SN8P2839 Series

USER'S MANUAL

Version 1.4

SN8P2839

SONiX 8-Bit Micro-Controller

AMENDENT HISTORY

Version	Date	Description
VER 1.0	2011.05.17	Modify UART Baud Rate table.
VER 1.1	2011.08.08	Modify Package TYPE.
VER 1.2	2011.09.27	Add Jmp/Call description.
VER 1.3	2012.02.15	Modify electrical characteristic.
VER 1.4	2012.07.16	Modify Function description.

Table of Content

1 PRODUCT OVERVIEW

1.1 FEATURES

- **Memory configuration**
  OTP ROM size: 24K * 16 bits (ROM Page 0/1)
  RAM size: 2048 * 8 bits (bank 0/1/2/3/4/5/6/7/8)
  16-levels stack buffer
  LCD RAM size: 4*36 bits(bank 15)

- **I/O pin configuration (Total 26 pins)**
  Bi-directional: P0,P1,P2,P4,P5
  Programmable open-drain: P10~P13,P50~P52
  Wakeup:
  P0,P1 level change trigger.
  P0,P1 Green mode ,Sleep mode wake up.
  Pull-up resisters:P0, P1, P2, P4, P5
  P4 pins shared with ADC inputs.
  P12,P13 shared with UART function.
  (P12/UTX,P13/URX)
  Event counter input:
  P00 is TC0 event counter input.
  P01 is TC1 event counter input.
  P15 is T1 event counter input.
  P11 is T2 event counter input.
  P14 is T3 event counter input.

- **Interrupt sources**
  9 internal interrupts:
  T0,T1,T2,T3,TC0,TC1,ADC,UART,SIO
  eight external interrupts:P0,P1
- **Timer System**
  T0: 8-bit timer with green mode wakeup function
  T1: 8-bit timer with event counter function
  T2:16-bit Timer with Capture Timer Function.
  T3:16-bit Timer with Capture Timer Function
  TC0:Auto-reload timer/PWM0/Buzzer output
  TC1:Auto-reload timer/PWM1/Buzzer output
  Real Time Clock (RTC): 0.5 second
  On chip watchdog timer

- **Powerful instructions**
  Four clocks per instruction cycle
  All instructions are one word length.
  Most of instructions are one cycle only
  Maximum instruction cycle is “2”.
  JMP instruction jumps to all ROM area.
  All ROM area lookup table function (MOVC)
  Hardware multiplier (MUL)

- **Single power supply:**
  Built-in Triple Charge-Pump @ VBAT=2.0V~3.5V
  VBAT=2.0V~6.0V
  DVDD=2.4V~5.5V
  VBAT Maximum Driving current 10mA @2.0V
- **Three Built-in Regulator**
  Digital Regulator: **DVDDR = 4.5V**
  Analog Regulator: **AVDDR1=3.8V, AVDDR2=2.5V**
- **Two-sets PGIA**
  Programmable Gain Instrumentation Amplifier
  Gain 1: 16/32/64/128
  Gain 2: 1.3 ~ 2.5
- **1 channel UART Interface.**
- **1 channel 12-bit DAC.**
- **Two-sets Operational Amplifier**
- **One channel SIO interface.**
- **Built-in Two Battery Measurement**
  1/2 VBAT, 1/4 VBAT optional
- **LCD driver:**
  1/4 duty, 1/3 bias
  4 common * 36 segment
  (Built-in internal resistor:10k,50k,100k,200k)

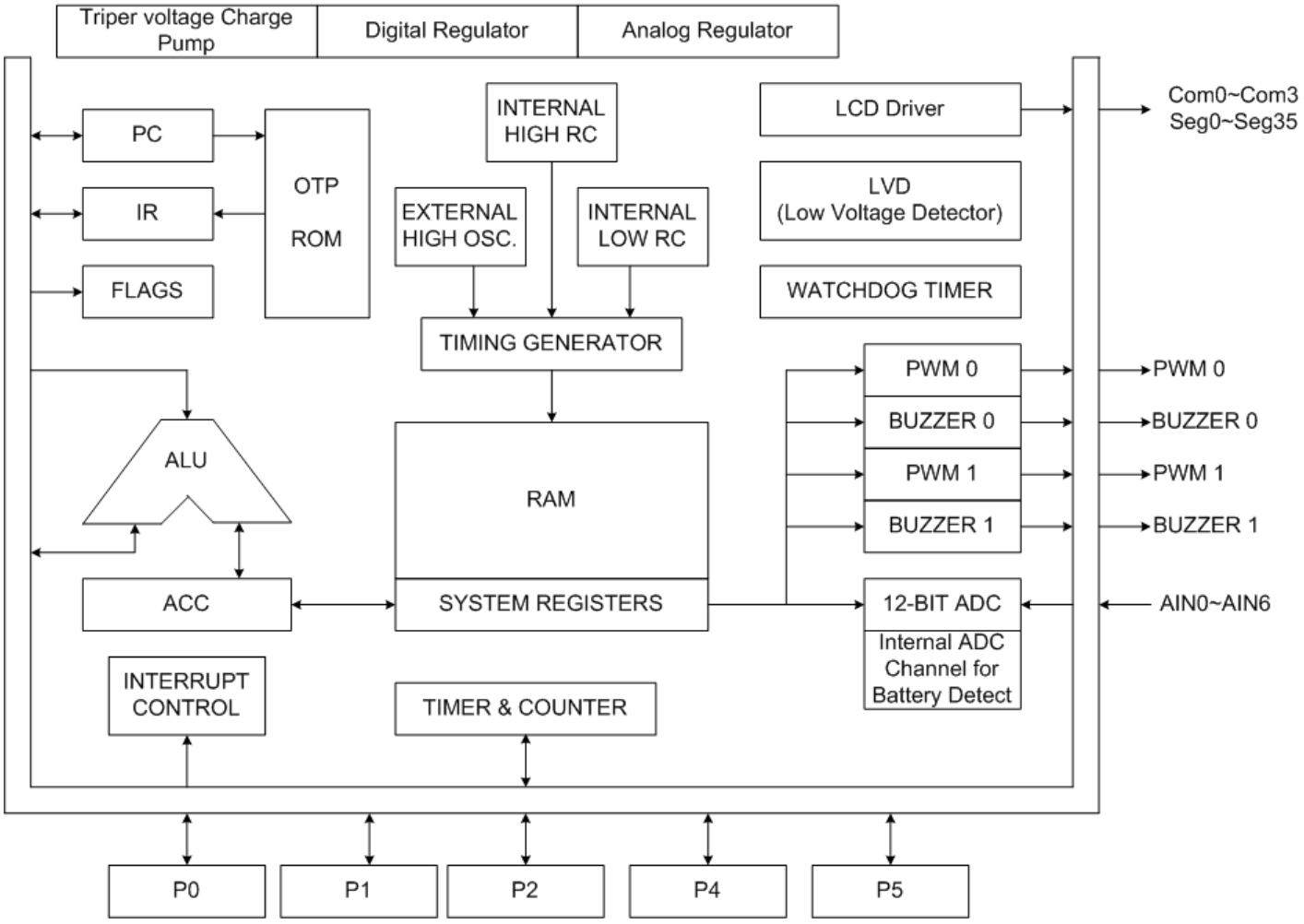
- **Build-in DVDD LVD=2.2V / 2.4V / 3.6V**
- **7+1 channels 12-bit ADC**
- Built-in Internal High Clock 16MHZ.

- **System clocks and Operating modes**
  Internal high clock: RC type up to 16MHz
  Internal low clock: RC type 16KHz(3V), 32KHz(5V)
  External high clock: Crystal type 32KHZ up to 16MHZ.
  External high clock: RC type up to 10MHZ.
  External low clokc: Crystal type 32KHZ.
  Normal mode: Both high and low clock active
  Slow mode: Low clock only
  Sleep mode: Both high and low clock stop
  Green mode: Periodical wakeup by T0/TC0 timer
- **Package**
  LQFP100

☞ Features Selection Table

CHIP	ROM	RAM	Stack	LCD	Timer						I/O	ADC ch	DAC	PWM Buzzer	SIO	UART	Wakeup Pin no.	Package
					T0	T1	T2	T3	TC0	TC1								
SN8P2839F	24K*16	2048*8	16	4*36	V	V	V	V	V	V	26	7+1	1	2	1	1	8	LQFP100

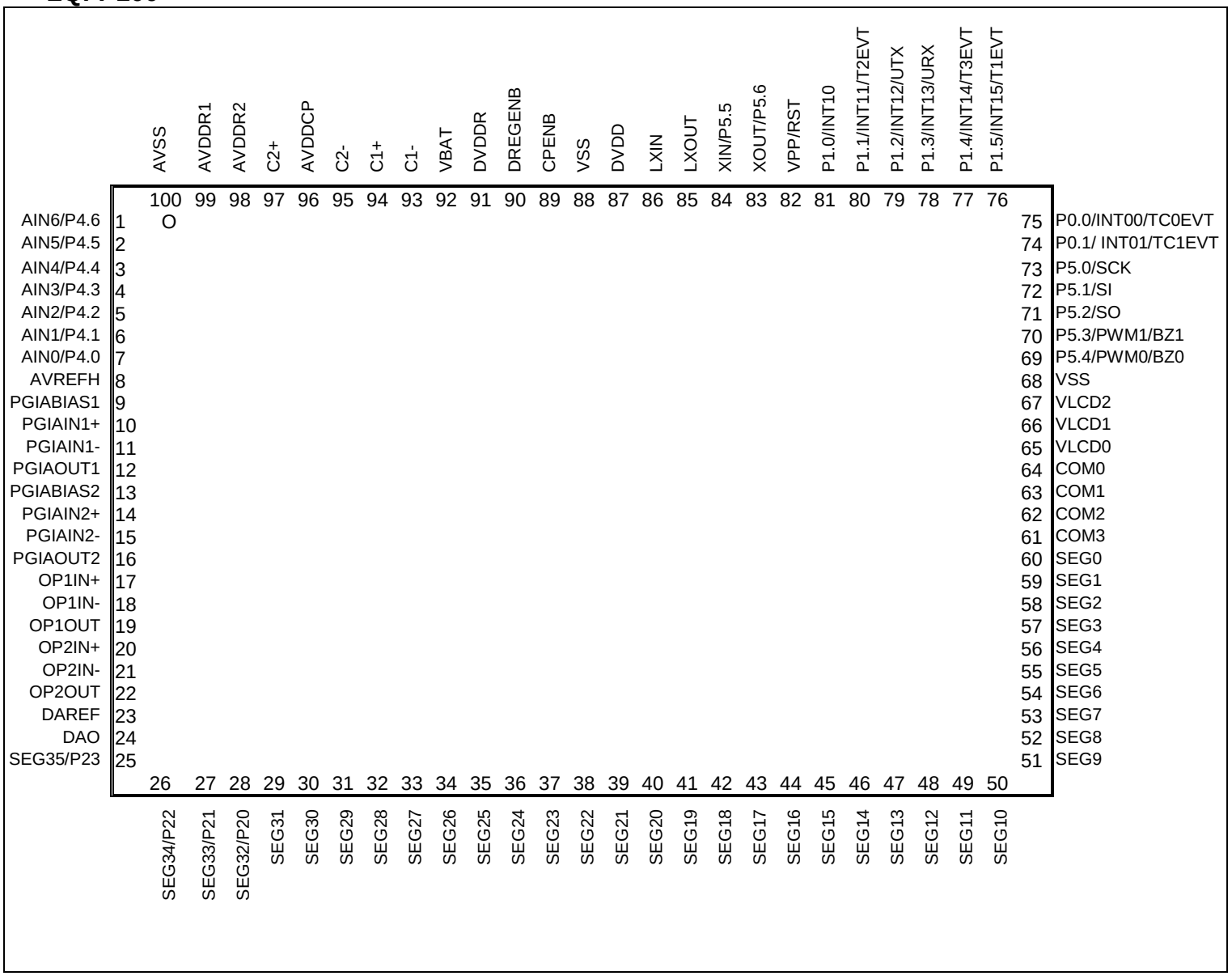
1.2 SYSTEM BLOCK DIAGRAM



1.3 PIN ASSIGNMENT

SN8P2839F

LQFP100



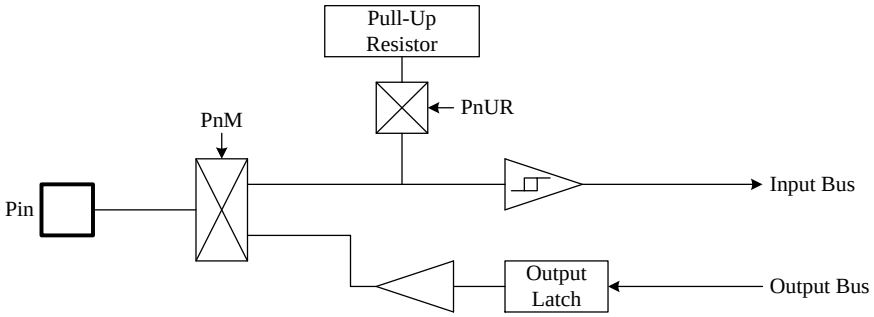
1.4 PIN DESCRIPTIONS

PIN NAME		DESCRIPTION
VBAT, VSS	P	Power supply input pins for Charge pump.
DVDDR	P	Digital Regulator Output DVDDR=4.5V
AVDDR1	P	Analog Regulator Output AVDDR=3.8V.
AVDDR2	P	Analog Regulator Output AVDDR=2.5V.
DVDD	P	Digital Function Power Input
AGND	P	Analog Ground.
AVDDCP	P	Tripler Charge Pump Voltage output. (Connect a 2.2uF or higher capacitor to ground) Maximum output of 15mA@2.0V(Sum of DVDDR and AVDDR)
CPENB	I	Charge pump Enable control pin. (Low Enable)
DREGENB	I	Digital Regulator Enable control pin. (Low Enable)
RST	I	System reset input pin. Schmitt trigger structure, active “low”, normal stay to “high”.
XIN, XOUT	I, O	External oscillator pins. RC mode from XIN.
LXIN, LXOUT	I, O	Low speed (32768 Hz) oscillator pins. RC mode from LXIN.
P00/INT00/TC0EVT	I	Port 00 bi-direction pin / Built-in pull-up resistances. Interrupt INT00/ Wake up from Sleep and Green mode. TC0 event counter input.
P01/INT01/TC1EVT	I	Port 01 bi-direction pin / Built-in pull-up resistances. Interrupt INT01/ Wake up from Sleep and Green mode. TC1 event counter input.
P10/INT10	I/O	Port 10 bi-direction pin/ Built-in pull-up resistances. P10 build-in Open Drain function. Interrupt INT10/ Wake up from Sleep and Green mode.
P11/INT11/T2EVT	I/O	Port 11 bi-direction pin/ Built-in pull-up resistances. P11 build-in Open Drain function. Interrupt INT11/ Wake up from Sleep and Green mode. T2 Capture timer Input.
P12/INT12/UTX	I/O	Port 12 bi-direction pins/ Built-in pull-up resistances. P12 build-in Open Drain function. (Note:P12 no support Open Drain function in ICE emulate.) Interrupt INT12/ Wake up from Sleep and Green mode. UART TX PIN.
P13/INT13/URX	I/O	Port 13 bi-direction pins/ Built-in pull-up resistances. P13 build-in Open Drain function. (Note:P13 no support Open Drain function in ICE emulate.) Interrupt INT13/ Wake up from Sleep and Green mode. UART RX PIN.
P14/INT14/T3EVT	I/O	Port 14 bi-direction pin/ Built-in pull-up resistances. Interrupt INT14/ Wake up from Sleep and Green mode. T3 Capture timer Input.
P15/INT15/T1EVT	I/O	Port 15 bi-direction pin/ Built-in pull-up resistances. Interrupt INT15/ Wake up from Sleep and Green mode. T1 Event Counter Clock Input
P5.0/SCK	I/O	Port 5.0 bi-direction pin and Built-in pull-up resisters. build-in Open Drain function. SIO SCK PIN.
P5.1/SI	I/O	Port 5.1 bi-direction pin and Built-in pull-up resisters. build-in Open Drain function. SIO SI PIN.
P5.2/SO	I/O	Port 5.2 bi-direction pin and Built-in pull-up resisters. build-in Open Drain function. SIO SO PIN.
P5.3 / BZ1 / PWM1	I/O	Port 5.3 bi-direction pin, TC1 signal output pin or PWM1 output pin. Built-in pull-up resisters.
P5.4 / BZ0 / PWM0	I/O	Port 5.4 bi-direction pin, TC0 signal output pin or PWM0 output pin. Built-in pull-up resisters.

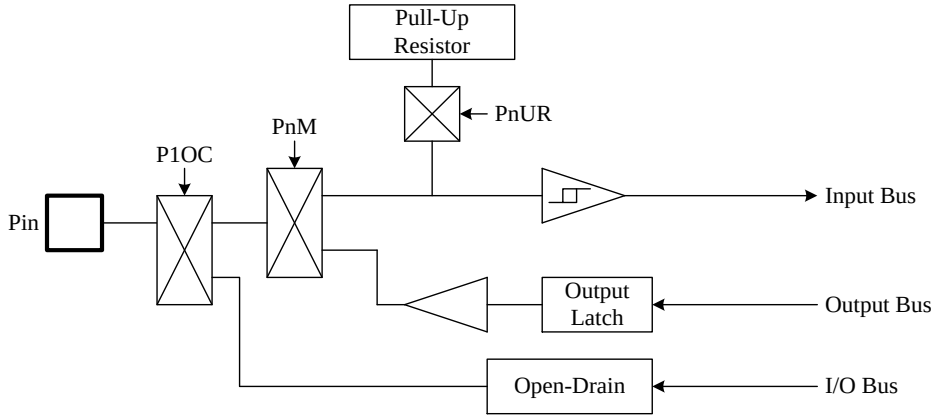
XIN/P5.5	I/O	Port 5.5 bi-direction pins / Built-in pull-up resistances and shared with XIN
XOUT/P5.6	I/O	Port 5.6 bi-direction pins / Built-in pull-up resistances and shared with XOUT
P4.0/AIN0 ~ P4.6/AIN6	I/O	Port 4.0 ~ Port 4.6 bi-direction pins / built-in pull-up resistances and shared with ADC channels AIN0~AIN6
AIN0 ~ AIN6	I	Analog signal input pins for ADC converter.
COM0 ~ COM3	O	LCD driver common pins.
SEG0 ~ SEG31	O	LCD driver segment pins.
SEG32/P20~ SEG35/P23	O	LCD driver segment pins and shared with P20~P23
AVREFH	I	ADC's reference high voltage input pins.
OP1IN+, OP1IN-	I	Operational Amplifier 1 Input Channels
OP1OUT	O	Operational Amplifier 1 Output Channel
OP2IN+, OP2IN-	I	Operational Amplifier 2 Input Channels
OP2OUT	O	Operational Amplifier 2 Output Channel
PGIAIN1+, PGIAIN1-	I	PGIA1 Input Channel.
PGIAOUT1	O	PGIA1 Output Channel
PGIABIAS1	I	PGIA1 Bias Voltage input
PGIAIN2+, PGIAIN2-	I	PGIA2 Input Channel.
PGIAOUT2	O	PGIA2 Output Channel
PGIABIAS2	I	PGIA2 Bias Voltage input
DAO	O	12-bit Digital to Analog Converter output
DAREF	I	DAC Reference Voltage Input.
C1+	A	Positive capacitor 1 terminal for charge pump regulator
C1-	A	Negative capacitor 1 terminal for charge pump regulator
C2+	A	Positive capacitor 2 terminal for charge pump regulator
C2-	A	Negative capacitor 2 terminal for charge pump regulator
VLCD2	P	LCD bias voltage. VLCD2=2/3*VLCD. VLCD=DVDD.
VLCD1	P	LCD bias voltage. VLCD1=1/3*VLCD. VLCD=DVDD.
VLCD0	P	LCD bias voltage. VLCD0=.VSS

1.5 PIN CIRCUIT DIAGRAMS

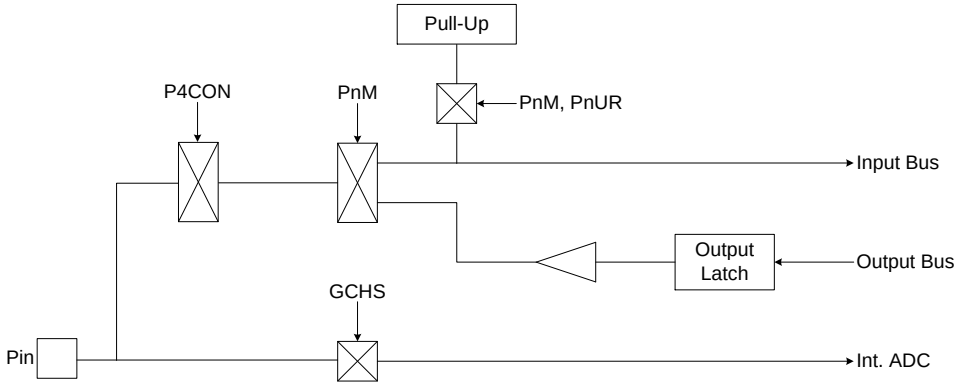
General purpose I/O pin:



General purpose I/O pin with open-drain structure:



Port 4 structure:

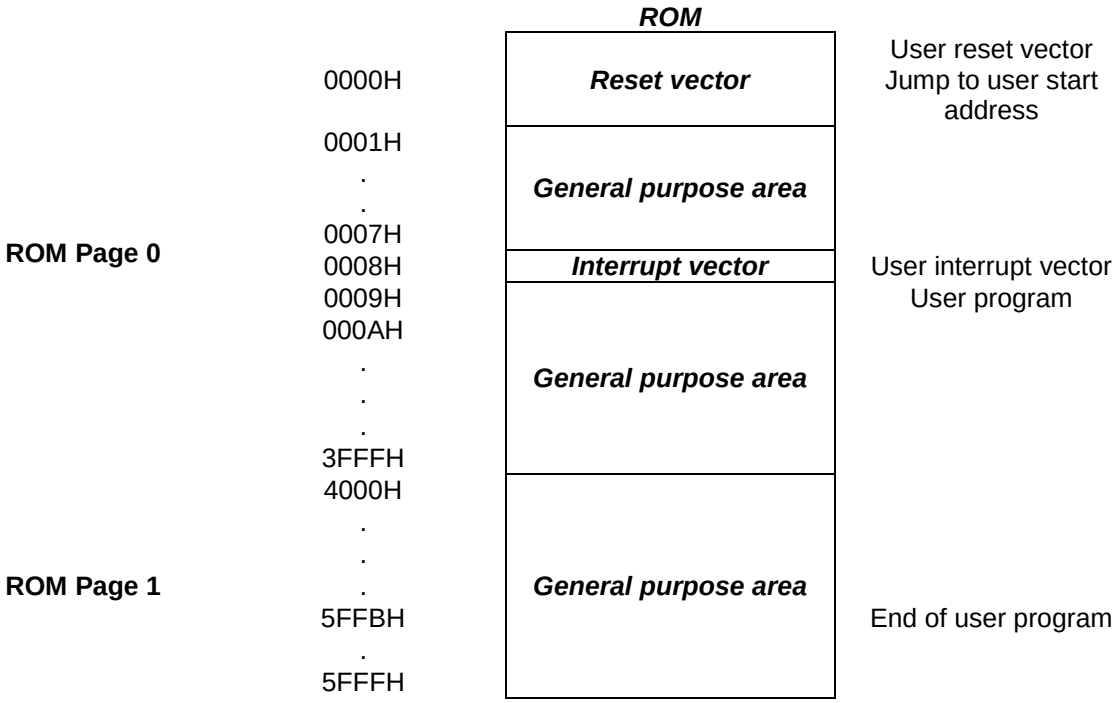


2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)

☞ ***24K words ROM***



2.2 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
```

2.3 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: The "PUSH" and "POP" operations aren't through instruction (PUSH, POP). MCU can executed save and load ACC and working registers (0x80~0x87) by hardware automatically.***

➢ **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.
                            (load ACC and working registers (0x80~0x87) by hardware automatically)
Int_:
        …
        …

        RETI                ; End of interrupt service routine
                            (load ACC and working registers (0x80~0x87) by hardware automatically)
        …

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.
                                        ;(load ACC and working registers (0x80~0x87) by hardware automatically)
                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.

                …
                …

                RETI                    ; End of interrupt service routine.
                                        ; (load ACC and working registers (0x80~0x87) by hardware automatically)
                …

                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.***

2.4 Jmp/Call DESCRIPTION

The Jmp and Call instruction length=2 machine word (2 ROM Address).When executing of "Jmp $+" or "Jmp $-",user have to check Jmp/Call instruction length.

✸ ***Note: Jmp and Call instruction length = 2 machine word (2 ROM Address).***

➢ **Example: Jmp $+**

```
ROM Address
  0020H         JMP     $+8             //Jump to "MOV A,#07H"
  0022H         JMP     $+5             //Jump to "MOV A,#06H"
  0024H         JMP     $+2             //Jump to "MOV A,#05H"
  0026H         MOV     A,#05H
  0027H         MOV     A,#06H
  0028H         MOV     A,#07H
  0029H         JMP     A0POINT
            …   …       …
```

2.5 LOOK-UP TABLE DESCRIPTION

In the ROM's data lookup function, X register is pointed to high byte address (bit 16~bit 23), 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   X, #TABLE1$H    ; To set lookup table1's high address
                B0MOV   Y, #TABLE1$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 is overflow, Y=Y+1.
                JMP     @F              ; Y is not overflow.
                INCMS   X               ; Y is overflow, X=X+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 X, Y registers will not increase automatically when Y, Z registers crosses boundary from 0xFF to 0x00. Therefore, user must take care such situation to avoid loop-up table errors. If Z register is overflow, Y register must be added one. If Y register is overflow, X register must be added one. The following INC_XYZ macro shows a simple method to process X, Y and Z registers automatically.***

➢ **Example: INC_XYZ macro.**

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

                INCMS   Y               ; Y+1
                JMP     @F              ; Not overflow

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

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

```
                B0MOV     X, #TABLE1$H    ; To set lookup table1's high address
                B0MOV     Y, #TABLE1$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_XYZ                   ; 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 loop-up table is to add X, Y or Z index register by accumulator. Please be careful if "carry" happen.

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

```
                B0MOV     X, #TABLE1$H    ; To set lookup table1's high address
                B0MOV     Y, #TABLE1$M    ; To set lookup table1'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.
                JMP       GETDATA         ; Y is not overflow.
                INCMS     X               ; Y is overflow, X=X+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
                …
```

2.6 JUMP TABLE DESCRIPTION

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

Program counter will occur ERROR when executing PCL addition instruction (B0ADD/ADD/ADC PCL,A) in ROM Address PCL=0xFD or 0xFE.
If the instruction (B0ADD/ADD/ADC PCL,A) in ROM Address =(xxxxFD) or(xxxxFE) , user have to move the instruction (B0ADD/ADD/ADC PCL,A) to the top of next program memory page (xxxx00H).
Here one page mean 256 words.

✸ ***Note: Program counter will occur ERROR when executing PCL addition instruction (B0ADD/ADD/ADC PCL,A) in ROM Address PCL=0xFD or 0xFE.***

➢ **Example: Jump Table ERROR.**

ROM Address

```
00FCH       MOV       A,#0
00FDH       B0ADD     PCL,a      //← ERROR!
                                 //"BOADD PCL,A" instruction in ROM ADDRESS PCL=0xFD.
                                 //After executing BOADD instruction, program counter
                                 occur ERROR.
00FEH       JMP       Here0
0100H       JMP       Here1
0102H       JMP       Here2
0104H       JMP       Here3
            …  …      …
```

Modify :

ROM Address

```
00FCH       MOV       A,#0
00FDH       NOP
00FEH       NOP
00FFH       NOP
0100H       B0ADD     PCL,a      //"B0ADD PCL,A"instruction in ROM ADDRESS PCL=0x00.
                                 //After executing BOADD instruction, jump to Here0.
0101H       JMP       Here0
0103H       JMP       Here1
0105H       JMP       Here2
0107H       JMP       Here3
```

➢ **Example: Jump Table ERROR.**

ROM Address

```
00FDH       MOV       A,#2
00FEH       B0ADD     PCL,a      //← ERROR!
                                 //"BOADD PCL,A" instruction in ROM ADDRESS PCL=0xFE.
                                 //After executing BOADD instruction, program counter
                                 occur ERROR.
00FFH       JMP       Here0
0101H       JMP       Here1
0103H       JMP       Here2
0105H       JMP       Here3
            …  …      …
```

Modify :

```
ROM Address
  00FDH     MOV     A,#2
  00FEH     NOP
  00FFH     NOP
  0100H     B0ADD   PCL,a       //"B0ADD PCL,A"instruction in ROM ADDRESS PCL=0x00.
                                //After executing BOADD instruction, jump to HERE1
  0101H     JMP     Here0
  0103H     JMP     Here1
  0105H     JMP     Here2
  0107H     JMP     Here3
```

➢ **Example: Jump Table correct.**

```
ROM Address
  00FBH     MOV     A,#4
  00FCH     B0ADD   PCL,a       //"BOADD PCL,A" instruction in ROM ADDRESS PCL=0xFC.
                                //After executing BOADD instruction, jump to Here2.
  00FDH     JMP     Here0
  00FFH     JMP     Here1
  0101H     JMP     Here2
  0103H     JMP     Here3
          … …       …
```

➢ **Example: The following @JMP_A_2ADR macro shows a simple method to jump Table.**

```
;-------------------------------------------------------------------------------------------
;   mov             a, #m                   //m=0,1,2,...(n-1). M<=125
;   @JMP_A_2ADR     n,MACRO_Buffer          //n=Total commands of jmp table. N<=126
;   jmp             here_0                  //m=0
;   jmp             here_1                  //m=1
;   jmp             here_2                  //m=2
;   jmp             here_3                  //m=3
;   jmp             here_4                  //m=4
;   ...         ...         ...
;-------------------------------------------------------------------------------------------

    .DATA
MACRO_Buffer    ds          1           //Using @JMP_A_2ADR should define this register.
    .CODE
                mov         A,#2
                @JMP_A_2ADR 5,MACRO_Buffer  //Jmp to here_2
                Jmp         here_0
                Jmp         here_1
                Jmp         here_2
                Jmp         here_3
                jmp         here_4
```

2.7 CODE OPTION TABLE

Code Option	Content	Function Description
High_Clk	IHRC_16M	Internal 16M RC Oscillator, Pin will be I/O port P55, P56.
	Ext_RC	Low cost RC for external high clock oscillator. XOUT pin is general purpose I/O pin.
	32K X'tal	Low frequency, power saving crystal (e.g. 32.768KHz) for external high clock oscillator.
	12M X'tal	High speed crystal /resonator (e.g. 12MHZ) for external high clock oscillator.
	4M X'tal	Standard crystal /resonator (e.g. 4MHZ) for external high clock oscillator.
Fcpu	Fhosc/1	Instruction cycle is 1 oscillator clocks.
	Fhosc/2	Instruction cycle is 2 oscillator clocks.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.
	Fhosc/8	Instruction cycle is 8 oscillator clocks.
	Fhosc/16	Instruction cycle is 16 oscillator clocks.
Watch_Dog	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
	Always_On	Watchdog timer is always on enable even in power down and green mode.
Noise Filter	Enable	Enable Noise Filter function to enhance EMI performance
	Disable	Disable Noise Filter function
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.
Low Power	Enable	Enable Low Power function to save Operating current.
	Disable	Disable Low Power function.
LVD	LVD_L	LVD=2.2V Reset
	LVD_M	LVD=2.2V Reset, 2.4V Indicator by bit LVD24 of PFLAG register
	LVD_H	LVD=2.4V Reset, 3.6V Indicator by bit LVD36 of PFLAG register
	LVD_MAX	LVD=3.6V Reset
External Reset Length	Enable	Enable External reset de-bounce function.
	Disable	Disable External reset de-bounce function.

✷ ***Note: In high noisy environment, set Watch_Dog as "Always_On" is strongly recommended.***

2.8 DATA MEMORY (RAM)

☞ ***2048X 8-bit RAM***

	Address	RAM location	
BANK 0	000h “ “ “ “ “ 07Fh	***General purpose area***	000h~07Fh of Bank 0 = To store general purpose data (128 bytes).
	080h “ “ “ “ “ 0FFh	***System register*** ***End of bank 0 area***	080h~0FFh of Bank 0 store system registers (128 bytes).
BANK 1	100h “ “ “ “ “ 1FFh	***General purpose area*** ***End of bank 1 area***	100h~1FFh of Bank 1 = To store general purpose data (256 bytes).
BANK 2	200h “ “ “ “ “ 2FFh	***General purpose area*** ***End of bank 2 area***	200h~2FFh of Bank 2 = To store general purpose data (256 bytes).
BANK 3	300h 3FFh	***General purpose area*** ***End of bank 3 area***	300h~3FFh of Bank 3 = To store general purpose data (256 bytes).
…..		***……..***	
BANK 8	800h 87Fh	***General purpose area*** ***End of bank 8 area***	800h~87Fh of Bank 8 = To store general purpose data (128 bytes).

2.8.1 RBANK Register

087H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RBANK	-	-	-	-	RBNKS3	RBNKS2	RBNKS1	RBNKS0
	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

RBANKS[3:0]: RAM Bank select bit.

0000 = RAM Bank0
0001 = RAM Bank1
0010 = RAM Bank2
0011 = RAM Bank3
0100 = RAM Bank4
0101 = RAM Bank5
0110 = RAM Bank6
0111 = RAM Bank7
1000 = RAM Bank8
1111 = RAM Bank15 (LCD RAM)

2.9 SYSTEM REGISTER

2.9.1 SYSTEM REGISTER TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	X	PFLAG	RBANK		LCDM	CPM	CPCKS	-	-	-	-
9	STK15L	STK15H	STK14L	STK14H	STK13L	STK13H	STK12L	STK12H	STK11L	STK11H	STK10L	STK10H	STK9L	STK9H	STK8L	STK8H
A	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H
B	-	ADM	ADB	ADR	SIOM	SIOR	SIOB		P0M	PGIA2CKS	PGIA2M	PGIA1CKS	PGIA1M	OPM	P4CON	PEDGE
C	P1W	P1M	P2M	-	P4M	P5M	INTRQ1	INTEN1	INTRQ0	INTEN0	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	P2	-	P4	P5	T1M	T1C	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	P2UR	-	P4UR	P5UR	@HL	@YZ	-	P1OC	T2M	T2CL	T2CH	CPT2M	CPT2CL	CPT2CH
F	URRX	URTX	URCR	UTXD	URXD	INTRQ2	INTEN2	DAM	DABL	DABH	T3M	T3CL	T3CH	CPT3M	CPT3CL	CPT3CH

2.9.2 SYSTEM REGISTER DESCRIPTION

L, H = Working & @HL addressing register
Y, Z = Working, @YZ and ROM addressing register
PFLAG = ROM page and special flag register
LCDM1 = LCD mode register
CPCKS = Charge-Pump Regulator clock selection
PGIAM = PGIA mode and gain register
ADM = ADC mode register
ADR = ADCs resolution selects register
PNM = Port N input/output mode register
PN = Port N data buffer
INTEN = Interrupt enable register
T0M = Timer 0 mode register
T0C = Timer 0 counting register
TC1M = Timer/Counter 1 mode register
TC1C = Timer/Counter 1 counting register
STKP = Stack pointer buffer
@HL = RAM HL indirect addressing index pointer
P4CON= Port 4 configuration setting
R = Working register and ROM look-up data buffer
OPTION= RTC and RCLK options.
RBANK= RAM bank select register
PGIACKS = PGIA clock selection
CPM = Charge pump mode
OPM = OPA mode register
ADB = ADC data buffer
P1W = Port 1 wakeup register
PNUR = Port N pull-up register
INTRQ = Interrupt request register
OSCM = Oscillator mode register
PCH, PCL = Program counter
TC0M = Timer/Counter 0 mode register
TC0C = Timer/Counter 0 counting register
TC0R = Timer/Counter 0 auto-reload data buffer
@YZ = RAM YZ indirect addressing index pointer

2.9.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Name
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
085H	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0	R/W	X
086H	NT0	NPD	LVD36	LVD24	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	RBNKS3	RBNKS2	RBNKS1	RBNKS0	R/W	RBANK
088H										
089H	RES1	RES0	-	-	RCLK	P2SEG		LCDENB	R/W	LCDM
08AH			-	-			AREGEN2	AREGEN1	R/W	CPM
08BH							CPCKS1	CPCKS0	R/W	CPCKS
08CH										
08DH										
08EH										
08FH										
090H	S15PC7	S15PC6	S15PC5	S15PC4	S15PC3	S15PC2	S15PC1	S15PC0	R/W	STK15L
091H	-	S15PC14	S15PC13	S15PC12	S15PC11	S15PC10	S15PC9	S15PC8	R/W	STK15H
092H	S14PC7	S14PC6	S14PC5	S14PC4	S14PC3	S14PC2	S14PC1	S14PC0	R/W	STK14L
093H	-	S14PC14	S14PC13	S14PC12	S14PC11	S14PC10	S14PC9	S14PC8	R/W	STK14H
094H	S13PC7	S13PC6	S13PC5	S13PC4	S13PC3	S13PC2	S13PC1	S13PC0	R/W	STK13L
095H	-	S13PC14	S13PC13	S13PC12	S13PC11	S13PC10	S13PC9	S13PC8	R/W	STK13H
096H	S12PC7	S12PC6	S12PC5	S12PC4	S12PC3	S12PC2	S12PC1	S12PC0	R/W	STK12L
097H	-	S12PC14	S12PC13	S12PC12	S12PC11	S12PC10	S12PC9	S12PC8	R/W	STK12H
098h	S11PC7	S11PC6	S11PC5	S11PC4	S11PC3	S11PC2	S11PC1	S11PC0	R/W	STK11L
099H	-	S11PC14	S11PC13	S11PC12	S11PC11	S11PC10	S11PC9	S11PC8	R/W	STK11H
09AH	S10PC7	S10PC6	S10PC5	S10PC4	S10PC3	S10PC2	S10PC1	S10PC0	R/W	STK10L
09BH	-	S10PC14	S10PC13	S10PC12	S10PC11	S10PC10	S10PC9	S10PC8	R/W	STK10H
09CH	S9PC7	S9PC6	S9PC5	S9PC4	S9PC3	S9PC2	S9PC1	S9PC0	R/W	STK9L
09DH	-	S9PC14	S9PC13	S9PC12	S9PC11	S9PC10	S9PC9	S9PC8	R/W	STK9H
09EH	S8PC7	S8PC6	S8PC5	S8PC4	S8PC3	S8PC2	S8PC1	S8PC0	R/W	STK8L
09FH	-	S8PC14	S8PC13	S8PC12	S9PC11	S8PC10	S8PC9	S8PC8	R/W	STK8H
0A0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0A1H	-	S7PC14	S7PC13	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0A2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0A3H	-	S6PC14	S6PC13	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0A4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0A5H	-	S5PC14	S5PC13	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0A6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0A7H	-	S4PC14	S4PC13	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0A8h	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0A9H	-	S3PC14	S3PC13	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0AAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0ABH	-	S2PC14	S2PC13	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0ACH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0ADH	-	S1PC14	S1PC13	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0AEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0AFH	-	S0PC14	S0PC13	S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H
0B0H										
0B1H	ADENB	ADS	EOC	GCHS	LVBAT	CHS2	CHS1	CHS0	R/W	ADM
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0B3H	-	ADCKS2	ADCKS1	ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0B4H	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA	R/W	SIOM
0B5H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	R/W	SIOR
0B6H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB
0B7H										
0B8h							P01M	P00M	R/W	P0M
0B9H	PGIA2CKS7	PGIA2CKS6	PGIA2CKS5	PGIA2CKS4	PGIA2CKS3	PGIA2CKS2	PGIA2CKS1	PGIA2CKS0	R/W	PGIA2CKS
0BAH	P4GS3	P4GS2	P4GS1	P4GS0	-	P3GS1	P3GS0	PGIA2ENB	R/W	PGIA2M
0BBH	PGIA1CKS7	PGIA1CKS6	PGIA1CKS5	PGIA1CKS4	PGIA1CKS3	PGIA1CKS2	PGIA1CKS1	PGIA1CKS0	R/W	PGIA1CKS
0BCH	P2GS3	P2GS2	P2GS1	P2GS0	-	P1GS1	P1GS0	PGIA1ENB	R/W	PGIA1M
0BDH	-	-	-	-	OP2UGB	OP1UGB	OP2ENB	OP1ENB	R/W	OPM
0BEH	-	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	W	P4CON
0BFH	-	-	-	-	P01G1	P01G0	P00G1	P00G0	R/W	PEDGE

0C0H	-	-	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H	-	-	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C2H					P23M	P22M	P21M	P20M	R/W	P2M
0C3H										
0C4H		P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C5H	-	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C6H	SIOIRQ	ADCIRQ	P15IRQ	P14IRQ	P13IRQ	P12IRQ	P11IRQ	P10IRQ	R/W	INTRQ1
0C7H	SIOIEN	ADCIEN	P15IEN	P14IEN	P13IEN	P12IEN	P11IEN	P10IEN	R/W	INTEN1
0C8H	TC1IRQ	TC0IRQ	T1IRQ	T0IRQ	RXIRQ	TXIRQ	P01IRQ	P00IRQ	R/W	INTRQ0
0C9H	TC1IEN	TC0IEN	T1IEN	T0IEN	RXIEN	TXIEN	P01IEN	P00IEN	R/W	INTEN0
0CAH	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CBH										
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	R/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	-	PC14	PC13	PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H							P01	P00	R/W	P0
0D1H	-	-	P15	P14	P13	P12	P11	P10	R/W	P1
0D2H					P23	P22	P21	P20	R/W	P2
0D3H										
0D4H		P46	P45	P44	P43	P42	P41	P40	R/w	P4
0D5H	-	P56	P55	P54	P53	P52	P51	P50	R/W	P5
0D6H	T1ENB	T1rate2	T1rate1	T1rate0	T1CKS				R/W	T1M
0D7H	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0	R/W	T1C
0D8H	T0ENB	T0RATE2	T0RATE1	T0RATE0	TC1X8	TC0X8	TC0GN	T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0RATE2	TC0RATE1	TC0RATE0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1RATE2	TC1RATE1	TC1RATE0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE	-	-	STKPB4	STKPB3	STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H							P01R	P00R	W	P0UR
0E1H	-	-	-	-	P13R	P12R	P11R	P10R	W	P1UR
0E2H					P23R	P22R	P21R	P20R	W	P2UR
0E3H										
0E4H		P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H	-	P56R	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
0E8H										
0E9H		P52OC	P51OC	P50OC	P13OC	P12OC	P11OC	P10OC	R/W	P1OC
0EAH	T2ENB	T2rate2	T2rate1	T2rate0	T2CKS				R/W	T2M
0EBH	T2CL7	T2CL6	T2CL5	T2CL4	T2CL3	T2CL2	T2CL1	T2CL0	R/W	T2CL
0ECH	T2CH7	T2CH6	T2CH5	T2CH4	T2CH3	T2CH2	T2CH1	T2CH0	R/W	T2CH
0EDH	CPT2EN				CPT2MD	CPT2Start	CPT2G1	CPT2G0	R/W	CPT2M
0EEH	CPT2C7	CPT2C6	CPT2C5	CPT2C4	CPT2C3	CPT2C2	CPT2C1	CPT2C0	R/W	CPT2CL
0EFH	CPT2C15	CPT2C14	CPT2C13	CPT2C12	CPT2C11	CPT2C10	CPT2C9	CPT2C8	R/W	CPT2CH
0F0H	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0	R/W	URRX
0F1H	UTXEN	UTXPEN	UTXPS	UTXBRK	URXBZ	UTXBZ	-	-	R/W	URTX
0F2H	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0	R/W	URCR
0F3H	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0	R/W	UTXD
0F4H	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0	R/W	URXD
0F5H							T3IRQ	T2IRQ	R/W	INTRQ2
0F6H							T3IEN	T2IEN	R/W	INTEN2
0F7H	DAEN								R/W	DAM
0F8H					DAB3	DAB2	DAB1	DAB0	W	DABL
0F9H	DAB11	DAB10	DAB9	DAB8	DAB7	DAB6	DAB5	DAB4	W	DABH
0FAH	T3ENB	T3rate2	T3rate1	T3rate0	T3CKS				R/W	T3M
0FBH	T3CL7	T3CL6	T3CL5	T3CL4	T3CL3	T3CL2	T3CL1	T3CL0	R/W	T3CL
0FCH	T3CH7	T3CH6	T3CH5	T3CH4	T3CH3	T3CH2	T3CH1	T3CH0	R/W	T3CH
0FDH	CPT3EN				CPT3MD	CPT3Start	CPT3G1	CPT3G0	R/W	CPT3M
0FEH	CPT3C7	CPT3C6	CPT3C5	CPT3C4	CPT3C3	CPT3C2	CPT3C1	CPT3C0	R/W	CPT3CL
0FFH	CPT3C15	CPT3C14	CPT3C13	CPT3C12	CPT3C11	CPT3C10	CPT3C9	CPT3C8	R/W	CPT3CH

✸ ***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.***

2.10 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
```

2.11 PROGRAM COUNTER

The program counter (PC) is a 15-bit binary counter separated into the high-byte 6 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 14.

	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	-	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

☞ ONE INSTRUCTION SKIPPING

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

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

Program counter will occur ERROR when executing PCL addition instruction (B0ADD/ADD/ADC PCL,A) in ROM Address PCL=0xFD or 0xFE.

If the instruction (B0ADD/ADD/ADC PCL,A) in ROM Address =(xxxxFD) or(xxxxFE) , user have to move the instruction (B0ADD/ADD/ADC PCL,A) to the top of next program memory page (xxxx00H).

Here one page mean 256 words.

- ***Note: Program counter will occur ERROR when executing PCL addition instruction (B0ADD/ADD/ADC PCL,A) in ROM Address PCL=0xFD or 0xFE.***
- ***Note: "Jmp" and "CALL" instruction length = 2 machine word (2 ROM Address).***
- ***Note: Please refer to the "2.6 JUMP TABLE DESCRIPTION" about jump table application.***

2.12 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 (DVDD > 3.6V).
1 = Active (DVDD <= 3.6V).

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

2.13 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
        …
        …
```

2.14 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	-	-	-	-	-	-	-	-

✸ ***Note: Please refer to the "2.5 LOOK-UP TABLE DESCRIPTION" about Y/Z register look-up table application.***

➢ **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
        …
```

2.15 X REGISTERS

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

- can be used as general working registers
- can be used as ROM data pointer with the MOVC instruction for look-up table

085H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0
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

✸ ***Note: Please refer to the "2.5 LOOK-UP TABLE DESCRIPTION" about X register look-up table application.***

2.16 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 "2.5 LOOK-UP TABLE DESCRIPTION" about R register look-up table application.***

2.17 ADDRESSING MODE

2.17.1 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.***

2.17.2 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.
```

2.17.3 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.
```

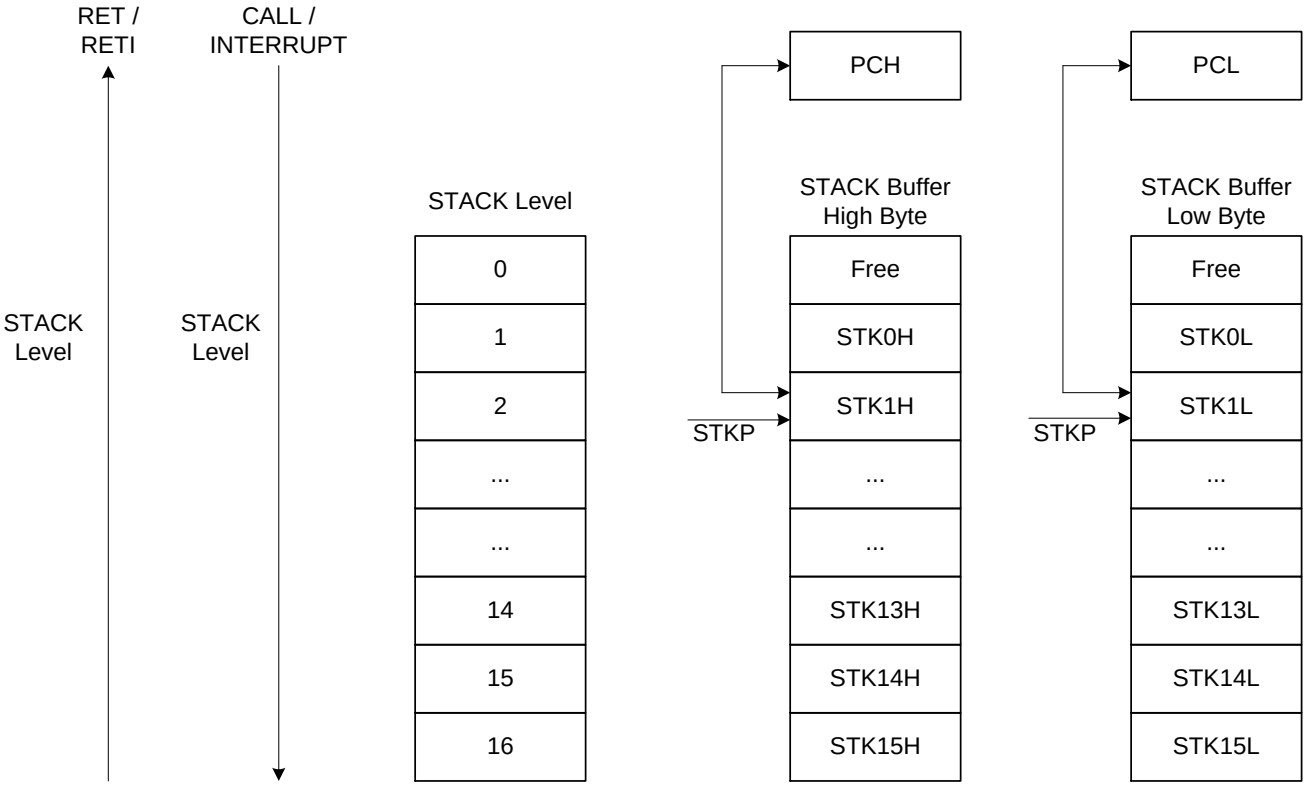
- **Example: Indirectly addressing mode with @YZ register**

```
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.
```

2.18 STACK OPERATION

2.18.1 OVERVIEW

The stack buffer has 16-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.



2.18.2 STACK REGISTERS

The stack pointer (STKP) is a 4-bit register to store the address used to access the stack buffer, 15-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	-	-	STKPB4	STKPB3	STKPB2	STKPB1	STKPB0
Read/Write	R/W	-	-	R/W	R/W	R/W	R/W	R/W
After reset	0	-	-	1	0	1	1	1

Bit[4:0] **STKPBn:** Stack pointer (n = 0 ~ 4)

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

✸ ***Note: STKP initial value=0x17H.***

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

```
MOV     A, #00010111B   ;#017H
B0MOV   STKP, A
```

090H~0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	SnPC14	SnPC13	SnPC12	SnPC11	SnPC10	SnPC9	SnPC8
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

090H~0AFH	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 = 15 ~ 0)

2.18.3 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
	STKB4	STKPB3	STKPB2	STKPB1	STKPB0	High Byte	Low Byte	
0	1	0	1	1	1	Free	Free	-
1	1	0	1	1	0	STK0H	STK0L	-
2	1	0	1	0	1	STK1H	STK1L	-
3	1	0	1	0	0	STK2H	STK2L	-
4	1	0	0	1	1	STK3H	STK3L	-
5	1	0	0	1	0	STK4H	STK4L	-
6	1	0	0	0	1	STK5H	STK5L	-
7	1	0	0	0	0	STK6H	STK6L	-
8	0	1	1	1	1	STK7H	STK7L	-
9	0	1	1	1	0	STK8H	STK8L	-
10	0	1	1	0	1	STK9H	STK9L	-
11	0	1	1	0	0	STK10H	STK10L	-
12	0	1	0	1	1	STK11H	STK11L	-
13	0	1	0	1	0	STK12H	STK12L	-
14	0	1	0	0	1	STK13H	STK13L	-
15	0	1	0	0	0	STK14H	STK14L	-
16	0	0	1	1	1	STK15H	STK15L	-
> 16	0	0	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
	STKPB4	STKPB3	STKPB2	STKPB1	STKPB0	High Byte	Low Byte	
16	0	0	1	1	1	STK15H	STK15L	-
15	0	1	0	0	0	STK14H	STK14L	-
14	0	1	0	0	1	STK13H	STK13L	-
13	0	1	0	1	0	STK12H	STK12L	-
12	0	1	0	1	1	STK11H	STK11L	-
11	0	1	1	0	0	STK10H	STK10L	-
10	0	1	1	0	1	STK9H	STK9L	-
9	0	1	1	1	0	STK8H	STK8L	-
8	0	1	1	1	1	STK7H	STK7L	-
7	1	0	0	0	0	STK6H	STK6L	-
6	1	0	0	0	1	STK5H	STK5L	-
5	1	0	0	1	0	STK4H	STK4L	-
4	1	0	0	1	1	STK3H	STK3L	-
3	1	0	1	0	0	STK2H	STK2L	-
2	1	0	1	0	1	STK1H	STK1L	-
1	1	0	1	1	0	STK0H	STK0L	-
0	1	0	1	1	1	Free	Free	-

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

- Power on reset
- External reset
- Watchdog reset
- LVD reset (brown-out reset protection)

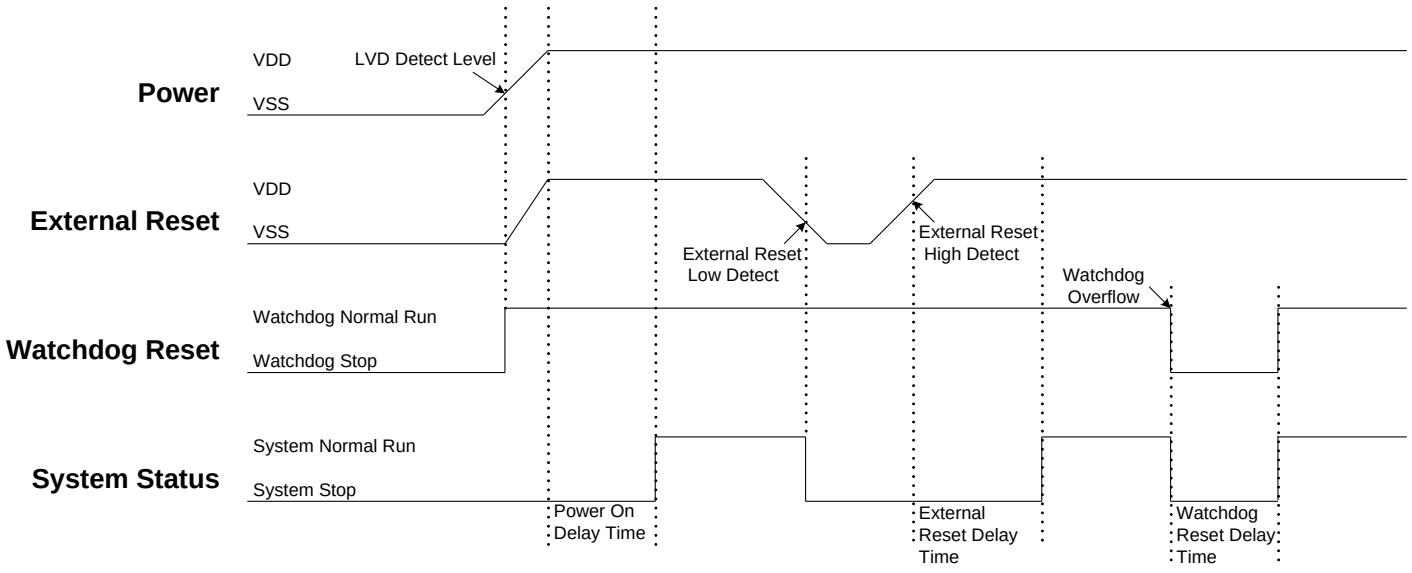
When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The NT0, NPD flags indicate system reset status. The system can depend on NT0, NPD status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

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 start-up of oscillator is not fixed. RC type oscillator's start-up time is very short, but the crystal type is longer. Under client terminal application, users have to take care the power on reset time for the master terminal requirement. The reset timing diagram is as following.



3.2 POWER ON RESET

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

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

3.3 WATCHDOG RESET

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

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

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

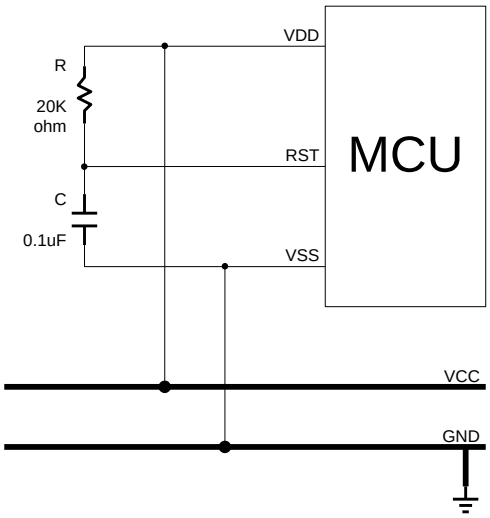
3.4 EXTERNAL RESET

External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the low voltage and the system is reset. The external reset operation 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.

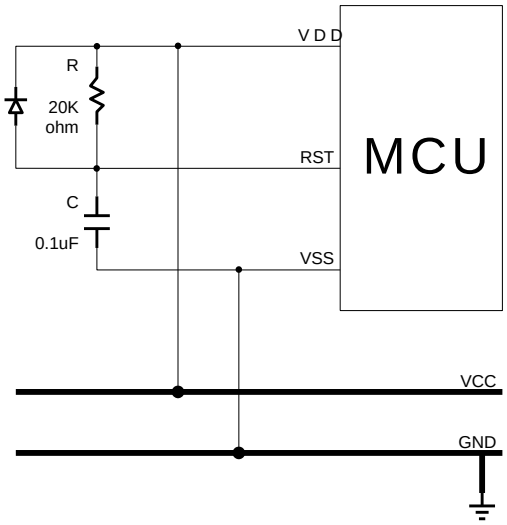
- **External reset:** System checks external reset pin status. If external reset pin is not high level, the system keeps reset status and waits external reset pin released.
- **System initialization:** All system registers is set as initial conditions and system is ready.
- **Oscillator warm up:** Oscillator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

3.4.1 EXTERNAL RESET CIRCUIT

The external reset circuit is a simple RC circuit as the following diagram.



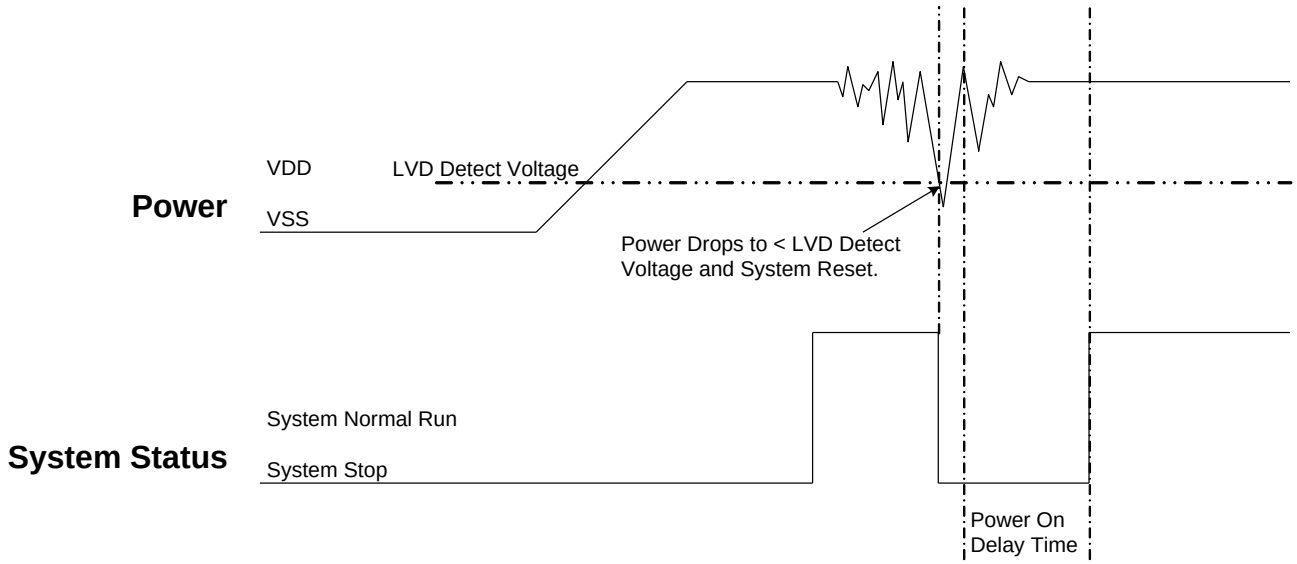
Simply RC Reset Circuit



Diode Reset Circuit for Brownout Reset

3.5 LOW VOLTAGE DETECTOR (LVD)

Low voltage detector (LVD) is built in system and the detect level is **2.2V/2.4V/3.6V**. If VDD dips below LVD detect level, system is reset. If VDD returns to above LVD detect level, system executes power on routine and program starts to work. Under high noisy situation, VDD is unstable and might dip to low voltage but not zero. This condition is called Brown Out. In brown out situation, the external reset would be error, and system can't reset by external reset and halts. System should be reset in brown out event. LVD operation provides a not zero voltage to do brown out detecting. If brown out occurs, LVD reset the system and protect system to keep in working status. LVD reset sequence is like power on reset sequence.



Brown Out Reset Diagram

4 SYSTEM CLOCK

4.1 OVERVIEW

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

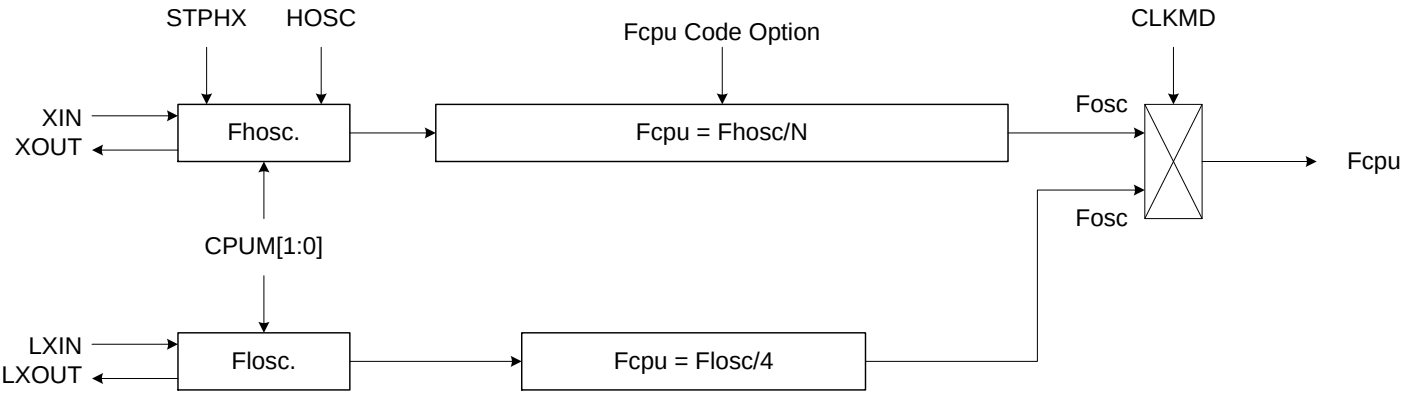
Both the high-speed clock and the low-speed clock can be system clock (Fosc). The system clock in slow mode is divided by 4 to be the instruction cycle (Fcpu).

☞ **Normal Mode (High Clock):** **Fcpu = Fhosc / N**, **N = 1,2,4,8,16 Select N by Fcpu code option.**

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

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.

4.2 CLOCK BLOCK DIAGRAM



System Clock Diagram

- HOSC: High_Clk code option.
- Fhosc: External high-speed clock / Internal high-speed RC clock.
- Flosc: External low-speed clock (32768 Hz crystal).
- Fosc: System clock source.
- Fcpu: Instruction cycle.

4.3 OSCM REGISTER

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

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

Bit 1 **STPHX:** External high-speed oscillator control bit.
0 = External high-speed oscillator free run.
1 = External high-speed oscillator free run stop. Internal low-speed RC oscillator is still running.

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

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

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

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

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

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

4.4 SYSTEM HIGH CLOCK

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

High_Clk Code Option	Description
IHRC	The high clock is internal 16MHz oscillator RC type. XIN and XOUT pins are general purpose I/O pins.
RC	The high clock is external RC type oscillator. XOUT pin is general purpose I/O pin.
32K	The high clock is external 32768Hz low speed oscillator.
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 INTERNAL HIGH RC

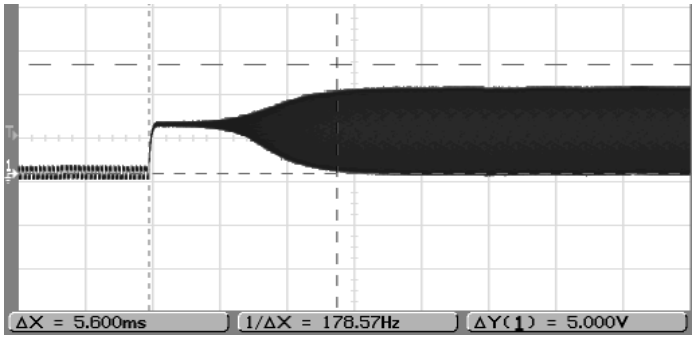
The chip is built-in RC type internal high clock (16MHz) controlled by “IHRC_16M” coed option. In “IHRC_16M” mode, the system clock is from internal 16MHz RC type oscillator and XIN / XOUT pins are general-purpose I/O pins.

- **IHRC:** High clock is internal 16MHz oscillator RC type. XIN/XOUT pins are general-purpose I/O pins.

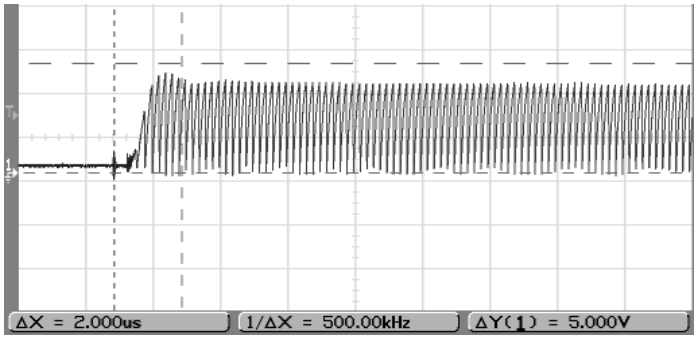
4.4.2 EXTERNAL HIGH CLOCK

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

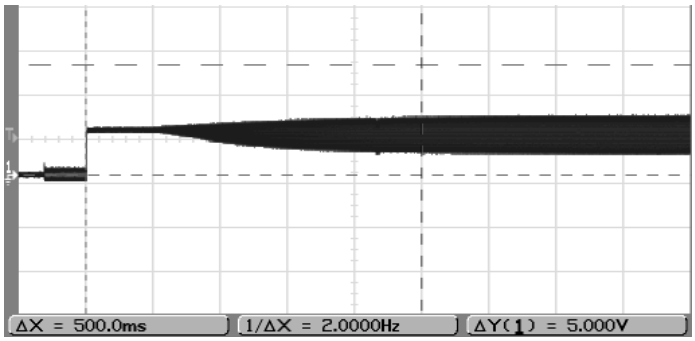
4MHz Crystal



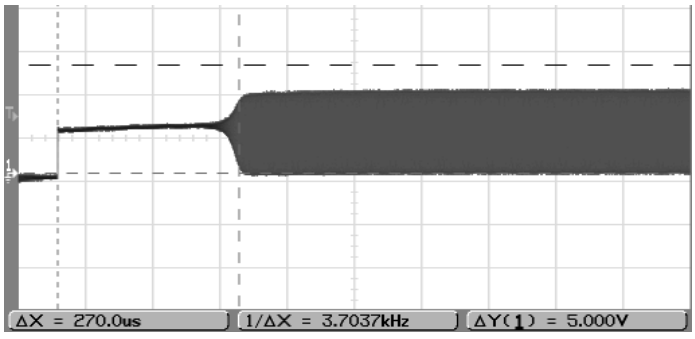
RC



32768Hz Crystal

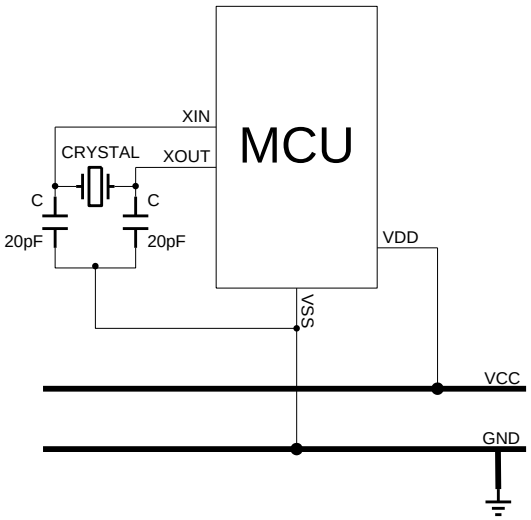


4MHz Ceramic



CRYSTAL/CERAMIC

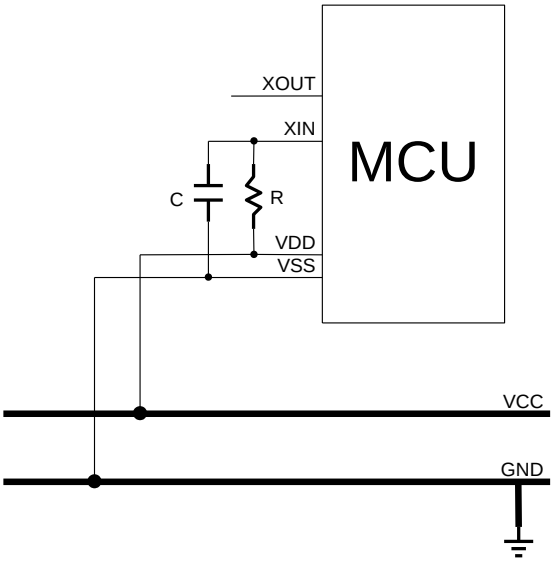
Crystal/Ceramic devices are driven by XIN, XOUT pins. For high/normal/low frequency, the driving currents are different. High_Clk code option supports different frequencies. 12M option is for high speed (ex. 12MHz). 4M option is for normal speed (ex. 4MHz). 32K option is for low speed (ex. 32768Hz).



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

RC

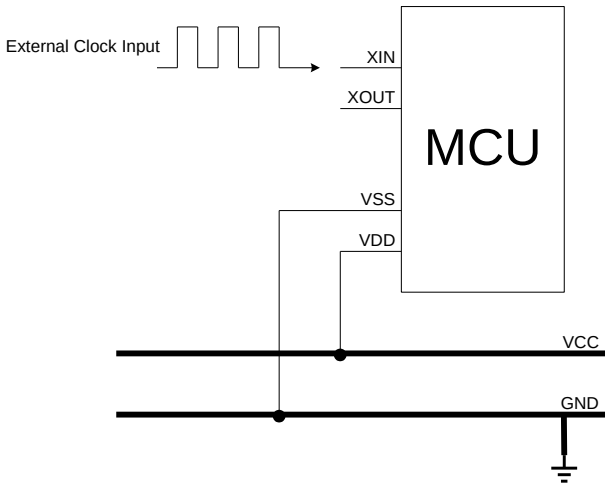
Selecting RC oscillator is by RC option of High_Clk code option. RC type oscillator's frequency is up to 10MHz. Using "R" value is to change frequency. 50P~100P is good value for "C". XOUT pin is general purpose I/O pin.



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

EXTERNAL CLOCK SIGNAL

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



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

4.5 SYSTEM LOW CLOCK

The system low clock source is the external low-speed oscillator with 32768 crystal.

☞ ***Flosc = ELOSC (External low crystal oscillator).***

☞ ***Slow mode Fcpu = Flosc / 4***

There are one conditions to stop external low clock. One is power down mode.

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

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

4.5.1 SYSTEM CLOCK MEASUREMENT

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

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

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

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

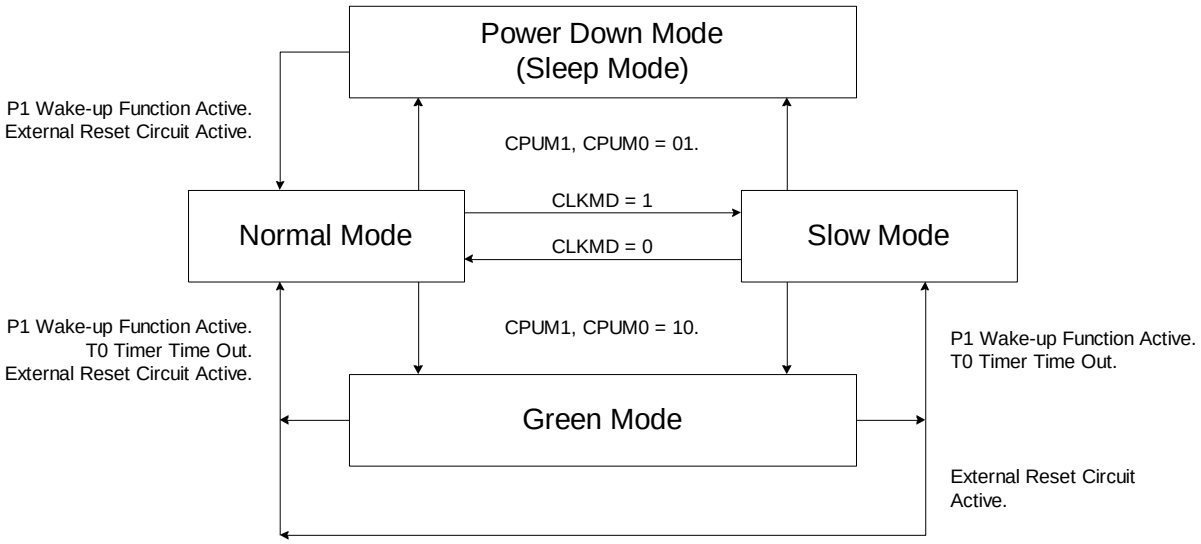
5 SYSTEM OPERATION MODE

5.1 OVERVIEW

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

- High-speed 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	STOP:STPHX=1 Run:STPHX=0	STOP:STPHX=1 Run:STPHX=0	Stop	
ELOSC	Running	Running	Running	Stop	
IHRC	Running	STOP:STPHX=1 Run:STPHX=0	STOP:STPHX=1 Run:STPHX=0	Stop	
ILRC	STOP:Watch Dog Disable. Run:Watch Dog Enable orAlways On.	STOP:Watch Dog Disable. Run:Watch Dog Enable or Always On.	STOP:Watch Dog Disable or Enable. Run:Watch Dog Always On.	STOP:Watch Dog Disable or Enable. Run:Watch Dog Always On.	
CPU instruction	Executing	Executing	Stop	Stop	
T0 timer	*Active	*Active	*Active	Inactive	* Active if T0ENB=1
T1 timer	*Active	*Active	Inactive	Inactive	* Active if T1ENB=1
TC0 timer	*Active	*Active	*Active	Inactive	* Active if TC0ENB=1
TC1 timer	*Active	*Active	Inactive	Inactive	* Active if TC1ENB=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	All inactive	
Wakeup source	-	-	P0,P1,T0,TC0 Reset	P0,P1, Reset	

- **EHOSC:** External high clock
- **ELOSC:** External low clock (32768 Hz)
- **IHRC:** Internal high clock (16M RC oscillator)
- **ILRC:** Internal Low clock

5.2 SYSTEM MODE SWITCHING EXAMPLE

➢ **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 10mS for external clock stable.

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

        MOV         A, #27          ; If VDD = 5V, internal RC=32KHz (typical) will delay
        B0MOV       Z, A
@@:     DECMS       Z               ; 0.125ms X 81 = 10.125ms 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 wake-up function.**

```
; Set T0 timer wakeup function.
        B0BCLR      FT0IEN      ; To disable T0 interrupt service
        B0BCLR      FT0ENB      ; To disable T0 timer
        MOV         A,#20H      ;
        B0MOV       T0M,A       ; To set T0 clock = Fcpu / 64
        MOV         A,#74H
        B0MOV       T0C,A       ; To set T0C initial value = 74H (To set T0 interval = 10 ms)
        B0BCLR      FT0IEN      ; To disable T0 interrupt service
        B0BCLR      FT0IRQ      ; To clear T0 interrupt request
        B0BSET      FT0ENB      ; To enable T0 timer
; Go into green mode
        B0BCLR      FCPUM0      ;To set CPUMx = 10
        B0BSET      FCPUM1
```

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

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

```
; Set T0 timer wakeup function with 0.5 sec RTC.
        B0BCLR      FRTC1       ; Set RTC timer period to 0.5 sec.
        B0BCLR      FRTC0

        B0BSET      FT0ENB      ; To enable T0 timer
        B0BSET      FT0TB       ; To enable RTC function
; Go into green mode
        B0BCLR      FCPUM0      ;To set CPUMx = 10
        B0BSET      FCPUM1
```

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 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 Fcpu 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

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	-	-	P15W	P14W	P13W	P12W	P11W	P10W
Read/Write	-	-	W	W	W	W	W	W
After reset	-	-	0	0	0	0	0	0

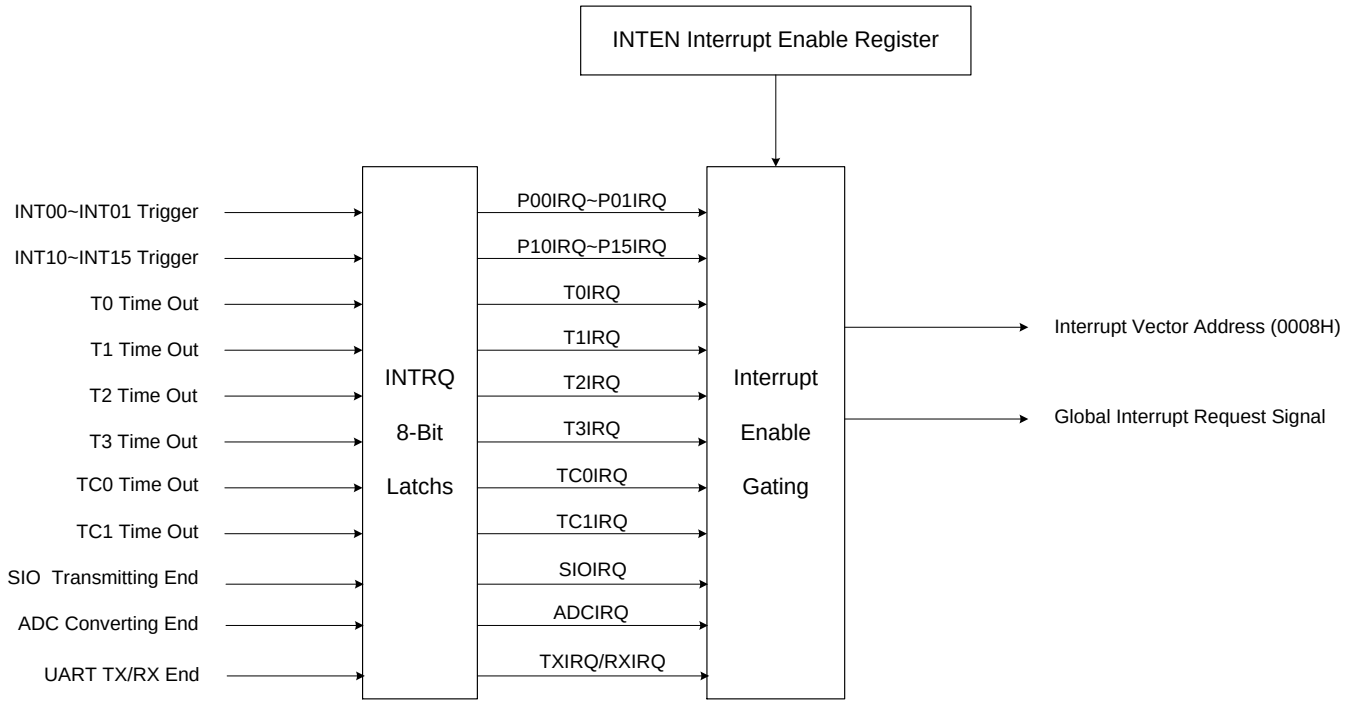
Bit[5:0] **P10W~P15W:** Port 1 wakeup function control bits.
0 = Disable P1n wakeup function.
1 = Enable P1n wakeup function.

Note: P0/P1 Green mode/sleep mode wakeup edge= level change.

6 INTERRUPT

6.1 OVERVIEW

This MCU provides 18interrupt sources, including 10 internal interrupt (T0/T1/T2/T3/TC0/TC1/ADC/SIO/URRX/URTX) and 8 external interrupt (INT00/INT01/INT10/INT11/INT12/INT13/INT14/INT15). Once interrupt service is executed, the GIE bit in STKP register will clear to “0” for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to “1” to accept the next interrupts’ request. All of the interrupt request signals are stored in INTRQ register.



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

6.2 INTEN INTERRUPT ENABLE REGISTER

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN0	TC1IEN	TC0IEN	T1IEN	T0IEN	RXIEN	TXIEN	P01IEN	P00IEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **TC1IEN:** TC1 timer interrupt control bit.
0 = Disable TC1 interrupt function.
1 = Enable TC1 interrupt function.

Bit 6 **TC0IEN:** TC0 timer interrupt control bit.
0 = Disable TC0 interrupt function.
1 = Enable TC0 interrupt function.

Bit 5 **T1IEN:** T1 timer interrupt control bit.
0 = Disable T1 interrupt function.
1 = Enable T1 interrupt function.

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

Bit 3 **RXIEN:** UART receive interrupt control bit.
0 = Disable UART receive interrupt function.
1 = Enable UART receive interrupt function.

Bit 2 **TXIEN:** UART transmit interrupt control bit.
0 = Disable UART transmit interrupt function.
1 = Enable UART transmit interrupt function.

Bit 1 **P01IEN:** External P01 interrupt (INT01) control bit.
0 = Disable INT01 interrupt function.
1 = Enable INT01 interrupt function.

Bit 0 **P00IEN:** External P00 interrupt (INT00) control bit.
0 = Disable INT00 interrupt function.
1 = Enable INT00 interrupt function.

0C7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN1	SIOIEN	ADCIEN	P15IEN	P14IEN	P13IEN	P12IEN	P11IEN	P10IEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **SIOIEN:** SIO interrupt control bit.
0 = Disable SIO interrupt function.
1 = Enable SIO interrupt function.

Bit 6 **ADCIEN:** ADC interrupt control bit.
0 = Disable ADC interrupt function.
1 = Enable ADC interrupt function.

Bit 5 **P15IEN:** External P15 interrupt (INT15) control bit.
0 = Disable INT15 interrupt function.
1 = Enable INT15 interrupt function.

Bit 4 **P14IEN:** External P14 interrupt (INT14) control bit.
0 = Disable INT14 interrupt function.
1 = Enable INT14 interrupt function.

Bit 3 **P13IEN:** External P13 interrupt (INT13) control bit.
0 = Disable INT13 interrupt function.
1 = Enable INT13 interrupt function.

Bit 2 **P12IEN:** External P12 interrupt (INT12) control bit.
0 = Disable INT12 interrupt function.
1 = Enable INT12 interrupt function.

Bit 1 **P11IEN:** External P11 interrupt (INT11) control bit.
0 = Disable INT11 interrupt function.
1 = Enable INT11 interrupt function.

Bit 0 **P10IEN:** External P10 interrupt (INT10) control bit.
0 = Disable INT10 interrupt function.
1 = Enable INT10 interrupt function.

- ***Note: If P00/P01=input mode, P00IEN/P01IEN=0, then P00/P01 interrupt not active, but P00IRQ/P01IRQ still active.***
- ***Note: If P10/P15=input mode, P10IEN/P15IEN=0, then P10/P15 interrupt not active, but P10IRQ/P15IRQ still active.***

0F6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN2							T3IEN	T2IEN
Read/Write							R/W	R/W
After reset							0	0

Bit 1 **T3IEN:** T3 timer interrupt control bit.
0 = Disable T3 interrupt function.
1 = Enable T3 interrupt function.

Bit 0 **T2IEN:** T2 timer interrupt control bit.
0 = Disable T2 interrupt function.
1 = Enable T2 interrupt function.

6.3 INTRQ INTERRUPT REQUEST REGISTER

INTRQ is the interrupt request flag register. The register includes all interrupt request indication flags. Each one of the interrupt requests occurs, the bit of the INTRQ register would be set “1”. The INTRQ value needs to be clear by programming after detecting the flag. In the interrupt vector of program, users know the any interrupt requests occurring by the register and do the routine corresponding of the interrupt request.

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ0	TC1IRQ	TC0IRQ	T1IRQ	T0IRQ	RXIRQ	TXIRQ	P01IRQ	P00IRQ
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **TC1IRQ:** TC1 timer interrupt request flag.
0 = None TC1 interrupt request.
1 = TC1 interrupt request.

Bit 6 **TC0IRQ:** TC0 timer interrupt request flag.
0 = None TC0 interrupt request.
1 = TC0 interrupt request.

Bit 5 **T1IRQ:** T1 timer interrupt request flag.
0 = None T1 interrupt request.
1 = T1 interrupt request.

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

Bit 3 **RXIRQ:** UART receive interrupt request flag.
0 = None UART receive interrupt request.
1 = UART receive interrupt request.

Bit 2 **TXIRQ:** UART transmit interrupt request flag.
0 = None UART transmit interrupt request.
1 = UART transmit interrupt request.

Bit 1 **P01IRQ:** External P01 interrupt (INT01) request flag.
0 = None INT01 interrupt request.
1 = INT01 interrupt request.

Bit 0 **P00IRQ:** External P00 interrupt (INT00) request flag.
0 = None INT00 interrupt request.
1 = INT00 interrupt request.

0C6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ1	SIOIRQ	ADCIRQ	P15IRQ	P14IRQ	P13IRQ	P12IRQ	P11IRQ	P10IRQ
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **SIOIRQ:** SIO interrupt request flag.
0 = None SIO interrupt request.
1 = SIO interrupt request.

Bit 6 **ADCIRQ:** ADC interrupt request flag.
0 = None ADC interrupt request.
1 = ADC interrupt request.

Bit 5 **P15IRQ:** External P15 interrupt (INT15) request flag.
0 = None INT15 interrupt request.
1 = INT15 interrupt request.

Bit 4 **P14IRQ:** External P14 interrupt (INT14) request flag.
0 = None INT14 interrupt request.
1 = INT14 interrupt request.

Bit 3 **P13IRQ:** External P13 interrupt (INT13) request flag.
0 = None INT13 interrupt request.
1 = INT13 interrupt request.

Bit 2 **P12IRQ:** External P12 interrupt (INT12) request flag.
0 = None INT12 interrupt request.
1 = INT12 interrupt request.

Bit 1 **P11IRQ:** External P11 interrupt (INT11) request flag.
0 = None INT11 interrupt request.
1 = INT11 interrupt request.

Bit 0 **P10IRQ:** External P10 interrupt (INT10) request flag.
0 = None INT10 interrupt request.
1 = INT10 interrupt request.

0F5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ2							T3IRQ	T2IRQ
Read/Write							R/W	R/W
After reset							0	0

Bit 1 **T3IRQ:** T3 timer interrupt request flag.
0 = None T3 interrupt request.
1 = T3 interrupt request.

Bit 0 **T2IRQ:** T2 timer interrupt request flag.
0 = None T2 interrupt request.
1 = T2 interrupt request.

6.4 GIE GLOBAL INTERRUPT OPERATION

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

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	-	-	STKPB4	STKPB3	STKPB2	STKPB1	STKPB0
Read/Write	R/W	-	-	R/W	R/W	R/W	R/W	R/W
After reset	0	-	-	1	0	1	1	1

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

Bit[4:0] **STKPBn:** Stack pointer (n = 0 ~ 4)

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

```
B0BSET      FGIE        ; Enable GIE
```

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

6.5 PUSH, POP ROUTINE

When any interrupt requests occurs, the system provides to jump to interrupt vector and execute interrupt routine. The first procedure is “PUSH” operation. The end procedure after interrupt service routine execution is “POP” operation. The “PUSH” and “POP” operations aren’t through instruction (PUSH, POP) and executed by hardware automatically.

- **“PUSH” operation: PUSH operation saves the contents of ACC and working registers (0x80~0x87) into hardware buffers. PUSH operation executes before program counter points to interrupt vector. The RAM bank keeps the status of main routine and doesn’t switch to bank 0 automatically. The RAM bank is selected by program.**
- **“POP” operation: POP operation reloads the contents of ACC and working registers (0x80~0x87) from hardware buffers. POP operation executes as RETI instruction executed. The RAM bank switches to last status of main routine after reloading RBANK content.**
- **0x80~0x87 working registers include L, H, R, Z, Y, X, PFLAG, RBANK.**
- **PFLAG working register without “PUSH” and “POP” NT0, NPD, LVD36, LVD24 bits.**

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

```
.DATA

.CODE
                ORG     0
                JMP     START

                ORG     8               (load ACC and working registers (0x80~0x87) by hardware automatically)
                JMP     INT_SERVICE

                ORG     10H
START:
                …

INT_SERVICE:
                …
                …

                …


                …
                RETI                    ; Exit interrupt service vector
                                        (load ACC and working registers (0x80~0x87) by hardware automatically)
                …
                ENDP
```

6.6 EXTERNAL INTERRUPT OPERATION (INT00~INT01)

Sonix provides 2 sets external interrupt sources in the micro-controller.INT0nd INT1 are external interrupt trigger sources and build in edge trigger configuration function. When the external edge trigger occurs, the external interrupt request flag will be set to “1” when the external interrupt control bit enabled. If the external interrupt control bit is disabled, the external interrupt request flag won’t active when external edge trigger occurrence. When external interrupt control bit is enabled and external interrupt edge trigger is occurring, the program counter will jump to the interrupt vector (ORG 8) and execute interrupt service routine.

The external interrupt builds in wake-up latch function. That means when the system is triggered wake-up from power down mode, the wake-up source is external interrupt source (P00 or P01), and the trigger edge direction matches interrupt edge configuration, the trigger edge will be latched, and the system executes interrupt service routine fist after wake-up.

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	-	-	P01G1	P01G0	P00G1	P00G0
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	1	0	1	0

Bit[3:2] **P01G[1:0]:** INT1 edge trigger select bits.
00 = reserved,
01 = rising edge,
10 = falling edge,
11 = rising/falling bi-direction.

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

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

```
        MOV     A, #98H
        B0MOV   PEDGE, A        ; Set INT0 interrupt trigger as bi-direction edge.

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

Example: INT0 interrupt service routine.

```
        ORG     8               ; Interrupt vector
        JMP     INT_SERVICE
INT_SERVICE:
        …
        …
        B0BTS1  FP00IRQ         ; Check P00IRQ
        JMP     EXIT_INT        ; P00IRQ = 0, exit interrupt vector

        B0BCLR  FP00IRQ         ; Reset P00IRQ
        …                       ; INT0 interrupt service routine
EXIT_INT:
        …
        RETI                    ; Exit interrupt vector
```

6.7 INT10~INT15 (P10~P15) INTERRUPT OPERATION

When the INT10~INT15 trigger occurs, the P10IRQ~P15IRQ will be set to "1" no matter the P10IEN~P15IEN is enable or disable. If the P10IEN~P15IEN = 1 and the trigger event P10IRQ~P15IRQ is also set to be "1". As the result, the system will execute the interrupt vector (ORG 8). If the P10IEN~P15IEN = 0 and the trigger event P10IRQ~P15IRQ is still set to be "1". Moreover, the system won't execute interrupt vector even when the P10IRQ~P15IRQ is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

✷ ***Note: The interrupt trigger direction of P10~P15 is falling edge.***

➢ **Example: INT10 interrupt request setup.**

```
        B0BSET      FP10IEN     ; Enable INT10 interrupt service
        B0BCLR      FP10IRQ     ; Clear INT10 interrupt request flag
        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: INT10 interrupt service routine.**

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

        …

        B0BTS1      FP10IRQ     ; Check P10IRQ
        JMP         EXIT_INT    ; P10IRQ = 0, exit interrupt vector

        B0BCLR      FP10IRQ     ; Reset P10IRQ
        …                       ; INT10 interrupt service routine
        …
EXIT_INT:

        …
        RETI                    ; Exit interrupt vector
```

6.8 T0 INTERRUPT OPERATION

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

➢ **Example: T0 interrupt request setup.**

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

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

        B0BSET      FGIE        ; Enable GIE
```

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

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

        …

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

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

        RETI                    ; Exit interrupt vector
```

6.9 T1 INTERRUPT OPERATION

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

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

```
        B0BCLR      FT1IEN      ; Disable T1 interrupt service
        B0BCLR      FT1ENB      ; Disable T1 timer
        MOV         A, #20H     ;
        B0MOV       T1M, A      ; Set T1 clock = Fcpu / 64 and falling edge trigger.
        CLR         T1C

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

        B0BSET      FGIE        ; Enable GIE
```

➢ **Example: T1 interrupt service routine.**

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

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

        B0BCLR      FT1IRQ      ; Reset T1IRQ
        B0MOV       A, T1C
        B0MOV       T1CBUF, A   ; Save pulse width.
        CLR         T1C
        …                       ; T1 interrupt service routine
        …
EXIT_INT:

        …
        RETI                    ; Exit interrupt vector
```

6.10 TC0 INTERRUPT OPERATION

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

➢ **Example: TC0 interrupt request setup.**

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

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

        B0BSET      FGIE            ; Enable GIE
```

➢ **Example: TC0 interrupt service routine.**

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

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

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

        …
        RETI                        ; Exit interrupt vector
```

6.11 TC1 INTERRUPT OPERATION

When the TC1C counter overflows, the TC1IRQ will be set to “1” no matter the TC1IEN is enable or disable. If the TC1IEN and the trigger event TC1IRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the TC1IEN = 0, the trigger event TC1IRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the TC1IEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

➢ **Example: TC0 interrupt request setup.**

```
        B0BCLR      FTC1IEN         ; Disable TC1 interrupt service
        B0BCLR      FTC1ENB         ; Disable TC1 timer
        MOV         A, #20H         ;
        B0MOV       TC1M, A
        MOV         A, #74H         ; Set TC1C initial value = 74H
        B0MOV       TC1C, A         ; Set TC1 interval = 10 ms

        B0BSET      FTC1IEN         ; Enable TC1 interrupt service
        B0BCLR      FTC1IRQ         ; Clear TC1 interrupt request flag
        B0BSET      FTC1ENB         ; Enable TC1 timer

        B0BSET      FGIE            ; Enable GIE
```

➢ **Example: TC1 interrupt service routine.**

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

        B0BTS1      FTC1IRQ         ; Check TC1IRQ
        JMP         EXIT_INT        ; TC1IRQ = 0, exit interrupt vector

        B0BCLR      FTC1IRQ         ; Reset TC1IRQ
        MOV         A, #74H
        B0MOV       TC1C, A         ; Reset TC1C.
        …                           ; TC1 interrupt service routine
        …
EXIT_INT:

        …
        RETI                        ; Exit interrupt vector
```

6.12 MULTI-INTERRUPT OPERATION

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

Interrupt Name	*Trigger Event Description*
T0IRQ	T0C overflow.
T1IRQ	T1C overflow.
TC0IRQ	TC0C overflow.
TC1IRQ	TC1C overflow.
RXIRQ/TXIRQ	UART transmitter successfully.
ADCIRQ	ADC transmitter successfully.
SIOIRQ	SIO transmitter successfully.
P00IRQ~P01IRQ	P00~P01 trigger controlled by PEDGE
P10IRQ~P15IRQ	P10~P15 rising/falling bi-direction trigger.

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

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

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

INTT0CHK:                                   ; Check T0 interrupt request
                B0BTS1      FT0IEN          ; Check T0IEN
                JMP         INTTC0CHK       ; Jump check to next interrupt
                B0BTS0      FT0IRQ          ; Check T0IRQ
                JMP         INTT0           ; Jump to T0 interrupt service routine
INTTC0CHK:                                  ; Check TC0 interrupt request
                B0BTS1      FTC0IEN         ; Check TC0IEN
                JMP         INT_EXIT        ; Jump to exit of IRQ
                B0BTS0      FTC0IRQ         ; Check TC0IRQ
                JMP         INTTC0          ; Jump to TC0 interrupt service routine
INT_EXIT:

                …

                RETI                        ; Exit interrupt vector
```

7 I/O PORT

7.1 I/O PORT MODE

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

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

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	-	-	P15M	P14M	P13M	P12M	P11M	P10M
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	0	0	0	0	0	0

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

0C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	-	P46M	P45M	P44M	P43M	P42M	P41M	P40M
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

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	P56M	P55M	P54M	P53M	P52M	P51M	P50M
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[7:0] **PnM[7:0]:** Pn mode control bits. (n = 0~5).
0 = Pn is input mode.
1 = Pn is output mode.

➢ **Example: I/O mode selecting**

```
CLR     P1M         ; Set all ports to be input mode.
CLR     P4M
CLR     P5M

MOV     A, #0FFH    ; Set all ports to be output mode.
B0MOV   P1M, A
B0MOV   P4M,A
B0MOV   P5M, A

B0BCLR  P4M.0       ; Set P4.0 to be input mode.

B0BSET  P4M.0       ; Set P4.0 to be output mode.
```

7.2 I/O PULL UP REGISTER

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

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR			P15R	P14R	P13R	P12R	P11R	P10R
Read/Write			W	W	W	W	W	W
After reset			0	0	0	0	0	0

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

0E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR		P46R	P45R	P44R	P43R	P42R	P41R	P40R
Read/Write		W	W	W	W	W	W	W
After reset		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		P56R	P55R	P54R	P53R	P52R	P51R	P50R
Read/Write		W	W	W	W	W	W	W
After reset		0	0	0	0	0	0	0

➢ **Example: I/O Pull up Register**

```
MOV     A, #0FFH        ; Enable Port1, 4, 5 Pull-up register,
B0MOV   P1UR, A         ;
B0MOV   P4UR,A
B0MOV   P5UR, A
```

7.3 I/O PORT DATA REGISTER

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

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1			P15	P14	P13	P12	P11	P10
Read/Write			R/W	R/W	R/W	R/W	R/W	R/W
After reset			0	0	0	0	0	0

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

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4		P46	P45	P44	P43	P42	P41	P40
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

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5		P56	P55	P54	P53	P52	P51	P50
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

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

```
B0MOV   A, P1     ; Read data from Port 1
B0MOV   A, P4     ; Read data from Port 4
B0MOV   A, P5     ; Read data from Port 5
```

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

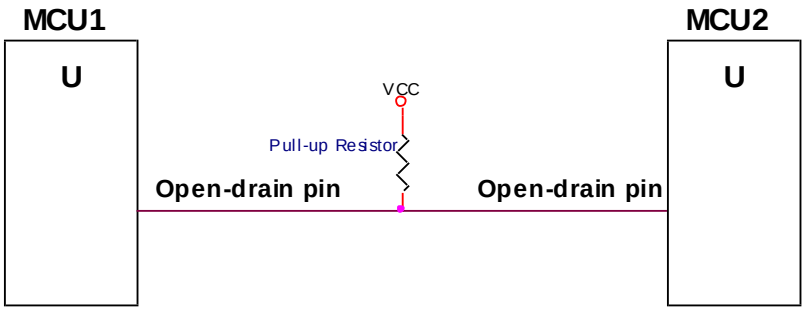
```
MOV     A, #0FFH  ; Write data FFH to all Port.
B0MOV   P1, A
B0MOV   P4, A
B0MOV   P5, A
```

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

```
B0BSET  P4.0      ; Set P4.0 and P5.3 to be “1”.
B0BSET  P5.3

B0BCLR  P4.0      ; Set P4.0 and P5.3 to be “0”.
B0BCLR  P5.3
```

7.4 I/O OPEN-DRAIN REGISTER

P10~P13/P50~P52 is built-in open-drain function. The I/O must be set as output mode when enable open-drain function. Open-drain external circuit is as following.



The pull-up resistor is necessary. Open-drain output high is driven by pull-up resistor. Output low is sunken by MCU's pin.

0E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1OC		P52OC	P51OC	P50OC	P13OC	P12OC	P11OC	P10OC
Read/Write		W	W	W	W	W	W	W
After reset		0	0	0	0	0	0	0

Bit 6 **P52OC:** P5.2 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 5 **P51OC:** P5.1 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 4 **P50OC:** P5.0 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 3 **P13OC:** P1.3 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 2 **P12OC:** P1.2 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 1 **P11OC:** P1.1 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

Bit 0 **P10OC:** P1.0 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

✸ ***Note: P12,P13 no support Open Drain function in ICE emulate.***

➢ **Example: Enable P1.0 to open-drain mode and output high.**

```
B0BSET      P1.0        ; Set P1.0 buffer high.

MOV         A, #01H     ; Enable P1.0 open-drain function.
B0MOV       P1OC, A
B0BSET      P10M        ; Enable P1.0 output mode.
```

✸ ***Note: P1OC is write only register. Setting P10OC must be used "MOV" instructions.***

➢ **Example: Disable P1.0 to open-drain mode and output low.**

```
MOV         A, #0       ; Disable P1.0 open-drain function.
B0MOV       P1OC, A
```

✸ ***Note: After disable open-drain function, I/O mode returns to last I/O mode.***

7.5 PORT 4 ADC SHARE PIN

The Port 4 is shared with ADC input function and no Schmitt trigger structure. Only one pin of port 4 can be configured as ADC input in the same time by ADM register. The other pins of port 4 are digital I/O pins. Connect an analog signal to COMS digital input pin, especially the analog signal level is about 1/2 VDD will cause extra current leakage. In the power down mode, the above leakage current will be a big problem. Unfortunately, if users connect more than one analog input signal to port 4 will encounter above current leakage situation. P4CON is Port4 Configuration register. Write "1" into P4CON.n will configure related port 4 pin as pure analog input pin to avoid current leakage.

0BEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON		P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
Read/Write		W	W	W	W	W	W	W
After reset		0	0	0	0	0	0	0

Bit[3:0] **P4CON[6:0]:** P4.n configuration control bits.
0 = P4.n can be an analog input (ADC input) or digital I/O pins.
1 = P4.n is pure analog input, can't be a digital I/O pin.

✸ ***Note: When Port 4.n is general I/O port not ADC channel, P4CON.n must set to "0" or the Port 4.n digital I/O signal would be isolated.***

Port 4 ADC analog input is controlled by GCHS and CHSn bits of ADM register. If GCHS = 0, P4.n is general purpose bi-direction I/O port. If GCHS = 1, P4.n pointed by CHSn is ADC analog signal input pin. Users should set P4 ADC input pin as input mode without pull-up.

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0
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 4 **GCHS:** Global channel select bit.
0 = Disable AIN channel.
1 = Enable AIN channel.

Bit[2:0] **CHS[2:0]:** ADC input channels select bit.
000 = AIN0, 001 = AIN1, … 110 = AIN6, 111 = AIN7.

✸ ***Note: For P4.n general purpose I/O function, users should make sure of P4.n's ADC channel is disabled.***

➢ **Example: Set P4.1 to be general purpose input mode. P4CON.1 must be set as "0".**

```
; Check GCHS and CHS[2:0] status.
        B0BCLR      FGCHS       ;If CHS[2:0] point to P4.1 (CHS[2:0] = 001B), set GCHS=0
                                ;If CHS[2:0] don't point to P4.1 (CHS[2:0] ≠ 001B), don't
                                care GCHS status.

; Clear P4CON.1
        B0BCLR      P4CON.1     ; Enable P4.1 digital function.

; Enable P4.1 input mode.
        B0BCLR      P4M.1       ; Set P4.1 as input mode.
```

➢ **Example: Set P4.1 to be general purpose output. P4CON.1 must be set as "0".**

```
; Check GCHS and CHS[2:0] status.
        B0BCLR      FGCHS       ;If CHS[2:0] point to P4.1 (CHS[2:0] = 001B), set GCHS=0.
                                ;If CHS[2:0] don't point to P4.1 (CHS[2:0] ≠ 001B), don't
                                care GCHS status.

; Clear P4CON.
        B0BCLR      P4CON.1     ; Enable P4.1 digital function.

; Set P4.1 output buffer to avoid glitch.
        B0BSET      P4.1        ; Set P4.1 buffer as "1".
; or
        B0BCLR      P4.1        ; Set P4.1 buffer as "0".

; Enable P4.1 output mode.
        B0BSET      P4M.1       ; Set P4.1 as output mode.
```

8 TIMERS

8.1 WATCHDOG TIMER (WDT)

The watchdog timer (WDT) is a binary up counter designed for monitoring program execution. If the program goes into the unknown status by noise interference, WDT overflow signal raises and resets MCU. Watchdog clock controlled by code option and the clock source is internal low-speed oscillator (10KHz @3V).

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

VDD	*Internal Low RC Freq.*	*Watchdog Overflow Time*
3V	10KHz	819.2ms

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

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

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

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

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

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

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

Watchdog timer application note is as following.

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

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

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

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

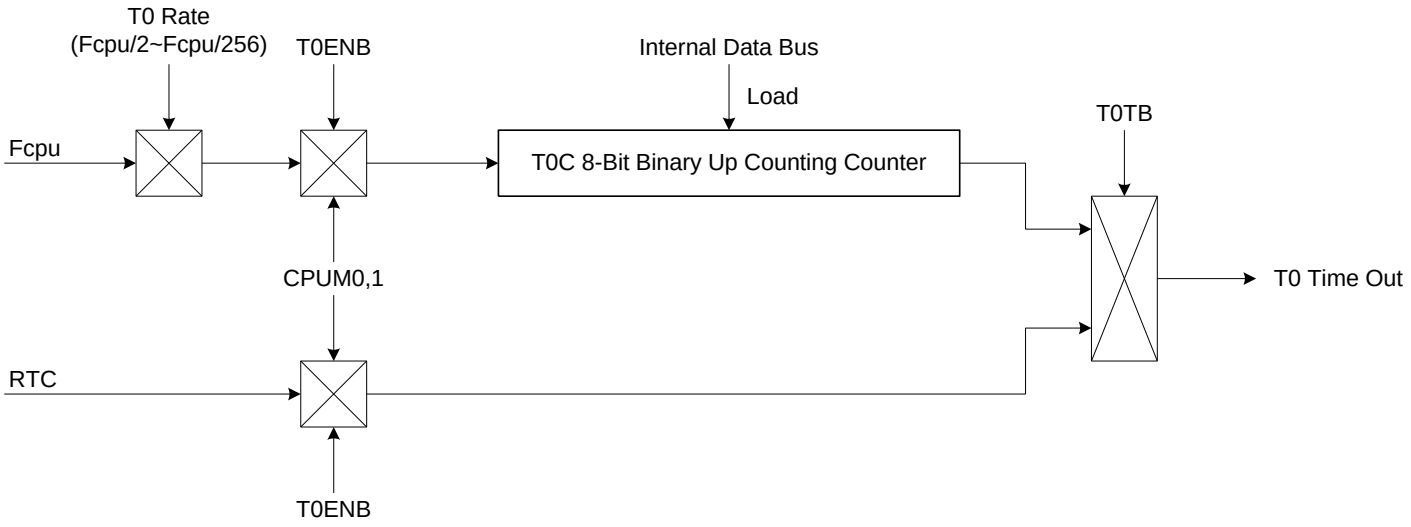
8.2 TIMER 0 (T0)

8.2.1 OVERVIEW

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

The main purposes of the T0 timer 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.***

8.2.2 T0M MODE REGISTER

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

Bit7 **T0ENB:** T0 timer control bit. 0 = disable, 1 = enable.

Bit[6:4] **T0RATE2~T0RATE0:** The T0 timer's clock source selects bits.

T0RATE [2:0]	T0 Clock Source
000	$F_{CPU}/256$
001	$F_{CPU}/128$
…	…
110	$F_{CPU}/4$
111	$F_{CPU}/2$

Bit3 **TC1X8:** Multiple TC1 timer speed eight times. Refer TC1M register for detailed information.
0 = Disable
1 = Enable

Bit2 **TC0X8:** Multiple TC0 timer speed eight times. Refer TC0M register for detailed information.
0 = Disable
1 = Enable

Bit1 **TC0GN:** Enable TC0 green mode wakeup function
0 = Disable
1 = Enable

Bit0 **T0TB:** Timer 0 as the Real-Time clock time base.
0 = Timer 0 function as a normal timer system.
1 = Timer 0 function as a Real-Time Clock. The clock source of timer 0 will be switched to external low clock (32.768K crystal oscillator).

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

8.2.3 T0C COUNTING REGISTER

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

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

The equation of T0C initial value is as following.

T0C 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.***

8.2.4 T0 TIMER OPERATION SEQUENCE

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

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

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

☞ **Set T0 timer rate.**

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

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

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

or

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

☞ **Set T0 interrupt interval time.**

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

☞ **Set T0 timer function mode.**

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

☞ **Enable T0 timer.**

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

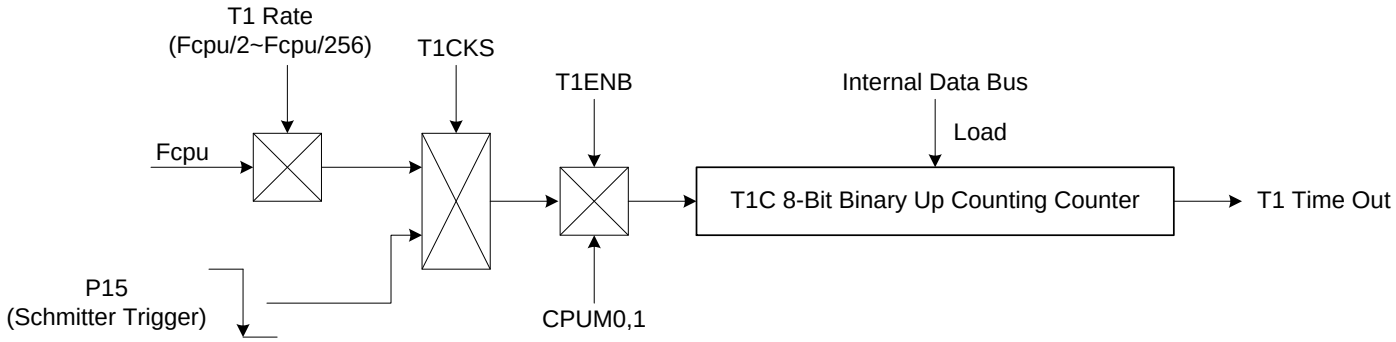
8.3 TIMER 1 (T1)

8.3.1 OVERVIEW

The T1 is an 8-bit binary up timer and event counter. If T1 timer occurs an overflow (from FFH to 00H), 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.

- ☞ **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 P15 input pin.



8.3.2 T1M MODE REGISTER

0D6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1rate2	T1rate1	T1rate0	T1CKS			
Read/Write	R/W	R/W	R/W	R/W	R/W			
After reset	0	0	0	0	0			

Bit7 **T1ENB:** T1 timer control bit.
0 = disable
1 = enable.

Bit[6:4] **T1RATE2~T1RATE0:** The T1 timer's clock source selects bits.

T1RATE [2:0]	T1 Clock Source
000	$F_{CPU}/256$
001	$F_{CPU}/128$
…	…
110	$F_{CPU}/4$
111	$F_{CPU}/2$

Bit 3 **T1CKS:** T1 clock source select bit.
0 = Internal clock.
1 = External clock from P15 pin.

8.3.3 T1C COUNTING REGISTER

T1C is an 8-bit counter register for T1 interval time control.

0D7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1C	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0
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 initial value is as following.

T1C initial value = 256 - (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 = 256 - (T1 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 T1.

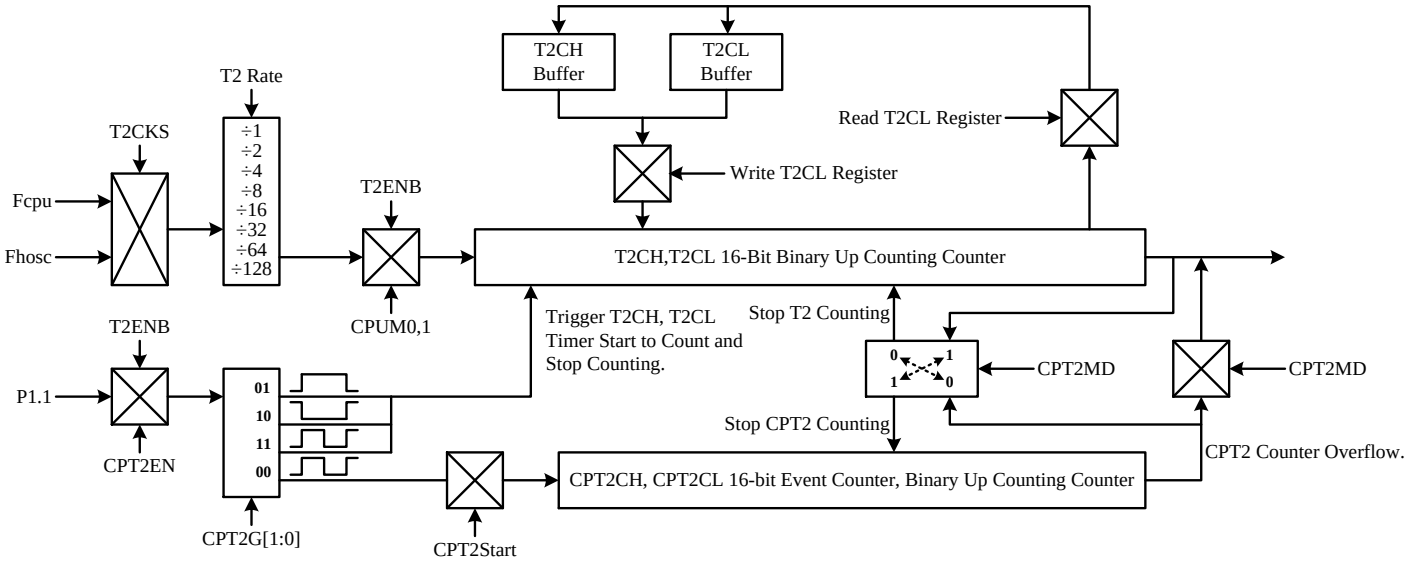
T1RATE	T1CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fcpu/256	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.125 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

8.4 T2 16-bit Timer with Capture Timer Function

8.4.1 OVERVIEW

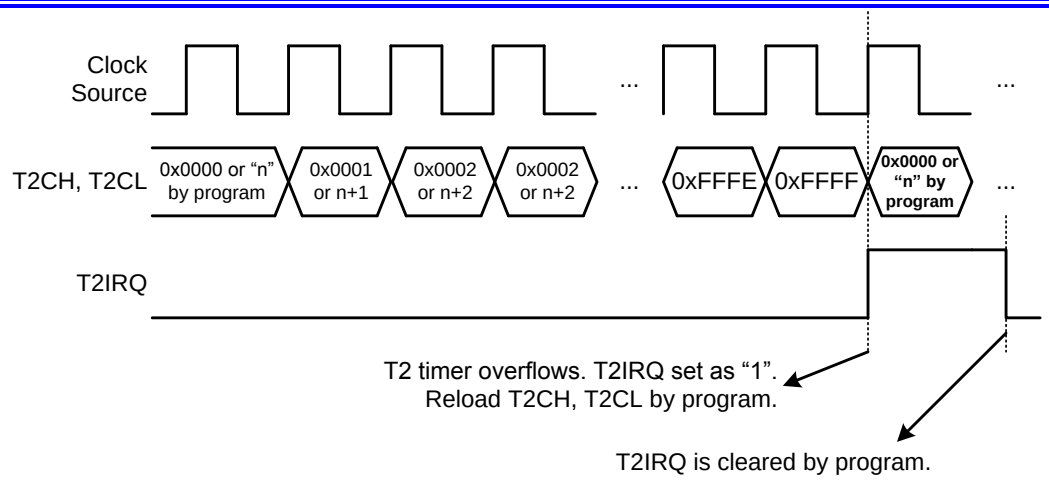
The T2 timer is a 16-bit binary up timer with basic timer and capture timer functions. The basic timer function supports flag indicator (T2IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T2M, T2CH/T2CL 16-bit counter registers. The capture timer supports high pulse width measurement, low pulse width measurement, cycle measurement and continuous duration from P1.1. T2 becomes a timer meter to count external signal time parameters to implement measure application. The main purposes of the T2 timer are as following.

- **16-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **16-bit measurement:** Measure the input signal pulse width and cycle depend on the T2 clock time base to decide the capture timer's resolution. The capture timer builds in programmable trigger edge selection to decide the start-stop trigger event.
- **16-bit capture timer:** The 16-bit event counter to detect event source for accumulative capture timer function. The event counter is up counting design.
- **Interrupt function:** T2 timer function and capture timer function support interrupt function. When T2 timer occurs overflow or capture timer stops counting, the T2IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Green mode function:** All T2 functions (timer, capture timer…) keeps running in green mode, but no wake-up function. Timer IRQ actives as any IRQ trigger occurrence, e.g. timer overflow…



8.4.2 T2 TIMER OPERATION

T2 timer is controlled by T2ENB bit. When T2ENB=0, T2 timer stops. When T2ENB=1, T2 timer starts to count. Before enabling T2 timer, setup T2 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…T2 16-bit counter (T2CH, T2CL) increases "1" by timer clock source. When T2 overflow event occurs, T2IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is T2CH, T2CL count from full scale (0xFFFF) to zero scale (0x0000). T2 doesn't build in double buffer, so load T2CH, T2CL by program when T2 timer overflows to fix the correct interval time. If T2 timer interrupt function is enabled (T2IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 0008H) and executes interrupt service routine after T2 overflow occurrence. Clear T2IRQ by program is necessary in interrupt procedure. T2 timer can works in normal mode, slow mode and green mode.



T2 provides different clock sources to implement different applications and configurations. T2 clock source includes Fcpu (instruction cycle) and Fhosc (high speed oscillator) controlled by T2CKS bit. T2CKS bit selects the clock source is from Fcpu or Fhosc. If T2CKS=0, T2 clock source is Fcpu through T2rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. If T2CKS=1, T2 clock source is Fhosc through T2 rate[2:0] pre-scalar to decide Fhosc/1~Fhosc/128. T2 length is 16-bit (65536 steps), and the one count period is each cycle of input clock.

T2CKS	T2rate[2:0]	T2 Clock	T2 Interval Time			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	2097.152	32	8388.608	128
0	001b	Fcpu/64	1048.576	16	4194.304	64
0	010b	Fcpu/32	524.288	8	2097.152	32
0	011b	Fcpu/16	262.144	4	1048.576	16
0	100b	Fcpu/8	131.072	2	524.288	8
0	101b	Fcpu/4	65.536	1	262.144	4
0	110b	Fcpu/2	32.768	0.5	131.072	2
0	111b	Fcpu/1	16.384	0.25	65.536	1
1	000b	Fhosc/128	524.288	8	2097.152	32
1	001b	Fhosc/64	262.144	4	1048.576	16
1	010b	Fhosc/32	131.072	2	524.288	8
1	011b	Fhosc/16	65.536	1	262.144	4
1	100b	Fhosc/8	32.768	0.5	131.072	2
1	101b	Fhosc/4	16.384	0.25	65.536	1
1	110b	Fhosc/2	8.192	0.125	32.768	0.5
1	111b	Fhosc/1	4.096	0.0625	16.384	0.25

8.4.3 T2M MODE REGISTER

T2M is T2 timer mode control register to configure T2 operating mode including T2 pre-scalar, clock source, capture parameters…These configurations must be setup completely before enabling T2 timer.

0EAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2M	T2ENB	T2rate2	T2rate1	T2rate0	T2CKS			
Read/Write	R/W	R/W	R/W	R/W	R/W			
After reset	0	0	0	0	0			

Bit 7 **T2ENB:** T2 counter control bit.
0 = Disable T2 timer.
1 = Enable T2 timer.

Bit [6:4] **T2RATE[2:0]:** T2 timer clock source select bits.
T2CKS=0 -> 000 = Fcpu/128, 001 = Fcpu/64, 010 = Fcpu/32, 011 = Fcpu/16, 100 = Fcpu/8, 101 = Fcpu/4, 110 = Fcpu/2,111 = Fcpu/1.
T2CKS=1 -> 000 = Fhosc/128, 001 = Fhosc/64, 010 = Fhosc/32, 011 = Fhosc/16, 100 = Fhosc/8, 101 = Fhosc/4, 110 = Fhosc/2,111 = Fhosc/1.

Bit 3 **T2CKS:** T2 clock source control bit.
0 = Fcpu.
1 = Fhosc.

8.4.4 T2CH, T2CL 16-bit COUNTING REGISTERS

T2 counter is 16-bit counter combined with T2CH and T2CL registers. When T2 timer overflow occurs, the T2IRQ flag is set as "1" and cleared by program. The T2CH, T2CL decide T2 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T2CH and T2CL registers, and then enable T2 timer to make sure the fist cycle correct. After one T2 overflow occurs, the T2CH and T2CL registers are loaded correct values by program.

0EBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2CL	T2CL7	T2CL6	T2CL5	T2CL4	T2CL3	T2CL2	T2CL1	T2CL0
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

0ECH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2CH	T2CH7	T2CH6	T2CH5	T2CH4	T2CH3	T2CH2	T2CH1	T2CH0
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 T2 timer counter length is 16-bit and points to T2CH and T2CL registers. The timer counter is double buffer design. The core bus is 8-bit, so access 16-bit data needs a latch flag to avoid the transient status affect the 16-bit data mistake occurrence. Under write mode, the write T2CH is the latch control flag. Under read mode, the read T2CL is the latch control flag. So, write T2 16-bit counter is to write T2CH first, and then write T2CL. The 16-bit data is written to 16-bit counter buffer after executing writing T2CL. Read T2 16-bit counter is to read T2CL first, and then read T2CH. The 16-bit data is dumped to T2CH, T2CL after executing reading T2CH.

- **Read T2 counter buffer sequence is to read T2CL first, and then read T2CH.**
- **Write T2 counter buffer sequence is to write T2CH first, and then write T2CL.**

The equation of T2 16-bit counter (T2CH, T2CL) initial value is as following.

T2CH, T2CL initial value = 65536 - (T2 interrupt interval time * T2 clock rate)

➢ **Example: To calculation T2CH and T2CL values to obtain 500ms T2 interval time. T2 clock source is Fcpu = 16MHz/16 = 1MHz. Select T2RATE=000 (Fcpu/128).**
T2 interval time = 500ms. T2 clock rate = 16MHz/16/128

*T2 16-bit counter initial value = 65536 - (T2 interval time * input clock)*
*= 65536 - (500ms * 16MHz / 16 / 128)*
*= 65536 - (500*10^{-3} * 16 * 10^{6} / 16 / 128)*
= F0BDH ***(T2CH = F0H, T2CL = BDH)***

8.4.5 T2 CPATURE TIMER

The 16-bit capture timer is controlled by CPT2EN bit, but the T2 must be enabled. Set T2ENB=1 and CPT2EN=1 to enable capture timer function. The capture timer is a pure counter and no clock source to decide interval time. Capture timer input source is P1.1 pin. CPT2G[1:0] bits select capture timer functions.

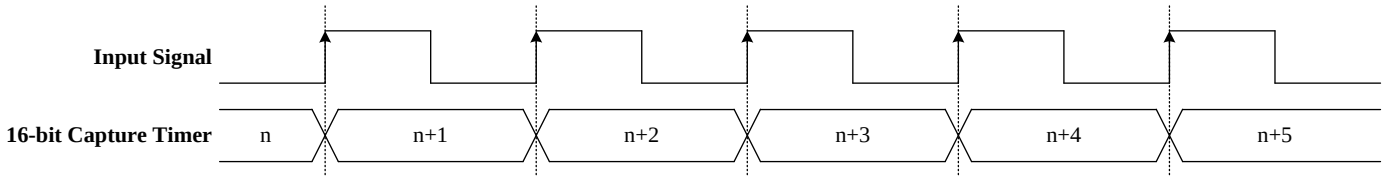
- **CPT2G[1:0] = 00: P1.1 Capture Timer Function.**
- **CPT2G[1:0] = 01: Measure P1.1 high pulse width.**
- **CPT2G[1:0] = 10: Measure P1.1 low pulse width.**
- **CPT2G[1:0] = 11: Measure P1.1 cycle.**

These functions must be combined T2 timer function to implement. The capture timer can measure high pulse width, low pulse width, cycle and capture duration of input signal (P1.1) controlled by CPT2G[1:0]. CPT2Start bit is to execute capture timer function. When CPT2Start is set as "1", the capture timer waits the right trigger edge to active 16-bit counter. The trigger edge finds, and the 16-bit counter starts to count which clock source is T2. When the second right edge finds, the 16-counter stops, CPT2Start is cleared and the T2IRQ actives.

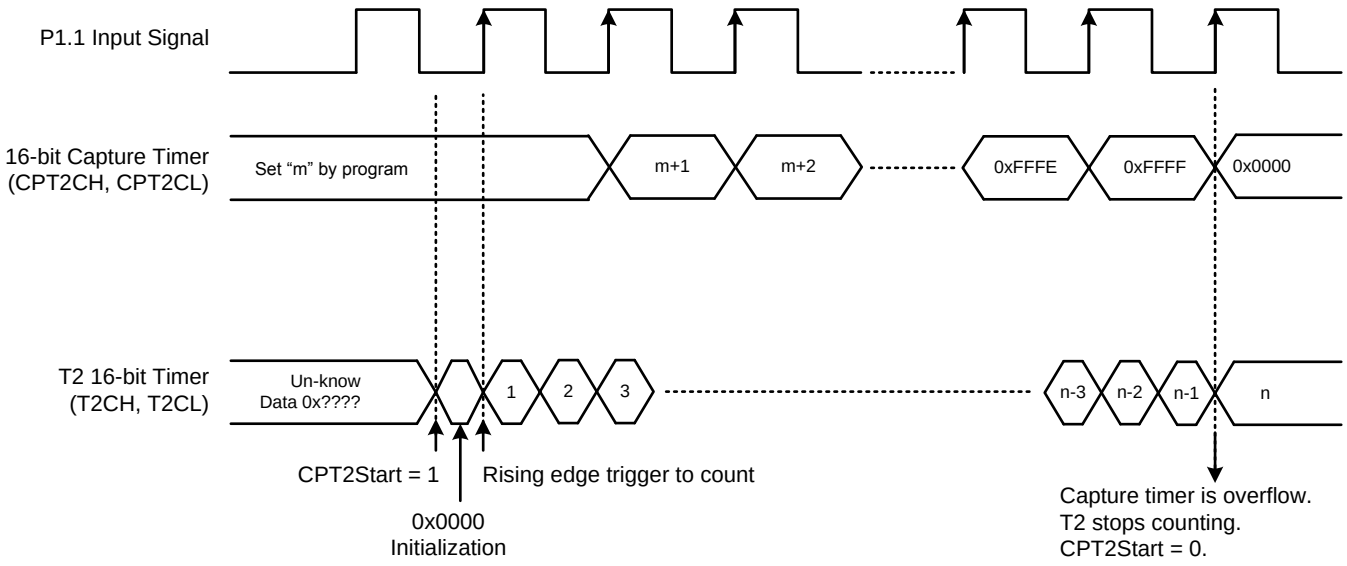
8.4.5.1 Capture Timer

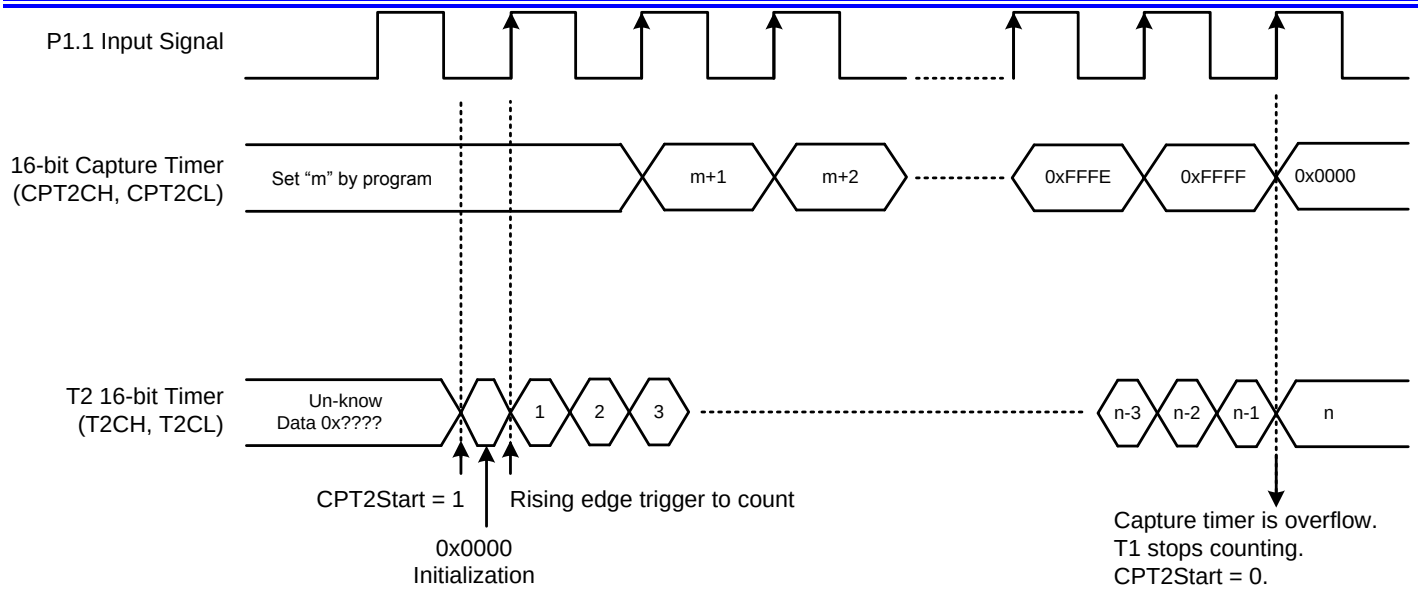
The capture timer function controlled by CPT2G[1:0] bits. Set CPT2G[1:0] = 00 to enable capture timer function. The capture timer function's purpose is to measure the period of a continuous signal. The function includes two modes for difference speed signal controlled by CPT2MD bit. To start capture timer operation is set CPT2Start bit as "1", and the trigger source is the first rising edge of the P1.1 input signal. Before the first rising edge, the capture timer and T2 timer keeps ideal status and wait the riding edge event. When catch the first edge, the capture timer and T2 timer start to count. Each of overflow event occurs (controlled by CPT2MD bit), the capture timer and T2 timer stop counting, CPT2Start bit is cleared, and T2IRQ is set as "1". If T2IEN = 1, the system executes T2 interrupt function and service routine.

- **Capture timer counting trigger source is the rising edge of input signal.**



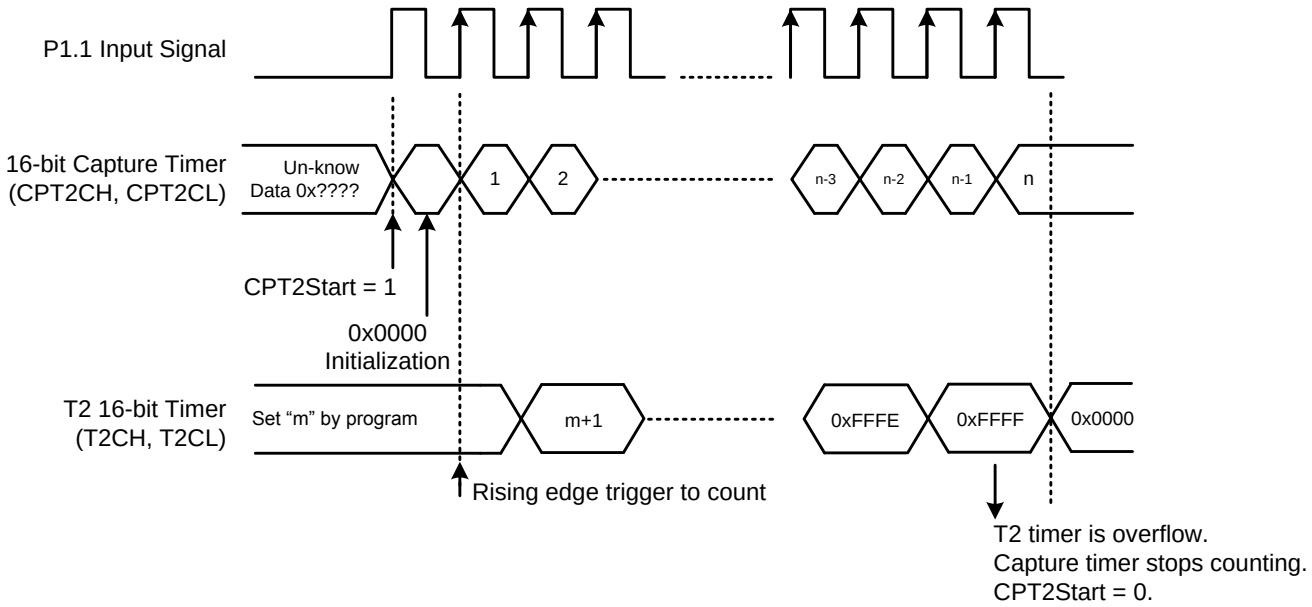
- **CPTMD = 0, Low-speed mode (T2ENB = 1. CPT2EN = 1. CPT2G[1:0] = 00.)**

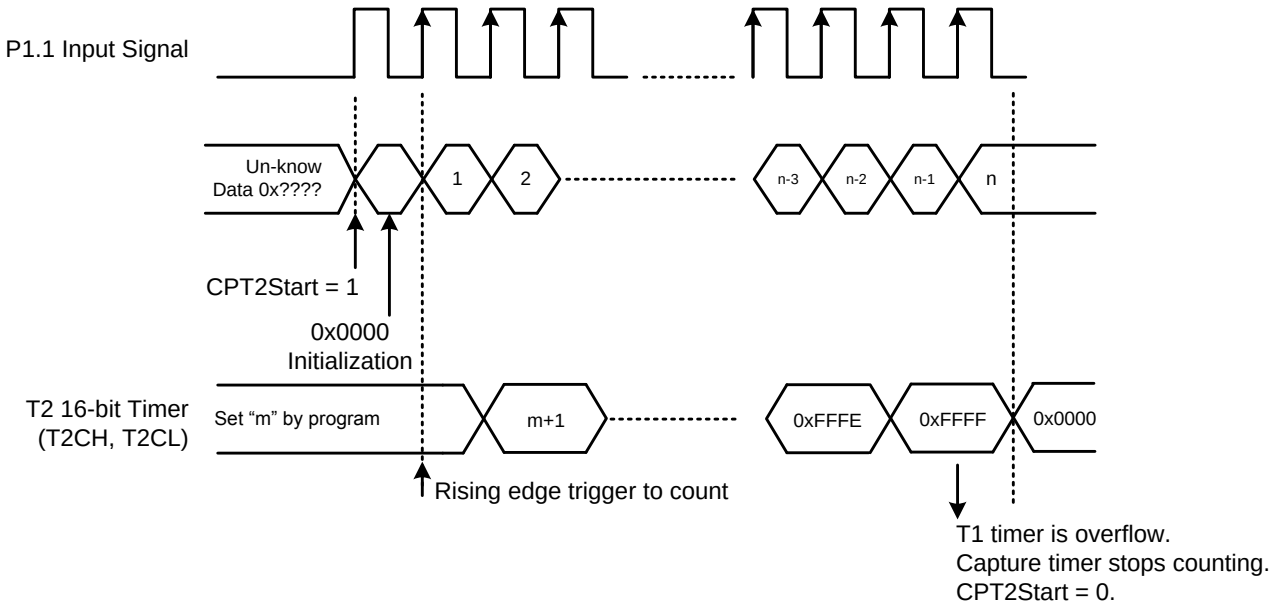




Input signal rate < T2 timer rate. Use T2 timer to measure input signal continuous duration. Set capture timer initial value (CPT2CH, CPT2CL = "m") and clear T2 counter (T2CH, T2CL = 0x0000) by program. Set CPT2Satrt bit ("1") to start capture timer counting. Capture timer and T2 start counting at the first rising edge of input signal. When capture timer overflow occurs (0xFFFF to 0x0000), T2 stops counting, CPT2Start is cleared ("0") automatically, and the T2IRQ sets as "1". The T2 16-bit counter value (T2CH, T2CL = "n") is the continuous signal's duration.

- **CPT2MD = 1, High-speed mode (T2ENB = 1. CPT2EN = 1. CPT2G[1:0] = 00.)**

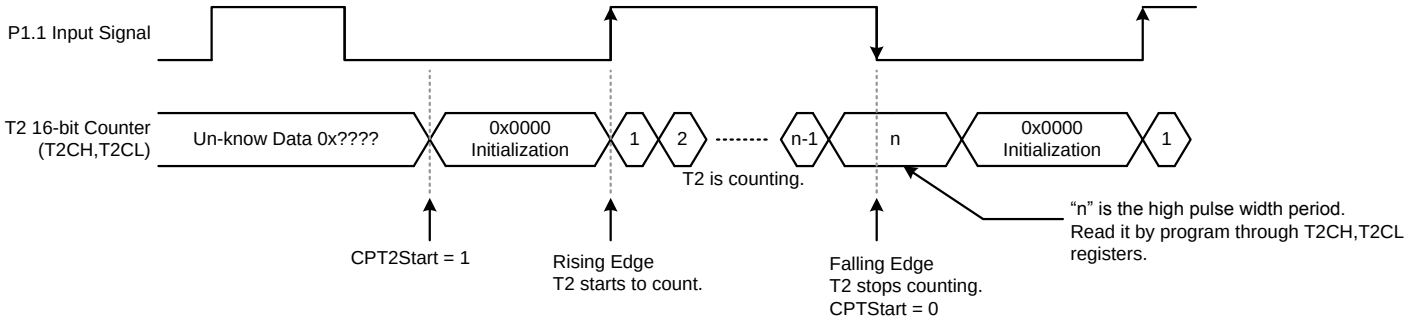




Input signal rate > T2 timer rate. Set a unique timer by T2 timer to measure input signal counts. Set T2 timer initial value (T2CH, T2CL = "m") and clear capture timer counter (CPT2CH, CPT2CL = 0x0000) by program. Set CPT2Satrt bit ("1") to start capture timer counting. Capture timer and T2 start counting at the first rising edge of input signal. When T2 timer overflow occurs (0xFFFF to 0x0000), capture timer stops counting, CPT2Start is cleared ("0") automatically, and the T2IRQ sets as "1". The capture timer 16-bit counter value (CPT2CH, CPT2CL = "n") is the continuous signal's counts.

8.4.5.2 High Pulse Width Measurement

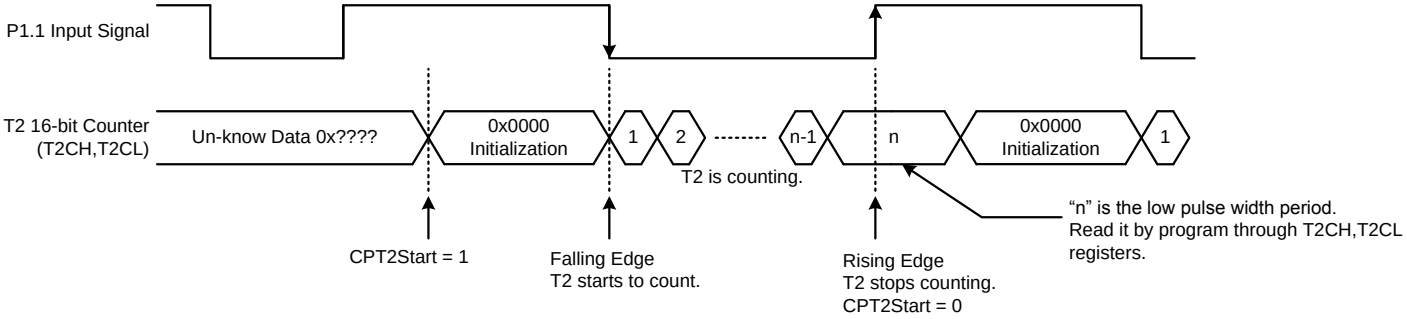
T2ENB = 1. CPT2EN = 1. CPT2G[1:0] = 01.



The high pulse width measurement is using rising edge to trigger T2 timer counting and falling edge to stop T2 timer. If set CPT2Start bit at high pulse duration, the capture timer will measure next high pulse until the rising edge occurrence. When the end of measuring high pulse width and T2 timer stops, the T2IRQ sets as "1", the T2 interrupt executes as T2IEN=1, and T2CH, T2CL 16-bit counter stores the period of high pulse width.

8.4.5.3 Low Pulse Width Measurement

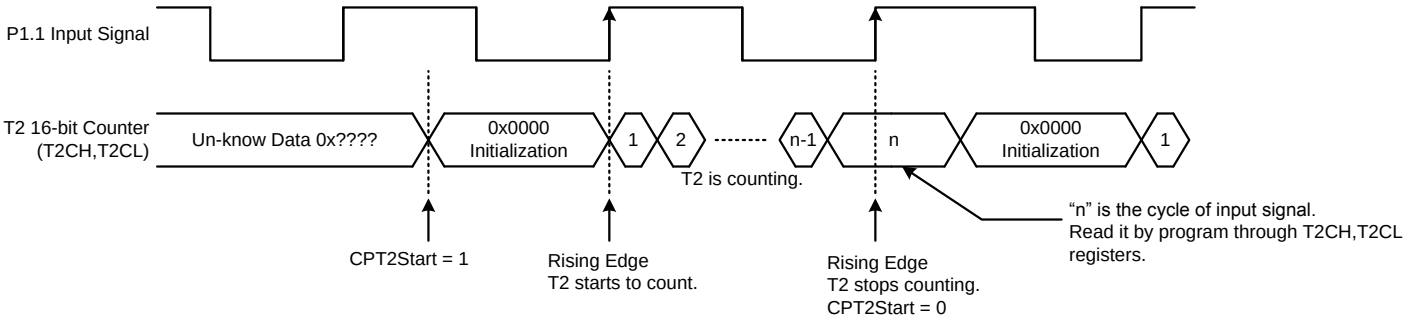
T2ENB = 1. CPT2EN = 1. CPT2G[1:0] = 10.



The low pulse width measurement is using falling edge to start T2 timer counting and rising edge to stop T2 timer. If set CPT2Start bit at low pulse duration, the capture timer will measure next low pulse until the falling edge occurrence. When the end of measuring low pulse width and T2 timer stops, the T2IRQ sets as "1", the T2 interrupt executes as T2IEN=1, and T2CH, T2CL 16-bit counter stores the period of low pulse width.

8.4.5.4 Input Cycle Measurement

T2ENB = 1. CPT2EN = 1. CPT2G[1:0] = 11.



The cycle measurement is using rising edge to start and stop T2 timer. If set CPT2Start bit at high or low pulse duration, the capture timer will measure next cycle until the rising edge occurrence. When the end of measuring cycle and T2 timer stops, the T2IRQ sets as "1", the T2 interrupt executes as T2IEN=1, and T2CH, T2CL 16-bit counter stores the period of input cycle.

8.4.6 CAPTURE TIMER CONTROL REGISTERS

0EDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPT2M	CPT2EN				CPT2MD	CPT2Start	CPT2G1	CPT2G0
Read/Write	R/W				R/W	R/W	R/W	R/W
After Reset	0				0	0	0	0

Bit 7 **CPT2EN:** Capture timer function control bit.
0 = Disable.
1 = Enable. **T2EN must be enabled.**

Bit 3 **CPT2MD:** Capture timer mode control bit.
0 = CPT2 overflow mode.
1 = T2 overflow mode.

Bit 2 **CPT2Start:** Capture timer counter control bit.
0 = Process end.
1 = Start to count and processing.

Bit [1:0] **CPT2G[1:0]:** Capture timer function control bit.
00 = Capture timer function.
01 = High pulse width measurement.
10 = Low pulse width measurement.
11 = Cycle measurement.

0EEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPT2CL	CPT2C7	CPT2C6	CPT2C5	CPT2C4	CPT2C3	CPT2C2	CPT2C1	CPT2C0
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

0EFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPT2CH	CPT2C15	CPT2C14	CPT2C13	CPT2C12	CPT2C11	CPT2C10	CPT2C9	CPT2C8
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 capture timer counter length is 16-bit and points to CPT2CH and CPT2CL registers. The timer counter is double buffer design. The core bus is 8-bit, so access 16-bit data needs a latch flag to avoid the transient status affect the 16-bit data mistake occurrence. Under write mode, the write CPT2CL is the latch control flag. Under read mode, the read CPT2CL is the latch control flag. So, write 16-bit counter is to write CPT2CH first, and then write CPT2CL. The 16-bit data is written to 16-bit counter buffer after executing writing CPT2CL. Read 16-bit counter is to read CPT2CL first, and then read CPT2CH. The 16-bit data is dumped to CPT2CH, CPT2CL after executing reading CPT2CL.

- **Read capture timer counter buffer sequence is to read CPT2CL first, and then read CPT2CH.**
- **Write capture timer counter buffer sequence is to write CPT2CH first, and then write CPT2CL.**

8.4.7 T2 TIMER OPERATION EXPLAME

- **T2 TIMER CONFIGURATION:**

```
; Reset T2 timer.
        MOV     A, #0x00        ; Clear T2M register.
        B0MOV   T2M, A

; Set T2 clock rate.
        MOV     A, #0nnn0000b   ; T2rate[2:0] bits.
        B0MOV   T2M, A

; Set T2CH, T2CL registers for T2 Interval time.
        MOV     A, #value1      ; Set high byte first.
        B0MOV   T2CH, A
        MOV     A, #value2      ; Set low byte.
        B0MOV   T2CL, A

; Clear T2IRQ
        B0BCLR  FT2IRQ

; Enable T2 timer and interrupt function.
        B0BSET  FT2IEN          ; Enable T2 interrupt function.
        B0BSET  FT2ENB          ; Enable T2 timer.
```

- **T2 CAPTURE TIMER FOR CONTINUOUS SIGNAL MEASUREMENT CONFIGURATION:**

```
; Reset T2 timer.
        CLR     T2M             ; Clear T2M register.

; Set T2 clock rate and select/enable T2 capture timer.
        MOV     A, #0nnnm000b   ; "nnn" is T2rate[2:0] for T2 clock rate selection.
        B0MOV   T2M, A          ; "m" is T2 clock source control bit.
        MOV     A, #000000mmb   ; "mm" is CPTG[1:0] for T2 capture timer function selection.
        B0MOV   CPT2M, A        ; CPT2G[1:0] = 00b, enable T2 capture timer.
                                ; CPT2G[1:0] = 01b/10b/11b, enable pulse width or cycle
                                  measurement.

; Select capture timer high-speed/low-speed mode.
        B0BCLR  FCPT2MD         ; CPT2 overflow mode.
; or
        B0BSET  FCPT2MD         ; T2 overflow mode.

; Clear T2CH, T2CL.
        CLR     T2CH            ; Clear high byte first.
        CLR     T2CL            ; Clear low byte.

; Set CPT2CH, CPT2CL 16-bit capture timer for continuous signal measurement.
        MOV     A, #value1      ; Set high nibble first.
        B0MOV   CPT2CH, A
        MOV     A, #value2      ; Set low byte.
        B0MOV   CPT2CL, A

; Clear T2IRQ
        B0BCLR  FT2IRQ

; Enable T2 timer, interrupt function and T2 capture timer function.
        B0BSET  FT2IEN          ; Enable T2 interrupt function.
        B0BSET  FT2ENB          ; Enable T2 timer.
        B0BSET  FCPT2EN         ; Enable T2 capture function.

; Set capture timer start bit.
        B0BSET  FCPT2Start
```

- **T2 CAPTURE TIMER FOR SINGLE CYCLE MEASUREMENT CONFIGURATION:**

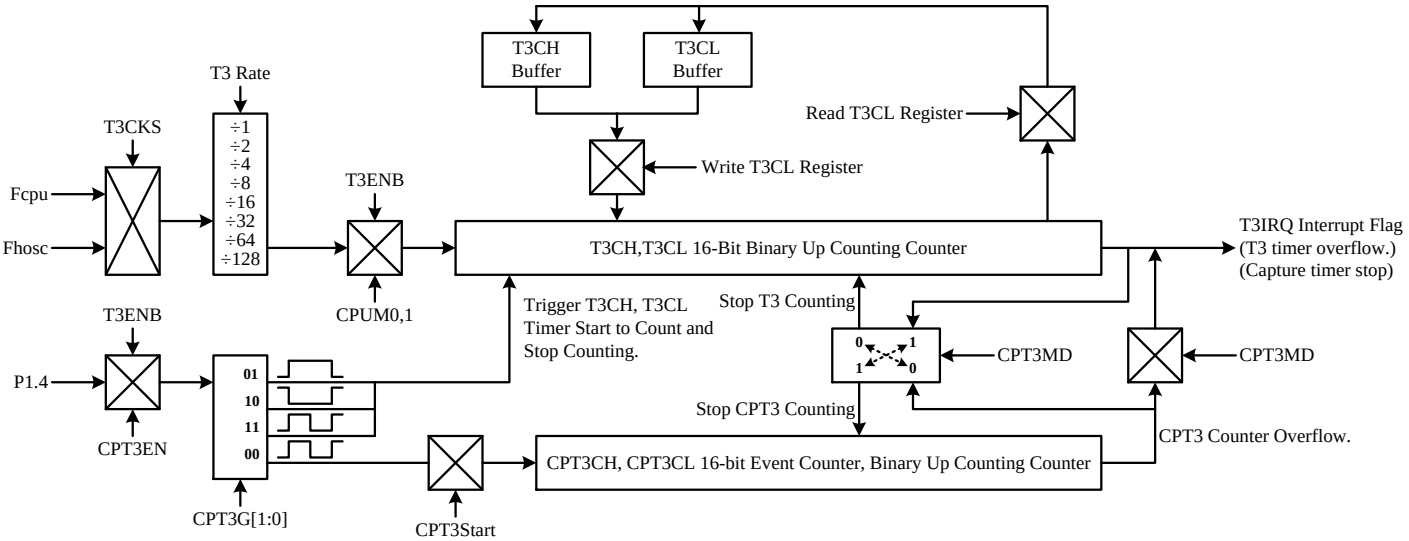
```
; Reset T2 timer.
        MOV         A, #0x00            ; Clear T2M register.
        B0MOV       T2M, A

; Set T2 clock rate, select input source, and select/enable T2 capture timer.
        MOV         A, #0nnnm000b       ; “nnn” is T2rate[2:0] for T2 clock rate selection.
        B0MOV       T2M, A              ; “m” is T2 clock source control bit.
        MOV         A, #000000mmb       ; “mm” is CPTG[1:0] for T2 capture timer function selection.
        B0MOV       CPT2M, A            ; CPT2G[1:0] = 00b, capture timer function.
                                        ; CPT2G[1:0] = 01b, high pulse width measurement.
                                        ; CPT2G[1:0] = 10b, low pulse width measurement.
                                        ; CPT2G[1:0] = 11b, cycle measurement.

; Clear T2CH, T2CL.
        CLR         T2CH                ; Clear high byte first.
        CLR         T2CL                ; Clear low byte.

; Clear T2IRQ
        B0BCLR      FT2IRQ

; Enable T2 timer, interrupt function and T2 capture timer function.
        B0BSET      FT2IEN              ; Enable T2 interrupt function.
        B0BSET      FT2ENB              ; Enable T2 timer.
        B0BSET      FCPTEN              ; Enable T2 capture function.
; Set capture timer start bit.
        B0BSET      FCPT2Start
```

8.5 T3 16-bit Timer with Capture Timer Function

8.5.1 OVERVIEW

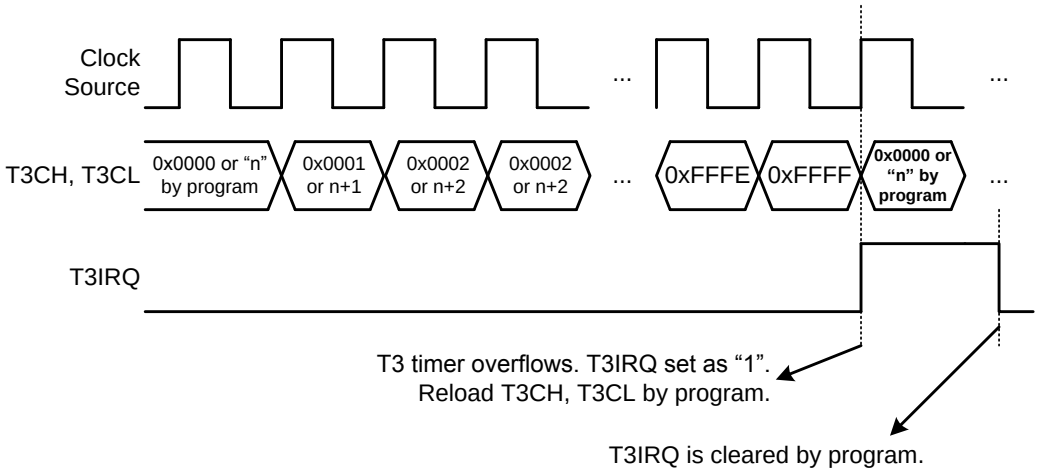
The T3 timer is a 16-bit binary up timer with basic timer and capture timer functions. The basic timer function supports flag indicator (T3IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T3M, T3CH/T3CL 16-bit counter registers. The capture timer supports high pulse width measurement, low pulse width measurement, cycle measurement and continuous duration from P1.4. T3 becomes a timer meter to count external signal time parameters to implement measure application. The main purposes of the T3 timer are as following.

- **16-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- **16-bit measurement:** Measure the input signal pulse width and cycle depend on the T3 clock time base to decide the capture timer's resolution. The capture timer builds in programmable trigger edge selection to decide the start-stop trigger event.
- **16-bit capture timer:** The 16-bit event counter to detect event source for accumulative capture timer function. The event counter is up counting design.
- **Interrupt function:** T3 timer function and capture timer function support interrupt function. When T3 timer occurs overflow or capture timer stops counting, the T3IRQ actives and the system points program counter to interrupt vector to do interrupt sequence.
- **Green mode function:** All T3 functions (timer, capture timer…) keeps running in green mode, but no wake-up function. Timer IRQ actives as any IRQ trigger occurrence, e.g. timer overflow…



8.5.2 T3 TIMER OPERATION

T3 timer is controlled by T3ENB bit. When T3ENB=0, T3 timer stops. When T3ENB=1, T3 timer starts to count. Before enabling T3 timer, setup T3 timer's configurations to select timer function modes, e.g. basic timer, interrupt function…T3 16-bit counter (T3CH, T3CL) increases "1" by timer clock source. When T3 overflow event occurs, T3IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is T3CH, T3CL count from full scale (0xFFFF) to zero scale (0x0000). T3 doesn't build in double buffer, so load T3CH, T3CL by program when T3 timer overflows to fix the correct interval time. If T3 timer interrupt function is enabled (T3IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 0008H) and executes interrupt service routine after T3 overflow occurrence. Clear T3IRQ by program is necessary in interrupt procedure. T3 timer can works in normal mode, slow mode and green mode.



T3 provides different clock sources to implement different applications and configurations. T3 clock source includes Fcpu (instruction cycle) and Fhosc (high speed oscillator) controlled by T3CKS bit. T3CKS bit selects the clock source is from Fcpu or Fhosc. If T3CKS=0, T3 clock source is Fcpu through T3rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. If T3CKS=1, T3 clock source is Fhosc through T3rate[2:0] pre-scalar to decide Fcpu/1~Fcpu/128. T3 length is 16-bit (65536 steps), and the one count period is each cycle of input clock.

T3CKS	T3rate[2:0]	T3 Clock	T3 Interval Time			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	2097.152	32	8388.608	128
0	001b	Fcpu/64	1048.576	16	4194.304	64
0	010b	Fcpu/32	524.288	8	2097.152	32
0	011b	Fcpu/16	262.144	4	1048.576	16
0	100b	Fcpu/8	131.072	2	524.288	8
0	101b	Fcpu/4	65.536	1	262.144	4
0	110b	Fcpu/2	32.768	0.5	131.072	2
0	111b	Fcpu/1	16.384	0.25	65.536	1
1	000b	Fhosc/128	524.288	8	2097.152	32
1	001b	Fhosc/64	262.144	4	1048.576	16
1	010b	Fhosc/32	131.072	2	524.288	8
1	011b	Fhosc/16	65.536	1	262.144	4
1	100b	Fhosc/8	32.768	0.5	131.072	2
1	101b	Fhosc/4	16.384	0.25	65.536	1
1	110b	Fhosc/2	8.192	0.125	32.768	0.5
1	111b	Fhosc/1	4.096	0.0625	16.384	0.25

8.5.3 T3M MODE REGISTER

T3M is T3 timer mode control register to configure T3 operating mode including T3 pre-scalar, clock source, capture parameters…These configurations must be setup completely before enabling T3 timer.

0FAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T3M	T3ENB	T3rate2	T3rate1	T3rate0	T3CKS			
Read/Write	R/W	R/W	R/W	R/W	R/W			
After reset	0	0	0	0	0			

Bit 7 **T3ENB:** T3 counter control bit.
0 = Disable T3 timer.
1 = Enable T3 timer.

Bit [6:4] **T3RATE[2:0]:** T3 timer clock source select bits.
T3CKS=0 -> 000 = Fcpu/128, 001 = Fcpu/64, 010 = Fcpu/32, 011 = Fcpu/16, 100 = Fcpu/8, 101 = Fcpu/4, 110 = Fcpu/2,111 = Fcpu/1.
T3CKS=1 -> 000 = Fhosc/128, 001 = Fhosc/64, 010 = Fhosc/32, 011 = Fhosc/16, 100 = Fhosc/8, 101 = Fhosc/4, 110 = Fhosc/2,111 = Fhosc/1.

Bit 3 **T3CKS:** T3 clock source control bit.
0 = Fcpu.
1 = Fhosc.

8.5.4 T3CH, T3CL 16-bit COUNTING REGISTERS

T3 counter is 16-bit counter combined with T3CH and T3CL registers. When T3 timer overflow occurs, the T3IRQ flag is set as "1" and cleared by program. The T3CH, T3CL decide T3 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T3CH and T3CL registers, and then enable T3 timer to make sure the fist cycle correct. After one T3 overflow occurs, the T3CH and T3CL registers are loaded correct values by program.

0FBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T3CL	T3CL7	T3CL6	T3CL5	T3CL4	T3CL3	T3CL2	T3CL1	T3CL0
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

0FCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T3CH	T3CH7	T3CH6	T3CH5	T3CH4	T3CH3	T3CH2	T3CH1	T3CH0
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 T3 timer counter length is 16-bit and points to T3CH and T3CL registers. The timer counter is double buffer design. The core bus is 8-bit, so access 16-bit data needs a latch flag to avoid the transient status affect the 16-bit data mistake occurrence. Under write mode, the write T3CH is the latch control flag. Under read mode, the read T3CL is the latch control flag. So, write T3 16-bit counter is to write T3CH first, and then write T3CL. The 16-bit data is written to 16-bit counter buffer after executing writing T3CL. Read T3 16-bit counter is to read T3CL first, and then read T3CH. The 16-bit data is dumped to T3CH, T3CL after executing reading T3CH.

- **Read T3 counter buffer sequence is to read T3CL first, and then read T3CH.**
- **Write T3 counter buffer sequence is to write T3CH first, and then write T3CL.**

The equation of T3 16-bit counter (T3CH, T3CL) initial value is as following.

T3CH, T3CL initial value = 65536 - (T3 interrupt interval time * T3 clock rate)

➢ **Example: To calculation T3CH and T3CL values to obtain 500ms T3 interval time. T3 clock source is Fcpu = 16MHz/16 = 1MHz. Select T3RATE=000 (Fcpu/128).**
T3 interval time = 500ms. T3 clock rate = 16MHz/16/128

*T3 16-bit counter initial value = 65536 - (T3 interval time * input clock)*
*= 65536 - (500ms * 16MHz / 16 / 128)*
*= 65536 - (500*10^{-3} * 16 * 10^{6} / 16 / 128)*
= F0BDH ***(T3CH = F0H, T3CL = BDH)***

8.5.5 T3 CPATURE TIMER

The 16-bit capture timer is controlled by CPT3EN bit, but the T3 must be enabled. Set T3ENB=1 and CPT3EN=1 to enable capture timer function. The capture timer is a pure counter and no clock source to decide interval time. Capture timer input source is P1.1 pin. CPT3G[1:0] bits select capture timer functions.

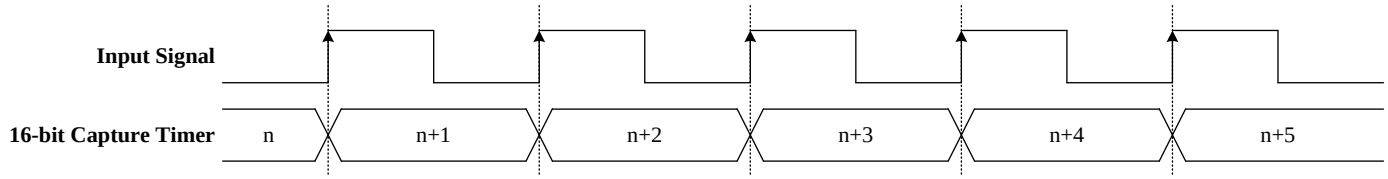
- **CPT3G[1:0] = 00: P1.1 Capture Timer Function.**
- **CPT3G[1:0] = 01: Measure P1.1 high pulse width.**
- **CPT3G[1:0] = 10: Measure P1.1 low pulse width.**
- **CPT3G[1:0] = 11: Measure P1.1 cycle.**

These functions must be combined T3 timer function to implement. The capture timer can measure high pulse width, low pulse width, cycle and capture duration of input signal (P1.1) controlled by CPT3G[1:0]. CPT3Start bit is to execute capture timer function. When CPT3Start is set as "1", the capture timer waits the right trigger edge to active 16-bit counter. The trigger edge finds, and the 16-bit counter starts to count which clock source is T3. When the second right edge finds, the 16-counter stops, CPT3Start is cleared and the T3IRQ actives.

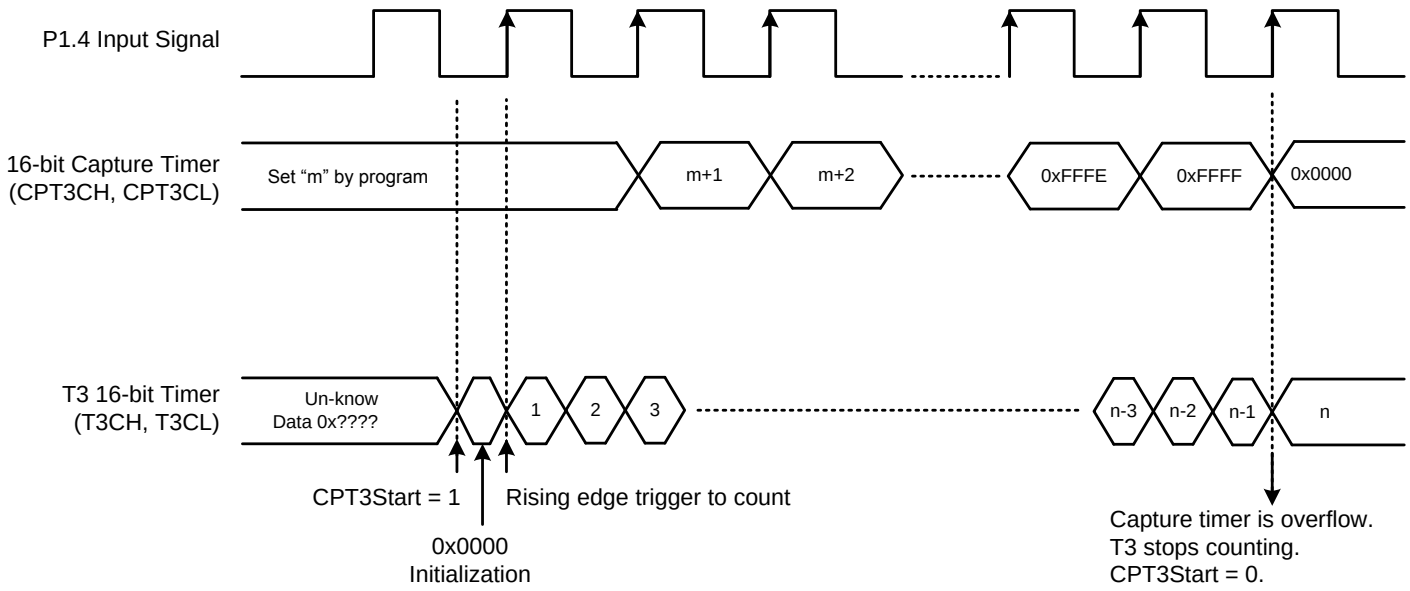
8.5.5.1 Capture Timer

The capture timer function controlled by CPT3G[1:0] bits. Set CPT3G[1:0] = 00 to enable capture timer function. The capture timer function's purpose is to measure the period of a continuous signal. The function includes two modes for difference speed signal controlled by CPT3MD bit. To start capture timer operation is set CPT3Start bit as "1", and the trigger source is the first rising edge of the P1.1 input signal. Before the first rising edge, the capture timer and T3 timer keeps ideal status and wait the riding edge event. When catch the first edge, the capture timer and T3 timer start to count. Each of overflow event occurs (controlled by CPT3MD bit), the capture timer and T3 timer stop counting, CPT3Start bit is cleared, and T3IRQ is set as "1". If T3IEN = 1, the system executes T3 interrupt function and service routine.

- **Capture timer counting trigger source is the rising edge of input signal.**

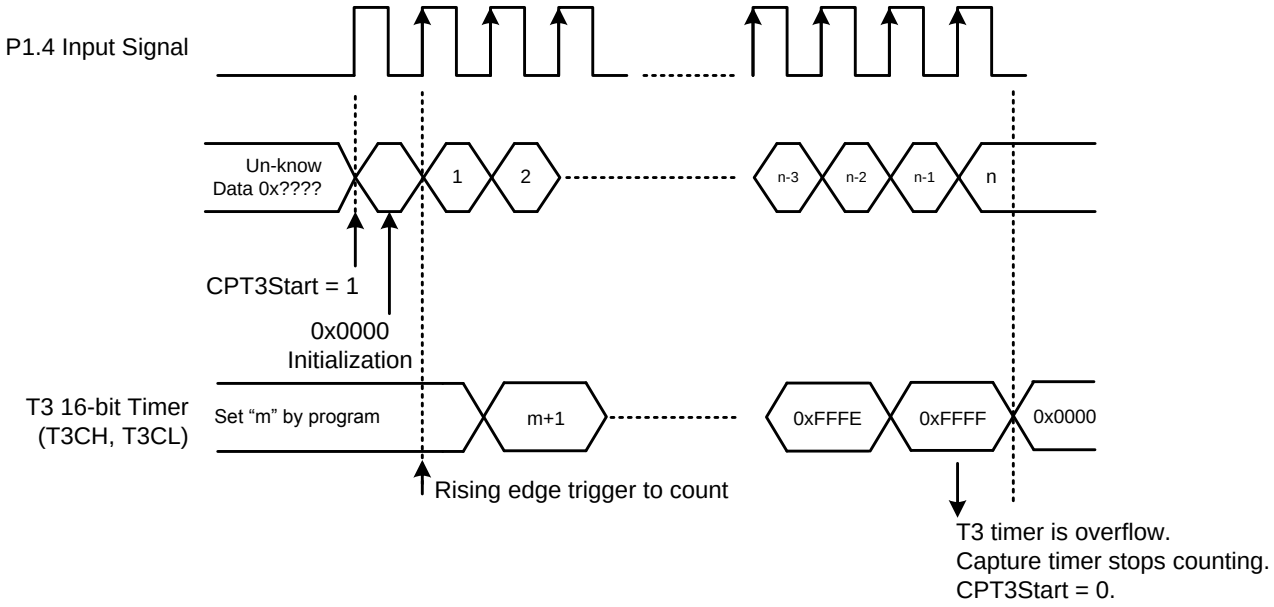


- **CPTMD = 0, Low-speed mode (T3ENB = 1. CPT3EN = 1. CPT3G[1:0] = 00.)**



Input signal rate < T3 timer rate. Use T3 timer to measure input signal continuous duration. Set capture timer initial value (CPT3CH, CPT3CL = “m”) and clear T3 counter (T3CH, T3CL = 0x0000) by program. Set CPT3Satrt bit (“1”) to start capture timer counting. Capture timer and T3 start counting at the first rising edge of input signal. When capture timer overflow occurs (0xFFFF to 0x0000), T3 stops counting, CPT3Start is cleared (“0”) automatically, and the T3IRQ sets as “1”. The T3 16-bit counter value (T3CH, T3CL = “n”) is the continuous signal’s duration.

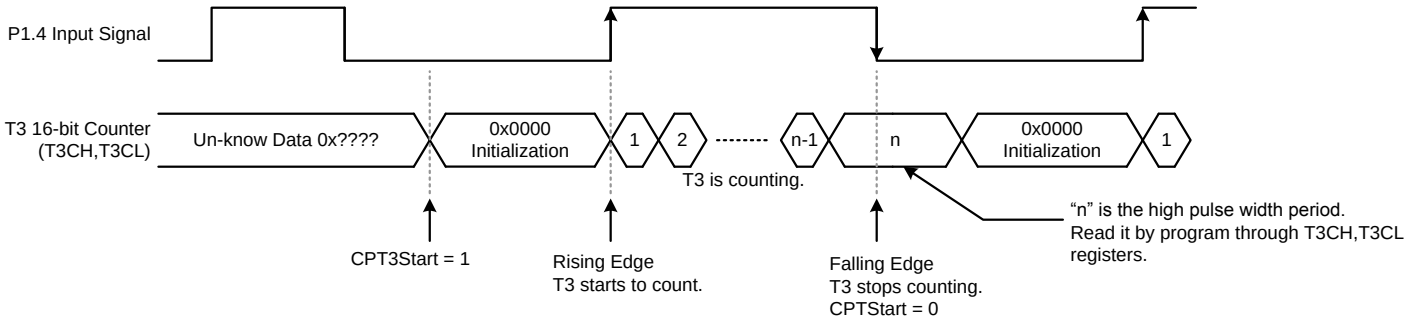
- **CPT3MD = 1, High-speed mode (T3ENB = 1. CPT3EN = 1. CPT3G[1:0] = 00.)**



Input signal rate > T3 timer rate. Set a unique timer by T3 timer to measure input signal counts. Set T3 timer initial value (T3CH, T3CL = “m”) and clear capture timer counter (CPT3CH, CPT3CL = 0x0000) by program. Set CPT3Satrt bit (“1”) to start capture timer counting. Capture timer and T3 start counting at the first rising edge of input signal. When T3 timer overflow occurs (0xFFFF to 0x0000), capture timer stops counting, CPT3Start is cleared (“0”) automatically, and the T3IRQ sets as “1”. The capture timer 16-bit counter value (CPT3CH, CPT3CL = “n”) is the continuous signal’s counts.

8.5.5.2 High Pulse Width Measurement

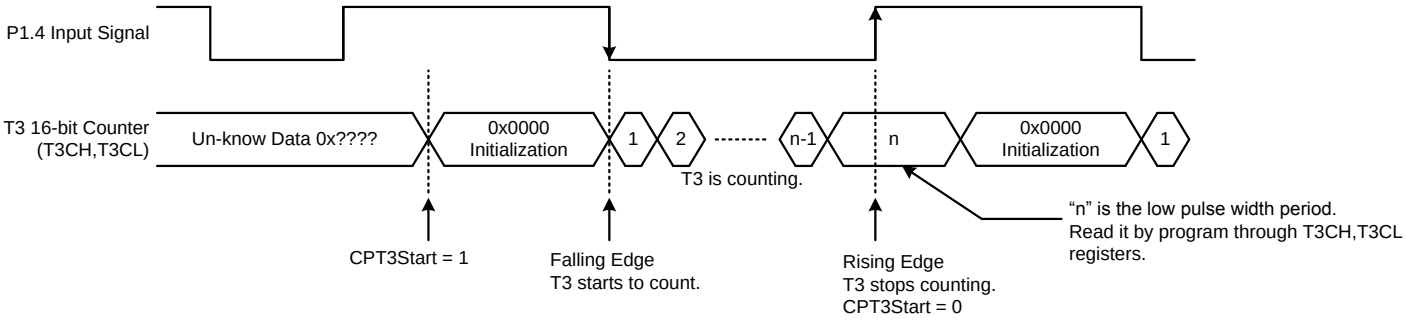
T3ENB = 1. CPT3EN = 1. CPT3G[1:0] = 01.



The high pulse width measurement is using rising edge to trigger T3 timer counting and falling edge to stop T3 timer. If set CPT3Start bit at high pulse duration, the capture timer will measure next high pulse until the rising edge occurrence. When the end of measuring high pulse width and T3 timer stops, the T3IRQ sets as “1”, the T3 interrupt executes as T3IEN=1, and T3CH, T3CL 16-bit counter stores the period of high pulse width.

8.5.5.3 Low Pulse Width Measurement

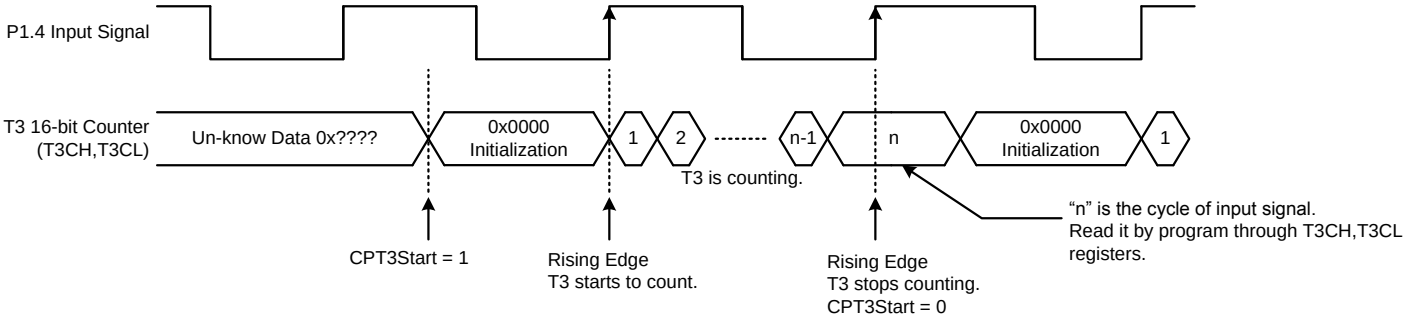
T3ENB = 1. CPT3EN = 1. CPT3G[1:0] = 10.



The low pulse width measurement is using falling edge to start T3 timer counting and rising edge to stop T3 timer. If set CPT3Start bit at low pulse duration, the capture timer will measure next low pulse until the falling edge occurrence. When the end of measuring low pulse width and T3 timer stops, the T3IRQ sets as "1", the T3 interrupt executes as T3IEN=1, and T3CH, T3CL 16-bit counter stores the period of low pulse width.

8.5.5.4 Input Cycle Measurement

T3ENB = 1. CPT3EN = 1. CPT3G[1:0] = 11.



The cycle measurement is using rising edge to start and stop T3 timer. If set CPT3Start bit at high or low pulse duration, the capture timer will measure next cycle until the rising edge occurrence. When the end of measuring cycle and T3 timer stops, the T3IRQ sets as "1", the T3 interrupt executes as T3IEN=1, and T3CH, T3CL 16-bit counter stores the period of input cycle.

8.5.6 CAPTURE TIMER CONTROL REGISTERS

0FDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPT3M	CPT3EN				CPT3MD	CPT3Start	CPT3G1	CPT3G0
Read/Write	R/W				R/W	R/W	R/W	R/W
After Reset	0				0	0	0	0

Bit 7 **CPT3EN:** Capture timer function control bit.
0 = Disable.
1 = Enable. **T3EN must be enabled.**

Bit 3 **CPT3MD:** Capture timer mode control bit.
0 = CPT3 overflow mode.
1 = T3 overflow mode.

Bit 2 **CPT3Start:** Capture timer counter control bit.
0 = Process end.
1 = Start to count and processing.

Bit [1:0] **CPT3G[1:0]:** Capture timer function control bit.
00 = Capture timer function.
01 = High pulse width measurement.
10 = Low pulse width measurement.
11 = Cycle measurement.

0FEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPT3CL	CPT3C7	CPT3C6	CPT3C5	CPT3C4	CPT3C3	CPT3C2	CPT3C1	CPT3C0
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

0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPT3CH	CPT3C15	CPT3C14	CPT3C13	CPT3C12	CPT3C11	CPT3C10	CPT3C9	CPT3C8
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 capture timer counter length is 16-bit and points to CPT3CH and CPT3CL registers. The timer counter is double buffer design. The core bus is 8-bit, so access 16-bit data needs a latch flag to avoid the transient status affect the 16-bit data mistake occurrence. Under write mode, the write CPT3CL is the latch control flag. Under read mode, the read CPT3CL is the latch control flag. So, write 16-bit counter is to write CPT3CH first, and then write CPT3CL. The 16-bit data is written to 16-bit counter buffer after executing writing CPT3CL. Read 16-bit counter is to read CPT3CL first, and then read CPT3CH. The 16-bit data is dumped to CPT3CH, CPT3CL after executing reading CPT3CL.

- **Read capture timer counter buffer sequence is to read CPT3CL first, and then read CPT3CH.**
- **Write capture timer counter buffer sequence is to write CPT3CH first, and then write CPT3CL.**

8.5.7 T3 TIMER OPERATION EXPLAME

- **T3 TIMER CONFIGURATION:**

```
; Reset T3 timer.
        MOV     A, #0x00        ; Clear T3M register.
        B0MOV   T3M, A

; Set T3 clock rate.
        MOV     A, #0nnn0000b   ; T3rate[2:0] bits.
        B0MOV   T3M, A

; Set T3CH, T3CL registers for T3 Interval time.
        MOV     A, #value1      ; Set high byte first.
        B0MOV   T3CH, A
        MOV     A, #value2      ; Set low byte.
        B0MOV   T3CL, A

; Clear T3IRQ
        B0BCLR  FT3IRQ

; Enable T3 timer and interrupt function.
        B0BSET  FT3IEN          ; Enable T3 interrupt function.
        B0BSET  FT3ENB          ; Enable T3 timer.
```

- **T3 CAPTURE TIMER FOR CONTINUOUS SIGNAL MEASUREMENT CONFIGURATION:**

```
; Reset T3 timer.
        CLR     T3M             ; Clear T3M register.

; Set T3 clock rate and select/enable T3 capture timer.
        MOV     A, #0nnnm000b   ; "nnn" is T3rate[2:0] for T3 clock rate selection.
        B0MOV   T3M, A          ; "m" is T3 clock source control bit.
        MOV     A, #000000mmb   ; "mm" is CPTG[1:0] for T3 capture timer function selection.
        B0MOV   CPT3M, A        ; CPT3G[1:0] = 00b, enable T3 capture timer.
                                ; CPT3G[1:0] = 01b/10b/11b, enable pulse width or cycle
                                  measurement.

; Select capture timer high-speed/low-speed mode.
        B0BCLR  FCPT3MD         ; CPT3 overflow mode.
; or
        B0BSET  FCPT3MD         ; T3 overflow mode.

; Clear T3CH, T3CL.
        CLR     T3CH            ; Clear high byte first.
        CLR     T3CL            ; Clear low byte.

; Set CPT3CH, CPT3CL 16-bit capture timer for continuous signal measurement.
        MOV     A, #value1      ; Set high nibble first.
        B0MOV   CPT3CH, A
        MOV     A, #value2      ; Set low byte.
        B0MOV   CPT3CL, A

; Clear T3IRQ
        B0BCLR  FT3IRQ

; Enable T3 timer, interrupt function and T3 capture timer function.
        B0BSET  FT3IEN          ; Enable T3 interrupt function.
        B0BSET  FT3ENB          ; Enable T3 timer.
        B0BSET  FCPT3EN         ; Enable T3 capture function.

; Set capture timer start bit.
        B0BSET  FCPT3Start
```

- **T3 CAPTURE TIMER FOR SINGLE CYCLE MEASUREMENT CONFIGURATION:**

```
; Reset T3 timer.
        MOV     A, #0x00        ; Clear T3M register.
        B0MOV   T3M, A

; Set T3 clock rate, select input source, and select/enable T3 capture timer.
        MOV     A, #0nnnm000b   ; "nnn" is T3rate[2:0] for T3 clock rate selection.
        B0MOV   T3M, A          ; "m" is T3 clock source control bit.
        MOV     A, #000000mmb   ; "mm" is CPTG[1:0] for T3 capture timer function selection.
        B0MOV   CPT3M, A        ; CPT3G[1:0] = 00b, capture timer function.
                                ; CPT3G[1:0] = 01b, high pulse width measurement.
                                ; CPT3G[1:0] = 10b, low pulse width measurement.
                                ; CPT3G[1:0] = 11b, cycle measurement.

; Clear T3CH, T3CL.
        CLR     T3CH            ; Clear high byte first.
        CLR     T3CL            ; Clear low byte.

; Clear T3IRQ
        B0BCLR  FT3IRQ

; Enable T3 timer, interrupt function and T3 capture timer function.
        B0BSET  FT3IEN          ; Enable T3 interrupt function.
        B0BSET  FT3ENB          ; Enable T3 timer.
        B0BSET  FCPTEN          ; Enable T3 capture function.
; Set capture timer start bit.
        B0BSET  FCPT3Start
```

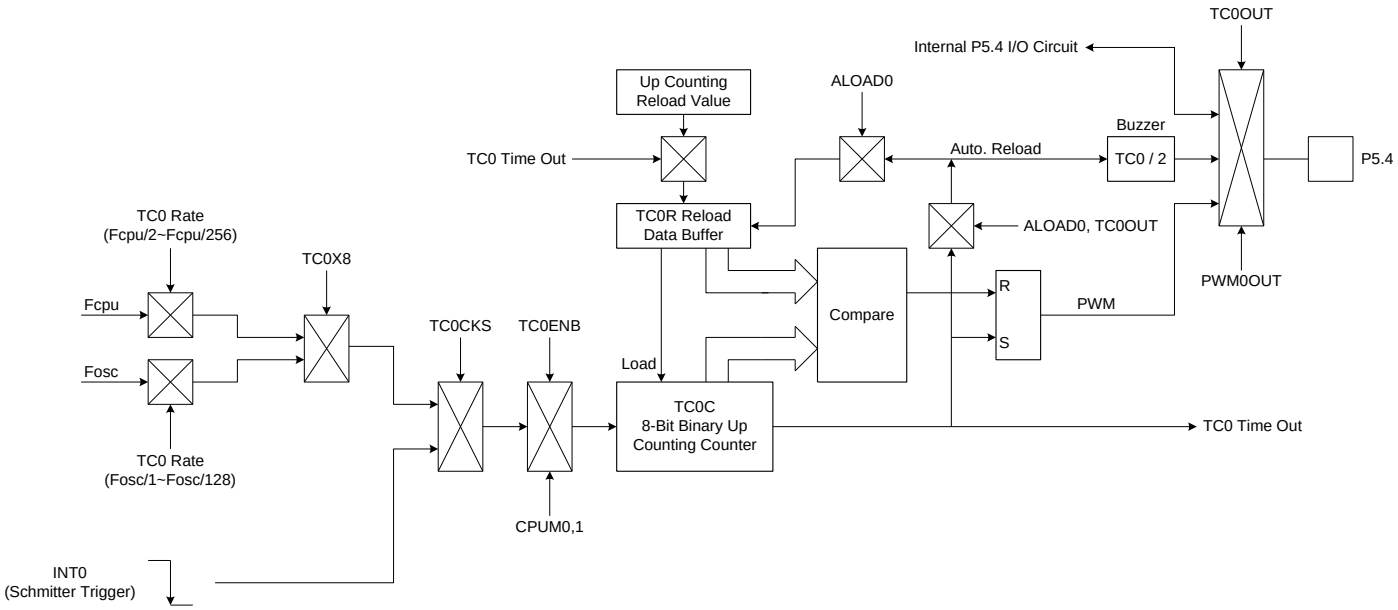
8.6 TIMER/COUNTER 0 (TC0)

8.6.1 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 P00 input pin.
- **Green mode wakeup function:** TC0 can be green mode wake-up time as TC0GN = 1. System will be wake-up by TC0 time out.
- **Buzzer output**
- **PWM output**



8.6.2 TC0M MODE REGISTER

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

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

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

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

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

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

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

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

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

8.6.3 TC0X8 FLAG

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

Bit 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.***

8.6.4 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

8.6.5 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.

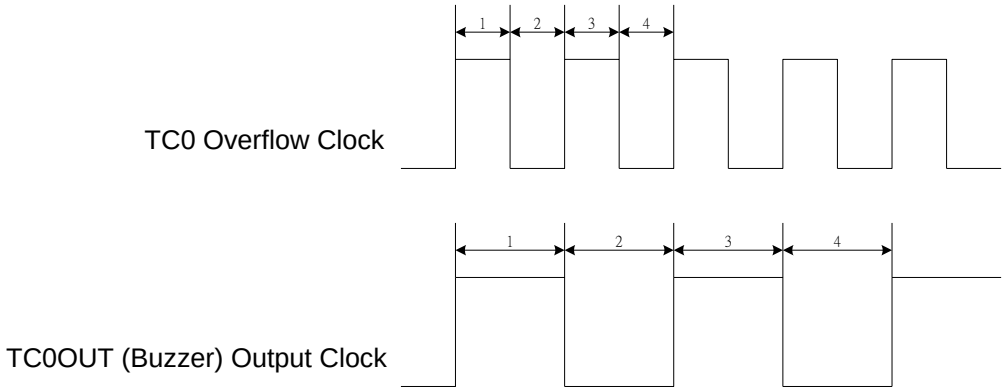
TC0CKS	TC0X8	PWM0	ALOAD0	TC0OUT	N	TC0R valid value	TC0R value binary type
0	0 (Fcpu/2~ Fcpu/256)	0	x	x	256	0x00~0xFF	00000000b~11111111b
		1	0	0	256	0x00~0xFF	00000000b~11111111b
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b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

8.6.6 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".***

8.6.7 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.
```

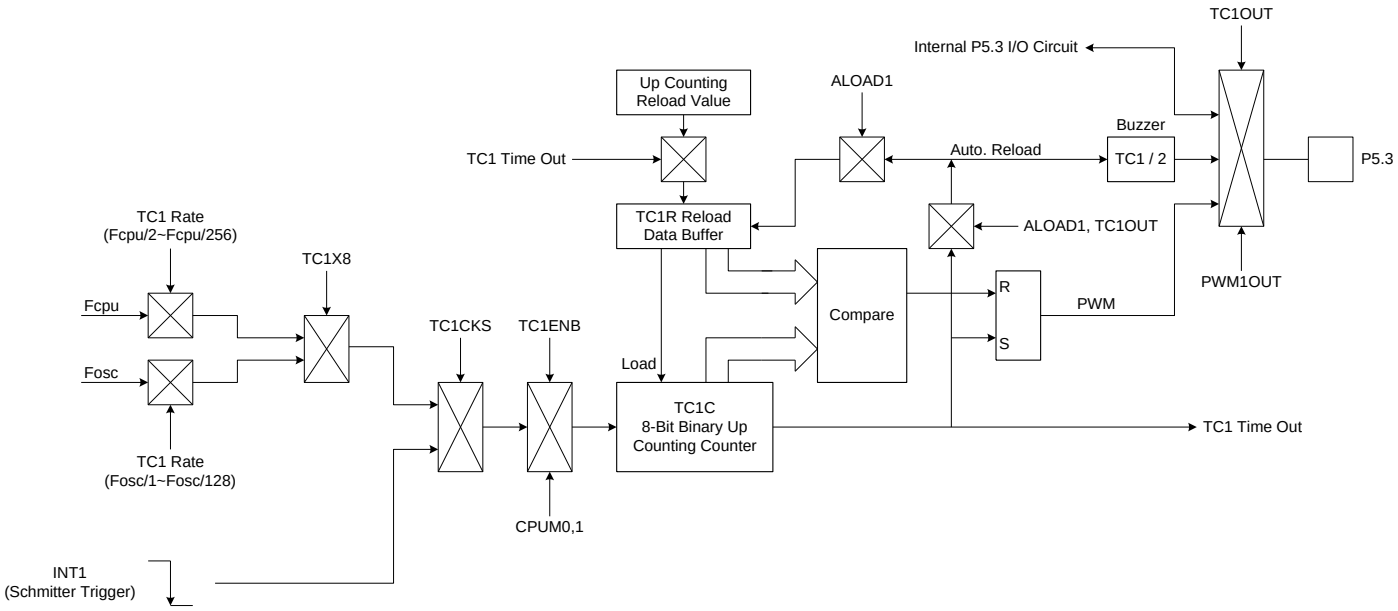
8.7 TIMER/COUNTER 1 (TC1)

8.7.1 OVERVIEW

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

The main purposes of the TC1 timer is as following.

- **8-bit programmable up counting timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **External event counter:** Counts system "events" based on falling edge detection of external clock signals at the P01 input pin.
- **Buzzer output**
- **PWM output**



8.7.2 TC1M MODE REGISTER

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

Bit 0 **PWM1OUT:** PWM output control bit.
0 = Disable PWM output.
1 = Enable PWM output. PWM duty controlled by TC1OUT, ALOAD1 bits.

Bit 1 **TC1OUT:** TC1 time out toggle signal output control bit. **Only valid when PWM1OUT = 0.**
0 = Disable, P5.3 is I/O function.
1 = Enable, P5.3 is output TC1OUT signal.

Bit 2 **ALOAD1:** Auto-reload control bit. **Only valid when PWM1OUT = 0.**
0 = Disable TC1 auto-reload function.
1 = Enable TC1 auto-reload function.

Bit 3 **TC1CKS:** TC1 clock source select bit.
0 = Internal clock (Fcpu or Fosc).
1 = External clock from P0.1/INT1 pin.

Bit [6:4] **TC1RATE[2:0]:** TC1 internal clock select bits.

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

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

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

8.7.3 TC1X8 FLAG

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

Bit 3 **TC1X8:** TC1 internal clock source control bit.
0 = TC1 internal clock source is Fcpu. TC1RATE is from Fcpu/2~Fcpu/256.
1 = TC1 internal clock source is Fosc. TC1RATE is from Fosc/1~Fosc/128.

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

8.7.4 TC1C COUNTING REGISTER

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

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

The equation of TC1C initial value is as following.

TC1C initial value = N - (TC1 interrupt interval time * input clock)

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

TC1CKS	TC1X8	PWM1	ALOAD1	TC1OUT	N	TC1C valid value	TC1C 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 TC1 interrupt. TC1 clock source is Fcpu (TC1KS=0, TC1X8=0) and no PWM output (PWM1=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC1RATE=010 (Fcpu/64).**

*TC1C initial value = N - (TC1 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 TC1, TC1X8 = 0.

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

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

TC1RATE	TC1CLOCK	High speed mode (Fcpu = 4MHz / 4)		Low speed mode (Fcpu = 32768Hz / 4)	
		Max overflow interval	One step = max/256	Max overflow interval	One step = max/256
000	Fosc/128	8.192 ms	32 us	1000 ms	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

8.7.5 TC1R AUTO-LOAD REGISTER

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

TC1 is double buffer design. If new TC1R value is set by program, the new value is stored in 1st buffer. Until TC1 overflow occurs, the new value moves to real TC1R buffer. This way can avoid TC1 interval time error and glitch in PWM and Buzzer output.

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

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

The equation of TC1R initial value is as following.

TC1R initial value = N - (TC1 interrupt interval time * input clock)

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

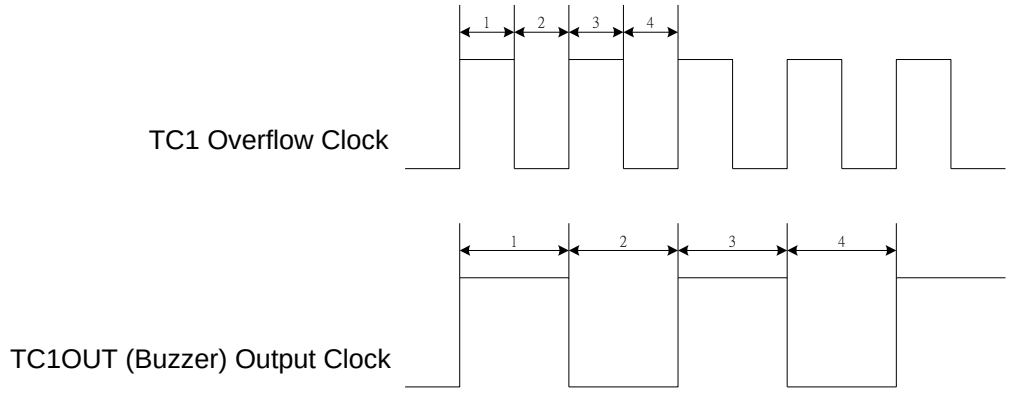
TC1CKS	TC1X8	PWM1	ALOAD1	TC1OUT	N	TC1R valid value	TC1R value binary type
0	0 (Fcpu/2~ Fcpu/256)	0	x	x	256	0x00~0xFF	00000000b~11111111b
		1	0	0	256	0x00~0xFF	00000000b~11111111b
		1	0	1	64	0x00~0x3F	xx000000b~xx111111b
		1	1	0	32	0x00~0x1F	xxx00000b~xxx11111b
		1	1	1	16	0x00~0x0F	xxxx0000b~xxxx1111b
	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 TC1 interrupt. TC1 clock source is Fcpu (TC1KS=0, TC1X8=0) and no PWM output (PWM1=0). High clock is external 4MHz. Fcpu=Fosc/4. Select TC1RATE=010 (Fcpu/64).**

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

8.7.6 TC1 CLOCK FREQUENCY OUTPUT (BUZZER)

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



➢ **Example: Setup TC1OUT output from TC1 to TC1OUT (P5.3). The external high-speed clock is 4MHz. The TC1OUT frequency is 0.5KHz. Because the TC1OUT signal is divided by 2, set the TC1 clock to 1KHz. The TC1 clock source is from external oscillator clock. TC1 rate is Fcpu/4. The TC1RATE2~TC1RATE1 = 110. TC1C = TC1R = 131.**

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

MOV     A,#131          ; Set the auto-reload reference value
B0MOV   TC1C,A
B0MOV   TC1R,A

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

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

8.7.7 TC1 TIMER OPERATION SEQUENCE

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

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

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

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

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

☞ **Set TC1 timer clock source.**

```
; Select TC1 internal / external clock source.
B0BCLR    FTC1CKS     ; Select TC1 internal clock source.
or
B0BSET    FTC1CKS     ; Select TC1 external clock source.

; Select TC1 Fcpu / Fosc internal clock source .
B0BCLR    FTC1X8      ; Select TC1 Fcpu internal clock source.
or
B0BSET    FTC1X8      ; Select TC1 Fosc internal clock source.
```

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

☞ **Set TC1 timer auto-load mode.**

```
B0BCLR    FALOAD1     ; Enable TC1 auto reload function.
or
B0BSET    FALOAD1     ; Disable TC1 auto reload function.
```

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

```
; Set TC1 interrupt interval time, TC1OUT (Buzzer) frequency or PWM duty.
MOV       A,#7FH      ; TC1C and TC1R value is decided by TC1 mode.
B0MOV     TC1C,A      ; Set TC1C value.
B0MOV     TC1R,A      ; Set TC1R value under auto reload mode or PWM mode.

; In PWM mode, set PWM cycle.
B0BCLR    FALOAD1     ; ALOAD1, TC1OUT = 00, PWM cycle boundary is 0~255.
B0BCLR    FTC1OUT
or
B0BCLR    FALOAD1     ; ALOAD1, TC1OUT = 01, PWM cycle boundary is 0~63.
B0BSET    FTC1OUT
or
B0BSET    FALOAD1     ; ALOAD1, TC1OUT = 10, PWM cycle boundary is 0~31.
B0BCLR    FTC1OUT
or
B0BSET    FALOAD1     ; ALOAD1, TC1OUT = 11, PWM cycle boundary is 0~15.
B0BSET    FTC1OUT
```

☞ **Set TC1 timer function mode.**

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

☞ **Enable TC1 timer.**

```
        B0BSET      FTC1ENB         ; Enable TC1 timer.
```

8.8 PWM0 MODE

8.8.1 OVERVIEW

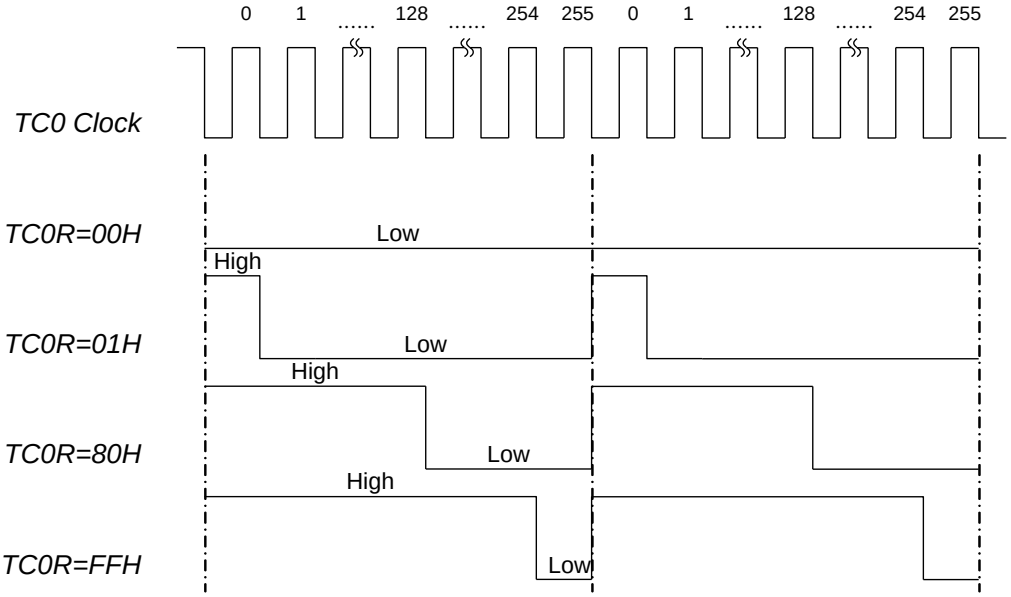
PWM function is generated by TC0 timer counter and output the PWM signal to PWM0OUT pin (P5.4). The value of the 8-bit counter (TC0C) is compared to the contents of the reference register (TC0R). When the reference register value (TC0R) is equal to the counter value (TC0C), the PWM output goes low. When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

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.***

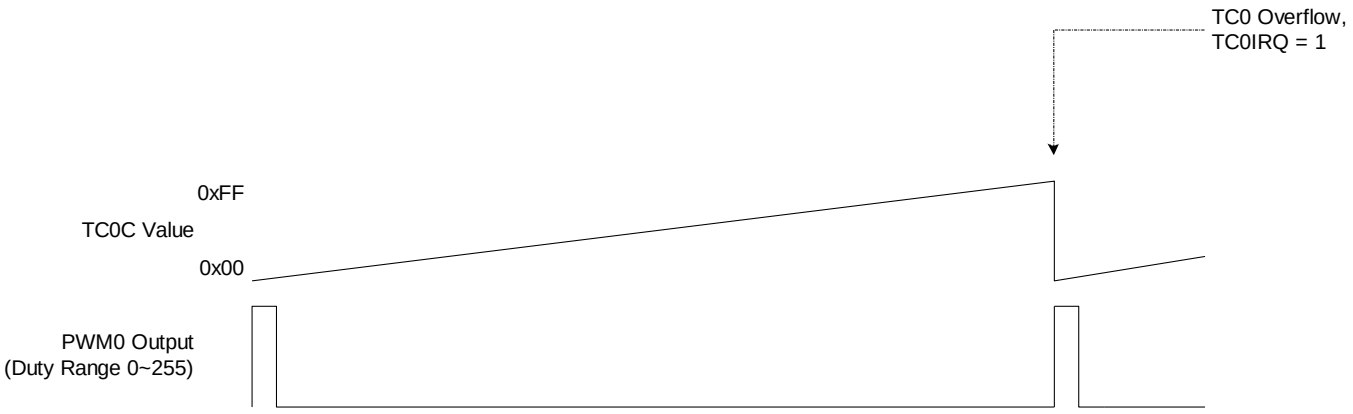
PWM duty range	TC0C valid value	TC0R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count

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



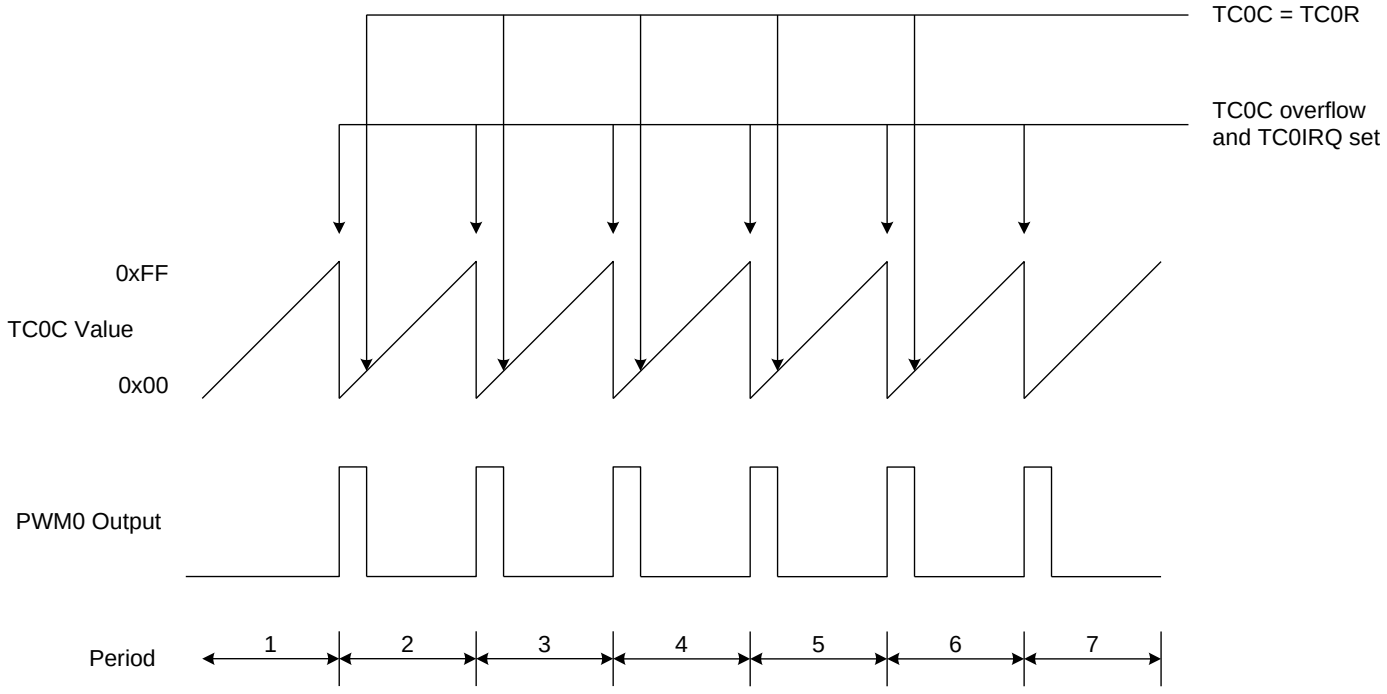
8.8.2 TCxIRQ and PWM Duty

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

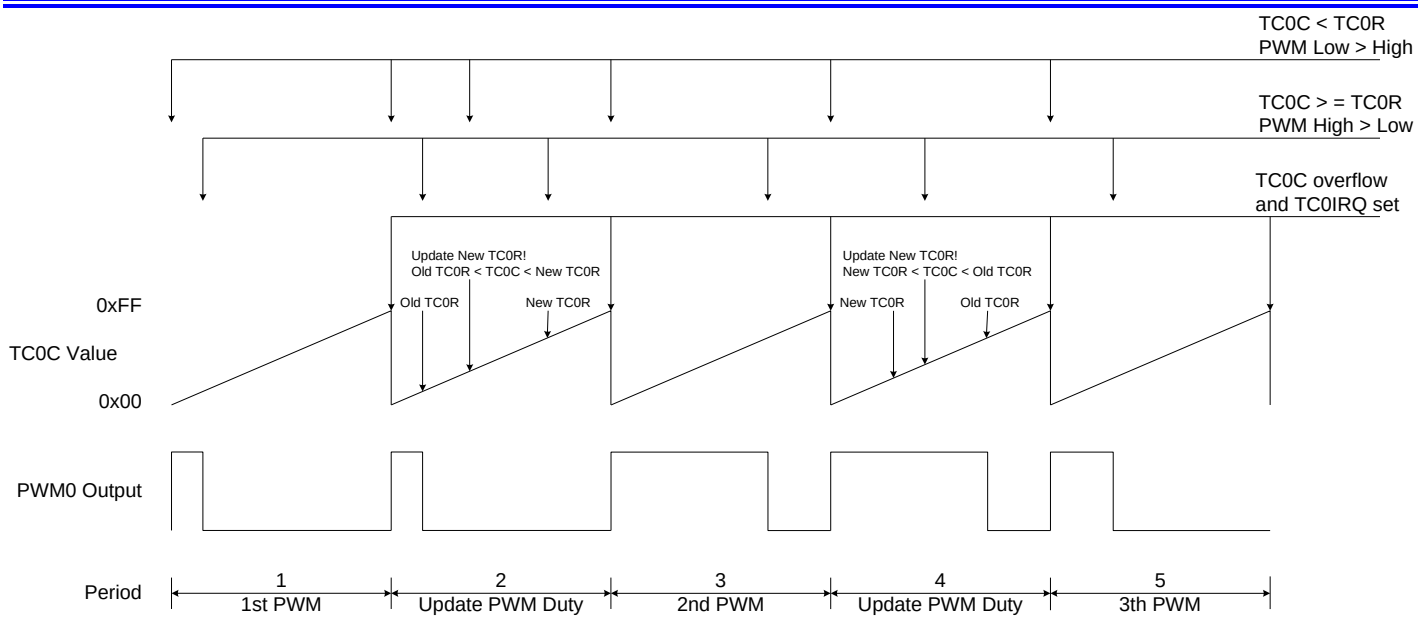


8.8.3 PWM Duty with TCxR Changing

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.

8.8.4 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.***

8.9 PWM1 MODE

8.9.1 OVERVIEW

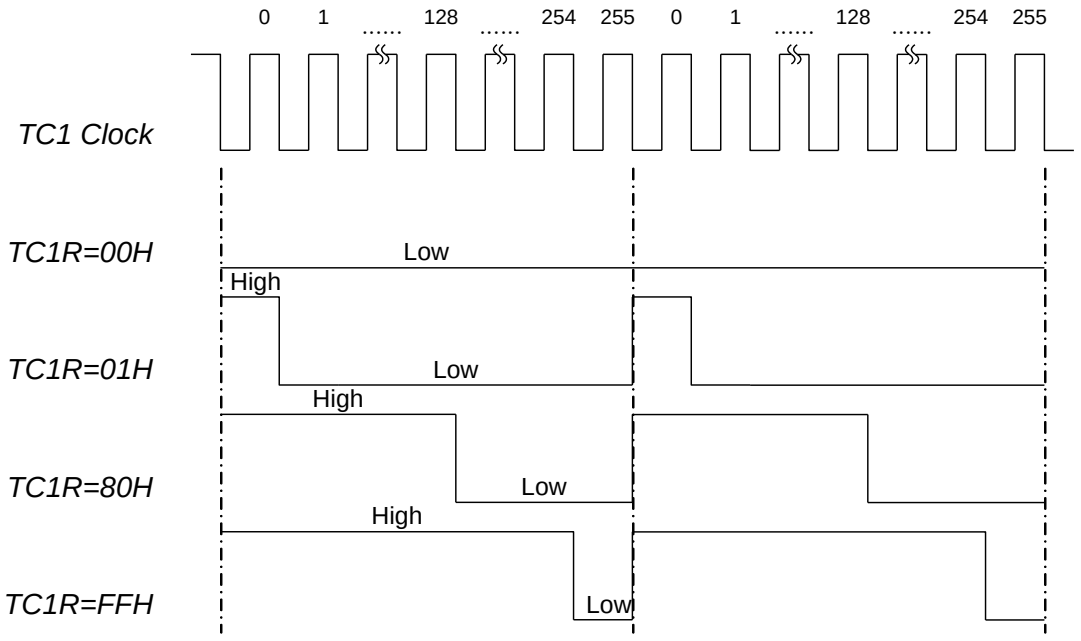
PWM function is generated by TC1 timer counter and output the PWM signal to PWM1OUT pin (P5.3). The 8-bit counter counts modulus 256. The value of the 8-bit counter (TC1C) is compared to the contents of the reference register (TC1R). When the reference register value (TC1R) is equal to the counter value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

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 TC1R.

✸ ***Note: TC1 is double buffer design. Modifying TC1R to change PWM duty by program, there is no glitch and error duty signal in PWM output waveform. Users can change TC1R any time, and the new reload value is loaded to TC1R buffer at TC1 overflow.***

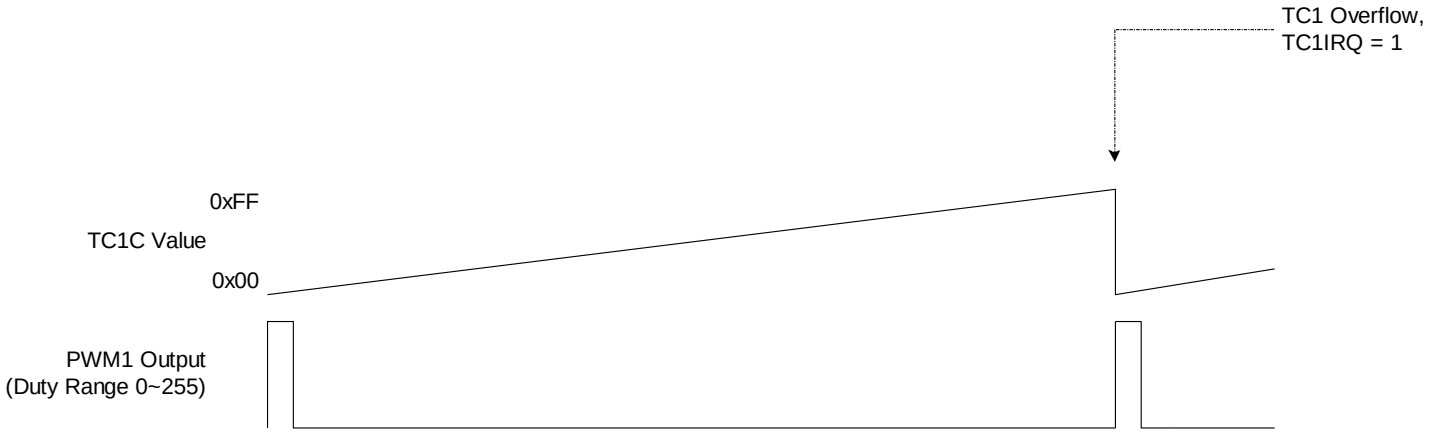
PWM duty range	TC1C valid value	TC1R valid bits value	MAX. PWM Frequency (Fcpu = 4MHz)	Remark
0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	Overflow per 256 count

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



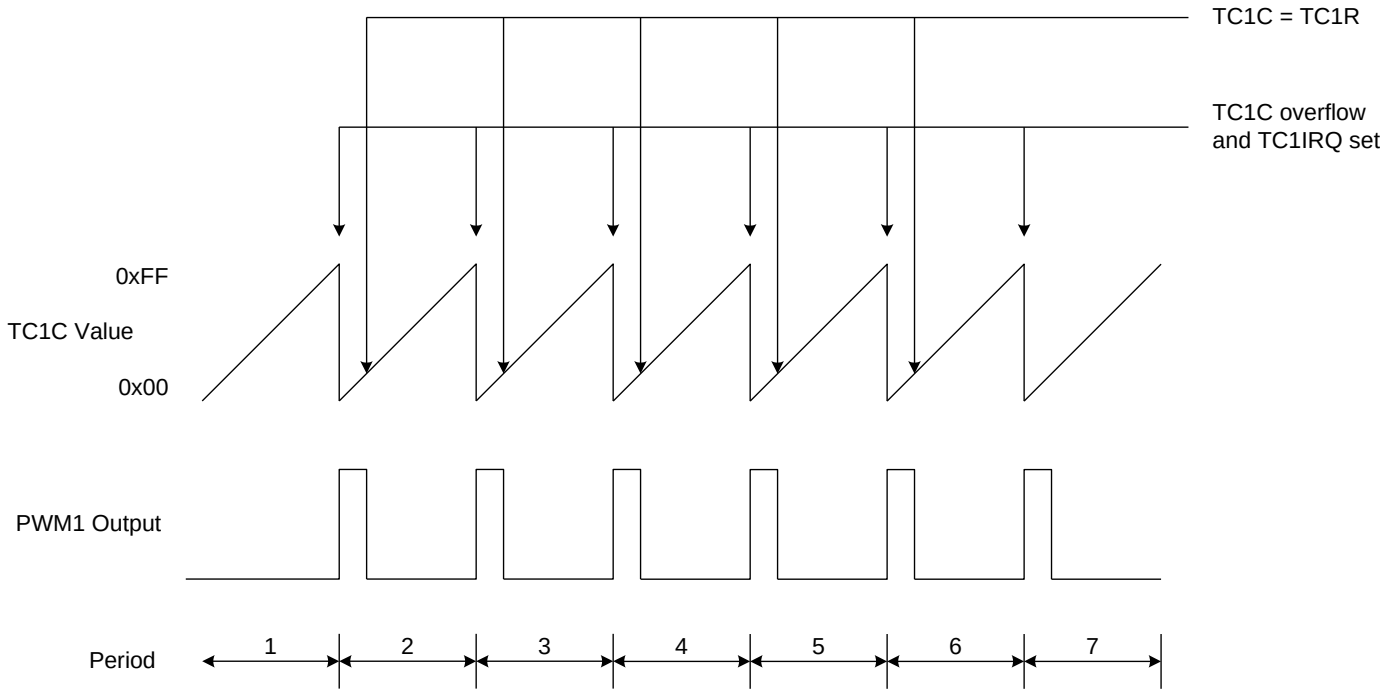
8.9.2 TCxIRQ and PWM Duty

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

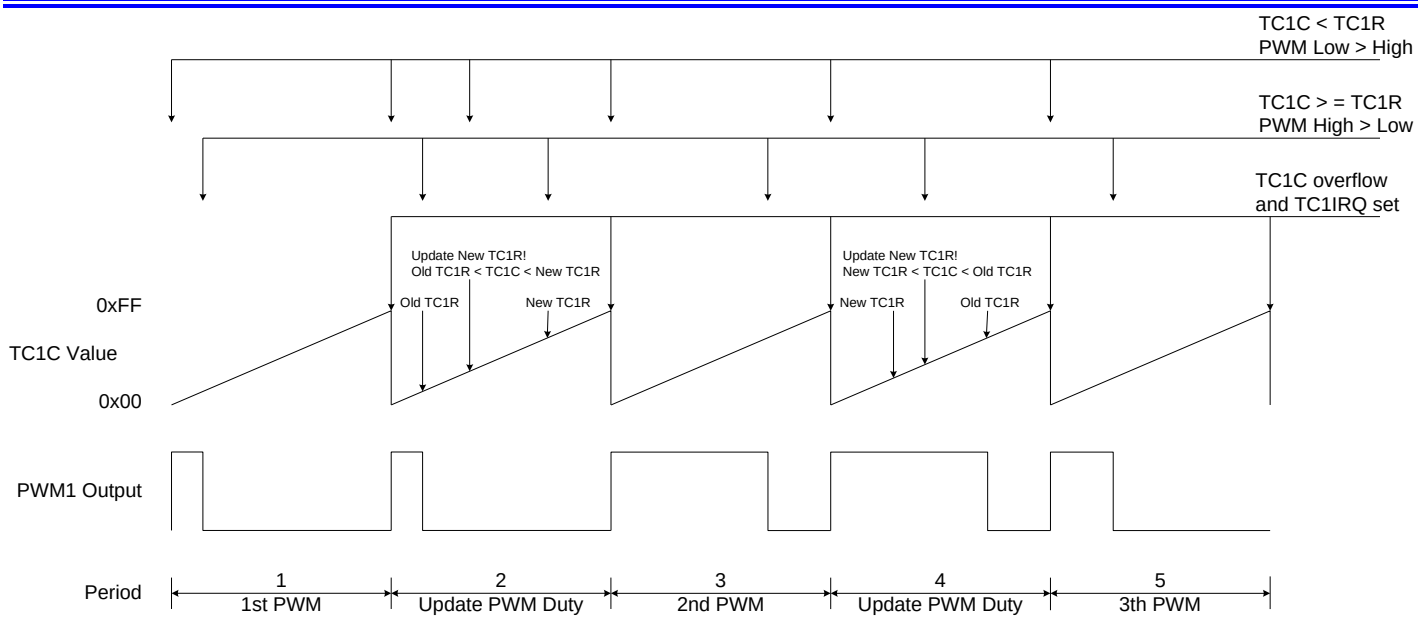


8.9.3 PWM Duty with TCxR Changing

In PWM mode, the system will compare TC1C and TC1R all the time. When TC1C<TC1R, the PWM will output logic “High”, when TC1C≧ TC1R, the PWM will output logic “Low”. If TC1C is changed in certain period, the PWM duty will change in next PWM period. If TC1R is fixed all the time, the PWM waveform is also the same.



Above diagram is shown the waveform with fixed TC1R. In every TC1C overflow PWM output “High, when TC1C≧ TC1R PWM output ”Low”. If TC1R is changing in the program processing, the PWM waveform will became as following diagram.



In period 2 and period 4, new Duty (TC1R) is set. TC1 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.

8.9.4 PWM PROGRAM EXAMPLE

➢ **Example: Setup PWM1 output from TC1 to PWM1OUT (P5.3). 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. TC1 rate is Fcpu/4. The TC1RATE2~TC1RATE1 = 110. TC1C = TC1R = 30.**

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

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

B0BCLR  FTC1OUT         ; Set duty range as 0/256~255/256.
B0BCLR  FALOAD1
B0BSET  FPWM1OUT        ; Enable PWM1 output to P5.3 and disable P5.3 I/O function
B0BSET  FTC1ENB         ; Enable TC1 timer
```

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

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

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

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

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

9 4x36 LCD DRIVER

9.1 OVERVIEW

The MCU builds in 4x36 (4 commons and 36 segments, 144 dots) LCD driver. The LCD supports 1/4 duty and 1/3 bias LCD panel. The LCD frame rate is 64Hz and clock source is external 32768Hz oscillator crystal or RC type

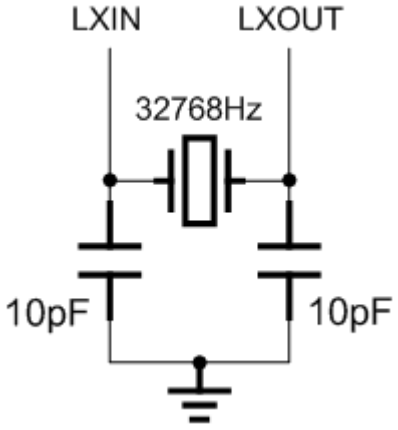
9.2 LCD REGISTERS

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM	RES1	RES0	-	-	RCLK	P2SEG		LCDENB
Read/Write	R/W	R/W	-	-	R/W	R/W		R/W
After reset	0	0	-	-	0	0		0

Bit [7:6] **RES[1:0]:** LCD internal resistor control bit.
11 = 10k ohm.
10 = 50k ohm.
01 = 100k ohm.
00 = 200k ohm.

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

➢ ***Note1: Circuit diagram when RCLK=0 –External Low Clock sets as Crystal mode.***



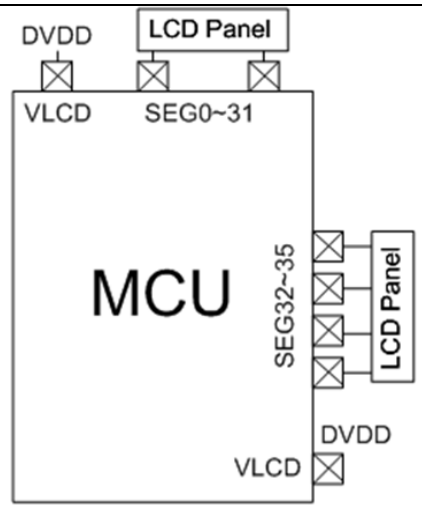
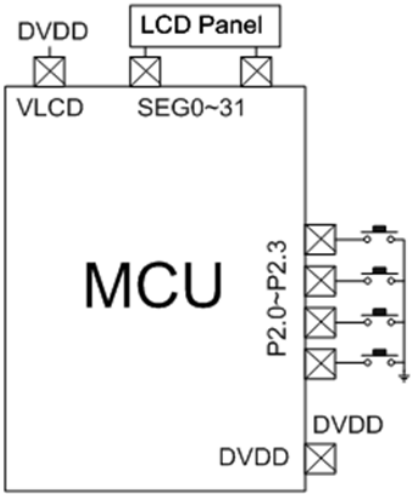
➢ ***Note2: Circuit diagram when "RCLK=1" will enable external Low Clock sets as RC mode.***

✸ ***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.***

Bit 2 **P2SEG:** Seg32~Seg35 shared with P20~P23 control bit..
0 = Disable.
1 = Enable.

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

✷ ***Note: Seg32~Seg35 pins are shared with P20~P23. When these pins are used as Port2 general purpose I/O mode, the P2SEG bit of LCDM register must be set as "1".***

P3SEG=0 SEG32~35 as LCD SEG	P3SEG=1 SEG32~35 as P2 IO Port
	

➢ **Example : Set LCD mode.**

```
; Set LCD 32768Hz clock source type.
                B0BCLR      FRCLK           ; Crystal/ceramic type.
or
                B0BSET      FRCLK           ; RC type.
; Set LCD Bias.

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

9.3 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~COM4 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	-	-	-	-
20h	SEG32	20h.0	20h.1	20h.2	20h.3				
21h	SEG33	21h.0	21h.1	21h.2	21h.3				
22h	SEG34	22h.0	22h.1	22h.2	22h.3				
23h	SEG35	23h.0	23h.1	23h.2	23h.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.
…
```

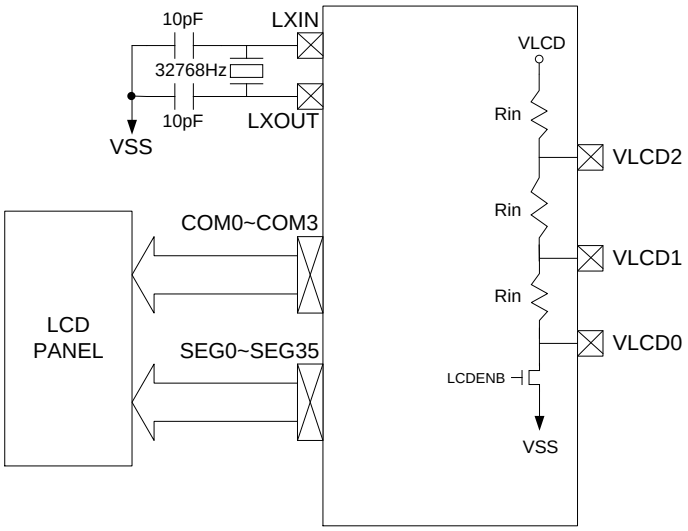
➢ **Example : Set LCD RAM data by directly addressing in bank 15.**

```
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
```

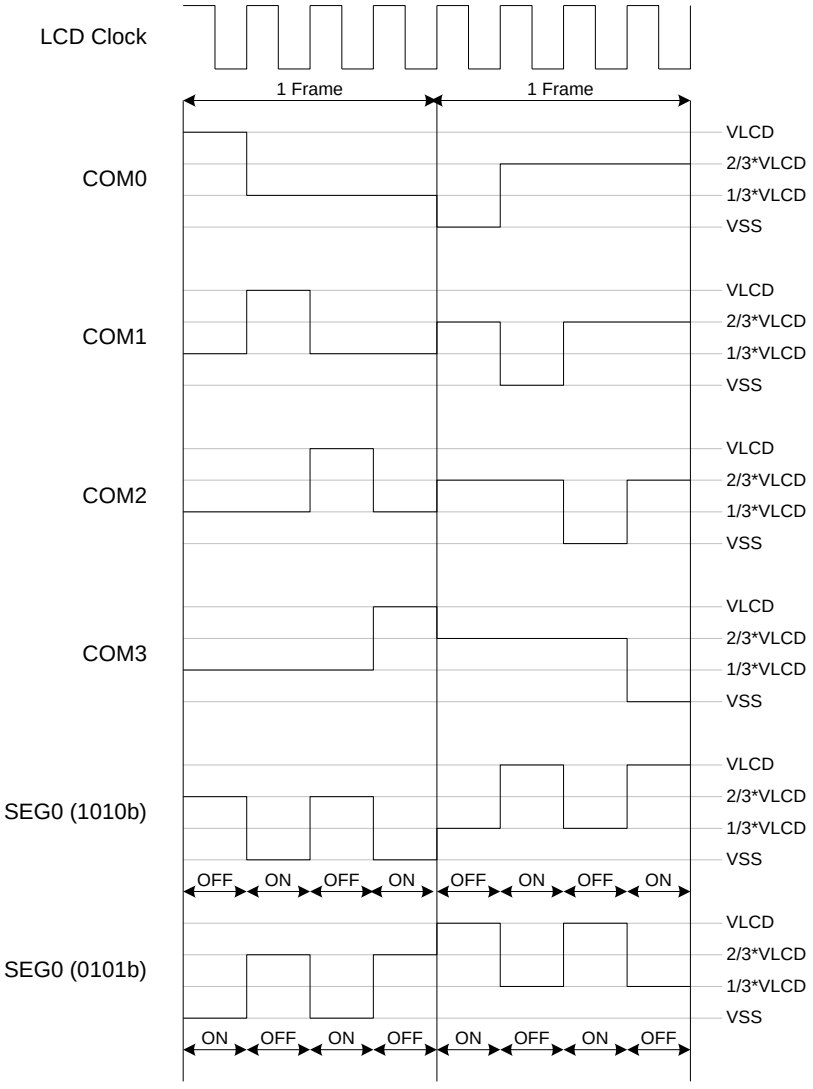
✸ ***Note: Access RAM data of bank 0 (system registers and user define RAM 0x0000~0x007F) is using "B0xxx" instructions.***

9.4 LCD WAVEFORM

1/4 duty , 1/3 bias.



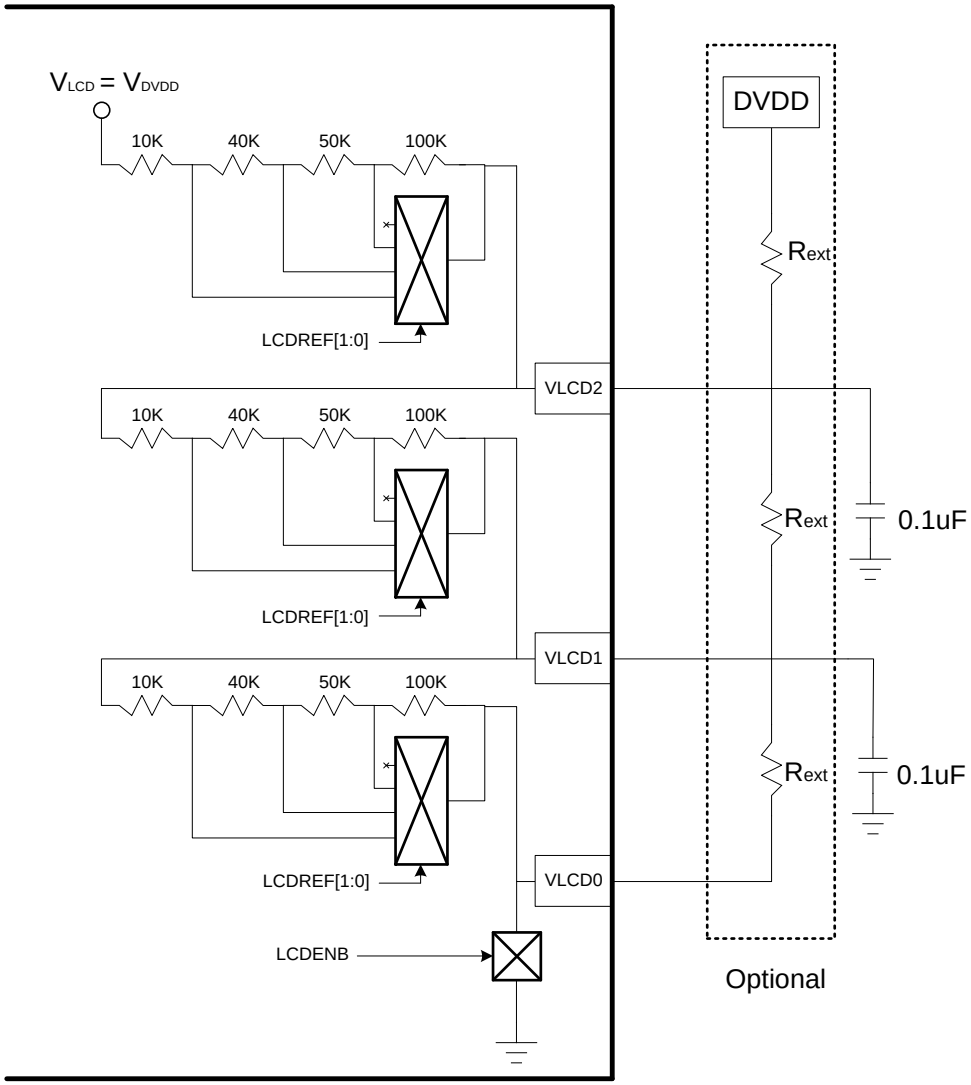
1/4 duty 1/3 bias, LCD Circuit.



9.5 LCD Block

In following diagram, in order to get suitable contrast level of LCD panel, users can add external resistor to bias pin (VLCD2, VLCD1, VLCD0) to adjust bias voltage and LCD drive current. Too much or less current makes the LCD to bring remnant images. In normal condition, the external bias resistor value is 200K, 100K, 50K, and 10K controlled by RES1 and RES0 of the LCDM register. Users can connect a resistor between VLCD0~VLCD2 pin and DVDD to adjust the LCD driving current.

$$\text{LCD Current consumption} = \frac{\text{DVDD}}{\left(\frac{\text{Rin} \times \text{Rext}}{\text{Rin} + \text{Rext}}\right) \times 3}$$



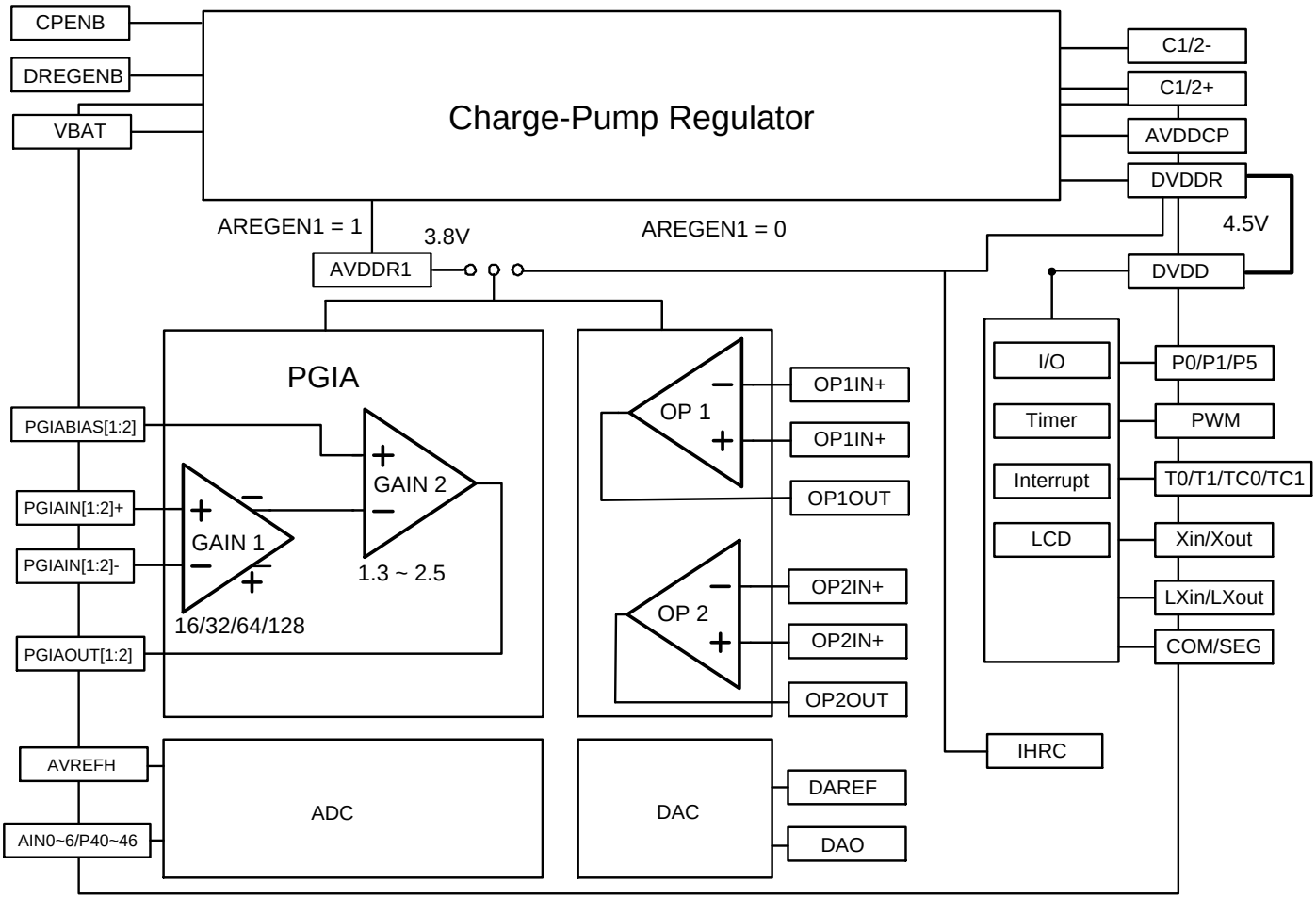
✸ ***Note: VLCD2 = 2/3*VLCD, VLCD1 = 1/3*VLCD, VLCD0 = GND connected by internal circuit***

10 Charge-Pump, PGIA and OPA

10.1 OVERVIEW

The SN8P2839 has built-in triple Voltage Charge-Pump/Regulator (CPR) to support a stable voltage to digital regulator 4.5V and analog regulator 3.8V/2.5V with total maximum 10mA current driving capacity. This CPR provides stable voltage for internal circuits (PGIA, ADC and OPA) and external sensor (e.g. blood pressure sensor, thermistor, etc). The SN8P1819A also integrated 12-bit Analog-to-Digital Converters (ADC) up to 2048-step resolution. A very low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selectable gains stage one of16x, 32x, 64x, 128x and gains stage two of 1.3x ~ 2.5x.

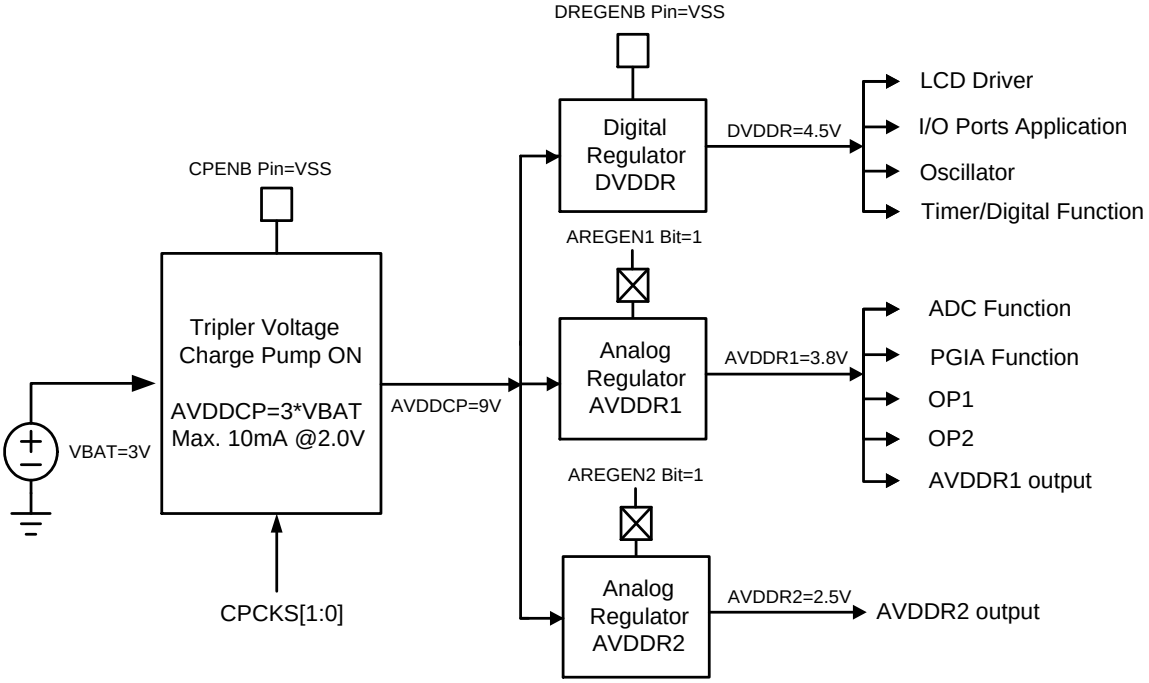
10.2 BLOCK DIAGRAM



Following figure illustrates a block diagram of the Charge-Pump Regulator, PGIA and OPA module. Charge pump function is controlled by CPENB pin and Digital Regulator is controlled by EREGENB pin. The Analog Regulator is controlled by register AREGENB.

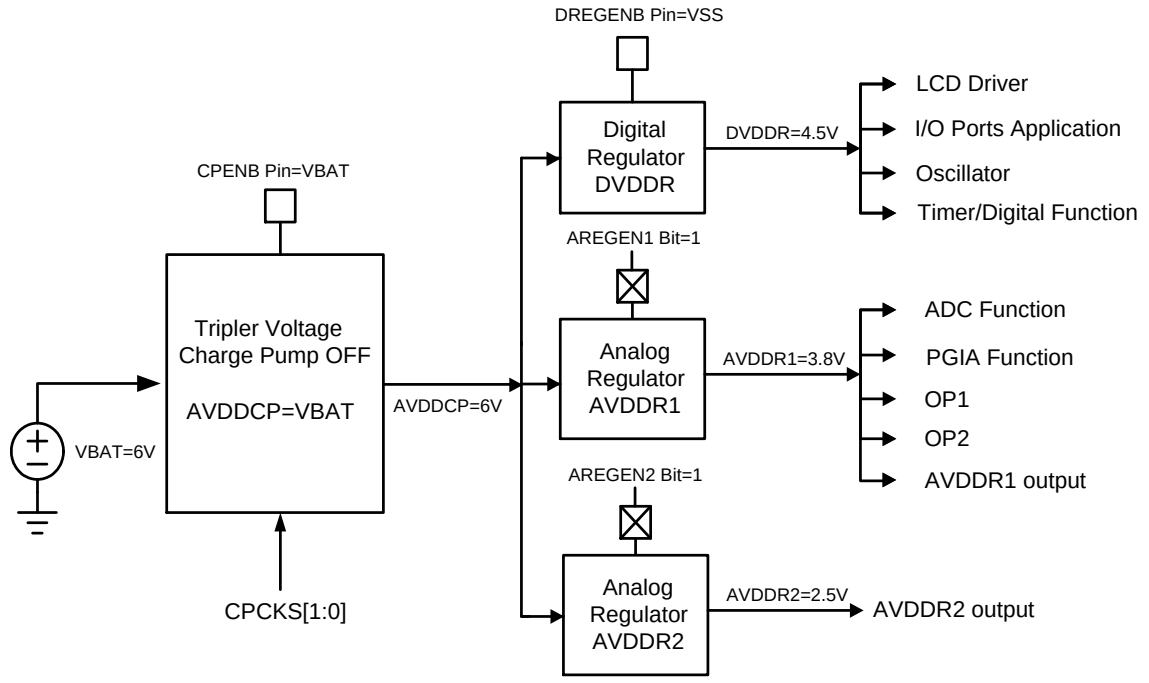
Digital Regulator supplied power for oscillator, Logic circuit, LCD and I/O application. Analog Regulator supplied power for ADC, PGIA and OP applications.

To obtain maximum range of ADC output, the ADC maximum input signal voltage should be close to but can't over AVREFH and the minimum input signal voltage should be close to but can't be under the Vss, Choosing a suitable reference voltage and a suitable gain of PGIA can reach this purpose.



VBAT=3V Block Diagram of Charge-Pump Regulator, PGIA and OPA module

- ***Note: CPENB=Vss Enable triple voltage pump function***
  CPENB=Vbat Disable triple voltage pump function
- ***Note: DREGENB=Vss Enable Digital Regulator***
  DREGENB=Vbat Disable Digital Regulator
- ***Note: The voltage of Triple Voltage Pump output is AVDDCP = VBAT * 3***



VBAT=3V Block Diagram of Charge-Pump Regulator, PGIA and OPA module

10.3 VOLTAGE CHARGE-PUMP REGULATOR (CPR)

SN8P1819A is built in a Charge Pump (CP), which can provide a stable 4.5V (pin DVDDR) for Digital Regulator and 3.8V (PIN AVDDR1)and 2.5V(PIN AVDDR2)with total maximum 10mA current driving capacity. Register CPM can control Analog Regulator working mode, if user turns on the analog regulator (AREGEN1 = 1), then power of PGIA, OPA and ADC is come from AVDDR=3.8V, be careful to turn on analog regulator first before enabling PGIA, OPA and ADC.

10.3.1 CPM-Regulator Mode Register

08AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPM			-	-	-	-	AREGEN2	AREGEN1
Read/Write							R/W	R/W
After reset							0	0

Bit 0 **AREGEN1:** Analog Regulator (AVDDR1 3.8V) ON/OFF control bit.
1 = Enable Analog Regulator.
0 = Disable. Analog Regulator.

Bit 1 **AREGEN2:** Analog Regulator (AVDDR2 2.5V) ON/OFF control bit.
1 = Enable Analog Regulator.
0 = Disable. Analog Regulator.

Pin CPENB, DREGENB, and bit AREGEN1,AREGEN2 controlled Charge-Pump, Regulator working mode. By change these setting, Charge-Pump, Regulator can be set as different voltage.

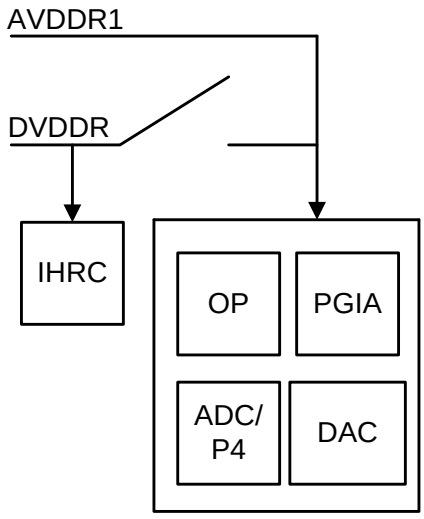
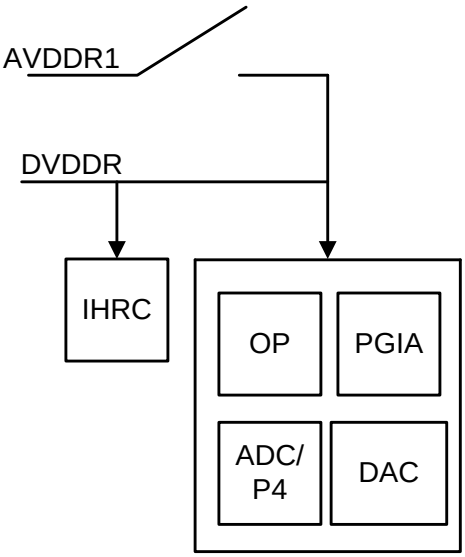
✸ ***Note: Don't enable triple Charge Pump when VBAT>3.5V.***

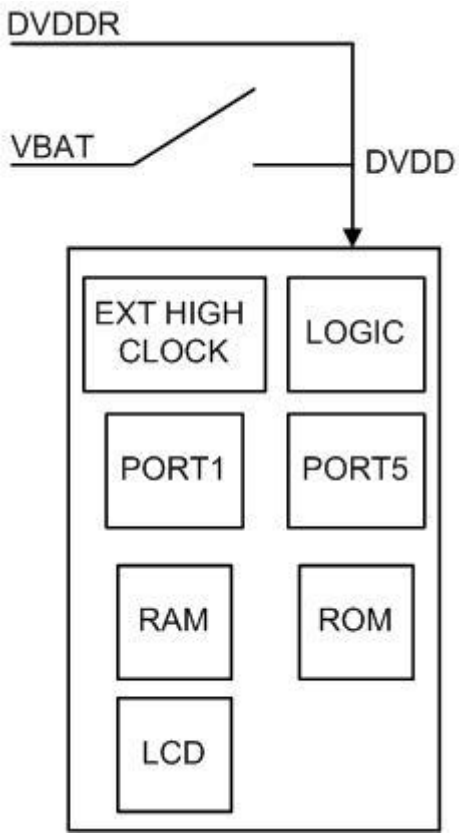
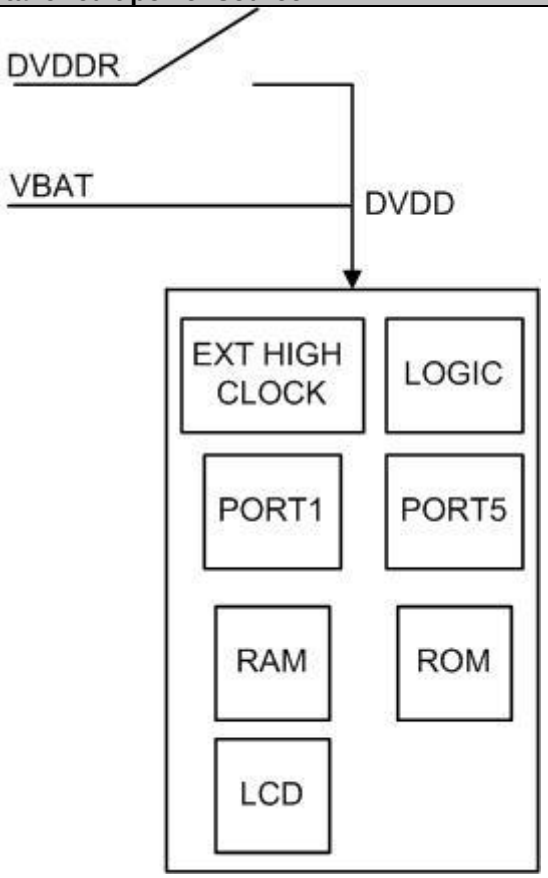
CHARGE PUMP、REGULATOR SELECTION TABLE					
VBAT PIN	**CPENB PIN**	**DREGENB PIN**	**AREGEN1 Register**	**AREGEN2 Register**	**RESULT**
3V	VSS	VSS	0	0	AVDDCP=9V、DVDDR=4.5V、AVDDR1=0V、AVDDR2=0V
				1	AVDDCP=9V、DVDDR=4.5V、AVDDR1=0V、AVDDR2=2.5V
			1	0	AVDDCP=9V、DVDDR=4.5V、AVDDR1=3.8V、AVDDR2=0V
				1	AVDDCP=9V、DVDDR=4.5V、AVDDR1=3.8V、AVDDR2=2.5V
6V	VBAT	VSS	0	0	AVDDCP=6V、DVDDR=4.5V、AVDDR1=0V、AVDDR2=0V
				1	AVDDCP=6V、DVDDR=4.5V、AVDDR1=0V、AVDDR2=2.5V
			1	0	AVDDCP=6V、DVDDR=4.5V、AVDDR1=3.8V、AVDDR2=0V
				1	AVDDCP=6V、DVDDR=4.5V、AVDDR1=3.8V、AVDDR2=2.5V

✸ ***Note: CPENB/DREGENB connect 100 ohm resistor to VSS or VBAT.***

DVDDR output voltage	
DREGENB PIN	**DVDDR voltage**
connect to VSS	4.5V
connect to VBAT	VBAT
AVDDCP output voltage	
CPENB PIN	**AVDDCP voltage**
connect to VSS	3*VBAT
connect to VBAT	VBAT
AVDDR1 output voltage	
AREGEN1 bit	**AVDDR1 voltage**
1	3.8V
0	0
AVDDR2 output voltage	
AREGEN2 bit	**AVDDR2 voltage**
1	2.5V
0	0

10.3.2 Power source Block Diagrem

Analog circuit power source	
AREGEN1 Register=1	**AREGEN1 Register=0**
Analog circuit power source=AVDDR1	**Analog circuit power source=DVDDR**
	

Digital circuit power source	
DVDD PIN connect to DVDDR PIN	**DVDD PIN connect to VBAT PIN**
Digital circuit power source =DVDDR	**Digital circuit power source =VBAT**
	

10.3.3 CPCKS-Charge Pump Clock Register

08BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPCKS	-	-	-	-	-	-	CPCKS1	CPCKS0
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

Bit [1:0] **CPCKS[1:0]:** sets the Charge-Pump working clock .

CPCKS [1:0]	CP Working Clock
00	Fastest clock
01	Faster clock
10	Slower clock
11	Slowest clock

✷ ***Note: Set CPCKS=[11] before enter Power down, slow and Green mode to save power..***

Example: Charge-Pump setting (Fosc = 4M X'tal)

```
@CPREG_Init:
        MOV     A, #01h
        B0MOV   CPM, A          ; Enable Analog Regualtor.
        MOV     A, #000
        B0MOV   CPCKS, A        ; Set CPCKS = 00

@Delay_10ms:
        CALL    @Wait_10ms      ; Delay 10ms for Regulator Stabilize
```

10.4 PROGRAMMABLE GAIN INSTRUMENTATION AMPLIFIER (PGIA)

SN8P2839 includes a low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selectable gains stage one of16x, 32x, 64x, 128x and gains stage two of 1.3x ~ 2.5x by register PGIAM.

PGIA gain calculation:

$$\text{PGIAOUT} = \frac{1}{2} \times \Delta V \times Gain1 \times Gain2 + Gain2 \times \left(PGIABIAS - Vdc\right) + PGIABIAS$$

$$\Delta V = (AI^{+}) - (AI^{-})$$

$$Vdc = \frac{1}{2} \times \left[(AI^{+}) + (AI^{-})\right]$$

$$\text{PGIA input range} : 0.4\text{V} \sim 2.8\text{V}$$

$$\text{PGIA output range} : \text{under } 2.8\text{V}$$

10.4.1 PGIAM- PGIA Mode Register

0BCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PGIA1M	P2GS3	P2GS2	P2GS1	P2GS0	-	P1GS1	P1GS0	PGIA1ENB
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

Bit0 **PGIA1ENB:** PGIA1 function enable control bit.
0 = Disable PGIA1 function
1 = Enable PGIA1 function

Bit[2:1] **P1GS [1:0]:** PGIA1 Gain1 selection control bit

Bit[7:1] **P2GS [3:0]:** PGIA1 Gain2 selection control bit

		P1GS [1:0]			
		00	**01**	**10**	**11**
P2GS [3:0]	**Gain**	**16**	**32**	**64**	**128**
0000	**1.32558**	21.2093	42.4186	84.8372	169.674
0001	**1.38095**	22.0952	44.1905	88.381	176.762
0010	**1.43902**	23.0244	46.0488	92.0976	184.195
0011	**1.5**	24	48	96	192
0100	**1.5641**	25.0256	50.0513	100.103	200.205
0101	**1.63158**	26.1053	52.2105	104.421	208.842
0110	**1.7027**	27.2432	54.4865	108.973	217.946

0111	**1.77778**	28.4444	56.8889	113.778	227.556
1000	**1.85714**	29.7143	59.4286	118.857	237.714
1001	**1.94118**	31.0588	62.1176	124.235	248.471
1010	**2.0303**	32.4848	64.9697	129.939	259.879
1011	**2.125**	34	68	136	272
1100	**2.22581**	35.6129	71.2258	142.452	284.903
1101	**2.33333**	37.3333	74.6667	149.333	298.667
1110	**2.44828**	39.1724	78.3448	156.69	313.379
1111	**2.57143**	41.1429	82.2857	164.571	329.143

0BAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PGIA2M	P4GS3	P4GS2	P4GS1	P4GS0	-	P3GS1	P3GS0	PGIA2ENB
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

Bit0 **PGIA2ENB:** PGIA2 function enable control bit.
0 = Disable PGIA2 function
1 = Enable PGIA2 function

Bit[2:1] **P3GS [1:0]:** PGIA2 Gain1 selection control bit

Bit[7:1] **P4GS [3:0]:** PGIA2 Gain2 selection control bit

		P3GS [1:0]			
		00	**01**	**10**	**11**
P4GS [3:0]	**Gain**	**16**	**32**	**64**	**128**
0000	**1.32558**	21.2093	42.4186	84.8372	169.674
0001	**1.38095**	22.0952	44.1905	88.381	176.762
0010	**1.43902**	23.0244	46.0488	92.0976	184.195
0011	**1.5**	24	48	96	192
0100	**1.5641**	25.0256	50.0513	100.103	200.205
0101	**1.63158**	26.1053	52.2105	104.421	208.842
0110	**1.7027**	27.2432	54.4865	108.973	217.946
0111	**1.77778**	28.4444	56.8889	113.778	227.556
1000	**1.85714**	29.7143	59.4286	118.857	237.714
1001	**1.94118**	31.0588	62.1176	124.235	248.471
1010	**2.0303**	32.4848	64.9697	129.939	259.879
1011	**2.125**	34	68	136	272
1100	**2.22581**	35.6129	71.2258	142.452	284.903
1101	**2.33333**	37.3333	74.6667	149.333	298.667
1110	**2.44828**	39.1724	78.3448	156.69	313.379
1111	**2.57143**	41.1429	82.2857	164.571	329.143

10.4.2 PGIACKS- PGIA CLOCK SELECTION

0BBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PGIA1CKS	PGIA1CKS7	PGIA1CKS6	PGIA1CKS5	PGIA1CKS4	PGIA1CKS3	PGIA1CKS2	PGIA1CKS1	PGIA1CKS0
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

PGIA1CKS [7:0] Sets the PGIA1 working clock, the suggestion PGIA1 clock is 12.5K Hz.

Refer to the following table for PGIA1CKS [7:0] register value setting in different Fosc frequency.

PGIA1 Clock= (Fhosc / (256-PGIA1CKS [7:0])) / 8

PGIA1CKS [7:0]	Fhosc	PGIA1 Working Clock
216	4M	(4M / 40) / 8 = 12.5K
176	8M	(8M / 80) / 8 = 12.5K
156	10M	(10M / 100) / 8 = 12.5K
136	12M	(12M / 120) / 8 = 12.5K
96	16M	(16M/160)/8=12.5K
56	20M	(20M / 200) / 8 = 12.5K

➢ ***Note: In general application, PGIA working clock is 12.5K Hz..***

0B9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PGIA2CKS	PGIA2CKS7	PGIA2CKS6	PGIA2CKS5	PGIA2CKS4	PGIA2CKS3	PGIA2CKS2	PGIA2CKS1	PGIA2CKS0
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

PGIA2CKS [7:0] Sets the PGIA2 working clock, the suggestion PGIA2 clock is 12.5K Hz.

Refer to the following table for PGIA2CKS [7:0] register value setting in different Fosc frequency.

PGIA2 Clock= (Fhosc / (256-PGIA2CKS [7:0])) / 8

PGIA2CKS [7:0]	Fhosc	PGIA2 Working Clock
216	4M	(4M / 40) / 8 = 12.5K
176	8M	(8M / 80) / 8 = 12.5K
156	10M	(10M / 100) / 8 = 12.5K
136	12M	(12M / 120) / 8 = 12.5K
96	16M	(16M/160)/8=12.5K
56	20M	(20M / 200) / 8 = 12.5K

➢ ***Note: In general application, PGIA working clock is 12.5K Hz..***

Example: PGIA1 setting (Fosc = 4M X'tal)

```
@CPREG_Init:
        MOV     A, #01h
        B0MOV   CPM, A          ; Enable Analog Regualtor.
        MOV     A, #000
        B0MOV   CPCKS, A        ; Set CPCKS = 00

@Delay_10ms:
        CALL    @Wait_10ms      ; Delay 10ms for Charge-Pump Stabilize

@PGIA_Init:
        MOV     A, #030h
        PGIAM   AMPM, A         ; Selected PGIA Gain1=16, Gain2=1.5, total Gain=24
        MOV     A, #0216
```

```
        B0MOV       PGIA1CKS, A     ;Set PGIACKS = 216 for PGIA working clock = 12.5K @ 4M
                                    X'tal

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

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

10.5 OPA – OPERATIONAL AMPLIFIER

SN8P2839 provide two built-in operational amplifiers (OPA) for low pass filer or constant current source circuit.

10.5.1 OPM- OPA Mode Register

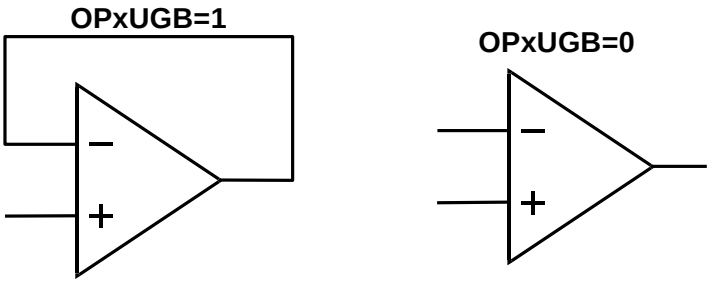
0BDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPM	-	-	-	-	OP2UGB	OP1UGB	OP2ENB	OP1IENB
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

OP1ENB: Operation amplifier one (OP1) function enable control bit.
0 = Disable OP1 function
1 = Enable OP1 function

OP2ENB: Operation amplifier two (OP2) function enable control bit.
0 = Disable OP2 function
1 = Enable OP2 function

OP1UGB: OP1 Unit Gain Buffer function enable control bit.
0 = Disable OP1 Unit Gain Buffer function.
1 = Enable OP1 Unit Gain Buffer function. OP1OUT short to OP1IN-

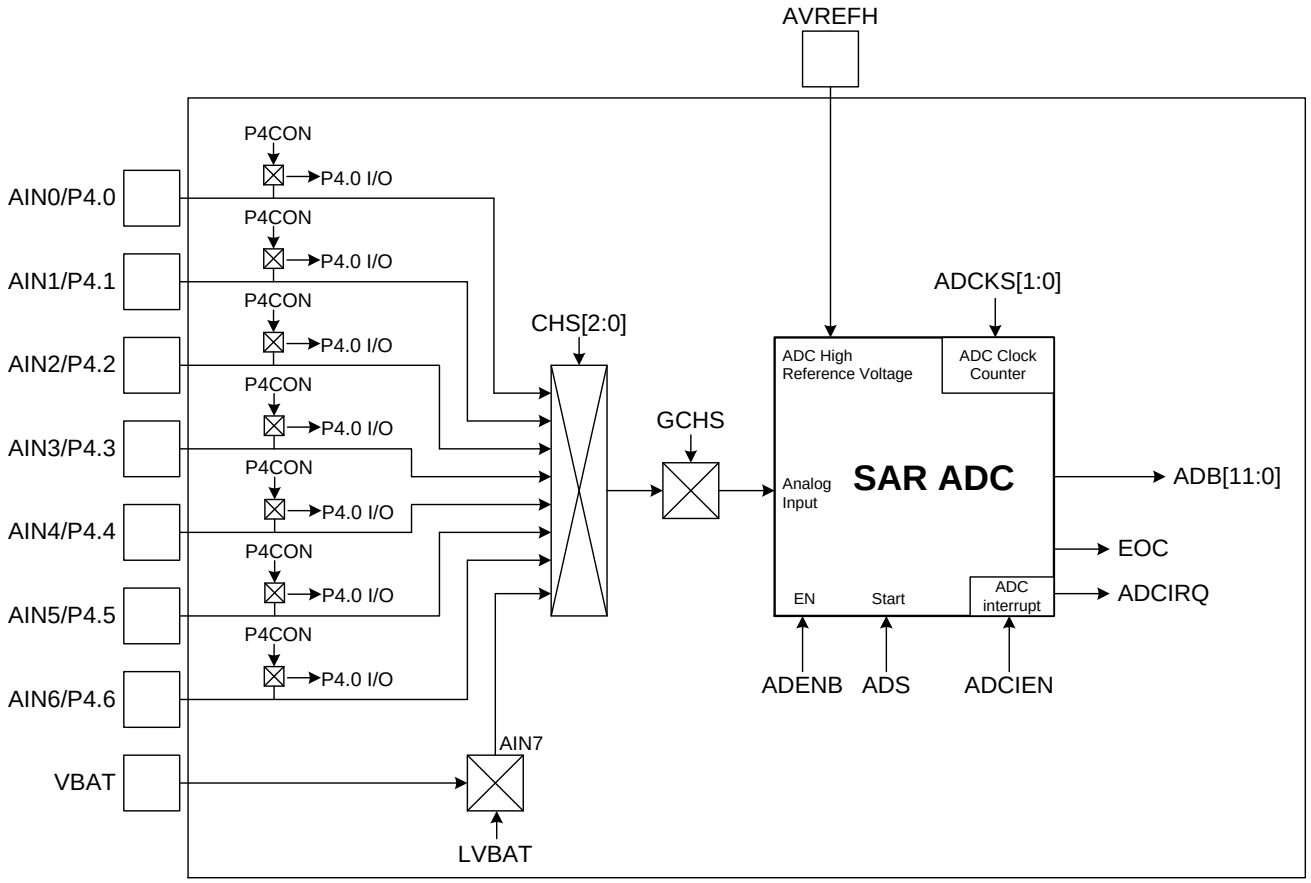
OP2UGB: OP2 Unit Gain Buffer function enable control bit.
0 = Disable OP2 Unit Gain Buffer function.
1 = Enable OP2 Unit Gain Buffer function. OP2OUT short to OP2IN-



11 7+1 CHANNEL ANALOG TO DIGITAL CONVERTER

11.1 OVERVIEW

This analog to digital converter has 7-input sources with up to 4096-step resolution to transfer analog signal into 12-bits digital data. The ADC build in 1/2VBAT or 1/4VBAT voltage detect. The sequence of ADC operation is to select input source (AIN0 ~ AIN6) at first, then set GCHS and ADS bit to "1" to start conversion. When the conversion is complete, the ADC circuit will set EOC bit to "1" and final value output in ADB register.



- ✸ ***Note: For 12-bit resolution the conversion time is 16 steps***
- ✸ ***Note: P4 shared with ADC channels.***
  - ● ***When AREGEN1=1(AVDDR1 Enable),then P4 I/O voltage=AVDDR1=3.8V.***
  - ● ***When AREGEN1=0(AVDDR1 Disable),then P4 I/O voltage=DVDDR.***
- ✸ ***Note: The analog input level must be between the AVREFH and VSS.***
- ✸ ***Note: The AVREFH level must be between the AVDDR1(3.8V) and 2.0V.***
- ✸ ***Note: Build in 1/2 VBAT, 1/4 VBAT detect.***
- ✸ ***Note: ADC programming notice:***
  - ● ***Set ADC input pin I/O direction as input mode***
  - ● ***Disable pull-up resistor of ADC input pin***
  - ● ***Set related bit of P4CON register to avoid extra power consumption in power down mode.***
  - ● ***Delay 100uS after enable ADC (set ADENB = "1") to wait ADC circuit ready for conversion.***
  - ● ***Disable ADC (set ADENB = "0") before enter sleep mode to save power consumption.***

11.2 ADM REGISTER

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	LVBAT	CHS2	CHS1	CHS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **ADENB:** ADC control bit.
0 = Disable.
1 = Enable.

Bit 6 **ADS:** ADC start bit.
0 = Stop.
1 = Starting.

Bit 5 **EOC:** ADC status bit.
0 = Progressing.
1 = End of converting and reset ADS bit.

Bit 4 **GCHS:** Global channel select bit.
0 = Disable AIN channel.
1 = Enable AIN channel.

Bit 3 **LVBAT:** AIN7 internal channel select bit,for voltage detect function.
0 = 1/2 VBAT.
1 = 1/4 VBAT.

Bit[2:0] **CHS[2:0]:** ADC input channels select bit.
000 = AIN0, 001 = AIN1, 010 = AIN2, 011 = AIN3
100 = AIN4, 101 = AIN5, 110 = AIN6, 111 = AIN7

✸ ***Note: If ADENB = 1, users should set P4.n/AINn as input mode without pull-up. System doesn't set automatically. If P4CON.n is set, the P4.n/AINn's digital I/O function including pull-up is isolated.***

11.3 ADR REGISTERS

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	-	ADCKS2	ADCKS1	ADCKS0	ADB3	ADB2	ADB1	ADB0
Read/Write	-	R/W	R/W	R/W	R	R	R	R
After reset	-	0	0	0	-	-	-	-

Bit 6,5,4 **ADCKS [2:0]:** ADC's clock source select bit.

ADCKS2	ADCKS1	ADCKS0	ADC Clock Source
0	0	0	Fcpu/16
1	0	0	Fcpu/8
1	0	1	Fcpu/4
1	1	0	Fcpu/2
1	1	1	Fcpu

Bit [3:0] **ADB [3:0]:** ADC Low-byte data buffer.

11.4 ADB REGISTERS

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
Read/Write	R	R	R	R	R	R	R	R
After reset	-	-	-	-	-	-	-	-

Bit[7:0] **ADB[7:0]:** ADC high-byte data buffer.

✸ ***Note: ADB11~ADB0 bits for 12-bit ADC resolution.***
✸ ***Note: ADB11~ADB4 bits for 8-bit ADC resolution.***

ADB is ADC data buffer to store AD converter result. The ADB is only 8-bit register including bit 4~bit11 ADC data. To combine ADB register and the low-nibble of ADR will get full 12-bit ADC data buffer. The ADC buffer is a read-only register.

The AIN's input voltage v.s. ADB's output data

AIN n	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
0/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	0
1/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	1
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
4094/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	0
4095/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	1

✸ ***Note: ADC buffer ADB[11:0] initial value after reset is unknown.***

11.5 P4CON REGISTERS

The Port 4 is shared with ADC input function. Only one pin of port 4 can be configured as ADC input in the same time by ADM register. The other pins of port 4 are digital I/O pins. Connect an analog signal to COMS digital input pin, especially the analog signal level is about 1/2 VDD will cause extra current leakage. In the power down mode, the above leakage current will be a big problem. Unfortunately, if users connect more than one analog input signal to port 4 will encounter above current leakage situation. P4CON is Port4 Configuration register. Write "1" into P4CON [7:0] will configure related port 4 pin as pure analog input pin to avoid current leakage.

0BEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	-	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
Read/Write	-	W	W	W	W	W	W	W
After reset	-	0	0	0	0	0	0	0

Bit[4:0] **P4CON[6:0]:** P4.n configuration control bits.
0 = P4.n can be an analog input (ADC input) or digital I/O pins.
1 = P4.n is pure analog input, can't be a digital I/O pin.

✸ ***Note: When Port 4.n is general I/O port not ADC channel, P4CON.n must set to "0" or the Port 4.n digital I/O signal would be isolated.***

11.6 ADC CONVERTING TIME

12-bit ADC conversion time = 1/(ADC clock /4)*16 sec

Fcpu = 4MHz (High clock, Fosc is 4MHz and Fcpu = Fosc/1)

ADCKS2	ADCKS1	ADCKS0	ADC Clock	ADC conversion time
0	0	0	Fcpu/16	1/(4MHz/16/4)*16 = 256 us
1	0	0	Fcpu/8	1/(4MHz/8/4)*16 = 128 us
1	0	1	Fcpu/4	1/(4MHz/4/4)*16 = 64 us
1	1	0	Fcpu/2	1/(4MHz/2/4)*16 = 32 us
1	1	1	Fcpu	1/(4MHz/4)*16 = 16 us

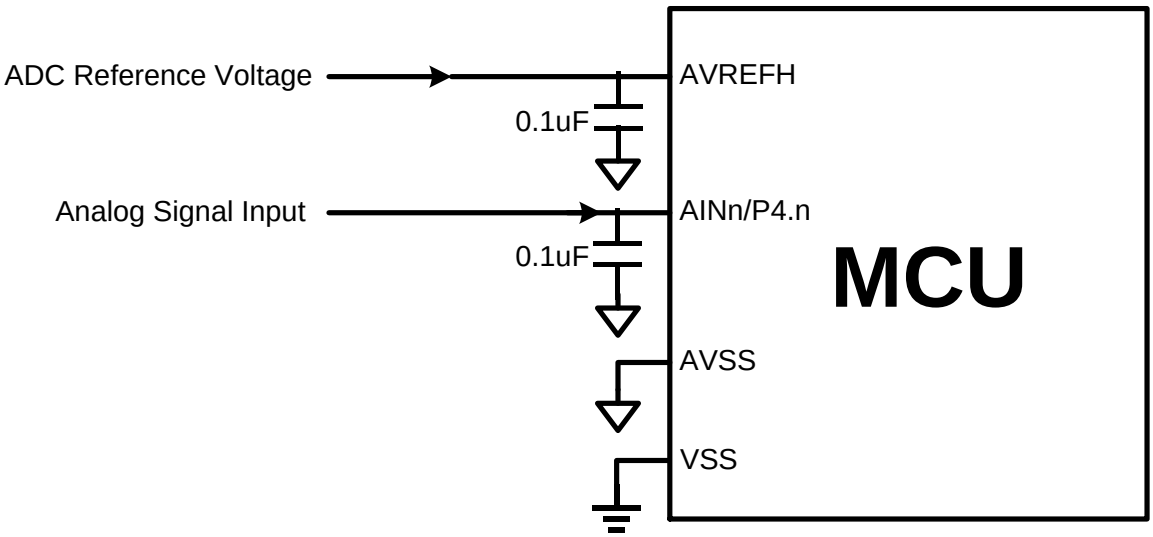
11.7 ADC ROUTINE EXAMPLE

➢ **Example : Configure AIN0 as 12-bit ADC input and start ADC conversion then enter power down mode.**

ADC0:

```
        MOV     A, #0FEh
        B0MOV   P4UR, A         ; Disable P4.0 pull-up resistor
        B0BCLR  FP40M           ; Set P4.0 as input pin
        MOV     A, #01h
        B0MOV   P4CON, A        ; Set P4.0 as analog input
        MOV     A, #00H
        B0MOV   ADR, A          ; To set ADC clock = Fcpu/16.
        MOV     A,#10H
        B0MOV   ADM,A           ; To enable GCHS and set AIN0 input.
        B0BSET  FADENB          ; Enable ADC circuit
        CALL    Delay100uS      ; Delay 100uS to wait ADC circuit ready for conversion
        B0BSET  FADS            ; To start conversion
WADC0:
        B0BTS1  FEOC            ; To skip, if end of converting =1
        JMP     WADC0           ; else, jump to WADC0
        B0MOV   A,ADB           ; To get AIN0 input data bit11 ~ bit4
        B0MOV   Adc_Buf_Hi, A
        B0MOV   A,ADR           ; To get AIN0 input data bit3 ~ bit0
        AND     A, 0Fh
        B0MOV   Adc_Buf_Low, A
Power_Down  .                   .
        B0BCLR  FADENB          ; Disable ADC circuit
        B0BCLR  FCPUM1
        B0BSET  FCPUM0          ; Enter sleep mode
```

11.8 ADC CIRCUIT



ADC reference high voltage is from external voltage. The capacitor (0.1uF) between AVREFH and AVSS to stable AVERFH voltage.

12 1 CHANNEL DIGITAL TO ANALOG CONVERTER

The DAC structure is 12-bit resolution, it can generate a analog signal on DAO pin. DAO output voltage range= 0V ~ DAREF. Max DAREF input voltage=AVDDR1 (3.8V).

12.1 DAC Register

0F7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAM	DAEN							
Read/Write	R/W							
After Reset	0							

Bit 7 **DAEN:** DAC control bit.
0 = Disable DAC function.
1 = Enable DAC function.

0F8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DABL					DAB3	DAB2	DAB1	DAB0
Read/Write					W	W	W	W
After Reset					0	0	0	0

Bit [3:0] **DAB [3:0]** = The low-nibble byte is DAC normal mode output configuration data buffer.

0F9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DABH	DAB11	DAB10	DAB9	DAB8	DAB7	DAB6	DAB5	DAB4
Read/Write	R/W	R/W	R/W	R/W	W	W	W	W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **DAB [11:4]** = The high-nibble byte is DAC normal mode output configuration data buffer.

- ***Note: DAB[11:0] written sequence is write DAB[3:0] first, and then write DAB[11:4]. End of writing DAB[11:4] signal means the end of DAB written.***
- ***Note: Max DAREF input voltage=AVDDR1 (3.8V).***

The DAB register, from bit0 to bit11, to generate analog signal on DAO pin. The DAO output signal voltage range is 0V ~ Vdd.

The DAB's data vs DAO's output voltage as following:

DAB[7:0]								DAB[3:0]				DAO Output
DAB11	DAB10	DAB9	DAB8	DAB7	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0	Voltage (V)
0	0	0	0	0	0	0	0	0	0	0	0	VSS
0	0	0	0	0	0	0	0	0	0	0	1	1/4095*DAREF
0	0	0	0	0	0	0	0	0	0	1	0	2/4095*DAREF
0	0	0	0	0	0	0	0	0	0	1	1	3/40955*DAREF
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	1	1	1	1	1	0	4094/4095*DAREF
1	1	1	1	1	1	1	1	1	1	1	1	DAREF

13 SERIAL INPUT/OUTPUT TRANSCEIVER (SIO)

13.1 OVERVIEW

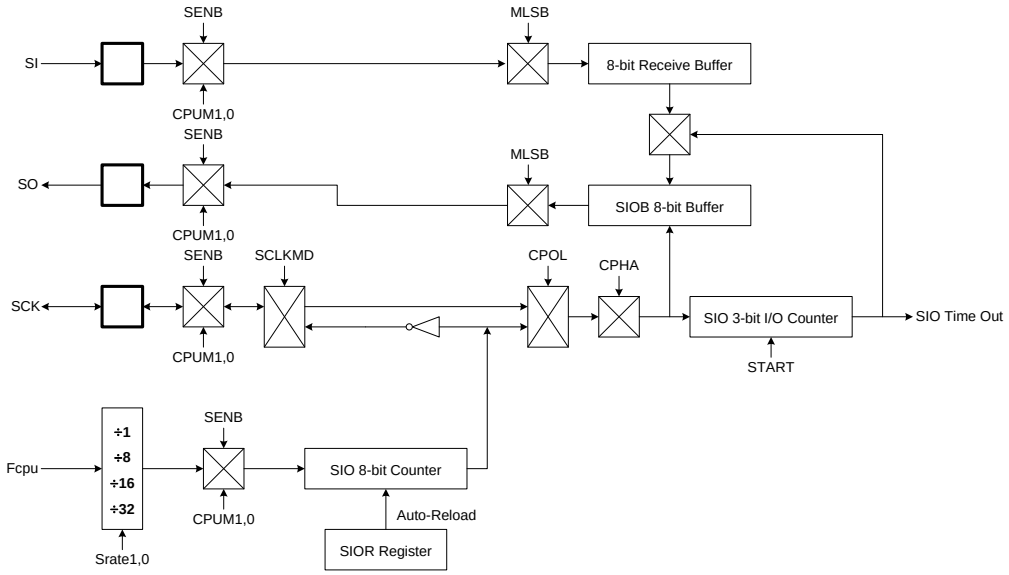
The SIO (serial input/output) transceiver is a serial communicate interface for data exchanging from one MCU to one MCU or other hardware peripherals. It is a simple 8-bit interface without a major definition of protocol, packet or control bits. The SIO transceiver includes three pins, clock (SCK), data input (SI) and data output (SO) to send data between master and slaver terminals. The SIO interface builds in 8-mode which are the clock idle status, the clock phases and data fist bit direction. The 8-bit mode supports most of SIO/SPI communicate format.

The SIO features include the following:

- **Full-duplex, 3-wire synchronous data transfer.**
- **Master (SCK is clock output) or Slave (SCK is clock input) operation.**
- **MSB/LSB first data transfer.**
- **The start phase of data sampling location selection is 1st-phase or 2nd-phase controlled register.**
- **SCK, SI, SO are programmable open-drain output pin for multiple salve devices application.**
- **Two programmable bit rates (Only in master mode).**
- **End-of-Transfer interrupt.**

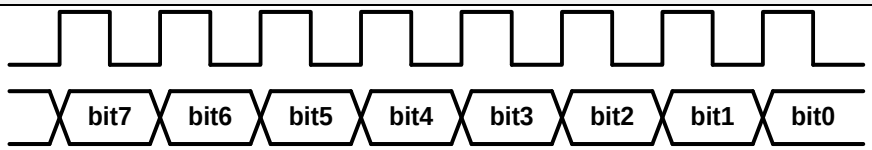
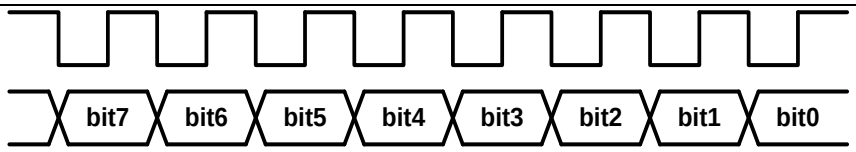
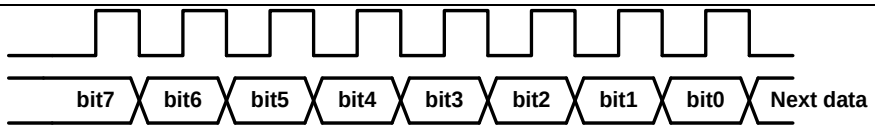
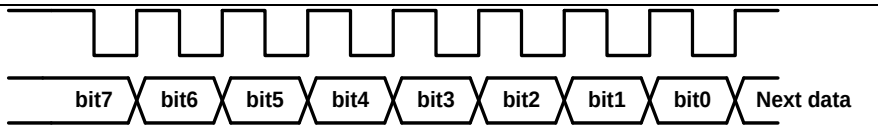
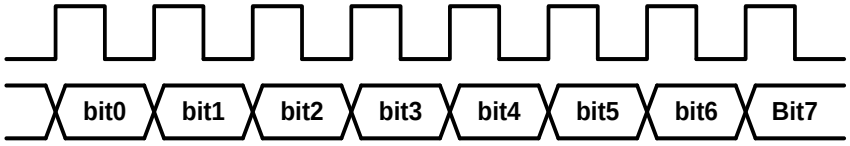
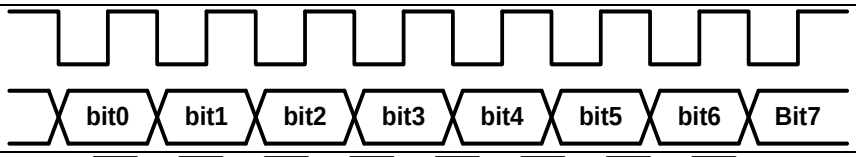
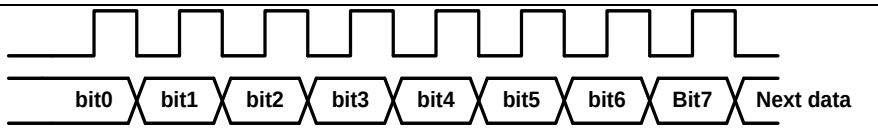
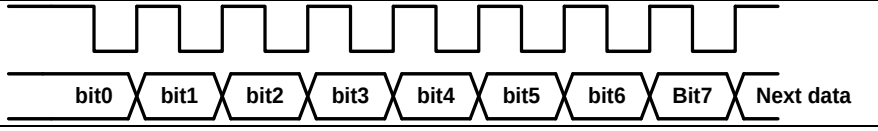
13.2 SIO OPERATION

The SIOM register can control SIO operating function, such as: transmit/receive, clock rate, data transfer direction, SIO clock idle status and clock control phase and starting this circuit. This SIO circuit will transmit or receive 8-bit data automatically by setting SENB and START bits in SIOM register. The SIO data buffer is double buffer design. When the SIO operating, the SIOB register stores transfer data and one internal buffer stores receive data. When SIO operation is successfully, the internal buffer reloads into SIOB register automatically. The SIO 8-bit counter and SIOR register are designed to generate SIO's clock source with auto-reload function. The 3-bit I/O counter can monitor the operation of SIO and announce an interrupt request after transmitting/ receiving 8-bit data. After transferring 8-bit data, this circuit will be disabled automatically and re-transfer data by programming SIOM register. CPOL bit is designed to control SIO clock idle status. CPHA bit is designed to control the clock edge direction of data receive. CPOL and CPHA bits decide the SIO format. The SIO data transfer direction is controlled by MLSB bit to decide MSB first or LSB first.



SIO Interface Circuit Diagram

The SIO supports 8-mode format controlled by MLSB, CPOL and CPHA bits. The edge direction is “Data Transfer Edge”. When setting rising edge, that means to receive and transmit one bit data at SCK rising edge, and data transition is at SCK falling edge. When setting falling edge, that means to receive and transmit one bit data at SCK falling edge, and data transition is at SCK rising edge.
“CPHA” is the clock phase bit controls the phase of the clock on which data is sampled. When CPHA=1, the SCK first edge is for data transition, and receive and transmit data is at SCK 2nd edge. When CPHA=0, the 1st bit is fixed already, and the SCK first edge is to receive and transmit data. The SIO data transfer timing as following figure:

MLSB	CPOL	CPHA	Diagrams	Description
0	0	1		SCK idle status = Low. The transfer first bit = MSB. SCK data transfer edge = Falling edge.
0	1	1		SCK idle status = High. The transfer first bit = MSB. SCK data transfer edge = Rising edge.
0	0	0		SCK idle status = Low. The transfer first bit = MSB. SCK data transfer edge = Rising edge.
0	1	0		SCK idle status = High. The transfer first bit = MSB. SCK data transfer edge = Falling edge.
1	0	1		SCK idle status = Low. The transfer first bit = LSB. SCK data transfer edge = Falling edge.
1	1	1		SCK idle status = High. The transfer first bit = LSB. SCK data transfer edge = Rising edge.
1	0	0		SCK idle status = Low. The transfer first bit = LSB. SCK data transfer edge = Rising edge.
1	1	0		SCK idle status = High. The transfer first bit = LSB. SCK data transfer edge = Falling edge.

SIO Data Transfer Timing

The SIO supports interrupt function. SIOIEN is SIO interrupt function control bit. SIOIEN=0, disable SIO interrupt function. SIOIEN=1, enable SIO interrupt function. When SIO interrupt function enable, the program counter points to interrupt vector (ORG 8) to do SIO interrupt service routine after SIO operating. SIOIRQ is SIO interrupt request flag, and also to be the SIO operating status indicator when SIOIEN = 0, but cleared by program. When SIO operation finished, the SIOIRQ would be set to "1", and the operation is the inverse status of SIO "START" control bit.
The SIOIRQ and SIO START bit indicating the end status of SIO operation is after one 8-bit data transferring. The duration from SIO transfer end to SIOIRQ/START active is about **"1/2*SIO clock"**, means the SIO end indicator doesn't active immediately.

✸ ***Note: The first step of SIO operation is to setup the SIO pins' mode. Enable SENB, select CPOL and CPHA bits. These bits control SIO pins' mode.***

13.3 SIOM MODE REGISTER

SIOM initial value = 0000 0000

0B4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOM	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **SENB:** SIO function control bit.
0 = Disable SIO function. P5.0~P5.2 are GPIO.
1 = Enable SIO function. P5.0~P5.2 are SIO pins. SIO pin structure can be push-pull structure and open-drain structure controlled by P1OC register.

Bit 6 **START:** SIO progress control bit.
0 = End of transfer.
1 = SIO transmitting.

Bit [5:4] **SRATE1,0:** SIO's transfer rate select bit. **These 2-bits are workless when SCKMD=1.**
00 = fcpu.
01 = fcpu/32
10 = fcpu/16
11 = fcpu/8.

Bit 3 **MLSB:** MSB/LSB transfer first.
0 = MSB transmit first.
1 = LSB transmit first.

Bit 2 **SCKMD:** SIO's clock mode select bit.
0 = Internal. (Master mode)
1 = External. (Slave mode)

Bit 1 **CPOL:** SCK idle status control bit.
0 = SCK idle status is low status.
1 = SCK idle status is high status.

Bit 0 **CPHA:** The Clock Phase bit controls the phase of the clock on which data is sampled.
0 = Data receive at the first clock phase.
1 = Data receive at the second clock phase.

Because SIO function is shared with Port5 for P5.0 as SCK, P5.1 as SI and P5.2 as SO. The following table shows the Port5[2:0] I/O mode behavior and setting when SIO function enable and disable.

SENB=1 (SIO Function Enable)		
P5.0/SCK	(SCKMD=1) SIO source = External clock	P5.0 will change to Input mode automatically, no matter what P5M setting.
	(SCKMD=0) SIO source = Internal clock	P5.0 will change to Output mode automatically, no matter what P5M setting.
P5.1/SI	P5.1 must be set as Input mode in P5M ,or the SIO function will be abnormal	
P5.2/SO	SIO = Transmitter/Receiver	P5.2 will change to Output mode automatically, no matter what P5M setting.
SENB=0 (SIO Function Disable)		
P5.0/P5.1/P5.2	Port5[2:0] I/O mode are fully controlled by P5M when SIO function Disable	

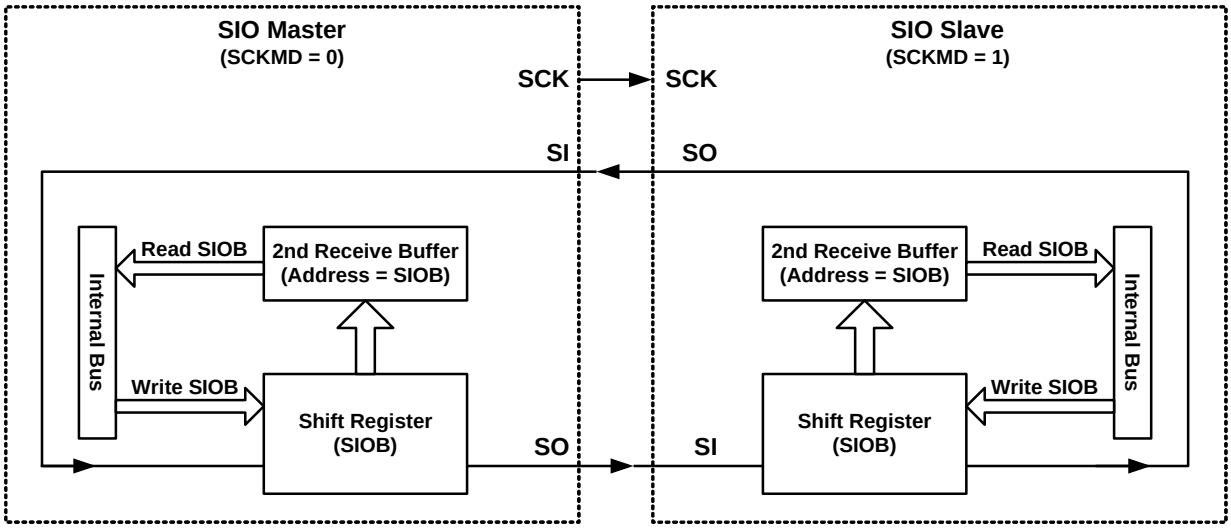
✷ ***Note:***
1. ***If SCKMD=1 for external clock, the SIO is in SLAVE mode. If SCKMD=0 for internal clock, the SIO is in MASTER mode.***
2. ***Don't set SENB and START bits in the same time. That makes the SIO function error.***
3. ***SIO pin can be push-pull structure and open-drain structure controlled by P1OC register.***

13.4 SIOB DATA BUFFER

SIOB initial value = 0000 0000

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOB	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0
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

SIOB is the SIO data buffer register. It stores serial I/O transmit and receive data. The system is single-buffered in the transmit direction and double-buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SIOB Data Register before the entire shift cycle is completed. When receiving data, however, a received byte must be read from the SIOB Data Register before the next byte has been completely shifted in. Otherwise, the first byte is lost. Following figure shows a typical SIO transfer between two micro-controllers. Master MCU sends SCK for initial the data transfer. Both master and slave MCU must work in the same clock edge direction, and then both controllers would send and receive data at the same time.



SIO Data Transfer Diagram

13.5 SIOR REGISTER DESCRIPTION

SIOR initial value = 0000 0000

0B5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOR	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

The SIOR is designed for the SIO counter to reload the counted value when end of counting. It is like a post-scaler of SIO clock source and let SIO has more flexible to setting SCK range. Users can set the SIOR value to setup SIO transfer time. To setup SIOR value equation to desire transfer time is as following.

SCK frequency = SIO rate / (256 - SIOR)

SIOR = 256 - (1 / (SCK frequency) * SIO rate)

- **Example: Setup the SIO clock to be 5KHz. Fosc = 3.58MHz. SIO's rate = Fcpu = Fosc/4.**

SIOR = 256 – (1/(5KHz) * 3.58MHz/4)
= 256 – (0.0002*895000)
= 256 – 179
= 77

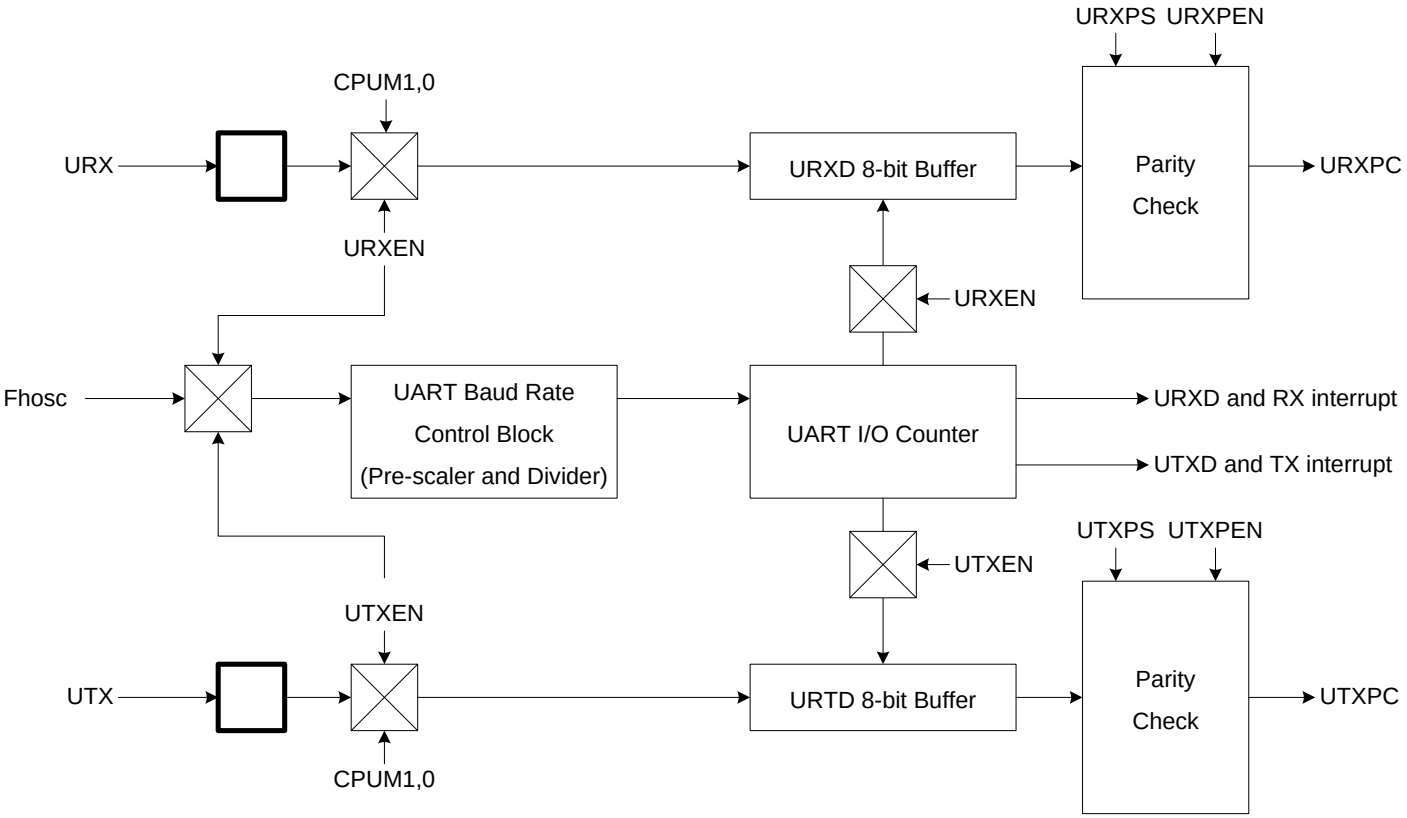
14 Universal Asynchronous Receiver/Transmitter (UART)

14.1 OVERVIEW

The UART interface is an universal asynchronous receiver/transmitter method. The serial interface is applied to low speed data transfer and communicate with low speed peripheral devices. The UART transceiver of Sonix 8-bit MCU allows RS232 standard and supports one byte data length. The transfer format has start bit, 8-bit data, parity bit and stop bit. Programmable baud rate supports different speed peripheral devices. UART I/O pins support push-pull and open-drain structures controlled by register.

The UART features include the following:

- **Full-duplex, 2-wire asynchronous data transfer.**
- **Programmable baud rate.**
- **8-bit data length.**
- **Odd and even parity bit.**
- **End-of-Transfer interrupt.**
- **Support break pocket function.**
- **Support wide range baud rate.**



UART Interface Structure Diagram

14.2 UART OPERATION

The UART RX and TX pins are shared with GPIO. When UART enables (URXEN=1, UTXEN=1), the UART shared pins transfers to UART purpose and disable GPIO function automatically. When UART disables, the UART pins returns to GPIO last status. The UART data buffer length supports 1-byte.

The UART supports interrupt function. URXIEN/UTXIEN are UART transfer interrupt function control bit. URXIEN=0, disable UART receiver interrupt function. UTXIEN=0, disable UART transmitter interrupt function. URXIEN=1, enable UART receiver interrupt function. UTXIEN=1, enable UART transmitter interrupt function. When UART interrupt function enable, the program counter points to interrupt vector (ORG 0013H/0014H) to do UART interrupt service routine after UART operating. URXIRQ/UTXIRQ is UART interrupt request flag, and also to be the UART operating status indicator when URXIEN=0 or UTXIEN=0, but cleared by program. When UART operation finished, the URXIRQ/UTXIRQ would be set to “1”.

The UART also builds in “Busy Bit” to indicate UART bus status. URXBZ bit is UART RX operation indicator. UTXBZ bit is UART TX operation indicator. If bus is transmitting, the busy bit is “1” status. If bus is finishing operation or in idle status, the busy bit is “0” status.

UART TX operation is controlled by loading UTXD data buffer. After UART TX configuration, load transmitted data into UTXD 8-bit buffer, and then UART starts to transmit the pocket following UART TX configuration.

UART RX operation is controlled by receiving the start bit from master terminal. After UART RX configuration, URX pin detects the falling edge of start bit, and then UART starts to receive the pocket from master terminal.

UART provides URXPC bit and UFMER bit to check received pocket. URXPC bit is received parity bit checker. If received parity is error, URXPC sets as “1”. If URXPC bit is zero after receiving pocket, the parity is correct. UFMER bit is received stream frame checker. The stream frame error definition includes “Start bit error”, “Stop bit error”, “Stream length error”, “UART baud rate error”... Each of frame error conditions makes UFMER bit sets as “1” after receiving pocket.

14.3 UART BAUD RATE

UART clock is 2-stage structure including a pre-scaler and an 8-bit buffer. UART clock source is generated from system oscillator called Fhosc. Fhosc passes through UART pre-scaler to get UART main clock called Fuart. UART pre-scaler has 8 selections (Fhosc/1, Fhosc/2, Fhosc/4, Fhosc/8, Fhosc/16, Fhosc/32, Fhosc/64, Fhosc/128) and 3-bit control bits (URS[2:0]). UART main clock (Fuart) purposes are the front-end clock and through UART 8-bit buffer (URCR) to obtain UART processing clock and decide UART baud rate.

UART Pre-scaler Selection, URS[2:0]	UART Main Clock Rate	Fuart (Fhosc=16MHz)
000b	Fhosc/1	16MHz
001b	Fhosc/2	8MHz
010b	Fhosc/4	4MHz
011b	Fhosc/8	2MHz
100b	Fhosc/16	1MHz
101b	Fhosc/32	0.5MHz
110b	Fhosc/64	0.25MHz
111b	Fhosc/128	0.125MHz

0F2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URCR	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0
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 UART baud rate clock source is Fhosc and divided by pre-scalar. The equation is as following.

UART Baud Rate = 1/2 *(Fuart * 1/(256 - URCR))…bps

Fhosc = 16MHz

Baud Rate	UART Pre-scaler	URS[2:0]	URCR (Hex)	UART Baud Rate	Accuracy (%)
1200	Fhosc/32	101b	30	1201	0.08%
2400	Fhosc/32	101b	98	2403	0.12%
4800	Fhosc/32	101b	CC	4807	0.14%
9600	Fhosc/32	101b	E6	9615	0.15%
19200	Fhosc/32	101b	F3	19230	0.15%
38400	Fhosc/2	000b	98	38461	0.15%
51200	Fhosc/2	001b	B2	51282	0.16%
57600	Fhosc/2	001b	BB	57971	0.64%
102400	Fhosc/2	001b	D9	102564	0.16%
115200	Fhosc/2	001b	DD	114285	-0.79%

☞ ***Note:***
1. ***We strongly recommend not to set URCR = 0xFF, or UART operation would be error.***
2. ***Fhosc=16MHZ,UART Pre-scaler not to set Fhosc/1,or UART operation would be error.***

14.4 UART transfer format

The UART transfer format includes “Bus idle status”, “Start bit”, “8-bit Data”, “Parity bit” and “Stop bit” as following.

UART Transfer Format with Parity Bit

UART Transfer Format without Parity Bit

Bus Idle Status: The bus idle status is the bus non-operating status. The UART receiver bus idle status of MCU is floating status and tied high by the transmitter device terminal. The UART transmitter bus idle status of MCU is high status. The UART bus will be set when URXEN and UTXEN are enabled.

Start Bit: UART is a asynchronous type of communication and need a attention bit to offer receiver the transfer starting. The start bit is a simple format which is high to low edge change and the duration is one bit period. The start bit is easily recognized by the receiver.

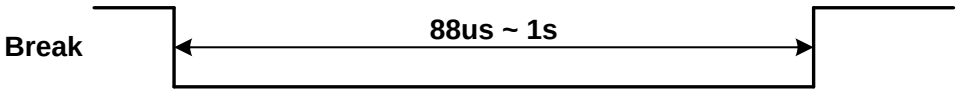
8-bit Data: The data format is 8-bit length, and LSB transfers first following start bit. The one bit data duration is the unit of UART baud rate controlled by register.

Parity Bit: The parity bit purpose is to detect data error condition. It is an extra bit following the data stream. The parity bit includes odd and even check methods controlled by URXPS/UTXPS bits. After receiving data and parity bit, the parity check executes automatically. The URXPC bit indicates the parity check result. The parity bit function is controlled by URXPEN/UTXPEN bits. If the parity bit function is disabled, the UART transfer contents remove the parity bit and the stop bit follows the data stream directly.

Stop Bit: The stop bit is like start bit using a simple format to indicate the end of UART transfer. The stop bit format is low to high edge change and the duration is one bit period.

14.5 BREAK POCKET

The break pocket is an empty stream to reset UART bus. Break pocket is like a long time zero pocket, and the period is 88us~1s.



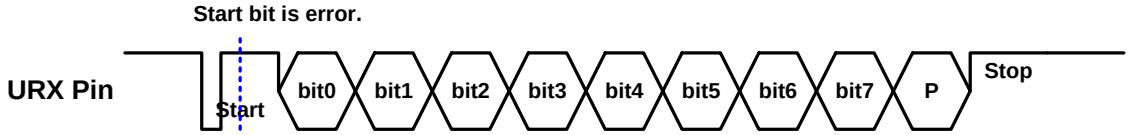
TX Break Pocket: UART builds in a UTXBRK bit to transmit Break pocket. When UTXEN = 1 (enable UART TX function), set UTXBRK bit to transmit Break pocket. When Break pocket finishes transmitting, UTXIRQ is set as “1”, and UTXBRK is cleared automatically. The period of transmitted break pocket is 25 UART baud rate clocks. If UART baud rate is 250000bps, the break pocket period is 100us.

UART TX Break Pocket Period = 25/UART Baud Rate…sec

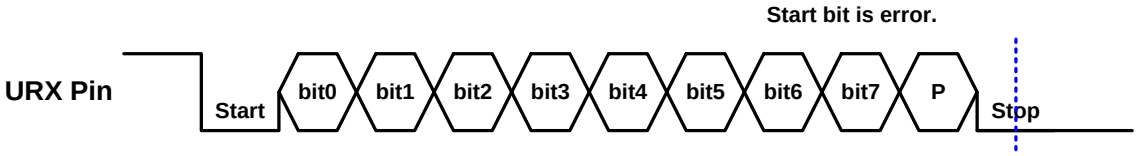
RX Break Pocket:
UART receives break pocket will get a frame error signal because the data period is longer than typical UART duration. UART can’t receive a complete data pocket. After receiving a UART pocket, the break pocket is still output low. UART issues frame error flag (UFMER = 1) and URXIRQ. Maybe the parity bit is error in parity mode. UART changes to initial status until detecting next start bit.

14.6 ABNORMAL POCKET

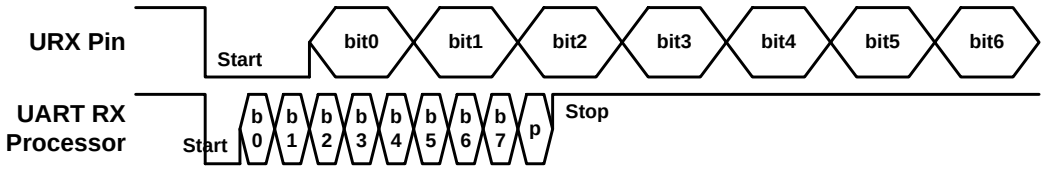
The abnormal pocket occurs in UART RX mode. Break pocket is one abnormal pocket of the UART architecture. The abnormal pocket includes Stream period error, start bit error, stop bit error…When UART receives abnormal pocket, the UFMER bit will be set “1”, and UART issues URXIRQ. The system finds the abnormal pocket through firmware. UART changes to initial status until detecting next start bit.



UART check the start bit is error and issue UFMER flag, but the UART still finishes receiving the pocket.



UART check the stop bit is error and issue UFMER flag, but the UART still finishes receiving the pocket.



If the host’s UART baud rate isn’t match to receiver terminal, the received pocket is error. But it is not easy to differentiate the pocket is correct or not, because the received error pocket maybe match UART rule, but the data is error. Use checking UFMER bit and URXPC bit status to decide the stream. If the two conditions seem like correct, but the pocket is abnormal, UART will accept the pocket as correct one.

14.7 UART RECEIVER CONTROL REGISTER

0F0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRX	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **URXEN**: UART RX control bit.
0 = Disable UART RX. URX pin is GPIO mode or returns to GPIO status.
1 = Enable UART RX. URX pin exchanges from GPIO mode to UART RX mode.

Bit 6 **URXPEN**: UART RX parity bit control bit.
0 = Disable UART RX parity bit function. The data stream doesn’t include parity bit.
1 = Enable UART RX parity bit function. The data stream includes parity bit.

Bit 5 **UTXPS**: UART RX parity bit format control bit.
0 = UART RX parity bit format is even parity.
1 = UART RX parity bit format is odd parity.

Bit 4 **URXPC**: UART RX parity bit checking flag.
0 = Parity bit is correct or no parity function.
1 = Parity bit is error.

Bit 3 **UFMER**: UART RX stream frame error flag bit.
0 = Collect UART frame.
1 = UART frame is error including start/stop bit, stream length.

Bit [2:0] **URS[2:0]**: UART per-scalar select bit.
000 = Fhosc/1, 001 = Fhosc/2, 010 = Fhosc/4, 011 = Fhosc/8, 100 = Fhosc/16, 101 = Fhosc/32, 110 = Fhosc/64, 111 = Fhosc/128.

14.8 UART TRANSMITTER CONTROL REGISTER

0F1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTX	UTXEN	UTXPEN	UTXPS	UTXBRK	URXBZ	UTXBZ	-	-
Read/Write	R/W	R/W	R/W	R/W	R	R	-	-
After reset	0	0	0	0	0	0	-	-

Bit 7 **UTXEN**: UART TX control bit.
0 = Disable UART TX. UTX pin is GPIO mode or returns to GPIO status.
1 = Enable UART TX. UTX pin exchanges from GPIO mode to UART TX mode and idle high status.
Bit 6 **UTXPEN**: UART TX parity bit control bit.
0 = Disable UART TX parity bit function. The data stream doesn't include parity bit.
1 = Enable UART TX parity bit function. The data stream includes parity bit.
Bit 5 **UTXPS**: UART TX parity bit format control bit.
0 = UART TX parity bit format is even parity.
1 = UART TX parity bit format is odd parity.
Bit 4 **UTXBRK**: UART TX BREAK pocket control bit.
0 = End of transmitting UART BREAK pocket.
1 = Start to transmit UART BREAK pocket.
Bit 3 **URXBZ**: UART RX operating status flag.
0 = UART RX is idle or the end of processing.
1 = UART RX is busy and processing.
Bit 2 **UTXBZ**: UART TX operating status flag.
0 = UART TX is idle or the end of processing.
1 = UART TX is busy and processing.

☞ ***Note: URXBZ and UTXBZ bits are UART operating indicators. After setting UART RX/TX operations, set a "NOP" instruction is necessary, and then check UART status through URXBZ and UTXBZ bits.***

14.9 UART DATA BUFFER

0F3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UTXD	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **UTXD:** UART transmitted data buffer.

0F4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URXD	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **URXD:** UART received data buffer.

14.10 UART OPERATION EXAMLPE

- **UART TX Configuration:**

```
; Select parity bit function.
        B0BCLR      FUTXPEN         ; Disable UART TX parity bit function.
;or
        B0BSET      FUTXPEN         ; Enable UART TX parity bit function.

; Select parity bit format.
        B0BCLR      FUTXPS          ; UART TX parity bit format is even parity.
;or
        B0BSET      FUTXPS          ; UART TX parity bit format is odd parity.

; Set UART baud rate.
        MOV         A, #value1      ; Set UART pre-scaler URS[2:0].
        B0MOV       URRX, A
        MOV         A, #value2      ; Set UART baud rate 8-bit buffer.
        B0MOV       URCR, A

; Enable UART TX pin.
        B0BSET      FUTXEN          ; Enable UART TX function and UART TX pin.

; Enable UART TX interrupt function.
        B0BCLR      FUTXIRQ         ; Clear UART TX interrupt flag.
        B0BSET      FUTXIEN         ; Enable UART TX interrupt function.

; Load TX data buffer and execute TX transmitter.
        MOV         A, #value3      ; Load 8-bit data to UTXD data buffer.
        B0MOV       UTXD, A
                                    ;After loading UTXD, UART TX starts to transmit.
        NOP                         ; One instruction delay for UTXBZ flag.

; Check TX operation.
        B0BTS0      FUTXBZ          ; Check UTXBZ bit.
        JMP         CHKTX           ; UTXBZ=1, TX is operating.
        JMP         ENDTX           ; UTXBZ=0, the end of TX.
```

☞ ***Note: UART TX operation is started through loading UTXD data buffer.***

- **Transmit Break Pocket:**

```
; Select parity bit function.
        B0BCLR      FUTXPEN     ; Disable UART TX parity bit function.
;or
        B0BSET      FUTXPEN     ; Enable UART TX parity bit function.

; Select parity bit format.
        B0BCLR      FUTXPS      ; UART TX parity bit format is even parity.
;or
        B0BSET      FUTXPS      ; UART TX parity bit format is odd parity.

; Set UART baud rate.
        MOV         A, #value1  ; Set UART pre-scaler URS[2:0].
        B0MOV       URRX, A
        MOV         A, #value2  ; Set UART baud rate 8-bit buffer.
        B0MOV       URCR, A

; Enable UART TX pin.
        B0BSET      FUTXEN      ; Enable UART TX function and UART TX pin.

; Enable UART TX interrupt function.
        B0BCLR      FUTXIRQ     ; Clear UART TX interrupt flag.
        B0BSET      FUTXIEN     ; Enable UART TX interrupt function.

; Start UART break pocket.
        B0BSET      FUTXBRK     ; Transmit UART break pocket.
        NOP                     ; One instruction delay for UTXBZ flag.

; Check TX operation.
        B0BTS0      FUTXBZ      ; Check UTXBZ bit.
        JMP         CHKTX       ; UTXBZ=1, TX is operating.
        JMP         ENDTX       ; UTXBZ=0, the end of TX.
```

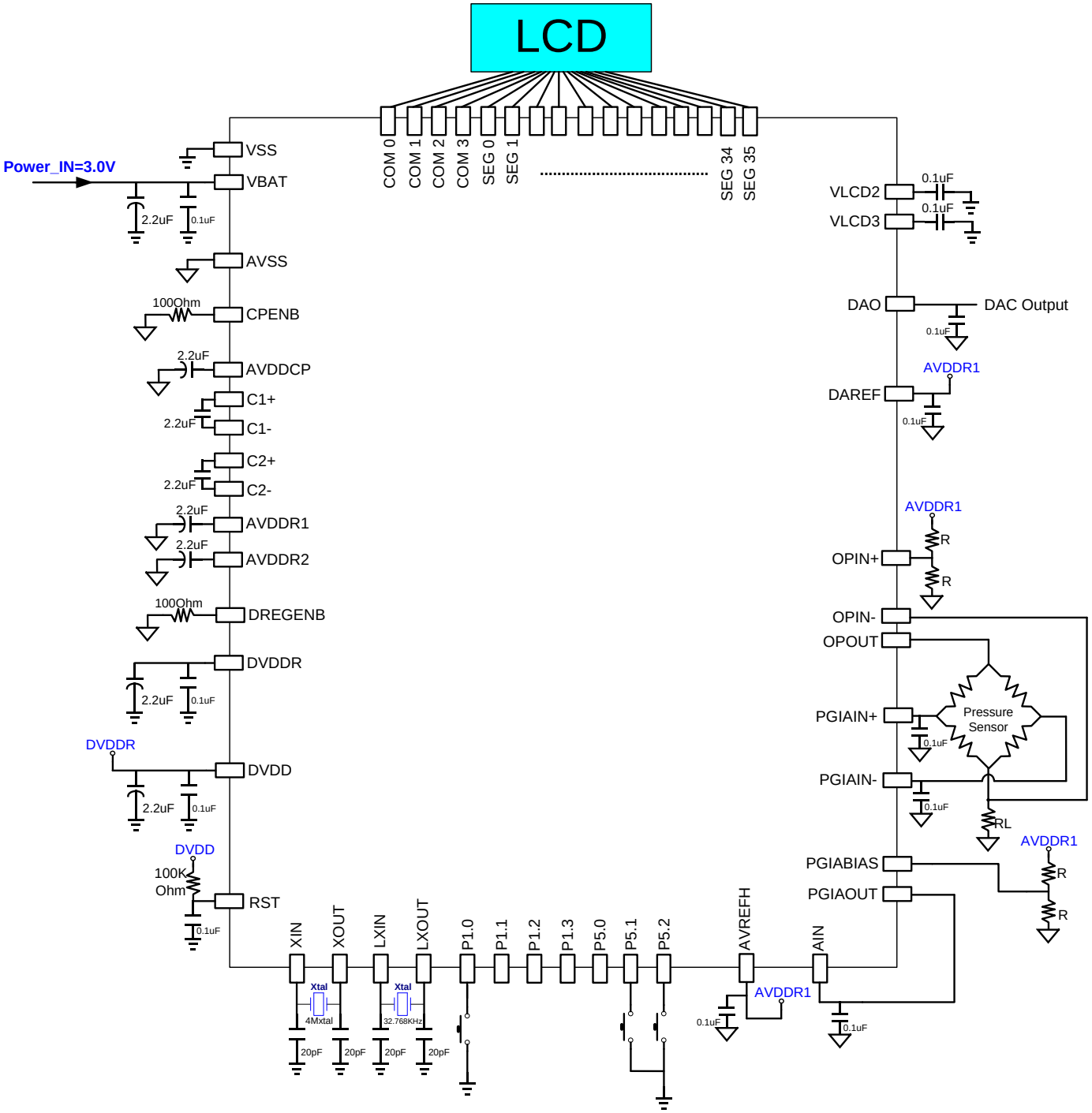
☞ ***Note: UART TX break pocket is controlled by UTXBRK bit and needn't load UTXD buffer.***

- **UART RX Configuration:**

```
; Select parity bit function.
        B0BCLR      FURXPEN     ; Disable UART RX parity bit function.
;or
        B0BSET      FURXPEN     ; Enable UART RX parity bit function.

; Select parity bit format.
        B0BCLR      FURXPS      ; UART RX parity bit format is even parity.
;or
        B0BSET      FURXPS      ; UART RX parity bit format is odd parity.

; Set UART baud rate.
        MOV         A, #value1  ; Set UART pre-scaler URS[2:0].
        B0MOV       URRX, A
        MOV         A, #value2  ; Set UART baud rate 8-bit buffer.
        B0MOV       URCR, A

; Enable UART RX pin.
        B0BSET      FURXEN      ; Enable UART RX function and UART RX pin.

; Enable UART RX interrupt function.
        B0BCLR      FURXIRQ     ; Clear UART RX interrupt flag.
        B0BSET      FURXIEN     ; Enable UART RX interrupt function.
        NOP                     ; One instruction delay for URXBZ flag.


; Check RX operation.
        B0BTS0      FURXBZ      ; Check URXBZ bit.
        JMP         CHKRX       ; URXBZ=1, RX is operating.
        JMP         ENDRX       ; URXBZ=0, the end of RX.
```

> ☞ ***Note: UART RX operation is started as start bit transmitted from master terminal.***

15 APPLICATION CIRCUIT



16 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 (bnak 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I, (M = only for Working registers R, Y, Z , RBANK & PFLAG)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
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
	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
	B0ADD	M,A	M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1
	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
	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
	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
	DAA		To adjust ACC's data format from HEX to DEC.	√	-	-	1
	MUL	A,M	R, A ← A * M, The LB of product stored in Acc and HB stored in R register. ZF affected by Acc.	-	-	√	2
LOGIC	AND	A,M	A ← A and M	-	-	√	1
	AND	M,A	M ← A and M	-	-	√	1
	AND	A,I	A ← A and I	-	-	√	1
	OR	A,M	A ← A or M	-	-	√	1
	OR	M,A	M ← A or M	-	-	√	1
	OR	A,I	A ← A or I	-	-	√	1
	XOR	A,M	A ← A xor M	-	-	√	1
	XOR	M,A	M ← A xor M	-	-	√	1
	XOR	A,I	A ← A xor I	-	-	√	1
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
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1
	CLR	M	M ← 0	-	-	-	1
	BCLR	M.b	M.b ← 0	-	-	-	1
	BSET	M.b	M.b ← 1	-	-	-	1
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1
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 + S
	INC	M	A ← M + 1.	-	-	√	1
	INCM	M	M ← M + 1.	-	-	√	1+N
	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 + S
	DEC	M	A ← M – 1.	-	-	√	1
	DECM	M	M ← M – 1.	-	-	√	1+N
	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
	CALLHL		Stack ← PC15~PC0, PC15~PC8 ← H register, PC7~PC0 ← L register	-	-	-	2

	CALLYZ	Stack ← PC15~PC0, PC15~PC8 ← Y register, PC7~PC0 ← Z register	-	-	-	2
M	RET	PC ← Stack	-	-	-	2
I	RETI	PC ← Stack, and to enable global interrupt	-	-	-	2
S C	NOP	No operation	-	-	-	1

***Note:** 1. Processing OSCM register needs to add extra one cycle.*
2. If branch condition is true then "S = 1", otherwise "S = 0".

17 ELECTRICAL CHARACTERISTIC

17.1 ABSOLUTE MAXIMUM RATING

(All of the voltages referenced to Vss)

Supply voltage (Vdd)..... 2.0V ~ 6.0V
Input in voltage (Vin).....Vss - 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr) ..... 0°C ~ + 70°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C
Power consumption (Pc).....500 mW

17.2 STANDARD ELECTRICAL CHARACTERISTIC

(All of voltages referenced to Vss, Vdd = 5.0V, fosc = 4 MHz, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Operating voltage	DVDD	Normal mode, Vpp = DVDD	2.4	5.0	5.5	V
		Programming mode, Vpp = 12.5V	4.5	5.0	5.5	
	VBAT	Battery(charge pump) input voltage	2.0	5.0	6.5	
RAM Data Retention voltage	Vdr		1.4	-	-	V
Internal POR	Vpor	Vdd rise rate to ensure internal power-on reset	-	0.05	-	V/ms
Input Low Voltage	ViL	Input with Schmitt trigger	Vss	-	0.2Vdd	V
Input High Voltage	ViH	Input with Schmitt trigger	0.8Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd	-	-	5	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 5V	50	100	150	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd	-	-	2	uA
Port1, Port2, Port4, Port 5 output source current	IoH	Vop = Vdd - 0.5V	10	-	-	mA
sink current	IoL	Vop = Vss + 0.5V	-	-	-9	mA
INTn trigger pulse width	Tint0	INT0 ~ INT2 interrupt request pulse width	2/fcpu	-	-	Cycle
		ADC				
AVREFH input voltage	Varfh	Vdd = 5.0V	2V	-	Vdd	V
AIN0 ~ AIN3 input voltage	Vani	Vdd = 5.0V	0	-	AVREFH	V
ADC clock Frequency	Fadclk	Vdd = 5.0V	32K	-	8M	Hz
		Vdd = 3.0V	32K	-	8M	Hz
ADC conversion cycle time	Fadcyl	Vdd = 2.4V~5.5V	64	-	-	1/Fadclk
ADC sampling Rate (set FADS=1 frequency)	Fadsmp	Vdd = 5.0V	-	-	125	K/sec
		Vdd = 3.0V	-	-	80	K/sec
Differential Nonlinearity	DNL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	±1	±2	±16	LSB
Integral Nonlinearity	INL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	±2	±4	±16	LSB
No Missing Code	NMC	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	8	10	12	Bits
ADC enable time	Tast	Ready to start convert after set ADENB = “1”	100	-	-	uS
ADC current consumption	I_{ADC}	Vdd=5.0V	-	0.6*	-	mA
		Vdd=3.0V	-	0.4*	-	mA
		DAC				
Positive Supply Current	Idac	Zero scale. DAB[11:0]=000000000000b	-	10	-	uA
Relative Accuracy	INL				±2	LSB
Differential Nonlinearity	DNL	Monotonic			±2	LSB
Output Leakage Current	Idaleak	Zero scale.			10	nA
Analog Output Current	Idao	Full scale.		2		mA
Output Voltage Setting Time	ts			2		us
DAREF input voltage	Vdref				3.8	V

Supply Current	Idd1	Run Mode Fcpu = Fosc/4	VBAT= 5V,4MHz/IHRC CPENB=Vdd, DREGENB=Vss	-	1.5	4	mA
			VBAT= 3V,4MHz/IHRC CPENB=Vss, DREGENB=Vss	-	1	3	mA
			VBAT= 5V,32768Hz CPENB=Vdd, DREGENB=Vss	-	30	60	uA
			VBAT= 3V,32768Hz CPENB=Vss, DREGENB=Vss	-	70	150	uA
	Idd2	Green mode exclude “High clock = 32.768K”	Vdd= 5V, CPENB=Vdd, DREGENB=Vss	-	22	40	uA
			Vdd= 3V, CPENB=Vss, DREGENB=Vss	-	45	70	uA
	Idd3	Sleep mode	Vdd=5V, CPENB=Vdd, DREGENB=Vdd	-	-	2	uA
LVD Detect Voltage	Vdet	Low voltage detect level(LVD_L)		2.0	-	2.3	V
Regulator voltage	AVDDR2	VBAT=3V/5V,@1K loading		2.45	2.55	2.65	V
	AVDDR1	VBAT=3V/5V,@1K loading		3.75	3.85	3.95	V
	DVDDR	VBAT=3V/5V,@1K loading		4.3	4.5	4.7	V
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	*25℃,* Vdd= 5V, Fcpu = 1MHz	15.2	16	16.8	Mhz

18 OTP Programming Pin

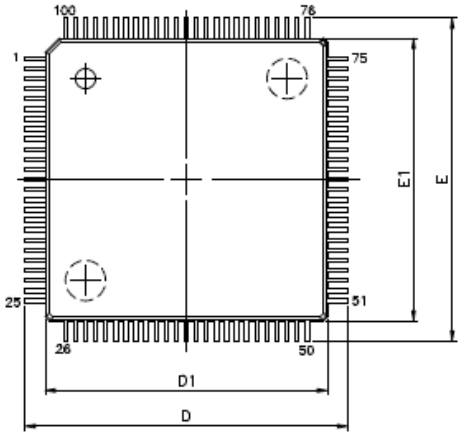
18.1 SN8P2839 Series Programming Pin Mapping:

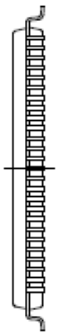
OTP Programming Pin of SN8P2839 Series			
Chip Name		SN8P2839	
Writer			
Number	Pin	Number	Pin
1	VDD	92/87/89/90/99/98	VBAT/DVDD/CPENB/DREGENB/AVDDR1/AVDDR2
2	GND	68/88/100	VSS/VSS/AVSS
3	CLK (PGCLK)	73	P50
4	CE		
5	PGM (OTPCLK)	81	P10
6	OE (SHIFDATA)	72	P51
7	D1		
8	D0		
9	D3		
10	D2		
11	D5		
12	D4		
13	D7		
14	D6		
15	VDD	92/87/89/90/99/98	VBAT/DVDD/CPENB/DREGENB/AVDDR1
16	VPP	82	RST
17	HLS		
18	RST		
19	-		
20	ALSB/PDB	80	P11

- **Disable Analog Regulator when OTP program.**

19 PACKAGE INFORMATION

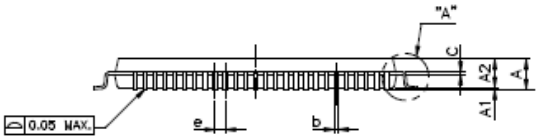
19.1 LQFP100

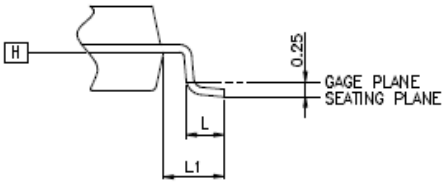




VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	--	--	1.60
A1	0.05	--	0.15
A2	1.35	1.40	1.45
b	0.17	0.20	0.27
c	0.09	0.127	0.20
D	16.00 BSC		
D1	14.00 BSC		
E	16.00 BSC		
E1	14.00 BSC		
e	0.50 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		





DETAIL A

NOTES:
1.JEDEC OUTLINE:MS-026 BED

2.DATUM PLANE H IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY.

3.DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.

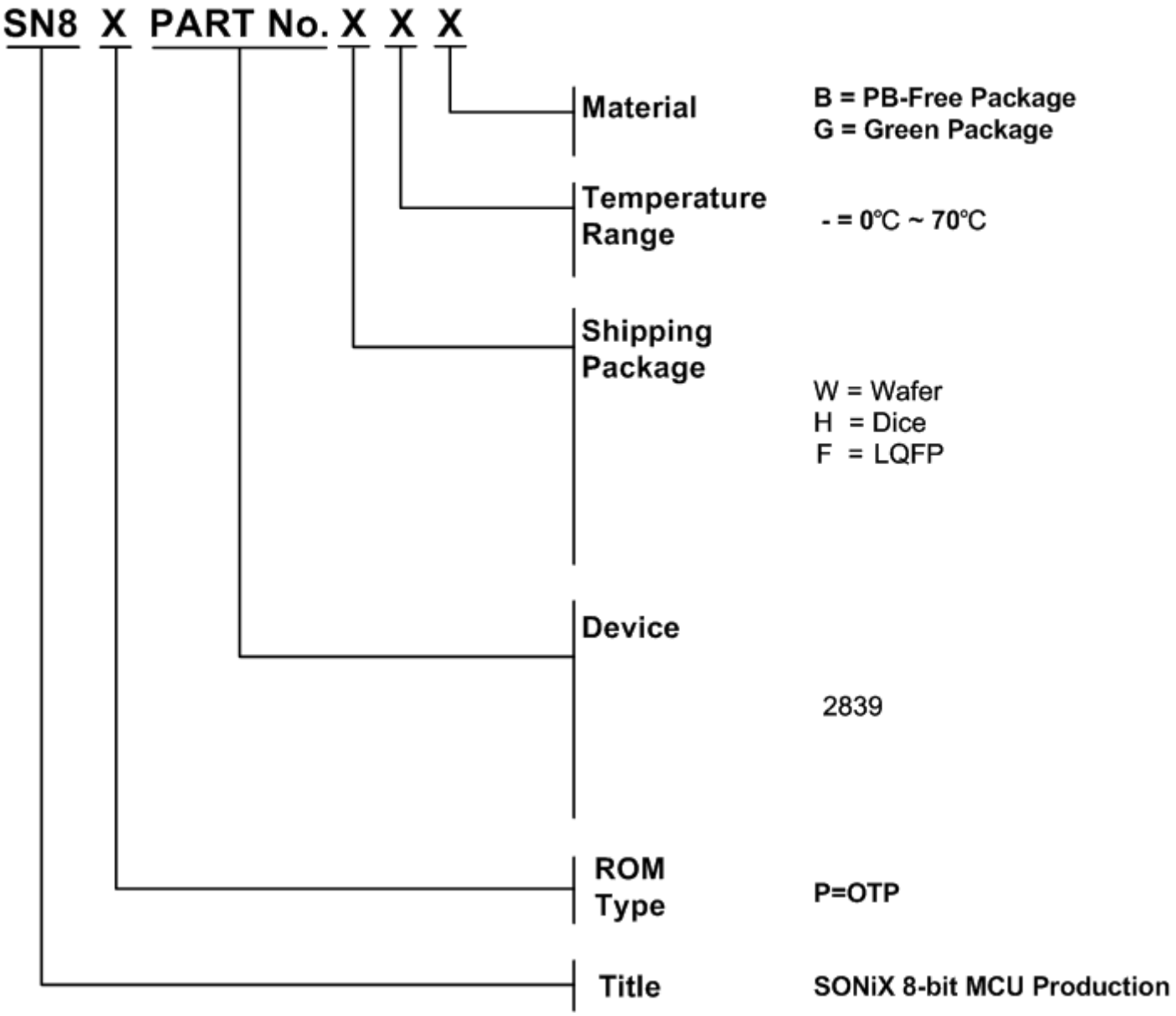
4.DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

20 Marking Definition

20.1 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.

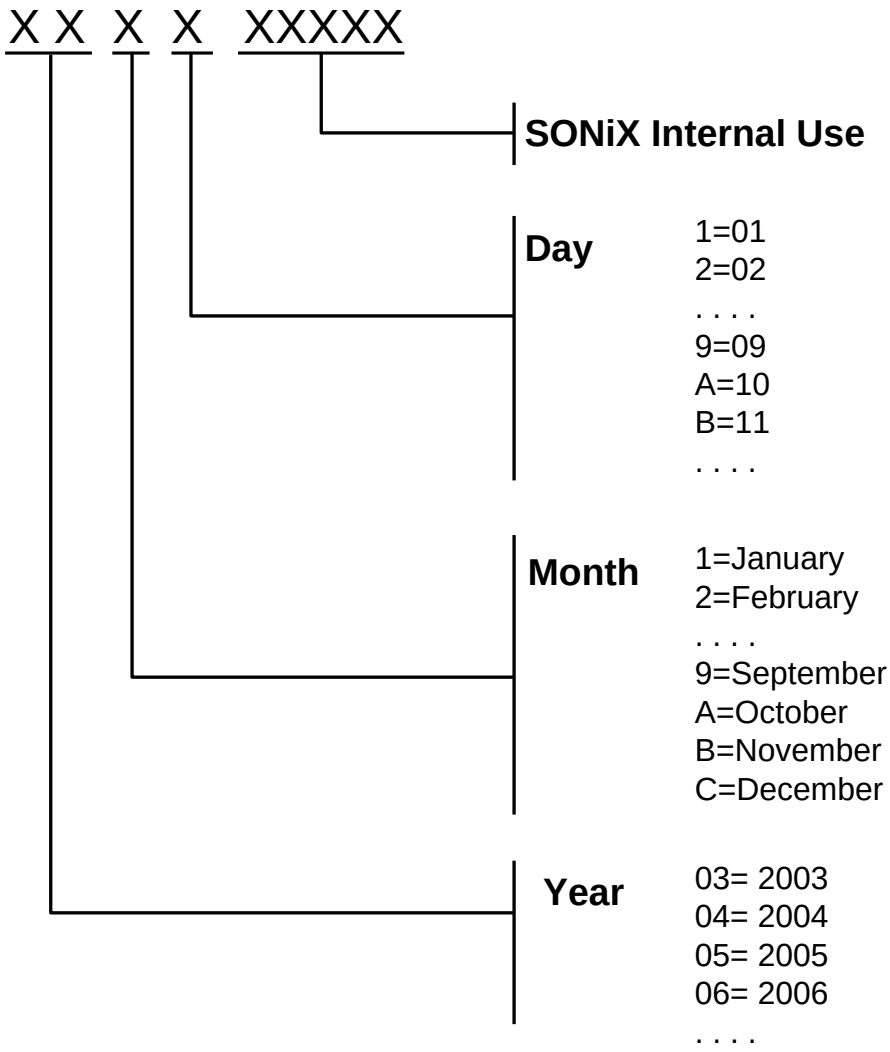
20.2 Marking indetification system



20.3 Marking Example

Name	ROM Type	Device	Package	Temperature	Material
SN8P2839FG	OTP	2839	LQFP	0℃~70℃	Green Package
SN8P2839FB	OTP	2839	LQFP	0℃~70℃	PB-Free Package

20.4 Datecode system



Main Office:
Address: 10F-1, No.36, Taiyuan Street, Chupei City, Hsinchu, Taiwan.
Tel: 886-3-5600 888
Fax: 886-3-5600 889
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