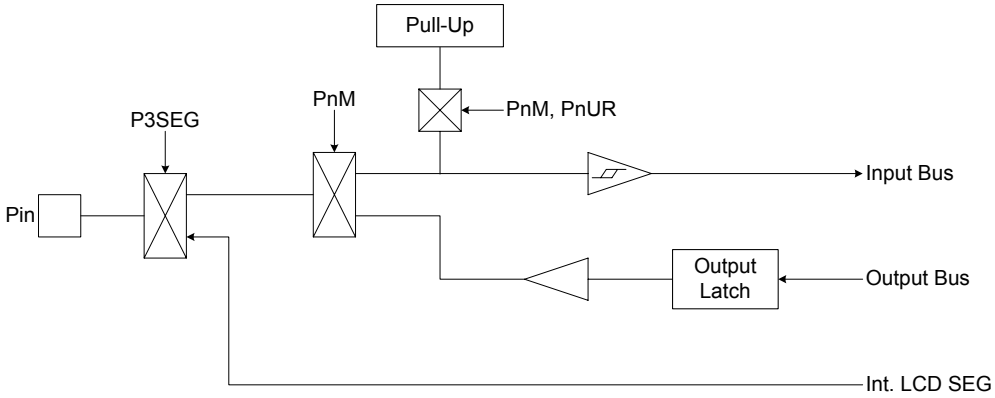
Port 0, 1, Port5.4 structure:



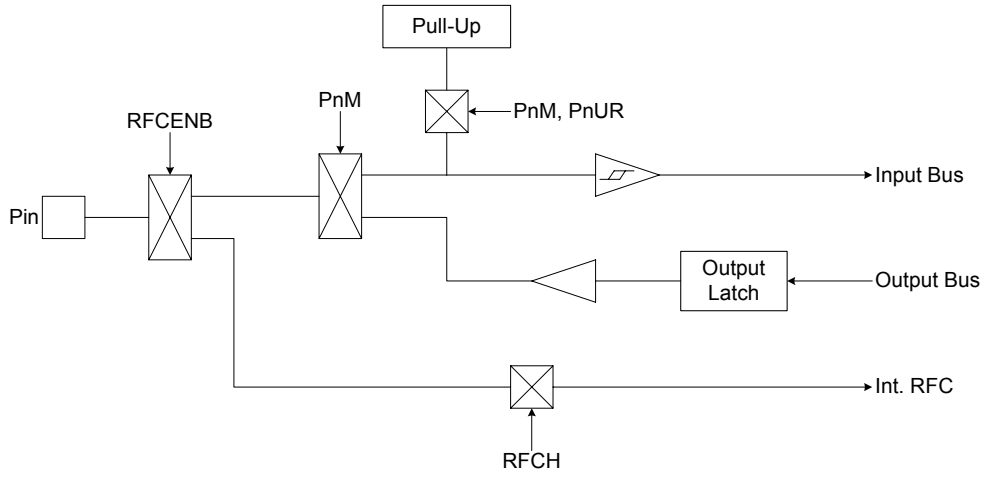
Port 0.2 structure:



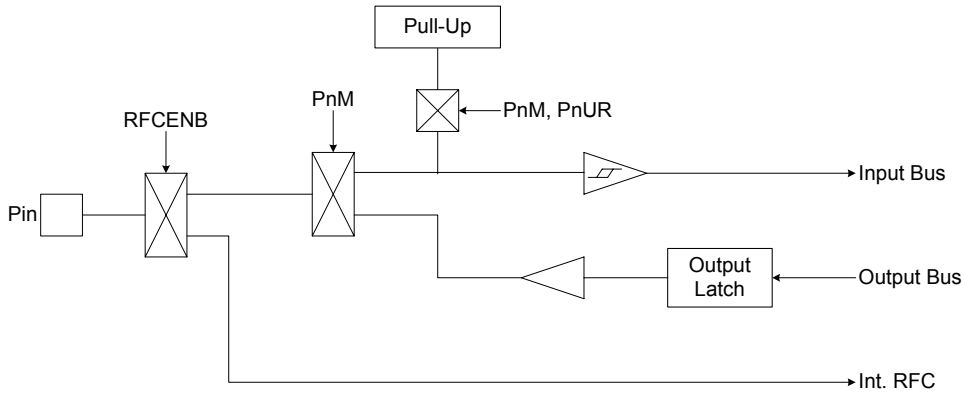
Port 3 structure:



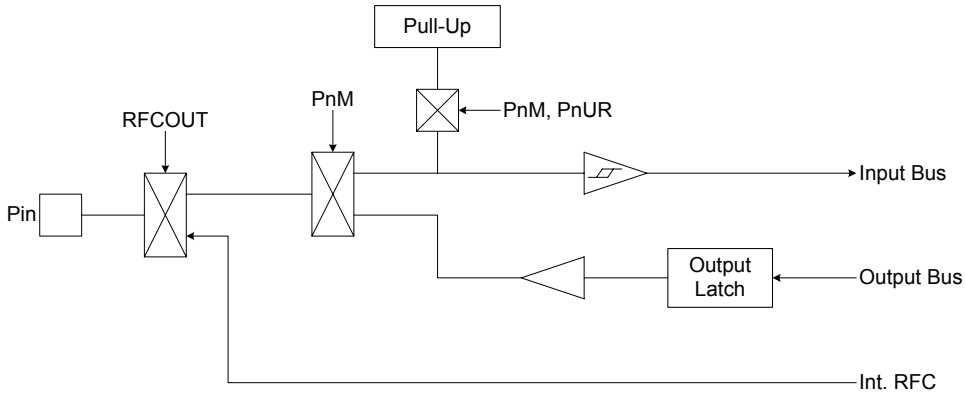
Port 5.0, P5.1, P5.2 structure:



Port 5.3 structure:



Port 5.5 structure:

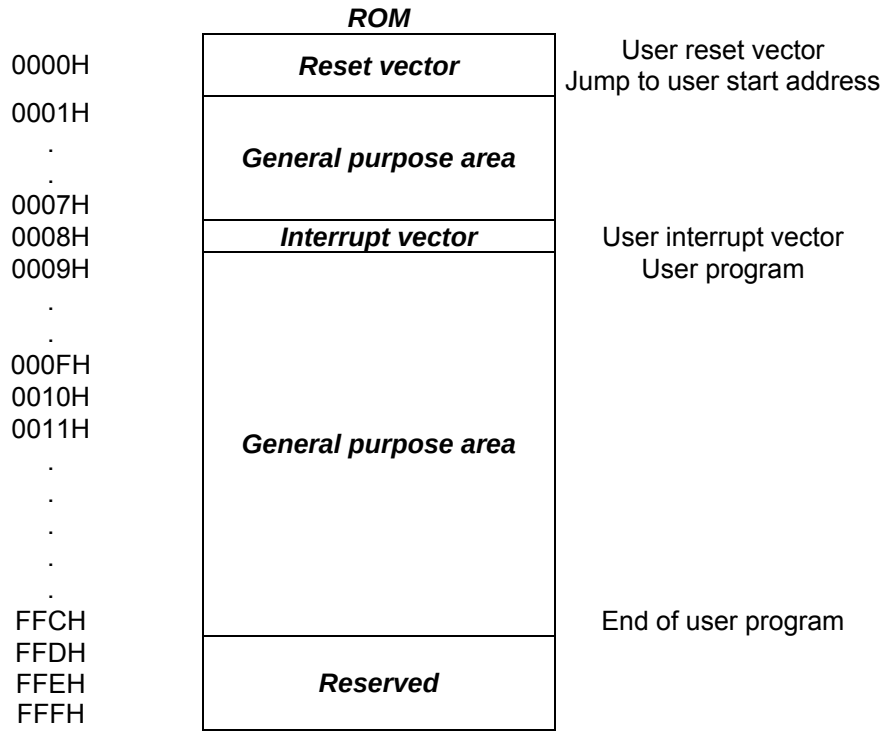


2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ ***4K words ROM***



2.1.1.1 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
```

2.1.1.2 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.***

2.1.1.3 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 take care such situation to avoid look-up table errors. If Z register overflows, 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
                …
```

2.1.1.4 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
        ADD     PCL, A
        ENDM
```

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

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

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

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

➢ **Example: "@JMP_A" operation.**

; Before compiling program.

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

; After compiling program.

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

2.1.1.5 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
```

2.1.2 CODE OPTION TABLE

Code Option	Content	Function Description
High_Clk	RC	Low cost RC for external high clock oscillator and XOUT becomes to P1.2 bit-direction I/O pin.
	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.
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.
Fcpu	Fhosc/1	Instruction cycle is oscillator clock. Notice: In Fosc/1, Noise Filter must be disabled.
	Fhosc/2	Instruction cycle is 2 oscillator clocks. Notice: In Fosc/2, Noise Filter must be disabled.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
Reset_Pin	Reset	Enable External reset pin.
	P02	Enable P0.2 input only without pull-up resister.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
Noise_Filter	Enable	Enable Noise Filter and the Fcpu is Fosc/4~Fosc/8.
	Disable	Disable Noise Filter and the Fcpu is Fosc/1~Fosc/8.
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.

✸ ***Note:***

1. ***In high noisy environment, enable "Noise Filter" and set Watch_Dog as "Always_On" is strongly recommended. Enable "Noise_Filter" will limit the Fcpu = Fosc/4 ~ Fosc/128.***
2. ***In high noisy environment, enable "Noise Filter" and set Watch_Dog as "Always_On" is strongly recommended.***
3. ***Fcpu code option is only available for High Clock. Fcpu of slow mode is Flosc/4.***

2.1.3 DATA MEMORY (RAM)

☞ ***128 X 8-bit RAM***

	Address	RAM location	
BANK 0	000h “ “ “ “ “ 07Fh	***General purpose area***	
	080h “ “ “ “ “ 0FFh	***System register***	080h~0FFh of Bank 0 store system registers (128 bytes).
		End of bank 0 area	
BANK 15	000h “ “ “ 01Fh	***LCD RAM area***	0x00~0x1F of Bank 15 (32 byte) is LCD RAM registers.

2.1.4 SYSTEM REGISTER

2.1.4.1 SYSTEM REGISTER TABLE

Bank 0:

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

2.1.4.2 SYSTEM REGISTER DESCRIPTION

R = Working register and ROM look-up data buffer.
PFLAG = ROM page and special flag register.
RBANK = RAM bank control register.
T1M = T1 mode register.
RFCM = RFC mode register.
PnM = Port n input/output mode register.
INTRQ = Interrupt request register.
OSCM = Oscillator mode register.
TC0R = TC0 auto-reload data buffer.
Pn = Port n data buffer.
TC0M = TC0 mode register.
T0C = T0 counting register.
PnUR = Port n pull-up resister control register.
STK0~STK7 = Stack 0 ~ stack 7 buffer.

Y, Z = Working, @YZ and ROM addressing register.
H, L = Working, @HL addressing register.
VLCD = LCD voltage control register.
T1CH, T1CL = T1 counting register.
PEDGE = P0.0 edge direction register.
P1W = P1 wakeup control register.
INTEN = Interrupt enable register.
LCDM = LCD mode register.
WDTR = Watchdog timer clear register.
PCH, PCL = Program counter.
T0M = TC0/TC1 speed-up and TC0 wake-up function register.
TC0C = TC0 counting register.
STKP = Stack pointer buffer.
@HL = RAM HL indirect addressing index pointer.
@YZ = RAM YZ indirect addressing index pointer.

2.1.4.3 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					RBNKS3	RBNKS2	RBNKS1	RBNKS0	R/W	RBANK
089H			CPCK1	CPCK0	VLCD2	VLCD1	VLCD0	VLCDCP	R/W	VLCD
0A0H	T1ENB	T1RATE2	T1RATE1	T1RATE0	T1CKS		T1EN	T1IRQ	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	RFCOUT					RFCH1	RFCH0	RFCENB	R/W	RFCM
0B8H					P03M		P01M	P00M	R/W	P0M
0BFH				P00G1	P00G0		T1G1	T1G0	R/W	PEDGE
0C0H		P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H		P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C3H	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M	R/W	P3M
0C5H			P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H			TC0IRQ	T0IRQ				P00IRQ	R/W	INTRQ
0C9H			TC0IEN	T0IEN				P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CBH					RCLK	P3SEG	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					P03	P02	P01	P00	R/W	P0
0D1H		P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D3H	P37	P36	P35	P34	P33	P32	P31	P30	R/W	P3
0D5H			P55	P54	P53	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0RATE2	T0RATE1	T0RATE0		TC0X8		T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H					P03R		P01R	P00R	W	P0UR
0E1H		P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E3H	P37R	P36R	P35R	P34R	P33R	P32R	P31R	P30R	W	P3UR
0E5H			P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H					S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H					S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H					S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H					S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H					S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH					S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH					S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH					S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✷ ***Note:***

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

2.1.5 LCD RAM

LCD RAM is mapping to every dots of COM & SEG combination in Bank 15. Only low nibble of one LVD RAM register is useful for COM0~COM3.

2.1.5.1 LCD RAM TABLE

Bank 15:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	LCD RAM AREA															
1																
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

2.1.5.2 BIT DEFINITION of LCD RAM

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	-	-	-	-

2.1.5.3 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
```

2.1.5.4 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.***

2.1.5.5 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 0x00 to 0xFF, the PC will add 2 steps to skip next instruction.

DECS instruction:

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

DECMS instruction:

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

☞ **MULTI-ADDRESS JUMPING**

Users can jump around the multi-address by either JMP instruction or ADD M, A instruction (M = PCL) to activate multi-address jumping function. Program Counter 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
                …
                …
```

2.1.5.6 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	X	X	X	X	X	X	X	X

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	X	X	X	X	X	X	X	X

➢ **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
        …
        …
```

2.1.5.7 Y, Z REGISTERS

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

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

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

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

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

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

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

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

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

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

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

2.1.5.8 R REGISTERS

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

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

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

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

2.2 ADDRESSING MODE

2.2.1 IMMEDIATE ADDRESSING MODE

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

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

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

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

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

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

2.2.2 DIRECTLY ADDRESSING MODE

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

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

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

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

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

2.2.3 INDIRECTLY ADDRESSING MODE

The indirectly addressing mode is to access the memory by the data pointer registers (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.
```

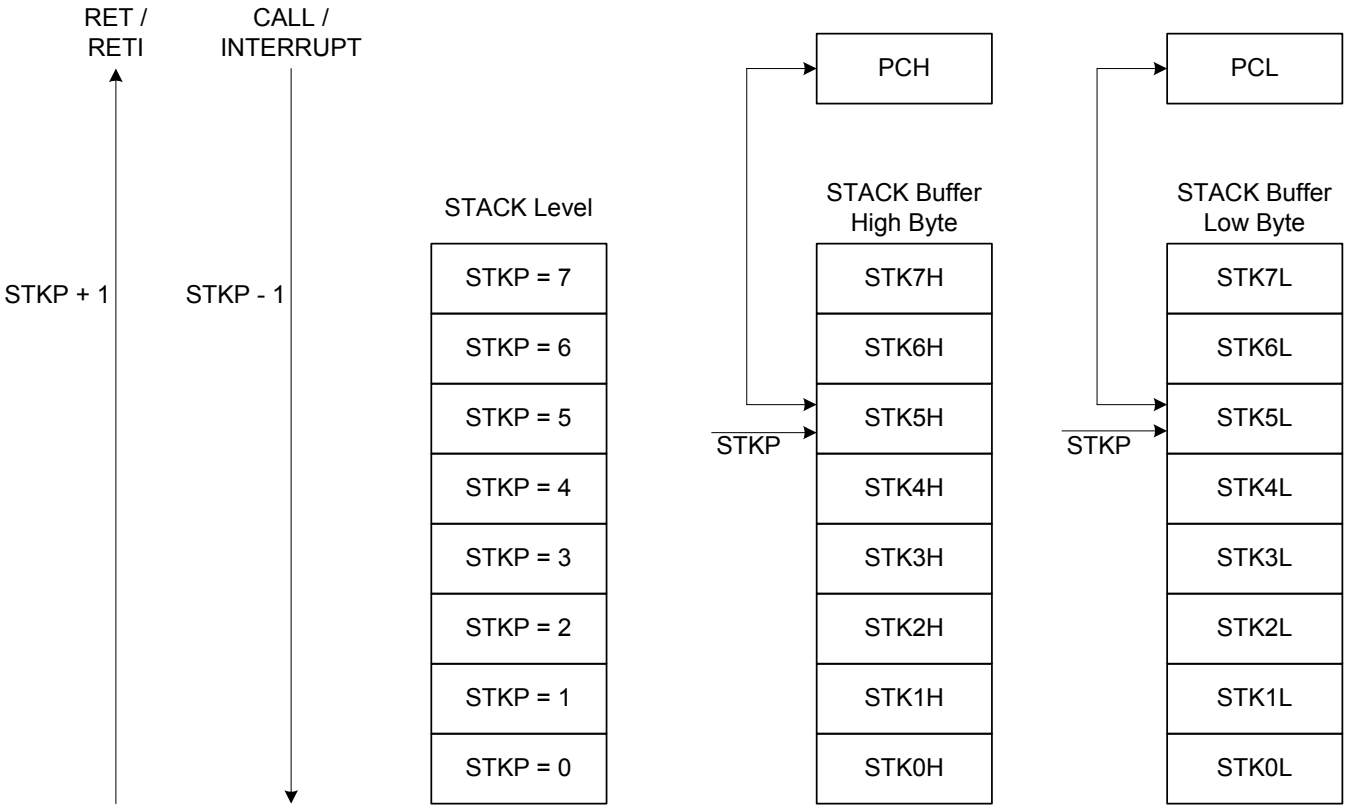
- **Example: Indirectly addressing mode with @YZ register**

```
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.
```

2.3 STACK OPERATION

2.3.1 OVERVIEW

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



2.3.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 12-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

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

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

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

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

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

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

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

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	-	-	SnPC11	SnPC10	SnPC9	SnPC8
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	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)

2.3.3 STACK OPERATION EXAMPLE

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

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

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

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

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

- Power on reset
- Watchdog reset
- Brown out reset
- External reset (only supports external reset pin enable situation)

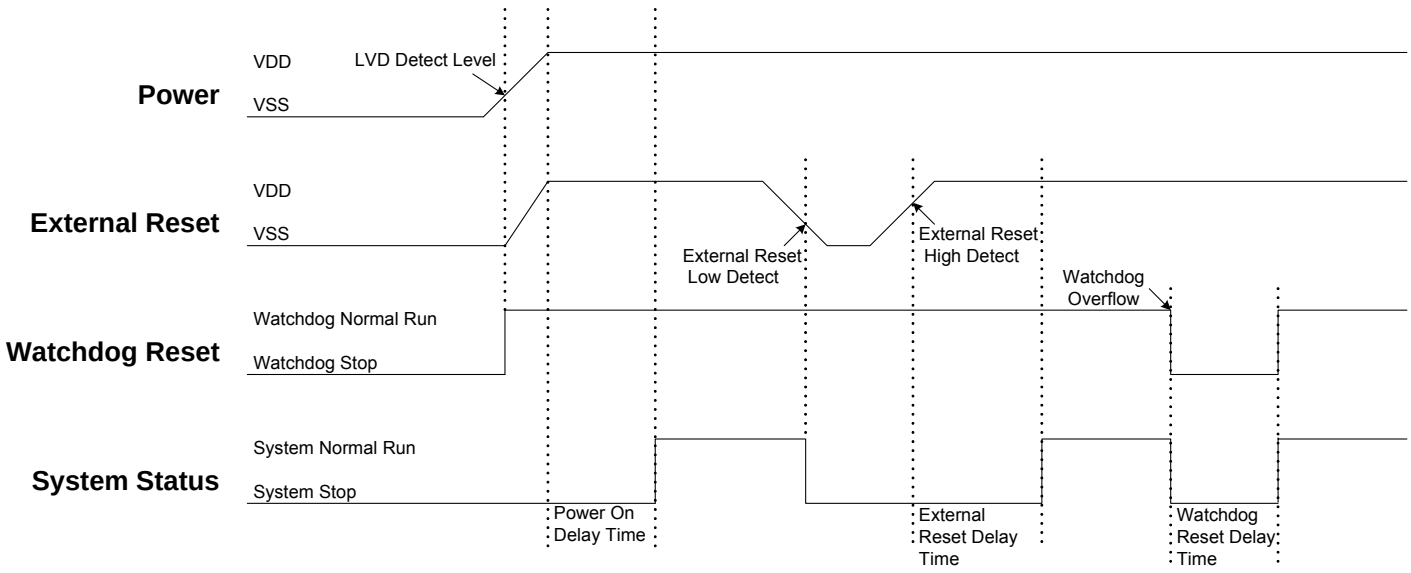
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

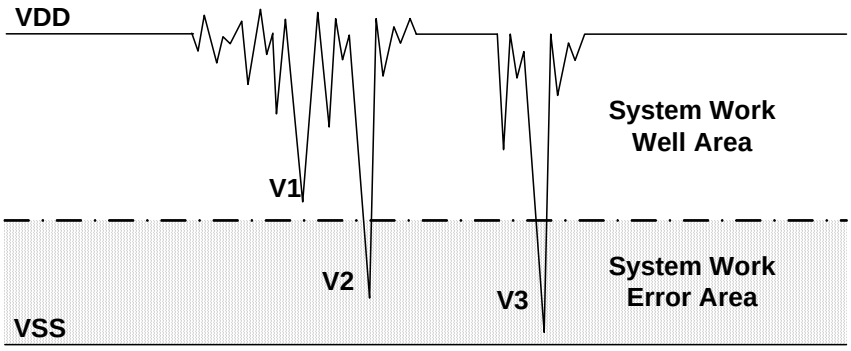
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the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

3.4.1 BROWN OUT DESCRIPTION

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



Brown Out Reset Diagram

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

DC application:

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

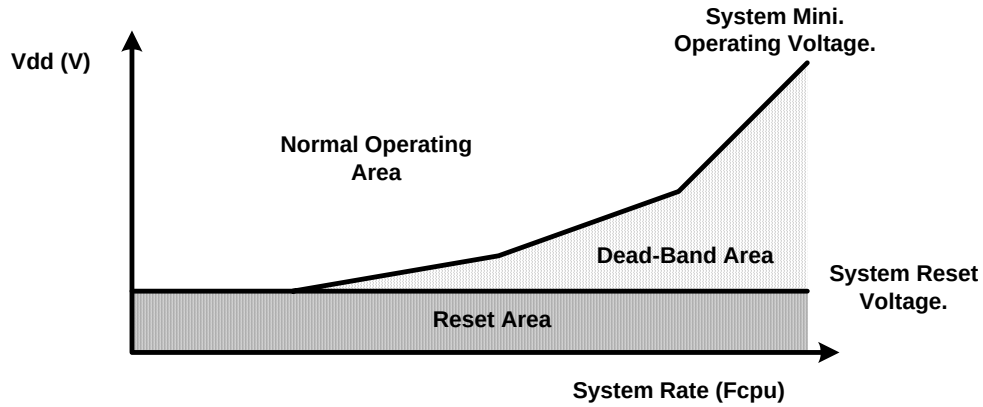
AC application:

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

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

3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION

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



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

3.4.3 BROWN OUT RESET IMPROVEMENT

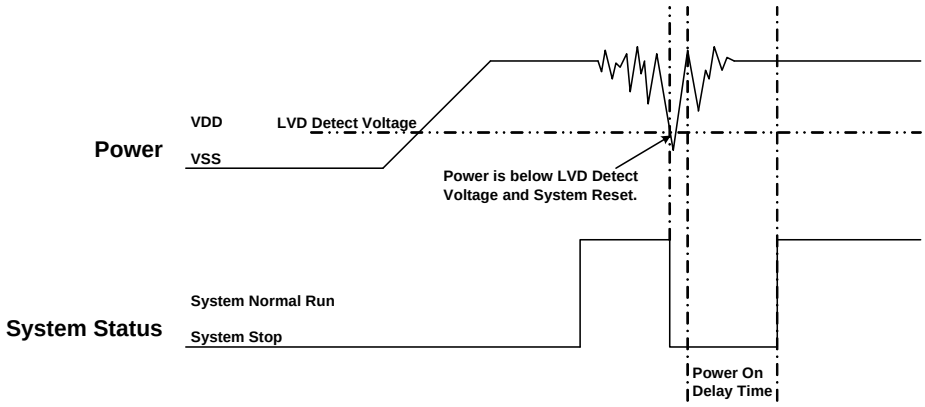
How to improve the brown reset condition? There are some methods to improve brown out reset as following.

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

✷ ***Note:***

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

LVD reset:



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

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
2.0V Reset	Available	Available	Available
2.4V Flag	-	Available	-
2.4V Reset	-	-	Available
3.6V Flag	-	-	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
LVD2_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"

✷ ***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.***

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.

3.5 EXTERNAL RESET

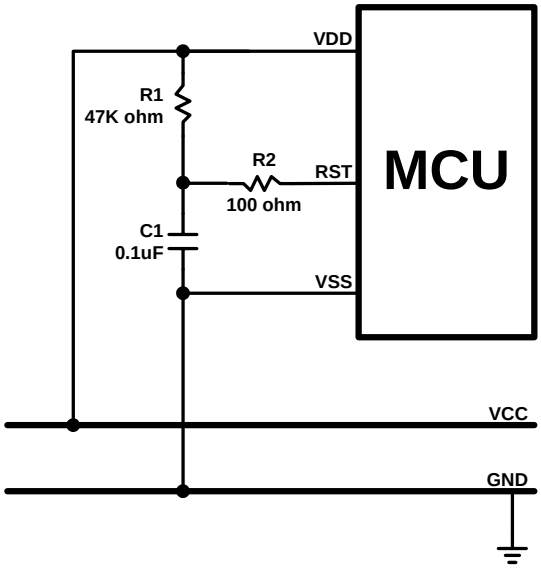
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the 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…

3.6 EXTERNAL RESET CIRCUIT

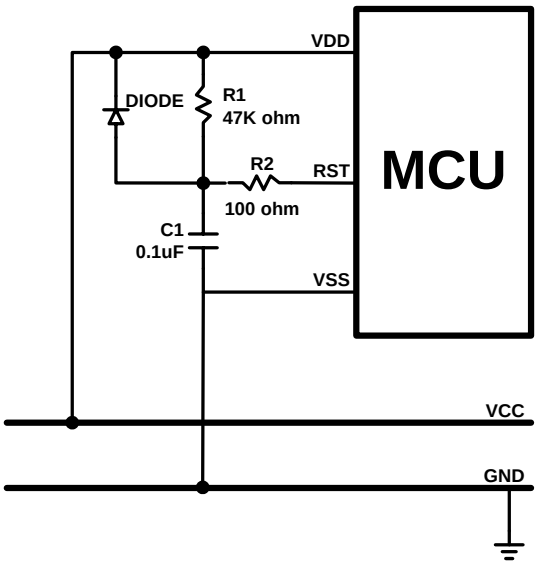
3.6.1 Simply RC Reset Circuit



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

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

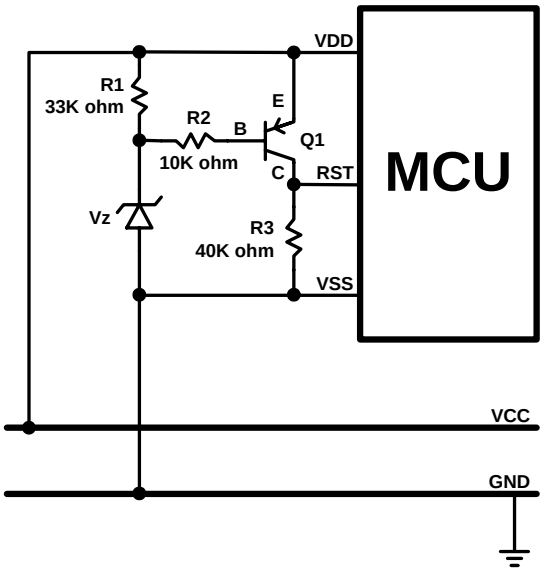
3.6.2 Diode & RC Reset Circuit



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

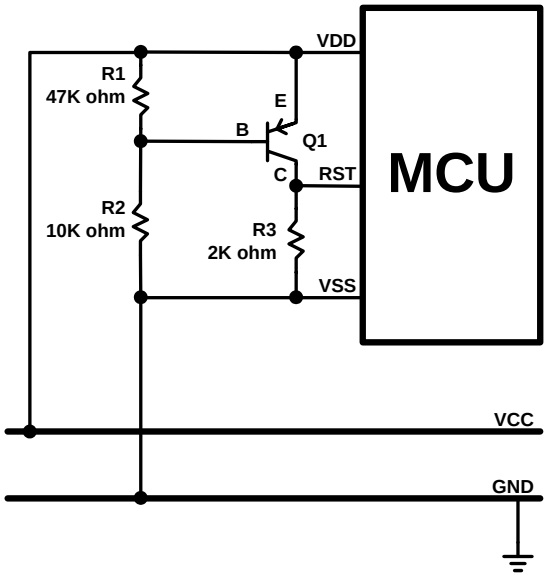
✸ ***Note: The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS).***

3.6.3 Zener Diode Reset Circuit



The zener diode reset circuit is a simple low voltage detector and can **improve brown out reset condition completely**. Use zener voltage to be the active level. When VDD voltage level is above "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.

3.6.4 Voltage Bias Reset Circuit

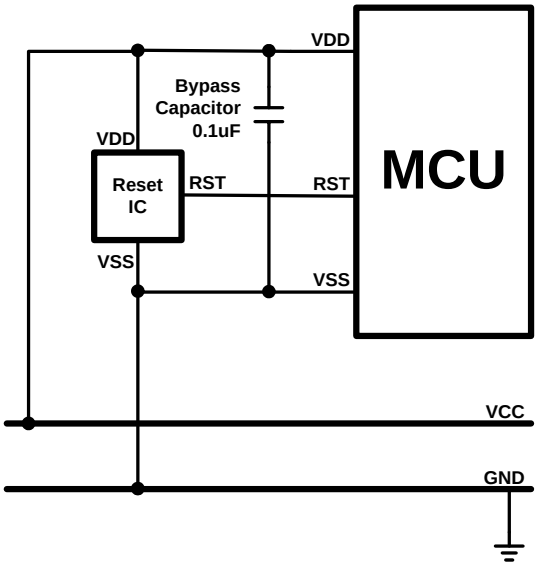


The voltage bias reset circuit is a low cost voltage detector and can **improve brown out reset condition completely**. The operating voltage is not accurate as zener diode reset circuit. Use R1, R2 bias voltage to be the active level. When VDD voltage level is above or equal to "0.7V x (R1 + R2) / R1", 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.***

3.6.5 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. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit. The low-speed clock is generated from external low-speed 32768Hz oscillator circuit.

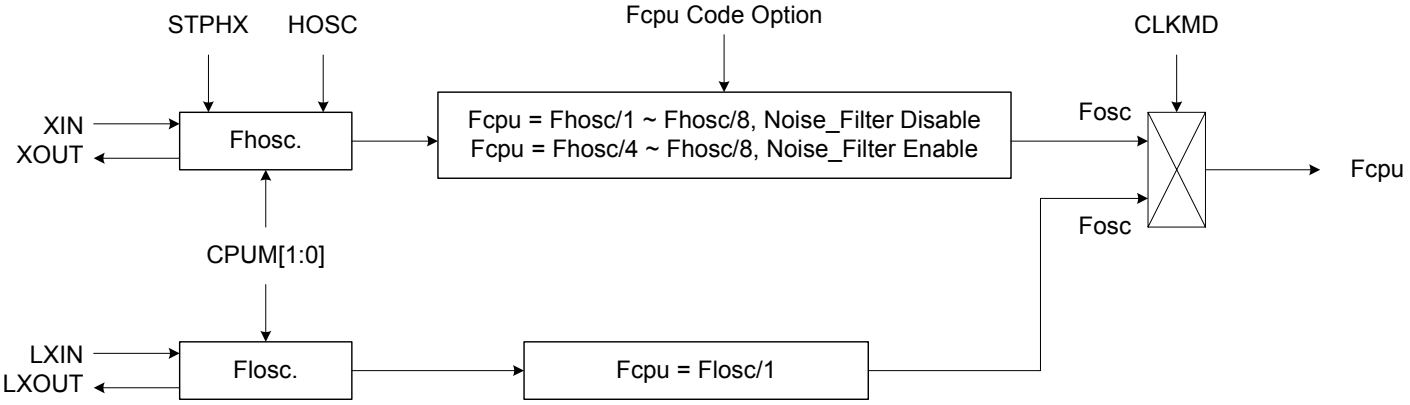
Both the high-speed clock and the low-speed clock can be system clock (Fosc). The system clock in slow mode is fixed Flosc (32768Hz) to be the instruction cycle (Fcpu).

☞ **Normal Mode (High Clock):** **Fcpu = Fhosc / N**, N = 1 ~ 8, Select N by Fcpu code option.

☞ **Slow Mode (Low Clock):** **Fcpu = Flosc/1.**

SONIX provides a **"Noise Filter"** controlled by code option. In high noisy situation, the noise filter can isolate noise outside and protect system works well. The minimum Fcpu of high clock is limited at **Fhosc/4** when noise filter enable.

4.2 CLOCK BLOCK DIAGRAM



- HOSC: High_Clk code option.
- Fhosc: External high-speed clock.
- Flosc: External 32768Hz low-speed clock.
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.3 OSCM REGISTER

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

0CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	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:** External high-speed oscillator control bit.
0 = External high-speed oscillator free run.
1 = External high-speed oscillator free run stop. Internal low-speed RC oscillator is still running.

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

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

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

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

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

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

4.4 SYSTEM HIGH CLOCK

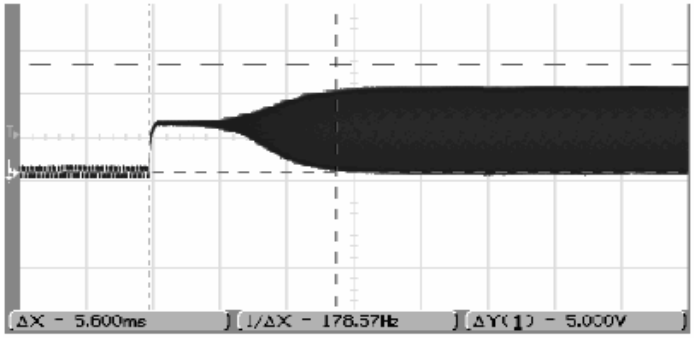
The system high clock is from external oscillator. The high clock type is controlled by “High_Clk” code option.

High_Clk Code Option	Description
RC	The high clock is external RC type oscillator. XOUT pin is general purpose I/O pin.
12M	The high clock is external high speed oscillator. The typical frequency is 12MHz.
4M	The high clock is external oscillator. The typical frequency is 4MHz.

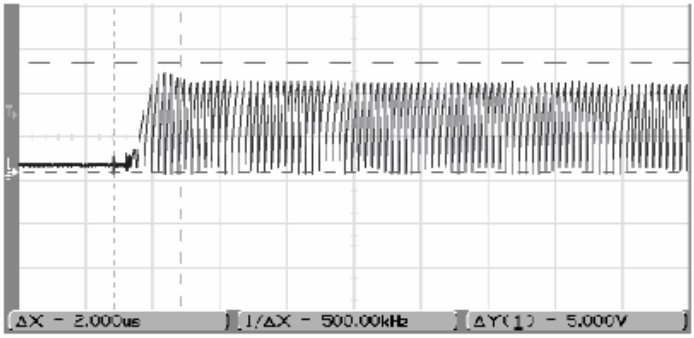
4.4.1 EXTERNAL HIGH CLOCK

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

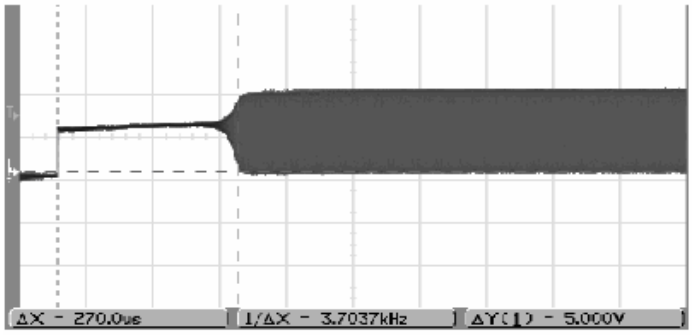
4MHz Crystal



RC

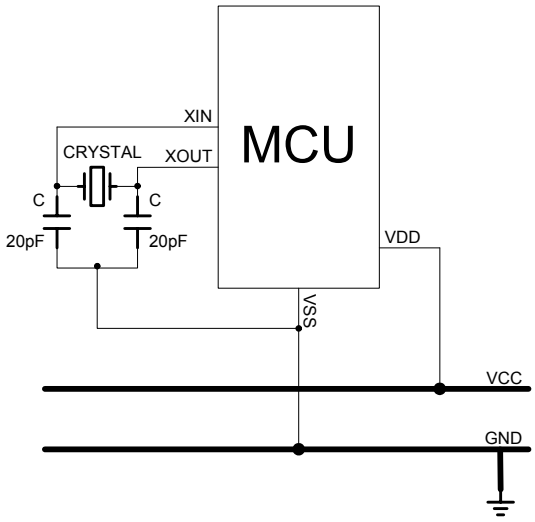


4MHz Ceramic



4.4.1.1 CRYSTAL/CERAMIC

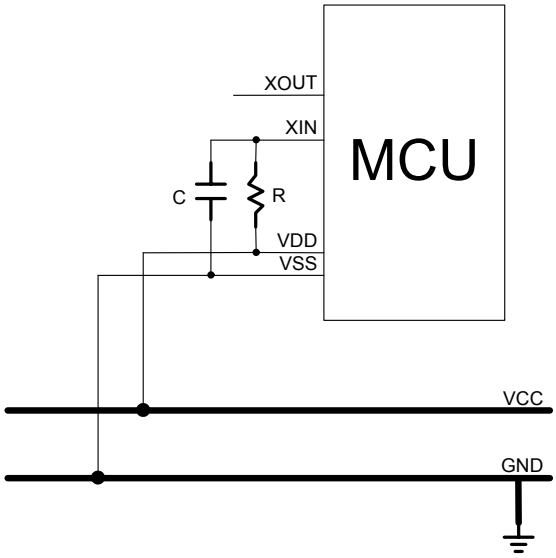
Crystal/Ceramic devices are driven by XIN, XOUT pins. For high/normal/low frequency, the driving currents are different. High_Clk code option supports different frequencies. 12M option is for high speed (ex. 12MHz). 4M option is for normal speed (ex. 4MHz).



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

4.4.1.2 RC

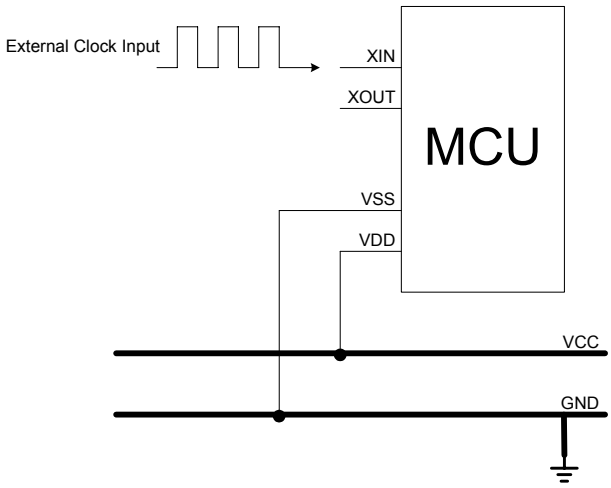
Selecting RC oscillator is by RC option of High_Clk code option. RC type oscillator's frequency is up to 10MHz. Using "R" value is to change frequency. 50P~100P is good value for "C". XOUT pin is general purpose I/O pin.



✷ ***Note: Connect the R and C as near as possible to the VDD pin of micro-controller.***

4.4.1.3 EXTERNAL CLOCK SIGNAL

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



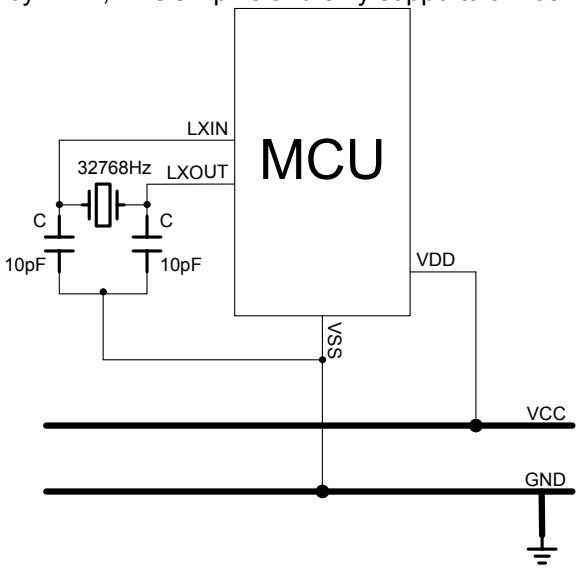
✸ ***Note: The GND of external oscillator circuit must be as near as possible to VSS pin of micro-controller.***

4.5 SYSTEM LOW CLOCK

System low clock is from external low-speed oscillator. External low clock includes three modules (Crystal/Ceramic, RC and external clock signal). The low clock oscillator module is controlled by RCLK bit of LCDM register. The external low clock supports system low clock source and LCD scanning clock source.

4.5.1 CRYSTAL/CERAMIC

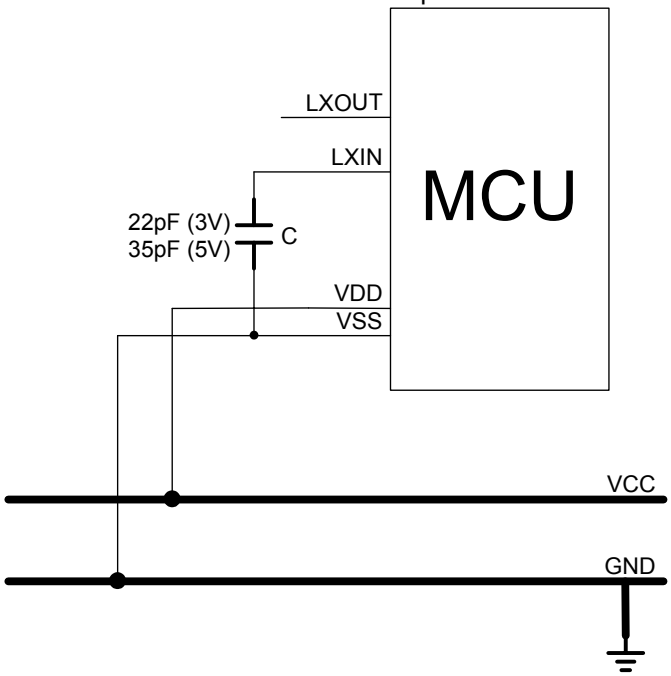
Crystal/Ceramic devices are driven by LXIN, LXOUT pins and only supports 32768Hz oscillator.



✸ ***Note: Connect the Crystal/Ceramic and C as near as possible to the LXIN/LXOUT/VSS pins of micro-controller. The capacitor between LXIN/LXOUT and VSS must be 10pF.***

4.5.2 RC

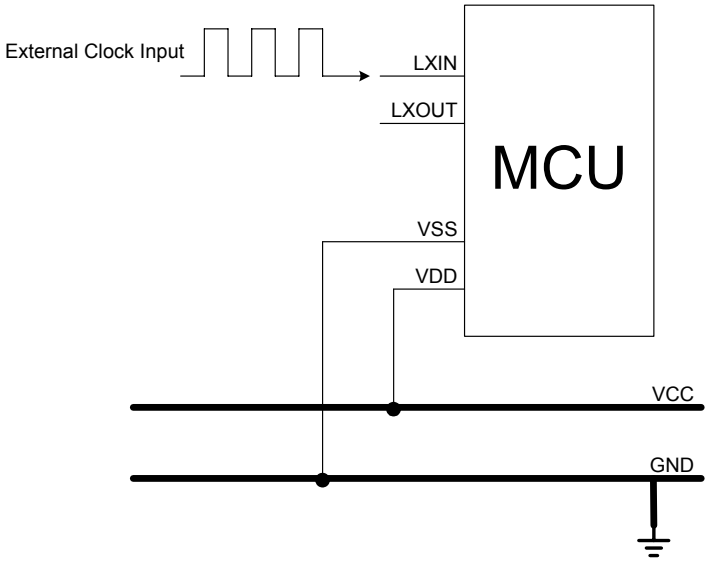
Selecting RC oscillator is by RCLK bit of LCDM register. RC type oscillator's frequency is 32768Hz. The external 32K RC type circuit only needs a capacitor, and don't connect a resistor between LXIN and VDD. **22pF @3V** and **35pF @5V** is a good value for "C" connected from LXIN to VSS. LXOUT pin is useless in external low RC mode.



➢ ***Note: Connect the C as near as possible to the VSS pin of micro-controller. The frequency of external low RC is decided by the capacitor value. Adjust capacitor value to about 32KHz frequency.***

4.5.3 EXTERNAL CLOCK SIGNAL

Selecting external clock signal input to be system clock is by RCLK bit of LCDM register. The external clock signal is input from LXIN pin. LXOUT pin is useless.



➢ ***he GND of external oscillator circuit must be as near as possible to VSS pin of micro-controller.***

4.5.4 SYSTEM CLOCK MEASUREMENT

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

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

```
        B0BSET    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
```

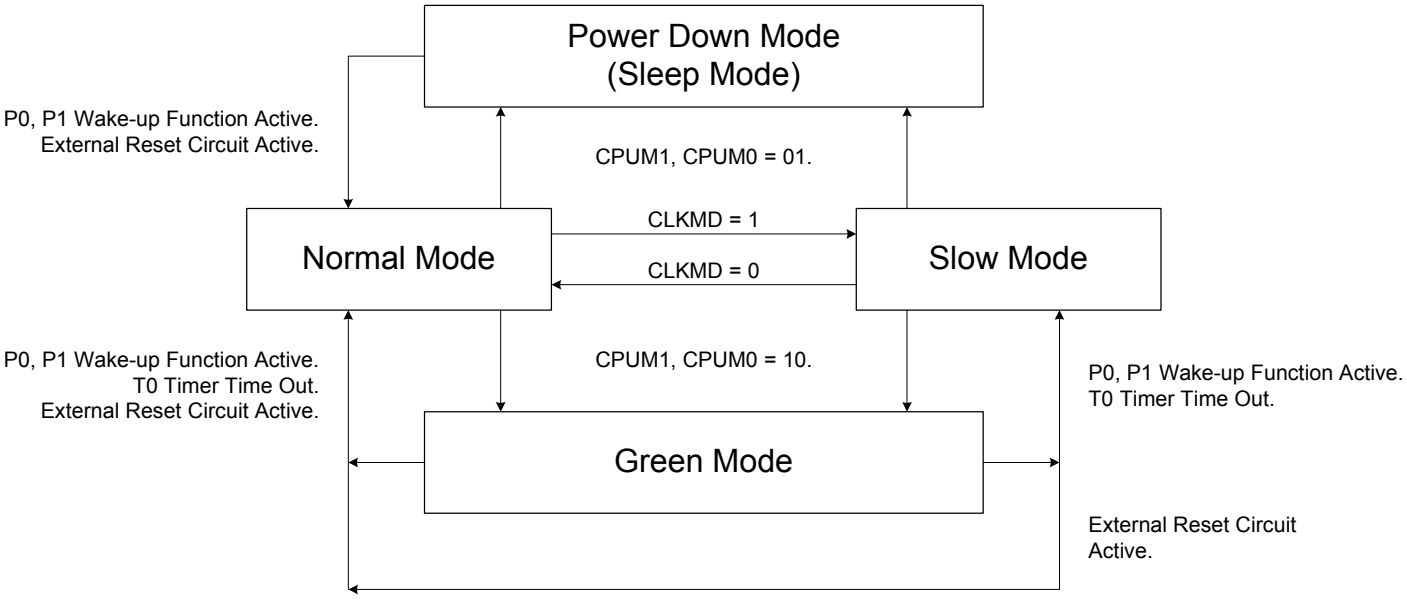
➢ ***Note: Do not measure the RC frequency directly from XIN; the probe impendence will affect the RC frequency.***

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

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



System Mode Switching Diagram

Operating mode description

MODE	NORMAL	SLOW	GREEN	POWER DOWN (SLEEP)	REMARK
EHOSC	Running	By STPHX	By STPHX	Stop	
ELRC	Running	Running	Running	Stop	
CPU instruction	Executing	Executing	Stop	Stop	
T0 timer	*Active	*Active	*Active	Inactive	* Active if T0ENB=1
TC0 timer	*Active	*Active	*Active	Inactive	* Active if TC0ENB=1
T1 timer	*Active	*Active	*Active	Inactive	* Active if T1ENB=1
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	Refer to code option description
Internal interrupt	All active	All active	T0, TC0, T1	All inactive	
External interrupt	All active	All active	All active	All inactive	
Wakeup source	-	-	P0, P1, T0 Reset	P0, P1, Reset	

EHOSC: External high clock.
ELRC: External low clock (32768Hz).

5.2 SYSTEM MODE SWITCHING

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

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

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

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

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

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

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

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

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

```
        B0BCLR      FSTPHX          ; Turn on the external high-speed oscillator.

        B0MOV       Z, #54          ; If VDD = 5V, internal RC=32KHz (typical) will delay
@@:     DECMS       Z               ; 0.125ms X 162 = 20.25ms for external clock stable
        JMP         @B

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

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

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

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

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

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

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

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

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

5.3 WAKEUP

5.3.1 OVERVIEW

Under power down mode (sleep mode) 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).

5.3.2 WAKEUP TIME

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

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

The value of the wakeup time is as the following.

The Wakeup time = 1/Fosc * 2048 (sec) + high clock start-up time

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

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

The wakeup time = 1/Fosc * 2048 = 0.512 ms (Fosc = 4MHz)
The total wakeup time = 0.512 ms + oscillator start-up time

✸ ***Note: The wakeup function of P0 can't be disabled. Make sure enable the pull-up resistor of P0 or use external pull-up resistors before enter sleep mode.***

5.3.3 P1W WAKEUP CONTROL REGISTER

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

0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	-	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 I/O PORT

6.1 I/O PORT MODE

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

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	-	-	-	-	P03M	-	P01M	P00M
Read/Write	-	-	-	-	R/W	-	R/W	R/W
After reset	-	-	-	-	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

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	-	-	P55M	P54M	P53M	P52M	P51M	P50M
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

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

✸ ***Note:***
1. ***Users can program them by bit control instructions (B0BSET, B0BCLR).***
2. ***P0.2 input only pin, and the P0M.2 keeps "1".***
3. ***P3 is shared with SEG24~SEG31 controlled by P3SEG bit of LCDM register.***
4. ***After power on or reset, P3 default state as input LOW.***

➢ **Example: I/O mode selecting**

```
CLR     P0M         ; Set all ports to be input mode.
CLR     P1M
CLR     P5M

MOV     A, #0FFH    ; Set all ports to be output mode.
B0MOV   P0M, A
B0MOV   P1M,A
B0MOV   P5M, A

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

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

➢ **Example: P3 I/O mode selecting**

```
B0BSET      FP3SEG      ; Enable P3 I/O function.
CLR         P3M         ; Set P3 port to be input mode.
MOV         A, #0FFH    ; Set P3 port to be output mode.
B0MOV       P3M, A
B0BCLR      P3M.0       ; Set P3.0 to be input mode.
B0BSET      P3M.0       ; Set P3.0 to be output mode.
```

6.2 I/O PULL UP REGISTER

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	-	-	-	P03R	-	P01R	P00R
Read/Write	-	-	-	-	W	-	W	W
After reset	-	-	-	-	0	-	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	-	P16R	P15R	P14R	P13R	P12R	P11R	P10R
Read/Write	-	W	W	W	W	W	W	W
After reset	-	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	-	-	P55R	P54R	P53R	P52R	P51R	P50R
Read/Write	-	-	W	W	W	W	W	W
After reset	-	-	0	0	0	0	0	0

✷ ***Note: P0.2 is input only pin and without pull-up resistor. The P0UR.2 keeps "1".***

➢ **Example: I/O Pull up Register**

```
MOV     A, #0FFH        ; Enable Port0, 1, 3, 5 Pull-up register,
B0MOV   P0UR, A         ;
B0MOV   P1UR, A
B0MOV   P3UR, A
B0MOV   P5UR, A
```

6.3 I/O PORT DATA REGISTER

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

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	-	-	P55	P54	P53	P52	P51	P50
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

✷ ***Note: P0.2 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
B0MOV   A, P3       ; Read data from Port 3
B0MOV   A, P5       ; Read data from Port 5
```

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

```
MOV     A, #0FFH    ; Write data FFH to all Port.
B0MOV   P0, A
B0MOV   P1, A
B0MOV   P3, A
B0MOV   P5, A
```

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

```
B0BSET  P1.0        ; Set P1.0 and P5.3 to be "1".
B0BSET  P5.3

B0BCLR  P1.0        ; Set P1.0 and P5.3 to be "0".
B0BCLR  P5.3
```

6.4 PORT 5 RFC SHARE PIN

Port 5 is shared with RFC input and output pins. P5.0~P5.2 pins are RFC channel 0~2 input pins. P5.3 is RFC feedback clock input. P5.5 is RFC output pins. If RFC enable (RFCENB=1), P5.0~P5.3 are switched to RFC input mode and digital I/O functions are disabled. P5.5 RFC output function is controlled by RFCOUT bit of RFCM register.

✷ ***Note:***

1. ***If RFCENB=1, P5.0~P5.3 must be set to input mode without pull-up resistors by program first.***
2. ***If RFCOUT=1, P5.5 must be set to input mode without pull-up resistor by program first.***

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RFCM	RFCOUT	-	-	-	-	RFCH1	RFCH2	RFCENB
Read/Write	R/W	-	-	-	-	R/W	R/W	R/W
After reset	0	-	-	-	-	0	0	0

Bit7 **RFCOUT:** RFCOUT control bit.
0 = RFC output disable. P5.5 is general purpose I/O mode.
1 = RFC output enable. P5.5 is RFC output pin.

Bit0 **RFCENB:** RFC mode control bit.
0 = RFC disable. P5.0~P5.3 are general purpose I/O mode.
1 = RFC enable. P5.0~P5.3 are RFC input mode.

➢ **Example: Set P5.0~P5.3 to general purpose I/O mode from RFC mode.**

```
B0BCLR      FRFCENB     ; Disable RFC mode, and P5.0~P5.3 switch to I/O mode.
…                       ; P5.0~P5.3 I/O mode control.
```

➢ **Example: Set P5.0~P5.3 to RFC mode.**

```
CLR         P5UR        ; Disable P5 pull-up resistor.
MOV         A, #11110000B
AND         P5M, A      ; Set P5.0~P5.3 to input mode.

B0BSET      FRFCENB     ; Enable RFC mode and P5.0~P5.3 switch to RFC input
                        ; mode.
…
```

➢ **Example: Set P5.5 to general purpose I/O mode from RFC output mode.**

```
B0BCLR      FRFCOUT     ; Disable RFC output mode, and P5.5 switches to I/O mode.
…                       ; P5.5 I/O mode control.
```

➢ **Example: Set P5.5 to RFC output mode.**

```
CLR         P5UR        ; Disable P5 pull-up resistor.
MOV         A, #11101111B
AND         P5M, A      ; Set P5.5 to input mode.

B0BSET      FRFCOUT     ; Enable RFC output mode and P5.5 switch to RFC output
                        ; mode.
…
```

7 TIMERS

7.1 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 (16KHz @3V, 32KHz @5V).

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

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

✷ ***Note: If watchdog is "Always_On" mode, it keeps running event under power down mode or green mode.***

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
```

Watchdog timer application note is as following.

- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.

- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the best structure to enhance the watchdog timer function.

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

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

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

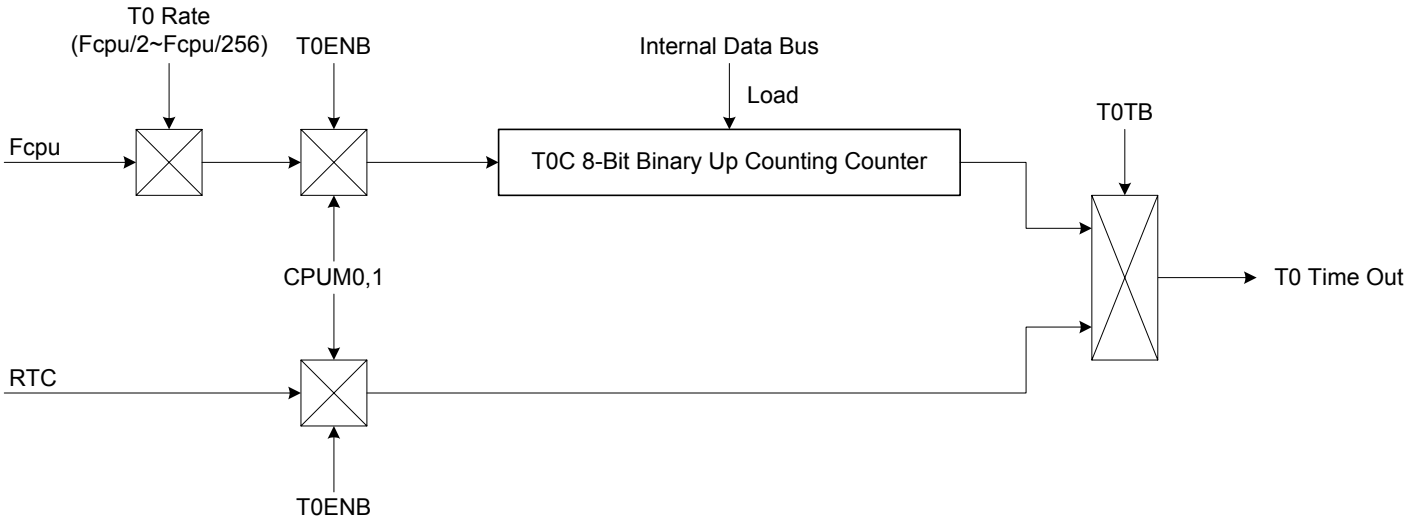
7.2 TIMER 0 (T0)

7.2.1 OVERVIEW

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

The main purposes of the T0 timer is as following.

- **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **RTC timer:** Generates interrupts at real time intervals based on the selected clock source. **RTC function is only available in T0TB=1.**
- **Green mode wakeup function:** T0 can be green mode wake-up time as T0ENB = 1. System will be wake-up by T0 time out.



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

7.2.2 T0M MODE REGISTER

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

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

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

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

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

7.2.3 T0C COUNTING REGISTER

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

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

The equation of T0C initial value is as following.

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

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

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

The basic timer table interval time of T0.

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

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

7.2.4 T0 TIMER OPERATION SEQUENCE

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

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

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

☞ **Set T0 timer rate.**

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

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

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

or

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

☞ **Set T0 interrupt interval time.**

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

☞ **Set T0 timer function mode.**

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

☞ **Enable T0 timer.**

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

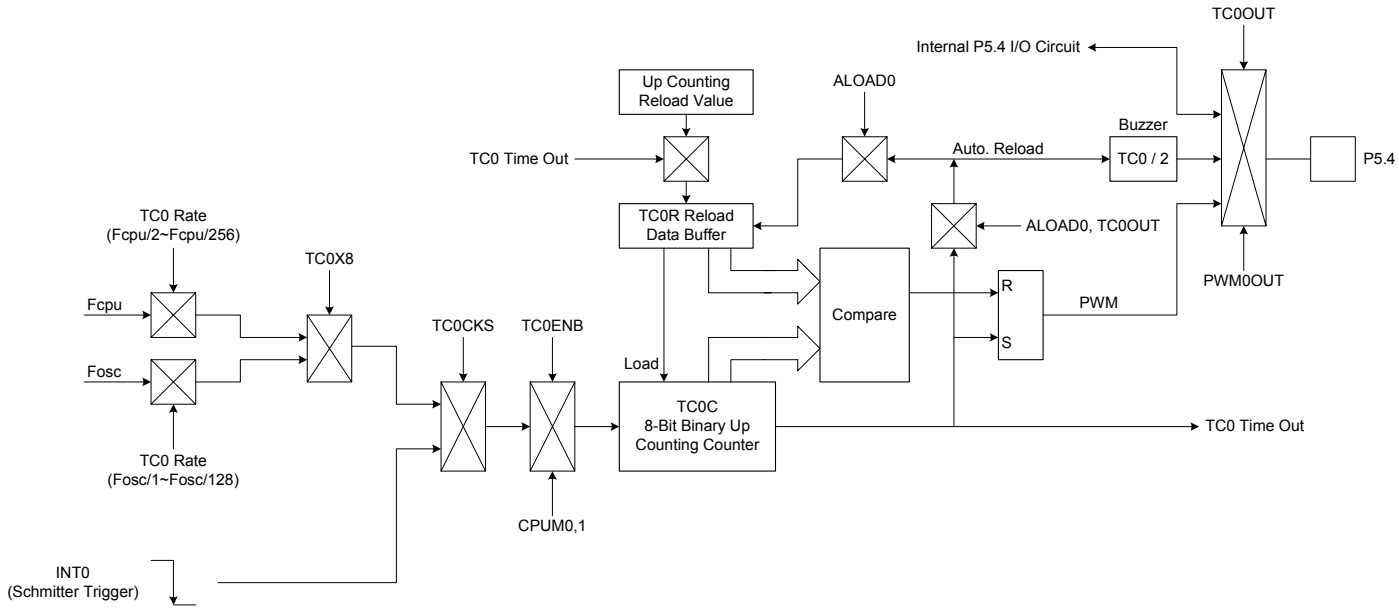
7.3 TIMER/COUNTER 0 (TC0)

7.3.1 OVERVIEW

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

The main purposes of the TC0 timer is as following.

- **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **External event counter:** Counts system "events" based on falling edge detection of external clock signals at the INT0 input pin.
- **Buzzer output**
- **PWM output**



7.3.2 TC0M MODE REGISTER

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

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

Bit 1 **TC0OUT:** TC0 time out toggle signal output control bit. **Only valid when PWM0OUT = 0.**
0 = Disable, P5.4 is I/O function.
1 = Enable, P5.4 is output TC0OUT signal.

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

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

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

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

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

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

7.3.3 TC0X8 FLAG

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	-	-	-	-	-	TC0X8	-	-
Read/Write	-	-	-	-	-	R/W	-	-
After reset	-	-	-	-	-	0	-	-

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

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

7.3.4 TC0C COUNTING REGISTER

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

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

The equation of TC0C initial value is as following.

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

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

TC0CKS	TC0X8	PWM0	ALOAD0	TC0OUT	N	TC0C valid value	TC0C value binary type	Remark
0	0 (Fcpu/2~Fcpu/256)	0	x	x	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	0	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b	Overflow per 64 count
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b	Overflow per 32 count
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b	Overflow per 16 count
	1 (Fosc/1~Fosc/128)	0	x	x	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	0	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b	Overflow per 64 count
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b	Overflow per 32 count
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b	Overflow per 16 count
1	-	-	-	-	256	0x00~0xFF	00000000b~11111111b	Overflow per 256 count

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

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

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

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

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

TC0RATE	TC0CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fosc/128	8.192 ms	32 us	1000 ms	7812.5 us
001	Fosc/64	4.096 ms	16 us	500 ms	3906.25 us
010	Fosc/32	2.048 ms	8 us	250 ms	1953.125 us
011	Fosc/16	1.024 ms	4 us	125 ms	976.563 us
100	Fosc/8	0.512 ms	2 us	62.5 ms	488.281 us
101	Fosc/4	0.256 ms	1 us	31.25 ms	244.141 us
110	Fosc/2	0.128 ms	0.5 us	15.625 ms	122.07 us
111	Fosc/1	0.064 ms	0.25 us	7.813 ms	61.035 us

7.3.5 TC0R AUTO-LOAD REGISTER

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

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 TC0 interval time error and glitch in PWM and Buzzer output.

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

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

The equation of TC0R initial value is as following.

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

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

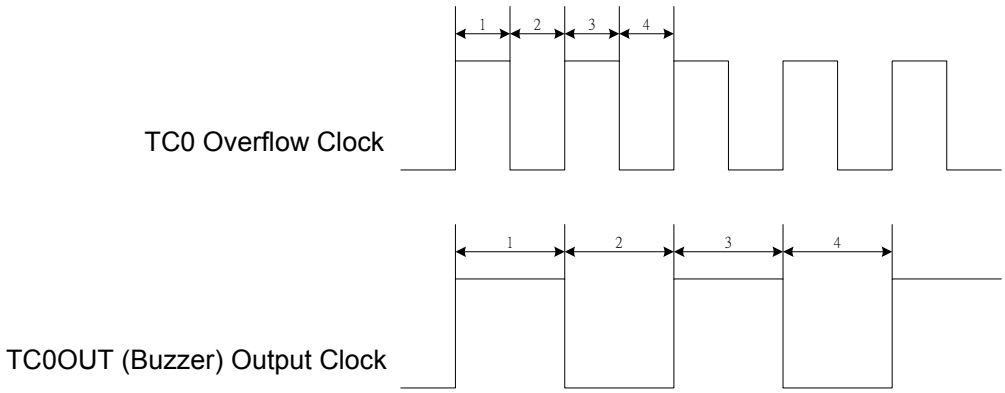
TC0CKS	TC0X8	PWM0	ALOAD0	TC0OUT	N	TC0R valid value	TC0R value binary type
0	0 (Fcpu/2~ Fcpu/256)	0	x	x	256	0x00~0xFF	00000000b~11111111b
		1	0	0	256	0x00~0xFF	00000000b~11111111b
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
	1 (Fosc/1~ Fosc/128)	0	x	x	256	0x00~0xFF	00000000b~11111111b
		1	0	0	256	0x00~0xFF	00000000b~11111111b
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
1	-	-	-	-	256	0x00~0xFF	00000000b~11111111b

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

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

7.3.6 TC0 CLOCK FREQUENCY OUTPUT (BUZZER)

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



➢ **Example: Setup TC0OUT output from TC0 to TC0OUT (P5.4). The external high-speed clock is 4MHz. The TC0OUT frequency is 0.5KHz. Because the TC0OUT signal is divided by 2, set the TC0 clock to 1KHz. The TC0 clock source is from external oscillator clock. T0C rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 131.**

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

MOV     A,#131          ; Set the auto-reload reference value
B0MOV   TC0C,A
B0MOV   TC0R,A

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

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

7.3.7 TC0 TIMER OPERATION SEQUENCE

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

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

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

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

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

☞ **Set TC0 timer clock source.**

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

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

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

☞ **Set TC0 timer auto-load mode.**

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

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

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

; In PWM mode, set PWM cycle.
B0BCLR    FALOAD0     ; ALOAD0, TC0OUT = 00, PWM cycle boundary is
B0BCLR    FTC0OUT     ; 0~255.
or
B0BCLR    FALOAD0     ; ALOAD0, TC0OUT = 01, PWM cycle boundary is
B0BSET    FTC0OUT     ; 0~63.
or
B0BSET    FALOAD0     ; ALOAD0, TC0OUT = 10, PWM cycle boundary is
B0BCLR    FTC0OUT     ; 0~31.
or
B0BSET    FALOAD0     ; ALOAD0, TC0OUT = 11, PWM cycle boundary is
B0BSET    FTC0OUT     ; 0~15.
```

☞ **Set TC0 timer function mode.**

```
        B0BSET      FTC0IEN         ; Enable TC0 interrupt function.
or
        B0BSET      FTC0OUT         ; Enable TC0OUT (Buzzer) function.
or
        B0BSET      FPWM0OUT        ; Enable PWM function.
```

☞ **Enable TC0 timer.**

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

7.4 PWM0 MODE

7.4.1 OVERVIEW

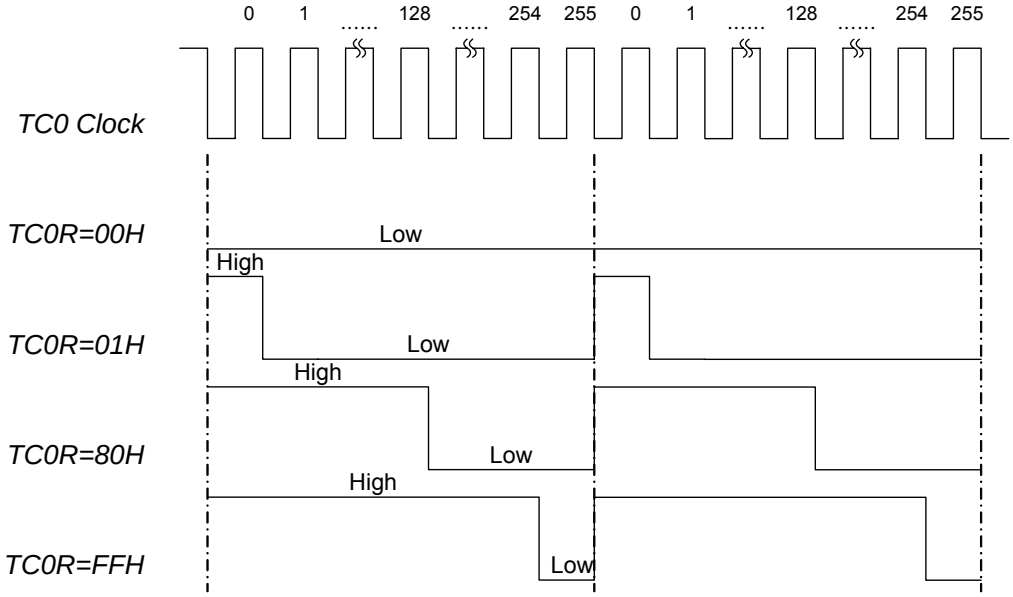
PWM function is generated by TC0 timer counter and output the PWM signal to PWM0OUT pin (P5.4). The 8-bit counter counts modulus 256, 64, 32, 16 controlled by ALOAD0, TC0OUT bits. The value of the 8-bit counter (TC0C) is compared to the contents of the reference register (TC0R). When the reference register value (TC0R) is equal to the counter value (TC0C), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWM0 output is TC0R/256, 64, 32, 16.

PWM output can be held at low level by continuously loading the reference register with 00H. Under PWM operating, to change the PWM's duty cycle is to modify the TC0R.

➢ ***Note: TC0 is double buffer design. Modifying TC0R to change PWM duty by program, there is no glitch and error duty signal in PWM output waveform. Users can change TC0R any time, and the new reload value is loaded to TC0R buffer at TC0 overflow.***

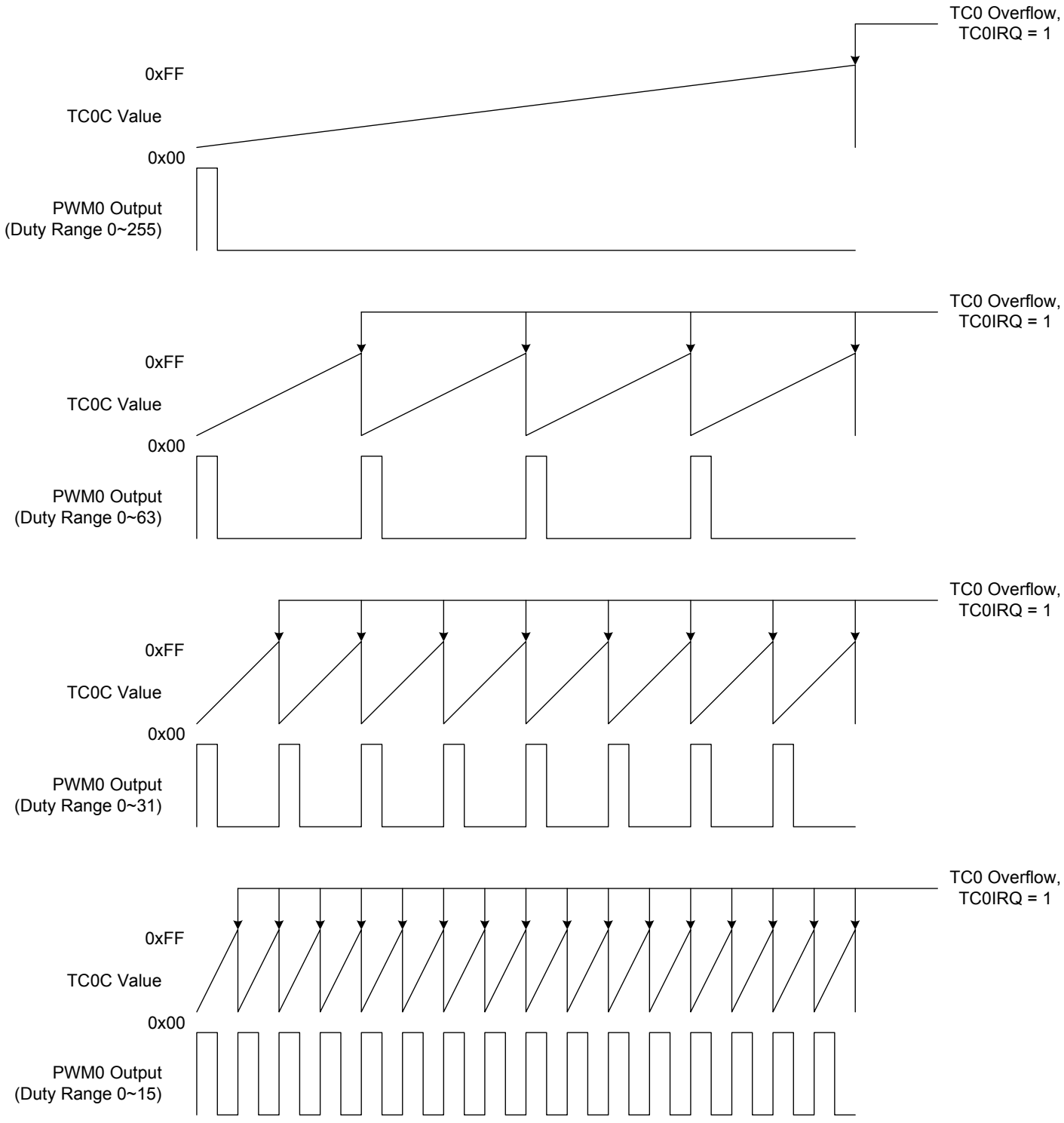
ALOAD0	TC0OUT	PWM duty range	TC0C valid value	TC0R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0	0	0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count
0	1	0/64~63/64	0x00~0x3F	0x00~0x3F	31.25K	Overflow per 64 count
1	0	0/32~31/32	0x00~0x1F	0x00~0x1F	62.5K	Overflow per 32 count
1	1	0/16~15/16	0x00~0x0F	0x00~0x0F	125K	Overflow per 16 count

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



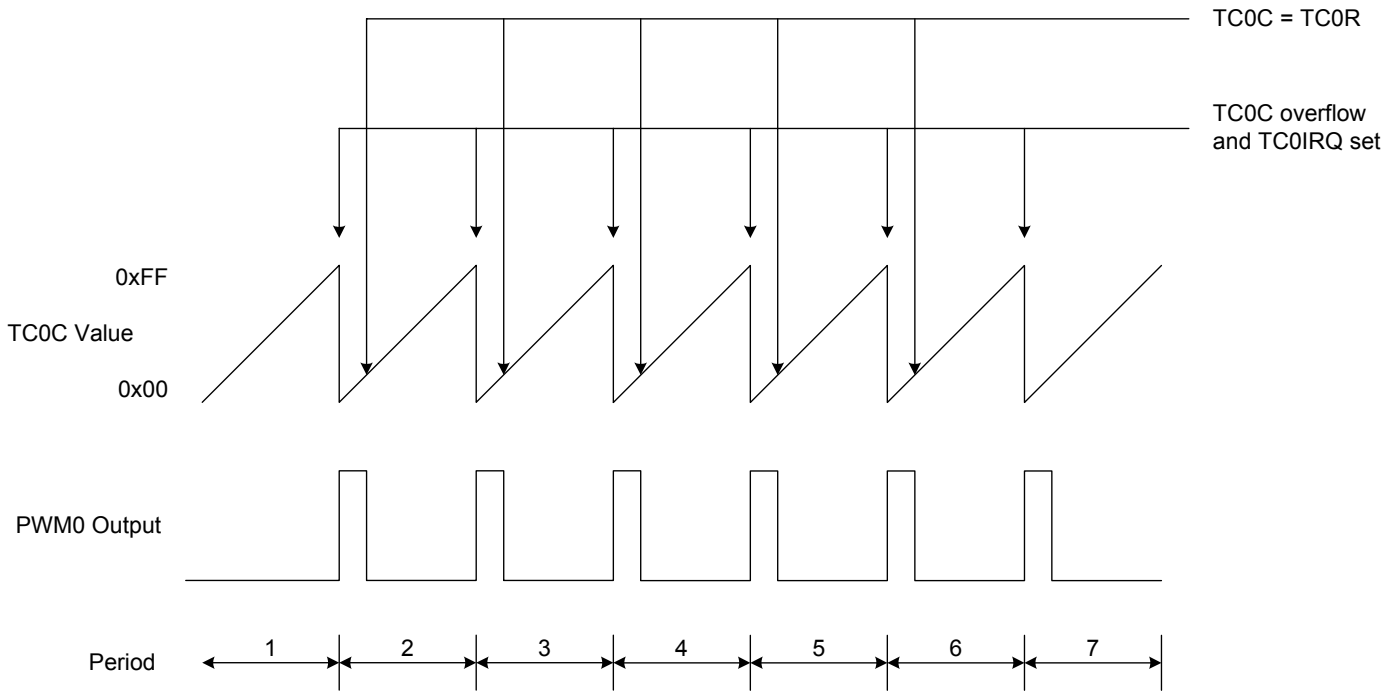
7.4.2 TCxIRQ and PWM Duty

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

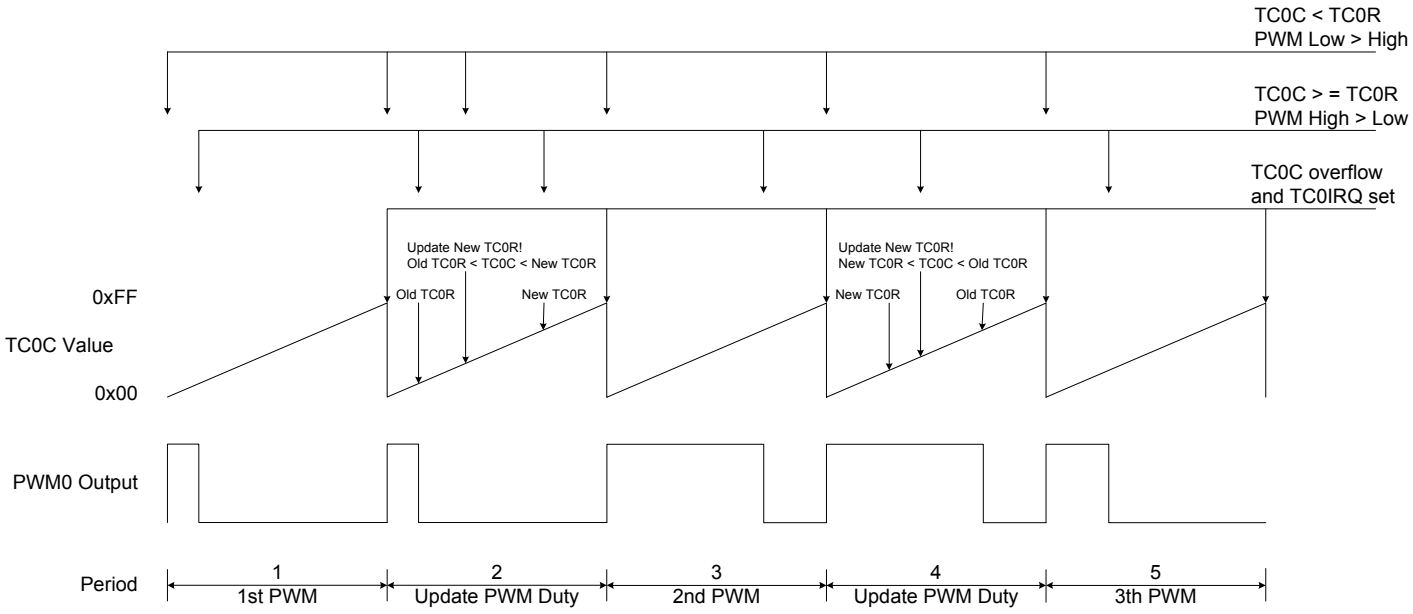


7.4.3 PWM Duty with TCxR Changing

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



Above diagram is shown the waveform with fixed TC0R. In every TC0C overflow PWM output "High, when TC0C≧ TC0R PWM output "Low". If TC0R is changing in the program processing, the PWM waveform will became as following diagram.



In period 2 and period 4, new Duty (TC0R) is set. TC0 is double buffer design. The PWM still keeps the same duty in period 2 and period 4, and the new duty is changed in next period. By the way, system can avoid the PWM not changing or H/L changing twice in the same cycle and will prevent the unexpected or error operation.

7.4.4 PWM PROGRAM EXAMPLE

➢ **Example: Setup PWM0 output from TC0 to PWM0OUT (P5.4). The external high-speed oscillator clock is 4MHz. Fcpu = Fosc/4. The duty of PWM is 30/256. The PWM frequency is about 1KHz. The PWM clock source is from external oscillator clock. TC0 rate is Fcpu/4. The TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 30.**

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

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

B0BCLR    FTC0OUT         ; Set duty range as 0/256~255/256.
B0BCLR    FALOAD0
B0BSET    FPWM0OUT        ; Enable PWM0 output to P5.4 and disable P5.4 I/O function
B0BSET    FTC0ENB         ; Enable TC0 timer
```

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

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

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

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

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

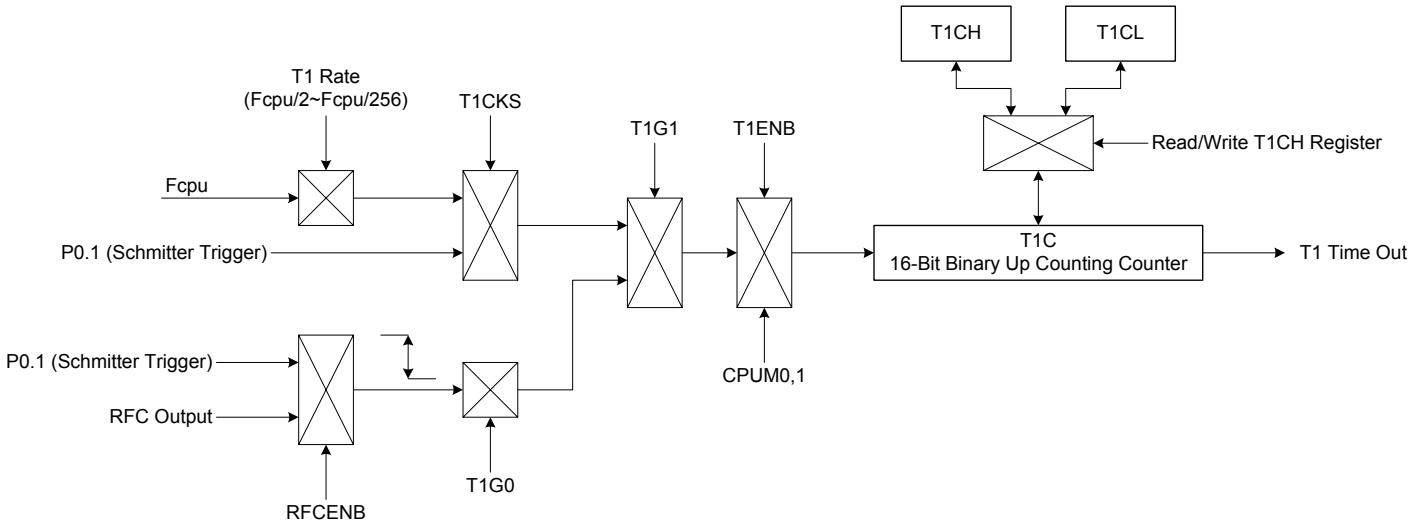
7.5 TIMER 1 (T1)

7.5.1 OVERVIEW

The T1 is an 16-bit binary up timer and event counter. If T1 timer occurs an overflow (from FFFFH to 0000H), it will continue counting and issue a time-out signal to trigger T1 interrupt to request interrupt service.

The main purposes of the T1 timer is as following.

- **16-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **External event counter:** Counts system "events" of external clock signals at the P0.1/T1IN input pin. The trigger edge is controlled by T1G1, T1G0 bits
- **RFC counter:** Counts RFC's events of RFC clock signals at RFC output. The trigger edge is controlled by T1G0 bit. Please refer to "RFC DESCRIPTION".
- **Input signal frequency measurement:** In T1G1 = 0 of event counter and RFC mode, the input signal frequency from T1IN/RFC can be measure by sampling other timers (T0, TC0). Please refer to "RFC DESCRIPTION" about RCF mode application.
- **Input signal plus width measurement:** In T1G1 = 1 of event counter and RFC mode, the input signal high and low plus width from T1IN/RFC can be measure by T1CH, T1CH buffers. Please refer to "RFC DESCRIPTION" about RCF mode application.



- **T1 purpose control table.**

T1 purpose	T1 clock source	T1CKS	RFCENB	T1G1	T1G0	Edge Direction
16-bit timer	Fcpu	0	0	0	-	-
16-bit event counter (P0.1 input frequency measurement)	P0.1/T1IN	1	0	0	0	Falling edge.
					1	Rising edge.
P0.1 pulse width measurement	P0.1/T1IN	0	0	1	0	High pulse width measurement.
					1	Low pulse width measurement.
RFC frequency measurement	RFC output	1	1	0	0	Falling edge.
					1	Rising edge.
RFC pulse width measurement	RFC output	0	1	1	0	High pulse width measurement.
					1	Low pulse width measurement.

➢ ***Note: In pulse width measurement mode, the T1CKS must be set as "0" by program.***

7.5.2 T1M MODE REGISTER

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1RATE2	T1RATE1	T1RATE0	T1CKS	-	T1IEN	T1IRQ
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 7 **T1ENB:** T1 counter control bit.
0 = Disable T1 timer.
1 = Enable T1 timer.

Bit [6:4] **T1RATE[2:0]:** T1 timer internal clock select bits.
000 = fcpu/256.
001 = fcpu/128.
…
110 = fcpu/4.
111 = fcpu/2.

Bit 3 **T1CKS:** T1 timer clock source select bit.
0 = Fcpu.
1 = External clock source from P0.1 when RFCENB=0, or RFC output when RFCENB=1.

Bit 1 **T1IEN:** T1 timer interrupt control bit.
0 = Disable T1 interrupt function.
1 = Enable T1 interrupt function.

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

7.5.3 T1G1, T1G0 FLAGS

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	-	P00G1	P00G0	-	T1G1	T1G0
Read/Write	-	-	-	R/W	R/W	-	R/W	R/W
After reset	-	-	-	1	0	-	0	0

Bit 1 **T1G1:** T1 pulse width measurement control bit.
0 = Disable T1 pulse width measurement.
1 = Enable T1 pulse width measurement. **T1CKS must be set as "0".**

Bit 0 **T1G0:** T1 trigger direction select bit.
0 = Falling edge trigger (T1G1=0). High pulse width measurement (T1G1=1).
1 = Rising edge trigger (T1G1=0). Low pulse width measurement (T1G1=1).

7.5.4 T1CH, T1CL COUNTING REGISTER

T1C is an 16-bit counter register for T1 interval time control. T1CH is high byte of T1C. T1CL is low byte of T1C.

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 equation of T1C [T1CH, T1CL] initial value is as following.

T1C initial value = 65536 - (T1 interrupt interval time * input clock)

- **Example: To set 10ms interval time for T1 interrupt. High clock is external 4MHz. Fcpu=Fosc/4. Select T1RATE=010 (Fcpu/64).**

*T1C initial value = 65536 - (T1 interrupt interval time * input clock)*
*= 65536 - (10ms * 4MHz / 4 / 64)*
*= 65536 - (10^{-2} * 4 * 10^{6} / 4 / 64)*
= 65380
= FF64H ;T1CH=0xFF, T1CL=0x64

The basic timer table interval time of T1.

T1RATE	T1CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fcpu/256	16.777 s	256 us	2048 s	31250 us
001	Fcpu/128	8.388 s	128 us	1024 s	15625 us
010	Fcpu/64	4.194 s	64 us	512 s	7812.5 us
011	Fcpu/32	2.097 s	32 us	256 s	3906.25 us
100	Fcpu/16	1.048 s	16 us	128 s	1953.125 us
101	Fcpu/8	524.288 ms	8 us	64 s	976.563 us
110	Fcpu/4	262.144 ms	4 us	31 s	488.281 us
111	Fcpu/2	131.072 ms	2 us	16 s	244.141 us

The T1 16-bit counter buffer is T1CH and T1CL combination. System provides a routine to process the 16-bit data buffer under 8-bit situation to make high/low bytes of 16-bit data processed at the same time. T1CH register is the key to control the T1 16-bit counter buffer processed through T1CH, T1CL buffers. Export T1C 16-bit buffer data to T1CH, T1CL registers is by reading T1CH register. Import T1C 16-bit buffer data from T1CH, T1CL registers is by writing T1CH register after setting T1CL register data.

➢ **Example: Reading T1C 16-bit buffer data is controlled by reading T1CH register. Read T1CH register data and low byte data of T1C 16-bit buffer exporting to T1CL register at the same time.**

```
B0MOV       A, T1CH         ; Read T1CH first and T1C low byte data exported to T1CL.
…
B0MOV       A, T1CL         ; Read T1CL data from buffer.
…
```

➢ ***Note: Read T1CH first when reading T1C 16-bit buffer.***

➢ **Example: Writing and setting T1C 16-bit buffer data is controlled by writing data into T1CH register. When writing T1CH register data, T1CH, T1CL data are imported into T1C 16-bit buffer at the same time. Setting T1CL register data first is necessary, or the T1C low byte data would be error.**

```
…
B0MOV       T1CL, A         ; Write T1CL data into T1CL buffer first.
…
B0MOV       T1CH, A         ; Write T1CH data and T1CH, T1CL are imported to T1C
                            ; 16-bit buffer.
```

➢ ***Note: Write T1CL first when writing T1C 16-bit buffer.***

➢ **Example: Write T1CL is by write T1CH. Only executing "CLR T1CL" and no do any T1CH writing operation can't clear T1CL buffer. Clear T1CL must be using "MOV" instruction as following.**

```
MOV         A, #0
B0MOV       T1CL, A         ; Write "0" into T1CL to clear T1CL.
…
B0MOV       T1CH, A         ; Write T1CH data and T1CH, T1CL are imported to T1C
                            ; 16-bit buffer.
```

➢ ***Note: Don't clear T1CL by "CLR" instruction.***

7.5.5 T1 TIMER OPERATION SEQUENCE

T1 timer operation sequence of setup T1 timer is as following.

☞ **Stop T1 timer counting, disable T1 interrupt function and clear T1 interrupt request flag.**

```
B0BCLR    FT1ENB      ; T1 timer.
B0BCLR    FT1IEN      ; T1 interrupt function is disabled.
B0BCLR    FT1IRQ      ; T1 interrupt request flag is cleared.
```

☞ **Set T1 timer rate.**

```
MOV       A, #0xxx0000b   ;The T1 rate control bits exist in bit4~bit6 of T1M. The
                          ; value is from x000xxxxb~x111xxxxb.
B0MOV     T1M,A           ; T1 timer is disabled.
```

☞ **Set T1 clock source from internal clock source (Fcpu), external clock source (P0.1/T1IN), or RFC.**

```
B0BCLR    FT1CKS      ; Select T1 internal clock source.
```
or
```
B0BSET    FT1CKS      ; Select T1 external clock source.
```
or
```
B0BSET    FRFCENB     ; Select T1 RFC clock source.
```

☞ **Set T1 interrupt interval time.**

```
MOV       A,#7FH
B0MOV     T1CL,A      ; Set T1CL value.
MOV       A,#7FH
B0MOV     T1CH,A      ; Set T1CH value.
```

☞ **Set T1 timer function mode.**

```
B0BSET    FT1IEN      ; Enable T1 interrupt function.
```
or
```
B0BSET    FT1G1       ; Enable T1 pulse width measurement.
```

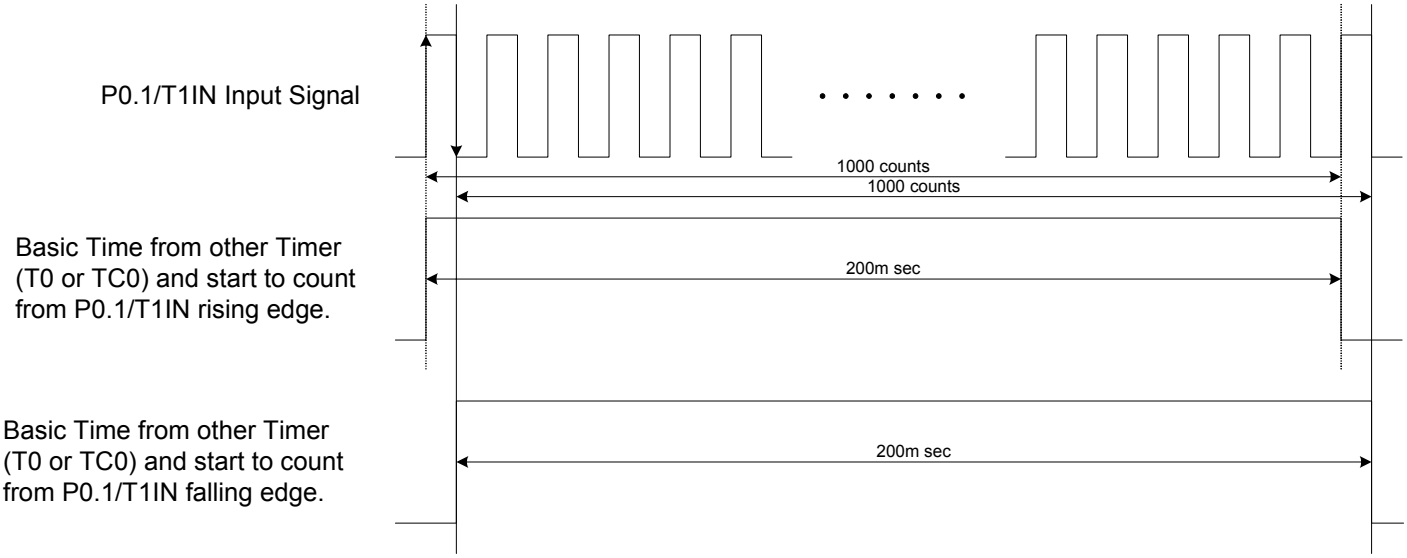
☞ **Enable T1 timer.**

```
B0BSET    FT1ENB      ; Enable T1 timer.
```

7.5.6 T1IN INPUT FREUQENCY MEASUREMENT

T1 is a 16-bit timer/counter. The 16-bit counter buffer (T1CH, T1CL) is long enough to measure high-speed signal's frequency from P0.1/T1IN input. For P0.1/T1IN input frequency measurement, T1 counts input signal clock number and using other timer (T0 to TC0) to generate one period time to be the basic time to calculate input frequency. When the period time is time out, disable T1 counter and the number of P0.1/T1IN input clock is stored in T1CH and T1CL registers. Using the basic time to divide by input clock number and get the one cycle period. The reciprocal of one cycle period is the input signal's frequency from P0.1/T1IN (F=1/T). If the T1CH, T1CL = 0x1388 =5000 and the basic time 50m sec, the input frequency is 100KHz. Input frequency = 1/ (50ms/5000) = 100KHz

- **T1 clock source is event counter input (P0.1/T1IN).**
- **T0 or TC0 clock source is internal clock (Fcpu).**



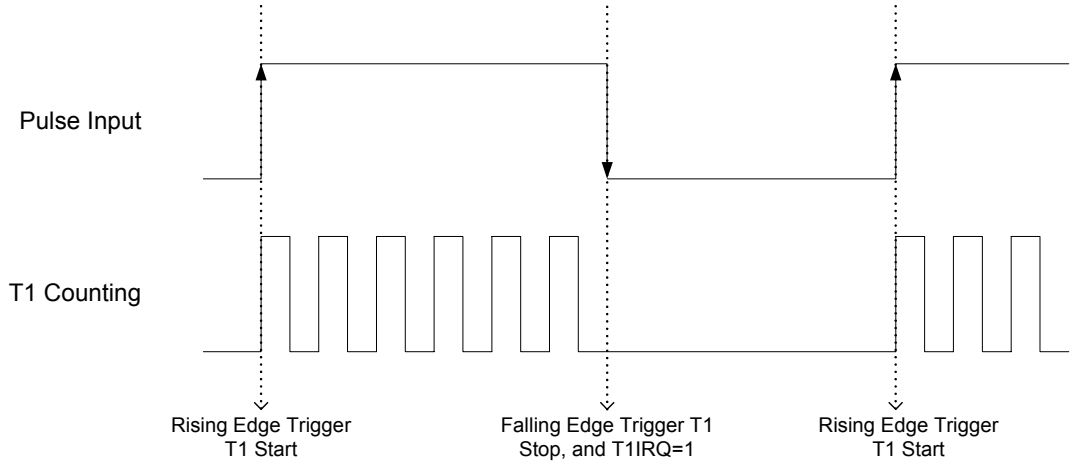
➢ **Example: Using T0 to measure P0.1/T1IN input 100KHz frequency. The basic time generated by T0 is 50ms. Fhosc=4MHz, Fcpu=Fhosc/4=1mips. The P0.1/T1IN input trigger is rising edge.**

```
; Set T1 mode.
        CLR     T1M             ; Clear T1M register.
        MOV     A, #0
        B0MOV   T1CL, A
        B0MOV   T1CH, A         ; Clear T1 counter buffers.

        B0BSET  FT1CKS          ; Set T1 to event counter mode.
        B0BSET  FT1G0           ; Event counter input trigger is rising edge.
        B0BCLR  FT1IRQ          ; Clear T1 interrupt request flag.

; Set T0 mode.
        B0BCLR  T0ENB           ; Disable T0 timer.
        B0BCLR  T0IEN           ; Disable T0 interrupt function.
        B0BCLR  T0IRQ           ; Disable T0 interrupt request flag.

        MOV     A, #00h
        B0MOV   T0M, A          ; Set T0 rate is Fcpu/256.

        MOV     A, #61
        B0MOV   T0C, A          ; Set T0 interval time to 50ms.


; Start to measure P0.1/T1IN input frequency.

Chk_1st_Eg:
        B0BTS1  P0.1            ; Check P0.1 rising edge.
        JMP     Chk_1st_Eg
        B0BSET  FT1ENB          ; Enable T1 counter.
        B0BSET  FT0ENB          ; Enable T0 timer.

Chk_50ms:
        B0BTS1  FT0IRQ          ; Check T0 50 ms time out.
        JMP     Chk_50ms
        B0BCLR  FT1ENB          ; Stop T1 counter.

Cal_Freq:                       ; Calculate P0.1/T1IN input frequency from T1CH, T1CL.
        B0MOV   A, T1CH         ; T1CH = 13H
        …
        B0MOV   A, T1CL         ; T1CL = 88H
        …
        …
```

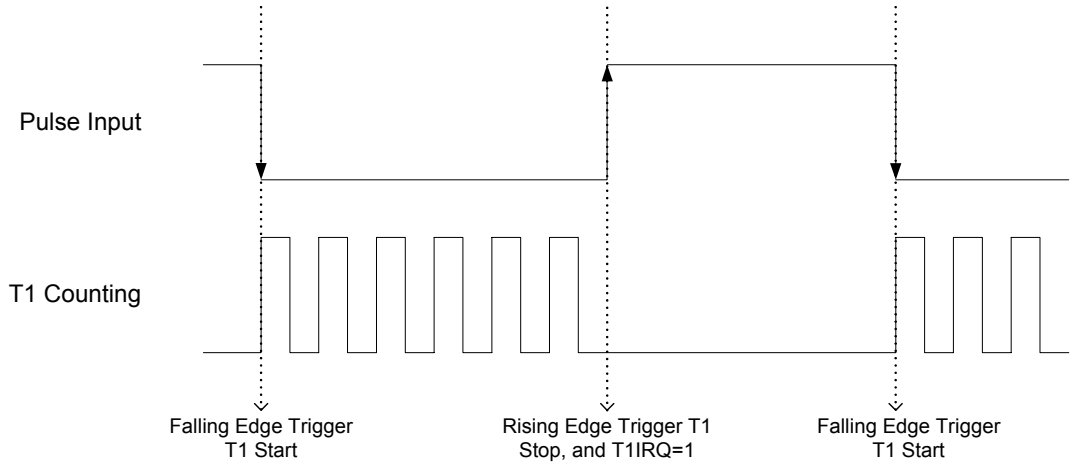
7.5.7 T1IN INPUT PULSE WIDTH MEASUREMENT

T1 builds in pulse width measurement function and controlled by T1G1 bit of PEDGE register. T1G1=0, disable pulse width measurement. T1G1=1, enable pulse width measurement. The input signal is from P0.0/T1IN or RFC. Please refer to “RFC DESCRIPTION” for RFC pulse width measurement. The section explains the pulse width measurement of P0.1/T1IN. The pulse includes high and low directions and controlled by T1G0 bit of PEDGE register. The time base of pulse width measurement is T1 and the clock source is from internal clock (Fcpu). T1IRQ=0, T1 starts to count when trigger occurrence. T1IRQ=1, T1 stops counting and T1CH, T1CL data are latched. T1IRQ is cleared by program and T1 starts to count automatically. The T1CH, T1CL dumped by program is during T1 stopping period, or T1 is counting and the T1C data is counted.

- **T1G1, T1G0=10: Enable high pulse width measurement. Rising edge trigger starts T1 to count and falling edge stops T1 counting.**



- **T1G1, T1G0=11: Enable low pulse width measurement. Falling edge trigger starts T1 to count and rising edge stops T1 counting.**



➢ **Example: Measure 50Hz clock input pulse width from P0.1/T1IN pin using T1 pulse width measurement. Fhosc=4MHz, Fcpu=Fhosc/4=1mips. T1 base time is 256us. Pulse width is T1C buffer * T1 base time. T1CH, T1CL = 0x0027=39, Pulse width = 256us*39 = 9.9ms. (Pulse width of 50Hz signal is 10ms)**

High Pulse Width Measurement.

```
; Set T1 mode.
        CLR     T1M         ; Clear T1M register and set T1 rate is Fcpu/256.
        MOV     A, #0
        B0MOV   T1CL, A
        B0MOV   T1CH, A     ; Clear T1 counter buffers.

        B0BSET  FT1G1       ; Enable pulse width measurement.
        B0BCLR  FT1G0       ; Select high pulse direction.
        B0BCLR  FT1IRQ      ; Clear T1 interrupt request flag.
        B0BSET  FT1ENB      ; Enable T1 timer.


; Check T1IRQ=1.

Chk_T1IRQ:
        B0BTS1  FT1IRQ      ; Check T1IRQ=1.
        JMP     Chk_T1IRQ
        B0MOV   A, T1CH     ; Pulse width measure end.
        …                   ; T1CH=00h
        B0MOV   A, T1CL     ; T1CL=27h
        …

; Measure next high pulse.
        MOV     A, #0
        B0MOV   T1CL, A
        B0MOV   T1CH, A     ; Clear T1 counter buffers.
        B0BCLR  T1IRQ       ; Clear T1 interrupt request flag and start T1 again.
        JMP     Chk_T1IRQ
```

Low Pulse Width Measurement.

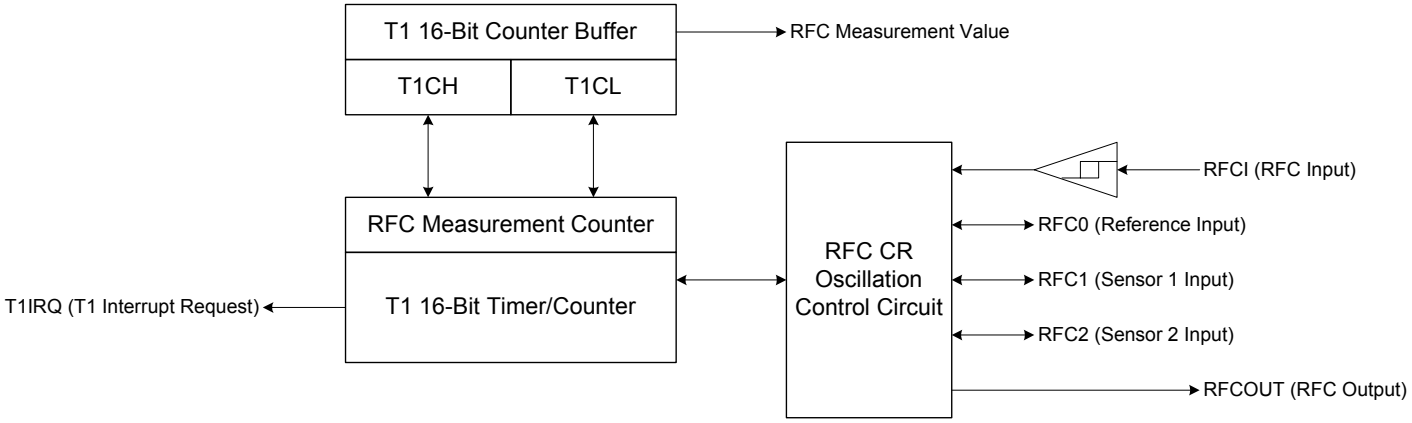
```
; Set T1 mode.
        CLR     T1M         ; Clear T1M register and set T1 rate is Fcpu/256.
        MOV     A, #0
        B0MOV   T1CL, A
        B0MOV   T1CH, A     ; Clear T1 counter buffers.

        B0BSET  FT1G1       ; Enable pulse width measurement.
        B0BSET  FT1G0       ; Select low pulse direction.
        B0BCLR  FT1IRQ      ; Clear T1 interrupt request flag.
        B0BSET  FT1ENB      ; Enable T1 timer.


; Check T1IRQ=1.
Chk_T1IRQ:
        B0BTS1  FT1IRQ      ; Check T1IRQ=1.
        JMP     Chk_T1IRQ
        B0MOV   A, T1CH     ; Pulse width measure end.
        …                   ; T1CH=00h
        B0MOV   A, T1CL     ; T1CL=27h
        …

; Measure next high pulse.
        MOV     A, #0
        B0MOV   T1CL, A
        B0MOV   T1CH, A     ; Clear T1 counter buffers.
```

```
        B0BCLR      T1IRQ           ; Clear T1 interrupt request flag and start T1 again.
        JMP         Chk_T1IRQ
```

8 RESISTANCE TO FREQURNCY CONVERTER (RFC)

8.1 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 three channels. One channel is used reference channel, and the other two 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 5. When RFCENB = 1, P5.0~P5.2, P5.3 are selected to RFC0~RFC2, RFCI. P5.5 is shared with RFCOUT controlled by RFCOUT bit of RFC register.



8.2 RFCM REGISTER

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RFCM	RFCOUT	-	-	-	-	RFCH1	RFCH0	RFCENB
Read/Write	R/W	-	-	-	-	R/W	R/W	R/W
After reset	0	-	-	-	-	0	0	0

Bit 7 **RFCOUT:** RFC output control bit.
0 = Disable RFC output, P5.5 is general purpose I/O.
1 = Enable RFC output, P5.5 is RFCOUT pin.

Bit[2:1] **RFCH[2:1]:** RFC input channels select bit.

RFCH[2:1]	RFC Channel	Description
00	RCF Channel 0 (P5.0)	Enable RFC channel 0 (P5.0) and set P5.1, P5.2 as analog input to avoid extra power consumption.
01	RCF Channel 1 (P5.1)	Enable RFC channel 1 (P5.1) and set P5.0, P5.2 as analog input to avoid extra power consumption.
10	RCF Channel 2 (P5.2)	Enable RFC channel 2 (P5.2) and set P5.0, P5.1 as analog input to avoid extra power consumption.
11	Reserved	-

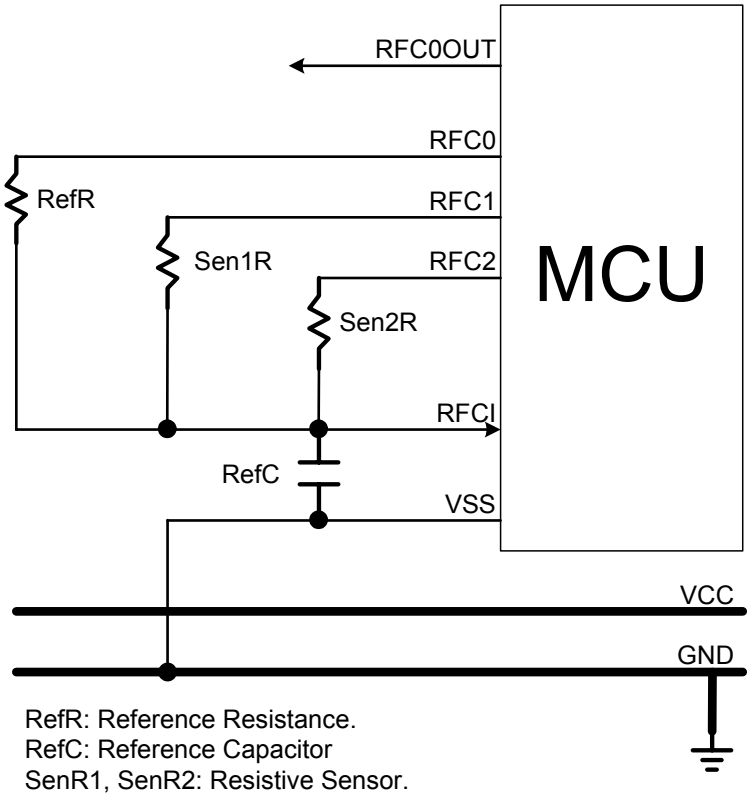
Bit 0 **RFCENB:** RFC control bit.
0 = Disable RFC function.
1 = Enable RFC function.

✷ ***Note:***

1. ***If RFCENB=1 then P5.0 is RFC0, P5.1 is RFC1, P5.2 is RFC2 and P5.3 is RFCI.***
2. ***If RFCOUT=1 then P5.5 is RFCOUT.***
3. ***Before RFC enable, P5.0~P5.3 must be set as input mode without pull-up resistor first by program.***

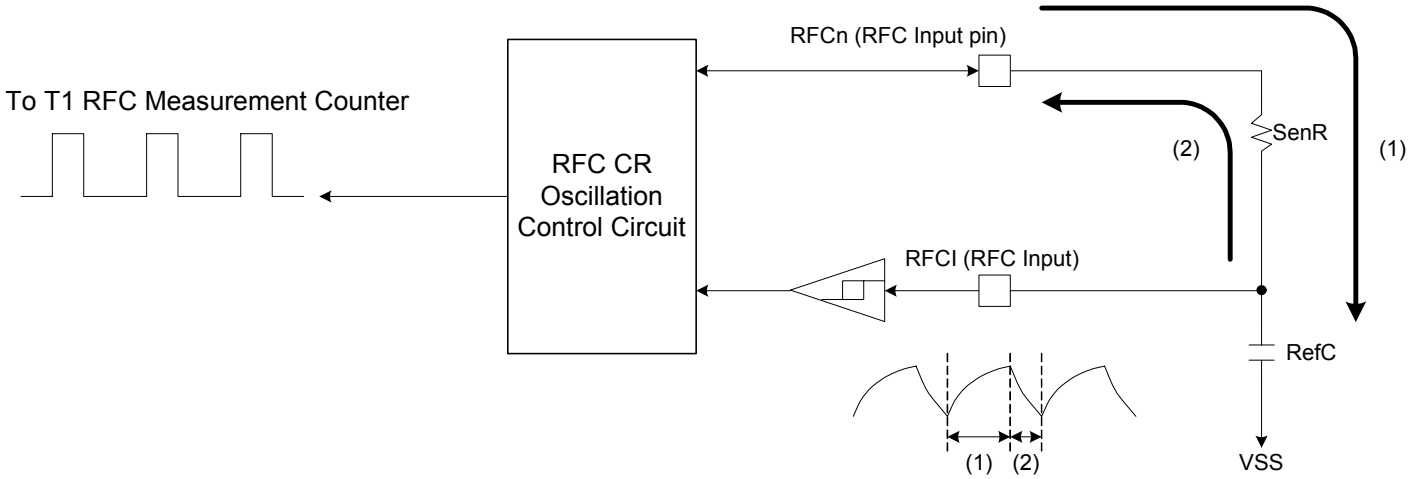
8.3 RFC CIRCUIT

RFC can configure with a reference resistance and two sensors. The connection diagram is as following. The RFC circuit supports two resistive sensor inputs and one reference resistance. One terminal of resistance is connected to RFC channel input, and the other is connected to RFCI pin. The capacitor of RFC circuit is commonly used for the reference resistance and resistive sensors, and connected between RFCI pin and VSS of MCU. These resistances are switched by RFCH[1:0] bits of RFC register. RFCH[1:0] = 00, RFC input selects to RFC0 pin (P5.0). RFCH[1:0] = 01, RFC input selects to RFC1 pin (P5.1). RFCH[1:0] = 10, RFC input selects to RFC2 pin (P5.2). One RFC input is selected, the other pins stay in analog input status to avoid extra power consumption. RFC digitally converting clock is output from RFCOUT pin (P5.5) controlled by RFCOUT bit of RFC register.

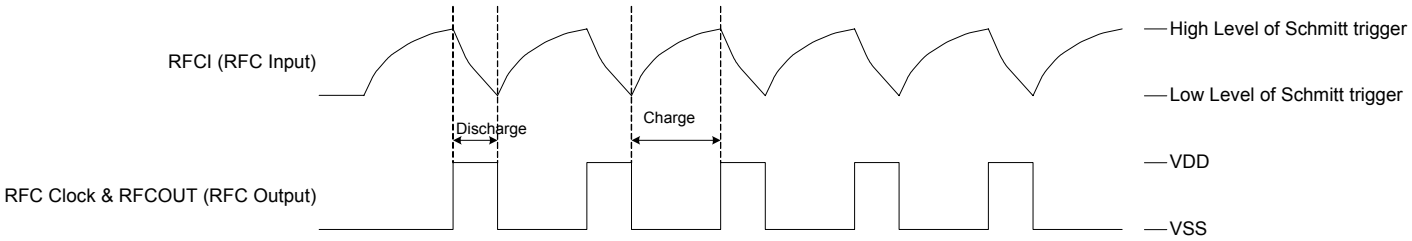


RefR: Reference Resistance.
RefC: Reference Capacitor
SenR1, SenR2: Resistive Sensor.

8.4 RFC OPERATION



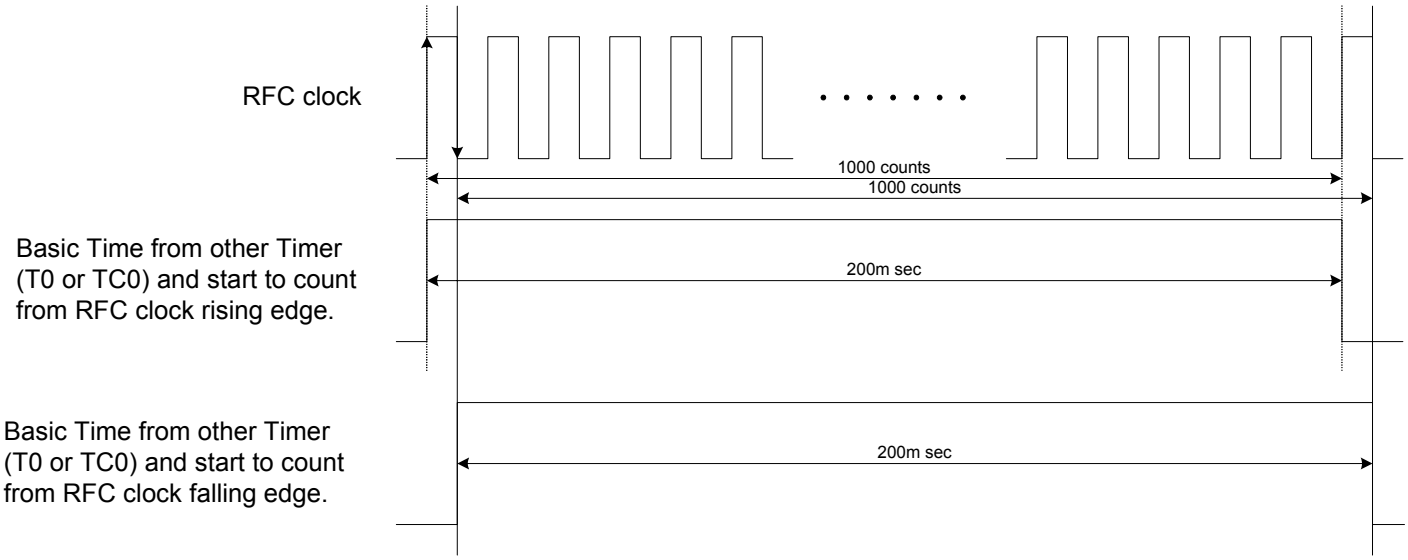
RFC operation is base on charge and discharge capacitor to obtain the converting clock. RFCI (RFC input pin) is Schmitt trigger structure. The RFCENB=1 turns on RFC function. RFCn (RFC channel pin RFCH[2:0]) charges and discharges the reference capacitor. Loop (1) of above figure is charge mode. The capacitor is charged by RFCn, and the voltage level into RFCI increases. RFCI detects the voltage level to Schmitt trigger high-level voltage, and then RFCn switches to discharge mode. Loop (2) is discharge mode. RFCn discharges the capacitor and the voltage level starts to fall down. RFCI detects the voltage level to Schmitt trigger low-level voltage, and then RFCn switches to charge mode. The result of charge and discharge switching is converted by CR 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 (P5.5) when RFCOUT =1.

8.4.1 RFC FREUQENCY MEASUREMENT

RFC frequency measurement supports high speed RFC frequency by T1 16-bit timer/counter. T1 is in event counter mode and the clock source is RFC clock. T1 counts RFC clock number and using other timer (T0 to TC0) to generate one period time to be the basic time to calculate RFC frequency. When the period time is time out, disable T1 counter and the number of RFC clock is stored in T1CH and T1CL registers. Using the basic time to divide by RFC clock number and get the one cycle period. The reciprocal of one cycle period is the input signal's frequency from RFC (F=1/T). If the T1CH, T1CL = 0x1388 =5000 and the basic time 50m sec, the RFC frequency = 1/ (50ms/5000) = 100KHz



➢ **Example: Using T0 to measure RFC 100KHz frequency from RFC0. The basic time generated by T0 is 50ms. Fhosc=4MHz, Fcpu=Fhosc/4=1mips. The RFC clock trigger is rising edge.**

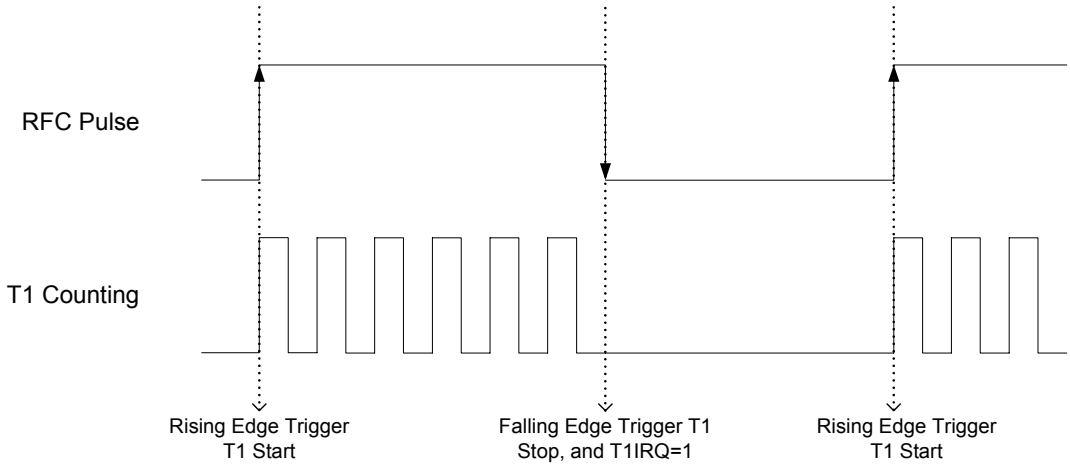
```
; Set T1 mode.
        CLR     T1M             ; Clear T1M register.
        MOV     A, #0
        B0MOV   T1CL, A
        B0MOV   T1CH, A         ; Clear T1 counter buffers.

        B0BSET  FT1CKS          ; Set T1 to event counter mode.
        B0BSET  FT1G0           ; Event counter input trigger is rising edge.
        B0BCLR  FT1IRQ          ; Clear T1 interrupt request flag.

; Set T0 mode.
        B0BCLR  FT0ENB          ; Disable T0 timer.
        B0BCLR  FT0IEN          ; Disable T0 interrupt function.
        B0BCLR  FT0IRQ          ; Disable T0 interrupt request flag.

        MOV     A, #00h
        B0MOV   T0M, A          ; Set T0 rate is Fcpu/256.

        MOV     A, #61
        B0MOV   T0C, A          ; Set T0 interval time to 50ms.
```

```
; Set RFC.
        CLR     RFC         ; Clear RFC register.
        B0BCLR  FRFCH0      ; Set RFCH[1:0]=00 to select RFC0 (RFC channel 0).
        B0BCLR  FRFCH1
        CLR     P5UR        ; Disable P5 pull-up resistor.
        MOV     A, #11110000B
        AND     P5M, A      ; Set P5.0~P5.3 to input mode.

        B0BSET  FRFCENB     ; Enable RFC function and the T1 event counter input is RFC
                            ; clock.
        B0BSET  FT1ENB      ; Enable T1 counter.
        B0BSET  FT0ENB      ; Enable T0 timer.

Chk_50ms:
        B0BTS1  FT0IRQ      ; Check T0 50 ms time out.
        JMP     Chk_50ms
        B0BCLR  FT1ENB      ; Stop T1 counter.

Cal_Freq:                   ; Calculate RFC frequency by T1CH, T1CL.
        B0MOV   A, T1CH     ; T1CH = 13H
        …
        B0MOV   A, T1CL     ; T1CL = 88H
        …
        …
```

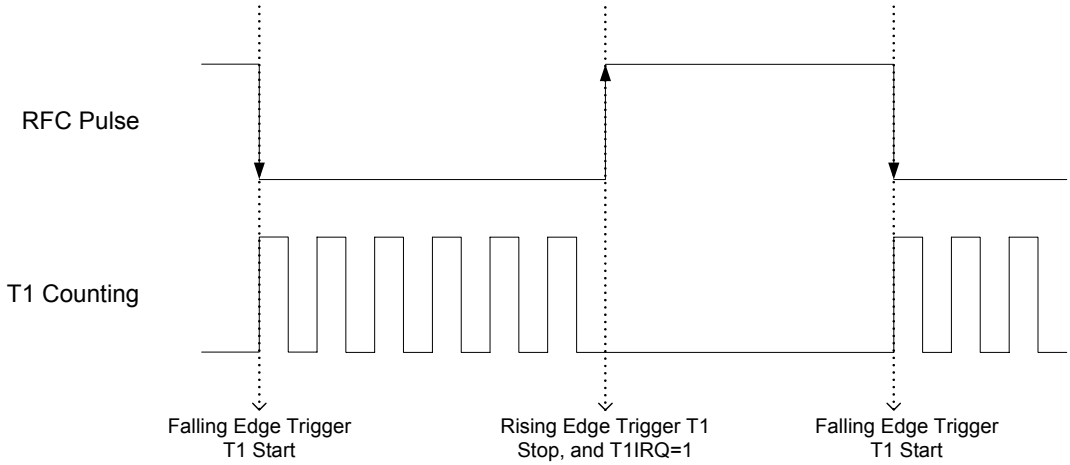
8.4.2 RFC PULSE WIDTH MEASUREMENT

RFC pulse width measurement supports low speed RFC clock using T1 pulse width measurement function. T1 input signal is from RFC when RFCENB=1. The pulse includes high and low directions and controlled by T1G0 bit of PEDGE register. The time base of pulse width measurement is T1 and the clock source is from internal clock (Fcpu). T1IRQ=0, T1 starts to count when trigger occurrence. T1IRQ=1, T1 stops counting and T1CH, T1CL data are latched. T1IRQ is cleared by program and T1 starts to count automatically. The T1CH, T1CL dumped by program is during T1 stopping period, or T1 is counting and the T1C data is counted.

- **T1G1, T1G0=10: Enable high pulse width measurement. Rising edge trigger starts T1 to count and falling edge stops T1 counting.**



- **T1G1, T1G0=11: Enable low pulse width measurement. Falling edge trigger starts T1 to count and rising edge stops T1 counting.**



➢ **Example: Measure 50Hz RFC clock input pulse width using T1 pulse width measurement. Fhosc=4MHz, Fcpu=Fhosc/4=1mips. T1 base time is 256us. Pulse width is T1C buffer * T1 base time. T1CH, T1CL = 0x0027=39, Pulse width = 256us*39 = 9.9ms. (Pulse width of 50Hz signal is 10ms)**

RFC High Pulse Width Measurement.

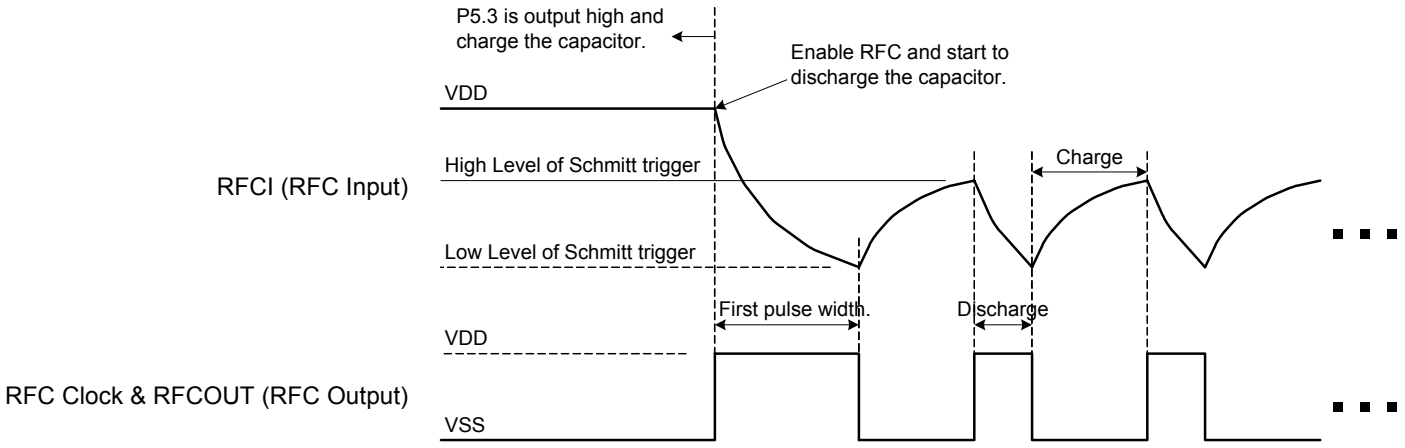
```
; Set T1 mode.
        CLR     T1M             ; Clear T1M register and set T1 rate is Fcpu/256.
        MOV     A, #0
        B0MOV   T1CL, A
        B0MOV   T1CH, A         ; Clear T1 counter buffers.

        B0BSET  FT1G1           ; Enable pulse width measurement.
        B0BCLR  FT1G0           ; Select high pulse direction.
        B0BCLR  FT1IRQ          ; Clear T1 interrupt request flag.

; Set RFC.
        CLR     RFC             ; Clear RFC register.
        B0BCLR  FRFCH0          ; Set RFCH[1:0]=00 to select RFC0 (RFC channel 0).
        B0BCLR  FRFCH1
        CLR     P5UR            ; Disable P5 pull-up resistor.
        MOV     A, #11110000B
        AND     P5M, A          ; Set P5.0~P5.3 to input mode.

        B0BSET  FRFCENB         ; Enable RFC function and the T1 pulse width trigger is RFC
                                ; clock.
        B0BSET  FT1ENB          ; Enable T1 timer.

; Check T1IRQ=1.

Chk_T1IRQ:
        B0BTS1  FT1IRQ          ; Check T1IRQ=1.
        JMP     Chk_T1IRQ
        B0MOV   A, T1CH         ; Pulse width measure end.
        …                       ; T1CH=00h
        B0MOV   A, T1CL         ; T1CL=27h
        …

; Measure next high pulse.
        MOV     A, #0
        B0MOV   T1CL, A
        B0MOV   T1CH, A         ; Clear T1 counter buffers.
        B0BCLR  T1IRQ           ; Clear T1 interrupt request flag and start T1 again.
        JMP     Chk_T1IRQ
```

Low Pulse Width Measurement.

```
; Set T1 mode.
        CLR     T1M         ; Clear T1M register and set T1 rate is Fcpu/256.
        MOV     A, #0
        B0MOV   T1CL, A
        B0MOV   T1CH, A     ; Clear T1 counter buffers.

        B0BSET  FT1G1       ; Enable pulse width measurement.
        B0BSET  FT1G0       ; Select low pulse direction.
        B0BCLR  FT1IRQ      ; Clear T1 interrupt request flag.

; Set RFC.
        CLR     RFC         ; Clear RFC register.
        B0BCLR  FRFCH0      ; Set RFCH[1:0]=00 to select RFC0 (RFC channel 0).
        B0BCLR  FRFCH1
        CLR     P5UR        ; Disable P5 pull-up resistor.
        MOV     A, #11110000B
        AND     P5M, A      ; Set P5.0~P5.3 to input mode.

        B0BSET  FRFCENB     ; Enable RFC function and the T1 pulse width trigger is RFC
                            ; clock.
        B0BSET  FT1ENB      ; Enable T1 timer.

; Check T1IRQ=1.

Chk_T1IRQ:
        B0BTS1  FT1IRQ      ; Check T1IRQ=1.
        JMP     Chk_T1IRQ
        B0MOV   A, T1CH     ; Pulse width measure end.
        …                   ; T1CH=00h
        B0MOV   A, T1CL     ; T1CL=27h
        …

; Measure next high pulse.
        MOV     A, #0
        B0MOV   T1CL, A
        B0MOV   T1CH, A     ; Clear T1 counter buffers.
        B0BCLR  T1IRQ       ; Clear T1 interrupt request flag and start T1 again.
        JMP     Chk_T1IRQ
```

8.5 RFC FIRST PULSE WIDTH MEASUREMENT

Measure the first pulse width is general application. Use P5.3 to charge RC until the capacitor voltage reaching VDD first. Then enable RFC function to discharge the capacitor electric charge and T1 starts to count. When the voltage drops to Schmitt-trigger low level, T1 stops and the T1 counting buffer value (T1CH, T1CL) is the pulse width. The first pulse width measurement of RFC is for high impedance resistive sensor application.



The P5.3 outputs high to charge the capacitor voltage to VDD. The RFC trigger edge initial status is low. When RFC enables (RFCENB=1), P5.3 becomes RFCI input pin. The system detects the first rising edge from the RFC trigger initial status to the capacitor high voltage and starts T1 counting. RFC channel discharges the capacitor and the voltage is dropping. The discharge voltage is from VDD to Schmitt-trigger low level. T1 stops when system detects RFCI input voltage equal to the low level of Schmitt-trigger. Reading T1CH and T1CL value by program and calculate the period of the pulse width. The First width length is longer than normal RFC pulse width length, because the discharge is from VDD to Schmitt-trigger low level. The normal RFC discharge is from Schmitt-trigger high level to low level.

✸ ***Note: The RFC first pulse width measurement only supports T1 high pulse width measurement.***

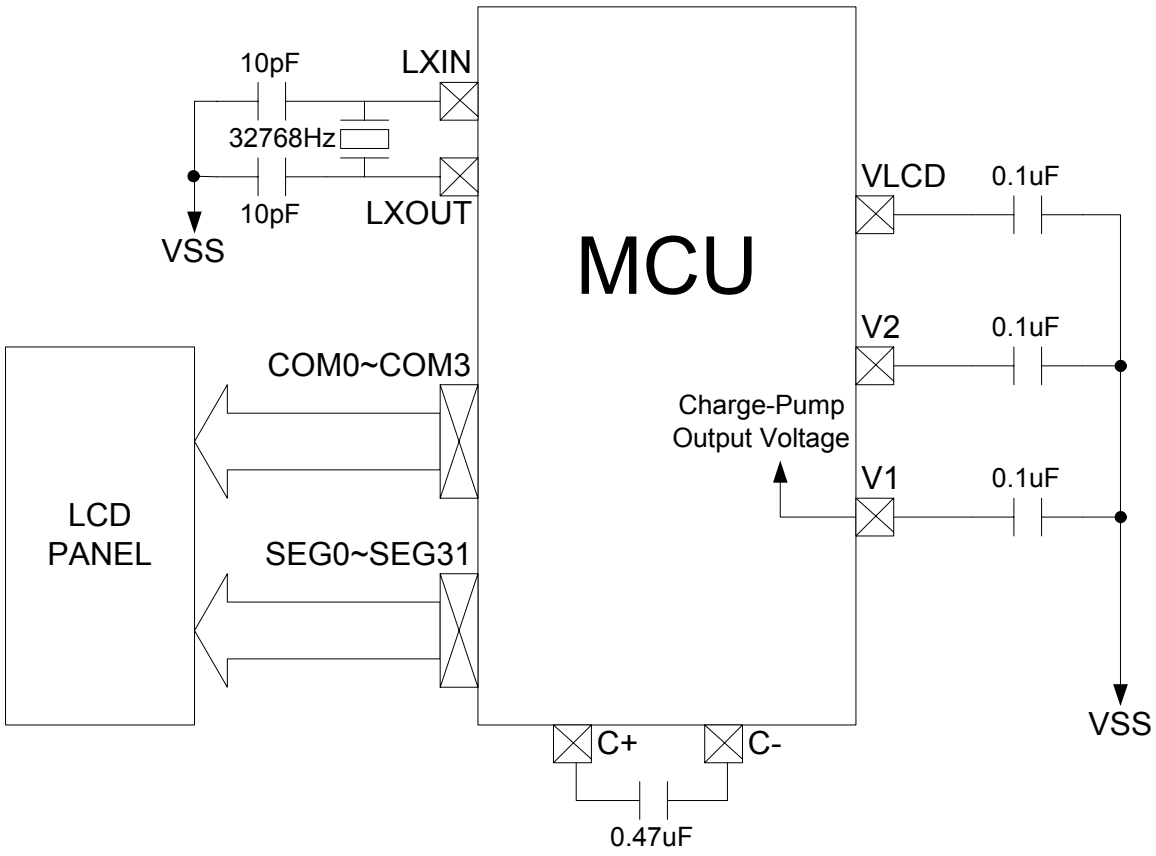
➢ **Example: Measure RFC first high pulse width using T1 pulse width measurement. Fhosc=4MHz, Fcpu=Fhosc/4=1mips. T1 base time is 256us. Pulse width is T1C buffer * T1 base time.**

```
; Charge the capacitor.
                B0BSET      P5.3
                B0BSET      P5M.3           ; Set P5.3 output high to charge the capacitor.

; Set T1 mode.
                CLR         T1M             ; Clear T1M register and set T1 rate is Fcpu/256.
                MOV         A, #0
                B0MOV       T1CL, A
                B0MOV       T1CH, A         ; Clear T1 counter buffers.

                B0BSET      FT1G1           ; Enable pulse width measurement.
                B0BCLR      FT1G0           ; Select high pulse direction.
                B0BCLR      FT1IRQ          ; Clear T1 interrupt request flag.

; Set RFC.
                CLR         RFC             ; Clear RFC register.
                B0BCLR      FRFCH0          ; Set RFCH[1:0]=00 to select RFC0 (RFC channel 0).
                B0BCLR      FRFCH1
                CLR         P5UR            ; Disable P5 pull-up resistor.
                MOV         A, #11110000B
                AND         P5M, A          ; Set P5.0~P5.3 to input mode.

                B0BSET      FRFCENB         ; Enable RFC function and the T1 pulse width trigger is RFC
                                            ; clock.
                B0BSET      FT1ENB          ; Enable T1 timer.

; Check T1IRQ=1.

Chk_T1IRQ:
                B0BTS1      FT1IRQ          ; Check T1IRQ=1.
                JMP         Chk_T1IRQ
                B0MOV       A, T1CH         ; Pulse width measure end.
                …
                B0MOV       A, T1CL
                …
```

9 4x32 LCD DRIVER

9.1 OVERVIEW

The MCU builds in 4x32 (4 commons and 32 segments, 128 dots) LCD driver. 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. LCD includes R type and C type (1/3 bias only) structures. R type is using external bias circuit to adjust LCD power and bias voltage. C type is using internal charge pump to adjust LCD power and bias voltage.



Basic C type LCD Circuit

9.2 LCD REGISTERS

0CBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM	-	-	-	-	RCLK	P3SEG	BIAS	LCDENB
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

Bit 3 **RCLK:** External low speed clock type selection bit.
0 = 32768Hz crystal/ceramic type connected to LXIN, LXOUT pins.
1 = RC type.

Bit 2 **P3SEG:** Seg24~Seg31 shared with P3.0~P3.7 control bit.
0 = Enable Seg24~Seg 31 pins.
1 = Enable P3.0~P3.7 pins.

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.

✸ ***Note: Segment 24~Segment 31 pins are shared with P3.0~P3.7. When these pins are used as Port3 general purpose I/O mode, the P3SEG bit of LCDM register must be set as "1".***

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VLCD	-	-	CPCK1	CPCK0	VLCD2	VLCD1	VLCD0	VLCDCP
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

Bit [5:4] **CPCK[1:0]:** VLCD charge-pump clock selection.

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

✸ ***Note:***

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

Bit [3:1] **VLCD[2:0]:** VLCD charge-pump voltage selection.

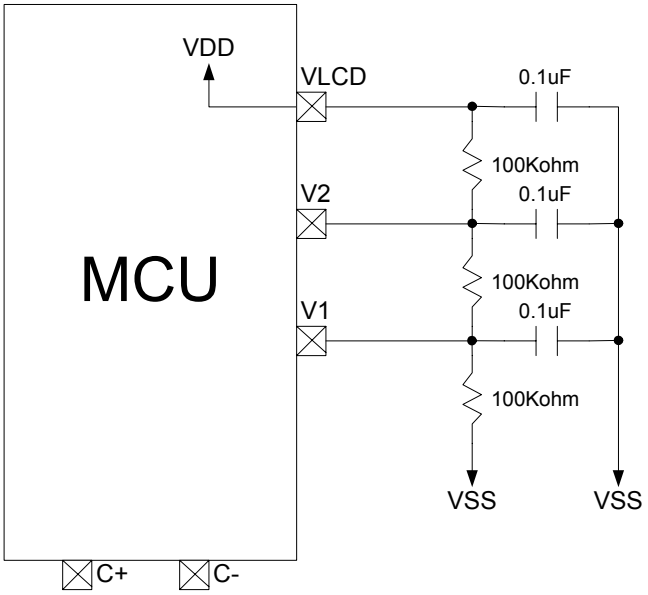
VLCD[2:0]	Charge-pump Output Voltage
000	0.9V
001	1.0V
010	1.1V
011	1.2V
100	1.3V
101	1.4V
110	1.5V
111	1.6V

Bit 0 **VLCDCP:** VLCD charge-pump control bit.
0 = Disable.
1 = Enable.

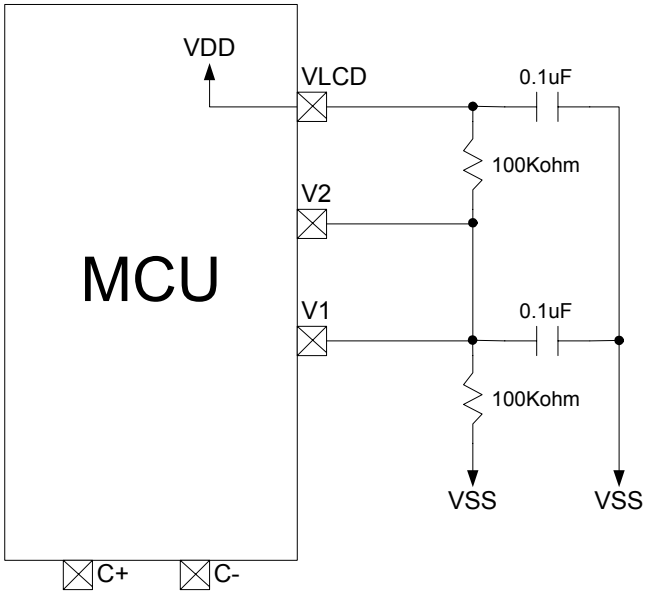
9.3 R-TYPE LCD MODE

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.

✷ ***Note:***

3. ***In R-type mode, C+/C- don't connect any devices.***
4. ***VLCD is internal VDD in R-type mode.***
5. ***1/2 bias, V2 connects to V1.***
6. ***The 0.1uF capacitors of VLCD/V1/V2 pins are necessary for power stable.***

➢ **Example : Set R-type LCD mode.**

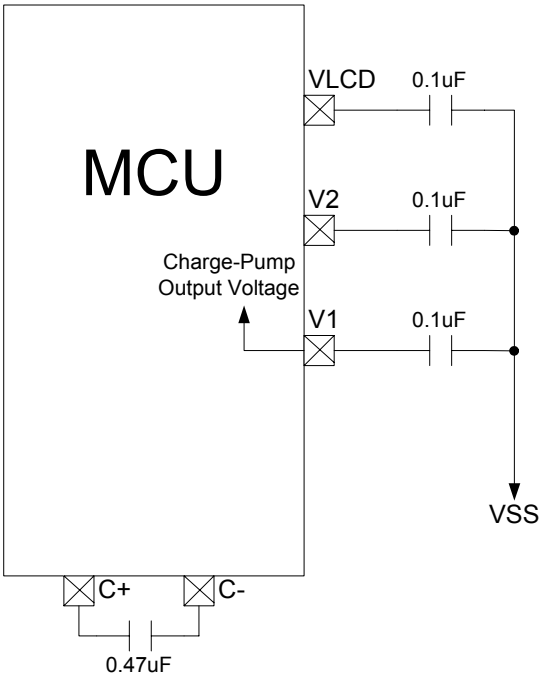
```
; Switch LCD mode to R-type mode.
        B0BCLR      FVLCDCP     ; Disable LCD charge-pump and LCD is switched to
                                ; R-type mode.
; Set LCD 32768Hz clock source type.
        B0BCLR      FRCLK       ; Crystal/ceramic type.
or
        B0BSET      FRCLK       ; RC type.
; Set LCD Bias.
        B0BCLR      FBIAS       ; 1/3 bias.
or
        B0BSET      FBIAS       ; 1/2 bias.
; Enable LCD.
        B0BSET      FLCDENB     ; Enable LCD driver..
```

9.4 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 controlled by VLCD[2:1] bits of VLCD register. V1 voltage is the charge-pump output voltage, and charge-pump output voltage measurement is from V1 pin. In C-type LCD mode, the VLCDCP bit of VLCD register must be “1”.

VLCD[2:0]	Charge-pump Voltage	V1	V2	VLCD
		1/3 BIAS	1/3 BIAS=2xV1	1/3 BIAS=3xV1
000	0.95V	0.95V	1.9V	2.85V
001	1.05V	1.05V	2.1V	3.15V
010	1.15V	1.15V	2.3V	3.45V
011	1.25V	1.25V	2.5V	3.75V
100	1.35V	1.35V	2.7V	4.05V
101	1.45V	1.45V	2.9V	4.35V
110	1.55V	1.55V	3.1V	4.65V
111	1.65V	1.65V	3.3V	4.95V

NOTE：VDD=5V、temperature=25°C、error=-10%~+10%



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

✷ ***Note:***

1. ***In C-type mode only support 1/3 bias***
2. ***In C-type mode, connect a 0.1uF or 0.47uF capacitor between C+ and C- pins. Users can adjust the capacitor value depend on the LCD panel size.***
3. ***V1 voltage is charge-pump output voltage.***
4. ***The 0.1uF capacitors of VLCD/V1/V2 pins are necessary for power stable. Users can adjust the capacitor value depend on the LCD panel size.***

➢ **Example : Set C-type LCD mode.**

```
; Set LCD 32768Hz clock source type.
        B0BCLR      FRCLK       ; Crystal/ceramic type.
or
        B0BSET      FRCLK       ; RC type.
; Set LCD Bias.
        B0BCLR      FBIAS       ; 1/3 bias.
or
        B0BSET      FBIAS       ; 1/2 bias.
; Enable LCD.
        B0BSET      FLCDENB     ; Enable LCD driver..

; Set charge-pump clock and output voltage level.
        MOV         A, #00xxxxx0h   ; Charge-pump clock controlled by CPCK[1:0]
                                    ; (00b~11b).
        B0MOV       VLCD, A         ; Charge-pump output voltage controlled by VLCD[2:0]
                                    ; (000b~111b).

; Switch LCD mode to C-type mode.
        B0BSET      FVLCDCP     ; Enable LCD charge-pump and LCD is switched to
                                ; C-type mode.

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

9.5 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.
…
…
```

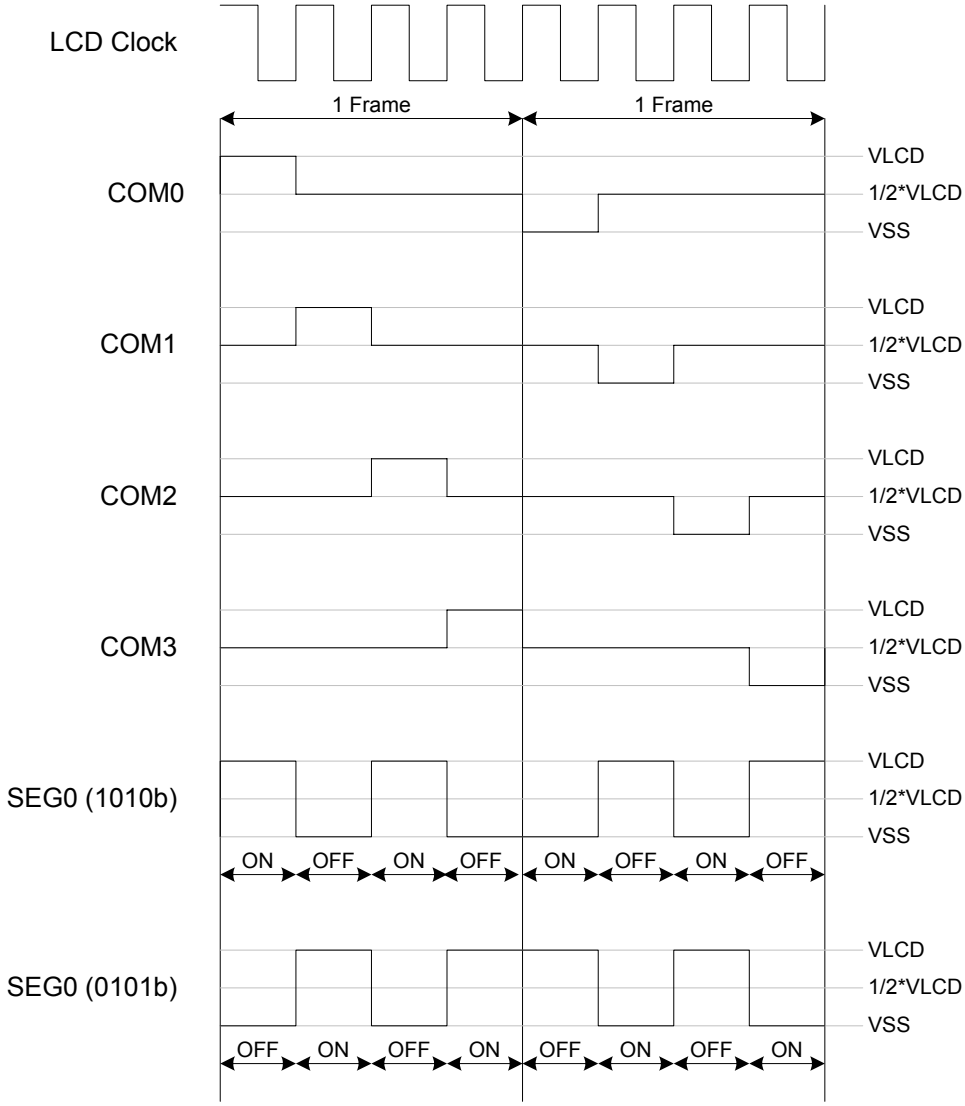
➢ **Example : Set LCD RAM data by directly addressing in bank 15.**

```
MOV     A, #15          ; 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
```

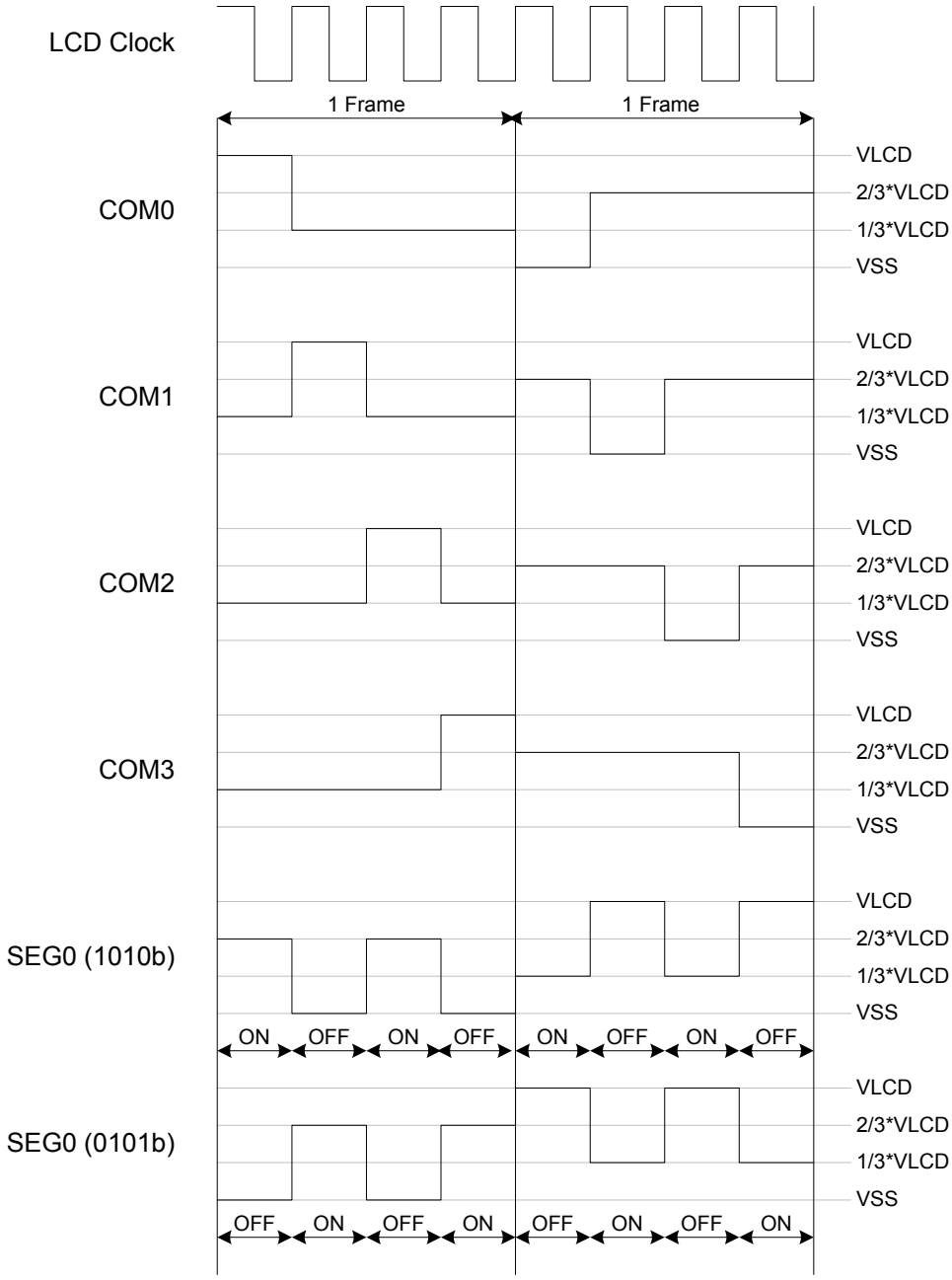
✸ ***Note: Access RAM data of bank 0 (system registers and user define RAM 0x0000~0x007F) is using "B0xxx" instructions.***

9.6 LCD WAVEFORM

1/2 bias, 1/4 duty.



1/3 bias, 1/4 duty.



10 INSTRUCTION TABLE

Field	Mnemonic		Description	C	DC	Z	Cycle
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	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
ARITHMETIC	ADC	A,M	A ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
	ADC	M,A	M ← A + M + C, if occur carry, then C=1, else C=0	√	√	√	1+N
	ADD	A,M	A ← A + M, if occur carry, then C=1, else C=0	√	√	√	1
	ADD	M,A	M ← A + M, if occur carry, then C=1, else C=0	√	√	√	1+N
	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
	ADD	A,I	A ← A + I, if occur carry, then C=1, else C=0	√	√	√	1
	SBC	A,M	A ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
	SUB	A,M	A ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1
	SUB	M,A	M ← A - M, if occur borrow, then C=0, else C=1	√	√	√	1+N
	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
LOGIC	AND	A,M	A ← A and M	-	-	√	1
	AND	M,A	M ← A and M	-	-	√	1+N
	AND	A,I	A ← A and I	-	-	√	1
	OR	A,M	A ← A or M	-	-	√	1
	OR	M,A	M ← A or M	-	-	√	1+N
	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
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1+N
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1+N
	CLR	M	M ← 0	-	-	-	1
	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
BRANCH	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
	CMPRS	A,M	ZF,C ← A – M, If A = M, then skip next instruction	√	-	√	1 + S
	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1+ S
	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1+ S
	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1+N+S
	BTS0	M.b	If M.b = 0, then skip next instruction	-	-	-	1 + S
	BTS1	M.b	If M.b = 1, then skip next instruction	-	-	-	1 + S
	B0BTS0	M.b	If M(bank 0).b = 0, then skip next instruction	-	-	-	1 + S
	B0BTS1	M.b	If M(bank 0).b = 1, then skip next instruction	-	-	-	1 + S
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
MISC	RET		PC ← Stack	-	-	-	2
	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
	PUSH		To push ACC and PFLAG (except NT0, NPD bit) into buffers.	-	-	-	1
	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".

11 ELECTRICAL CHARACTERISTIC

11.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2308Q ..... 0°C ~ + 70°C
SN8P2308QD..... –40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

11.2 ELECTRICAL 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, Vpp = Vdd, 25°C		2.4	5.0	5.5	V
		Normal mode, Vpp = Vdd, -40°C~85°C		2.5	5.0	5.5	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
Internal POR	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		0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd, T=25°C		-	-	2	uA
		Vin = Vdd, T=-40°C~85°C		-	-	5	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	180	
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
All Port source current	IoH	Vop = Vdd – 0.5V(source)		8	12	-	mA
sink current	IoL	Vop = Vss + 0.5V(sink)		8	15	-	
INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current (I/O No loading, Code option: Watchdog: disable LVD: LVD_L)	Idd1	normal Mode (Fcpu = Fosc/4)	Vdd= 5V, 4MHz	-	2.5	5	mA
			Vdd= 3V, 4MHz	-	1	2	mA
	Idd2	Slow Mode (Stop High clock, External 32768Hz crystal in LXIN, LXOUT)	Vdd= 5V, 32768Hz (LCD Off)	-	25	50	uA
			Vdd= 3V, 32768Hz (LCD Off)	-	15	30	uA
	Idd3	Sleep Mode	Vdd= 5V, 25°C	-	1	2	uA
			Vdd= 3V , 25°C	-	0.8	1.6	uA
			Vdd= 5V, -40°C~85°C	-	10	21	uA
			Vdd= 3V, -40°C~85°C	-	10	21	uA
	Idd4	High clock Green Mode	Vdd= 5V, 4MHz	-	0.60	1.2	mA
			Vdd= 3V, 4MHz	-	0.25	0.5	mA
	Idd5	Low clock Green Mode (Stop High clock, External 32768Hz crystal in LXIN, LXOUT, CPCK[1:0] = "11", LCD No loading, C+, C-: 0.1uF)	Vdd= 5V, 32768Hz (LCD Off)	-	15	30	uA
			Vdd= 3V, 32768Hz (LCD Off)	-	5	10	uA
			Vdd= 5V, 32768Hz (LCD On)	-	20	40	uA
			Vdd= 3V, 32768Hz (LCD On)	-	10	20	uA
LVD Voltage	Vdet0	Low voltage reset level.		1.7	2.0	2.3	V
	Vdet1	Low voltage reset level. Fcpu = 1 MHz. Low voltage indicator level. Fcpu = 1 MHz.		2.0	2.4	3	V
	Vdet2	Low voltage indicator level. Fcpu = 1 MHz		2.7	3.6	4.5	V

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

12 DEVELOPMENT TOOL VERSION

12.1 ICE (In Circuit Emulation)

- **SN8ICE 2K ICE:** Full function emulates SN8P2308.

✸ ***SN8ICE 2K ICE emulation notice:***
  a. Operation voltage of ICE: 3.0V ~ 5.0V.
  b. Recommend maximum emulation speed at 5V: 8 MIPS (e.g. 16Mhz Crystal and Fcpu = Fhosc/2).
  c. Use SN8P2308 EV-KIT to emulate LVD, LCD and RFC.

✸ ***Note: S8KD-2 ICE doesn't support SN8P2308 emulation.***

12.2 OTP WRITER

- **Easy Writer V1.0:** OTP programming is controlled by ICE without firmware upgrade suffers. Please refer easy writer user manual for detailed information.
- **MP-Easy Writer V1.0:** Stand-alone operation to support SN8P2308 mass production.
- **Writer V3.0:** Stand-alone operation to support SN8P2308 mass production.

12.3 IDE

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

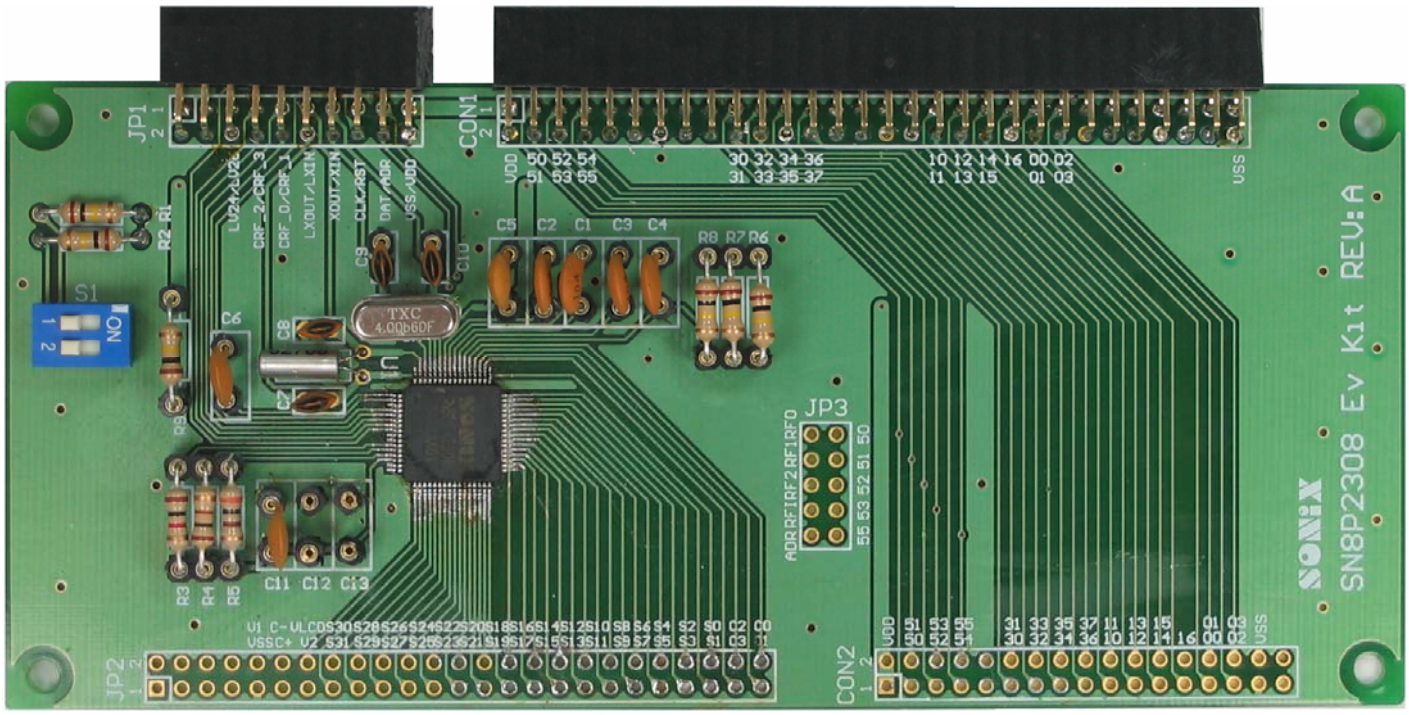
- M2IDE_V107 or later.
  - **Support SN8ICE_2K ICE**
  - **Support Assembly and ICE debug function**
  - **Support Easy Writer and MP-Easy Writer**

➢ ***Note: SN8IDE_1.99X don't support SN8P2308 emulation.***

12.4 SN8P2308 EV KIT

12.4.1 PCB DESCRIPTION

SONIX provides SN8P308 EV Kit for function emulation. For SN8P2308 ICE emulation, the EV kit provides RFC, LCD and LVD 2.4V/3.6V selection circuits.



CON1: I/O port. Connect to SN8ICE 2K CON1.

CON2: I/O port. Connect to target board.

JP1: LVD 2.4V, 3.6V input pins, LCD and RFC control pins. Connect to SN8ICE 2K JP6.

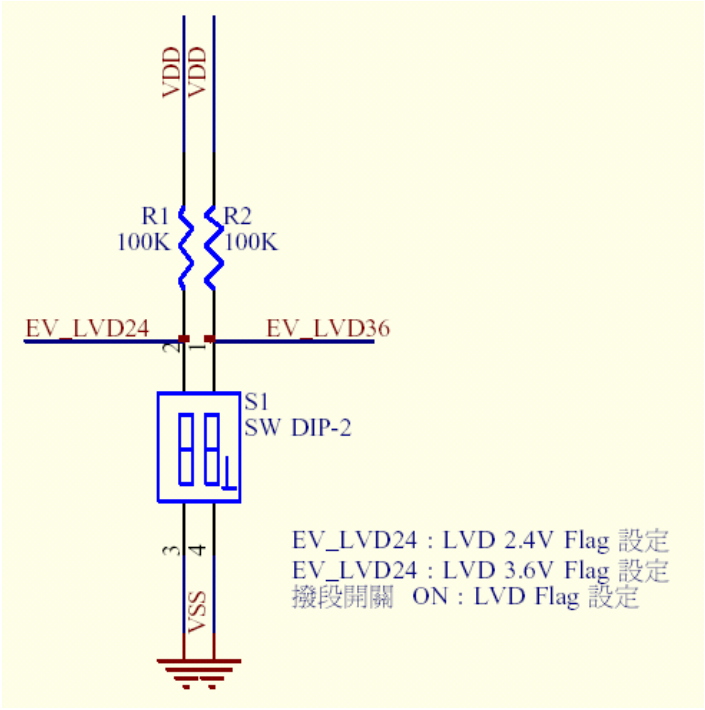
JP2: LCD COM0~COM3, SEG0~SEG31 output pins. Connect to LCD panel.

JP3: Connect the P5.0 ~ P53 of CON2 with RF0 ~ RFI of SN8P2308 EV chip.

C6, R9: SN8P2308 EV chip reset circuit. C6=0,1uF, R9=100K ohm.

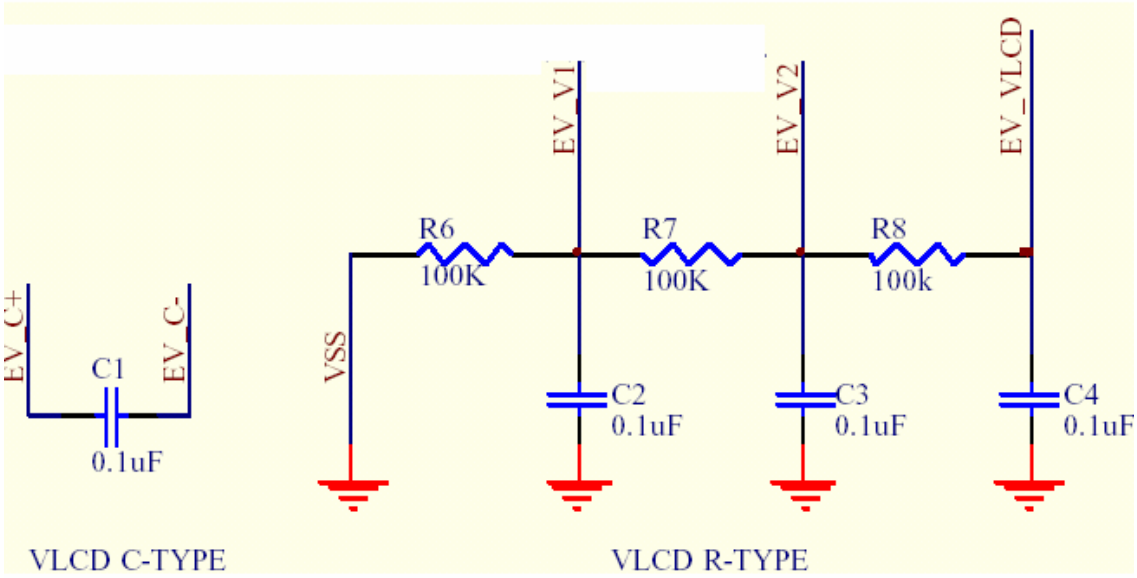
S2: LVD 2.4V/3.6V control switch. To emulate LVD 2.4V flag/reset function and LVD 3.6V flag function.

Switch No.	On	Off
1	LVD 2.4V Active	LVD 2.4V Inactive
2	LVD 3.6V Active	LVD 3.6V Inactive



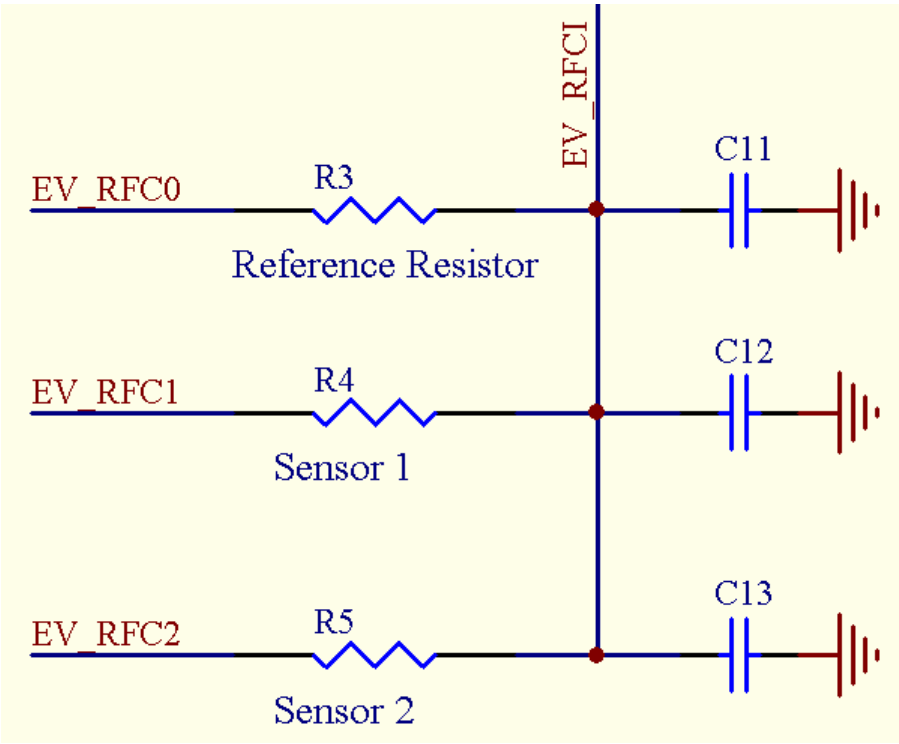
C1~C4, R6~R8: LCD circuit devices. Please refer LCD section to adjust circuit for different LCD modes.

Note: Set port 3 as input mode (P3M = "0") if LCD segment 24 ~ segment 31 is enabled (P3SEG = "0").



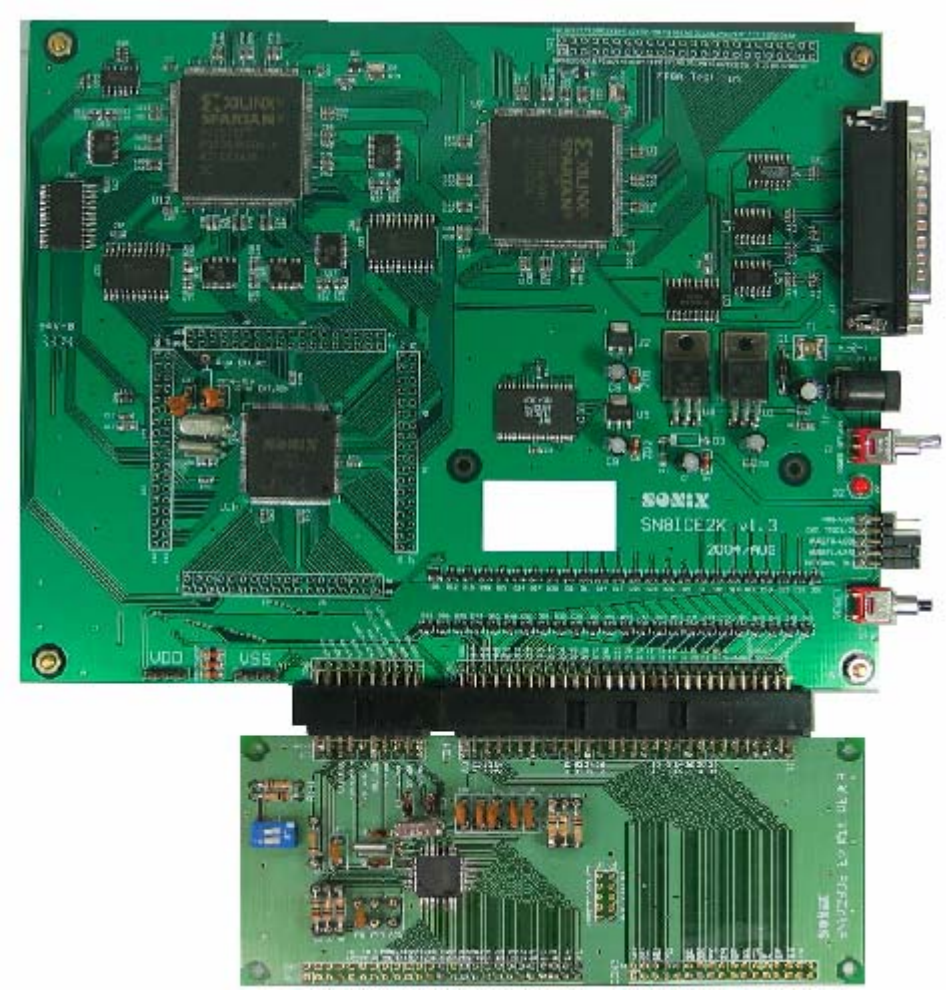
RFC circuit emulation:

- **R3:** RFC channel 0 for “Reference Resister”
- **R4:** RFC channel 1 for “sensor 1”
- **R5:** RFC channel 2 for “sensor 2”
- **C11 ~ C13:** Put a Capacitor for RFC oscillator circuit
- **Short JP3 can connect the RF0, RF1, RF2 and RFI of SN8P2308 EV-Chip to CON2 for target board.**
- **Disable Port 5 pull-up in RFC circuit emulation**



12.4.2 SN8P2308 EV KIT CONNECT TO SN8ICE 2K

The connection from SN8P2308 EV KIT to SN8ICE 2K is as following.

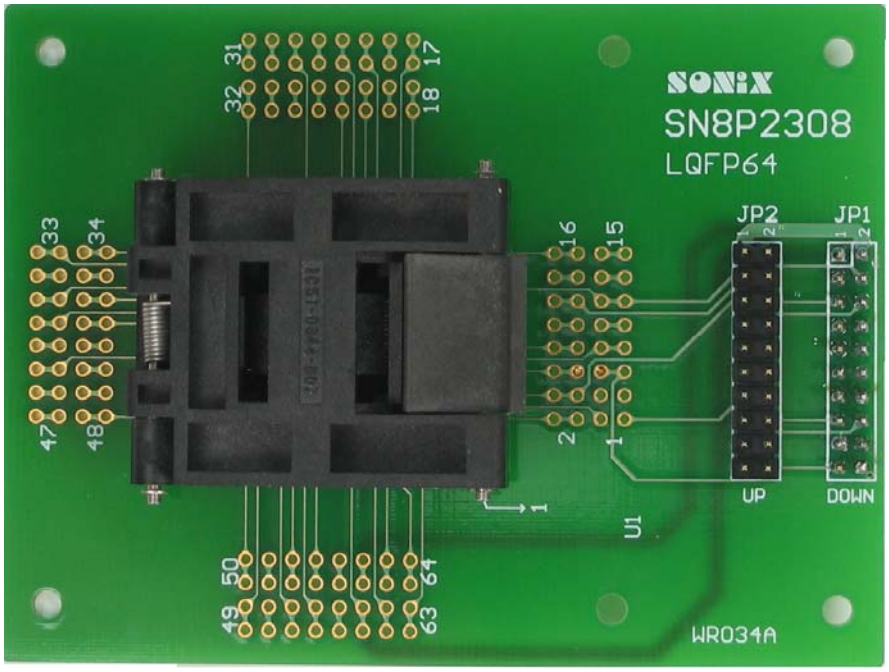


12.5 TRANSITION BOARD FOR OTP PROGRAMMING

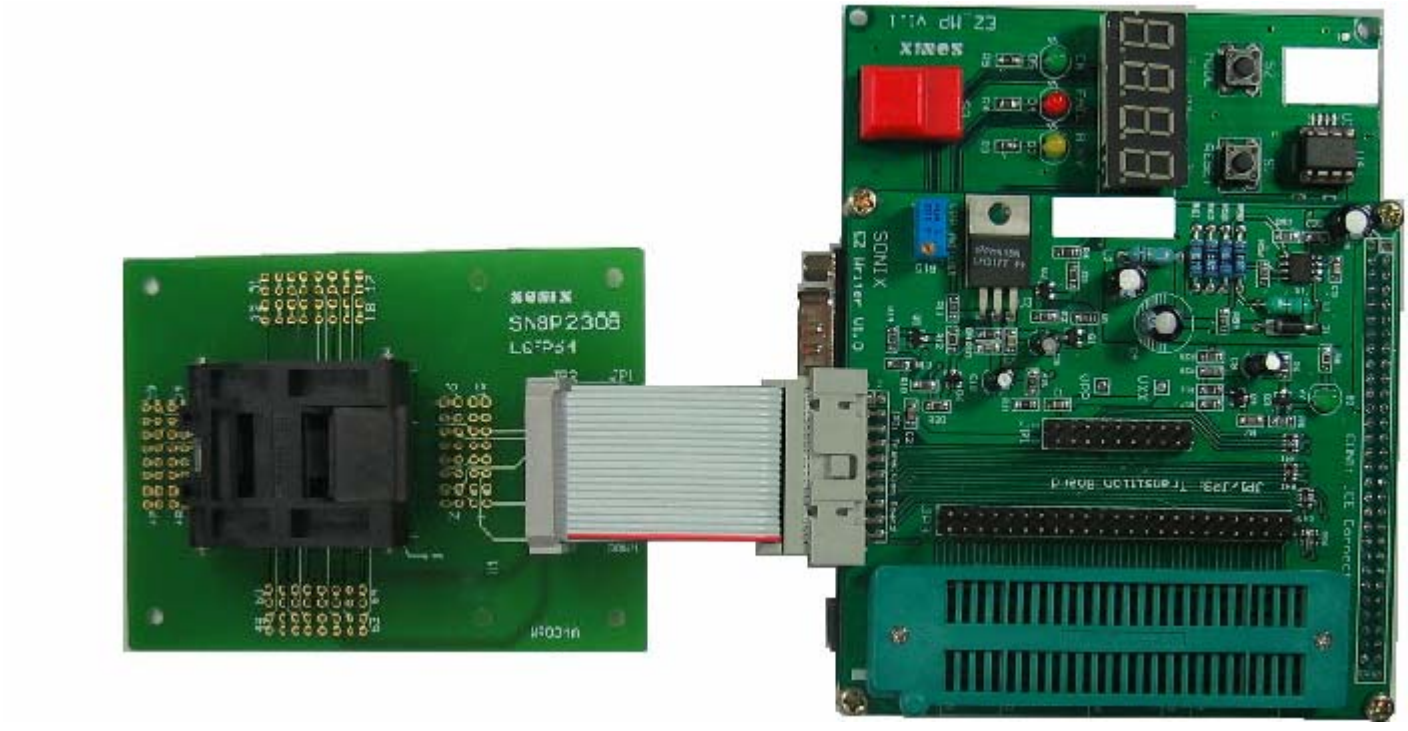
12.5.1 SN8P2308 TRANSITION BOARD

SN8P2308 transition board is for SN8P2308 OTP programming by LQFP 64 pin socket connection.

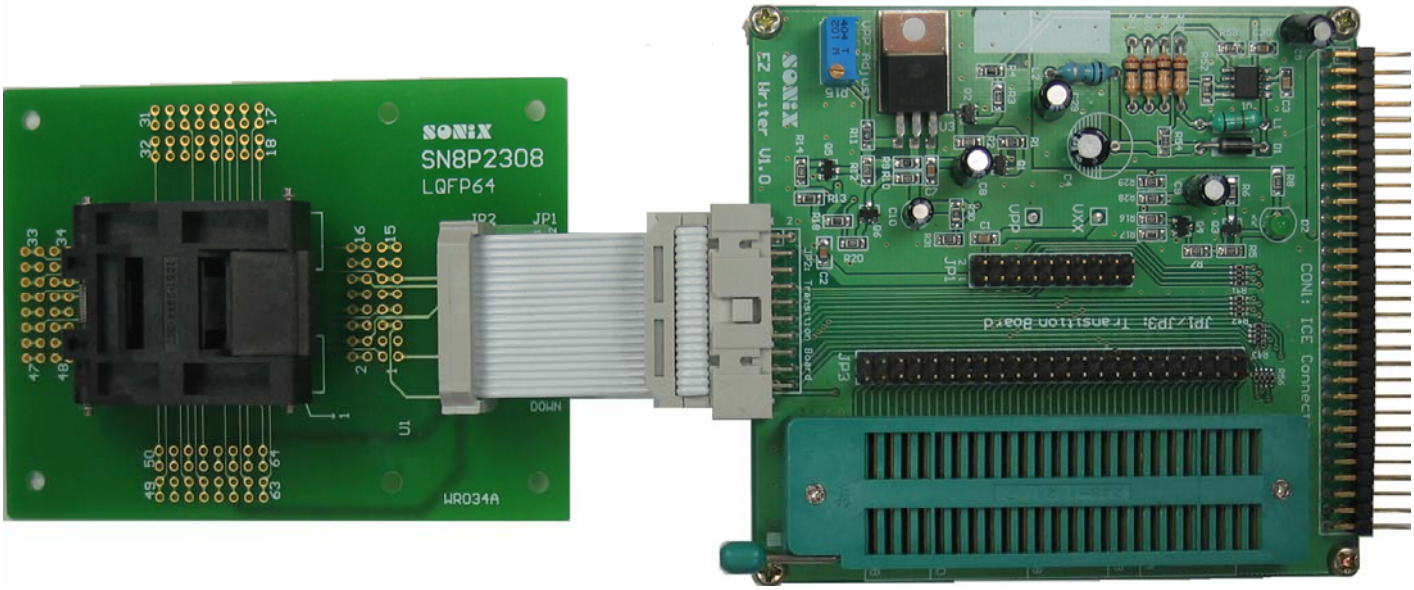
- **JP1/JP2:** Connect to EZ writer, EZ_MP writer or writer V3.0.
- **U1:** LQFP 64 pin socket.



12.5.2 CONNECT TO MP-Easy WRITER



12.5.3 CONNECT TO Easy WRITER



➢ ***Note: Connect Easy Writer to ICE must be through an 60-pin cable***

12.6 OTP PROGRAMMING PIN

12.6.1 EASY WRITER TRANSITION BOARD SOCKET PIN ASSIGNMENT:

Easy Writer JP1/JP2

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

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

Easy Writer JP3 (Mapping to 48-pin text tool)

DIP1	1	48	DIP48
DIP2	2	47	DIP47
DIP3	3	46	DIP46
DIP4	4	45	DIP45
DIP5	5	44	DIP44
DIP6	6	43	DIP43
DIP7	7	42	DIP42
DIP8	8	41	DIP41
DIP9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP38
DIP13	13	36	DIP36
DIP14	14	35	DIP35
DIP15	15	34	DIP34
DIP16	16	33	DIP33
DIP17	17	32	DIP32
DIP18	18	31	DIP31
DIP19	19	30	DIP30
DIP20	20	29	DIP29
DIP21	21	28	DIP28
DIP22	22	27	DIP27
DIP23	23	26	DIP26
DIP24	24	25	DIP25

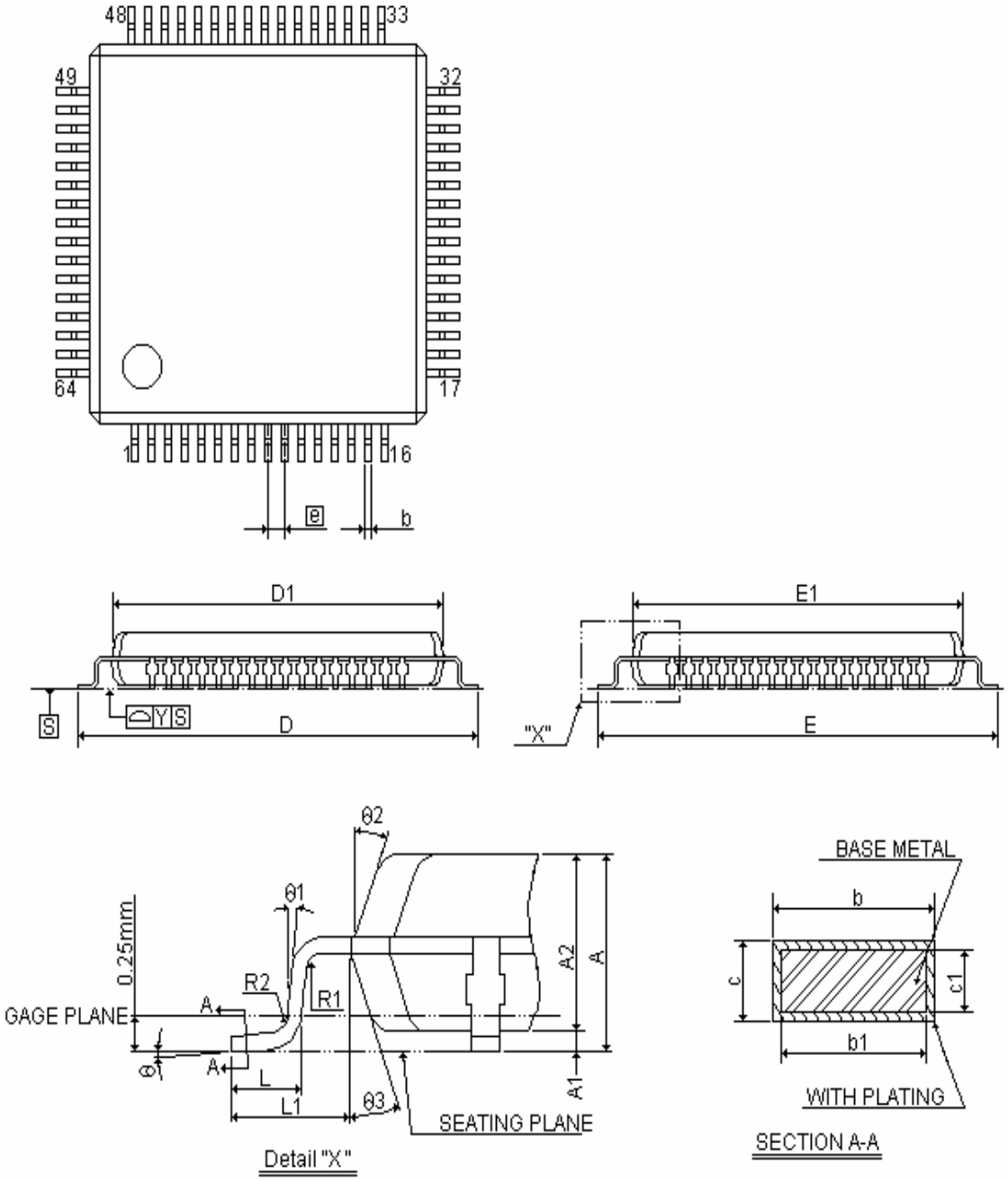
JP3 for MP transition board

12.6.2 SN8P2308 PROGRAMMING PIN MAPPING:

Programming Information of SN8P2300 Series									
Chip Name		SN8P2308Q							
EZ Writer Connector		OTP IC / JP3 Pin Assigment							
Number	Name	Number	Pin						
1	VDD	59	VDD						
2	GND	60	VSS						
3	CLK	11	P5.0						
4	CE	-	-						
5	PGM	4	P1.0						
6	OE	12	P5.1						
7	D1	-	-						
8	D0	-	-						
9	D3	-	-						
10	D2	-	-						
11	D5	-	-						
12	D4	-	-						
13	D7	-	-						
14	D6	-	-						
15	VDD	-	-						
16	VPP	1	RST						
17	HLS	-	-						
18	RST	-	-						
19	-	-	-						
20	ALSB/PDB	5	P1.1						

13 PACKAGE INFORMATION

13.1 LQFP 64 PIN



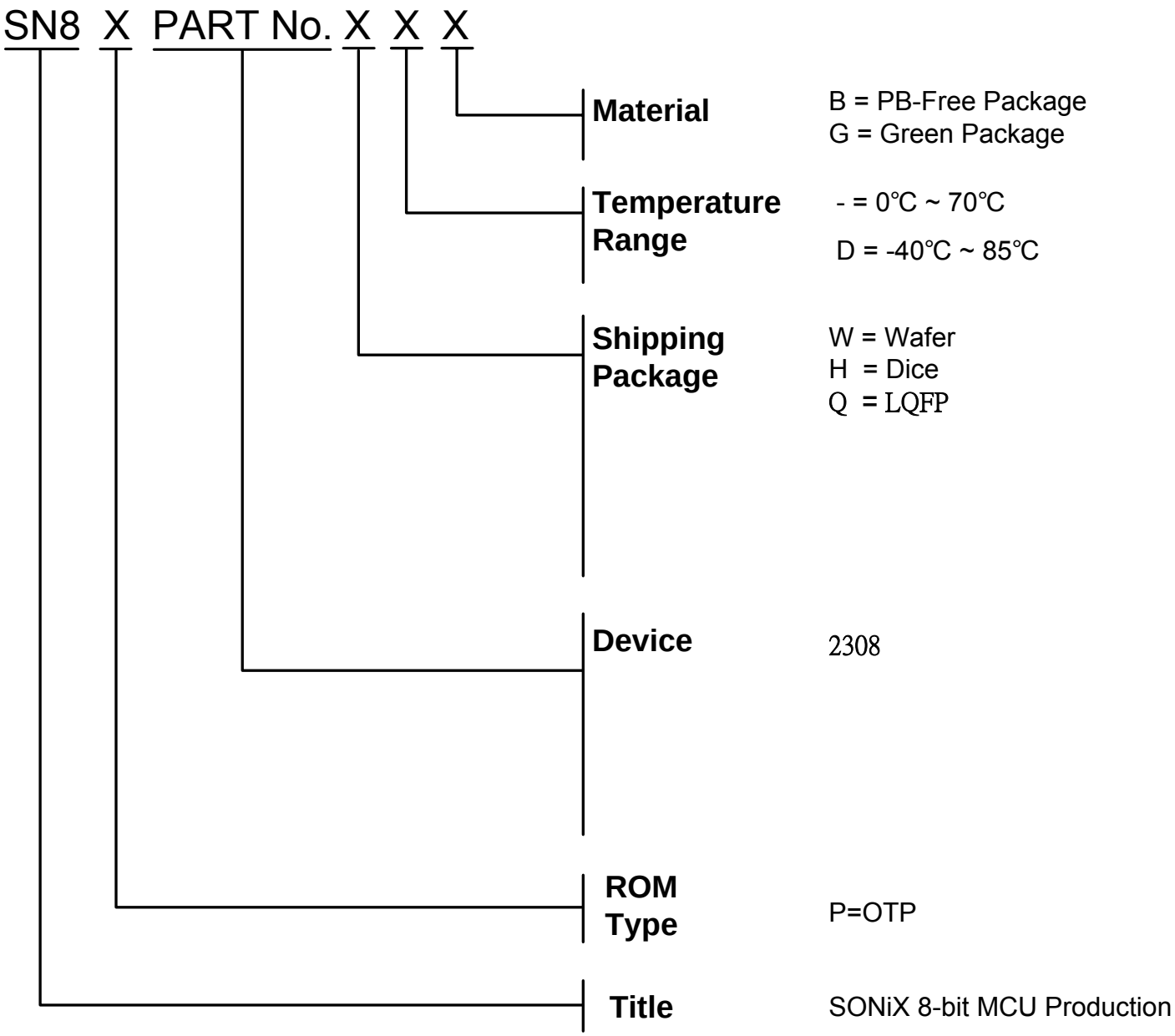
SYMBLE	DIMENSION (MM)			DIMENSION (MIL)		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A			1.60			63
A1	0.05	1.40	0.15	2	55	6
A2	1.36	0.22	1.45	35	9	57
b	0.17	0.22	0.27	7	8	11
b1	0.17		0.23	7		12
c	0.09		0.20	4		8
c1	0.09		0.16	4		6
D	11.75	12.00	12.25	463	473	483
D1	9.95	10.00	10.05	392	394	396
E	11.75	12.00	12.25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

14 Marking Definition

14.1 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.

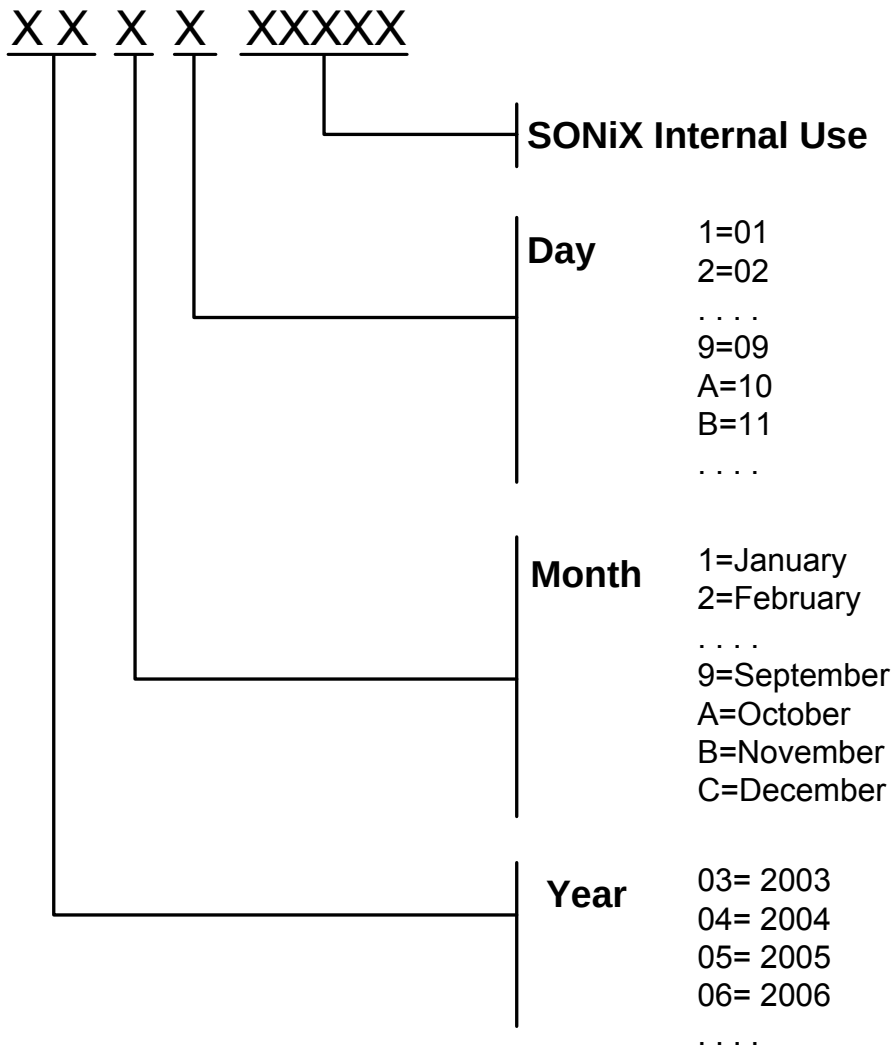
14.2 MARKING INDETIFICATION SYSTEM



14.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P2308QB	OTP	2308	LQFP	0℃~70℃	PB-Free Package
SN8P2308QDB	OTP	2308	LQFP	-40℃~85℃	PB-Free Package

14.4 DATECODE SYSTEM



Main Office:
Address: 9F, NO. 8, Hsien Cheng 5th St, Chupei City, Hsinchu, Taiwan R.O.C.
Tel: 886-3-551 0520
Fax: 886-3-551 0523
Taipei Office:
Address: 15F-2, NO. 171, Song Ted Road, Taipei, Taiwan R.O.C.
Tel: 886-2-2759 1980
Fax: 886-2-2759 8180
Hong Kong Office:
Address: Flat 3 9/F Energy Plaza 92 Granville Road, Tsimshatsui East Kowloon.
Tel: 852-2723 8086
Fax: 852-2723 9179
Technical Support by Email:
Sn8fae@sonix.com.tw