CMS79FT73x User Manual

Enhanced 8-bit CMOS Microcontroller with Flash Memory

Rev. 1.5.0

Manual

1. Product Description

1.1 Features

- memory
  - Flash: 8Kx16
  - Universal RAM: 256x8
  - Dedicated RAM for built-in touch feature: 400x8
- 8 level stack buffer
- Clean instructions (68 instructions)
- look-up table
- built-in WDT timer
- built-in low voltage detection circuit
- Interrupt source
  - 3 timer interrupt
  - Interrupt for change in electrical level RB port
  - Other peripherals interrupt
- timer
  - 8-bit timer: TIMER0, TIMER2
  - 16-bit timer: TIMER1 which supports external
  - 32.768kHz crystal oscillator as timer clock source
- Capture, compare and PWM mod (CCP)
  - 10-bit PWM precision
  - 2 PWM circuit with configurable and adjustable period and duty cycle
  - Able to configurate between RB4/RB3 or RC6/RC7
- built-in 128-byte EEPROM
  - Can re-write/erase up to 100000 times
- built-in touch button detection mod, no need for external touch capacitor
  - Can pass dynamic/static 10V EMC test
  - Adjustable touch sensitivity
  - All pins can be configurated to touch channel
- Working voltage: 2.6V~5.5V@16MHz
  2.0V~5.5V@8MHz
- Working temperature: -40˚C~85˚C
- Multiple ways of oscillation
  - Internal RC: design frequency of 8MHz/16MHz
  - External XT: up to 16MHz
- Instructions period (single instruction or double instructions)
- built-in LED driver mod
  - Support up to 11 segments, 8 bits
  - Large driving current up to 150mA at COM port
  - Adjustable current of 2-30mA at SEG port
- built-in LCD1/2 Bias COM driver mod
  - Support up to 8 COM port
  - COM port driver current selection
- built-in MSSP communication mod (SPI/ I^2C)
  - I^2C supports master control/slave mode (7-bit addressing)
  - I^2C slave mode supports broadcast call
  - SPI supports master/slave mode
- built-in 2 USART communication mod
  - Supports synchronous master/slave mode and asynchronous duplex mode
  - serial port 1 can be configurated between RB4/RB3 or RC6/RC7
- High precision 12-bit ADC
  - Built-in high precision 1.2V reference voltage

    ±1.5% @VDD=2.5V~5.5V T_A=25˚C

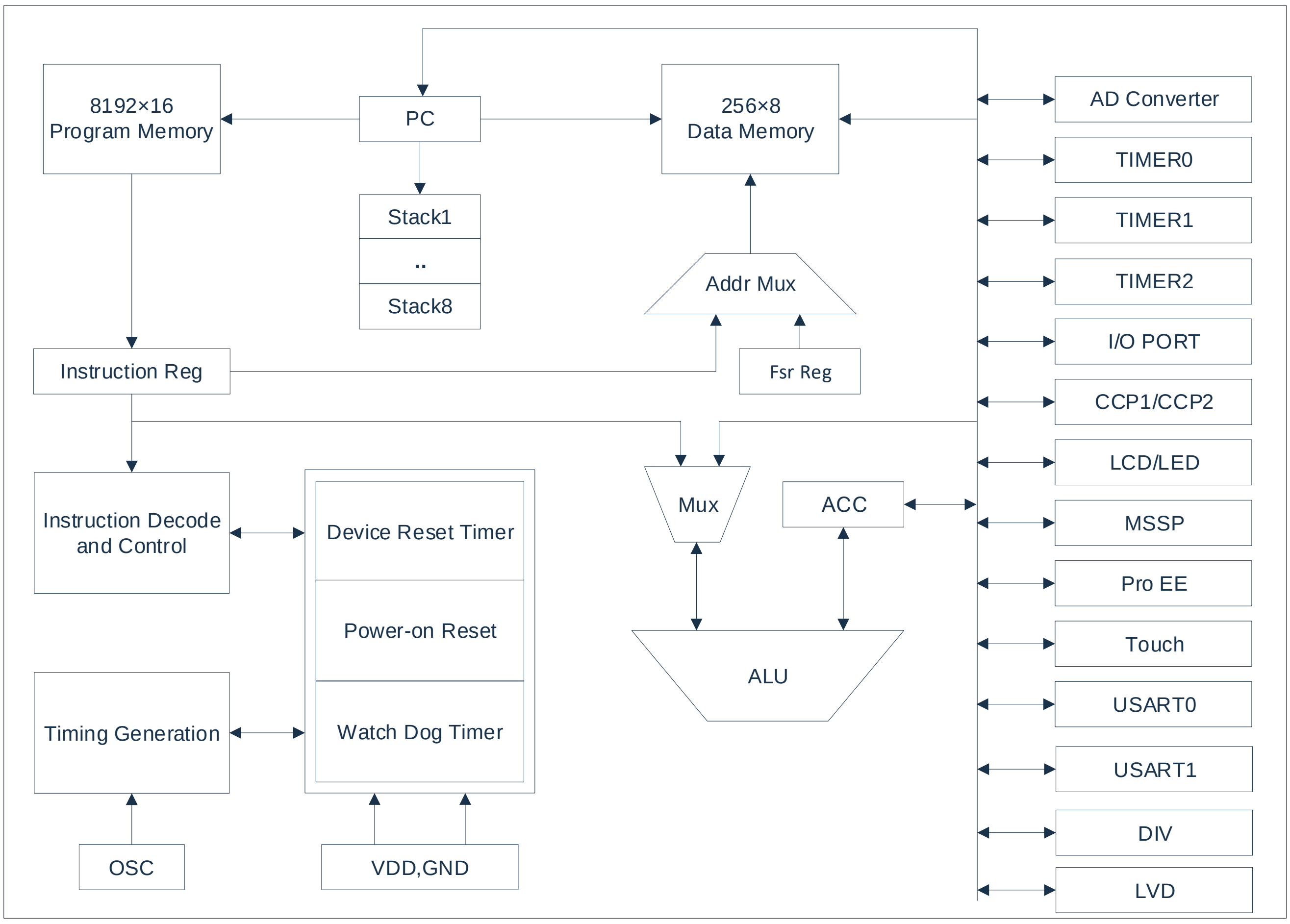
    ±2% @VDD=2.5V~5.5V T_A=-40˚C~85˚C
- built-in LVD mod
  - Choice of voltage:

    2.2V/2.4V/2.7V/3.0V/3.3V/3.7V/4.0V/4.3V

PRODUCT	ROM	RAM	Pro EE	I/O	LED	LCD	ADC	Touch	USART	PACKAGE
CMS79FT736	8Kx16	256x8	128x8	18	$8_{seg}x4_{com}$	$1/2Biasx4_{com}$	12Bitx18	16	2	SOP20
CMS79FT738	8Kx16	256x8	128x8	26	$11_{seg}X8_{com}$	$1/2Biasx8_{com}$	12Bitx26	16	2	SOP28

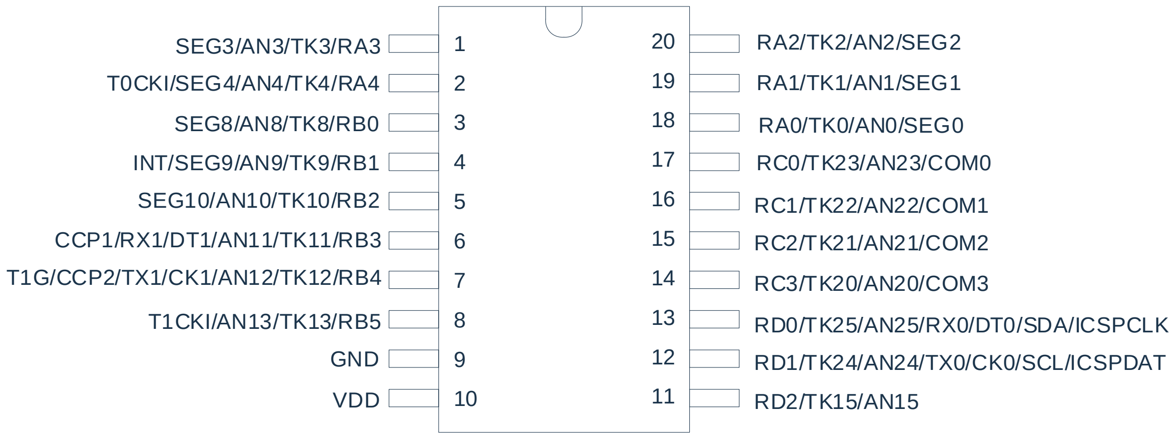
Note: ROM----program memory Pro EE---program EEPROM

1.2 System Structure Diagram



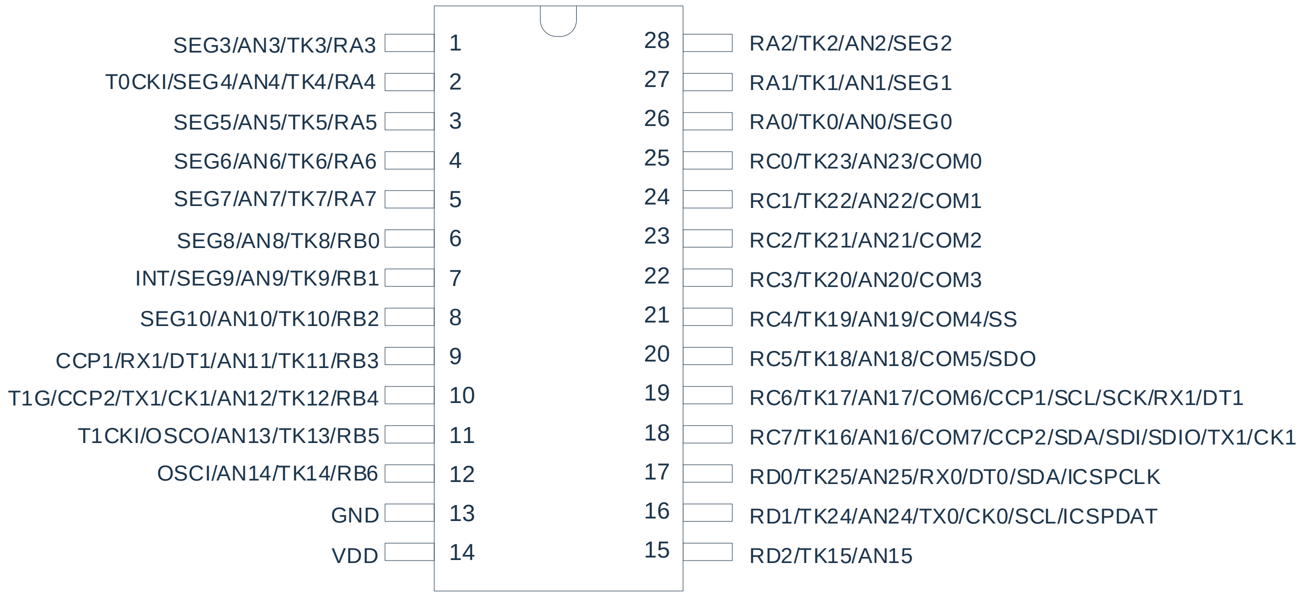
1.3 Pin Allocation

1.3.1 CMS79FT736



CMS79FT736

1.3.2 CMS79FT738



CMS79FT738

Note：
1) RB4/RB3 and RC6/RC7's serial port function is congurated by CONFIG
2) RB4/RB3 and RC6/RC7's CCP function is congurated by CONFIG

Pin description:

Pin name	IO type	description
VDD, GND	P	Voltage input pin and ground
OSCIN/OSCOUT	P	Oscillator in/out pin
RA0-RA7	I/O	Programmable in/ push-pull out pin, with pull-up resistance function
RB0-RB6	I/O	Programmable in/push-pull out pin， with pull-up resistance, pull-down resistance, electrical level interrupt function
RC0-RC7	I/O	Programmable in/ push-pull out pin, with pull-up resistance function
RD0-RD2	I/O	Programmable in/ push-pull out pin, with pull-up resistance function
ICSPCLK/ICSPDAT	I/O	Program clock/data pin
TK0-TK25	-	touch button input pin
AN0-AN23	I	12 位 ADC input pin
SEG0-SEG10	O	LED driversegment ouput
COM0-COM7	O	LED/LCD driver common port
T0CKI	I	TIMER0 external clock input pin
T1CKI	I	TIMER1 external clock input pin
T1G	I	TIMER1 gate control input pin
CCP1	I/O	capture/compare/PWM1
CCP2	I/O	capture/compare/PWM2
SCK	I/O	SPI clock input pin
SDI	I	SPI data input pin
SDO	O	SPI data output pin
SS	I	SPI slave choice input pin
SCL	I/O	I^2Cclock input/output pin
SDA	I/O	I^2Cdata input/output pin
TX0/CK0	I/O	USART0 asynchronous transmit output/synchronous clock input/output pin
RX0/DT0	I/O	USART0 asynchronous receive input/synchronous data input/output pin
TX1/CK1	I/O	USART1 asynchronous transmit output/synchronous clock input/output pin
RX1/DT1	I/O	USART1 asynchronous receive input/synchronous data input/output pin

1.4 System Configuration Register

System configuration register (CONFIG)is the initial FLASH choice of the MCU. It can only be burned by CMS burner. User cannot visit. It includes the following:

1. OSC (choice of oscillation)

◆ INTRC	Internal RC oscillation
◆ XT	External crystal oscillation

2. INTRC_SEL (internal oscillation frequency)

◆ INTRC8M	F_{HSI} choose internal 8MHz RC oscillation
◆ INTRC16M	F_{HSI} choose internal 16MHz RC oscillation

3. WDT (watchdog choice)

◆ ENABLE	Enable watchdog timer
◆ DISABLE	Disable watchdog timer

4. PROTECT (encyption)

◆ DISABLE	Disable FLASH code encyption
◆ ENABLE	Enable FLASH code encryption, after which the read value from burning the simulator is uncertain.

5. LVR_SEL (low voltage detection selection)
   - ◆ 2.0V
   - ◆ 2.6V

6. ICSPPORT_SEL (simulation port selection)

◆ ICSP	ICSPCLK, DAT port keep as similation port, all functions disabled
◆ NORMAL	ICSPCLK, DAT port as normal port

7. USART1_SEL (TX1/RX1) (USART1 port selection)

◆ RC7/RC6	Select RC7 as TX1, RC6 as RX1,
◆ RB4/RB3	Select RB4 as TX1, RB3 as RX1

8. CCP_SEL (CCP port selection)

◆ RC6/RC7	Select RC6 as CCP1, RC7 as CCP2
◆ RB3/RB4	Select RB3 as CCP1, RB4 as CCP2

9. IIC_SEL (IIC port selection)

◆ RD1/RD0	Select RD1 as SCL, RD0 as SDA
◆ RC6/RC7	Select RC6 as SCL, RC7 as SDA

1.5 Online Serial Programming

Can perform serial programming on MCU t the final application circuit. Programming is done through the following:

- Power wire
- Ground wire
- Data wire
- Clock wire

This ensures users to use un-programmed devices to make circuit and only program the MCU just before the product being delivered. Therefore, the lastest version of firmware can be burned into the MCU.

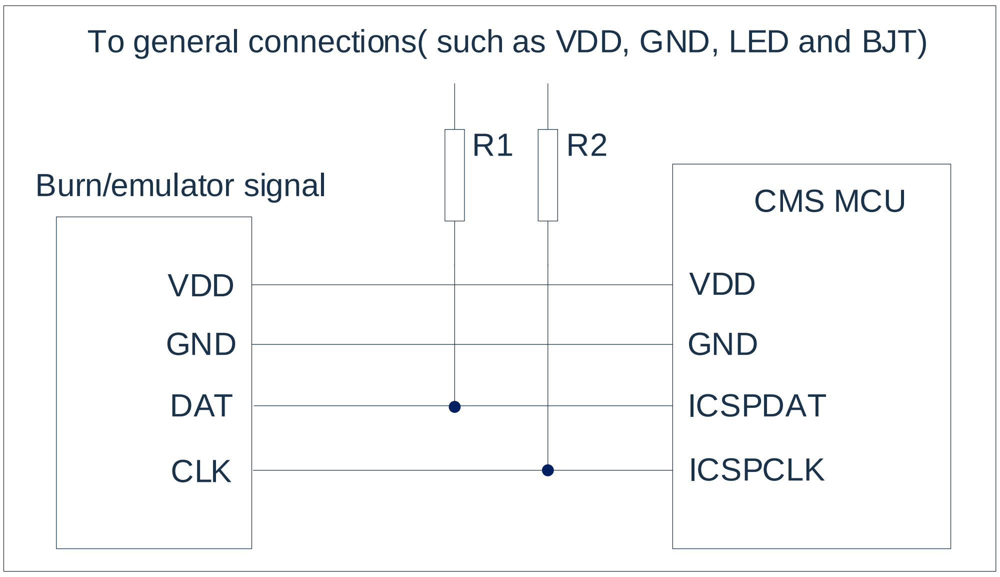


Fig 1-1： Typical connection for online serial programming

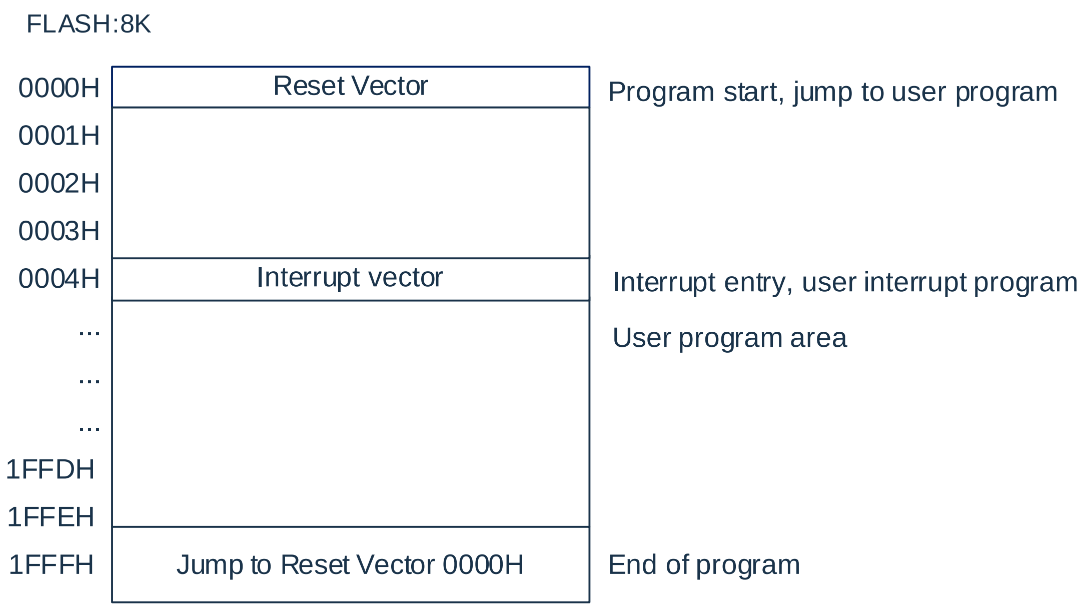
In the above figure, R1 and R2 are the electrical isolation devices, normally represented by resistor with the following resistance: R1≥4.7K, R2≥4.7K.

2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program Memory

CMS79FT73x program memory space



2.1.1.1 Reset Vector (0000H)

MCU has 1-byte long system reset vector (0000H). It has 3 ways to reset:

- Power-on reset
- Watchdog reset
- Low voltage reset (LVR)

When any above reset happens, program will start to execute from 0000H, system registerwill be recovered to default value. PD and TO from STATUS register can determine the which reset is performed from above. The following program illustrayes how to define the reset vector from FLASH.

Example: define reset vector

```
                ORG          0000H                 ; system reset vector
                JP           START
                ORG          0010H                 ; start of user program
        START:
                ...                                ; user program
                ...
                END                                ; program end
```

2.1.1.2 Interrupt Vector

The address for interrupt vector is 0004H. Once the interrupt responds, the current value for program counter PCwill be saved to stack buffer and jump to 0004H to execute interruptservice program. All interruptwill enter 0004H. User will determine which interrupt to execute according to the bit of register of interrupt flag bit. The following program illustrate how to write interrupt service program.

Example: define interrupt vector, interrupt program is placed after user program

```
            ORG         0000H               ; systemresetvector
            JP          START
            ORG         0004H               ; start of user program
INT_START:
            CALL        PUSH                ; save ACC and STATUS
            ...                             ; user interrupt program
            ...
INT_BACK:
            CALL        POP                 ; back to ACC and STATUS
            RETI                            ; interrupt back
START:
            ...                             ; user program
            ...
            END                             ; program end
```

> Note: MCU does not provide specific unstack and push instructions, so user needs to protect interrupt scene.

Example: interrupt-in protection

```
PUSH:
            LD          ACC_BAK,A           ; save ACC to ACC_BAK
            SWAPA       STATUS              ; swap half-byte of STATUS
            LD          STATUS_BAK,A        ; save to STATUS_BAK
            RET                             ; back
```

Example: interrupt-out restore

```
POP:
            SWAPA       STATUS_BAK          ; swap the half-byte data from STATUS_BAK to ACC
            LD          STATUS,A            ; pass the value in ACC to STATUS
            SWAPR       ACC_BAK             ; swap the half-byte data in ACC_BAK
            SWAPA       ACC_BAK             ; swap the half-byte data from ACC_BAK to ACC
            RET                             ; back
```

2.1.1.3 Look-up Table

Any address in FLASH can be use as look-up table.

Related instructions:

● TABLE [R]	Pass the lower byte in table to register R, pass higher byte to TABLE_DATAH.
● TABLEA	Pass the lower byte in table to ACC, pass higher byte to TABLE_DATAH.

related register:

● TABLE_SPH (110H)	Read/write register to indicate higher 5 bits in the table.
● TABLE_SPL (111H)	Read/write register to indicate lower 8 bits in the table.
● TABLE_DATAH (112H)	Read only register to save higer bit information in the table

Note：Write the table address into TABLE_SPH and TABLE_SP before using look-up. If main program and interrupt service programboth use look-up tablein structions, the value for TABLE_SPH in the main program may change due to the look-up instructions from interrupt and hence cause error. Avoid using look-up table instruction in both main program and interrupt service. Dsiable the interrupt before using the look-up table instruction and enable interrupt after the look-up instructions are done.

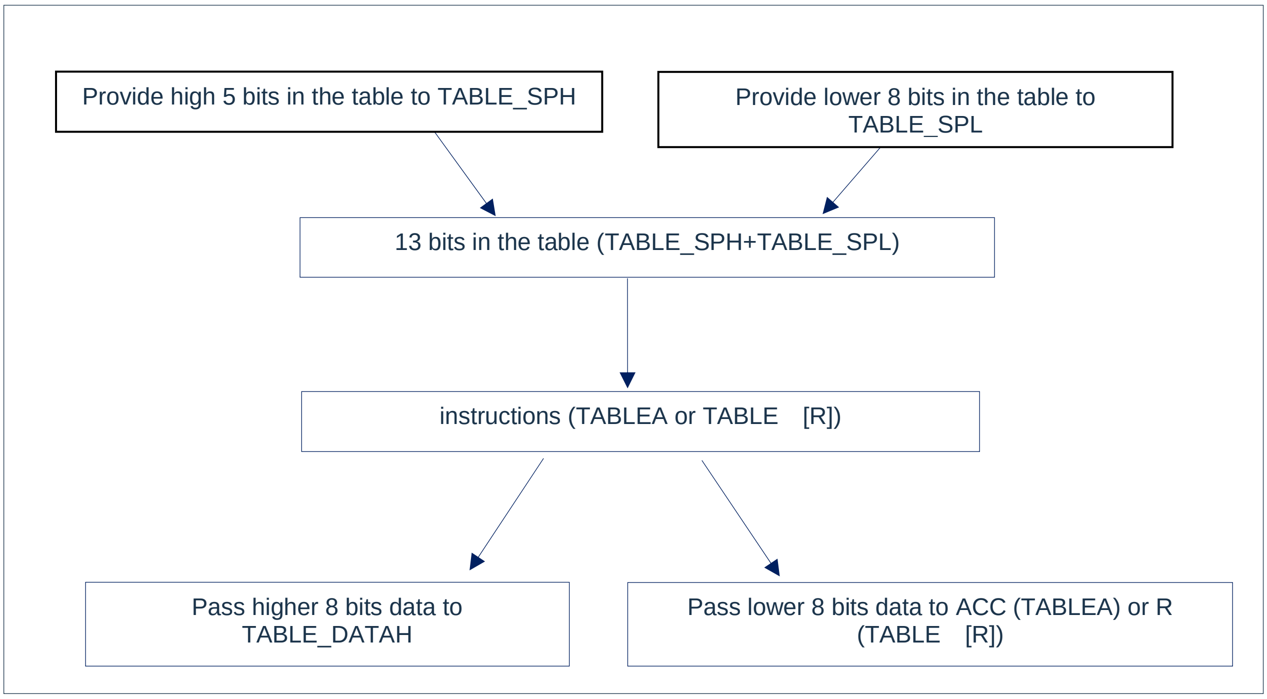

Provide high 5 bits in the table to TABLE_SPH
Provide lower 8 bits in the table to TABLE_SPL
13 bits in the table (TABLE_SPH+TABLE_SPL)
instructions (TABLEA or TABLE [R])
Pass higher 8 bits data to TABLE_DATAH
Pass lower 8 bits data to ACC (TABLEA) or R (TABLE [R])

Fig 2-1: Flow chart for table usage

The following illustrates how to use the table in the program.

```
...                                     ;continue from user program
LDIA        02H                         ;lower bits address in the table
LD          TABLE_SPL,A
LDIA        06H                         ; higher bits address in the table
LD          TABLE_SPH,A
TABLE       R01                         ;table instructions, pass the lower 8 bits (56H) to
                                        R01
LD          A,TABLE_DATAH               ;pass the higher 8 bits from look-up table   (34H) to
                                        ACC
LD          R02,A                       ;pass the value from ACC (34H)to R02
...                                     ;user program

ORG         0600H                       ;start address of table
DW          1234H                       ;table content at 0600H
DW          2345H                       ;table content at 0601H
DW          3456H                       ;table content at 0602H
DW          0000H                       ;table content at 0603H
```

2.1.1.4 Jump Table

Jump table can achieve multi-address jump feature. Since the addition of PCL and ACC is the new value of PCL, multi-address jump is then achieved through adding different valueof ACC to PCL. If the value of ACC isn, then PCL+ACC represent the current address plus n. After the execeution of the current instructions, the value of PCL will add 1 (refer to the following examples). If PCL+ACC overflows, then PC will not carry. As such, user can achieve multi-address jump through setting different values of ACC.

PCLATH is the PC high bit buffer register. Before operating on PCL, value must be given to PCLATH.

Example: correct illustration of multi-address jump

```
FLASH address
              LDIA        01H
              LD          PCLATH,A      ;must give value to PCLATH
              …
0110H:        ADDR        PCL           ;ACC+PCL
0111H:        JP          LOOP1         ;ACC=0, jump to LOOP1
0112H:        JP          LOOP2         ;ACC=1, jump to LOOP2
0113H:        JP          LOOP3         ;ACC=2, jump to LOOP3
0114H:        JP          LOOP4         ;ACC=3, jump to LOOP4
0115H:        JP          LOOP5         ;ACC=4, jump to LOOP5
0116H:        JP          LOOP6         ;ACC=5, jump to LOOP6
```

Example: wrong illustration of multi-address jump

```
FLASH address
              CLR         PCLATH
              …
00FCH:        ADDR        PCL           ;ACC+PCL
00FDH:        JP          LOOP1         ;ACC=0, jump toLOOP1
00FEH:        JP          LOOP2         ;ACC=1, jump toLOOP2
00FFH:        JP          LOOP3         ;ACC=2, jump toLOOP3
0100H:        JP          LOOP4         ;ACC=3, jump to0000H address
0101H:        JP          LOOP5         ;ACC=4, jump to0001H address
0102H:        JP          LOOP6         ;ACC=5, jump to0002H address
```

Note: Since PCl overflow will not carry to the higher bits, the program cannot be placed at the partition of the FLASH space when using PCL to achieve multi-address jump.

2.1.2 Data Memory

List of data memory of CMS79FT73x

	address		address		address		address
INDF	00H	INDF	80H	INDF	100H	INDF	180H
TMR0	01H	OPTION_REG	81H	TMR0	101H	OPTION_REG	181H
PCL	02H	PCL	82H	PCL	102H	PCL	182H
STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H
FSR	04H	FSR	84H	FSR	104H	FSR	184H
PORTA	05H	TRISA	85H	RCSTA1	105H	LVDCON	185H
PORTB	06H	TRISB	86H	PORTB	106H	TRISB	186H
PORTC	07H	TRISC	87H	WPUA	107H	DIVS1	187H
PORTD	08H	TRISD	88H	WPUC	108H	DIVS0	188H
ANSEL0	09H	ANSEL1	89H	ANSEL2	109H	DIVCON	189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
PIR1	0CH	PIE1	8CH	EEDAT	10CH	DIVE3/DIVQ3	18CH
PIR2	0DH	PIE2	8DH	EEADR	10DH	DIVE2/DIVQ2	18DH
TMR1L	0EH	WPUD	8EH	EEDATH	10EH	DIVE1/DIVQ1	18EH
TMR1H	0FH	OSCCON	8FH	EEADRH	10FH	DIVE0/DIVQ0	18FH
T1CON	10H	WDTCON	90H	TABLE_SPH	110H	----	190H
TMR2	11H	SSPCON2	91H	TABLE_SPL	111H	----	191H
T2CON	12H	PR2	92H	TABLE_DATAH	112H	----	192H
SSPBUF	13H	SSPADD	93H	LEDCON0	113H	----	193H
SSPCON	14H	SSPSTAT	94H	LEDCON1	114H	----	194H
CCPR1L	15H	WPUB	95H	LEDADD	115H	----	195H
CCPR1H	16H	IOCB	96H	LEDDATA	116H	----	196H
CCP1CON	17H	WPDB	97H	SEGEN2	117H	----	197H
RCSTA0	18H	SPBRG1	98H	SEGEN1	118H	----	198H
TXREG0	19H	PWMCON	99H	SEGEN0	119H	----	199H
RCREG0	1AH	PWM1CYC	9AH	COMEN	11AH	----	19AH
CCPR2L	1BH	PWM2CYC	9BH	EECON1	11BH	----	19BH
CCPR2H	1CH	ADRESL	9CH	EECON2	11CH	----	19CH
CCP2CON	1DH	ADRESH	9DH	TXREG1	11DH	----	19DH
TXSTA0	1EH	ADCON0	9EH	RCREG1	11EH	----	19EH
SPBRG0	1FH	ADCON1	9FH	TXSTA1	11FH	----	19FH
Universal register 96 byte	20H	Universal register 80byte	A0H	Universal register 80byte	120H	----	1A0H
	6FH		EFH		16FH		1EFH
	70H	Fast memory spce 70H-7FH	F0H	Fast memory space 70H-7FH	170H	Fast memory space 70H-7FH	1F0H
	--		--		--		--
	7FH		FFH		17FH		1FFH
BANK0		BANK1		BANK2		BANK3	

Data memory consists of 512×8 bits. It can be devided into to space: special function register and universal data memory. Most of data memory are able to write/read data, only some data memory are read-only. Special register address is from 00H-1FH, 80-9FH, 100-11FH, 180-19FH.

Summary of special registers in CMS79FT73x Bank0

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00H	INDF	Look-up for this unit will use FSR, not physical register.								xxxxxxxx
01H	TMR0	TIMER0 data register								xxxxxxxx
02H	PCL	Lower bit of program counter								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	memory pointers for indirect addressing of data memory								xxxxxxxx
05H	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxxxxxx
06H	PORTB	----	RB6	RB5	RB4	RB3	RB2	RB1	RB0	-xxxxxxx
07H	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxxxxxx
08H	PORTD	----	----	----	----	----	RD2	RD1	RD0	-----xxx
09H	ANSEL0	Analog input control register0								00000000
0AH	PCLATH	----	---	----	Write buffer of higher 5 bits of program counter					---00000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0CH	PIR1	----	ADIF	RC0IF	TX0IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-0000000
0DH	PIR2	LVDIF	----	----	EEIF	BCLIF	TX1IF	RC1IF	CCP2IF	0—00000
0EH	TMR1L	Data register of 16-bits TIMER1 register lower bit								xxxxxxxx
0FH	TMR1H	Data register of 16-bits TIMER1 register higher bit								xxxxxxxx
10H	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	00000000
11H	TMR2	TIMER2 mod register								00000000
12H	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-0000000
13H	SSPBUF	Synchronous serial port receive buffer /transmit register								xxxxxxxx
14H	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	00000000
15H	CCPR1L	Lower bit of capture/compare/PWM register1								xxxxxxxx
16H	CCPR1H	Higher bit of capture/compare/PWM register1								xxxxxxxx
17H	CCP1CON	----	----	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--000000
18H	RCSTA0	SPEN0	RX9EN0	SREN0	CREN0	RCIDL0	FERR0	OERR0	RX9D0	00001000
19H	TXREG0	USART0 transmit data register								00000000
1AH	RCREG0	USART0 receive data register								00000000
1BH	CCPR2L	Lower bit of capture/compare/PWM register2								xxxxxxxx
1CH	CCPR2H	Higher bit of capture/compare/PWM register2								xxxxxxxx
1DH	CCP2CON	----	----	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--000000
1EH	TXSTA0	CSRC0	TX9EN0	TXEN0	SYNC0	SCKP0	----	TRMT0	TX9D0	00000-10
1FH	SPBRG0	BRG07	BRG06	BRG05	BRG04	BRG03	BRG02	BRG01	BRG00	00000000

Summary of special registers in CMS79FT73x Bank1

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
80H	INDF	Look-up for this unit will use FSR, not physical register.								xxxxxxxx
81H	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	11111011
82H	PCL	Lower bit of program counter								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
84H	FSR	memory pointers for indirect addressing of data memory								xxxxxxxx
85H	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	11111111
86H	TRISB	----	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	-1111111
87H	TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	11111111
88H	TRISD	----	----	----	----	----	TRISD2	TRISD1	TRISD0	-----111
89H	ANSEL1	analog input control register1								00000000
8AH	PCLATH	----	----	----	Write buffer of higher 5 bits of program counter					---00000
8BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	PIE1	----	ADIE	RC0IE	TX0IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-0000000
8DH	PIE2	LVDIE	----	----	EEIE	BCLIE	TX1IE	RC1IE	CCP2IE	0—00000
8EH	WPUD	----	----	----	----	----	WPUD2	WPUD1	WPUD0	-----000
8FH	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	SCS	-110---0
90H	WDTCON	----	----	----	----	----	----	----	SWDTEN	-------0
91H	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	01000000
92H	PR2	TIMER2 period register								11111111
93H	SSPADD	Synchronous serial port (I^2C mode) address register								00000000
93H	SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	11111111
94H	SSPSTAT	----	CKE	D/A	P	S	R/W	UA	BF	-0000000
95H	WPUB	----	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	-0000000
96H	IOCB	----	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0	-0000000
97H	WPDB	----	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0	-0000000
98H	SPBRG1	BRG17	BRG16	BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	00000000
99H	PWMCON	----	CYC2EN	CK2[1:0]		----	CYC1EN	CK1[1:0]		-000-000
9AH	PWM1CYC	PWM1 period data register								11111111
9BH	PWM2CYC	PWM2 period data register								11111111
9CH	ADRESL	Lower bit of A/D result register								xxxxxxxx
9DH	ADRESH	Higher bit of A/D result register								xxxxxxxx
9EH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	ADON	00000000
9FH	ADCON1	ADFM	CHS4	----	----	----	----	----	----	00------

Summary of special registers in CMS79FT73x Bank2

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
100H	INDF	Look-up for this unit will use FSR , not physical register.								xxxxxxxx
101H	TMR0	TIMER0 mod register								xxxxxxxx
102H	PCL	Lower bit of program counter (PC)								00000000
103H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
104H	FSR	memory pointers for indirect addressing of data memory								xxxxxxxx
105H	RCSTA1	SPEN1	RX9EN1	SREN1	CREN1	RCIDL1	FERR1	OERR1	RX9D1	00001000
106H	PORTB	----	RB6	RB5	RB4	RB3	RB2	RB1	RB0	-xxxxxxx
107H	WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	00000000
108H	WPUC	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0	00000000
109H	ANSEL2	analog input control register2								0
10AH	PCLATH	----	----	---	Write buffer of higher 5 bits of program counter					---00000
10BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
10CH	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx
10DH	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	00000000
10EH	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	xxxxxxxx
10FH	EEADRH	----	----	----	EEADRH4	EEADRH3	EEADRH2	EEADRH1	EEADRH0	---00000
110H	TABLE_SPH	----	----	----	Pointers for higher 5 bits of the table					---xxxxx
111H	TABLE_SPL	Pointers for low bits in table								xxxxxxxx
112H	TABLE_DATAH	Data for high bits in table								xxxxxxxx
113H	LEDCON0	LCDEN	LEDEN	COMSEL1	COMSEL0	LEDCLK3	LEDCLK2	LEDCLK1	LEDCLK0	00000000
114H	LEDCON1	----	LEDF	SEGOUT1	SEGOUT0	----				-000----
115H	LEDADD	LEDCS	----	COMSEL2	LEDADD[4:0]					0-000000
116H	LEDDATA	LED data register								xxxxxxxx
117H	SEGEN2	current control register2 for SEG port driver				----	----	----	----	0000----
118H	SEGEN1	current control register1 for SEG port driver								-----000
119H	SEGEN0	current control register0 for SEG port driver								00000000
11AH	COMEN	control register for COM port								00000000
11BH	EECON1	EEPGD	----	----	----	WRERR	WREN	WR	RD	0---x000
11CH	EECON2	EEPROM control register2 (not physical register)								--------
11DH	TXREG1	USART1 transmit data register								00000000
11EH	RCREG1	USART1 receive data register								00000000
11FH	TXSTA1	CSRC1	TX9EN1	TXEN1	SYNC1	SCKP1	----	TRMT1	TX9D1	00000-10

Summary of special registers in CMS79FT73x Bank3

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
180H	INDF	Look-up for this unit will use FSR , not physical register.								xxxxxxxx
181H	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	11111011
182H	PCL	Lower bit of program counter (PC)								00000000
183H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
184H	FSR	memory pointers for indirect addressing of data memory								xxxxxxxx
185H	LVDCON	LVD_RES	----	----	----	LVD_SEL2	LVD_SEL1	LVD_SEL0	LVDEN	00---0000
186H	TRISB	----	TRISB6	TRISB5	TRISB4	TRISB73	TRISB2	TRISB1	TRISB0	-1111111
187H	DIVS1	Higher 8 bits for divisor								00000000
188H	DIVS0	Lower 8 bits for divisor								00000000
189H	DIVCON	DIVEN	CAL_END	----	----	----	----	----	DIV_CLK	01-----0
18AH	PCLATH	----	----	----	Write buffer of higher 5 bits of program counter					---00000
18BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
18CH	DIVE3/DIVQ3	dividend or quotient BIT[31:24]								00000000
18DH	DIVE2/DIVQ2	dividend or quotient BIT[23:16]								00000000
18EH	DIVE1/DIVQ1	dividend or quotient BIT15:8]								00000000
18FH	DIVE0/DIVQ0	dividend or quotient BIT[7:0]								00000000

2.2 Addressing Mode

2.2.1 Direct Addressing

Operate on RAM through accumulator (ACC)

Example: pass the value in ACC to 30H register

```
LD          30H,A
```

Example: pass the value in 30H register to ACC

```
LD          A,30H
```

2.2.2 Immediate Addressing

Pass the immediate value to accumulator (ACC).

Example: pass immediate value 12H to ACC

```
LDIA        12H
```

2.2.3 Indirect Addressing

Data memory can be direct or indirect addressing. Direct addressing can be achieved through INDF register, INDF is not physical register. When load/save value in INDF, address is the value in FSR register (lower 8 bits) and IRP bit in STATUS register (9th bit) ， and point to the register of this address. Therefore, after setting the FSR register and the IRP bit of STATUS register, INDF register can be regarded as purpose register. Read INDF (FSR=0) indirectly will produce 00H. Write INDF register indirectly will casue an empty action. The following example shows how indirect addressing works.

Example: application of FSR and INDF

```
        LDIA        30H
        LD          FSR,A           ;Points to 30H for indirect addressing
        CLRB        STATUS,IRP      ;clear the 9th bit of pointer
        CLR         INDF            ;clear INDF, which mean clear the 30H address RAM tha
                                    FSR points to
```

Example: clear RAM (20H-7FH) for indirect addressing:

```
        LDIA        1FH
        LD          FSR,A           ;Points to 1FH for indirect addressing
        CLRB        STATUS,IRP
LOOP:
        INCR        FSR             ;address add 1， initial address is 30H
        CLR         INDF            ;clear the address where FSR points to
        LDIA        7FH
        SUBA        FSR
        SNZB        STATUS,C        ;clear until the address of FSR is 7FH
        JP          LOOP
```

2.3 Stack

Stack buffer of the chip has 8 levels. Stack buffer is not part of data memorynor program memory. It cannot be written nor read. Operation on stack buffer is through stack pointers， which also cannot be written nor read. After system resets, SP points to the top of the stack. When sub-program happens or interrupts happens, value in program counter (PC)will be transferred to stack buffer. When return from interrupt or return from sub-program, value is transferred back to PC. The following digram illustrates its working principle.

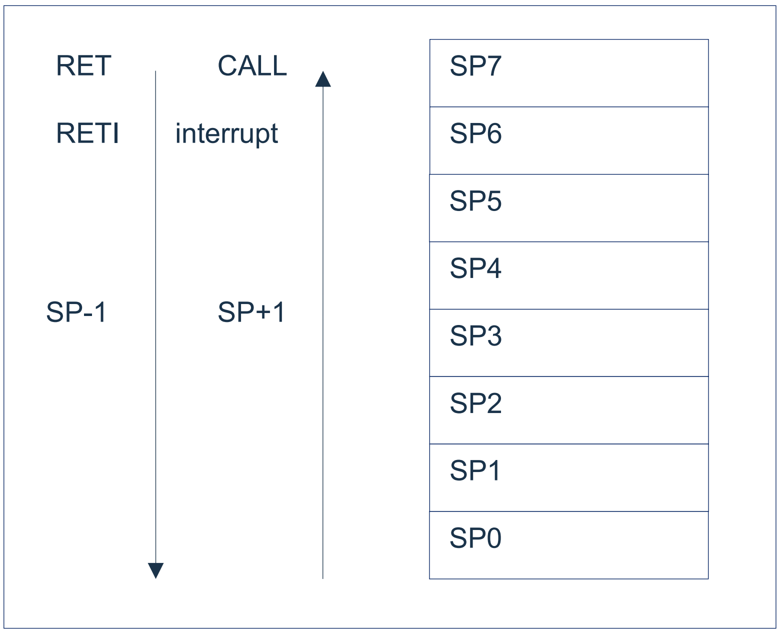


Fig 2-2： stack buffer working principle

Stack bufferwill follow one principle: ‘first in last out’

Note: stackbuffer has only 8 levels, if the stack is full and interrupt happens which can not be screened out, then only the indication bit of the interrupt will be noted down. The response for the interrupt will be suppressed until the pointer of stack starts to decrease. This feature can prevent overflow of the stack caused by interrupt. Similarly, when stack is full and sub-program happens, then stack will overflow and the contents which enter the stack first will be lost, only the last 8 return address will be saved.

2.4 Accumulator (ACC)

2.4.1 General

ALU is the 8-bit arithmetic-logic unit. All math and logic related calculations in MCU are done by ALU. It can perform addition, subtraction, shift and logical calculation on data; ALU can also control STATUS to represent the status of the product of the calculation.

ACC register is an 8-bit register to store the product of calculation of ALU. It does not belong to data memory. It is in CPU and used by ALU during calculation. Hence it cannot be addressed. It can only be used through the instructions provided.

2.4.2 ACC Applications

Example: use ACC for data transfer

```
LD          A,R01       ;pass the value in register R01 to ACC
LD          R02,A       ;pass the value in ACC to register R02
```

Example: use ACC for immediate addressing

```
LDIA        30H         ;load the ACC as 30H
ANDIA       30H         ;run ‘AND’ between value in ACC and immediate number
                        30H,save the result in ACC
XORIA       30H         ; run ‘XOR’ between value in ACC and immediate number
                        30H,save the result in ACC
```

Example: use ACC as the first operand of the double operand instructions

```
HSUBA       R01         ;ACC-R01，save the result in ACC
HSUBR       R01         ;ACC-R01，save the result in R01
```

Example: use ACC as the second operand of the double operand instructions

```
SUBA        R01         ;R01-ACC，save the result in ACC
SUBR        R01         ; R01-ACC，save the result in R01
```

2.5 Program Status Register (STATUS)

STATUS register includes:

- status of ALU.
- Reset status.
- Selection bit of Data memory (GPR and SFR)

Just like other registers, STATUS register can be the target register of any other instruction. If A instructions that affects Z, DC or C bit that use STATUS as target register, then it cannot write on these 3 status bits. These bits are cleared or set to 1 according to device logic. TO and PD bit also cabbot be written. Hence the instructions which use STATUS as target instruction may not result in what is predicted.

For example, CLRSTATUS will clear higher 3 bits and set the Z bit to 1. Hence the value of STATUS will be 000u u1uu (u will not change.). Hence, it is recommended to only use CLRB, SETB, SWAPA and SWAPR instructions to change STATUS register because these will not affect any status bits.

program status register STATUS (03H)

03H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	1	1	X	X	X

Bit7	IRP:	Selection bit of register memory (for indirect addressing)
	1=	Bank2 and Bank3 (100h-1FFh)；
	0=	Bank0 and Bank1 (00h-FFh).
Bit6~Bit5	RP [1:0]:	Selection bit of memory；
	00:	Select Bank 0；
	01:	Select Bank 1；
	10:	Select Bank 2；
	11:	Select Bank 3.
Bit4	TO:	Time out bit；
	1=	Power on or CLRWDT instructions or STOP instructions；
	0=	WDT time out.
Bit3	PD:	Power down；
	1=	Power on or CLRWDT instructions；
	0=	STOP instructions.
Bit2	Z:	Bit for result inzero；
	1=	Result is 0；
	0=	Result is not 0
Bit1	DC:	Carry bit；
	1=	When carry happens to higher bits or no borror happens in Lower 4 bits in the result；
	0=	When no carry happens to higher bits or borror happens in Lower 4 bits in the result.
Bit0	C:	Carry/borrow bit ；
	1=	When carry happens at the highest bit or no borrow happens；
	0=	When no carry happens at the highest bit or borrow happens

TO and PD bit can reflect the reason for reset of chip. The following is the events which affects the TO and PD and the status of TO nad PD after these events.

events	TO	PD
Power on	1	1
WDT overflow	0	X
STOP instructions	1	0
CLRWDT instructions	1	1
sleep	1	0

Events which affect TO/PD

TO	PD	Reset reason
0	0	WDT overflow awaken MCU
0	1	WDT overflow non-sleep status
1	1	Power on

TO/PD status after reset

2.6 Pre-scaler (OPTION_REG)

OPTION_REG registercan be read or written. Each control bit for configuration is as follow:

- TIMER0/WDT pre-scaler
- TIMER0
- PORTB pull up resistance control

pre-scaler OPTION_REG (81H)

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	0	1	1

Bit7	RBPU:	PORTB pull up enable bit
	1=	Forbidden PORTB pull up.
	0=	Enable PORTB pull up according to latched value of ports.
Bit6	INTEDG:	Edge selection bit for triggering interrupt
	1=	INT pin rising edge triggered interrupt
	0=	INT pin falling edge triggered interrupt
Bit5	T0CS:	Selection bit for TIMER0 clock source.
	0=	Internal instructions period clock ($F_{SYS}/4$).
	1=	transition edge on T0CKI pin
Bit4	T0SE:	Edge selection bit for TIMER0 clock source
	0=	Increase when T0CKI pin signal transite from low to high
	1=	Increase when T0CKI pin signal transite from high to low
Bit3	PSA:	pre-scaler allocation
	0=	pre-scaler allocates to TIMER0 mod
	1=	pre-scaler allocates to WDT
Bit2~Bit0	PS2~PS0:	configuration bit for pre-allocation parameters.

PS2	PS1	PS0	TMR0 frequency ratio	WDTfrequency ratio
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

Pre-scaler register is a 8-bit counter. When surveil on register WDT， it is a post scaler; when it is used as timer or counter, it is called pre-scaler. There is only 1 physical scaler and can only be used for WDT or TIMER0, but not at the same time. This means that if it is used for TIMER0, the WDT cannot use pre-scaler and vice versa.

When used for WDT，CLRWDT instructions will clear pre-scaler and WDT timer

When used for TIMER0，all instruction related to writing TIMER0 (such as ：CLR TMR0, SETB TMR0,1 .etc)will clear pre-scaler.

Whether TIMER0 or WDT uses pre-scaler is full controlled by software. This can be changed dynamically. To avoid unintended chip reset, when switch from TIMER0 to WDT, the following instructions should be executed.

```
CLRB      INTCON,GIE          ; Disable enable bit for interrupt to avoid
                              entering interrupt during the following time
                              series
LDIA      B'00000111'
ORR       OPTION_REG,A        ; set pre-scaler as its max value
CLR       TMR0                ; clear TMR0
SETB      OPTION_REG,PSA      ; set pre-scaler to allocate to WDT
CLRWDT                        ; clear WDT
LDIA      B'xxxx1xxx'         ; set new pre-scaler
LD        OPTION_REG,A
CLRWDT                        ; clear WDT
SETB      INTCON,GIE          ; when interrupt is needed, enable bis is
                              turned on here
```

When switch from WDT to TIMER0 mod，the following instructions should be executed.

```
CLRWDT                        ;clear WDT
LDIA      B'00xx0xxx'         ;set new pre-scaler
LD        OPTION_REG,A
```

Note: in order for TIMER0 to have 1:1 pre-scalling，pre-scaller can be allocated to WDT through PSA position 1 of selection register.

2.7 Program Counter (PC)

program counter (PC)controls the instruction sequence in programe memory FLASH， it can addressin the whole range of FLASH. After obtaining instruction code, PC will increase by 1 and point to the address of the next instruction code. When executing jump, passing value to PCL, sub-program, initializing reset, interrupt, interrupt return, sub-program return and other actions, PC will load the address which is related to the instruction, rather than the address of the next instruction.

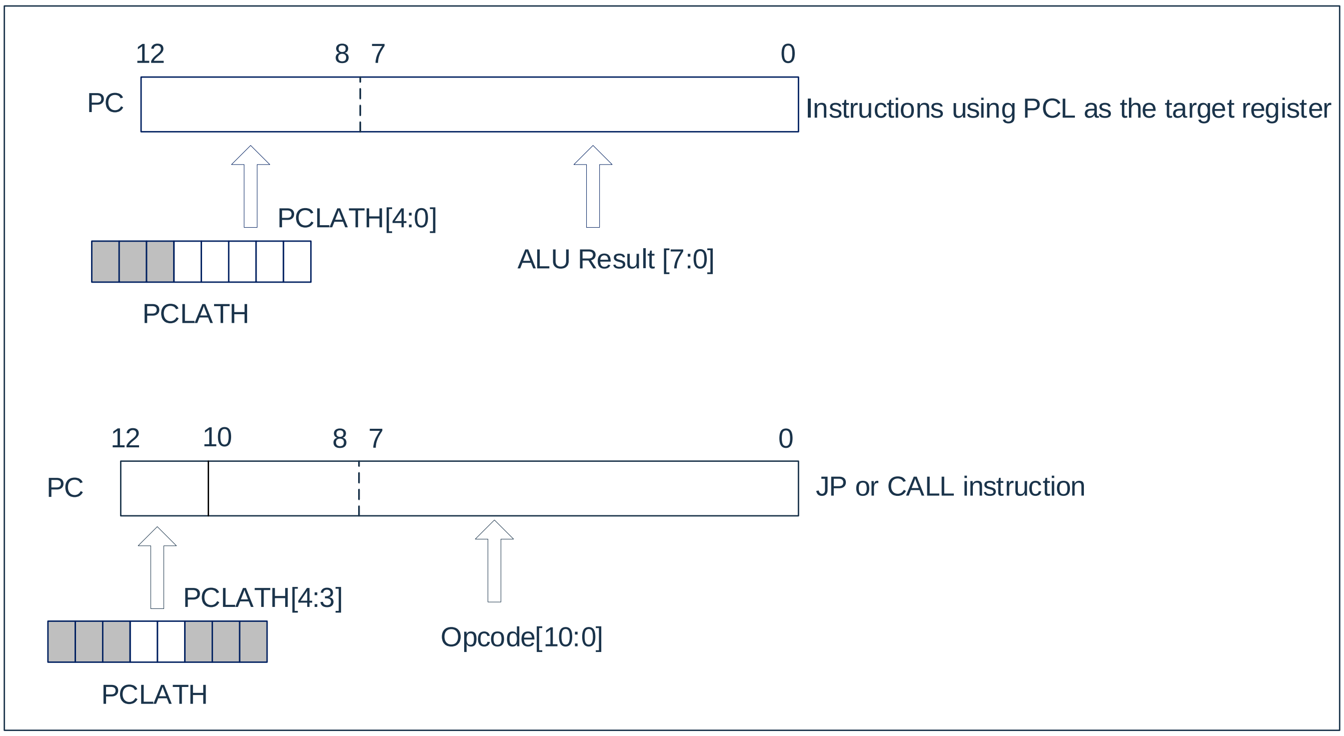
When encountering condition jump instructions and the condition is met, the instruction read during the current instruction will be discarded and an empty instruction period will be inserted. After this, the correct instruction can be obtained. If not, the next instruction will follow the order.

Program counter (PC)is 13 Bit， user can access lower 8 bits through PCL (02H). The higher 5 bits cannot be accessed. It can hold address for 8K×16Bit program. Passing a value to PCL will cause a short jump which range until the 256 addressof the current page.

Note: When using PCL for short jump, it is needed to pass some value to PCLATH

The following are the value of PC under special conditions.

reset	PC=0000；
interrupt	PC=0004 (original PC+1will be add to stack automatically)；
CALL	PC=program defined address (original PC+1will be add to stack automatically)；
RET、RETI、RET i	PC=value coming out from stack；
Operating on PCL	PC[12:8] unchange，PC[7:0]=user defined value；
JP	PC=program defined value；
Other instructions	PC=PC+1；



The following example illustrate the precautions when using JP or CALL

```
        ORG     00H
                JP          LABEL1          Target address LABEL1 located at 300H address,
                                            current value of PCLATH is 00H under the same 2K
                                            range. Hence there is no need to change PCLATH before
                                            executing JP instructions.
                …
        ORG     300H
LABEL1:
                LDIA        08H             Target address LABEL2 located as 900H address,
                LD          PCLATH,A        current value of PCLATH is 00H， under the different 2K
                JP          LABEL2          range. Hence it needed to change PCLATH before
                                            executing JP instructions.
                …
      ORG       7FEH
LABLE4:
                NOP                         ;7FEH
                NOP                         ;7FFH
                NOP                         ;800H
                LDIA        08H             Target address LABEL5 located as 880H address,
                LD          PCLATH,A        current value of PCLATH is 00H   (program is executed
                                            normally， when PC change from 7FFH to 800H,
                JP          LABLE5          PCLATH will not change), under the different 2K range.
                                            Hence it needed to change PCLATH before executing JP
                                            instructions.
                …
       ORG      880H
LABLE5:
                NOP
                RET
                …
        ORG     900H
LABLE2:
                NOP
                                            Target address LABEL3 located at E00H address,
                CALL        LABLE3          current value of PCLATH is 08H under the same 2K
                                            range. Hence there is no need to change PCLATH before
                                            executing CALL instructions.
                LDIA        00H             Target address LABEL4 located as 7FE address， current
                LD          PCLATH,A        value of PCLATH is 08H， under the different 2K range.
                CALL        LABLE4          Hence it needed to change PCLATH before executing
                                            CALL instructions.
                NOP
                …
                …
        ORG     0E00H
LABLE3:
                NOP
                RET
                …
```

2.8 Watchdog Timer (WDT)

Watchdog timer is a self-oscillated RC oscillation timer. There is no need for any external devices. Even the main clock of the chip stops working, WDT can still function/ WDT overflow will cause reset.

2.8.1 WDT Period

WDT and TIMER0 share 8-bit pre-scaler. After all reset, default overflow period fo WDT is 128ms. The way to calculate WDT overflow is 16ms*pre-scalling parameter. If WDT period needs to be changed, you can configure OPTION_REG register. The overflow period is affected by environmental temperature, voltage of the power source and other parameter.

“CLRWDT” and “STOP” instructions will clear counting value inside the WDT timer and pre-scaler (when pre-scaler is allocated to WDT). WDT generally is used to prevent the system and MCU program from being out of control. Under normal condition, WDT shouldbe cleared by “CLRWDT” instructions before overflow to prevent reset being generated. If program is out of control for some reason such that “CLRWDT” instructions is not able to execute before overflow, WDT overflow will then generate reset to make sure the system restarts. If reset is generated by WDT overflow, then ‘TO’bit of STATUS will be cleared to 0. User can judge whether the reset is caused by WDT overflow according to this.

Note:
1) If WDT is used, ‘CLRWDT’ instructions must be placed somewhere is the program to make sure it is cleared before WDT overflow. If not, chip will keep resetting and the system cannot function normally.
2) It is not allowed to clear WDT during interruptso that the main program ‘run away’ can be detected.
3) There should be 1 clear WDT in the main program. Try not to clear WDT inside the sub program, so that the protection feature of watchdog timer can be used largely.
4) Different chips has slightly different overflow time in watchdog timer. When setting clear time for WDT, try to leave extra time for WDT overflow time so that unnecessary WDT reset can be avoided.

2.8.2 Watchdog Timer Control Register WDTCON

WDTCON (90H)

90H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	---	---	---	---	---	---	---	SWDTEN
R/W	---	---	---	---	---	---	---	R/W
Reset value	---	---	---	---	---	---	---	0

Bit7~Bit1	Not used，read as 0	
Bit0	SWDTEN:	Software enable or disable watchdog timer bit
	1=	Enable WDT
	0=	Disable WDT (reset value)

Note: if WDT configuration bit in CONFIG equals 1，then WDT is always enabled and is unrelated to the status of control bit of SWDTEN. if WDT configuration bit in CONFIG equals 0，then it is able to disable WDT using the control bit of SWDTEN.

3. System Clock

3.1 General

When clock signalis input from OSCIN pin (or generated by internal oscillation), 4 non-overlapping orthogonal clock signals called Q1、Q2、Q3、Q4 are produced. Inside IC , each Q1 makes program counter (PC)increase 1, Q4 obtain this instruction from program memory unit and lock it inside instructions register. Compile and execute the instruction obtained between next Q1 and Q4, which means that 4 clock period for 1 executed instruction. The following diagram illustrate the time series of clock and execution of instruction period.

1 instruction period contains 4 Q period. The execution of instructions has pipeline structure. Obtaining instructions only require 1 instruction period, compiling and executing use another instruction period. Since pipeline structure is used, the effective executing time for every instructionis 1 instruction period. If 1 instruction casue PC addressto change (such as JP), then the pre-loaded instruction code is useless and 2 instrcution period is needed to complete this instruction. This is why every operation on PC consumes 2 clock period.

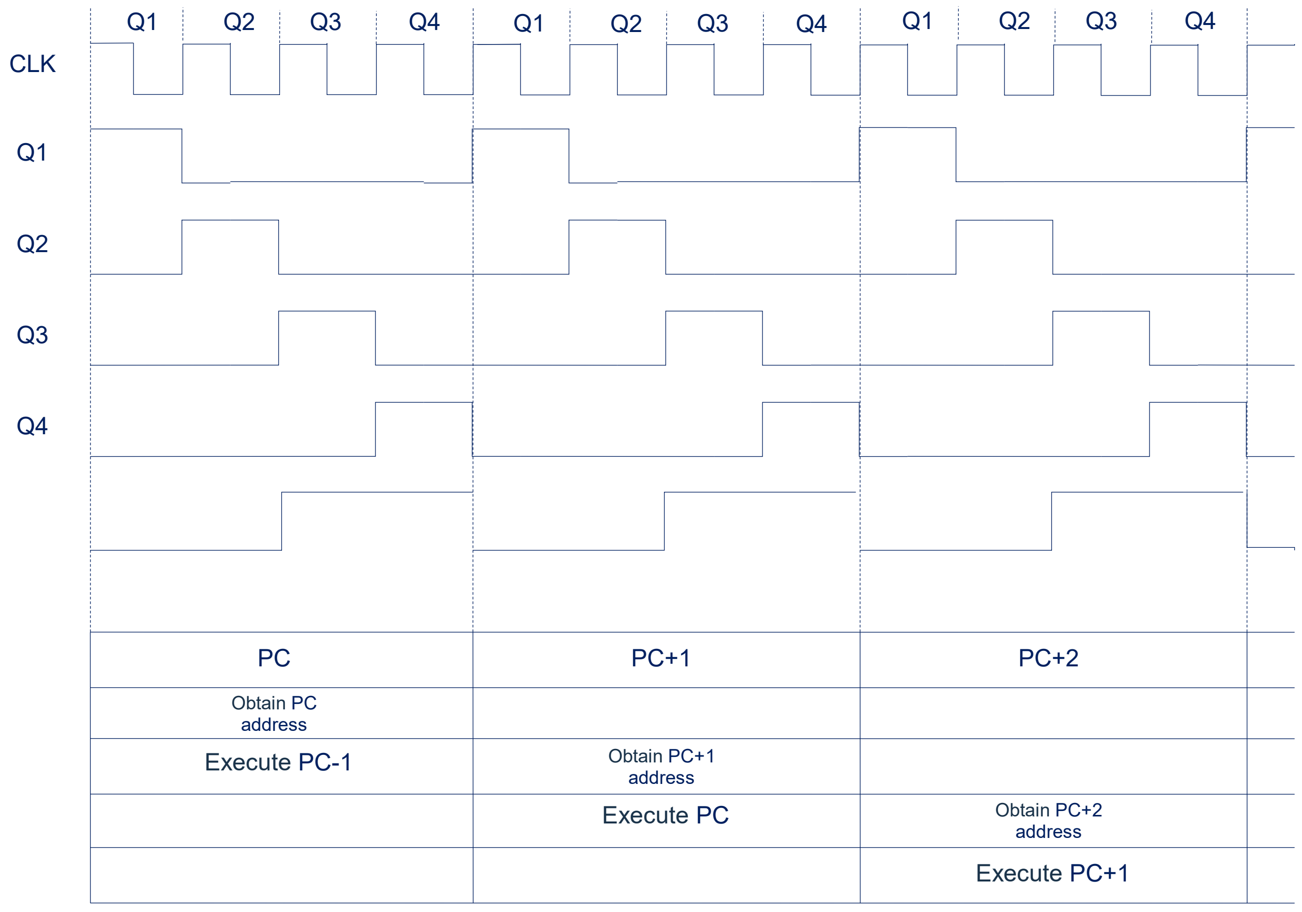


Fig 3-1: time series for clock and instruction period

Following is the relationship between working frequency of system and the speed of instructions:

System frequency (F_{SYS})	Double instruction period	Single instruction period
1MHz	8μs	4μs
2MHz	4μs	2μs
4MHz	2μs	1μs
8MHz	1μs	500ns

3.2 System Oscillator

Chip has 2 ways of oscillation，internalRC oscillation and external XT oscillation.

3.2.1 Internal RC Oscillation

Default oscillation is internal RC oscillation. Its frequency is 8MHz or 16MHz, which is set by OSCCON register.

When select internalRC as oscillator 时，OSCIN and OSCOUT can be used as normal I/O ports.

3.2.2 External XT Oscillation

Select OSC to be XT during burning process, chip works under external XT oscillation mode.此时 internal RC oscillation stops working and OSCIN and OSCOUT are oscillation ports.

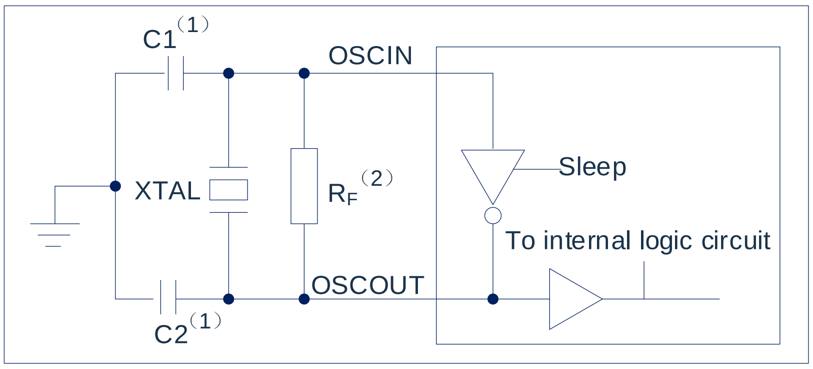


Fig 3-2: Typical XT oscillation

Recommend parameters:

Type	Frequency	Recommend R_F	Recommend C1～C2
XT	4MHz	1MΩ	10Pf～47pF
XT	8MHz	1MΩ	10Pf～47pF
XT	16MHz	1MΩ	10Pf～47pF

3.3 Reset Time

Reset Time is the time for chip to change from reset to stable oscillation. The value is about 16ms.

Note: Reset time exists for both power on reset and other resets.

3.4 Oscillator Control Register

Oscillator control (OSCCON)register controls the system clock and frequency selection. Oscillator tune register OSCTUNE can tune the frequency of internal oscillationin the software.

OSCCON (8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	---	SCS
R/W	---	R/W	R/W	R/W	---	---	---	R/W
reset value	---	1	1	0	---	---	---	0

Bit7	Not used, read 0	
Bit6~Bit4	IRCF<2:0>:	Slection bit for frequency division of Internal oscillator
	111=	F_{SYS} = F_{HSI} /1
	110=	F_{SYS} = F_{HSI} /2 (default)
	101=	F_{SYS} = F_{HSI} /4
	100=	F_{SYS} = F_{HSI} /8
	011=	F_{SYS} = F_{HSI} /16
	010=	F_{SYS} = F_{HSI} /32
	001=	F_{SYS} = F_{HSI} /64
	000=	F_{SYS} = 32kHz (LFINTOSC).
Bit3~Bit1	Not used	
Bit0	SCS:	Selection bit for system clock
	1=	Internal oscillator as system clock
	0=	clock source defined by CONFIG

Note: F_{HSI} as internal oscillator has frequency of 8MHz/16MHz; F_{SYS} is the working frequency of the system.

3.5 Clock Block Diagram

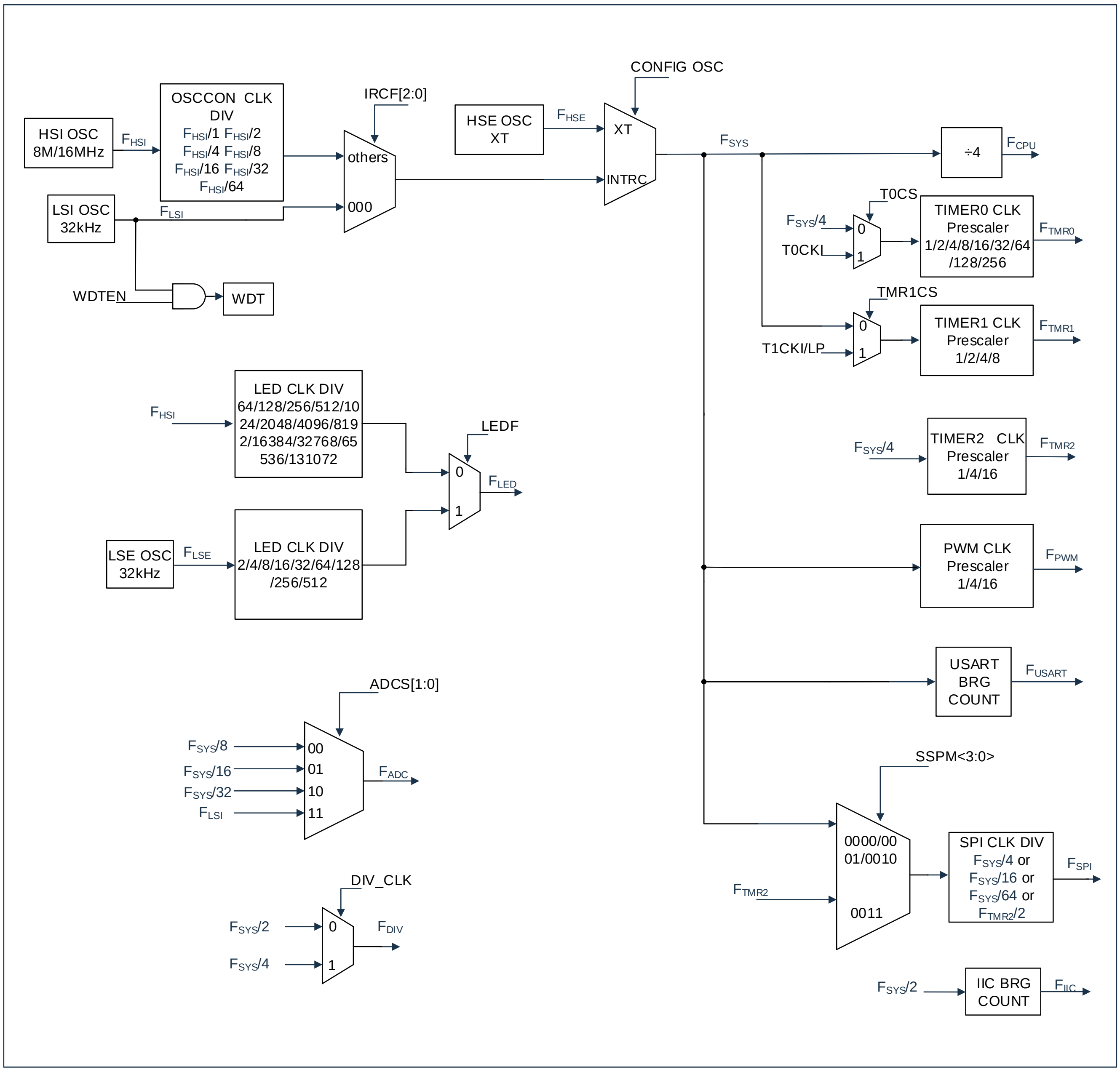


Fig 3-2：clock block diagram

4. Reset

Chip has 3 ways of reset:

- ◆ Power on reset;
- ◆ Low voltage reset;
- ◆ Watchdog overflow reset under normal working condition.

When any reset happens, all system registers reset to defaultcondition, program stops executing and PC is cleared. When finishing resetting, program executes from reset vector 0000H. TO nad PD bit from STATUS can provide information for system reset (see STATUS). User can control the route of the program according to the status of PD and TO.

Any reset requires certain respond time. System provides completed reset procedures to make sure the reset is processed normally.

4.1 Power on Reset

Power on reset is highly related to LVR. Power on process of the sysems should be increasing, after passing some time, the normal electrical level is then reached. The normal time series for power on is as follows:

- Power on: system detects the voltage of the source to increase and wait for it to stabilize;
- System initialization: all system registerset to initial value;
- Oscillator starts working: oscillator starts to provide system clock;
- Executing program: power on process ends, program starts to be executed.

4.2 Power off Reset

4.2.1 General

Power off reset is used for voltage drop caused by external factors (such as interference or change in external load). Voltage drop may enter system dead zone. System dead zone means power source cannot satisfy the minimal working voltage of the system.

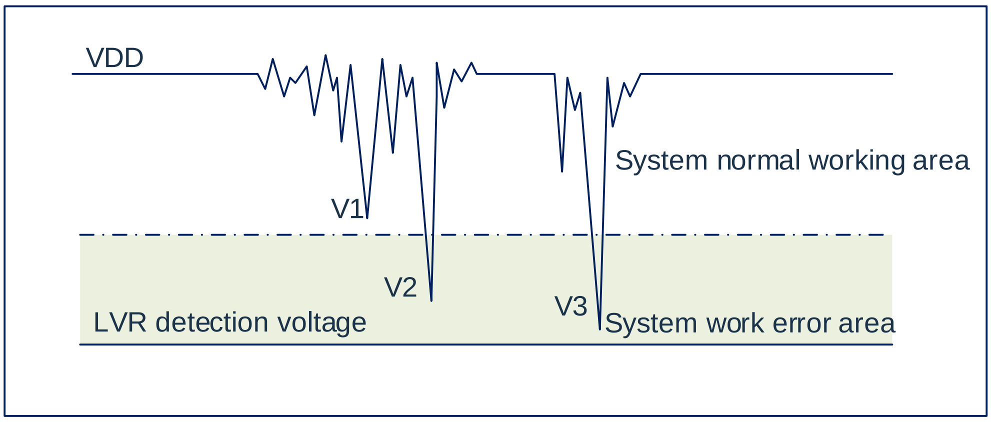


Fig 4-1: power off reset

The above is a typical power off reset case. VDD is under serious interference and the voltage is dropped to a low value. The system works normally above the dotted line and the system enters an unknown situation below the dotted line. This zone is called dead zone. When VDD drops to V1, system still works normally. When VDD drops to V2 and V3, system enters the dead zone and may cause error.

System will enter the dead zone under the following situation:

- DC:
  - Battery provides the power under DC. When the voltage of the battery is too low or the driver of MCU is over-loaded, system voltage may drop and enter the dead zone. Here, power source will not drop further to LVD detection voltage, hence system remains staying at the dead zone.
- AC:
  - When the system is powered by AC, voltage of DC is affected by the noise in AC source. When external over-loaded, such as driving motor, this action will also interfer the DC source. VDD drops below the minimal working voltage due to interference, system may enter untableworking condition.
  - Under AC condition, system power on/off take long time. Power on protection can ensure the system to power on normally, but power off situation is similar to DC case, when AC source is off, VDD drops and may enter dead zone easily.

As illustrated in the above diagram, the normal working voltage is higher than the system reset volateg, at the same time, reset voltage is decided by LVR. When the execution speed increases, the minimal working voltage should increase. However, the system reset volatage is fixed, hence there is a dead zone between the minimal working voltage and system reset voltage.

4.2.2 Improvements for Power off Reset

Suggestions to improve the power off reset:

- ◆ Choose higher LVR voltage;
- ◆ Turn on watchdog timer;
- ◆ Lower working frequency of the system;
- ◆ Increase the gradient of the voltage drop.

Watchdog timer

Watchdog timer is used to make sure the program is run normally. When system enter the dead zone or error happens, watchdog timer overflow and system reset.

Lower the working speed of the system

Higher the working frequency, higher the minimal working voltage system. Dead zone is increase when system works at higher frequency. Therefore, lower the working speed can lower the minimal working voltage and then decrease the probability of entering the dead zone.

Increase the gradient of the voltage drop

This method is used under AC. Voltage drops slowly under AC and cause the system to stay longer at the dead zone. If the system is power on at this moment, error may happen. It is then suggested to insert a resistor between power source and ground to ensure the MCU pass the dead zone and enter the reset zone faster.

4.3 Watchdog Reset

Watchdog reset is a protection for the system. Under normal condition, program clear the the watchdog timer. If error happends and system is under unknown status, watchdog timer overflow and then system reset. After watchdofg reset, system restarts and enter normal working condition.

Time series for watchdog reset:

- Watchdog timer status: system detects watchdog timer. If overflow, then system reset;
- Initialization: all system registerset to default;
- oscillator starts working: oscillator starts to provide system clock;
- program: reset ends, program starts to be executed.

For applications of watchdog timer, see chapters at 2.8

5. Sleep Mode

5.1 Enter Sleep Mode

System can enter sleep modewhen executing STOP instructions. If WDT enabled, then:

- ◆ WDT is cleared and continue to run.
- ◆ PD bit in STATUS register is cleared.
- ◆ TO bit set to 1.
- ◆ Turn off oscillator driver device.
- ◆ I/O port keep at the status before STOP (driver is high level, low lower, or high impedence).

Under sleep mode, to avoid current consumption, all I/O pin should keep at VDD or GND to make sure no extermal circuit is consuming the current from I/O pin. To avoid input pin, suspend and invoke current, high impedence I/O should be pulled to high or low level externally. Internal pull up resistance should also be considered.

5.2 Awaken from Sleep Mode

Awaken through any of the following events:

1. Watchdog timer awake (WDT force enable)
2. PORTB electrical level interrupt or peripherals interrupt

The above 2 events are regards as the extentionof the execution of the program. TO and PD bit in STATUS register are used to find the reason for reset. PD is set to 1 when power on and clear to 0 when STOP instruction is executing.TO is cleared when WDT awaken happens.

When executes STOP instructions， next instruction (PC+1)is withdrawed frist. If it is intended to awaken the system using interrupt, the corresponding enalble bit should be set to 1 for the interrupt. Awaken is not related to GIE bit. If GIE is cleared, system will continue to execute the instruction after STOP instruction， and then junp to interrupt address (0004h) to execute. To avoid instruction after STOP instruction being executed, user should put one NOP instruction after STOP instruction. When system is awaken from sleep mode, WDT will be cleared to 0 and has nothing to do with the reason for awakening.

5.3 Interrupt Awakening

When forbidden overall interrupt (GIE clear)，and there exist 1 interrupt source with its interrupt enable bit and indication bit set to 1, one event from the following will happen:

- If interrupt happens before STOP instructions, then STOP instruction is executed as NOP instructions. Hence, WDT and its pre-scaler and post-scaler will not be cleared, and TO bit will not be set to 1, PD will not be cleared to 0.
- If interrupt happens during or after STOP instruction, then system is awaken from sleep mode. STOP will be executed before system being fully awaken. Hence, WDT and its pre-scaler, post-scaler will be cleared to, TO bit set to 1 and PD bit cleared to 0. Even if the indication bit is 0 before executing the STOP instruction, it can be set to 1 before STOP instruction is finished. To check whether STOP is executed, PD bit can be checked, if is 1, then STOP instruction is executed as NOP. Before executing STOP instruction, 1 CLRWDT instruction must be execited to make sure WDT is cleared.

5.4 Sleep Mode Application

Before system enters sleepmode，if user wants small sllep current, please check all I/O status. If suspended I/O port is required by user, set all suspended ports as output to make sure each I/O has a fixed status and avoid increasing sleep current when I/O is input；turn off AD and other peripherals mod；WDT functions can be turned off to decrease the sleep current.

example：procedures for entering sleep mode

```
SLEEP_MODE：
            CLR         INTCON          ; disable interrupt
            LDIA        B'00000000'
            LD          TRISA,A
            LD          TRISB,A         ;all I/O set as output
            LD          TRISC,A
            LD          TRISE,A
            ...                         ;turn off other functions
            LDIA        0A5H
            LD          SP_FLAG,A       ;set sleep status memory register
            CLRWDT                      ;clear WDT
            STOP                        ;execute STOP instruction
```

5.5 Sleep Mode Awaken Time

When MCU is awaken from sleep mode, oscillation reset time is needed. The specific relationship is shown in the table below:

System main clock source	System clock frequency (IRCF<2:0>)	T_{WAIT}
Internal high-speed RC oscillation (F_{HSI})	$F_{SYS}=F_{HSI}$	$T_{WAIT}=1032*1/F_{HSI}+16*1/F_{HSI}$
	$F_{SYS}= F_{HSI}/2$	$T_{WAIT}=1032*2/F_{HSI}+16*1/F_{HSI}$
	…	…
	$F_{SYS}= F_{HSI}/64$	$T_{WAIT}=1032*64/F_{HSI}+16*1/F_{HSI}$
Interna llow speedRCoscillation ($F_{LFINTOSC}$)	----	$T_{WAIT}=15/F_{LFINTOSC}$
XT oscillation (F_{XT})	----	$T_{WAIT}=2048/F_{XT}$

6. I/O Port

Chip has 4 I/O port: PORTA、PORTB、PORTC、PORTD (max. of 26 I/O).read/write port data register can directly read/write these ports.

Port	Bit	Pin Description	I/O
PORTA	0	Schmitt trigger input，push-pull output，AN0，TK0，LEDdriverSEG □	I/O
	1	Schmitt trigger input，push-pull output，AN1，TK1，LEDdriverSEG □	I/O
	2	Schmitt trigger input，push-pull output，AN2，TK2，LEDdriverSEG □	I/O
	3	Schmitt trigger input，push-pull output，AN3，TK3，LEDdriverSEG □	I/O
	4	Schmitt trigger input，push-pull output，AN4，TK4，LEDdriverSEG □，T0CKI	I/O
	5	Schmitt trigger input，push-pull output，AN5，TK5，LEDdriverSEG □	I/O
	6	Schmitt trigger input，push-pull output，AN6，TK6，LEDdriverSEG □	I/O
	7	Schmitt trigger input，push-pull output，AN7，TK7，LEDdriverSEG □	I/O
PORTB	0	Schmitt trigger input，push-pull output，AN8，TK8，LEDdriverSEG □	I/O
	1	Schmitt trigger input，push-pull output，AN9，TK9，LEDdriverSEG □，externalinterruptinput	I/O
	2	Schmitt trigger input，push-pull output，AN10，TK10，LEDdriverSEG □	I/O
	3	Schmitt trigger input，push-pull output，AN11，TK11，CCP，RX1/DT1	I/O
	4	Schmitt trigger input，push-pull output，AN12，TK12，CCP，TX1/CK1，T1G	I/O
	5	Schmitt trigger input，push-pull output，AN13，TK13，T1CKI，OSCO	I/O
	6	Schmitt trigger input，push-pull output，AN14，TK14，OSCI	I/O
PORTC	0	Schmitt trigger input，push-pull output，AN23，TK23，LCD/LEDdriverCOM □	I/O
	1	Schmitt trigger input，push-pull output，AN22，TK22，LCD/LEDdriverCOM □	I/O
	2	Schmitt trigger input，push-pull output，AN21，TK21，LCD/LEDdriverCOM □	I/O
	3	Schmitt trigger input，push-pull output，AN20，TK20，LCD/LEDdriverCOM □	I/O
	4	Schmitt trigger input，push-pull output，AN19，TK19，LCD/LEDdriverCOM □	I/O
	5	Schmitt trigger input，push-pull output，AN18，TK18，LCD/LEDdriverCOM □	I/O
	6	Schmitt trigger input，push-pull output，AN17，TK17，LCD/LEDdriverCOM □，CCP，RX1/DT1	I/O
	7	Schmitt trigger input，push-pull output，AN16，TK16，LCD/LEDdriverCOM □，CCP，TX1/CK1	I/O
PORTD	0	Schmitt trigger input，push-pull output，AN25，TK25，program clock input，RX0/DT0	I/O
	1	Schmitt trigger input，push-pull output，AN24，TK24，program data input/output，TX0/CK0	I/O
	2	Schmitt trigger input，push-pull output，AN15，TK15	I/O

< Table 6-1: port configuration summary >

6.1 I/O Port Structure

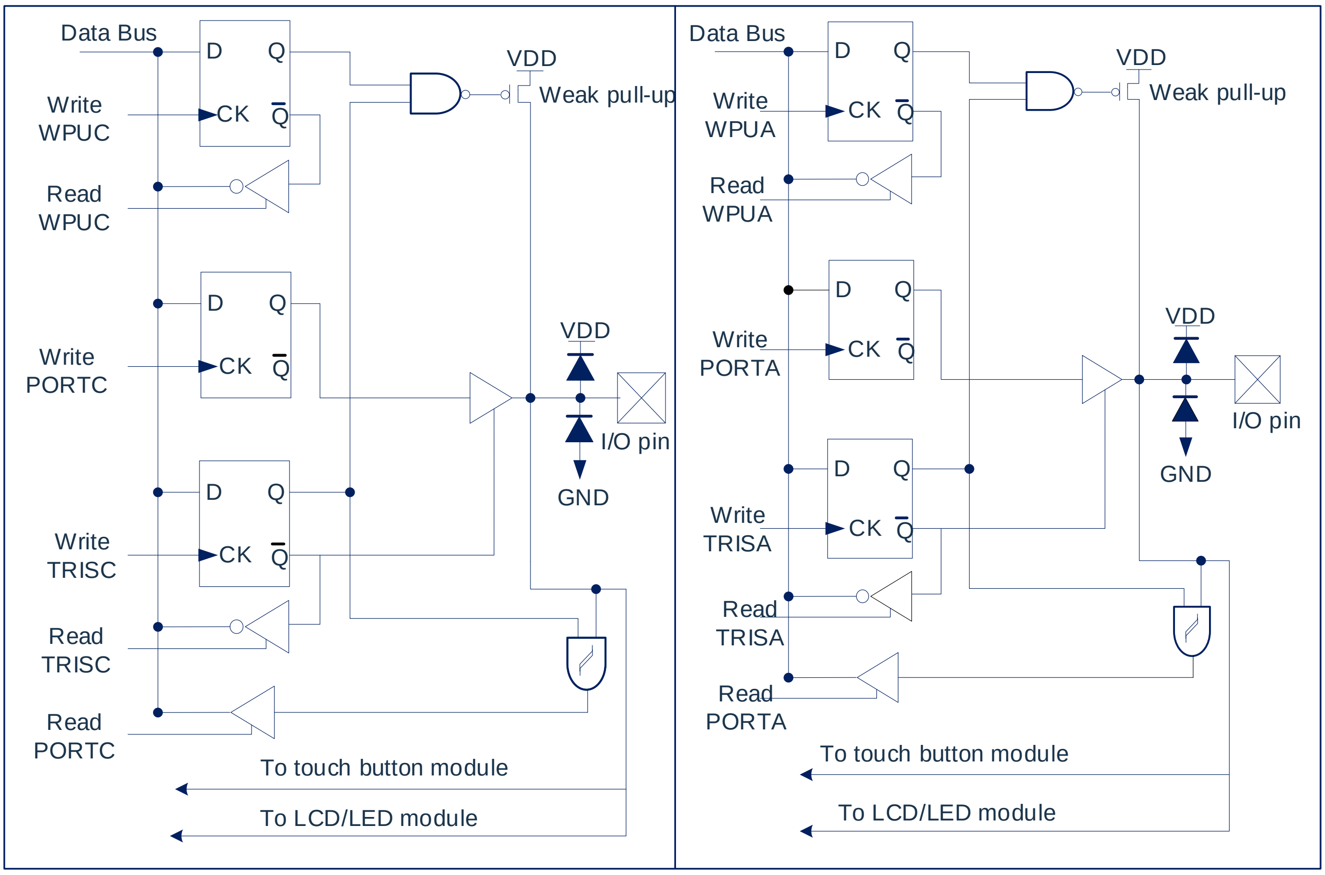


Fig 6-1: I/Oport structure (1)

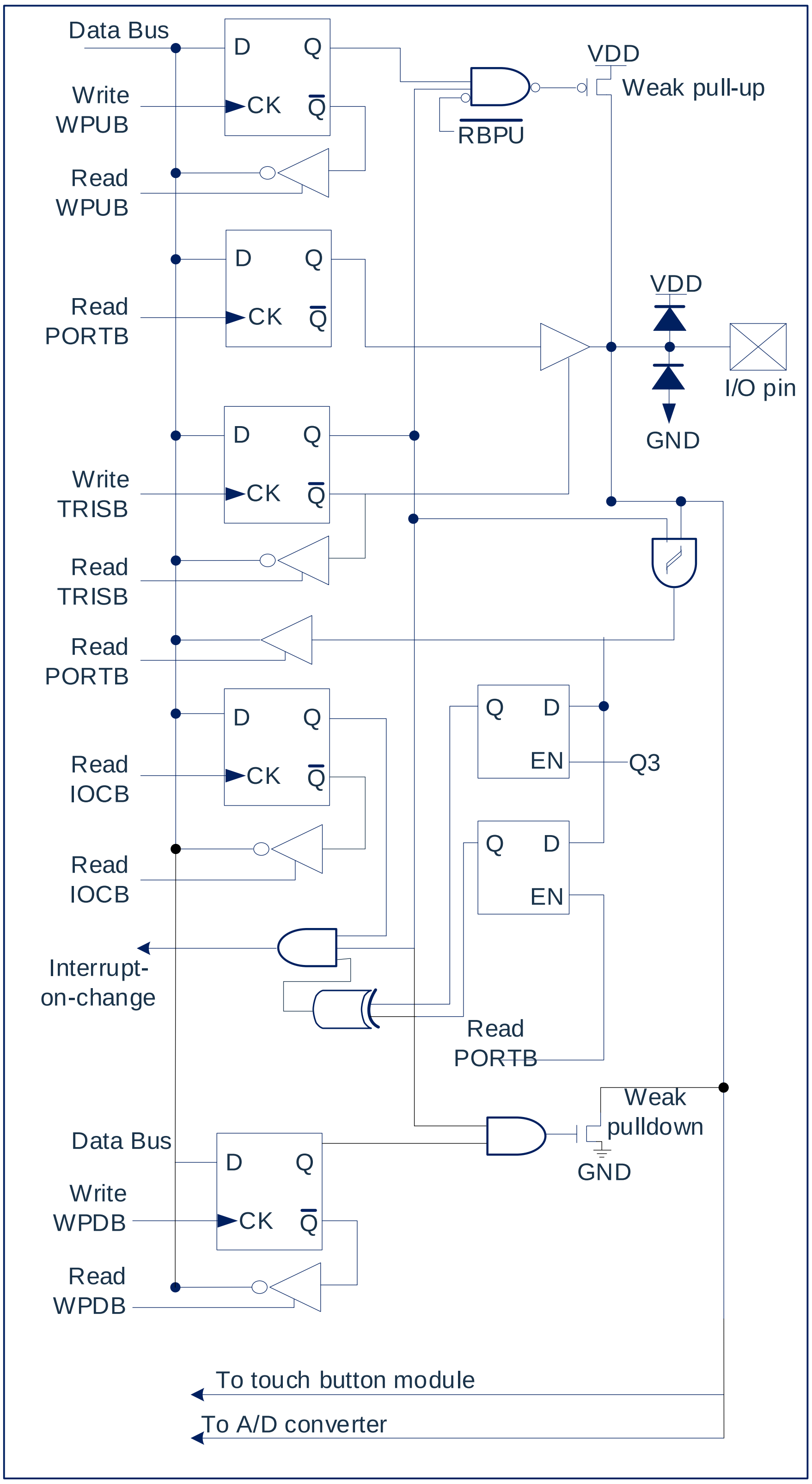


Fig 6-2: I/O port structure (2)

6.2 PORTA

6.2.1 PORTA Data and Direction Control

PORTA is 8 Bit bi-directional port. Its corresponding data direction register is TRISA. Setting 1 bit of TRISA to be 1 can configure the corresponding pin to be input. Setting 1 bit of TRISA to be 0 can configure the corresponding pin to be output.

Reading PORTA register reads the pin status. Writing PORTA write to port latch. All write operation are read-change-write. Hence, write 1 port means read the pin electrical level of the port, change the value and write the value intoport latch. Even when PORTA pin is used as analog input, TRISA registerstill control the direction of PORTA pin. When use PORTA pin as analog input, user must make sure the bits in TRISA register are kept as 1.

Registes related to PORTA ports are PORTA、TRISA、WPUA、ANSEL0 and etc.

PORTA data register PORTA (05H)

05H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
reset value	X	X	X	X	X	X	X	X

Bit7~Bit0	PORTA<7:0>:	PORTAI/O pinbit；
	TRISAx=1	
	1=	Port pin level>V_{IH}；
	0=	Port pin level<V_{IL}.
	TRISAx=0	
	1=	Port output high level；
	0=	Port output low level.

PORTA direction register TRISA (85H)

85H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISA	TRISA6	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
reset value	1	1	1	1	1	1	1	1

Bit7~Bit0	TRISA<7:0>:	PORTA；
	1=	PORTA pin set to be input；
	0=	PORTA pin set to be output

example： procedure for PORTA

```
LDIA        B’11110000’        ;set PORTA<3:0> as output port，PORTA<7:4>as input port
LD          TRISA,A
LDIA        03H                ;PORTA<1:0>output high level，PORTA<3:2>output low level
LD          PORTA,A            ;since PORTA<7:4>are input ports，0 or 1 does not matter
```

6.2.2 PORTA Pull Up Resistance

Each PORTA pin has an internal weak pull up that can be individually configured. The control bits WPUA<7:0> enable or disable each weak pull up. When the portpin is configured as output, its weak pull up will be automatically cut off.

PORTA pull up resistance register WPUA (107H)

107H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	WPUA<7:0>:		Weak pull up register bit
		1=	Enable pull up
		0=	Disable pull up

Note: If pin is configured as output, weak pull up will be automatically disabled

6.2.3 PORTA Analog Control Selection

The ANSEL0 register is used to configure the input mode of I/Opin to analog mode. Setting the appropriate bit in ANSEL0 to 1 will cause all digital read operations of the corresponding pin to return to 0 and make the analog function of the pin work normally. The state of the ANSEL0 bit has no effect on the digital output function. The pin with TRIS cleared and ANSEL0 set to 1 will still be used as a digital output, but the input mode will become an analog mode. This can cause unpredictable results when performing read-modify-write operations on the affected port.

PORT A analog selection register ANSEL0 (09H)

09H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	ANS<7:0>:		Analog selection bit, select the digital or analog function of pin AN<7:0>
		1=	Analog input
		0=	Digital I/O

6.3 PORTB

6.3.1 PORTB Data and Direction

PORTB is a 7Bit wide bi-directional port. The corresponding data direction register is TRISB. Set a bit in TRISB to 1 (=1) to make the corresponding PORTBpin as the input pin. Clearing a bit in TRISB (=0) will make the corresponding PORTB pin as the output pin.

Reading the PORTB register reads the pin status and writing to the register will write the port latch. All write operations are read-modify-write operations. Therefore, writing a port means to read the pin level of the port first, modify the read value, and then write the modified value into the portdata latch. Even when the PORTB pin is used as an analog input, the TRISB register still controls the direction of the PORTB pin. When using the PORTB pin as an analog input, the user must ensure that the bits in the TRISB register remain set as 1.

Related registers with PORTB port include PORTB, TRISB, WPUB, IOCB, WPDB, ANSEL1, etc.

PORTB data register PORTB (06H)

06H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTB	----	RB6	RB5	RB4	RB3	RB2	RB1	RB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	----	X	X	X	X	X	X	X

Bit7	Not used	
Bit6~Bit0	PORTB<6:0>:	PORTB I/O pin bit
	TRISBx=1	
	1=	Port pin level >V_{IH};
	0=	Port pin level<V_{IL}
	TRISBx=0	
	1=	Port output high level;
	0=	Port output low level

PORTB direction register TRISB (86H)

86H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB	----	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
reset value	----	1	1	1	1	1	1	1

Bit7	Not used	
Bit6~Bit0	TRISB<6:0>:	PORTB tri-state control bit
	1=	PORTB pin configured as input
	0=	PORTB pin configured as output

example: PORTB port procedure

```
CLR         PORTB           ;clear data register
LDIA        B’00110000’     ;set PORTB<5:4> as input port, others as output port
LD          TRISB,A
```

6.3.2 PORTB Pull up Resistance

Each PORTB pin has an internal weak pull up that can be individually configured. The control bits WPUB<6:0> enable or disable each weak pull up. When the port pin is configured as output, its weak pull up will be automatically cut off. At power-on reset, weak pull up is prohibited by the RBPU bit of the OPTION_REG register.

PORTB pull up resistance register WPUB (95H)

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	----	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	----	0	0	0	0	0	0	0

Bit7	Not used	
Bit6~Bit0	WPUB<6:0>:	Weak pull up register bit
	1=	Enable pull up
	0=	Disable pull up

Note:
1) To individually enable any pull up, the global RBPU bit of OPTION_RE Gregister must be cleared.
2) If the pin is configured as output or analog input, weak pull up will be automatically prohibited.

6.3.3 PORTB Analog Selection Control

The ANSEL1 register is used to configure the input mode of I/Opin to analog mode. Setting the appropriate bit in ANSEL1 to 1 will cause all digital read operations of the corresponding pin to return to 0 and make the analog function of the pin work normally. The state of the ANSEL1 bit has no effect on the digital output function. The pin whose TRIS is cleared and ANSEL1 is set to 1 is still used as a digital output, but the input mode will become an analog mode. This can cause unpredictable results when executing read-modify-write operations on the affected port.

PORTB analog selection register ANSEL1 (89H)

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL1	----	ANS14	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
reset value	----	0	0	0	0	0	0	0

Bit7	Reference PORTD analog Selection Control	
Bit6~Bit0	ANS<14:8>:	Analog selection bits, select the analog or digital functions of pin AN<14:8>.
	1=	Pin set as analog input.
	0=	Digital I/O

6.3.4 PORTB Level Chage Interrupt Interrupt

All PORTB pins can be individually configured as level change interrupt pins. The control bit IOCB<6:0> allows or disables the interrupt function of each pin. Disable pin level change interrupt function when power on reset.

For the pin that has allowed level change interrupt, compare the value on the pin with the old value latched when PORTB was read last time. Perform a logical OR operation with the output "mismatch" of the last read operation to set the PORTB level change interrupt flag (RBIF) in the INTCON register as 1.

This interrupt can wake up the device from sleep mode, and the user can clear the interrupt in the interrupt service program in the following ways:

-Read or write to PORTB. This will end the mismatch state of the pin level.

-Clear the flag bit RBIF.

The mismatch status will continuously set the RBIF flag bit as 1. Reading or writing PORTB will end the mismatch state and allow the RBIF flag to be cleared. The latch will keep the last read value from the undervoltage reset. After reset, if the mismatch still exists, the RBIF flag will continue to be set as 1.

Note: If the level of the I/O pin changes during the read operation (beginning of the Q2 cycle), the RBIF interrupt flag bit will not be set as 1. In addition, since reading or writing to a port affects all bits of the port, special care must be taken when using multiple pins in interrupt-on-change mode. When dealing with the level change of one pin, you may not notice the level change on the other pin.

PORTB level change interrupt register IOCB (96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCB	----	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	----	0	0	0	0	0	0	0

Bit7	Not used	
Bit6~Bit0	IOCB<6:0>	Control bit of level change interrupt of PORTB
	1=	enable level change interrupt
	0=	disable level change interrupt

6.3.5 PORTB Pull Down Resistance

Each PORTB pin has an internal weak pull-down that can be individually configured. The control bits WPDB<6:0> enable or disable each weak pull-down. When the port pin is configured as output, its weak pull-down will automatically cut off.

PORTB pull down resistance registerWPDB (97H)

97H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDB	----	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
resetvalue	----	0	0	0	0	0	0	0

Bit7	Not used	
Bit6~Bit0	WPDB<6:0>:	Weak pull-down register bit
	1=	Enable pull down
	0=	Disable pull down

Note: If the pin is configured as output or analog input, weak pull-down will be automatically disabled.

6.4 PORTC

6.4.1 PORTC Data and Direction

PORTC is an 8-bit wide bidirectional port. The corresponding data direction register is TRISC. Set a certain position in TRISC to 1 (=1) to make the corresponding PORTC pin as the input pin. Clearing a bit in TRISC (=0) will make the corresponding PORTC pin as the output pin.

Reading the PORTC register reads the pin status and writing to the register will write the port latch. All write operations are read-modify-write operations. Therefore, writing a port means reading the pin level of the port first, modifying the read value, and then writing the modified value to the port data latch.

PORTC data register PORTC (07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0	PORTC<7:0>	PORTC I/O pin bit
	TRISCx=1	
	1=	Port pin level >V_{IH};
	0=	Port pin level <V_{IL}
	TRISCx=0	
	1=	Port output high level;
	0=	Port output low level

PORTC direction register TRISC (87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

Bit7~Bit0	TRISC<7:0>:	Control bit of PORTC tri-state
	1=	PORTC pin configured as input
	0=	PORTC pin configured as output

Example: procedure for PORTC

```
CLR         PORTC           ;clear data register
LDIA        B'01110000'     ;set PORTC<3:0> as output, PORTC<6:4> as input
LD          TRISC,A
```

6.4.2 PORTC pull up resistance

Each PORTC pin has an internal weak pull up that can be individually configured. The control bits WPUC<7:0> enable or disable each weak pull up. When the port pin is configured as output, its weak pull up will be automatically cut off.

PORTCpull up resistance register WPUC (108H)

108H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUC	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	WPUC<7:0>:	Weak pull up register bit
	1=	Enable pull up
	0=	Disable pull up

Note: If the pin is configured as output or analog input, weak pull up will be automatically disabled.

6.4.3 PORTC analog control selection

The ANSEL2 register is used to configure the input mode of I/Opin to analog mode. Setting the appropriate bit in ANSEL2 to 1 will cause all digital read operations of the corresponding pin to return to 0 and make the analog function of the pin work normally. The state of the ANSEL2 bit has no effect on the digital output function. The pin with TRIS cleared and ANSEL2 set to 1 is still used as a digital output, but the input mode will become an analog mode. This can cause unpredictable results when performing read-modify-write operations on the affected port.

PORTB analog selection register ANSEL2 (109H)

109H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL2	ANS23	ANS22	ANS21	ANS20	ANS19	ANS18	ANS17	ANS16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	ANS<23:16>:	Analog selection bit, select the analog or digital function of pin AN<23:16> respectively.
	1=	Pin set as Analog input
	0=	Set as Digital I/O

6.5 PORTD

6.5.1 PORTD data and direction

PORTD is a 3-bit wide bidirectional port. The corresponding data direction register is TRISD. Set a certain position in TRISD to 1 (=1) to make the corresponding PORTD pin as the input pin. Clearing a bit in TRISD (=0) will make the corresponding PORTD pin as the output pin.

Reading the PORTD register reads the pin status and writing to the register will write the port latch. All write operations are read-modify-write operations. Therefore, writing a port means reading the pin level of the port first, modifying the read value, and then writing the modified value to the port data latch.

PORTD data register PORTD (08H)

08H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTD	----	----	----	----	----	RD2	RD1	RD0
R/W	----	----	----	----	----	R/W	R/W	R/W
Reset value	----	----	----	----	----	X	X	X

Bit7~Bit3	Not used	
Bit2~Bit0	PORTD<2:0>:	PORTD I/O pin bit
	TRISDx=1	
	1=	Port pin level >V_{IH};
	0=	Port pin leve<V_{IL}
	TRISDx=0	
	1=	Port output high level;
	0=	Port output low level

PORTD direction register TRISD (88H)

88H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISD	----	----	----	----	----	TRISD2	TRISD1	TRISD0
R/W	----	----	----	----	----	R/W	R/W	R/W
Reset value	----	----	----	----	----	1	1	1

Bit7~Bit3	Not used	
Bit2~Bit0	TRISD<2:0>:	Control bit of PORTD tri-state
	1=	PORTD pin configured as input
	0=	PORTD pin configured as output

example: procedure for PORTD

```
CLR         PORTD          ;clear data register
LDIA        B'01111111'    ;set PORTD<2:0> as input
LD          TRISD,A
```

6.5.2 PORTD pull up resistance

Each PORTD pin has an internal weak pull up that can be individually configured. The control bits WPUD<2:0> enable or disable each weak pull up. When the port pin is configured as output, its weak pull up will be automatically cut off.

PORTD pull up resistance register WPUD (8EH)

8EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUD	----	----	----	----	----	WPUD2	WPUD1	WPUD0
R/W	----	----	----	----	----	R/W	R/W	R/W
Reset value	----	----	----	----	----	0	0	0

Bit7~Bit3	Not used	
Bit2~Bit0	WPUD<2:0>:	Weak pull up register bit
	1=	Enable pull up
	0=	Disable pull up

Note: If the pin is configured as output or analog input, weak pull up will be automatically prohibited. The AN25/AN24 channel corresponding to RD0/RD1 has no analog selection register.

6.5.3 PORTD Analog Selection Control

The ANSEL1 register is used to configure the input mode of I/Opin to analog mode. Setting the appropriate bit in ANSEL1 to 1 will cause all digital read operations of the corresponding pin to return to 0 and make the analog function of the pin work normally. The state of the ANSEL1 bit has no effect on the digital output function. The pin whose TRIS is cleared and ANSEL1 is set to 1 is still used as a digital output, but the input mode will become an analog mode. This can cause unpredictable results when executing read-modify-write operations on the affected port.

PORTB analog selection register ANSEL1 (89H)

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL1	ANS15	----	----	----	----	----	----	----
R/W	R/W	----	----	----	----	----	----	----
reset value	0	----	----	----	----	----	----	----

Bit7	ANS<15>:	Analog selection bits, select the analog or digital functions of pin AN<15>.
	1=	Pin set as analog input.
	0=	Digital I/O
Bit6~Bit0	Reference PORTB analog Selection Control	

6.6 I/O Usage

6.6.1 Write I/O Port

The chip's I/O port register, like the general universal register, can be written through data transmission instructions, bit manipulation instructions, etc.

Example: write I/O port program

```
        LD          PORTA,A         ;pass value of ACC to PORTA
        CLRB        PORTB,1         ;clear PORTB.1
        CLR         PORTC           ;clear PORTC
        SET         PORTA           ;set all output port of PORTA as 1
        SETB        PORTB,1         ;set PORTB.1as 1
```

6.6.2 Read I/O Port

Example: write I/O port program

```
        LD          A,PORTA         ;pass value of PORTA to ACC
        SNZB        PORTA,1         ; check whether PORTA, port 1 is 1, if it is 1, skip the next
                                    statement
        SZB         PORTA,1         ; check if PORTA, 1 port is 0, if 0, skip the next statement
```

Note: When the user reads the status of an I/O port, if the I/O port is an input port, the data read back by the user will be the state of the external level of the port line. If the I/O port is an output port then the read value will be the data of the internal output register of this port.

6.7 Precautions for I/O Port Usage

When operating the I/O port, pay attention to the following aspects:

1. When I/O is converted from output to input, it is necessary to wait for several instruction periods for the I/O port to stabilize.
2. If the internal pull up resistor is used, when the I/O is converted from output to input, the stable time of the internal level is related to the capacitance connected to the I/O port. The user should set the waiting time according to the actual situation. Prevent the I/O port from scanning the level by mistake.
3. When the I/O port is an input port, its input level should be between "VDD+0.7V" and "GND-0.7V". If the input port voltage is not within this range, the method shown in the figure below can be used.

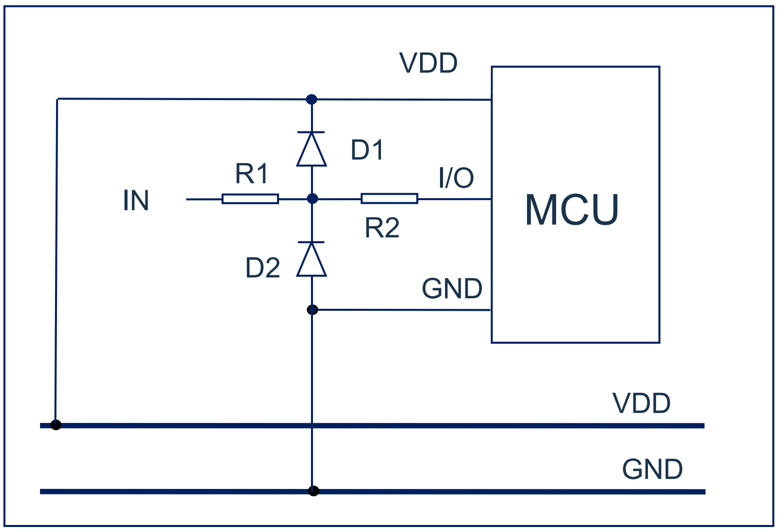


Fig 6-3: The input voltage is not within the specified range

4. If a longer cable is connected to the I/O port, please add a current limiting resistor near the chip I/O to enhance the MCU's anti-EMC capability.

7. Interrupt

7.1 Interrupt General

The chip has the following interrupt source:

- TIMER0 overflow interrupt
- TIMER2 match interrupt
- PORTB level change interrupt
- CCP1/CCP2 interrupt
- USART0/1receive/transmit interrupt
- LVD interrupt
- TIMER1 overflow interrupt
- INT interrupt
- A/D interrupt
- MSSP interrupt
- Program EEPROM write interrupt

The interrupt control register (INTCON) and the peripherals interrupt request register (PIR1, PIR2) record various interrupt requests in their respective flag bits. The INTCON register also includes various interrupt enable bits and global interrupt enable bits.

The global interrupt enable bit GIE (INTCON<7>) allows all unmasked interrupts when set to 1, and prohibits all interrupts when cleared. Each interrupt can be prohibited through the corresponding enable bits in the INTCON, PIE1, and PIE2 registers. GIE is cleared when reset.

Executing the "return from interrupt" instructions, RETI, will exit the interrupt service program and set the GIE bit to 1, thereby re-allowing unshielded interrupt.

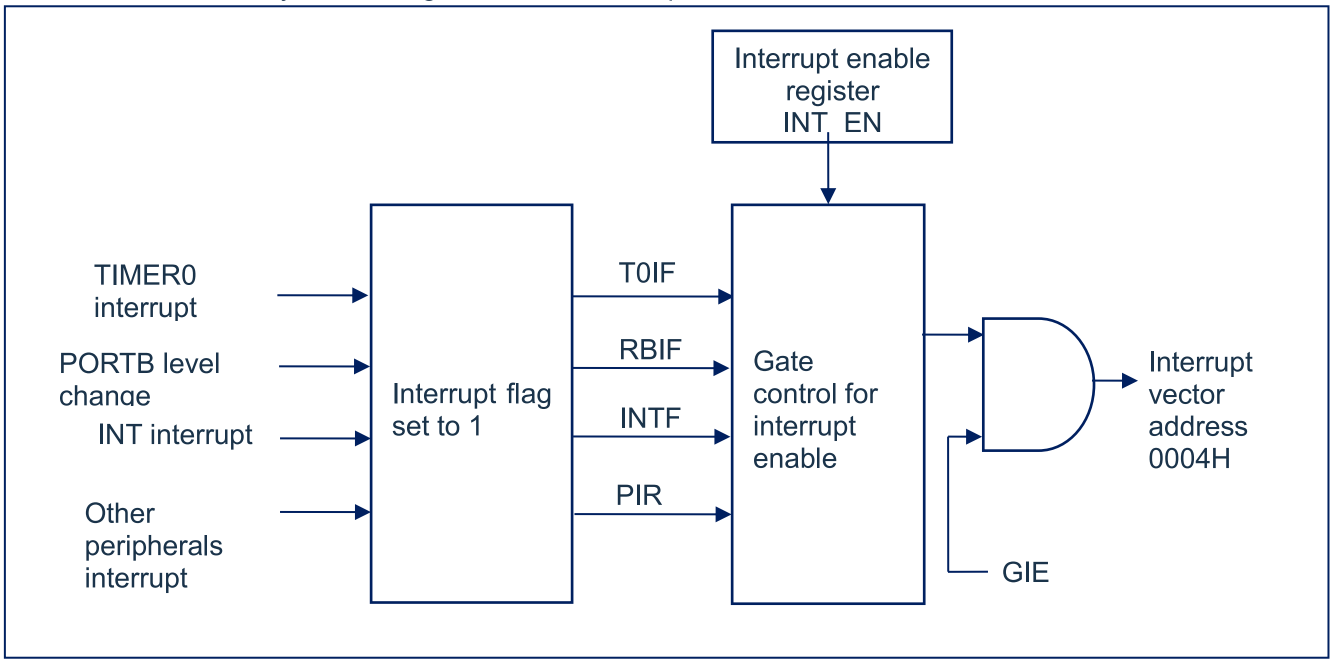

Interrupt enable register INT_EN
TIMER0 interrupt
PORTB level change
INT interrupt
Other peripherals interrupt
Interrupt flag set to 1
T0IF
RBIF
INTF
PIR
Gate control for interrupt enable
GIE
Interrupt vector address 0004H

Fig 7-1： interrupt theory

7.2 Interruptcontrol Register

7.2.1 Interrupt Control Register

The interrupt control registerINTCON is a readable and writable register, including the allowable and flag bits for TMR0 register overflow and PORTB port level change interrupt.

When an interrupt condition occurs, regardless of the state of the corresponding interrupt enable bit or the global enable bit GIE (in the INTCON register), the interrupt flag bit will be set to 1. The user software should ensure that the corresponding interrupt flag bit is cleared before allowing an interrupt.

Interrupt control register INTCON (0BH)

0BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7	GIE:	Global interrupt enable bit;
	1=	Enable all unshielded interrupt;
	0=	Disable all interrupt
Bit6	PEIE:	Peripherals interrupt enable bit;
	1=	Enable all unshielded peripherals interrupt;
	0=	Disable all peripherals interrupt.
Bit5	T0IE:	TIMER0 overflow interrupt enable bit;
	1=	Enable TIMER0 interrupt;
	0=	Disable TIMER0 interrupt
Bit4	INTE:	INT external interrupt enable bit;
	1=	Enable INT external interrupt;
	0=	Disable INT external interrupt
Bit3	RBIE:	PORTB level change interruptenable bit (1);
	1=	Enable PORTB level change interrupt;
	0=	Disable PORTB level change interrupt
Bit2	T0IF:	TIMER0 overflow interrupt enable bit (2);
	1=	TMR0 register overflow already (must clear through software);
	0=	TMR0 register not overflow
Bit1	INTF:	INT external interrupt flag bit;
	1=	INT external interrupt happens (must clear through software);
	0=	INT external interrupt not happen
Bit0	RBIF:	PORTB level change interrupt flag bit;
	1=	The level of at least one pin in the PORTB port has changed (must clear through software);
	0=	None of the PORTB universal I/O pin status has changed.

Note:
1) The IOCB register must also be enabled, and the corresponding port must be set to input state.
2) The T0IF bit is set as 1when TMR0 rolls over to 0. Reset will not change TMR0 and should be initialized before clearing the T0IF bit.

7.2.2 Peripherals Interrupt Enable Register

The peripherals interrupt enable register has PIE1 and PIE2. Before allowing any peripherals interrupt, the PEIE bit of the INTCON register must be set to 1.

Peripherals interrupt enable register PIE1 (8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	---	ADIE	RC0IE	TX0IE	SSPIE	CCP1IE	TMR2IE	TMR1IE
R/W	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	0	0	0	0	0	0	0

Bit7	Not used, read 0	
Bit6	ADIE:	A/D converter (ADC)interrupt enable bit;
	1=	enable ADC interrupt;
	0=	disable ADC interrupt
Bit5	RC0IE:	USART0 receive interrupt enable bit;
	1=	enable USART0 receive interrupt;
	0=	disable USART0 receive interrupt.
Bit4	TX0IE:	USART0 transmit interrupt enable bit;
	1=	enable USART0 transmit interrupt;
	0=	disable USART0 transmit interrupt.
Bit3	SSPIE:	Main synchronous serial port (MSSP)interrupt enable bit;
	1=	enable MSSP interrupt;
	0=	disable MSSP interrupt.
Bit2	CCP1IE:	CCP1 interrupt enable bit;
	1=	enable CCP1 interrupt;
	0=	disable CCP1 interrupt.
Bit1	TMR2IE:	TIMER2 and PR2 match interrupt enable bit;
	1=	enable TMR2 and PR2 match interrupt;
	0=	disable TMR2 and PR2 match interrupt.
Bit0	TMR1IE:	TIMER1 overflow interrupt enable bit;
	1=	enable TIMER1 overflow interrupt;
	0=	disable TIMER1 overflow interrupt.

Peripherals interrupt enable registerPIE2 (8DH)

8DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	LVDIE	---	---	EEIE	BCLIE	RC1IE	TX1IE	CCP2IE
R/W	R/W	---	---	R/W	R/W	R/W	R/W	R/W
Reset value	0	---	---	0	0	0	0	0

Bit7	LVDIE	LVDenable bit;
	1=	enable LVDinterrupt;
	0=	disable LVDinterrupt.
Bit6~Bit5	Not used.	
Bit4	EEIE:	Program EEPROM write operation interrupt enable bit;
	1=	enable program EEPROM write operationinterrupt;
	0=	disable program EEPROM write operation interrupt.
Bit3	BCLIE:	bus conflict interrupt enable bit;
	1=	enable bus conflict interrupt;
	0=	disable bus conflict interrupt.
Bit2	RC1IE:	USART1 receive interrupt enable bit;
	1=	enable USART1 receive interrupt;
	0=	disable USART1 receive interrupt.
Bit1	TX1IE:	USART1t ransmit interrupt enable bit;
	1=	enable USART1 transmit interrupt;
	0=	disable USART1 transmit interrupt.
Bit0	CCP2IE:	CCP2 interrupt enable bit;
	1=	enable CCP2 interrupt;
	0=	disable CCP2 interrupt.

7.2.3 Peripherals Interrupt Request Register

The peripherals interrupt request register is PIR1 and PIR2. When an interrupt condition occurs, regardless of the state of the corresponding interrupt enable bit or the global enable bit GIE, the interrupt flag bit will be set to 1. The user software should ensure that the interrupt is set before allowing an interrupt. The corresponding interrupt flag bit is cleared.

Peripherals interrupt request register PIR1 (0CH)

0CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	---	ADIF	RC0IF	TX0IF	SSPIF	CCP1IF	TMR2IF	TMR1IF
R/W	---	R/W	R	R	R/W	R/W	R/W	R/W
Reset value	---	0	0	0	0	0	0	0

Bit7 Not used, read 0

Bit6 ADIF: A/D converter interrupt flag bit;
1= A/D conversion complete (must clear through software);
0= A/D conversion not complete or not start.

Bit5 RC0IF: USART0 receive interrupt flag bit;
1= USART0 receive buffer full (clear through reading RCREG);
0= USART0 receive buffer empty.

Bit4 TX0IF: USART0 transmit interrupt flag bit;
1= USART0 transmit buffer empty (clear through TXREG);
0= USART0 transmit buffer full.

Bit3 SSPIF: Main synchronous serial port (MSSP)interruptf lag bit.
1= The MSSP interrupt condition is met. Before returning from the interrupt service program, it must clear through software. The conditions for making this bit 1 are::
- SPI;
- transmit/receive happens;
- I^{2}C slave/master control;
- transmit/ receive happens;
- I^{2}C master control;
- The start condition that occurs is done by MSSP mod;
- The stop condition that occurs is completed by MSSP mod;
- The restart condition that occurs is done by MSSPmod;
- The respond condition that occurs is done by MSSPmod;
- The start condition occurs when the MSSP mod is idle (multi-host system);
- The stop condition occurs when the MSSP mod is idle (multi-host system);.

0= No MSSP interrupt condition is met.

Bit2 CCP1IF: CCP1 interrupt flag bit.
Capture mode: 1= Capture for TMR1 register happens (must clear through software);
0= Capture for TMR1 register not happen
Compare mode:
1= Compare match for TMR1 register happens (must clear through software);
0= Compare match for TMR1 register not happen.
PWM mode: Not used under this mode.

Bit1 TMR2IF: TIMER2 and PR2 match interrupt flag bit.
1= TIMER2 and PR2 match heppens (must clear through software);
0= TIMER2 and PR2 not match.

Bit0 TMR1IF: TIMER1 overflow interrupt flag bit.
1= TMR1 register overflow (must clear through software);
0= TMR1 register not overflow.

Peripherals interrupt request registerPIR2 (0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	LVDIF	---	---	EEIF	BCLIF	TX1IF	RX1IF	CCP2IF
R/W	R/W	---	---	R/W	R/W	R	R	R/W
Reset value	0	---	---	0	0	0	0	0

Bit7	LVDIF	LVDinterruptflag bit；
	1=	LVD interrupt happens；
	0=	LVDinterrupt not happen.
Bit6~Bit5	Not used.	
Bit4	EEIF:	Program EEPROM write operation interrupt flag bit;
	1=	write operation complete (must clear through software);
	0=	write operation not complete or not start.
Bit3	BCLIF:	bus conflict interrupt flag bit;
	1=	When configured as I^2Cmaster control mode, bus conflict happens in MSSP;
	0=	No bus conflict.
Bit2	TX1IF:	USART1 transmit interrupt flag bit；
	1=	USART1 transmit buffer empty (clear through writing TXREG1);
	0=	USART1 transmit buffer full.
Bit1	RC1IF:	USART1 receive interrupt flag bit;
	1=	USART1 receive buffer full (clear through reading RCREG1);
	0=	USART1 receive buffer empty.
Bit0	CCP2IF:	CCP2 interrupt flag bit.
	Capture mode:	
		1= capture of TMR1 register happens (must clear through software);
		0= capture of TMR1 register not happen.
	Compare mode:	1= compare match of TMR1 register heppens (must clear through software);
		0= compare match of TMR1 register not heppen.
	PWM mode:	Not used under this mode.

7.3 Protection Methods for Interrupt

After an interrupt request occurs and is responded, the program goes to 0004H to execute the interrupt sub-routine. Before responding to the interrupt, the contents of ACC and STATUS must be saved. The chip does not provide dedicated stack saving and unstack recovery instructions, and the user needs to protect ACC and STATUS by himself to avoid possible program operation errors after the interrupt ends.

Example: Stack protection for ACC and STATUS

```
                ORG         0000H
                JP          START               ;start of user program address
                ORG         0004H
                JP          INT_SERVICE         ;interrupt service program
                ORG         0008H
      START:
                ...
                ...
INT_SERVICE:
       PUSH:                                    ;entrance for interruptservice program, save ACC
                                                and STATUS
                LD          ACC_BAK,A           ;save the value of ACC (ACC_BAK needs to be
                                                defined)
                SWAPA       STATUS
                LD          STATUS_BAK,A        ;save the value of STATUS (STATUS_BAK needs
                                                to be defined)
                ...
                ...
        POP:                                    ;exit for interrupt serice program, restore ACC
                                                and STATUS
                SWAPA       STATUS_BAK
                LD          STATUS,A            ;restore STATUS
                SWAPR       ACC_BAK             ;restore ACC
                SWAPA       ACC_BAK
                RETI
```

7.4 Interrupt Priority and Multi-interrupt Nesting

The priority of each interrupt of the chip is equal. When an interrupt is in progress, it will not respond to the other interrupt. Only after the "RETI" instructions are executed, the next interrupt can be responded to.

When multiple interrupts occur at the same time, the MCU does not have a preset interrupt priority. First, the priority of each interrupt must be set in advance; second, the interrupt enable bit and the interrupt control bit are used to control whether the system responds to the interrupt. In the program, the interrupt control bit and interrupt request flag must be checked.

8. TIMER0

8.1 TIMER0 General

TIMER0 is composed of the following functions:

- 8-bit timer/counter register (TMR0);
- 8-bit pre-scaler (shared with watchdog timer);
- Programmable internal or external clock source;
- Programmable external clock edge selection;
- overflow interrupt.

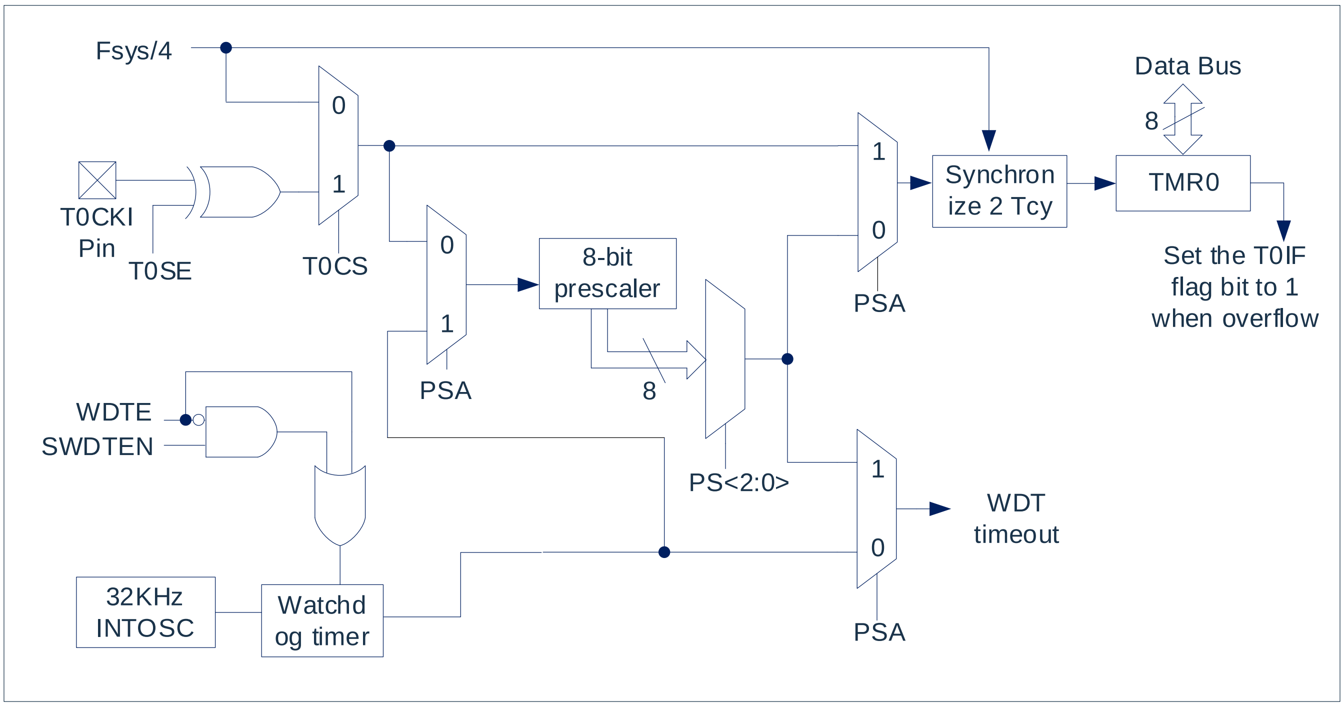


Fig 8-1: TIMER0/WDT mod structure

Note:
1. T0SE, T0CS, PSA, PS<2:0> are the bits in OPTION_RE Gregister.
2. SWDTEN is a bit in the WDTCON register.
3. WDTE bit is in CONFIG.

8.2 Working Principle for TIMER0

The TIMER0 mod can be used as an 8-bit timer or an 8-bit counter.

8.2.1 8-bit Timer Mode

When used as a timer, the TIMER0 mod will be incremented every instruction period (without pre-scaler). The timer mode can be selected by clearing the T0CS bit of the OPTION_REG register to 0. If a write operation is performed to the TMR0 register, the next two Each instruction period will be prohibited from incrementing. The value written to the TMR0 register can be adjusted so that a delay of two instruction periods is included when writing TMR0.

8.2.2 8-bit Counter Mode

When used as a counter, the TIMER0 mod will increment on every rising or falling edge of the T0CKI pin. The incrementing edge depends on the T0SE bit of the OPTION_REG register. The counter mode can be selected by setting the T0CS bit of the OPTION_REG register to 1.

8.2.3 Software Programmable Pre-scaler

TIMER0 and watchdog timer (WDT) share a software programmable pre-scaler, but they cannot be used at the same time. The allocation of the pre-scaler is controlled by the PSA bit of the OPTION_REG register. To allocate the pre-scaler to TIMER0, the PSA bit must be cleared to 0.

TIMER0mod has 8 selections of prescaler ratio, ranging from 1:2 to 1:256. The prescaler ratio can be selected through the PS<2:0> bits of the OPTION_REG register. To make TIMER0 mod have a 1:1 prescaler, the pre-scaler must be assigned to the WDT mod.

The pre-scaler is not readable and writable. When the pre-scaler is assigned to the TIMER0 mod, all instructions written to the TMR0 register will clear the pre-scaler. When the pre-scaler is assigned to the WDT, the CLRWDT instructions will also clear the pre- scaler and WDT.

8.2.4 Switch Prescaler Between TIMER0 and WDT Module

After assigning the pre-scaler to TIMER0 or WDT, an unintentional device reset may occur when switching the prescaler. To change the pre-scaler from TIMER0 to WDT mod, the following instructions must be executed sequence.

Modify pre-scaler (TMR0-WDT)

```
CLRB      INTCON,GIE         ; Turn off the interrupt enable bit to avoid
                             entering the interrupt program when the
                             following specific time series is executed
LDIA      B'00000111'
ORR       OPTION_REG,A       ;set pre-scaler to max. value
CLR       TMR0               ;clear TMR0
SETB      OPTION_REG,PSA     ;set pre-scaler allocate to WDT
CLRWDT                       ;clear WDT
LDIA      B'xxxx1xxx'        ;set new pre-scaler
LD        OPTION_REG,A
CLRWDT                       ;clear WDT
SETB      INTCON,GIE         ;if the program needs to use interrupt, turn
                             on the enable bit here
```

To change the pre-scaler from WDT to TIMER0 mod, the following sequence of instructions must be executed.

Modify pre-scaler (WDT-TMR0)

```
        CLRWDT                                  ;clear WDT
        LDIA            B'00xx0xxx'             ;set new pre-scaler
        LD              OPTION_REG,A
```

8.2.5 TIMER0 Interrupt

When the TMR0 register overflows from FFh to 00h, a TIMER0 interrupt is generated. Every time the TMR0 register overflows, regardless of whether TIMER0 interrupt is allowed, the T0IF interrupt flag bit of the INTCON register will be set to 1. The T0IF bit must be cleared in software. TIMER0 interrupt enable bit is the T0IE bit of the INTCON register.

Note: Because the timer is turned off in sleep mode, the TIMER0 interrupt cannot wake up the processor.

8.3 TIMER0 Related Register

There are two registers related to TIMER0, 8-bit timer/counter (TMR0), and 8-bit programmable control register (OPTION_REG).

TMR0 is an 8-bit readable and writable timer/counter, OPTION_REG is an 8-bit write-only register, the user can change the value of OPTION_REG to change the working mode of TIMER0, etc. Please refer to the application of 0 prescaler register (OPTION_REG).

8-bit timer/counter TMR0 (01H)

01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR0								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

OPTION_REG register (81H)

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	0	1	1

Bit7	RBPU:	PORTB pull up enable bit.
	1=	Disable PORTB pull up.
	0=	Enable PORTB pull up according to latch value in ach port.
Bit6	INTEDG:	Interrupt edge selection bit.
	1=	The rising edge of the INT pin triggers interrupt.
	0=	The falling edge of the INT pin triggers interrupt.
Bit5	T0CS:	TMR0 clock source selection bit.
	1=	Transition edge of T0CKI pin.
	0=	Interna linstruction period clock (F_{SYS}/4).
Bit4	T0SE:	TIMER0 clock source edge selection bit.
	1=	Increment when the T0CKI pin signal transitions from high to low.
	0=	Increment when the T0CKI pin signal transitions from low to high.
Bit3	PSA:	pre-scaler allocation bit.
	1=	pre-scaler allocated to WDT.
	0=	pre-scaler allocated toTIMER0 mod.
Bit2~Bit0	PS2~PS0:	Pre-allocated parameter configuration bits.

PS2	PS1	PS0	TMR0 Frequency division ratio	WDT Frequency division ratio
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

9. TIMER1

9.1 TIMER1 General

TIMER1 mod is a 16-bit timer/counter with the following characteristics:

- 16-bit timer/counter register (TMR1H: TMR1L)
- Programmable internal or external clock source
- 3-bit pre-scaler
- Optional LP oscillator
- Synchronous or asynchronous operation
- Through T1Gpingate control TIMER1 (enable counting)
- Wake up when overflow (external clock asynchronous mode only)
- overflow interrupt
- Special event trigger function (with ECCP)
- Time base with capture/compare function

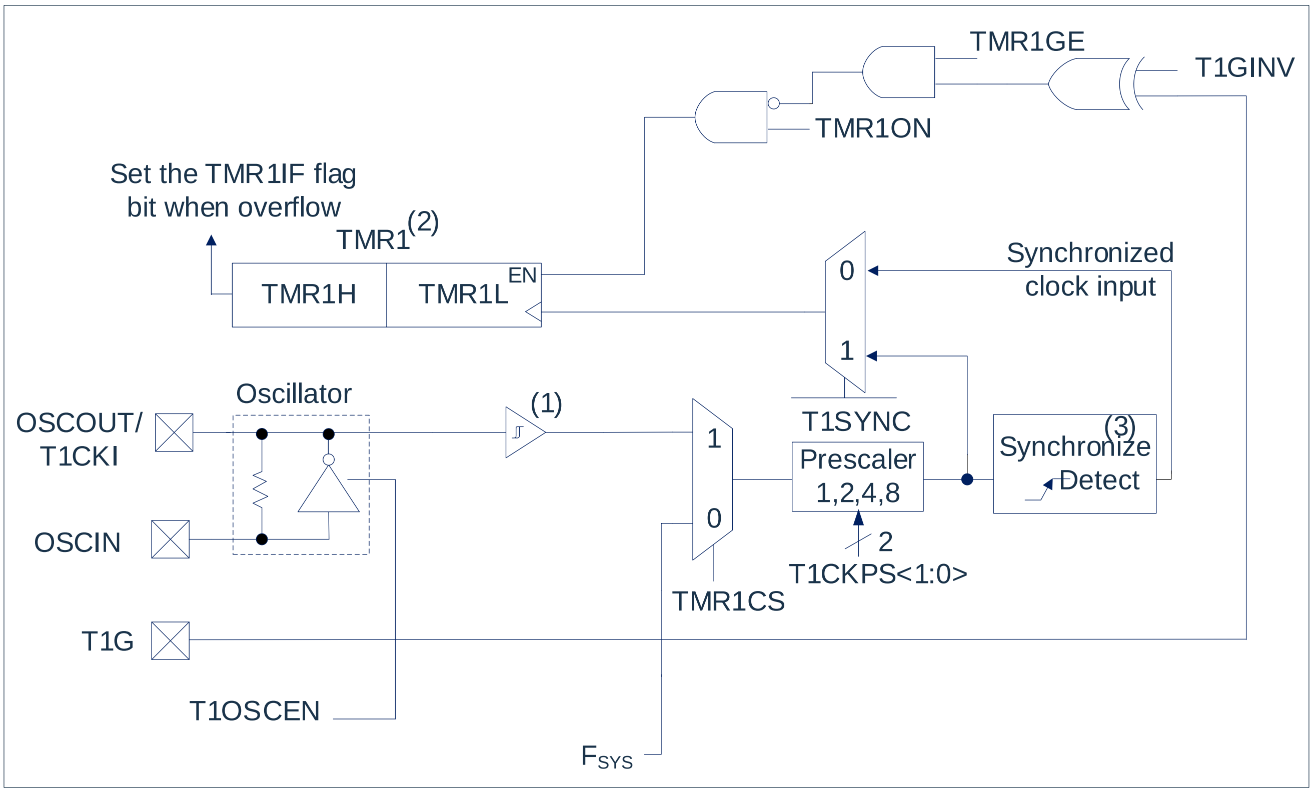


Fig 9-1: TIMER1 structure

Note:
1. The ST buffer is in low power mode when using the LP oscillator, but in high speed mode when using T1CKI.
2. The Timer1 register increments on the rising edge.
3. Do not perform synchronous during sleep.

9.2 Working Principle for TIMER1

TIMER1 mod is a 16-bit incremental counter accessed through a pair of register TMR1H: TMR1L. Writing to TMR1H or TMR1L can directly update the counter.

When used with internal clock source, this mod can be used as a counter. When used with external clock source, this mod can be used as a timer or counter.

9.3 Clock Source Selection

The TMR1CS bit of the T1CON register is used to select the clock source. When TMR1CS=0, the frequency of the clock source is FSYS. When TMR1CS=1, the clock source is provided by external.

clock source	TMR1CS
F_{SYS}	0
T1CKIpin	1

9.3.1 Internal Clock Source

After selecting the internal clock source, the TMR1H:TMR1L register will increase in frequency with a multiple of FSYS. The specific multiple is determined by the TIMER1 pre-scaler.

9.3.2 External Clock Source

After selecting the external clock source, TIMER1mod can be used as a timer or counter.

When counting, TIMER1 is incremented on the rising edge of external clock inputT1CKI. In addition, the clock in counter mode can be synchronous or asynchronous with the microcontroller system clock.

If you need an external clock oscillator, TIMER1 can use LP oscillator as clock source.

In counter mode, when one or more of the following conditions occur, a falling edge must be passed before the counter can count up for the first time on the subsequent rising edge (see Figure 9-2):

- Enable TIMER1.
- A write operation was performed on TMR1H or TMR1L.
- When TIMER1 is disabled, T1CKI is high; when TIMER1 is re-enabled, T1CKI is low.

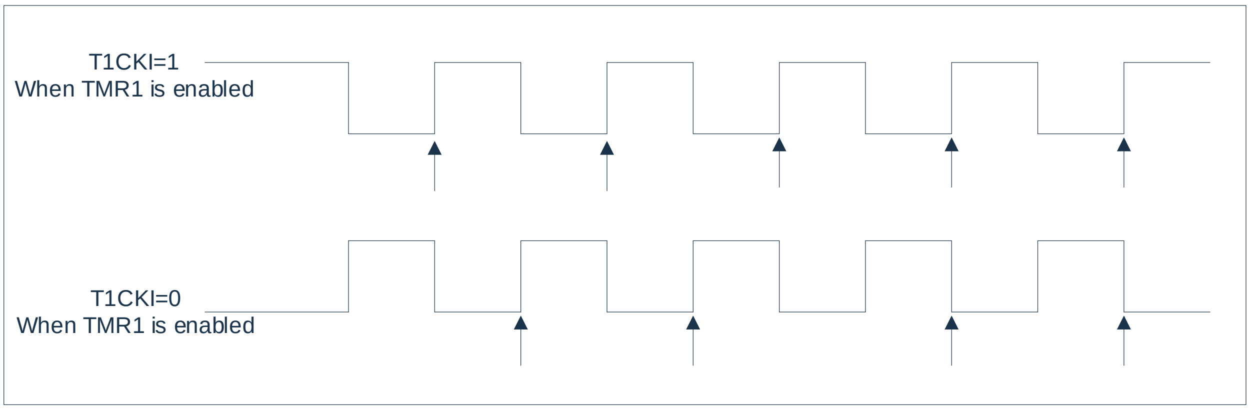


Fig 9-2：incremental edge of TIMER1

Note:
1) The arrow indicates that the counter is incrementing.
2) In the counter mode, a falling edge must be passed before the counter can perform the first increment technique on the subsequent rising edge.

9.4 TIMER1 Pre-scaler

TIMER1 has four selections of prescaler ratios, allowing the input clock to be divided by 1, 2, 4 or 8. The T1CKPS bit of the T1CON register controls the prescaler counter. The prescaler counter cannot be directly read or written; but, the prescaler counter can be cleared by writing to TMR1H or TMR1L.

9.5 TIMER1 Oscillator

A built-in low-power 32.768KHz oscillator is connected between the T1OSI (input) pin and T1OSO (amplifier output) pin. Set the T1OSCEN control bit of the T1CON register to 1 to enable the oscillator. This oscillator will be in sleep mode Continue to run, but TIMER1 must be selected as the asynchronous counting mode.

The TIMER1 oscillator is exactly the same as the LP oscillator. The user must provide a software delay to ensure the normal oscillation of the oscillator.

When the TIMER1 oscillator is enabled, PORTB5 and PORTB are set as analog inputs.

Note: The oscillator can be used after a period of start-up and stabilization time. Therefore, before enabling TIMER1, set T1OSCEN to 1 and pass an appropriate delay.

9.6 TIMER1 Working Principle Under Aasynchronous Counter Mode

If the control bit T1SYNC in the T1CON register is set to 1, the external clock input will not be synchronous. The timer continues to count up asynchronously with the internal phase clock. The timer will continue to run in the sleep state, and will generate an interrupt during overflow, thereby waking up Processor. However, you should be especially careful when using software to read/write timers (see Section 9.6.1 "Read and Write to TIMER1 in Asynchronous Counter Mode").

Note:
1) When switching from synchronous operation to asynchronous operation, an increment may be missed.
2) When switching from asynchronous operation to synchronous operation, a false increment may occur.

9.6.1 Read and Write Operations to TIMER1 In Asynchronous Counter Mode

When the timer uses an external asynchronous clock to work, the read operation of TMR1H or TMR1L will ensure that it is valid (the hardware is responsible). But users should keep in mind that reading two 8-bit values to read a 16-bit timer has its own problems. This is because the timer may overflow between two read operations.

For write operations, it is recommended that the user stop the timer before writing the required value. When the register is counting up, writing data to the timer register may cause write contention. This will cause unpredictability in the register pair TMR1H:TMR1L Value.

9.7 TIMER1 Gate Control

Software can configure the TIMER1 gate control signal source as T1G pin, which allows the device to directly use T1G to time external events.

Note: The TMR1GE bit of the T1CON register must be set to 1 to use the gate control signal of TIMER1.

You can use the T1GINV bit of the T1CON register to set the polarity of the TIMER1gate control signal. The gate control signal can come from T1Gpin. This bit can configure TIMER1 to time the high-level time or low-level time between events.

9.8 TIMER1 Interrupt

After a pair of TIMER1 registers (TMR1H:TMR1L) count up to FFFFH, the overflow returns to 0000H. When TIMER1 overflows, the TIMER1 interrupt flag bit of the PIR1 register is set to 1. To allow the overflow interrupt, the user should set the following bit to 1:

- ◆ TIMER1 interrupt enable bit in PIE1 register;
- ◆ PEIE bit in INTCON register;
- ◆ GIE bit in INTCON register.

Clear the TMR1IF bit in the interrupt service program to clear the interrupt.

Note: Before allowing the interrupt again, the register pair TMR1H:TMR1L and the TMR1IF bit should be cleared.

9.9 TIMER1 Working Principle During Sleep

TIMER1 can work in sleep mode only when it is set to asynchronous counter mode. In this mode, the external crystal or clock source can be used to make the counter count up. The timer can wake up the device through the following settings:

- ◆ The TMR1ON bit in the T1CON register must be set to 1;
- ◆ The TMR1IE bit in the PIE1 register must be set to 1;
- ◆ The PEIE bit in the INTCON register must be set to 1.

The device will be woken up at overflow and execute the next instruction. If the GIE bit in the INTCON register is 1, the device will call the interrupt service routine (0004h).

9.10 ECCP Capture/Compare Time Base

The ECCP mod uses the TMR1H:TMR1L pair of registers as the time base for working in capture or compare mode.

- -In capture mode, the value of the register pair TMR1H:TMR1L is copied to the register pair CCPRxH:CCPRxL when a configuration event occurs.
- -In compare mode, when the value in the CCPRxH:CCPRxL pair of registers matches the value in the TMR1H:TMR1L pair of registers, an event will be triggered. This event can be used to trigger special events.

For more information, please refer to the "capture/compare/PWM mod (CCP1 and CCP" chapter).

9.11 ECCP Special Events Flip-flop

If ECCP is configured to trigger a special event, flip-flop will clear the MR1H:TMR1L pair of registers. This special event will not cause TIMER1interrupt. You can still configure ECCPmod to generate an ECCP interrupt.

In this working mode, the CCPRxH: CCPRxL pair register actually becomes the period register of TIMER1.

To use special event flip-flop, you should make TIMER1 and FSYS synchronous.TIMER1 working in asynchronous mode can cause loss of special event trigger signal.

When the operation of writing to TMR1H or TMR1L occurs at the same time as the signal triggered by a special event from ECCP, the writing operation has priority.

For more information, please refer to the "capture/compare/PWMmod (CCP1 and CCP" chapter).

9.12 TIMER1 Control Register

TIMER1control register T1CON (10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
reset value	0	0	0	0	0	0	0	0

Bit7	T1GINV:	TIMER1 gate control signal polarity bit;
	1=	TIMER1 gate control signal is active high (TIMER1 counts when the gate control signal is high level);
	0=	The TIMER1 gate control signal is active low (TIMER1 counts when the gate control signal is low).
Bit6	TMR1GE:	TIMER1 gate control enable bit.
		If TMR1ON=0，ignore this bit.
		If TMR1ON=1: 1= TIMER1 counting is controlled by TIMER1gate control function;
		0=TIMER1always counts.
Bit5~Bit4	T1CKPS<1:0>:	TIMER1 input clock frequency ratio selection bit;
	11=	1:8;
	10=	1:4;
	01=	1:2;
	00=	1:1.
Bit3	T1OSCEN:	LP oscillator enable controlbit;
	1=	Enable LP oscillatoras the clock source of TIMER1;
	0=	Disable LP oscillator.
Bit2	T1SYNC:	TIMER1 external clock input synchronous control bit.
		TMR1CS=1: 1= not synchronous with externalclockinput;
		0= synchronous with external clock input.
		TMR1CS=0：ignore this bit，TIMER1 uses internal clock.
Bit1	TMR1CS:	TIMER1 clock source selection bit；
	1=	From LP oscillator clock source or clock source from T1CKI pin (rising edge trigger);
	0=	Internal clock source F_{SYS}.
Bit0	TMR1ON:	TIMER1enable bit;
	1=	Enable TIMER1;
	0=	Disable TIMER1.

10. TIMER2

10.1 TIMER2 General

TIMER2 mod is an 8-bit timer/counter with the following characteristics:

- 8-bit timer register (TMR2);
- 8-bit period register (PR2);
- Interrupt when TMR2 matches PR2;
- Software programmable prescaler ratio (1:1, 1:4 and 1:16);
- Software programmable postscaler ratio (1:1 to 1:16).

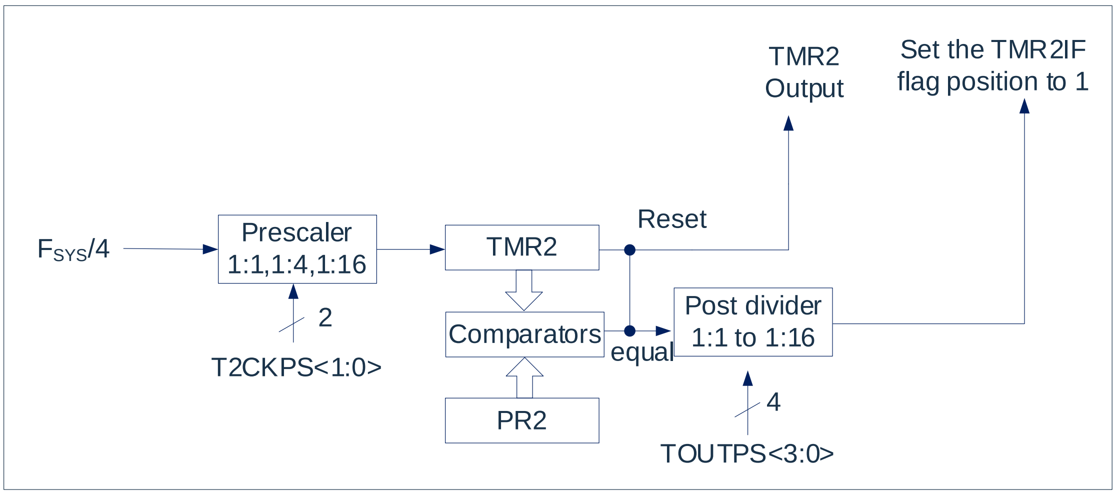


Fig 10-1: TIMER2 structure

10.2 Working Principle of TIMER2

The input clock of the TIMER2 mod is the system instruction clock (FSYS/4). The clock is input to the TIMER2 pre-scaler. There are several division ratios to choose from: 1:1, 1:4 or 1:16. pre-scaler the output is then used to increment TMR2register.

Continue to compare the values of TMR2 and PR2 to determine when they match. TMR2 will increase from 00h until it matches the value in PR2. When a match occurs, the following two events will occur:

- TMR2 is reset to 00h in the next increment period;
- TIMER2 post-scaler increments.

The matching output of the TIMER2 and PR2 comparator is then input to the post-scaler of TIMER2. The post-scaler has a prescaler ratio of 1:1 to 1:16 to choose from. The output of the TIMER2 post-scaler is used to make PIR1 The TMR2IF interrupt flag bit of the register is set to 1.

Both TMR2 and PR2 registers can be read and written. At any reset, TMR2 register is set to 00h and PR2 register is set to FFh.

Enable TIMER2 by setting the TMR2ON bit of the T2CON register; disable TIMER2 by clearing the TMR2ON bit.

The TIMER2 pre-scaler is controlled by the T2CKPS bit of the T2CON register; the TIMER2 postscaler is controlled by the TOUTPS bit of the T2CON register.

The pre-scaler and postscaler counters are cleared under the following conditions:

- When TMR2ON=0
- Any device reset occurs (power-on reset, watchdog timer reset, or undervoltage reset).

Note: Writing T2CON will not clear TMR2. When TMR2ON=0, the TMR2 register cannot be written.

10.3 TIMER2 related register

There are two registers related to TIMER2, namely data memory TMR2 and control register T2CON.

TIMER2 data register TMR2 (11H)

11H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

TIMER2 control register T2CON (12H)

12H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
Read write	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	----	0	0	0	0	0	0	0

Bit7	Not used, read 0.	
Bit6~Bit3	TOUTPS<3:0>:	TIMER2 output frequency division ratio selection bit.
	0000=	1:1;
	0001=	1:2;
	0010=	1:3;
	0011=	1:4;
	0100=	1:5;
	0101=	1:6;
	0110=	1:7;
	0111=	1:8;
	1000=	1:9;
	1001=	1:10;
	1010=	1:11;
	1011=	1:12;
	1100=	1:13;
	1101=	1:14;
	1110=	1:15;
	1111=	1:16.
Bit2	TMR2ON:	TIMER2 enable bit;
	1=	Enable TIMER2;
	0=	Disable TIMER2.
Bit1~Bit0	T2CKPS<1:0>:	TIMER2 clock frequency division ratio selection bit;
	00=	1;
	01=	4;
	1x=	16.

11. Analog to Digital Conversion (ADC)

11.1 ADC general

The analog-to-digital converter (ADC) can convert the analog input signal into a 12-bit binary number that represents the signal. The analog input channels used by the device share a sample and hold circuit. The output of the sample and hold circuit is connected to the input of the analog to digital converter. The analog-to-digital converter uses the successive approximation method to generate a 12-bit binary result, and save the result in the ADC result register (ADRESL and ADRESH).

ADC reference voltage is always generated internally. ADC can generate an interrupt after conversion is completed.

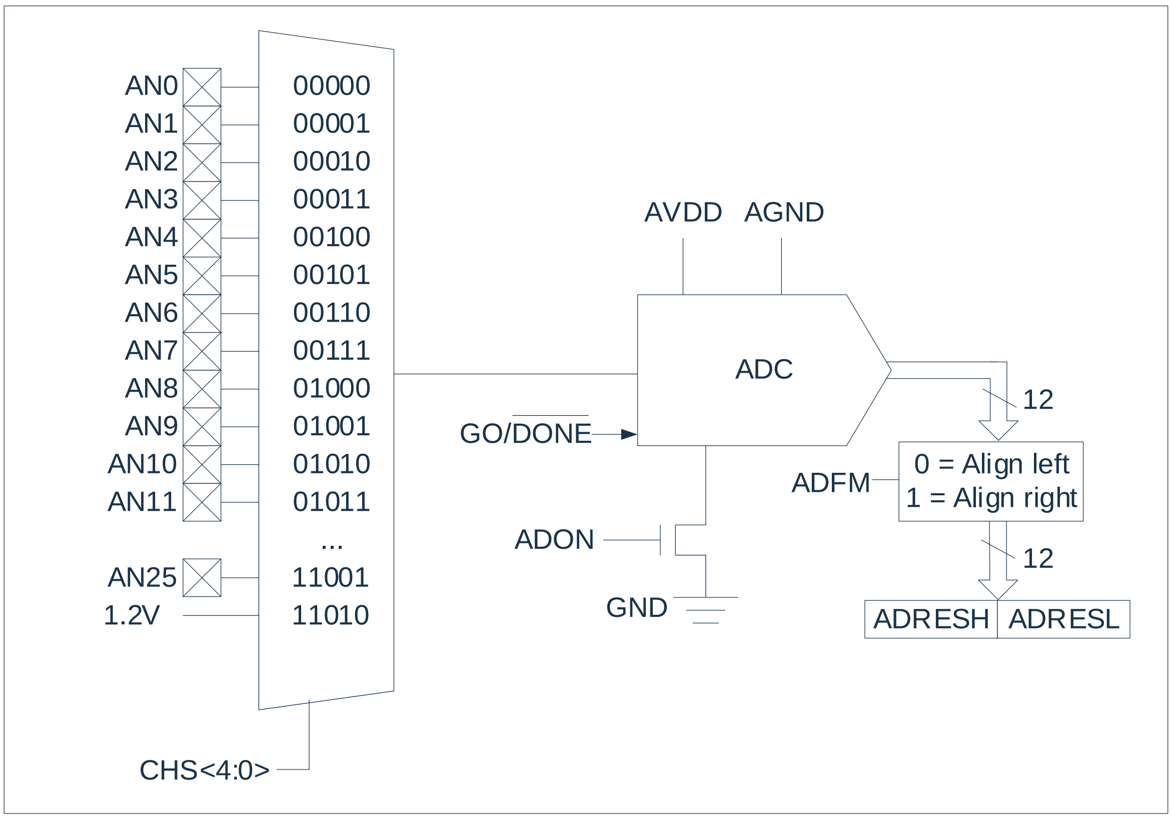


Fig 11-1: ADC structure

11.2 ADC configuration

When configuring and using ADC, the following factors must be considered:

- Port configuration;
- Channel selection;
- ADC conversion clock source;
- Interrupt control;
- The storage format of the result.

11.2.1 Port configuration

ADC can convert both analog signal and digital signal. When converting analog signal, the I/O pin should be configured as analog input pin by setting the corresponding TRIS bit to 1. For more information, please refer to the corresponding port chapter.

Note: Applying analog voltage to pins defined as digital inputs may cause overcurrent in the input buffer.

11.2.2 Channel selection

The CHS bit of the ADCON0 register determines which channel is connected to the sample and hold circuit.

If the channel is changed, a certain delay will be required before the next conversion starts. For more information, please refer to the "ADC working principle" chapter.

11.2.3 ADC reference voltage

The ADC reference voltage is always provided by the chip's VDD and GND.

11.2.4 Converter clock

The ADCS bit of the ADCON0 register can be set by software to select the clock source for conversion. There are 4 possible clock frequencies to choose from:

- $F_{SYS}/8$
- $F_{SYS}/16$
- $F_{SYS}/32$
- F_{RC} (special internal oscillator)

The time to complete one-bit conversion is defined as TAD. A complete 12-bit conversion requires 49 TAD periods.

Must comply with the corresponding TAD specification to get the correct conversion result. The following table is an example of correct selection of ADC clock.

Note: Unless FRC is used, any change in the system clock frequency will change the ADC clock frequency, which will negatively affect the ADC conversion results.

For different reference voltages and different VDDs, you need to refer to the following table to set a reasonable frequency division.

Reference voltage	Working voltage (V)	Fastest division setting		Conversion time (us)
		F_{SYS} = 16MHz	F_{SYS} = 8MHz	
VDD	4.0~5.5	$F_{SYS}/16$	$F_{SYS}/8$	49
VDD	2.7~4.0	$F_{SYS}/32$	$F_{SYS}/16$	98

11.2.5 ADC Interrupt

ADC mod allows an interrupt to be generated after the completion of the analog-to-digital conversion. The ADC interrupt flag bit is the ADIF bit in PIR1register. The ADC interrupt enable bit is the ADIE bit in PIE1register. The ADIF bit must be cleared by software. The ADIF bit after each conversion is completed Will be set to 1, regardless of whether ADC interrupt is allowed.

11.2.6 Output Formatting

The result of 12-bit A/D conversion can be in two formats: left-justified or right-justified. The output format is controlled by the ADFM bit in ADCON1register.

When ADFM=0, the AD conversion result is left aligned and the AD conversion result is 12Bit; when ADFM=1, the AD conversion result is right aligned, and the AD conversion result is 10 Bit.

11.3 ADC working principle

11.3.1 Start conversion

To enable ADC mod, you must set the ADON bit of the ADCON0 register to 1, and set the GO/ ("DONE")¯ bit of the ADCON0 register to 1 to start analog-to-digital conversion.

> Note: It is not possible to set GO/$\overline{\text{DONE}}$ position to 1 with the same instructions that open A/Dmod.

11.3.2 Complete conversion

When the conversion is complete, the ADC mod will:

- Clear the GO/$\overline{\text{DONE}}$ bit;
- Set ADIF flag bit to 1;
- Update the ADRESH: ADRESL register with the new conversion result.

11.3.3 Stop conversion

If you must terminate the conversion before conversion is completed, you can use software to clear the GO/$\overline{\text{DONE}}$bit. The ADRESH: ADRESL register will not be updated with the uncompleted analog-to-digital conversion result. Therefore, the ADRESH: ADRESL register will remain on the value obtained by the second conversion. In addition, after the A/D conversion is terminated, a delay of 2 TAD must be passed before the next acquisition can be started. After the delay, the input signal of the selected channel will automatically start to be collected.

> Note: Device reset will force all registers to enter the reset state. Therefore, reset will close the ADC mod and terminate any pending conversions.

11.3.4 Working principle of ADC in sleep mode

ADC mod can work in sleep mode. This operation requires ADC clock source to be set to FRC option. If FRC clock source is selected, ADC must wait for one more instruction speriod before starting conversion. This allows the execution of STOP instructions to reduce conversion If the ADC interrupt is allowed, the device will wake up from sleep mode when the conversion ends. If the ADC interrupt is disabled, even if the ADON bit remains set, the ADC mod will be closed after the conversion is complete. If the ADC clock source is not FRC, even if the ADON bit remains set, executing the STOP instructions will abort the current conversion and close the A/D mod.

11.3.5 A/D conversion procedure

The following steps give an example of using ADC for analog-to-digital conversion:

1. port configuration:
   - Configure pin as input pin (see TRIS register).
2. configuration ADC mod:
   - Select ADC conversion clock;
   - Select ADC input channel;
   - Choose the format of the result;
   - Start the ADC mod.
3. configuration ADC interrupt (optional):
   - Clear ADC interrupt flag bit;
   - Allow ADC interrupt;
   - Allow peripherals interrupt;
   - Allow global interrupt.
4. Wait for the required acquisition time.
5. Set GO/$\overline{\text{DONE}}$ to 1 to start conversion.
6. Wait for the ADC conversion to end by one of the following methods:
   - Query GO/ ("$\overline{\text{DONE}}$") bit
   - Wait for ADC interrupt (allow interrupt).
7. Read ADC results.
8. Clear the ADC interrupt flag bit (if interrupt is allowed, this operation is required).

Note: If the user tries to resume sequential code execution after waking the device from sleep mode, the global interrupt must be disabled.

example：AD conversion

```
            LDIA        B'10000000'
            LD          ADCON1,A
            SETB        TRISA,0             ;set PORTA.0 as input
            LDIA        B'11000001'
            LD          ADCON0,A
            CALL        DELAY               ;delay
            SETB        ADCON0,GO
            SZB         ADCON0,GO           ;wait ADCto complete
            JP          $-1
            LD          A,ADRESH            ;save the highest bit of ADC
            LD          RESULTH,A
            LD          A,ADRESL            ; save the lowest bit of ADC
            LD          RESULTL,A
```

11.4 ADC Related Register

There are mainly 4 RAMs related to AD conversion, namely control register ADCON0 and ADCON1, data register ADRESH and ADRESL.

AD control register ADCON0 (9EH)

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit6	ADCS<1:0>:	A/D conversion clock selection bit.
	00=	$F_{SYS}/8$
	01=	$F_{SYS}/16$
	10=	$F_{SYS}/32$
	11=	F_{RC} (A dedicated internal oscillator generates a clock with a frequency of up to 32KHz)
Bit5~Bit2	CHS<3:0>:	The lower four bits of the analog channel selection bit and CHS4 form a five-bit channel selection.
	CHS<4:0>:	
	00000=	AN0
	00001=	AN1
	00010=	AN2
	00011=	AN3
	00100=	AN4
	00101=	AN5
	00110=	AN6
	00111=	AN7
	01000=	AN8
	01001=	AN9
	01010=	AN10
	01011=	AN11
	…	
	10111=	AN23
	11000=	AN24
	11001=	AN25
	11010=	1.2V (fixed reference voltage)
Bit1	GO/$\overline{\text{DONE}}$:	A/D conversion status bit.
	1=	A/D conversion is in progress. Set this bit to 1 to start A/D conversion. When A/D conversion is completed, this bit is automatically cleared by hardware.
	0=	A/D conversion complete or not in progress.
Bit0	ADON:	ADC enable bit.
	1=	Enable ADC;
	0=	Disable ADC, not consuming current.

AD data register high bit ADCON1 (9FH)

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	CHS4	----	----	----	----	----	----
Read/write	R/W	R/W	----	----	----	----	----	----
Reset value	0	0	----	----	----	----	----	----

Bit7	ADFM:	A/D conversion result format selection bit
	1=	Right alignment
	0=	left alignment
Bit6	CHS4:	Channel selection bit
Bit5~Bit0	Not used	

AD data register high bit ADRESH (9DH)，ADFM=0

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
read/write	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0 ADRES<11:4>: ADC result register bit.

The higher 8 bits of the 12-bit conversion result.

AD data register lower bit ADRESL (9CH)，ADFM=0

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES3	ADRES2	ADRES1	ADRES0	----	----	----	----
read/write	R	R	R	R	----	----	----	----
Reset value	X	X	X	X	----	----	----	----

Bit7~Bit4 ADRES<3:0>: ADC result register bit.

The lower 4 bits of the 12-bit conversion result.

Bit3~Bit0 Not used

AD data registerhigh bit ADRESH (9DH)，ADFM=1

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADRES11	ADRES10
read/write	----	----	----	----	----	----	R	R
Reset value	----	----	----	----	----	----	X	X

Bit7~Bit2 Not used.

Bit1~Bit0 ADRES<11:10>: ADC result register bit.

The higher 2 bits of the 12-bit conversion result.

AD data register lower bits ADRESL (9CH)， ADFM=1

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
read/write	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0	ADRES<9:2>:	ADC result register bit.
		The 2-9 bits of the 12-bit conversion result.

Note: In the case of ADFM=1, the AD conversion result only saves the upper 10 bits of the 12-bit result, where ADRESH saves the upper 2 bits, and ADRESL saves the 2nd to 9th digits.

12. LCD/LED Driver Mod

Chip built-in LCD/LED driver mod, they share the control register.

The LED driver mod can support up to driver 11*8 digital tubes. The program only needs to set the relevant control bits and display data, and the chip pins will automatically output the driver LED waveform (hardware driver).

The LCD driver mod can be driver 1/2 Bias LCD, and the waveform of the output driver LCD must be controlled by a program (software driver).

12.1 LCD/LED Function Enable

Set the 7th bit LCDEN of LEDCON0 (113H) to 1, and clear the 6th bit LEDEN to 0 to enable the LCD driver function;

Set the 6th bit LEDEN of LED CON0 (113H) to 1, and clear the 7th bit LCDEN to 0 to enable the LED driver function;

Set both LCDEN and LEDEN to 0, turn off the LCD/LED mod.

Note: Please do not set LCDEN and LEDEN to 1 at the same time.

12.2 LCD/LED Pin Function Configuration

If the LCD driver mod is enabled and the COM port function is enabled, the corresponding I/O port will be forced to be an analog input state, regardless of the state of the corresponding TRIS bit.

If the LED driver function is enabled, the corresponding SEG port and COM port must be set to output state, and output "0", that is, the corresponding TRIS bit and PORT position are "0".

12.3 LCD/LED COM Port Configuration

The setting method of LED COM port is as follows:

Set the I/O port direction and data register. The LCD function sets the corresponding pin to input state, and the LED function sets the corresponding pin to output state and output low level.

COMSEL [2:0]	No. of COM port (only LED)
000	4
001	5
010	6
011	8
100	2
101	3
110	3
111	4

Set the COM_EN register, and set the corresponding pin as the COM port of the LED function.

If the user's COM ports are not arranged in order during use. For example, if COM3-COM6 is used as the COM port for LED function, and COM0-COM2 is used as ordinary I/O port, you can do the following settings:

- Set the number of COM ports to 8 COM, COMSEL="011";
- Set COM3-COM6 of the COM_EN register to 1, COM0-COM2, COM7 to 0.

At this time, COM3-COM6 is the COM port of LED function, and its output duty cycle is 1/8. And COM0-COM2, COM7 can be used as ordinary I/O ports.

12.4 LCD/LED SEG Port Configuration

The SEG port that enables the LCD/LED function must meet the following conditions:

1. Set the status of the corresponding pin, the LED function sets the corresponding pin to the output state and output "0";
2. Set the corresponding pins in SEGEN0, SEGEN1, and SEGEN2register as LED driver functions;
3. Set SEG port output current in SEGEN2 register (LED function only).

12.5 LED Data Configuration

Setting the LED display data requires the following steps:

1. Set the SEGOUT [1:0] bits of the LEDCON1 register to "1x";
2. Set the 7th bit of LEDADD register LEDCS=1 to allow read/write data;
3. Set the data address 0-6 of LEDADD;
4. Set LEDDATA data (the pin is not used for LED function, and the corresponding LEDDATA bit needs to be set to "0");
5. Repeat steps 3-4 to set other address data;
6. Turn off the data read/write bit LEDCS=0 after setting.

Relationship of LED address and data:

LEDADD	LEDDATA								
00H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG0
01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG1
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG22
17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG23
	COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0	

12.6 LCD/LED Related Register

LCD/LED driver related register: control register LEDCON0、LEDCON1; address register LEDADD; data register LEDDATA; register COMEN、SEGEN0、SEGEN1、SEGEN2.LCD/LED control register LEDCON0 (113H)

113H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LEDCON0	LCDEN	LEDEN	COMSEL [1:0]		LEDCLK [3:0]			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7	LCDEN:	LCD mod enable bit;	
	0:	Disable LCD mod;	
	1:	Enable LCD mod.	
Bit6	LEDEN:	LED mod enable bit;	
	0:	Disable LEDmod;	
	1:	Enable LEDmod.	
Bit5~Bit4	COMSEL [1:0]:	Number of LED mod COM port selection (COMSEL[2] in LEDADD register 5th bit);	
	000:	4COM;	
	001:	5COM;	
	010:	6COM;	
	011:	8COM;	
	100:	2COM;	
	101:	3COM;	
	110:	3COM;	
	111:	4COM.	
Bit3~Bit0	LEDCLK [3:0]:	LED frequency selection (Select the clock source according to the 6th LEDF of LEDCON1);	
		LEDF=0	LEDF=1
	0000:	F_{HSI} /64	not used;
	0001:	F_{HSI} /128	not used;
	0010:	F_{HSI} /256	not used;
	0011:	F_{HSI} /512	External 32.768KHz oscillation/2;
	0100:	F_{HSI} /1024	External 32.768KHz oscillation/4;
	0101:	F_{HSI} /2048	External 32.768KHz oscillation/8;
	0110:	F_{HSI} /4096	External 32.768KHz oscillation/16;
	0111:	F_{HSI} /8192	External 32.768KHz oscillation/32;
	1x00:	F_{HSI} /16384	External 32.768KHz oscillation/64;
	1x01:	F_{HSI} /32768	External 32.768KHz oscillation/128;
	1x10:	F_{HSI} /65536	External 32.768KHz oscillation/256;
	1x11:	F_{HSI} /131072	External 32.768KHz oscillation/512.

LCD control register LEDCON1 (114H)

114H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LEDCON1	----	LEDF	SEGOUT[1:0]		----		LCDISEL[1:0]	
R/W	----	R/W	R/W	R/W	----	----	R/W	R/W
Reset value	----	0	0	0	----	----	0	0

Bit7	not used.	
Bit6	LEDF:	LED clock source selection;
	0:	Internal clock;
	1:	external32.768KHz oscillation clock. (Need to connect a 32.768KHz crystal oscillator to the OSCI/OSCO pin of the chip)
Bit5~Bit4	SEGOUT:	SEG output mode selection;
	00:	SEG output all 0;
	01:	SEG output all 1;
	1x:	SEG output as data from LEDDATA.
Bit3~Bit2	not used.	
Bit1~Bit0	LCDISEL[1:0]:	LCD output current selection bit; (only for LCD)
	00=	100Ua@5V;
	01=	200Ua@5V;
	10=	400Ua@5V;
	11=	800Ua@5V.

LED address register LEDADD (115H)

115H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LEDADD	LEDCS	----	COMSEL[2]	LEDADD[4:0]				
R/W	R/W	----	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	----	0	0	0	0	0	0

Bit7	LEDCS:	LED data read/write enable bit;
	0:	Disable read/write LED data;
	1:	Enable read/write LED data.
Bit6	not used	
Bit5	COMSEL[2]:	COMselect the highest bit，COMSEL[2:0] set number of COM ports and Bias;
Bit4~Bit0	LEDADD[4:0]:	LED address selection; LED address range 00H-0AH

LED data register LEDDATA (116H)

116H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LEDDATA	LEDDATA[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
reset value	X	X	X	X	X	X	X	X

Bit7~Bit0	LEDDATA[7:0]:	LED data setting, write the data corresponding to the address of LEDADD.

LCD/LED COM port control register COMEN (11AH)

11AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
COMEN	COM7EN	COM6EN	COM5EN	COM4EN	COM3EN	COM2EN	COM1EN	COM0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	COMxEN:	COM port function configuration ;
	0:	COMx as normal I/O (x=0-7);
	1:	COMx as COM (x=0-7) ofLCD/LED

LED SEG port control register SEGEN0 (119H)

119H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN0	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	SEGxEN:	SEG port function configuration ;
	0:	SEGx as normal I/O (x=0-7);
	1:	SEGx as COM (x=0-7) ofLCD/LED

LED SEG port control register SEGEN1 (118H)

118H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN1	----	----	----	----	----	SEG10EN	SEG9EN	SEG8EN
R/W	----	----	----	----	----	R/W	R/W	R/W
Reset value	----	----	----	----	----	0	0	0

Bit7~Bit3	not used	
Bit2~Bit0	SEGxEN:	SEG port function configuration ;
	0:	SEGx as normal I/O (x=8-10);
	1:	SEGx as COM (x=8-10) ofLCD/LED

LED SEG port control register SEGEN2 (117H)

117H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN2	SEGDRI[3:0]				----	----	----	----
R/W	R/W	R/W	R/W	R/W	----	----	----	----
Reset value	0	0	0	0	----	----	----	----

Bit7~Bit4	SEGDRI[3:0]:	SEG port driver current configuration; (仅 LED 功能有效)
	0000:	SEG port driver current as 0;
	0001:	SEG port driver current as 2mA;
	0010:	SEG port driver current as 4mA;
	0011:	SEG port driver current as 6mA;
	…	
	1110:	SEG port driver current as 28mA;
	1111:	SEG port driver current as 30mA.
Bit3~Bit0	not used	

13. Capture/Compare/PWM Mod (CCP1 and CCP2)

The chip contains two capture/compare/PWM (CCP1) and (CCP2). The operations of CCP1 and CCP2 mod are basically the same.

Note: CCPRx and CCPx in this document refer to CCPR1 or CCPR2 and CCP1 or CCP2 respectively.

The capture/compare/PWM mod is peripherals that allow users to time and control different events. In capture mode, the peripherals can time the duration of the event. Capture mode allows the user to trigger an external event after the predetermined timing expires. PWM mode can generate pulse width modulation signal with variable frequency and duty cycle.

When CCP is used in capture/compare mode, timer TIMER1 is required.

CCPx control register CCPxCON

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CCPxCON	---	---	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
read/write	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6	not used.	
Bit5~Bit4	DC1B<1:0>:	Lower 2 bits of PWM duty cycle;
	capturemode:	not used;
	comparemode:	not used;
	PWM mode:	These two bits are the lower 2 bits of the 10-bit PWM duty cycle. The higher 8 bits of the duty cycle are in CCPR1L.
Bit3~Bit0	CCP1M<3:0>:	CCP mode selection bit;
	0000=	capture/compare/PWM off (reset ECCP mod);
	0001=	not used (stay);
	0010=	comparemode, output level turns when matching (CCP1IF set 1);
	0011=	not used (stay);
	0100=	Capture mode, Capture occurs on every falling edge;
	0101=	Capture mode, Capture occurs on every rising edge;
	0110=	Capture mode, Capture occurs for every 4 rising edges;
	0111=	Capture mode, Capture occurs for every 16 rising edges;
	1000=	Compare mode, Output is high when compare matches (CCP1IF set to 1);
	1001=	Compare mode, Output is low when compare matches (CCP1IF set to 1);
	1010=	Compare mode, generate interrupt when compare matches (CCP1IF set 1, CCP1 pin not affected);
	1011=	Compare mode, Trigger special events (CCP1IF bit set as 1, CCP1 reset TMR1 or TMR2);
	11xx=	PWM mode.

13.1 Capture Mode

In capture mode, when an event occurs on the corresponding CCPx pin, CCPRxH: CCPRxL is the 16-bit value of the register capture TMR1 register. The event that triggers the capture can be defined as one of the following four, and is determined by the CCPxM in the CCPxCON register<3:0> bit configuration:

- Every falling edge;
- Every rising edge;
- Every 4 rising edges;
- Every 16 rising edges.

The event type is selected by the mode selection bit CCPxM3:CCPxM0 (CCPxCON<3:0>). When a capture occurs, the interrupt request flag bit CCPxIF in the PIRx register is set to 1; it must be cleared by software. If another capture occurs before the value in the register CCPRxH and CCPRxL is read, then the previous capture value will be overwritten by the new capture value (see Figure 13-1).

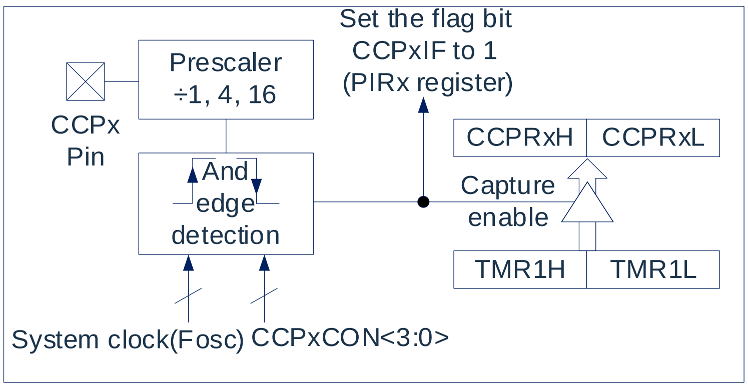


Fig 13-1: capture mode working structure

13.1.1 CCP Pin Configuration

In capture mode, the corresponding CCPx pin should be configured as input by setting the corresponding TRIS control bit to 1.

Note: If the CCPx pin is configured as output, a write operation to the port may trigger a capture event.

13.1.2 TIMER1 Mode Selection

TIMER1 must run in timer mode or synchronous counter mode CCP mod to use the capture function. Capture operation cannot be performed in asynchronous counter mode.

13.1.3 Software interrupt

When the capture mode is changed, a false capture interrupt may occur. The user should keep the CCPxIE interrupt enable bit in the PIEx register cleared to avoid false interrupts. The interrupt flag bit CCPxIF in the PIRx register should also be cleared after any change in the operation mode.

13.1.4 CCP pre-scaler

The CCPxM<3:0> bits in the CCPxCON register specify four pre-scaler settings. Whenever the CCP mod is closed or the capture mode is disabled, the pre-scaler counter will be cleared. This means that any reset will clear the pre-scaler. Divide counter.

Switching from one capture prescaler ratio to another capture prescaler ratio will not clear the prescaler counter, but may cause false interrupts. To avoid this unexpected operation, you should change the prescaler Turn off the mod by clearing the CCPxCON register before frequency ratio.

Change the capture prescaler ratio

```
CLR      CCP1CON       ; turn off CCP1
LDIA     B'00000101'
LD       CCP1CON,A     ; pass new value to CCP1
```

13.2 Compare Mode

In compare mode, the value of the 16-bit CCPRx register will be constantly compared with the value of a pair of TMR1 registers. When the two match, the following situations may occur in the CCPx mod:

- CCPx output high level;
- CCPx output low level;
- Generate special events to trigger signal;
- Generate software interrupt

The action on the pin depends on the value of the CCPxM<3:0> control bits in the CCPxCON register, and all capture modes will generate interrupt.

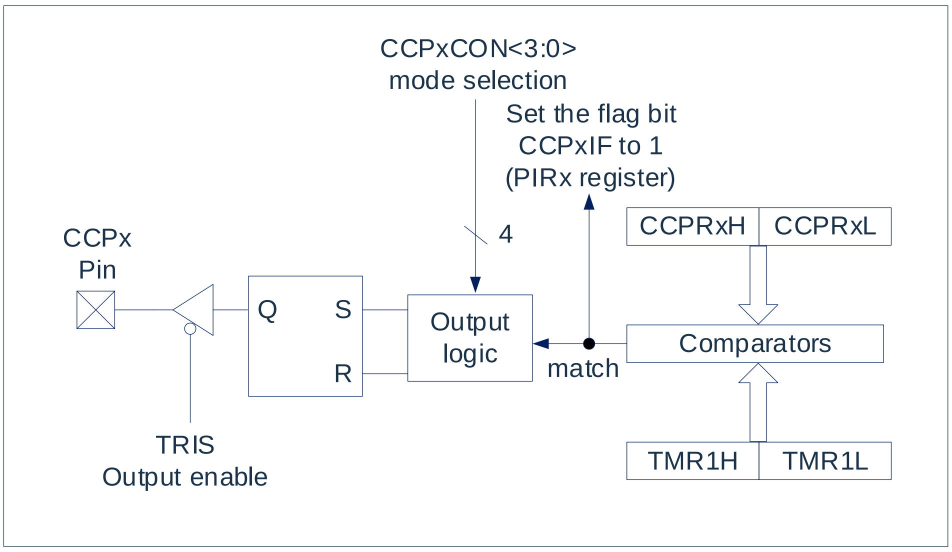


Fig 13-2: compare mode working structure

Special events trigger signal will:
- Clear the TMR1H and TMR1L registers.
- Will not set the TMR1IF flag bit in the PIR1 register.
- Make GO/ ("$\overline{\text{DONE}}$") bit 1 to start ADC conversion.

13.2.1 CCP Pin Configuration

The user must configure the CCPx pin to output by clearing the corresponding TRIS bit.

Note: Clearing the CCPxCON register will force the CCPx compare output latch to the default low level, which is not a port I/O data latch.

13.2.2 TIMER1 Mode Selection

In compare mode, TIMER1 must run in timer mode or synchronous counter mode. In asynchronous counter mode, compare operations may not be possible.

13.2.3 Software Interrupt Mode

When the software interrupt mode is selected (CCPxM<3:0>=1010), the CCPx mod will not control the CCPx pin (see CCPxCON register).

13.2.4 Special Events Trigger Signal

When the special event trigger mode (CCPxM<3:0>=1011) is selected, the CCPx mod will complete the following operations:

- reset TIMER1;
- If ADC is enabled, ADC conversion will also be started.

In this mode, the CCPx mod does not control the CCPx pin (see CCPxCONregister).

When the TMR1H/TMR1L register pair matches the CCPRxH/CCPRxL register pair, the CCP will immediately generate a special event trigger output. The TMR1H/TMR1L register pair will not reset until the next rising edge of the TIMER1 clock. Therefore, the CCPRxH/CCPRxL register pair actually become a 16-bit programmable period register of TIMER1.

Note:
1) The signal triggered by a special event from the CCP mod will not set the TMRxIF interrupt flag bit in the PIR1 register to be 1.
2) Change the contents of the CCPRxH and CCPRxL register pair between the edge that triggers the signal when a special event is generated and the clock edge that causes TIMER1 reset to clear the matching condition and prevent reset from occurring.

13.3 PWM Mode

PWM mode generates pulse width modulation signal on CCPxpin. Both PWM1 and PWM2 have their own independent period counters. The duty cycle, period and resolution are determined by the following registers:

- PWMCON
- PWMxCYC
- CCPRxL
- CCPxCON

PWM control register PWMCON (99H)

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON	----	CYC2EN	CK2[1:0]		----	CYC1EN	CK1[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7	not used.		
Bit6	CYC2EN:	PWM2 period counter enable bi.	
	1=	enable.	
	0=	disable.	
Bit5-Bit4	CK2[1:0]:	PWM2 period counter clock prescale selection bit.	
	00=	Prescale 1.	
	01=	Prescale 4.	
	1X=	Prescale 16.	
Bit3	not used.		
Bit2	CYC1EN:	PWM1 period counter enable bit.	
	1=	enable.	
	0=	disable.	
Bit1~Bit0	CK1[1:0]:	PWM1 period counter clock prescale selection bit.	
	00=	Prescale 1.	
	01=	Prescale 4.	
	1X=	Prescale 16.	

PWM1 period data register PWM1CYC (9AH)

9AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1CYC								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

PWM2 period data register PWM2CYC (9BH)

9BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM2CYC								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

In pulse width modulation (PWM) mode, CCP mod can output a PWM signal with a resolution of up to 10 bits on the CCPx pin. Since the CCPx pin is multiplexed with the port data latch, the corresponding TRIS bit must be cleared to enable CCPx the output driver of the pin.

Note: Clearing the CCPxCON register will give up CCPx's control over the CCPx pin.

The following Figure 13-3 is a simplified block diagram of PWM operation, and Figure 13-4 is a typical waveform of the PWM signal.

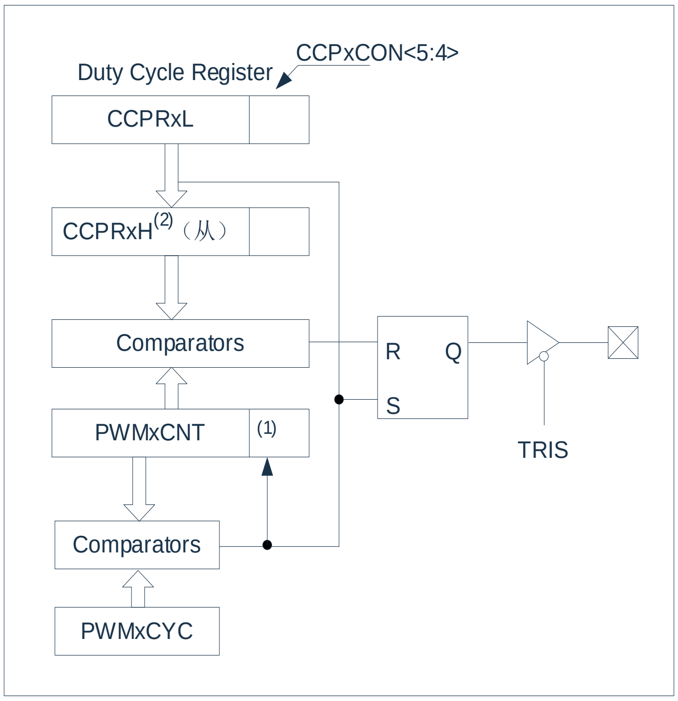


Fig 13-3: simplified PWM block diagram

Note:
1) The value of the 8-bit timer PWMxCNT register is combined with a 2-bit internal system clock (F_{SYS}) or 2 bits of the pre-scaler to generate a 10-bit time base.
2) In PWM mode, CCPRxH is a read-only register.

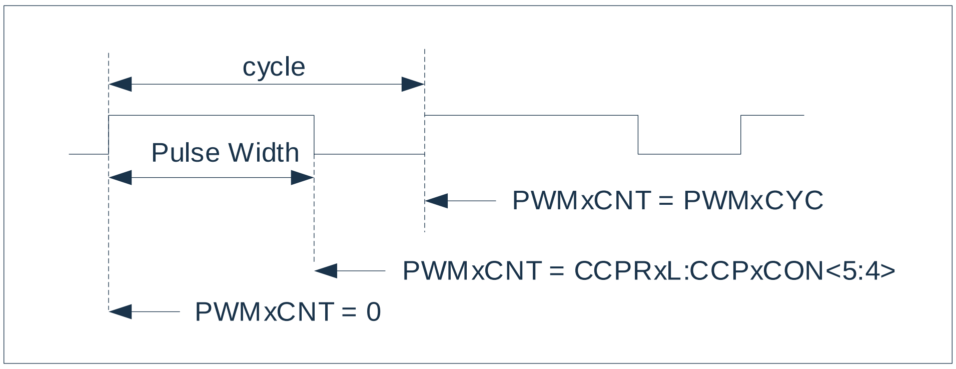


Fig 13-4: CCP PWM output

13.3.1 PWM Period

The PWM period is specified by writing the PWMxCYC register of PWMxCNT.

Formula 1： PWM period：

$$\text{PWM period}=[(PWMxCYC)+1]*4*Tosc*(PWMxCNT\ prescaler\ value)$$

Note: T_{OSC}=1/F_{SYS}

When PWMxCNT is equal to PWMxCYC, the following three events will occur in the next up-counting period:

- PWMxCNT is cleared;
- CCPx pin is set to 1 (exception: if PWM duty cycle=0%, CCPx pin will not be set to 1);
- The PWM duty cycle is latched from CCPRxL to CCPRxH.

13.3.2 PWM Duty Cycle

The PWM duty cycle can be specified by writing a 10-bit value to the following multiple registers: DCxB<1:0> bits of the CCPRxL register and CCPxCON register. CCPRxL stores the upper 8 bits of the duty cycle, and DCxB<1 of the CCPxCON register: The 0> bit saves the lower 2 bits of the duty cycle. You can write the DCxB<1:0> bits of the CCPRxL and CCPxCON register at any time, but until the values in PWMxCYC and PWMxCNT match (that is, the period ends), the value of the duty cycle It is latched into CCPRxH. In PWM mode, CCPRxH is a read-only register.

Formula 2: Pulse width calculation formula:

$$\text{pulse width}=(CCPRxL{:}CCPxCON<5{:}4>)*T_{OSC}*(PWMxCNT\ prescaler\ value)$$

Formula 3: PWM duty cycle calculation formula:

$$\text{duty cycle}=\frac{(CCPRxL{:}CCPxCON<5{:}4>)}{4\ (PWMxCYC+1)}$$

The CCPRxH register and a 2-bit internal latch are used to provide double buffering for the PWM duty cycle. This double buffering structure is extremely important to avoid glitches during the PWM operation.

The value of the 8-bit timer PWMxCNT register is combined with a 2-bit internal system clock (F_{SYS}) or 2 bits of the pre-scaler to generate a 10-bit time base. The system clock is used when the PWMxCNT prescaler ratio is 1:1.

When the 10-bit time base matches the combined value of CCPRxH and 2-bit latch, the CCPx pin is cleared (see Figure 13-3).

.

13.3.3 PWM Resolution

Resolution determines the number of duty cycles within a given period. For example, a 10-bit resolution will generate 1024 discrete duty cycles, and an 8-bit resolution will generate 256 discrete duty cycles.

When PWMxCYC is 255, the maximum resolution of PWM is 10 bits. As shown in formula 4, resolution is a function of the value of the PWMxCYC register.

formula4：PWM resolution:

$$\text{resolution}=\frac{\log[4\ (PWMxCYC+1)]}{\log\ (2)}$$

Note: If the pulse width is greater than the period value, the specified PWM pin will remain unchanged.

The following table shows the value of PWM frequency and resolution when FSYS=8MHz.

PWM frequency and resolution example (F_{SYS}=8MHz)

PWM frequency	1.22KHz	4.90KHz	19.61KHz	76.92KHz	153.85KHz	200.0KHz
Timer prescaler (1,4 or16)	16	4	1	1	1	1
PWMxCYC	0x65	0x65	0x65	0x19	0x0C	0x09
Largest resolution (bit)	8	8	8	6	5	5

13.3.4 Operations Under Sleep Mode

In sleep mode, the PWMxCNT register will not increment and the mod state will remain unchanged. If the CCPx pin has an output, it will continue to maintain the output value unchanged. When the device is woken up, PWMxCNT will continue to work from the original state.

13.3.5 System Clock Frequency Changes

The PWM frequency is generated by the systemclock frequency. Any change in the system clock frequency will change the PWM frequency.

13.3.6 Effect of Reset

Any reset will force all ports to be input mode and force the CCP register to enter its reset state.

13.3.7 Configurate PWM

The following steps should be performed when configuring CCP mod to PWM operation mode:

1. Disable the outputdriver of PWMpin (CCPx) by setting the corresponding TRIS bit to 1 to make it an input pin.
2. Set the PWM period by loading the PWMxCYC register.
3. By loading the CCPxCON register configuration CCP mod PWM mode with appropriate values.
4. Set the PWM duty cycle by loading the DCxB<1:0> bits in the CCPRxL register and CCPxCON register.
5. Configure and start the PWMxCNT period counter:
   - Clear the TMR2IF interrupt flag bit in the PIR1 register.
   - Set the PWMxCNT prescaler ratio by loading the CK2 or CK1 bit of the PWMCON register.
   - Enable PWMxCNT by setting the CYC2EN and CYC1EN bits in the PWMCON register to 1.
6. After the new PWM period starts, enable PWM output:
   - Wait for PWMxCNT overflow.
   - Enable the CCPx pin output driver by clearing the corresponding TRIS bit.

14. Universal Synchronous/Asynchronous Transmitter (USART0/USART1)

The universal synchronous/asynchronous transmitter (USART) mod is a serial I/O communication peripheral. This mod includes all the clock generators, shift registers and data buffers necessary to perform input or output serial data transmissions that are not related to device program execution. USART It can also be called a serial communication interface (Serial Communications Interface, SCI), it can be configured as a duplex asynchronous system that can communicate with peripherals such as CRT terminals and personal computers; it can also be configured as an integrated circuit with A/D or D/A, Serial EEPROM and other peripherals or half-duplex synchronous system of other microcontroller communication. The microcontroller with which it communicates usually does not have an internal clock that generates baud rate, it needs a master control synchronous device to provide an external clock signal.

The functions of USART0 and USART1 are exactly the same. The following description is based on USART0 mod.

The USART mod includes the following functions:

- Duplex asynchronous transmit and receive
- Single character output buffer
- Double character input buffer
- Frame error detection from receive to character
- Half-duplex synchronous slave mode
- Character length can be programmed to 8 or 9 bits
- Input buffer overflow error detection
- Half-duplex synchronousmaster controlmode
- In synchronous mode, programmable clock polarity

Figure 14-1 and Figure 14-2 below are the block diagrams of the USART transmitter.

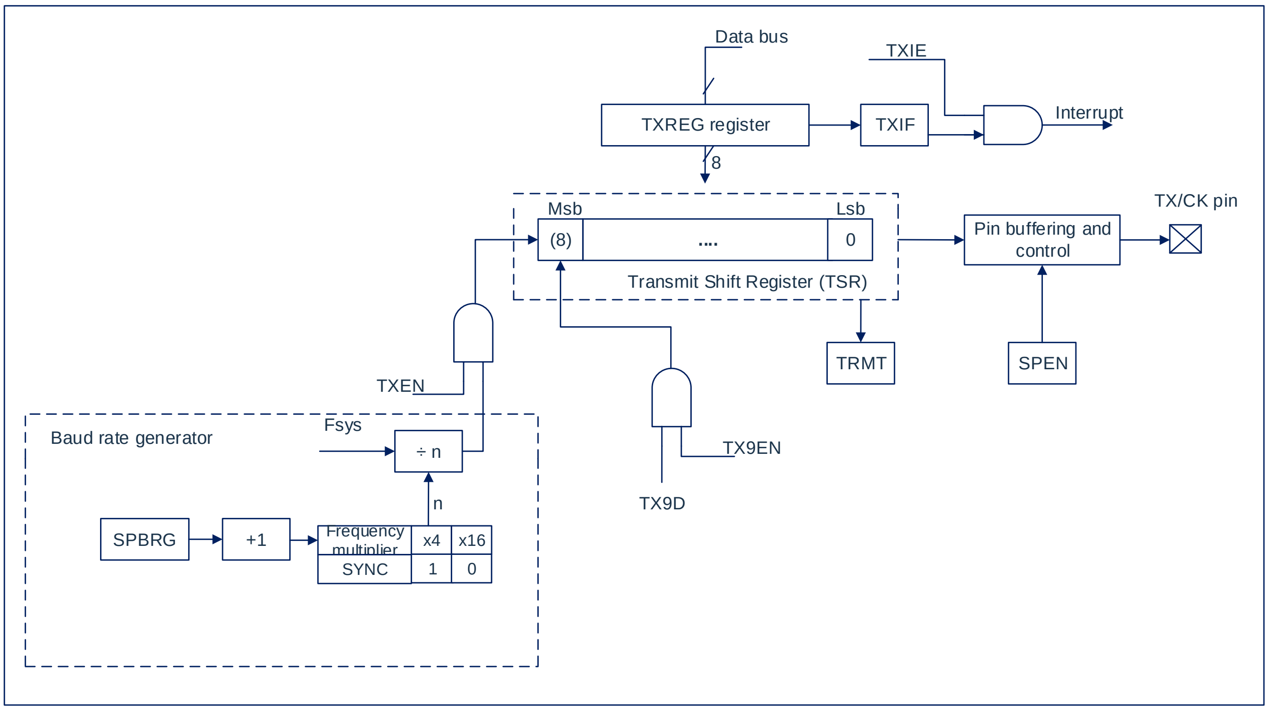


Fig 14-1: USART transmit block diagram

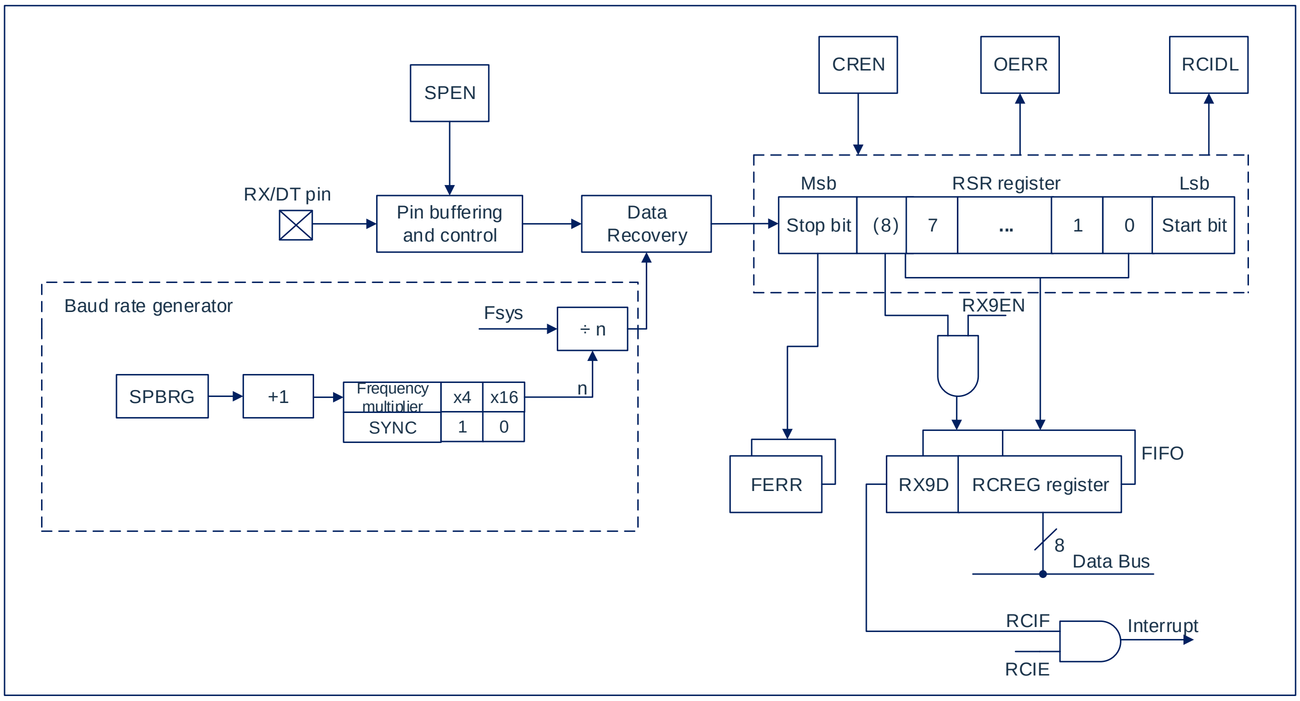


Fig 14-2: USART receive block diagram

The operation of the USART mod is controlled by 3 registers:

- transmit status and control register (TXSTA)
- Receive status and control register (RCSTA)

14.1 USART Asynchronous Mode

USART uses the standard non-return-to-zero (NRZ) format for transmit and receive data. Two levels are used to implement NRZ:

It represents the VOH mark state (mark state) of 1data bit, and the VOL space state (space state) of 0 data bit. When using NRZ format to continuously transmit data bits of the same value, the output level will maintain the level of the bit, and It will return the mid-level value after each bit is transmitted. NRZ transmit port is idle in the mark state. The character of each transmit includes a start bit, followed by 8 or 9 data bits and one or more terminations the stop bit of character transmit. The start bit is always in the space state, and the stop bit is always in the mark state. The most commonly used data format is 8 bits. The duration of each transmit bit is 1/ (baud rate). On-chip dedicated 8 Bit/16-bit baud rate generator can be used to generate standard baud rate frequency through system oscillator.

USART first transmit and receive LSb. USART's transmitter and receiver are functionally independent, but use the same data format and baud rate. Hardware does not support parity check, but it can be implemented by software (parity bit is the first 9 data bits).

14.1.1 USART Asynchronous Generator

Figure 15-1 shows the block diagram of the USART transmit device. The core of the transmit device is the serial transmit shift register (TSR), which cannot be directly accessed by software. TSR obtains data from the TXREG transmit buffer register.

14.1.1.1 Enable Transmit

Enable USART transmit by configuring the following three control bits for asynchronous operation:

- TXEN=1
- SYNC=0
- SPEN=1

It is assumed that all other USART control bits are in their default state.

Set the TXEN bit of the TXSTA register to 1 to enable the USART transmitter circuit. Clear the SYNC bit of the TXSTA register to zero and use the USART configuration for asynchronous operation.

Note:
1) When the SPEN bit and TXEN bit are set to 1, the SYNC bit is cleared, TX/CKI/Opin is automatically configured as an output pin, regardless of the state of the corresponding TRIS bit.
2) When the SPEN bit and CREN bit are set to 1, the SYNC bit is cleared, and RX/DTI/Opin is automatically configured as an input pin, regardless of the state of the corresponding TRIS bit.

14.1.1.2 Transmit Data

Write a character to the TXREG register to start transmit. If this is the first character, or the previous character has been completely removed from the TSR, the data in TXREG will be immediately transmitted to the TSR register. If all or part of the TSR is still stored.The previous character, the new character data will be stored in TXREG until the stop bit of the previous character is transmitted. Then, after the stop bit is transmitted, after a TCY, the data to be processed in TXREG will be transmitted to TSR. When After data is transmitted from TXREG to TSR, the start bit, data bit, and stop bit sequence are transmitted immediately.

14.1.1.3 Transmit Interrupt

As long as the USART transmitter is enabled and there is no data to be transmitted in TXREG, the TXIF interrupt flag bit of the PIR1 register is set to 1. In other words, only when the TSR is busy processing the character and there are new characters queued for transmit in the TXREG, the TXIF bit It is in the cleared state. When writing TXREG, the TXIFflag bit is not cleared immediately. TXIF is cleared at the second instructions period after writing the instructions. Querying TXIF immediately after writing TXREG will return an invalid result. TXIF is a read-only bit and cannot Set or cleared by software.

TXIFinterrupt can be enabled by setting the TXIE interrupt enable bit of PIE1register. However, as long as TXREG is empty, the TXIFflag bit will be set to 1 regardless of the status of the TXIEenable bit.

If you want to use interrupt when transmitting data, set the TXIE bit to 1 only when the data is to be transmitted. After writing the last character to be transmitted to TXREG, clear the TXIE interrupt enable bit.

14.1.1.4 TSR Status

The TRMT bit of the TXSTA register indicates the status of the TSR register. The TRMT bit is a read-only bit. When the TSR register is empty, the TRMT bit is set to 1, and when a character is transferred from the TXREG to the TSR register, the TRMT is cleared. The TRMT bit remains Clear the state until all bits are removed from the TSR register. There is no interrupt logic related to this bit, so the user must query this bit to determine the state of the TSR bit.

Note: The TSR register is not mapped to the data memory, so the user cannot directly access it.

14.1.1.5 Transmit 9-bit Character

The USART supports 9-bit character transmit. When the TX9EN bit of the TXSTA register is 1, the USART will shift out 9 bits of each character to be transmitted. The TX9D bit of the TXSTA register is the 9th bit, which is the highest data bit. When the 9-bit data is transmitted, it must Before writing the 8 least significant bits to TXREG, write the TX9D data bit. After writing the TXREG register, the 9 data bits will be transferred to the TSR shift register immediately.

14.1.1.6 Configure Asynchronous Transmit

1. Initialize the SPBRG register to obtain the required baud rate (see "USART baud rate generator (BRG)"
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit to 1.
3. If 9-bit transmit is required, set the TX9EN control bit to 1. When the receiver is set for address detection, set the 9th bit of the data bit to 1, indicating that the 8 lowest data bits are address.
4. Set the TXEN control bit to 1 to enable transmit; this will cause the TXIF interrupt flag bit to be set to 1.
5. If interrupt is required, set the TXIE interrupt enable bit in PIE1register to 1; if the GIE and PEIE bits in the INTCON register are also set to 1, interrupt will occur immediately.
6. If you choose to transmit 9-bit data, the 9th bit should be loaded into the TX9Ddata bit.
7. Load 8-bit data into TXREG register to start transmitting data.

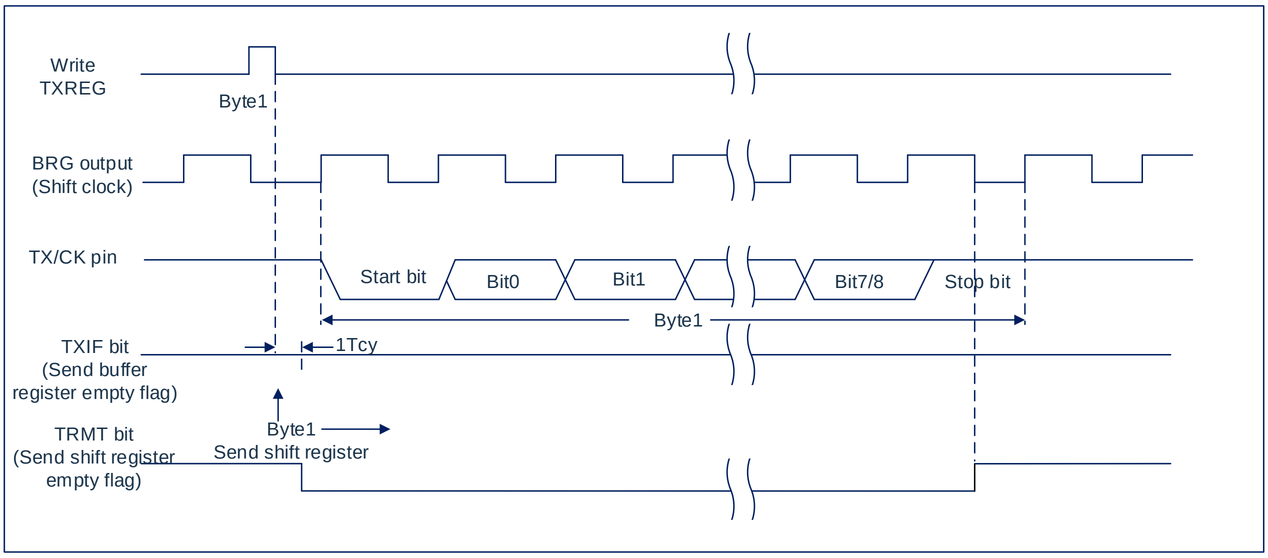


Fig 14-3: asynchronous transmit

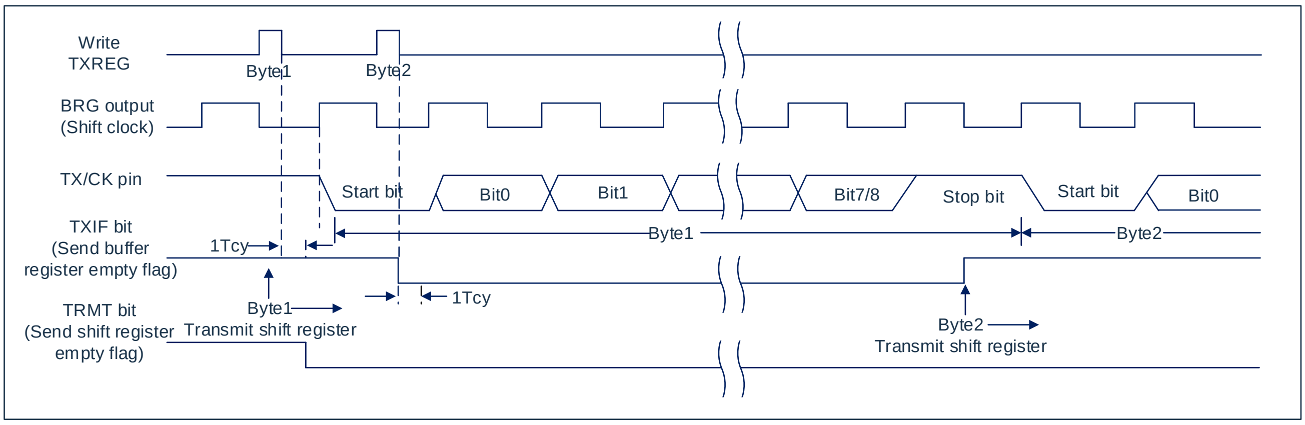


Fig14-4: asynchronous transmit (back to back)

Note: This time series diagram shows two consecutive transmit.

14.1.2 USART Asynchronous Receiver

Asynchronous mode is usually used in RS-232 system. Figure 14-2 shows the block diagram of the receiver. Receive data and driver data recovery circuit on RX/DTpin. The data recovery circuit is actually a 16 times baud rate as the operating frequency High-speed shifter, while the serialreceive shift register (Receive Shift Register, RSR) works at the bit rate. When all the 8-bit or 9-bit data bits of the character are shifted in, they are immediately transferred to a 2-character FIFO (FIFO) buffer. FIFO buffer allows to receive 2 complete characters and the start bit of the third character, and then software must provide the received data to the USART receiver. FIFO and RSR register cannot be directly accessed by software. The RCREG register accesses the received data.

14.1.2.1 Enable Receiver

Enable the USART receiver by configuring the following three control bits for asynchronous operation.

- CREN=1
- SYNC=0
- SPEN=1

Assuming that all other USART control bits are in the default state. Set the CREN bit of the RCSTA register to 1 to enable the USART receiver circuit. Clear the SYNC bit of the TXSTA register to zero and configure the USART for asynchronous operation.

Note:
1) When the SPEN bit and TXEN bit are set to 1, the SYNC bit is cleared, and the TX/CKI/O pin is automatically configured as an output pin, regardless of the state of the corresponding TRIS bit.
2) When the SPEN bit and CREN bit are set to 1, the SYNC bit is cleared, and the RX/DTI/O pin is automatically configured as an input pin, regardless of the state of the corresponding TRIS bit.

14.1.2.2 Receive Data

Receiver data recovery circuit starts the receive character at the falling edge of the first bit. The first bit, usually called the start bit, is always 0. The data recovery circuit counts half a bit time to the center of the start bit. Check whether the bit is still zero. If the bit is not zero, the data recovery circuit will give up receiving the character without error, and continue to look for the falling edge of the start bit. If the zero check of the start bit passes, then the data recovery circuit counts a complete bit time and reaches the center position of the next bit. The majority detection circuit samples the bit and moves the corresponding sampling result 0 or 1 into the RSR. Repeat the process until all data bits are completed Sampling and moving it all into RSRregister. Measure the time of the last bit and sample its level. This bit is the stop bit and is always 1. If the data recovery circuit samples 0 at the stop bit position, the character frame error The flag will be set to 1, otherwise, the frame error flag of the character will be cleared.

When all data bits and stop bits are received, the character in the RSR will be immediately transferred to the receive FIFO of the USART and the RCIF interrupt flag bit of PIR1register is set to 1. The character at the top of the FIFO is moved out of the FIFO by reading the RCREG register.

Note: If you receive FIFO overflow, you cannot continue to receive other characters until the overflow condition is cleared.

14.1.2.3 Receive Interrupt

As long as the USART receiver is enabled and there is no unread data in the receive FIFO, the RCIF interrupt flag bit in the PIR1 register will be set to 0. The RCIF interrupt flag bit is read-only and cannot be set or cleared by software.

RCIF interrupt is enabled by setting all of the following bits:

- RCIE interrupt enable bit of PIE1 register;
- PEIE peripherals interrupt enable bit of INTCON register;
- GIE global interrupt enable bit of INTCON register.

If there is unread data in the FIFO, regardless of the state of the interrupt enable bit, the RCIF interrupt flag bit will be set to 1.

14.1.2.4 Receive Frame Error

Each character in the Receive FIFO buffer has a corresponding frame error status bit. The frame error indicates that the stop bit was not received within the expected time.

The framing error status is obtained by the FERR bit of the RCSTA register. The FERR bit must be read after reading the RCREG register.

Framing error (FERR=1) will not prevent receiving more characters. There is no need to clear the FERR bit.

Clearing the SPEN bit of the RCSTA register will reset the USART and forcibly clear the FERR bit. Framing error itself will not cause interrupt.

Note: If all characters received in the receive FIFO buffer have framing errors, repeated reading of RCREG will not clear the FERR bit.

14.1.2.5 Receive Overflow Error

The receive FIFO buffer can store 2 characters. However, if the third character is received before accessing the FIFO, an overflow error will occur. At this time, the OERR bit of the RCSTA register will be set to 1. The character inside FIFO buffer can be read, but before the error is cleared, no other characters can be received. The error can be cleared by clearing the CREN bit of the RCSTA register or by clearing the SPEN bit of the RCSTA register to make USART reset.

14.1.2.6 Receive 9-bit Character

The USART supports 9-bit data receive. When the RX9EN bit of the RCSTA register is set to 1, the USART will shift the 9 bits of each character received into the RSR. You must read the RX9D data bit after reading the lower 8 bits in RCREG.

14.1.2.7 Asynchronous Receive Configuration

1. Initialize the SPBRG register to obtain the required baud rate.
   (Please refer to the "USART baud rate generator (BRG)" chapter.
2. Set the SPEN bit to 1 to enable the serial port. The SYNC bit must be cleared to perform asynchronous operations.
3. If interrupt is required, set the RCIE bit in the PIE1 register and the GIE and PEIE bits in the INTCON register to 1.
4. If you need to receive 9 bits of data, set the RX9EN bit to 1.
5. Set the CREN bit to 1 to enable receive.
6. When a character is transferred from the RSR to the receive buffer, set the RCIF interrupt flag bit to 1. If the RCIE interrupt enable bit is also set to 1, an interrupt will also be generated.
7. Read the RCREG register and get the received 8 low data bits from the receive buffer.
8. Read the RCSTA register to get the error flag bit and the 9th data bit (if 9-bit datareceive is enabled).
9. If overflow occurs, clear the OERR flag by clearing the CREN receiver enable bit.

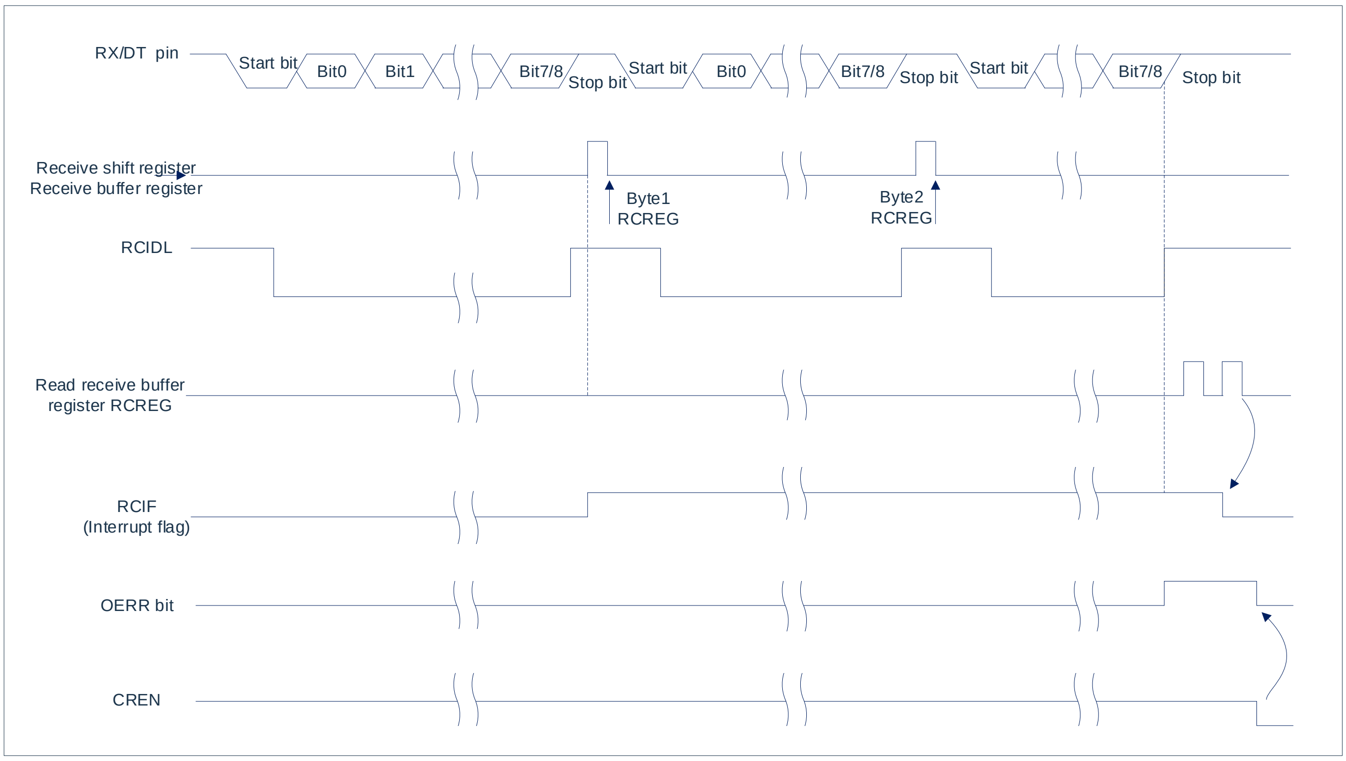


Fig 14-5: Asynchronous receive

Note: This time series diagram shows the situation of three words received in RX input pin. Reading RCREG (receive buffer) after the third word results in OERR (overflow) bit 1.

14.2 Clock Precision for Asynchronous Operations

The output of the internal oscillation circuit (INTOSC) is calibrated by the manufacturer. But when VDD or temperature changes, INTOSC will have a frequency shift, which will directly affect the asynchronous baud rate. The baud rate clock can be adjusted by the following methods, but some type of reference is required clock source.

14.3 USART Related Register

TXSTA0: transmit status and control register (1EH)

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXSTA0	CSRC0	TX9EN0	TXEN0	SYNC0	SCKP0	--	TRMT0	TX9D0
read/write	R/W	R/W	R/W	R/W	R/W	--	R	R/W
Reset value	0	0	0	0	0	0	1	0

Bit7	CSRC0:	clock sourceselection bit;
	Asynchronous mode:	Any value;
	Synchronous mode:	
		1=master control mode (internal BRG generate clock signal);
		0=slave mode (external clock source generate clock).
Bit6	TX9EN0:	9-bit transmit enable bit;
	1=	Select 9-bit transmit;
	0=	Select 8-bit transmit.
Bit5	TXEN0:	Transmit enable bit (1);
	1=	Enable transmit;
	0=	Disable transmit.
Bit4	SYNC0:	USART mode selection bit;
	1=	Synchronous mode;
	0=	Asynchronous mode.
Bit3	SCKP0:	Synchronous clock polarity selection bit.
	Asynchronous mode:	
		1= Invert the level of the data character and transmit to the TX/CK pin;
		0= Directly transmit data character to TX/CK pin.
	Synchronous mode:	
		0= Data is transmitted on the rising edge of clock;
		1= Data is transmitted on the falling edge of clock.
Bit2	not used	
Bit1	TRMT0:	Transmit shift register status bit;
	1=	TSR empty;
	0=	TSR full.
Bit0	TX9D0:	9th bit of Transmit data.
		Can be address/data bit or parity check bit.

Note: In synchronous mode, SREN/CREN will invert the value of TXEN.

RCSTA0：receive status and control register (18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCSTA0	SPEN0	RX9EN0	SREN0	CREN0	RCIDL0	FERR0	OERR0	RX9D0
read/write	R/W	R/W	R/W	R/W	R	R	R	R
Reset value	0	0	0	0	1	0	0	0

Bit7	SPEN0:	serialportenable bit;
	1=	Enable serial port (RX/DT and TX/CK pin configured as serial portp in);
	0=	Disable serial port (hold on reset).
Bit6	RX9EN0:	9-bit receive enable bit;
	1=	Select 9-bit receive;
	0=	Select 8-bit receive;.
Bit5	SREN0:	Single byte receive enable bit.
	Asynchronous mode:	any value.
	Synchronous master control mode:	
		1=enable single byte receive;
		0=disable single byte receive.
		Clear after receive completed.
	Synchronous slave mode:	any value.
Bit4	CREN0:	Continuous receive enable bit.
	Asynchronous mode:	
		1=enable receive;
		0=disable receive.
	Synchronous mode:	
		1=enable continuous receive until clear CREN enable bit (CREN cover SREN);
		0=disable continuous receive.
Bit3	RCIDL0:	Receive idle flag bit.
	Asynchronous mode:	1=receiver idle;
		0= already receive initial bit，receiving data.
	Synchronous mode:	any value.
Bit2	FERR0:	frame error bit.
	1=	frame error (It can be updated by reading the RCREG register and receive the next valid byte);
	0=	No frame error.
Bit1	OERR0:	Overflow error bit.
	1=	Overflow error (clear by clearing CREN bit);
	0=	No overflow error.
Bit0	RX9D0:	Receive until 9th bit of the data.
		This bit can be the address/data bit or the parity check bit, which must be calculated by the user firmware.

14.4 USART Baud Rate Generator (BRG)

The baud rate generator (BRG) is an 8-bit, dedicated to supporting the asynchronous and synchronous working modes of USART.

The SPBRG register determines the period of the free-running baud rate timer.

Table 14-1 contains the formula for calculating baud rate. Formula 1 is an example of calculating baud rate and baud rate error.

Table 14-1 shows the typical baud rate and baud rate error values under various asynchronous modes that have been calculated, which is convenient for you to use.

Writing a new value to the SPBRG register pair will cause the BRG timer to reset (or clear). This can ensure that BRG can output a new baud rate without waiting for a timer overflow.

If the system clock changes during a valid receive process, a receive error may occur or data loss may occur. To avoid this problem, the state of the RCIDL bit should be checked to ensure that the receive operation is idle before changing the system clock.

formula1: claculate baud rate error

For device with F_{SYS}=8MHz, target baud rate=9600bps， asynchronous mode is 8-bit BRG:

$$\text{target baud rate}=\frac{Fsys}{16([SPBRG]+1)}$$

solve SPBRG:

$$X=\frac{\frac{FSYS}{\text{target baud rate}}}{16}-1=\frac{\frac{8000000}{9600}}{16}-1=[51.08]=51$$

$$\text{calculated baud rate}=\frac{8000000}{16\ (51+1)}=9615$$

$$\text{error}=\frac{\text{claculated baud rate}-\text{target baud rate}}{\text{target baud rate}}=\frac{(9615-9600)}{9600}=0.16\%$$

Table 14-1: baud rateformula

Configuration bit	BRG/USARTmode	baud rateformula
SYNC		
0	8bit/asynchronous	F_{SYS}/[16 (n+1)]
1	8bit/synchronous	F_{SYS}/[4 (n+1)]

Note: n= value of SPBRG register.

Table 14-2: baud rate in asynchronous mode

Target baud rate	SYNC=0					
	F_{SYS}=8.00MHz			F_{SYS}=16.00MHz		
	Real baud rate	error (%)	SPBRG value	Real baud rate	error (%)	SPBRG value
2400	2404	0.16	207	----	----	----
9600	9615	0.16	51	9615	0.16	103
10417	10417	0	47	10417	0	95
19200	19230	0.16	25	19230	0.16	51

14.5 USART Synchronous Mode

Synchronous serial communication is usually used in a system with a master control device and one or more slave devices. The master control device contains the necessary circuits to generate the baud rate clock and provides clock for all devices in the system. The slave device can use master control clock, so no internal clock generation circuit is needed.

In synchronous mode, there are two signal lines: bi-directional data line and clock line. The slave device uses the external clock provided by the master control device to move the serial data in or out of the corresponding receive and transmit shift register. Because of the use of bi-directional data lines, synchronous operation can only use half-duplex mode. Half-duplex means: master control device and slave device can receive and transmit data, but can not receive or transmit at the same time. USART can be used as a master control device, or as a slave device.

14.5.1 Synchronous Master Control Mode

The following bits are used to configure the USART for synchronous master control operation：

- SYNC=1
- CSRC=1
- SREN=0 (to transmit)； SREN=1 (to receive)
- CREN=0 (to transmit)； CREN=1 (to receive)
- SPEN=1

Set the SYNC bit of the TXSTA register to 1 to use the USART configuration for synchronous operation. Set the CSRC bit of the TXSTA register to 1 to configure the device as a master control device. Clear the SREN and CREN bits of the RCSTA register to zero to ensure that the device is in transmit mode. Otherwise, the device is configured to receive mode. Set the SPEN bit of the RCSTA register to 1, enable USART.

14.5.1.1 Master Control Clock

Synchronous data transmission uses an independent clock line to transmit data synchronously. The device configured as a master control device transmits clock signal on the TX/CK pin. When the USART is configured for synchronous transmit or receive operation, the TX/CK output driver automatically enables. Serial data bits are changed on the rising edge of each clock to ensure that they are valid on the falling edge. The time of each data bit is a clock period, and there can only be as many clock periods as there are data bits.

14.5.1.2 Clock Polarity

The device provides clock polarity options to be compatible with Microwire. The clock polarity is selected by the SCKP bit of the TXSTA register. Set the SCKP bit to 1 to set the clock idle state to high. When the SCKP bit is 1, data on the falling edge of each clock changes. Clear the SCKP bit and set the clock idle state to low. When the SCKP bit is cleared, data changes on each rising edge of the clock.

14.5.1.3 Synchronous Master control Transmit

The RX/DT pin output data of the device. When the USART configuration is synchronous master control transmit operation, the RX/DT and TX/CK output pins of the device are automatically enabled.

Write a character to the TXREG register to start the transmit. If all or part of the previous character is still stored in the TSR, the new character data is stored in TXREG until the stop bit of the previous character is transmitted. If this is the first character, Or the previous character has been completely removed from the TSR, the data in TXREG will be immediately transferred to the TSR register. When the character is transferred from TXREG to TSR, it will immediately begin to transmit data. Each data bit changes on the rising edge of the master control clock and remain effective until the rising edge of the next clock.

Note: The TSR register is not mapped to the data memory, so the user cannot directly access it.

14.5.1.4 Synchronous Master Control Transmit Configuration

1. Initialize the SPBRG register to obtain the required baud rate.
   (Please refer to the chapter "USART baud rate generator (BRG)".)
2. Set the SYNC, SPEN and CSRC bits to 1, enable synchronous master control serial port.
3. Clear the SREN and CREN bits to disable receive mode.
4. Set the TXEN bit to 1 to enable transmit mode.
5. If you need to transmit a 9-bit character, set TX9EN to 1.
6. If interrupt is required, set the TXIE bit in the PIE1 register and the GIE and PEIE bits in the INTCON register to 1.
7. If you choose to transmit 9-bit character, you should load the 9th bit of data into the TX9D bit.
8. Start transmit by loading data into TXREG register.

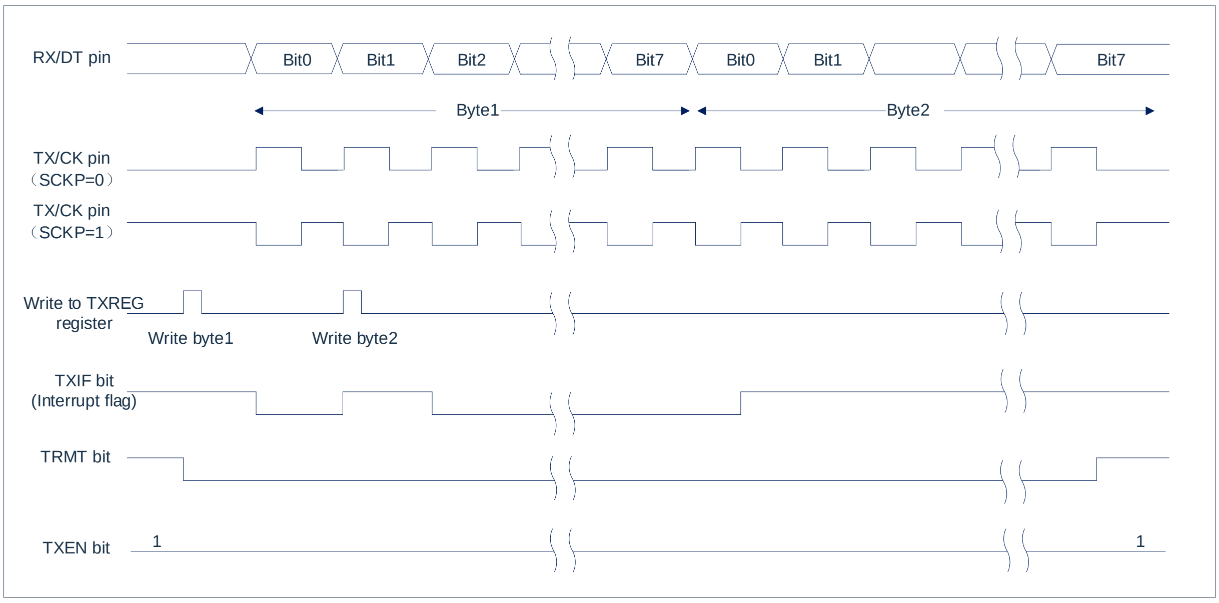


Fig 14-6: synchronous transmit

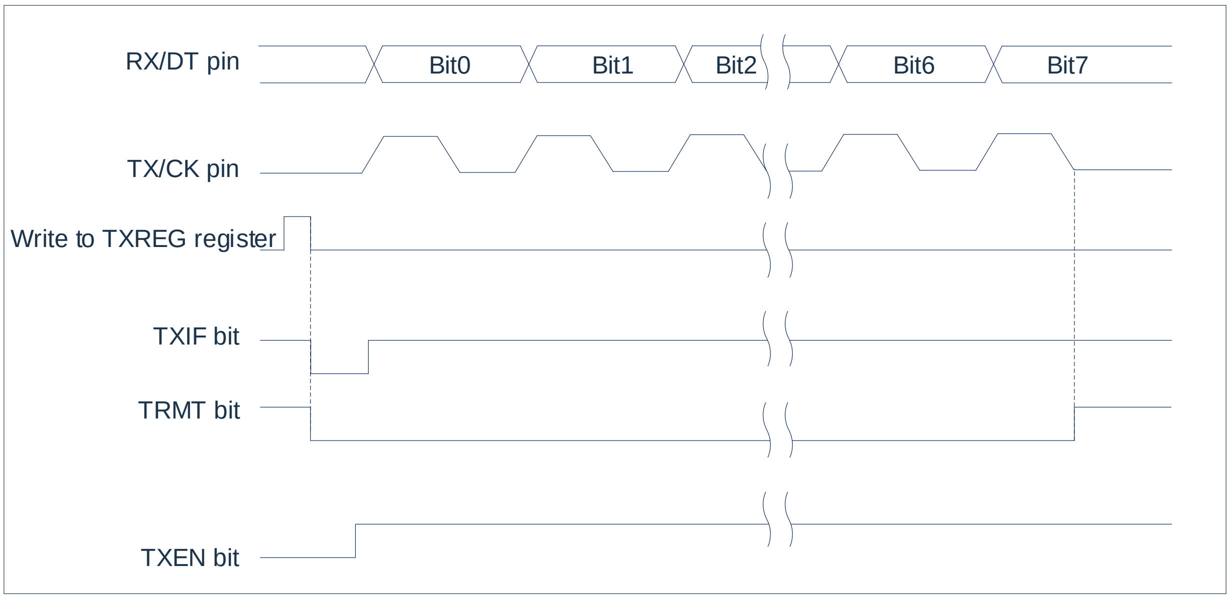


Fig 14-7: synchronous transmit (through TXEN)

14.5.1.5 Synchronous Master Control Receive

RX/DT pin receive data. When the USART configuration is synchronous master control receive, the output driver of the RX/DT pin of the device is automatically disabled.

In synchronous mode, set the single word receive enable bit (SREN bit of RCSTAregister) or continuous receive enable bit (CREN bit of RCSTA register) to 1 enable receive. When SREN is set to 1, the CREN bit is cleared, the number of clock period generated is as much as the number of data bit in single character. After a character transmission is over, the SREN bit is automatically cleared. When CREN is set to 1, a continuous clock will be generated until CREN is cleared. If CREN is cleared during a character transmission, The CK clock stops immediately and discards the incomplete character. If both SREN and CREN are set to 1, when the first character transfer is completed, the SREN bit is cleared, and CREN takes precedence.

Set the SREN or CREN bit to 1, start receiving. Sample the data on RX/DT pin at the falling edge of the TX/CK clock pin signal, and shift the sampled data into the receive shift register (RSR). When the RSR receives a complete character, the RCIF bit is set to 1, the character is automatically moved into the 2 byte receive FIFO. The lower 8 bits of the top character in the receive FIFO can be read through RCREG. As long as there are unread characters in the receive FIFO, the RCIF bit remains as 1.

14.5.1.6 Slave Clock

Synchronous data transmission uses an independent clock line synchronous with the data line. Clock signal on the TX/CK line of the slave device is received. When the device is configured to operate synchronously from the transmit or receive, the output driver of the TX/CK pin automatically disable. The serial data bit is changed at the leading edge of the clock signal to ensure that it is valid on the back edge of each clock. Each clock period can only transmit one bit of data, so how many data bits must be received is determined by how many data bits transmitted.

14.5.1.7 Receive Overflow Error

The receive FIFO buffer can store 2 characters. Before reading the RCREG to access the FIFO, if the third character is received completely, an overflow error will occur. At this time, the OERR bit of the RCSTA register will be set to 1. The previous data in the FIFO is not Will be rewritten. Two characters in the FIFO buffer can be read, but before the error is cleared, no other characters can be received. The OERR bit can only be cleared by clearing the overflow condition. If an overflow occurs, the SREN bit is set to 1, the CREN bit is in the cleared state, and the error is cleared by reading the RCREG register. If CREN is set to 1 during overflow, you can clear the CREN bit of the RCSTA register or clear the SPEN bit to reset USART, to clear the error.

14.5.1.8 Receive 9-bit Character

The USART supports receive 9-bit characters. When the RX9EN bit of the RCSTA register is 1, the USART moves the 9-bit data of each character received into the RSR. When reading 9-bit data from the receive FIFO buffer, it must read 8 lower bit of RCREG first.

14.5.1.9 Synchronous Master Control Receive Configuration

1. Initialize the SPBRG register to obtain the required baud rate. (Note: SPBRG>05H must be met)
2. Set the SYNC, SPEN and CSRC bits to 1 to enable synchronous master control serial port.
3. Make sure to clear the CREN and SREN bits.
4. If interrupt is used, set the GIE and PEIE bits of the INTCON register to 1, and set the RCIE bit of the PIE1 register to 1.
5. If you need to receive a 9-bit character, set the RX9EN bit to 1.
6. Set the SREN bit to 1 to enable receive, or set the CREN bit to 1 to enable continuous receive.
7. When the character receive is completed, set the RCIF interrupt flag bit to 1. If the enable bit RCIE is set to 1, an interrupt will also be generated.
8. Read the RCREG register to get the received 8-bit data.
9. Read the RCSTA register to get the 9th data bit (when 9-bit receive is enabled), and judge whether an error occurs during the receive process.
10. If an overflow error occurs, clear the CREN bit of the RCSTA register or clear SPEN to reset USART to clear the error.

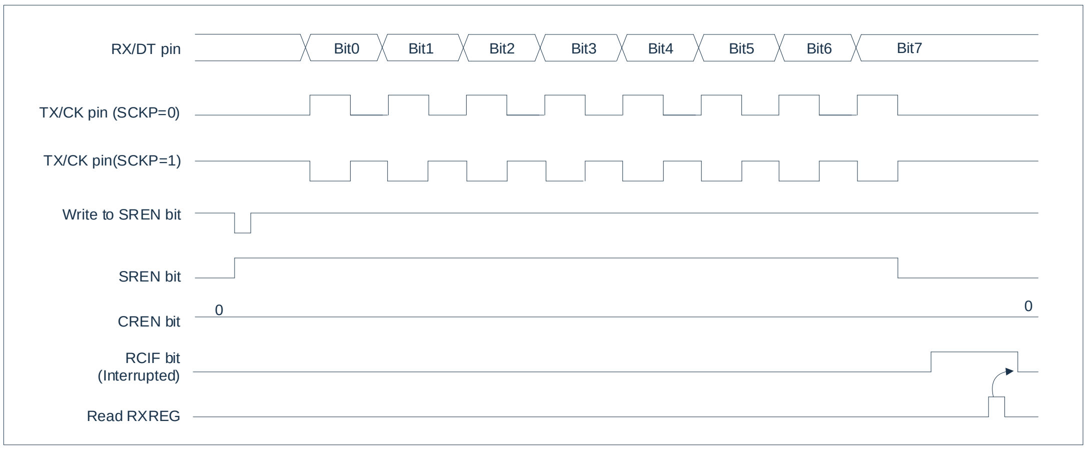


Fig 14-8: synchronous receive (master control mode, SREN)

Note: The time series diagram illustrates the synchronous master control mode when SREN=1.

14.5.2 Synchronous Slave Mode

The following bits are used to configure USART for synchronous slave operation:

- SYNC=1
- CSRC=0
- SREN=0 (to transmit); SREN=1 (to receive)
- CREN=0 (to transmit); CREN=1 (to receive)
- SPEN=1

Set the SYNC bit of the TXSTA register to 1 to configure the device for synchronous operation. Set the CSRC bit of the TXSTA register to 1 to configure the device as a slave device. Clear the SREN and CREN bits of the RCSTA register to zero to ensure that the device is in transmit mode. Otherwise, the device will be configured as receive mode. Set the SPEN bit of the RCSTA register to 1, enable USART.

14.5.2.1 USART Synchronous Slave Transmit

The working principle of synchronous master control and slave mode is the same (see chapter "synchronous master control transmission").

14.5.2.2 Synchronous Slave Transmit Configuration

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the CREN and SREN bits.
3. If interrupt is used, set the GIE and PEIE bits of the INTCON register to 1, and set the TXIE bit of the PIE1 register.
4. If you need to transmit 9-bit data, set the TX9EN bit to 1.
5. Set the TXEN bit to 1 to enable transmit.
6. If you choose to transmit 9-bit data, write the most significant bit to the TX9D bit.
7. Write the lower 8 bits of data to the TXREG register to start transmission.

14.5.2.3 USART Synchronous Slave Receive

Except for the following differences, the working principle of synchronous master control and slave mode is the same.

1. The CREN bit is always set to 1, so the receiver cannot enter the idle state.
2. SREN bit, can be "any value" in slave mode.

14.5.2.4 Synchronous Slave Receive Configuration

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. If interrupt is used, set the GIE and PEIE bits of the INTCON register to 1, and also set the RCIE bit of the PIE1 register.
3. If you need to receive a 9-bit character, set the RX9EN bit to 1.
4. Set the CREN bit to 1, enable receive.
5. When the receive is completed, set the RCIF bit to 1. If RCIE is set to 1, an interrupt will also be generated.
6. Read the RCREG register and get the received 8 low data bits from the receive FIFO buffer.
7. If you enable 9-bit mode, get the most significant bit from the RX9D bit of the RCSTA register.

If an overflow error occurs, clear the CREN bit of the RCSTA register or clear the SPEN bit to reset USART to clear the error.

15. Master Control Synchronous Serial Port (MSSP)Mod

15.1 Master Control SSP (MSSP)Mod General

master control synchronous serial port (Master Synchronous Serial Port, MSSP) mod is a serial interface for communicating with other peripherals or microcontrollers. These peripherals devices can be serial EEPROM, shift register, display driver or A/D converter, etc.

MSSP mod has the following two working modes:

- serial peripherals ports (SPI).
- I^{2}C.
  - Full master control mode.
  - Slave mode (Support broadcast address call).

I^{2}C interface supports the following modes at hardware level：

- master controlmode.
- Multi master mode.
- Slave mode.

15.2 SPI Mode

SPI mode allows simultaneous transmit and receive 8-bit data at the same time.SPI supports 3-wire mode and 4-wire mode communication.

The following three pins are used under 3-wire mod:

- Serial data input (SDIO)——RC7/SDIO
- Serial clock (SCK)——RC6/SCK
- slave selection (SS)——RC4/SS

The following three pins are used under 4-wire mod：

- serial data output (SDO)——RC5/SDO
- serial data input (SDI)——RC7/SDI
- serial clock (SCK)——RC6/SCK
- slave selection (SS)——RC4/SS

15.2.1 SPI Related Register

SSPSTAT: SSP status register (94H)

94H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPSTAT	---	CKE	MODE	---	---	---	---	---
read/write	---	R/W	R/W	---	---	---	---	---
Reset value	---	0	0	---	---	---	---	---

Bit7	Save, write 0	
Bit 6	CKE:	SPI clock edge selection bit. **(Note: In slave mode, CKE must be set to 0)**
	CKP=	0
		0= Transmit data on the rising edge of SCK pin；
		1= Transmit data on the falling edge of SCK pin.
	CKP =	1
		0= Transmit data on the falling edge of SCK pin；
		1= Transmit data on the rising edge of SCK pin.
Bit5	MODE:	Mode selection
		1=3-wire mode (When need to transmit, SDIO port TRIS bit needs to be cleared to 0; when need to receive, SDIO port TRIS needs to be set to 1)
		0=4-wire mode
Bit4~Bit0	Not used in SPI mode.	

SSPCON：SSP control register (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7	WCOL:	Write conflict detection bit.
	1=	In the process of transmit/receive data, try to write to the SSPBUF register.
	0=	No conflict.
Bit6	SSPOV:	Receive overflow flag bit.
	1=	When SSPBUF still keeps the previous data, a new byte is received. When overflow occurs, the data in SSPSR will be lost. Overflow will only occur in slave mode. In slave mode, even if transmit data only, user must read SSPBUF to avoid overflow. In master control mode, the overflow bit is not set to 1, because every time you receive or transmit new data, it must be started by writing to the SSPBUF register (this bit must be clear through software).
	0=	No overflow.
Bit5	SSPEN:	Synchronous serial port enable bit.
	1=	Enable serial port and configure SCK, SDO, SDI and SS as serial port pin.
	0=	disable serialp ort and configure these pins as I/O port pins.
Bit4	CKP:	Clock polarity selection bit.
	1=	Clock is high when idle.
	0=	Clock is low when idle.
Bit3~Bit0	SSPM<3:0>:	Synchronous serial port mode selection bit;
	0000=	SPI master control mode，clock= $F_{SYS}/4$;
	0001=	SPI master control mode，clock= $F_{SYS}/16$;
	0010=	SPI master control mode，clock= $F_{SYS}/64$;
	0011=	SPI master control mode，clock= TMR2 output/2;
	0100=	SPI slave mode，clock= SCKpin，enable SSpin control;
	0101=	SPI slave mode，clock= SCKpin，disable SSpin control，SS can be used as I/O pin;
	0110=	Save;
	0111=	Save;
	1000=	I^{2}C master control mode，clock= F_{SYS}/ (4 * (SSPADD+1));
	1001=	Disable load function;
	1010=	Save;
	1011=	Save;
	1100=	Save;
	1101=	Save;
	1110=	I^{2}C slave mode, 7-bit address, and allow start bit and stop bit interrupt;
	1111=	save.

15.2.2 SPI Working Principle

When initializing the SPI, several options need to be specified. They can be specified by programming the corresponding control bits (SSPCON<5:0> and SSPSTAT<7:6>). These control bits are used to specify the following options:

- master control mode (SCK as clock output)
- clock polarity (SCK idle state)
- clock rate (only in master control mode)
- slave selection mode (only in slave mode)
- Slave mode (SCK as clock input)
- Sampling phase of input data (the middle or end of data output time)
- clock edge (output data on the rising/falling edge of SCK)
-

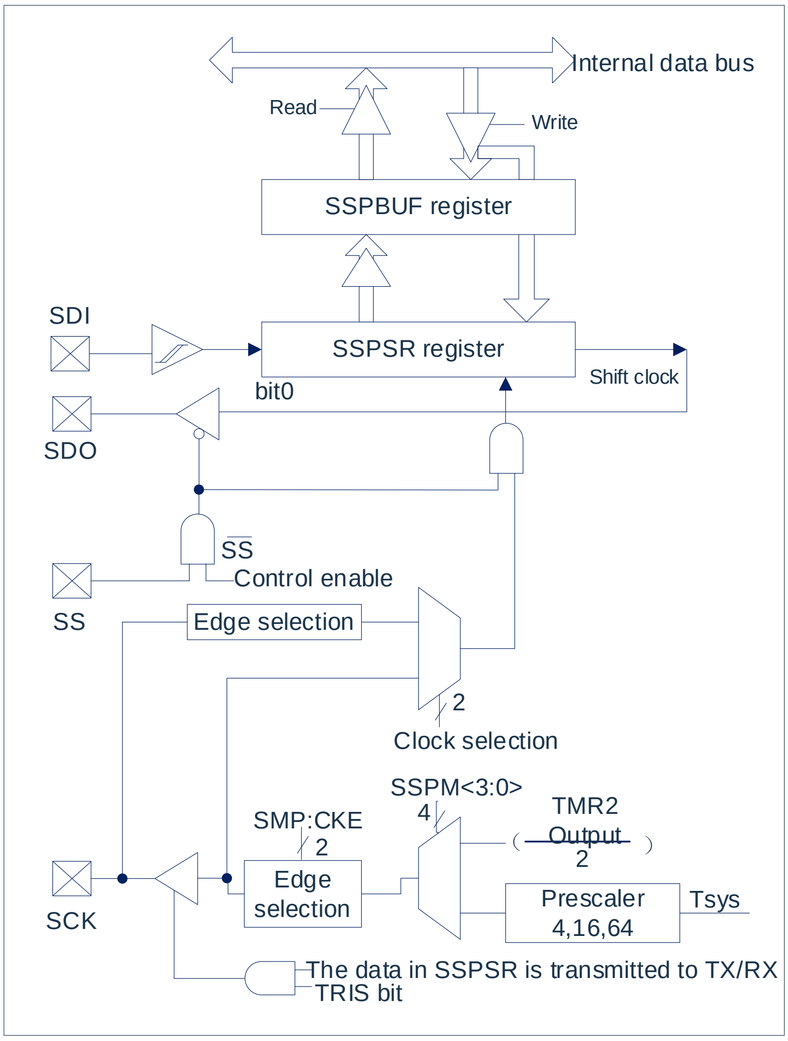


Fig 15-1 MSSP mod block diagram in SPI mode

Note: I/O pin has diode protection to VDD and VSS.

MSSP mod consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). SSPSR moves data in and out of the device, with the most significant bit first. SSPBUF saves the data written to the SSPSR last time until the new receive. The data is ready. Once the 8-bit data receive is completed, the byte is moved into the SSPBUF register. Then, the interrupt flag bit SSPIF of the PIR1 register is set to 1. This double-buffered data receive method (SSPBUF) allows reading the newly received data before starting to receive the next byte. During the data transmit/receive period, any attempt to write to the SSPBUF register will be ignored, and the write conflict detection bit WCOL of the SPCON register will be set to 1. At this time, the user must clear the WCOL bit by software, Otherwise it cannot be judged whether the next write operation to SSPBUF is successfully completed.

When the application software is waiting for the receive valid data, it should read the previous data in the SSPBUF before the next data byte to be transmitted is written into the SSPBUF. The buffer full flag bit BF (SSPSTAT register) is used to indicate when the SSPBUF has been loaded the received data (transmit is completed). If the SPI is only used as a transmitter, you don’t need to pay attention to the received data. MSSP interrupt can usually be used to determine when the transmit or receive is completed. If you do not use interrupt to handle the data transmission and reception and use software to query, this method also ensures that no write conflicts occur.

15.2.3 Enable SPI I/O

To enable the serial port, the MSSP enable bit SSPEN of the SSPCON register must be set to 1. To reset or reconfigure the SPI mode, first clear the SSPEN bit, reinitialize the SSPCON register, and then set the SSPEN bit to 1. This will set SDI, SDO, The SCK and SS pins are configured as serial port pins. To use these pins as serial ports, the data direction bits (in the TRIS register) must be programmed correctly, as follows:

- SDI controlled by SPI mod;
- TRISC<5> of SDO must be cleared;
- The TRISC<6> bit of SCK (master control mode) must be cleared;
- The TRISC<6> bit of SCK (slave mode) must be set to 1;
- The TRISC<4> of SS (slave mode) must be set to 1.

For any unwanted serial port function, you can skip it by setting the corresponding data direction (TRIS) register to the opposite value.

15.2.4 Master Control Mode

The master device controls SCK, so it can start data transmission at any time. The master device determines when the slave device should broadcast data according to the software protocol.

In master control mode, once data is written into the SSPBUF register, it will start to transmit or receive. If SPI is only used as a receiver, you can disable SDO output (program it to input). SSPSR register is connected to the SDI pin at the set clock rate the signal performs continuous shift input. After each byte receive is completed, it will be treated as a normal receive byte and loaded into the SSPBUF register (corresponding to interrupt and status position 1). This can be used as a “line activity monitoring” mode, which is very useful.

The clock polarity can be selected by programming the CKP bit of the SSPCON register accordingly. Figure 15-2, Figure 15-3, Figure 15-4, and Figure 15-5 show the SPI communication waveforms, where MSb is first transmitted. In master control mode, the SPI clock rate (bit rate) can be programmed by the user to one of the following rates:

- $F_{SYS}/4$ (or TCY)
- $F_{SYS}/16$ (or 4.TCY)
- $F_{SYS}/64$ (or 16.TCY)
- TIMER2 output/2

Figure 15-2 shows the waveform of the master control mode. When the CKE bit of the SSPSTAT register is 1, the SDO data is valid before the clock edge appears on the SCK. The figure indicates the time to load the received data into the SSPBUF.

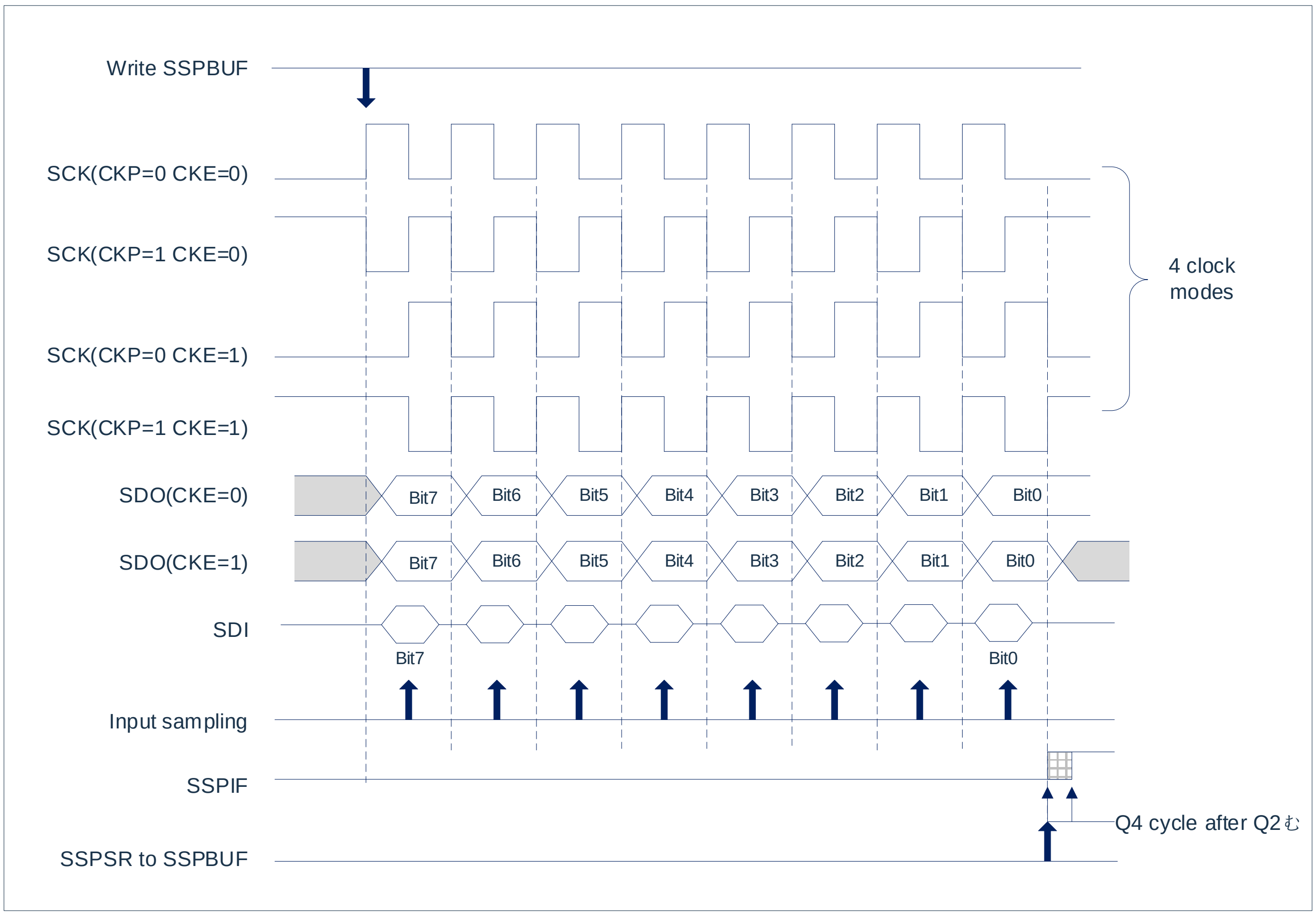


Fig15-2: SPI mode waveform (master control mode)

15.2.5 Slave Mode

In slave mode, when an external clock pulse appears on the SCK pin, transmit and receive data. When the last bit of data is latched, the SSPIF interrupt flag bit of PIR1register is set to 1.

In slave mode, the clock is provided by the external clock source on the SCK pin. The external clock must meet the minimum time requirements for high and low levels specified in the electrical specifications.

In the sleep state, the slave device can still transmit/receive data. When a byte is received, the device is awakened from the sleep state.

15.2.6 Slave Synchronous Selection

SS pin allows the device to work in synchronous slave mode. SPI must work in slave mode, and enable SSpin to control SSPxCON1<3:0> = 04h). To use SS pin as input in, the pin driver cannot be set to low level. When the SS pin is low, the transmit and receive of the data are enabled, and the SDO pin is used by the driver. When the SS pin is high, the SDO pin is no longer drived even during the data transmit process. It becomes a floating output. According to the needs of the application, an external pull up/ pull down resistor can be connected.

After SPI mod reset, the bit counter is forced to 0. This can be achieved by forcing the SS pin to be pulled high or clearing the SSPEN bit. Connecting the SDO pin and the SDI pin can simulate a two-wire communication. When SPI When it needs to work as a receiver, SDOpin can be configured as input. This will disable the transmit data from SDO. Because SDI will not cause a bus conflict, it can always be reserved as input (SDI function).

Note: When SPI works in slave mode and SSpin control is enabled (SSPxCON1<3:0> = 0100), if SSpin is set to VDD level, SPI mod will be reset.

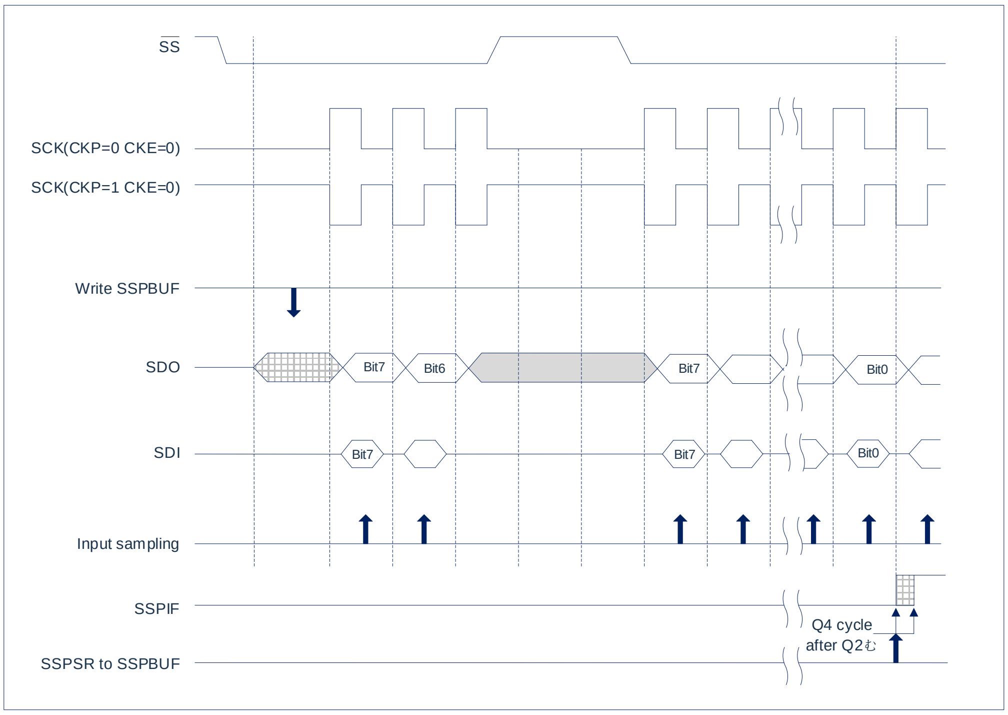


Fig 15-3: Slave synchronous waveform

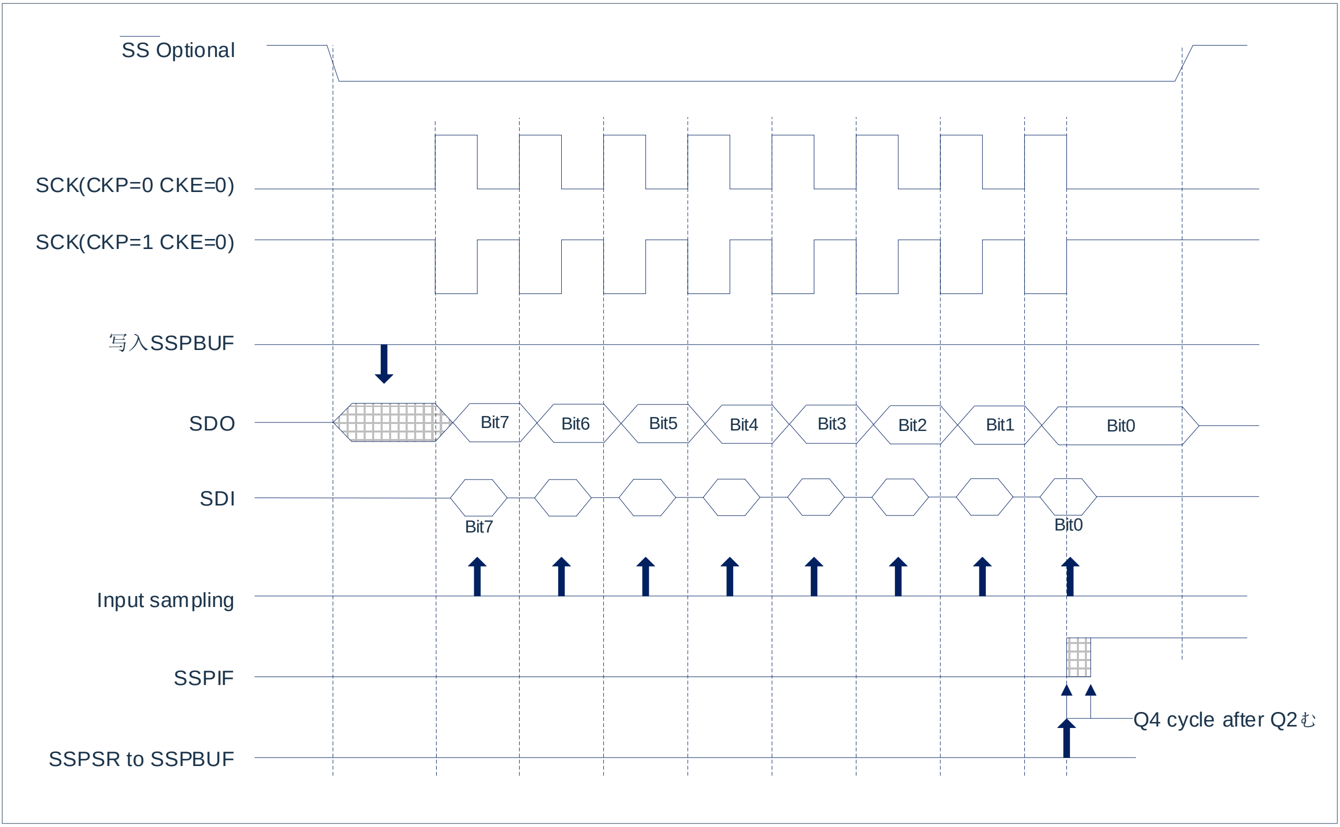


Fig 15-4: SPI mode waveform (slave mode，CKE=0)

15.2.7 Sleep Operation

In sleep mode, all mod clocks will stop, and before the device is awakened, transmit/receive will remain in this stagnant state. When the device returns to running mode, the mod will resume to transmit and receive data.

15.2.8 Effect of Reset

reset will disable MSSP mod and terminate the current transmission.

15.3 I^{2}C Mod

When MSSP mod works in I^{2}C mode, it can realize all master control and slave functions (including broadcast call support), and use hardware to provide interrupts of the start and stop bits to determine when the bus is idle (multi-master function).

There are two pins for data transmission. You can select the I2C function pin assignment in the system configuration register to RD0/RD1 or RC6/RC7. When using the I2C function, the user must pass TRISD<1:0> or TRISC<7:6 > to configure these pins as input pins. By setting the MSSP enable bit, SSPEN, of the SSPCON register to 1, MSSP mod function is enbaled.

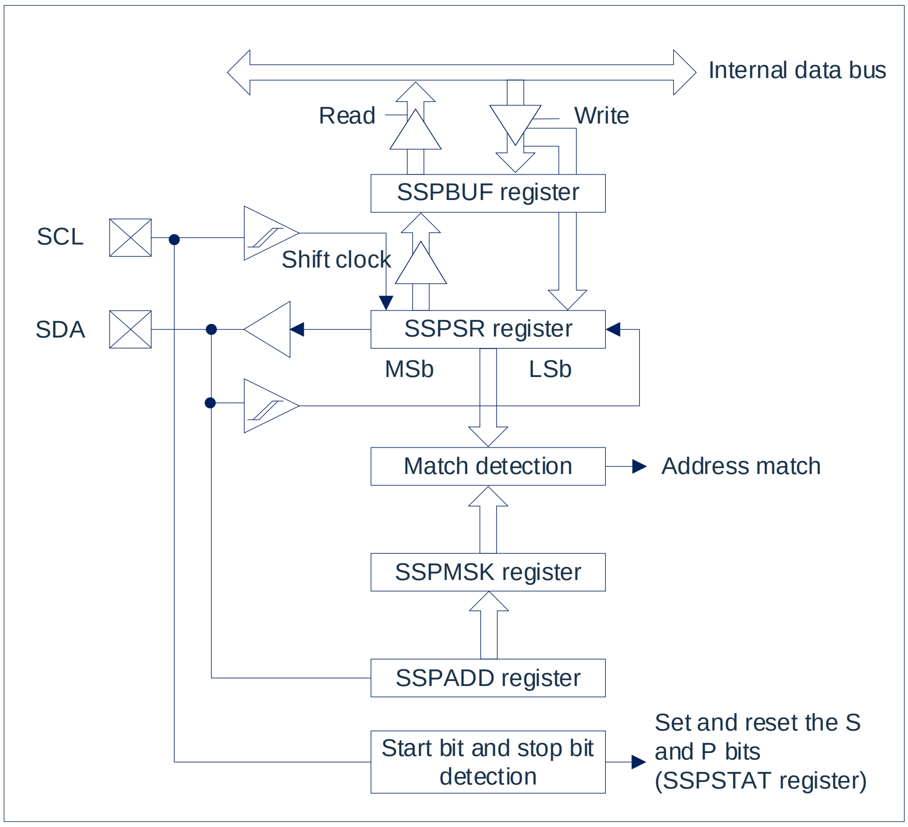


Fig15-5: MSSP block diagram (I^{2}C mode)

Note: The I/O pin has protection diodes connected to VDD and VSS.

MSSP mod has 7 registers for I^{2}C operation. They are:

- MSSP control register1 (SSPCON)
- MSSP status register (SSPSTAT)
- MSSP shift register (SSPSR): not directly accessible
- MSSP masking register (SSPMSK)
- MSSP control register2 (SSPCON2)
- serial receive/transmit buffer register (SSPBUF)
- MSSP address register (SSPADD)

You can use SSPCON register to control the operation of I^{2}C. You can use the SSPM<3:0> mode selection bit (SSPCON register) to select one of the following I^{2}C modes:

- I2C slave mode, 7-bit address, allow start bit and stop bit interrupt
- I2C master control mode, clock=FSYS/ (4* (SSPADD+1))

If the SCL and SDA pins have been programmed as input pins (set the corresponding TRIS bit to 1), selecting any I^{2}C mode and SSPEN bit as 1 will force the SCL and SDA pins to be open drain.

15.3.1 Related Register Illustration

SSPSTAT: SSP status register (94H)

94H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPSTAT	----	IDLE	D/A	P	S	R/W	----	BF
read/write	----	R	R	R	R	R	----	R
Reset value	----	1	0	0	0	0	----	0

Bit7	not used	Not used in I^{2}C mode
Bit6	IDLE	master control mode idle bit (Only the master control mode is valid, all master control operations can use this bit to determine whether to terminate)
	1=	No master control operation on the bus
	0=	The master control operation is in progress on the bus
Bit5	D/A:	data/address bit.
	1=	Indicates that the last receive or transmit byte is data.
	0=	Indicates that the last receive or transmit byte is address.
Bit4	P:	Stop bit (this bit is cleared when MSSP mod is disabled (SSPEN is cleared)).
	1=	Indicates that the stop bit was finally detected (the bit is 0 when reset).
	0=	Indicates that the stop bit was not detected at the end.
Bit3	S:	Start bit (this bit is cleared when disable MSSP mod (SSPEN is cleared)).
	1=	Indicates that the start bit was finally detected (the bit is 0 when reset).
	0=	The start bit was not detected at the end.
Bit2	R/W:	Read/write bit. This bit is used to save the R/W bit information after the last address match. This bit is only valid from the address match to the next start bit, stop bit or non-ACK bit.
	In I^{2}C slave mode:	
		1= read.
		0= write.
	I^{2}C master control mode:	
		1= transmiting.
		0= not transmiting. The result of logic OR operation between this bit and SEN, RSEN, PEN, RCEN or ACKEN will indicate whether MSSP is in idle mode.
Bit1	not used	
Bit0	BF	buffer full status bit.
	receive:	
		1= receive complete, SSPBUF full.
		0= receive not complete， SSPBUF empty.
	transmit:	
		1 = data transmiting (not including ACK and stop bit), SSPBUF full.
		0 = data transmit complete (not including ACK and stop bit), SSPBUF empty.

SSPCON: SSP control register (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7	WCOL:	Write conflict detection bit.
	master control mode:	1= Trying to write to the SSPBUF register when I^{2}C does not meet the condition of starting transmit data.
		0= no conflict.
	Slave mode:	
		1= While transmitting the previous word, write the SSPBUF register again (must clear through software).
		0= no conflict.
Bit6	SSPOV:	Receive overflow flag bit. (only valid in slave receive mode)
	1=	When the SSPBUF register still maintains the previous data, it receives a new byte. In the transmit mode, the SSPOV bit can be any value (this bit must be clear through software).
	0=	No overflow.
Bit5	SSPEN:	Synchronous serial port enable bit (These pins must be correctly configured as input or output pins).
	1=	Enable serial port and configure SDA and SCL pin as serial port pin.
	0=	Disable serial port and configure these pins as I/O port pins.
Bit4	CKP:	Clock polarity selection bit.
	In I^{2}C slave mode:	SCK release control.
		1 = enable clock.
		0 = Keep clock line is low (clock extension) (used to ensure data establishment time).
	In I^{2}C master control mode:	Not used.
Bit3~Bit0	SSPM<3:0>:	Synchronous serial port mode selection bit.
	0000=	SPI master control mode, clock= $F_{SYS}/4$.
	0001=	SPI master control mode, clock= $F_{SYS}/16$.
	0010=	SPI master control mode, clock= $F_{SYS}/64$.
	0011=	SPI master control mode, clock= TMR2 output/2.
	0100=	SPI slave mode, clock= SCKpin, enable SS pin control.
	0101=	SPI slave mode, clock= SCKpin, disable SS pin control, SS can be used as I/O pin.
	0110=	save.
	0111=	save.
	1000=	I^{2}C master control mode, clock= $F_{SYS}/$ (4 * (SSPADD+1)).
	1001=	Disable load function.
	1010=	save.
	1011=	save.
	1100=	save.
	1101=	save.
	1110=	I^{2}C slave mode, 7-bit address, allow start bit and stop bit interrupt.
	1111=	save.

SSPCON2: SSP control register2 (91H)

91H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
read/write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Reset value	0	1	0	0	0	0	0	0

Bit7	GCEN:	Broadcast call enable bit (only in I^2C slave mode).
	1=	It is allowed to generate interrupt when receiving to the general call address (0000h) in SSPSR.
	0=	Disable broadcast call address.
Bit6	ACKSTAT:	ACK status bit (only in I^2C master control mode).
	In master control transmit mode:	
		1 = Did not receive a response from the slave device.
		0 = A response from the slave device has been received.
Bit5	ACKDT:	ACK data bit (only in I^2C master control mode).
	In master control receive mode:	The value of the user's response sequence after the receive is completed.
		1 = not respond.
		0 = respond.
Bit4	ACKEN:	ACK enable bit (only in I^2C master control mode).
	In master control receive mode:	
		1 = Start the response sequence on the SDA and SCL pin, transmit ACKDT data bit. Automatically cleared by hardware.
		0 = Response sequence idle.
Bit3	RCEN:	Receive enable bit (only in I^2C master control mode).
	1=	Enable I^2C receive mode.
	0=	Receive idle.
Bit2	PEN:	stop enable bit (only in I^2Cmaster controlmode).
		1 = Start stop condition on SDA and SCL pin. Automatically cleared by hardware.
		0 =idle.
Bit1	RSEN:	Repeat enable bit (only in I^2Cmaster controlmode).
	1=	Initiate repeated start conditions on the SDA and SCL pins. Automatically cleared by hardware.
	0=	idle.
Bit0	SEN:	Start enable bit.
	In master control mode:	
		1 = Start the start conditions on the SDA and SCL pins. Automatically cleared by hardware.
		0 = idle.
	In slave mode:	
		1 = Both transmit and receive will enable clock extension (enable clock extension).
		0 = disable clock extention.

15.3.2 Master Control Mode

The master control mode works by generating interrupt when the start and stop conditions are detected. The stop (P) bit and the start (S) bit are cleared when reset or disable MSSP mod. When the P bit is set to 1, the control of I2C bus can be obtained; otherwise the bus is idle, and both the P and S bits are zero.

In master control mode, the SCL line is manipulated by the MSSP hardware, and SDA pin must be configured as input (the corresponding pin TRIS bit is set to 1). The following events will set the MSSP interrupt flag bit SSPIF to 1 (if MSSP interrupt is allowed, interrupt will be generated):

- Start condition
- Stop condition
- Data transmission byte has been transmitted/received
- Reply to transmit
- Repeated start conditions

15.3.3 I^{2}C Master Control Mode Support

The master control mode can be enabled by setting the corresponding SSPM bit in SSPCON to 1 or clearing it and setting the SSPEN bit to 1. Once the master control mode is enabled, the user can select the following 6 operations:

1. Issue a start condition on SDA and SCL.
2. Issue a repeated start condition on SDA and SCL.
3. Write the SSPBUF register to start data/address transmit.
4. Generate a stop condition on SDA and SCL.
5. Configure the I2C port to receive data.
6. The response condition is generated after the data byte is received.

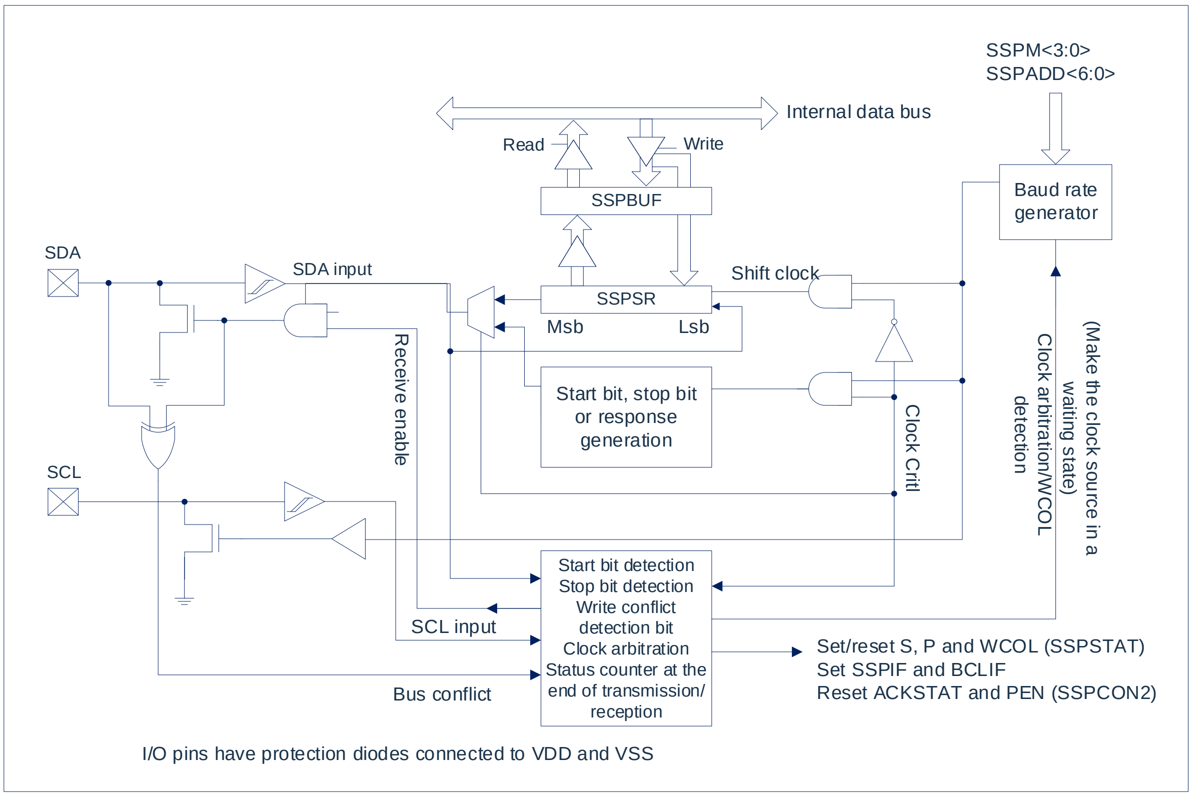

SSPM<3:0>
SSPADD<6:0>
Internal data bus
Read
Write
SSPBUF
Baud rate generator
SDA
SDA input
Shift clock
SSPSR
Msb
Lsb
Receive enable
Start bit, stop bit or response generation
Clock Crit
(Make the clock source in a waiting state)
Clock arbitration/WCOL detection
SCL
Start bit detection
Stop bit detection
Write conflict detection bit
Clock arbitration
Status counter at the end of transmission/ reception
SCL input
Bus conflict
Set/reset S, P and WCOL (SSPSTAT)
Set SSPIF and BCLIF
Reset ACKSTAT and PEN (SSPCON2)
I/O pins have protection diodes connected to VDD and VSS

Fig 15-6: MSSP block diagram (I^{2}C^{TM} master control mode)

Note: When configured as I^{2}C master mode, MSSP module does not allow event queuing. For example, before the end of the start condition, the user is not allowed to issue another start condition and write to the SSPBUF register immediately to initiate the transfer. In this case, SSPBUF will not be written and the WCOL bit will be set to 1, which indicates that no write operation to SSPBUF has occurred.

15.3.3.1 I^{2}C Master Control Mode Opeartion

All serial clock pulses and start/stop conditions are generated by the master device. The stop condition or the repeated start condition can end the transmission. Because the repeated start condition is also the beginning of the next serial transmission, the I^{2}C bus will not be released. In the master control transmit mode, the serial data is output through SDA, and the serial clock is output by SCL. The first byte of the transmit includes the address (7 bits) and read/write (R/W) bits of the receiver. In this case, R/W bit will be logic 0. Serial data transmits 8 bits each time. Every time a byte is transmitted, an acknowledge bit will be received. The output of the start and stop conditions indicates the start and end of the serial transmission.

In master control receive mode, the first byte of transmit includes the address (7 bits) of the transmit device and the R/W bit. In this case, the R/W bit will be logic 1. Therefore, the first byte of transmit byte is a 7-bit slave device address, followed by 1 to indicate receive. The serial data is received through SDA, while the serial clock is output by SCL. Every time 8 bits of serial data are received. Every time a byte is received, an answer bit will be transmitted. Start and stop conditions indicate the start and end of transmit, respectively.

In I^{2}C mode, the baud rate generator used in SPI mode is used to set the SCL clock frequency to 100KHz, 400KHz or 1MHz. The reload value of the baud rate generator is located in the lower 7 bits of the SSPADD register. When a write to SSPBUF occurs during operation, the baud rate generator will automatically start counting. If the specified operation is completed (ie, the last data bit of transmit is followed by ACK), the internal clock will automatically stop counting, and the SCL pin will remain in its last state.

The following is a typical transmit event sequence:

- The user generates a start condition by setting the start enable bit SEN (SSPCON2 register) to 1.
- SSPIF set to 1. Before performing any other operations, MSSP mod will wait for the required startup time.
- The user will load the SSPBUF from the device address to transmit.
- The address is moved out of the SDA pin until all 8 bits are transmitted.
- MSSP mod shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
- MSSP mod sets the SSPIF bit to 1 at the end of the 9th clock period, generating an interrupt.
- The user loads 8-bit data into SSPBUF.
- Data is moved out from the SDA pin until all 8 bits are transmitted.
- MSSP mod shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
- MSSP mod sets the SSPIF bit to 1 at the end of the 9th clock, generating an interrupt.
- The user generates a stop condition by setting the stop enable bit (PEN) bit (SSPCON2 register) to 1.
- Once the stop condition is completed, an interrupt will be generated.

15.3.4 Baud Rate Generator

In I^{2}C master control mode, the baud rate generator reloaded value is located in the lower 7 bits of the SSPADD register (Figure 15-7). When the value is loaded, the baud rate generator will automatically start counting and decrement to 0, and then stop until the next reload. BRG will count down twice on the Q2 and Q4 clock periods in each instructions period (TCY). In I^{2}C master control mode, BRG will be automatically reloaded. For example, when clock arbitration occurs, BRG will be reloaded when SCL pin is sampled to high level (Figure 15-8).

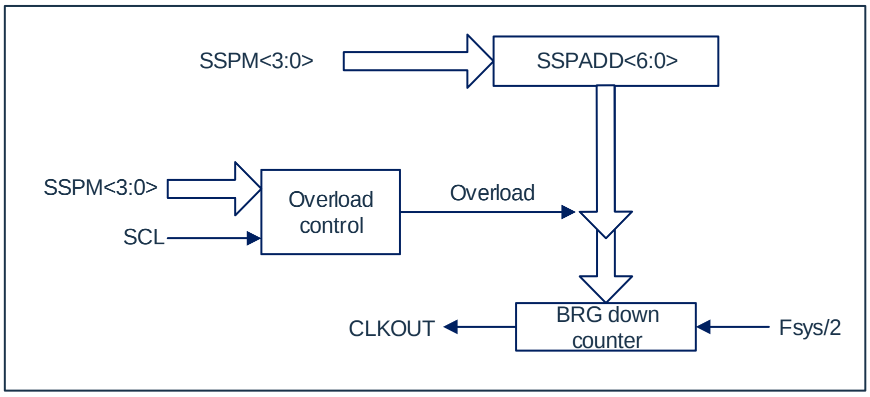
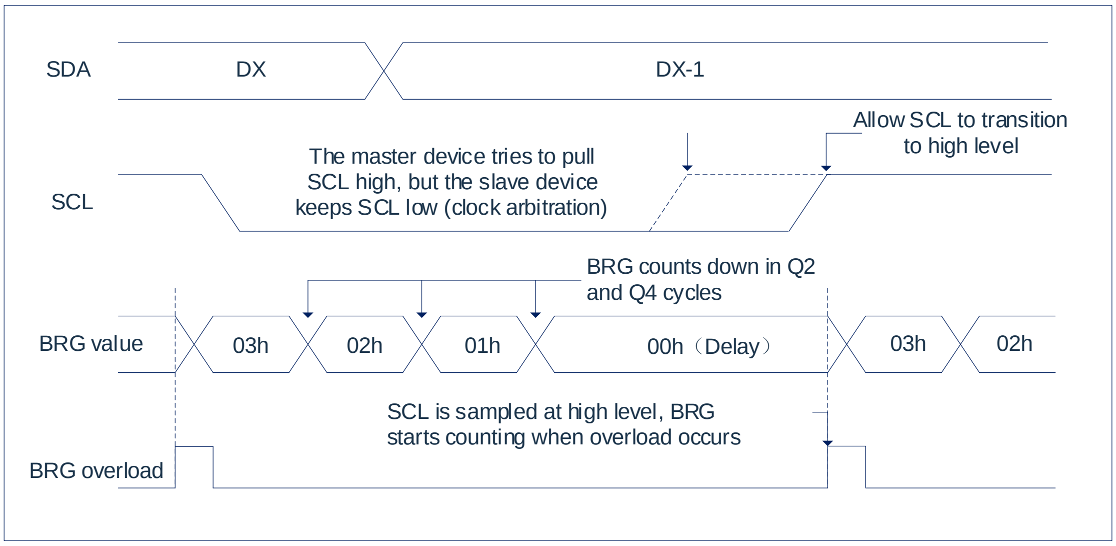


Fig 15-7: baud rate generator block diagram


SDA
DX
DX-1
SCL
The master device tries to pull SCL high, but the slave device keeps SCL low (clock arbitration)
Allow SCL to transition to high level
BRG counts down in Q2 and Q4 cycles
BRG value
03h
02h
01h
00h（Delay）
03h
02h
SCL is sampled at high level, BRG starts counting when overload occurs
BRG overload
Fig 15-8: Time seires of baud rate generator with clock arbitration

15.3.5 I^{2}C Master Control Mode Transmit

Transmit a data byte and a 7-bit address can be achieved directly by writing a value to the SSPBUF register. This operation will set the buffer full flag bit BF to 1, and the baud rate generator will start counting, and at the same time start the next transmit. After the falling edge of SCL is valid, each bit of address/data will be shifted out to the SDA pin. In a baud rate generator full return count period (TBRG), SCL remains low. Data should be released to high level in SCL when SCL pin is released to high level, it will remain high for the entire TBRG. During this period and a period of time after the next SCL falling edge, the data on the SDA pin must remain stable. After the 8th bit is shifted out (the falling edge of the 8th clock period), the BF flag bit is cleared, and the master device releases SDA.

At this time, if an address match occurs or data is correctly received, the addressed slave device will respond with an ACK bit at the 9th bit time. The ACK status is written to the ACKDT bit at the falling edge of the 9th clock period. After master device receiving the response, the response status bit ACKSTAT will be cleared; if the response is not received, the bit will be set to 1. After the 9th clock, the SSPIF bit will be set to 1, and the master control clock (baud rate generator) will be suspended until until the next data byte is loaded into SSPBUF, SCLpin remains low and SDA remains unchanged.

After writing SSPBUF, each bit of address is shifted out on the falling edge of SCL until all 7 bits of address and R/W bit are shifted out. At the falling edge of the eighth clock, the master device pulls the SDA pin to high level to allow the slave device to send an acknowledgment response. At the falling edge of the 9th clock, the master device determines whether the address is recognized by the slave device by sampling the SDA pin. The status of the ACK bit is loaded into the ACKSTAT status bit (SSPCON2 register). After the 9th clock falling edge of the transmit address, SSPIF is set to 1, the BF flag bit is cleared, the baud rate generator is turned off until the next write operation to SSPBUF, and the SCL pin remains low, allowing the SDA pin to suspend.

15.3.5.1 BF Status Indication

In transmit mode, the BF bit (SSPSTAT register) is set to 1 when the CPU writes SSPBUF, and is cleared after all 8 bits of data are shifted out.

15.3.5.2 WCOL Status Indication Bit

If the user writes SSPBUF during the transmit process (that is, when the SSPSR is still moving out of the data byte), WCOL is set to 1 and the contents of the buffer remain unchanged (no write operation has occurred). WCOL must clear through software.

15.3.5.3 ACKSTAT Status Indication

In transmit mode, when the slave device transmits a response (ACK=0), the ACKSTAT bit (SSPCON2 register) is cleared; when the slave device does not respond (ACK=1), the bit is 1. The slave device recognizes its address (Including the broadcast call address) or after receiving the data correctly, a response will be transmitted.

15.3.6 I^{2}C Master Control Mode Receive

By programming receive enable bit RCEN (SSPCON2 register) to enable master control mode receive. The baud rate generator starts counting, and each time the count returns, the state of the SCL pin changes (from high to low or from low to high), and data is shifted into SSPSR. After the falling edge of the eighth clock, the receive enable flag bit is automatically cleared, the content of SSPSR is loaded into SSPBUF, the BF flag bit is set to 1, the SSPIF flag bit is set to 1, the baud rate generator pauses counting, and the SCL remains at low level. At this time, MSSP is in idle state, waiting for the next command. When the CPU reads the buffer, the BF flag bit will be automatically cleared. By setting the response sequence enable bit ACKEN (SSPCON2 register) to 1, the user can end the receive transmit response bit.

15.3.6.1 BF Status Indication

When receiving, when the address or data byte is loaded from SSPSR into SSPBUF, the BF bit is set to 1, and the BF bit is cleared when reading the SSPBUF register.

15.3.6.2 WCOL Status Indication

If the user writes SSPBUF during the receive process (that is, when the SSPSR is still moving into the data byte), the WCOL bit is set to 1, and the buffer content remains unchanged (no write operation has occurred).

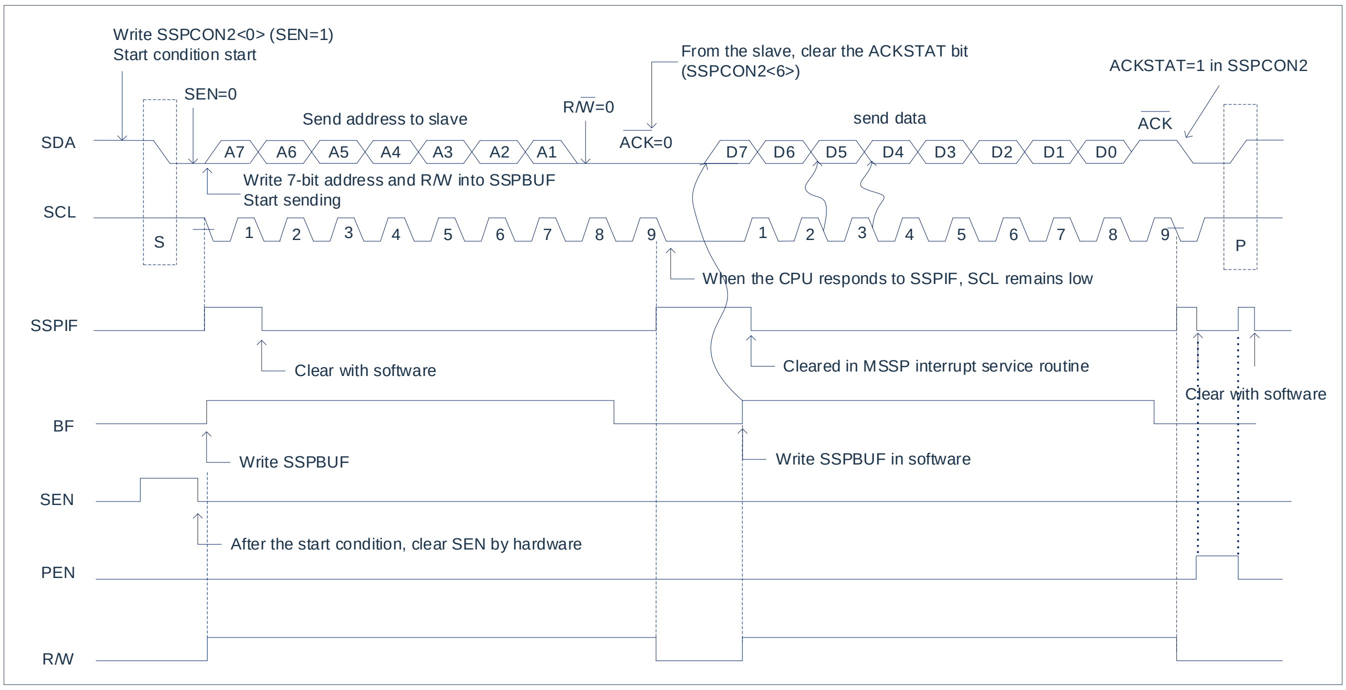


Fig15-9: time seires of I^{2}C^{TM} master control mode transmit

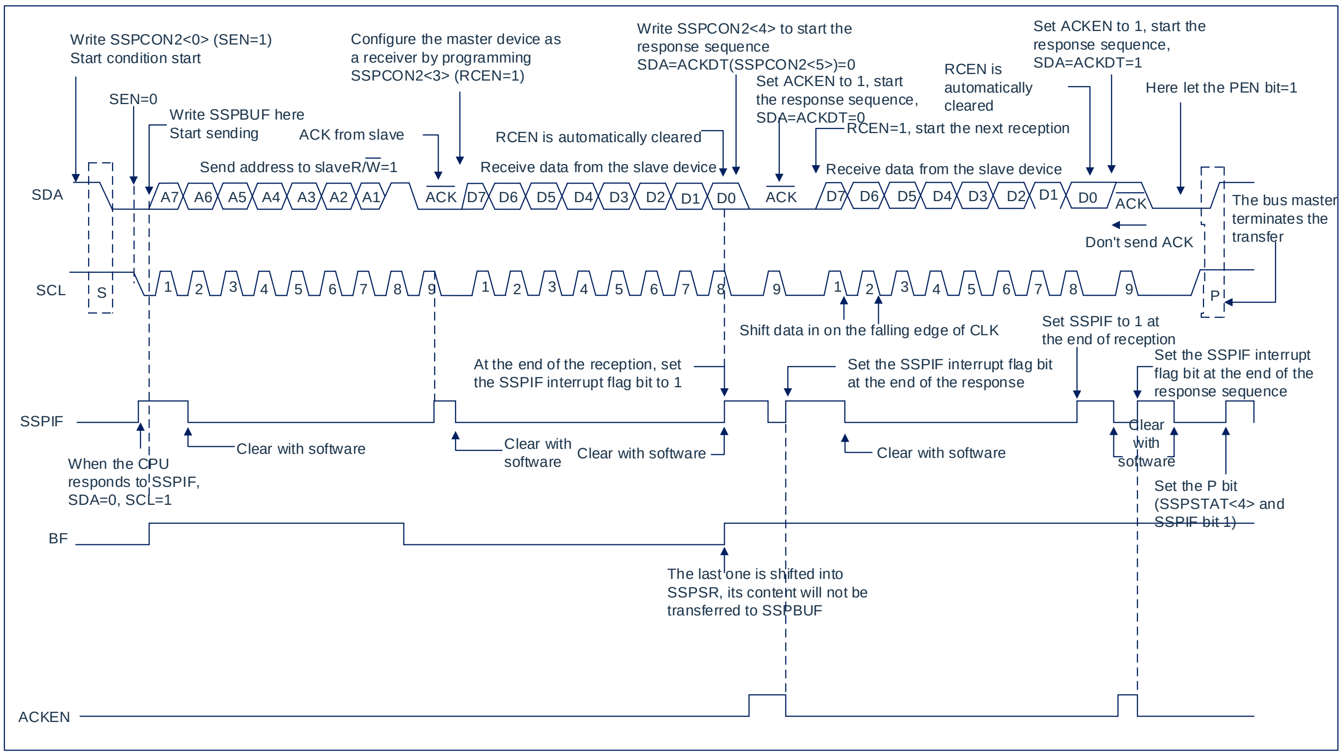

Write SSPCON2<0> (SEN=1)
Start condition start
SEN=0
Write SSPBUF here
Start sending
ACK from slave
Send address to slaveR/W=1
Configure the master device as a receiver by programming SSPCON2<3> (RCEN=1)
RCEN is automatically cleared
Receive data from the slave device
Write SSPCON2<4> to start the response sequence SDA=ACKDT(SSPCON2<5>)=0
Set ACKEN to 1, start the response sequence, SDA=ACKDT=0
RCEN is automatically cleared
RCEN=1, start the next reception
Receive data from the slave device
Set ACKEN to 1, start the response sequence, SDA=ACKDT=1
Here let the PEN bit=1
The bus master terminates the transfer
Don't send ACK
SDA
SCL
S
P
A7 A6 A5 A4 A3 A2 A1 ACK D7 D6 D5 D4 D3 D2 D1 D0 ACK D7 D6 D5 D4 D3 D2 D1 D0 ACK
1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9
Shift data in on the falling edge of CLK
At the end of the reception, set the SSPIF interrupt flag bit to 1
Set the SSPIF interrupt flag bit at the end of the response
Set SSPIF to 1 at the end of reception
Set the SSPIF interrupt flag bit at the end of the response sequence
SSPIF
Clear with software
When the CPU responds to SSPIF, SDA=0, SCL=1
Clear with software
Clear with software
Clear with software
Clear with software
Set the P bit (SSPSTAT<4> and SSPIF bit 1)
BF
The last one is shifted into SSPSR, its content will not be transferred to SSPBUF
ACKEN

Fig 15-10: Time series of I^{2}C^{TM} master control mode receive (7-bit address)

15.3.7 I^{2}C Master Control Mode Start Condition Time Series

To initiate a start condition, the user should set the start condition enable bit SEN of the SSPCON2 register to 1. When both SDA and SCL pins are sampled as high, the baud rate generator reloads the contents of SSPADD<6:0> and starts counting. When the baud rate generator timeout (TBRG) occurs, if both SCL and SDA are sampled as high level, the SDA pin is low level by the driver. When SCL is high level, setting the SDA driver to low level is the startup condition. Set the S bit (SSPSTAT register) to 1. Then the baud rate generator reloads the contents of SSPADD<6:0> and resumes counting. When the baud rate generator times out (TBRG), the SEN bit of the SSPCON2 register will be automatically cleared by hardware. The baud rate generator is pasued, the SDA line remains low, and the start condition ends.

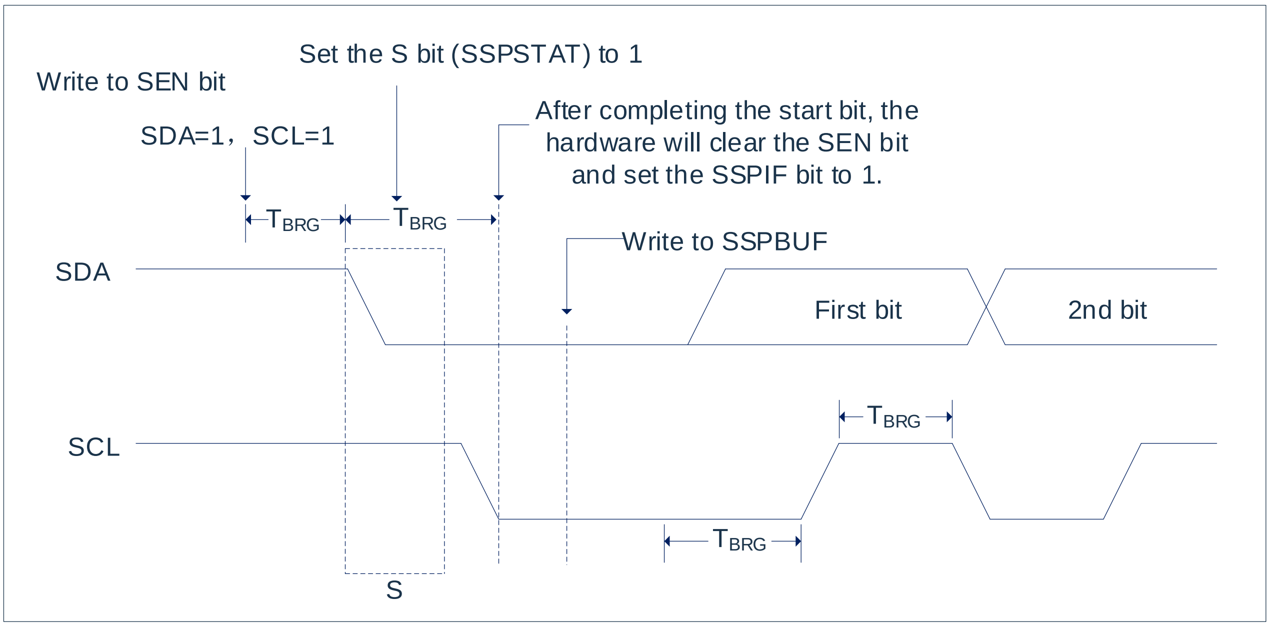

Set the S bit (SSPSTAT) to 1
Write to SEN bit
SDA=1，SCL=1
After completing the start bit, the hardware will clear the SEN bit and set the SSPIF bit to 1.
TBRG
TBRG
Write to SSPBUF
SDA
First bit
2nd bit
TBRG
SCL
TBRG
S

Fig 15-11： time series for the frist starting bit

15.3.7.1 WCOL Status Indication

When the startup sequence is in progress, if the user writes SSPBUF, WCOL is set to 1, and the buffer content remains unchanged (no write operation has occurred).

Note: Since event queues are not allowed, the lower 5 bits of SSPCON2 cannot be written before the start condition ends.

15.3.8 I^{2}C Master Control Mode Repeat Condition Time Series

When the RSEN bit (SSPCON2 register) is programmed to be high and the I2C logic mod is in an idle state, a repeated start condition will occur. When the RSEN bit is 1, the SCL pin is pulled low. When the SCL pin is sampled low, baud rate generator loads the contents of SSPADD<6:0> and starts counting. In a baud rateg enerator counting period (TBRG), the SDA pin is released (its pin level is pulled high). When baud rate generator timeout, if SDA is sampled as high, SCL pin will be pulled high. When SCL pin is sampled as high, the baud rate generator will be reloaded into the contents of SSPADD<6:0> and start counting. SDA and SCL must be in one count period TBRG and sampled as high level. Then the SDA pin is pulled low (SDA = 0) and keeps a count period TBRG while SCL is high level. Then the RSEN bit (SSPCON2 register) will be automatically cleared, The baud rate generator will not be reloaded, and the SDA pin remains low. Once the start condition is detected on the SDA and SCL pins, the S bit (SSPSTAT register) will be set to 1. The SSPIF bit will not be set to 1 until the baud rate generator times out.

Once the SSPIF bit is set to 1, the user can write the 7-bit address into SSPBUF. When the first 8 bits are transmitted and an ACK is received, the user can transmit 8-bit data.

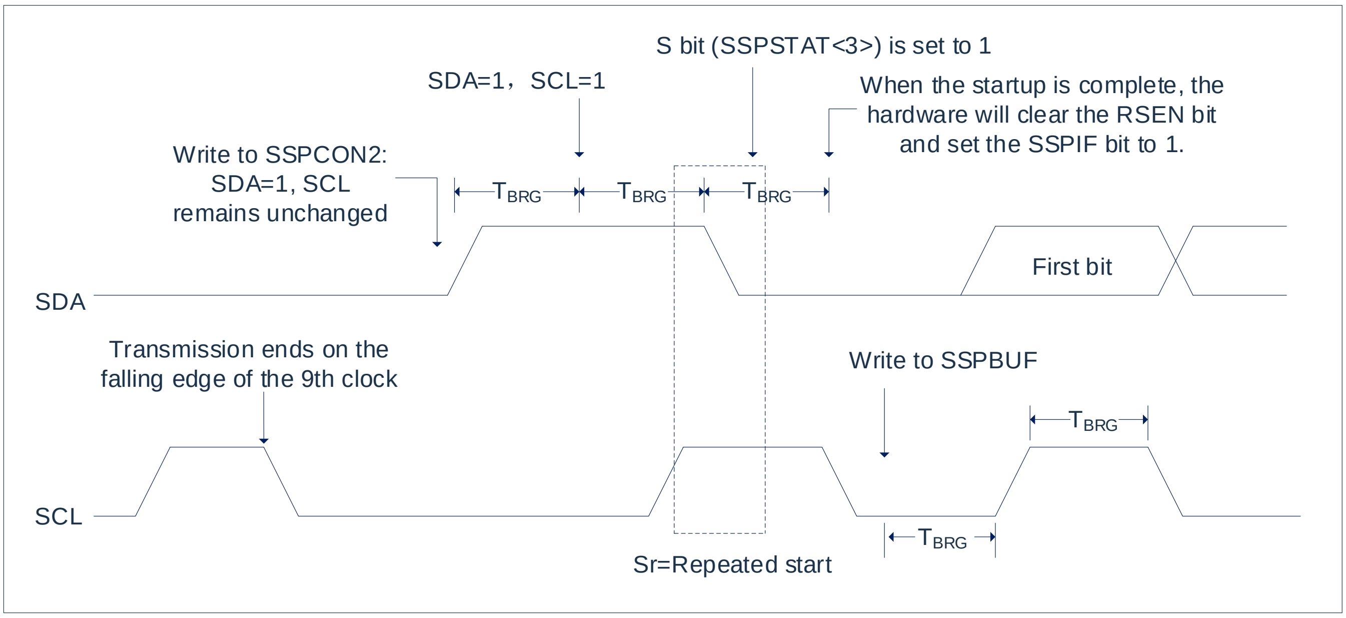

S bit (SSPSTAT<3>) is set to 1
SDA=1，SCL=1
When the startup is complete, the hardware will clear the RSEN bit and set the SSPIF bit to 1.
Write to SSPCON2: SDA=1, SCL remains unchanged
TBRG
TBRG
TBRG
First bit
SDA
Transmission ends on the falling edge of the 9th clock
Write to SSPBUF
TBRG
SCL
TBRG
Sr=Repeated start

Fig 15-12: time series of repreat condition

15.3.8.1 WCOL Status Indication

When the repeated start sequence is in progress, if the user writes SSPBUF, WCOL is set to 1, and the buffer content remains unchanged (no write operation has occurred).

Note: As events are not allowed to be queued, the lower 5 bits of SSPCON2 cannot be written until the repeated start condition ends.

15.3.9 ACK Time Series

Set the ACK enable bit ACKEN (SSPCON2register) to 1 to enable the acknowledgement. When this bit is set to 1, the SCL pin is pulled low, and the content of the ACK data bit appears on the SDA pin. If the user wants to generate a response, it should clear the ACKDT bit to zero; otherwise, the user should set the ACKDT bit to 1 before the start of the ACK. Then the baud rate generator counts the full return period (TBRG), and then the SCL pin level is pulled high. When the SCL pin is sampled as at high level (clock arbitration), baud rate generator counts for another TBRG period. Then SCLpin is pulled low. After that, the ACKEN bit is automatically cleared, baud rate generator is turned off, and MSSP mod enters idle mode.

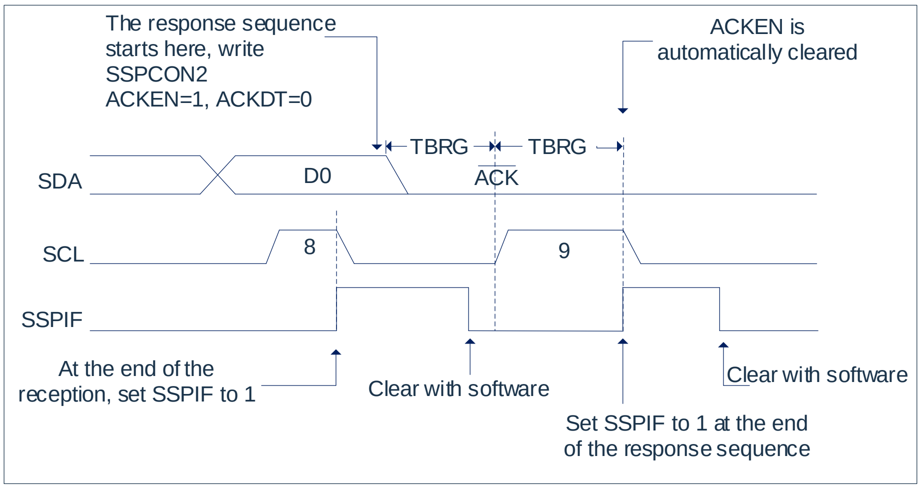

The response sequence starts here, write SSPCON2 ACKEN=1, ACKDT=0
ACKEN is automatically cleared
TBRG
TBRG
SDA
D0
$\overline{ACK}$
SCL
8
9
SSPIF
At the end of the reception, set SSPIF to 1
Clear with software
Clear with software
Set SSPIF to 1 at the end of the response sequence

Fig 15-13: times series for ACK

Note: T_{BRG}= 1 baud rate generator period.

15.3.9.1 WCOL Status Indication Bit

If the user writes SSPBUF while the ACK sequence is in progress, WCOL will be set to 1 and the contents of the buffer will remain unchanged (no write operation has occurred).

15.3.10 Stop Condition

At the end of receive/transmit, by setting the enable bit of the stop sequence, PEN (SSPCON2 register), the SDA pin will generate a stop bit. At the end of receive/transmit, the SCL pin will remain low after the falling edge of the 9th clock Level. When the PEN bit is 1, the master control device sets SDA low. When the SDA line is sampled low, the baud rate generator is reloaded with the value and counts down to 0. When the baud rate generator times out, The SCL pin is pulled to a high level, and after a TBRG (baud rategenerator counts back to zero), SDA pin is pulled to a high level again. When SDA pin is sampled as high and SCL is also high, the P bit (SSPSTAT register) set to 1. After a TBRG period, the PEN bit is cleared and the SSPIF bit is set to 1.

15.3.10.1 WCOL Status Indication

If the user attempts to write SSPBUF during the stop sequence, the WCOL bit will be set to 1, and the contents of the buffer will not change (no write operation has occurred).

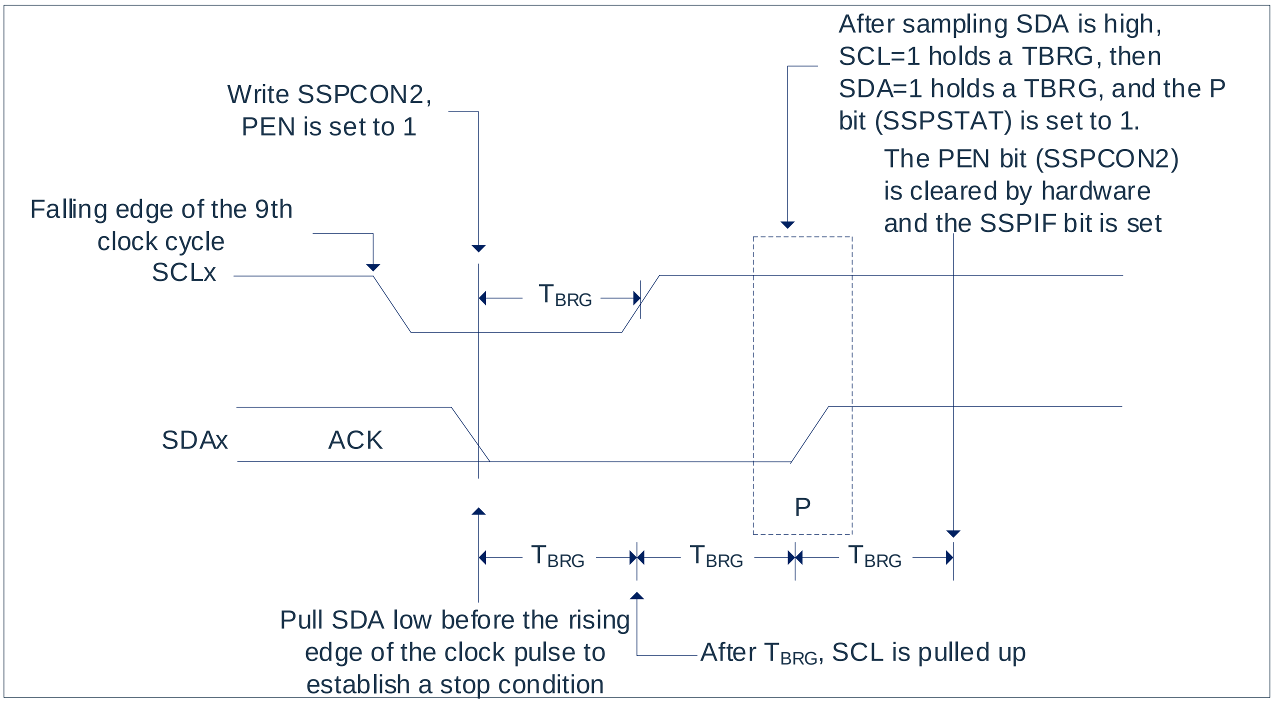

After sampling SDA is high, SCL=1 holds a TBRG, then SDA=1 holds a TBRG, and the P bit (SSPSTAT) is set to 1.
Write SSPCON2, PEN is set to 1
The PEN bit (SSPCON2) is cleared by hardware and the SSPIF bit is set
Falling edge of the 9th clock cycle
SCLx
T_{BRG}
SDAx
ACK
P
T_{BRG}
T_{BRG}
T_{BRG}
Pull SDA low before the rising edge of the clock pulse to establish a stop condition
After T_{BRG}, SCL is pulled up

Fig15-14： stop condition receive or transmit mode

Note: T_{BRG}=1 baud rate generator period.

15.3.11 Clock Arbitration

If during any receive, transmit, or repeated start/stop conditions, the master device pulls up the SCL pin (allowing the SCL pin to float high), clock arbitration will occur. If the SCL pin is allowed to float high, the baud rate generator (BRG) will pause counting until the SC L pin is actually sampled high. When the SCL pin is sampled high, the baud rate generator will be reloaded with the contents of SSPADD<6:0> and start counting. This can ensure that when the external device pulls the clock low, the SCL always maintains high for at least one BRG full return period.

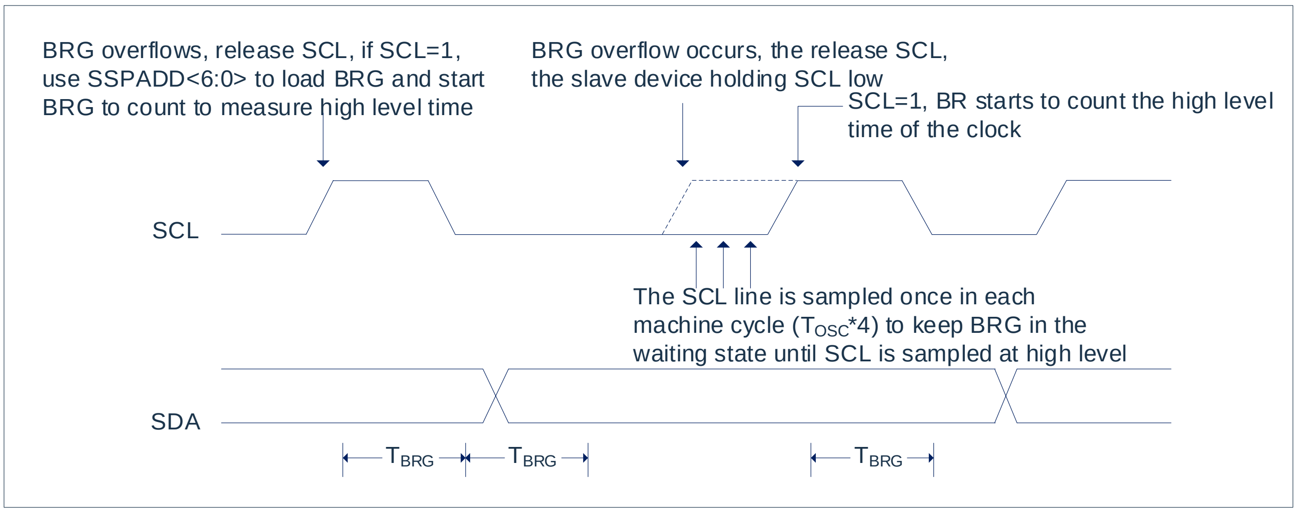

BRG overflows, release SCL, if SCL=1, use SSPADD<6:0> to load BRG and start BRG to count to measure high level time
BRG overflow occurs, the release SCL, the slave device holding SCL low
SCL=1, BR starts to count the high level time of the clock
SCL
The SCL line is sampled once in each machine cycle (TOSC*4) to keep BRG in the waiting state until SCL is sampled at high level
SDA
TBRG
TBRG
TBRG

Fig 15-15: clock arbitration in master control transmit mode

15.3.12 Multi Master Mode

In multi-master mode, it can be determined when the bus is free by generating interrupt when the start and stop conditions are detected. The stop (P) bit and the start (S) bit are cleared when reset or disable MSSP mod. When the P bit is set to 1, you can get control of the I^{2}C bus; otherwise, the bus is in an idle state, and the P and S bits are cleared. When the bus is busy, if a stop condition occurs, an interrupt will be generated (if MSSP interrupt is allowed).

When working in multi-master mode, you must monitor the SDA line for arbitration to see if the signal level is the expected output level. This check is done by hardware, and the result is placed in the BCLIF bit.

Arbitration may fail under the following conditions:

- address transmission
- data transmission
- Start condition
- Repeated start condition
- ACK conditions

15.3.13 Multi Master Communication, Bus Conflict and Bus Arbitration

Multi-master mode is supported by bus arbitration. When the master device outputs the address/data bit to the SDA pin, if one master device outputs 1 on SDA by floating the SDA pin to high level, and the other master device outputs 0, bus arbitration will occur. If the expected data on the SDA pin is 1, and the data actually sampled on the SDA pin is 0, a bus conflict has occurred. The master device will set the bus conflict interrupt flag bit BCLIF to 1, and reset the I^2C port to idle state.

If a bus conflict occurs during the transmit process, the transmit stops, the BF flag bit is cleared, the SDA and SCL lines are pulled high, and SSPBUF is allowed to be written. After the bus conflict interrupt service program is executed, if the I^2C bus is free, user can resume communication by issuing a start condition. If a bus conflict occurs during the start, repeated start, stop, or response condition, the condition is aborted, the SDA and SCL lines are pulled high, and the corresponding control bit in the SSPCON2 register is cleared. After executing the bus conflict interrupt service program, if the I^2C bus is free, the user can resume communication by issuing a start condition. The master device will continue to monitor SDA and SCL pin. If a stop condition occurs, the SSPIF bit will be set to 1. No matter what bus occurs What is the progress of the transmit during conflict, writing SSPBUF will start transmitting data from the first data bit.

In multi-master mode, the interrupt can be generated when the start and stop conditions are detected to determine when the bus is free. When the P bit is set to 1, you can obtain control of the I2C bus, otherwise the bus is free and the S and P bits are cleared.

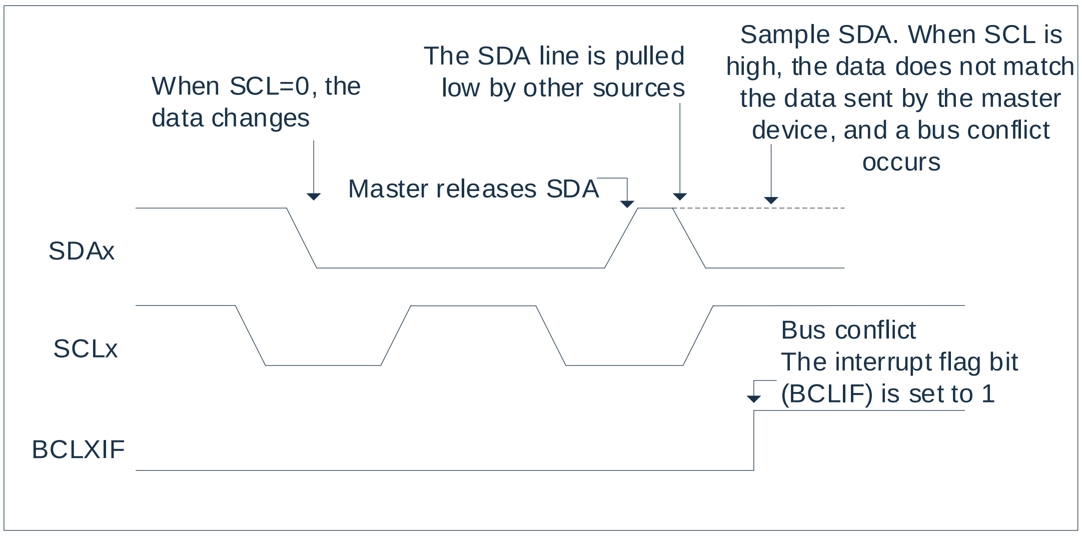

When SCL=0, the data changes
The SDA line is pulled low by other sources
Sample SDA. When SCL is high, the data does not match the data sent by the master device, and a bus conflict occurs
Master releases SDA
SDAx
SCLx
Bus conflict The interrupt flag bit (BCLIF) is set to 1
BCLXIF

Fig 15-16: time series for transmit and ACK bus conflict

15.3.14 Slave Mode

In slave mode, SCL pin and SDA pin must be configured as input (TRISC<7:6> is set to 1). When needed (such as from the transmitter), the MSSP mod will use output data to rewrite the input state.

When the address matches or the data transmitted after the address matches is received, the hardware will automatically generate an acknowledge (ACK) pulse, and load the data received in the SSPSR register at the time into the SSPBUF register.

As long as one of the following conditions is met, MSSP mod will not generate this ACK pulse:

-The buffer full flag bit BF (SSPCON register) is 1 before the received data to be transmitted.

-Before receiving the transmitted data, the overflow flag bit SSPOV (SSPCON register) has been set 1.

In this case, the value of SSPSR register will not be loaded into SSPBUF, but the SSPIF bit of PIR1 register will be set to 1. The BF bit is cleared by reading the SSPBUF register, and the SSPOV bit is cleared by software.

To ensure normal operation, SCL clock input must meet the minimum high-level time and minimum low-level time requirements.

15.3.14.1 Addressing

Once MSSP mod is enabled, it will wait for the start condition to be generated. After the start condition occurs, 8 bits of data are shifted into the SSPSR register. All input bits are sampled on the rising edge of the clock (SCL) line. RegisterSSPSR<7:1> The value will be compared with the value of the SSPADD register. The comparison is performed on the falling edge of the 8th clock pulse (SCL). If the address matches and the BF bit and SSPOV bit are zero, the following events will occur:

- The value of SSPSR register is loaded into SSPBUF register.
- The buffer full flag bit BF is set to 1.
- Generate ACK pulse.
- On the falling edge of the 9th SCL pulse, the MSSP interrupt flag bit SSPIF of the PIR1 register is set to 1 (interrupt is generated if interrupt is allowed)

15.3.14.2 Receive

When the R/W bit of the address byte is cleared and an address match occurs, the R/W bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register.

When there is an address byte overflow condition, an acknowledge pulse (ACK) will not be generated. The overflow condition means that the BF bit (SSPSTAT register) is set to 1, or the SSPOV bit (SSPCON register) is set to 1. Each data transmission byte will generate an MSSP interrupt. The interrupt flag bit SSPIF of the PIR1 register must be cleared by software. The SSPSTAT register is used to determine the status of the byte.

15.3.14.3 Transmit

When the R/W bit of the received addressbyte is 1 and an address match occurs, the R/W bit of the SSPSTAT register is 1. The received address is loaded into the SSPBUF register. The ACK pulse is transmitted on the 9th bit while the SDA pin remains low. The transmitted data must be loaded into the SSPBUF register and also into the SSPSR register. Then the CKP bit (SSPCON register) should be set to 1 to enable the SCL pin. Before transmitting another clock pulse, the master control device must monitor the SCL pin. The slave device can suspend the data transmission with the master control device by extending the clock. 8 data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high level.

Each byte of data transmission will generate an MSSP interrupt. The SSPIF flag bit must be clear through software, and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set at the falling edge of the 9th clock pulse. The ACK pulse from the main receiver will be latch on the rising edge of the 9th pulse of SCL input. If the SDA line is high (no ACK), then the data transfer has been completed. In this case, if the slave device latches the ACK, reset the slave logic (Reset SSPSTAT register), while the slave device monitors the appearance of the next start bit. If the SDA line is low (ACK), then the data to be transmitted must be loaded into the SSPBUF register, which will also load the SSPSR register. CKP should be set 1 to enable SCL.

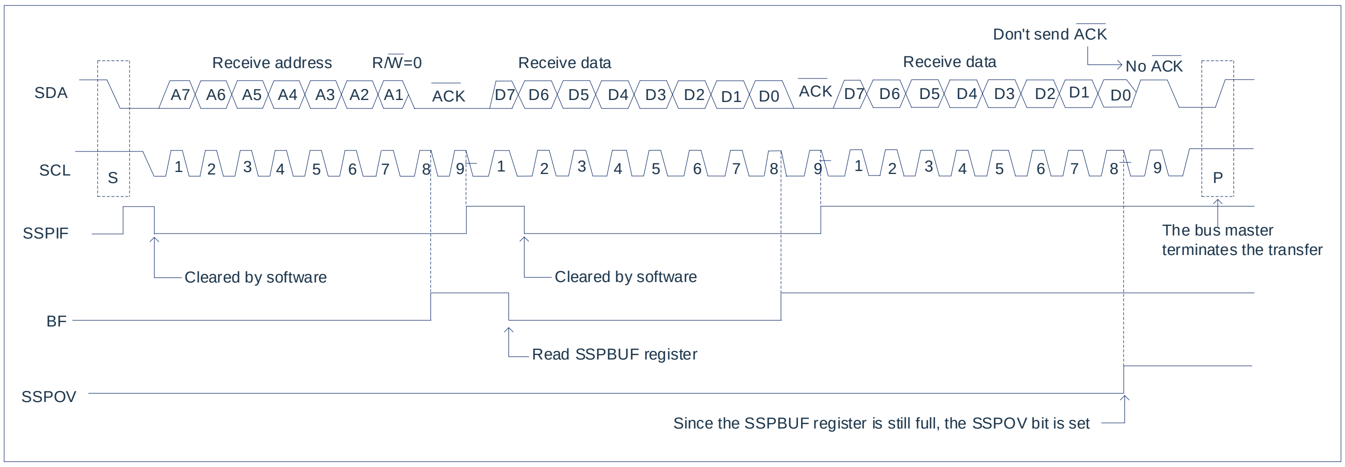


Fig 15-17: Time series for I^{2}C^{TM} slave mode receive (7-bit address)

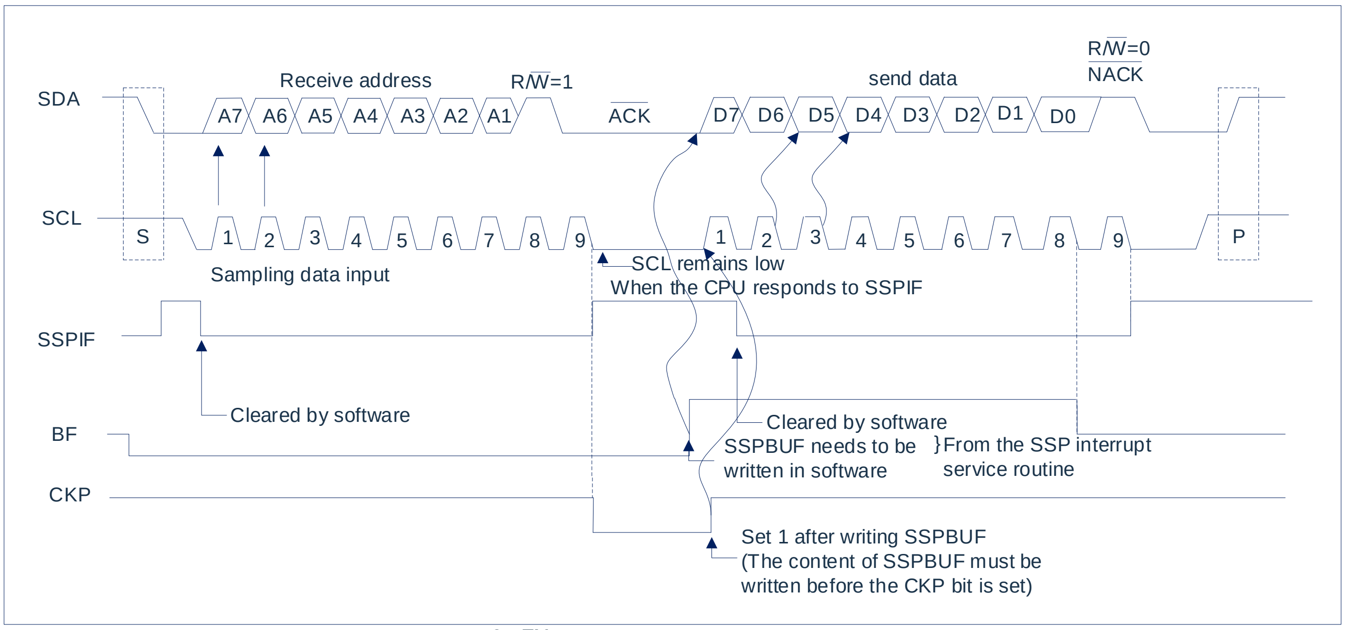


Fig 15-18: I^2C^{TM} slave mode transmit (7-bit address)

15.3.15 SSP Masking Register

In I^{2}C slave mode, the SSP mask (SSPMSK) register is used to mask the value in the SSPSR register under the address compare operation. A bit of 0 in the SSPMSK register will make the corresponding bit in the SSPSR register a "don't care".

This register is reset to all 1s when any reset condition occurs. Therefore, it has no effect on the standard SSP operation before writing the mask value. The register must be initialized before selecting the I^{2}C slave mode by setting the SSPM<3:0> bits. This register can only be accessed after the appropriate mode is selected through the SSPM<3:0> bits of SSPCON.

The SSP masking register is valid in the following situations:

- 7-bit address mode: perform address compare with A<7:1>.
- 10-bit address mode: only perform address compare with A<7:0>

SSP masking is invalid during the period from receive to the first (high) byte of address.

SSPMSK: SSP masking register (93H) [1]

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0 [2]
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

Bit7~Bit1	MSK<7:1>:	Mask bit.
	1=	Bit n of the received address is compared with SSPADD<n> to detect the match of the I^{2}C address.
	0=	Bit n of the received address is not used to detect I^{2}C address matching.
Bit 0	not used.	

Note:
1) When the SSPCON bit SSPM<3:0> = 1001, any read or write operation to the SSPADDSFR address is performed through the SSPMSK register.
2) In all other SSP modes, this bit is invalid

15.3.16 Operation Under Sleep Mode

In sleep mode, I^{2}C mod cannot be used.

15.3.17 Effect of Reset

Reset will disable MSSP mod and terminate the current transmission.

16. Program EEPROM and Program Memory Control

16.1 General

The devices in this series have 8K words of program memory, the address range is from 000h to 1FFFh, which is read-only in all address ranges; the device has a 128-byte program EEPROM, and the address range is 0h to 07Fh, which is available in all address ranges. It can be read/write.

These memories are not directly mapped to the register file space, but indirectly addressed through the special function register (SFR). A total of 6 SFR registers are used to access these memories：

- EECON1
- EECON2
- EEDAT
- EEDATH
- EEADR
- EEADRH

When accessing the program EEPROM, the EEDAT register stores 8-bit read/write data, and the EEADR register stores the address of the program EEPROM unit being accessed.

When accessing the program memory of the device, the EEDAT and EEDATH register form a double byte word to save the 16-bit data to be read, and the EEADR and EEADRH register form a double byte word to save the 13-bit EEPROM cell address to be read.

Program memory allows reading in units of bytes. Program EEPROM allows byte read/write. A byte write operation can automatically erase the target cell and write new data (erase before writing).

The writing time is controlled by the on-chip timer. The writing and erasing voltages are generated by the on-chip charge pump, which is rated to work within the voltage range of the device for byte or word operations.

When the device is protected by code, the CPU can still continue to read/write the program EEPROM and program memory. When the code is protected, the device programmer will no longer be able to access the program EEPROM or program memory.

Note:
1) Program memory refers to ROM space, that is, the space where instructions code is stored, which can only be read; Program EEPROM is a space for storing user data, which can be read/write.
2) The normal writing voltage range of program EEPROM is 3.0V~5.5V, writing current is 20mA@VDD=5V.

16.2 Related Register

16.2.1 EEADR and EEADRH Register

The EEADR and EEADRH registers can address up to 128 bytes of program EEPROM or up to 8K bytes of program memory.

When the program memory address value is selected, the high byte of the address is written into the EEADRH register and the low byte is written into the EEADR register. When the program EEPROM address value is selected, only the low byte of the address is written into the EEADR register.

16.2.2 EECON1 and EECON2 Register

EECON1 is the control register to access the program EEPROM.

The control bit EEPGD determines whether to access program memory or program EEPROM. When this bit is cleared, as with reset, any subsequent operations will be performed on the program EEPROM. When this bit is set to 1, any subsequent operations will be performed on the program memory. Program memory is read-only.

The control bits RD and WR start reading and writing respectively. Software can only set these bits to 1 and cannot be cleared. After the read or write operation is completed, they are cleared by hardware. Since the WR bit cannot be cleared by software, it can be used to avoid accidentally terminating write operations prematurely.

-When WREN is set to 1, the program EEPROM is allowed to be written. When power is on, the WREN bit is cleared. When the normal write operation is LVR reset or WDT timeout reset interrupt, the WRERR bit will be set to 1. In these cases, after reset, the user can check the WRERR bit and rewrite the corresponding unit.

-When the write operation is completed, the interrupt flag bit EEIF in the PIR1 register is set to 1. This flag bit must be cleared by software.

EECON2 is not a physical register. Reading result of EECON2 is all 0s.

The EECON2 register is only used when executing the program EEPROM write sequence.

EEPROM data register EEDAT (10CH)

10CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0	EEDAT<7:0>:	To read or write the lower 8 bits of data from the program EEPROM, or read the lower 8 bits of data from the program memory.

EEPROM address register EEADR (10DH)

10DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	EEADR<7:0>:	Specify the lower 8 bits of address for program EEPROM read/write operations, or the lower 8 bits of address for program memory read operations.

EEPROM data register EEDATH (10EH)

10EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0	EEDATH<7:0>:	The upper 8 bits of data read from the program EEPROM/program memory.

EEPROM address register EEADRH (10FH)

10FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADRH	---	---	---	EEADRH4	EEADRH3	EEADRH2	EEADRH1	EEADRH0
read/write	---	---	---	R/W	R/W	R/W	R/W	R/W
reset value	---	---	---	0	0	0	0	0

Bit7~Bit5	not used，read 0.	
Bit4~Bit0	EEADRH<4:0>:	Specify the upper 5 address of the program memory read operation.

EEPROM control register EECON1 (11BH)

11BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD	---	EETIME1	EETIME0	WRERR	WREN	WR	RD
read/write	R/W	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	---	0	0	X	0	0	0

Bit7	EEPGD:	Program/program EEPROMselection bit；
	1=	Operate program memory；
	0=	Operate program EEPROM.
Bit6	not used	
Bit5~Bit4	EETIME[1:0]	Maximum programming waiting time；　**(For more EETIME information, please refer**
	00=	1.25ms
	01=	2.5ms (VDD=4.0~5.5V，suggested TEMP=0~85℃)
	10=	5ms
	11=	10ms (suggested other than 2.5ms)
Bit3	WRERR:	EEPROM error flag bit；
	1=	Write error (any WDT reset or undervoltage reset during normal operation, or the time set by EETIME is up but the self-check has not been successful);
	0=	Write complete.
Bit2	WREN:	EEPROM write enable bit；
	1=	Enable write period；
	0=	Disable write memory.
Bit1	WR:	Write control bit；
	1=	Start write period (Once the write operation is completed, this bit is cleared by hardware, and the WR bit can only be set to 1, but not cleared by software);；
	0=	Write period complete.
Bit0	RD:	Read control bit；
	1=	Start the memory read operation　(the RD is cleared by hardware, and the RD bit can only be set to 1, but not cleared by software);
	0=	Not start memory read operation.

16.3 Read Program EEPROM

To read the program EEPROM cell, the user must write the address to the EEADR register, clear the EEPGD control bit of the EECON1 register, and then set the control bit RD to 1. Once the read control bit is set, the program EEPROM controller will use the second instruction period to read data. This will cause the second instruction following the "SETB EECON1, RD" instruction to be ignored (1). In the next clock period, the corresponding address value of the program EEPROM will be latched into the EEDAT register In, the user can read these two registers in subsequent instructions. EEDAT will save this value until the next time the user reads or writes data to the unit.

Note: The two instructions after the program memory read operation must be NOP. This prevents the user from executing dual period instructions on the next instruction after the RD position is 1.

example: read program EEPROM

```
EEPDATA_READ:
            LD          A,RADDR         ; Put the address to be read into the EEADR register
            LD          EEADR,A
            CLRB        EECON1,EEPGD    ;access data memory
            SETB        EECON1,RD       ;start reading
            NOP
            NOP
            LD          A,EEDAT         ;read and load data to ACC
            LD          RDATA,A
EEPDATA_READ_BACK:
            RET
```

16.4 Write Program EEPROM

To write a program EEPROM storage unit, the user should first write the unit's address to the EEADR register and write data to the EEDAT register. Then the user must start writing each byte in a specific order.

If you do not follow the following instructions exactly (that is, first write 55h to EECON2, then write Aah to EECON2, and finally set the WR bit to 1) to write each byte, the write operation will not be started. Interrupt should be disabled in this code.

In addition, the WREN bit in EECON1 must be set to 1 to enable write operations. This mechanism can prevent EEPROM from being written by mistake due to code execution errors (abnormal) (ie program runaway). When not updating EEPROM, the user should always keep the WREN bit cleared. The WREN bit cannot be cleared by hardware.

After a write process is started, clearing the WREN bit will not affect the write period. Unless the WREN bit is set, the WR bit will not be set to 1. When the write period is completed, the WR bit is cleared by hardware and the EE write is completed interrupt flag bit (EEIF) is set to 1. user can allow this interrupt or query this bit. EEIF must be cleared by software.

Note: During the writing of the program EEPROM, the CPU will stop working, the CLRWDT command must be executed before the writing operation starts to avoid WDT overflow to reset the chip during this period.

example：write program EEPROM

```
EEPDATA_WRITE:
            LD          A,WADDR             ; Put the address to be written into the EEADR
                                            register
            LD          EEADR,A
            LD          A,WDATA             ; put the data to be written to the EEDAT register
            LD          EEDAT,A
            CLRWDT
            CLR         EECON1
            SETB        EECON1,EETIME0
            SETB        EECON1,EETIME1      ;EE programming time 10ms，user-defined
            CLRB        EECON1,EEPGD        ;access data memory
            SETB        EECON1,WREN         ;enable write period
            CLRB        F_GIE_ON            ;save interrupt enabled status
            SZB         INTCON,GIE
            SETB        F_GIE_ON
            CLRB        INTCON,GIE          ;disable interrupt
            SZB         INTCON,GIE          ;ensure interrupt is disabled
            JP          $-2

            LDIA        055H
            LD          EECON2,A
            LDIA        0AAH
            LD          EECON2,A
            SETB        EECON1,WR           ;start writing
            NOP
            NOP
```

```
                CLRWDT
                CLRB            EECON1,WREN                 ;write complete，turn off write enable bit

                SZB             F_GIE_ON                    ;restore interrupt enabled status
                SETB            INTCON,GIE

                SNZB            EECON1,WRERR                ;check EEPROM write
                JP              EEPDATA_WRITE_BACK
                SZDECR          WERR_C                      ; Exit when the count expires，user-defined
                JP              EEPDATA_WRITE               ;rewrite when EEPROM write error
EEPDATA_WRITE_BACK:
                RET
```

16.5 Read Program Memory

To read the program memory unit, the user must write the high and low bits of the address to the EEADR and EEADRH registers respectively, set the EEPGD bit of EECON1register to 1, and then set the control bit RD to 1. Once the read control bit is set, the program memory controller will use the second instructionsperiod to read data. This will cause the second instructions following the "SETB EECON1,RD" instructions to be ignored. In the next clock period, the value of the corresponding address of the program memory will be latched to EEDAT In the EEDATH register, the user can read these two registers in the subsequent instructions. The EEDAT and EEDATH register will save this value until the next time the user reads or writes data to the unit.

Note:
1) The two instructions after the program memory read operation must be NOP. This prevents the user from executing double period instructions in the next instruction after the RD position is 1.
2) If the WR bit is 1 when EEPGD=1, it will reset to 0 immediately without performing any operation.

example：read flash program memory

```
LD        A,RADDRL          ; Put the address to be read into the EEADR register
LD        EEADR,A
LD        A,RADDRH          ; Put the high bit of the address to be read into
                            EEADRHregister
LD        EEADRH,A
SETB      EECON1,EEPGD      ;select to operate on program memory
SETB      EECON1,RD         ;enable read
NOP
NOP
LD        A,EEDAT           ;save read data
LD        RDATL,A
LD        A,EEDATH
LD        RDATH,A
```

16.6 Write Program Memory

program memoryis read only, cannot be written.

16.7 Precautions on Program EEPROM

16.7.1 Programming Time for Program EEPROM

The program EEPROM programming time is not fixed. The time required to program different data varies from 100us to 10ms. The EETIME bit of the EECON1 register determines the maximum time for program EEPROM programming. Program EEPROM mod built-in self-calibration during the programming process, if the self-verification is successful or the time set by EETIME has expired, the write operation will be terminated when one of the conditions is met. During the programming, the CPU stops working, the peripherals mod works normally, and the program needs to be well dealt with accordingly.

16.7.2 Number of Times for Programming EEPROM

The number of times of the program EEPROM are related to the programming time set by EETIME, as well as voltage and temperature. For details, please refer to the following diagram.

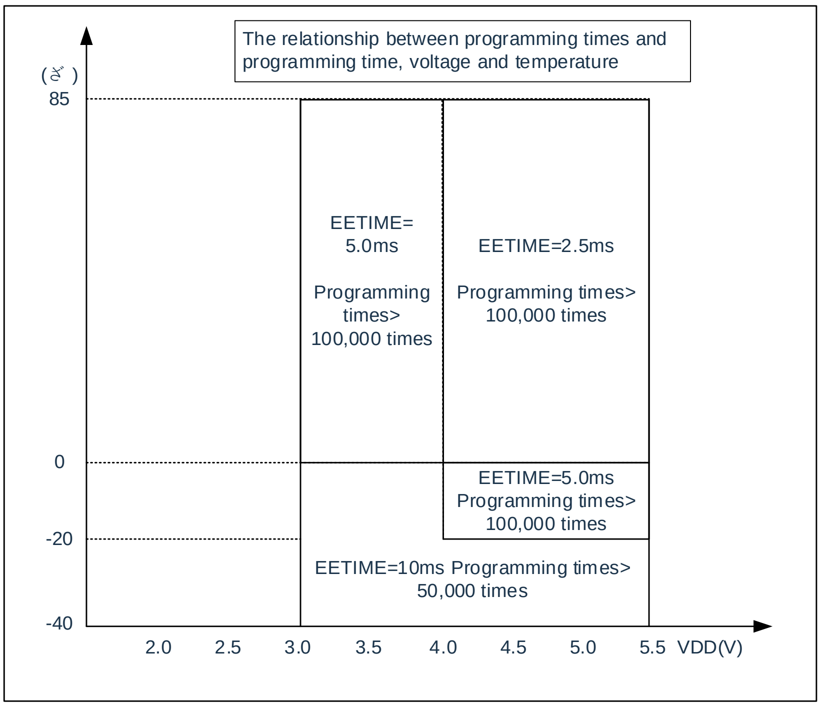


Fig 16-1 The relationship between program EEPROM programming times and programming time, voltage and temperature

16.7.3 Write Verification

According to specific applications, good programming habits generally require verification of the value written into the program EEPROM against the expected value.

16.7.4 Protection to Avoid Writing Wrongly

In some cases, the user may not want to write data to the program EEPROM. In order to prevent accidental writing of EEPROM, various protection mechanisms are embedded in the chip. The WREN bit is cleared when the power is turned on. Moreover, the power-on delay timer (the delay time is 18ms) Will prevent writing to the EEPROM.

The start sequence of the write operation and the WREN bit will work together to prevent false write operations in the following situations:

- Undervoltage
- Power glitch
- Software failure

17. LVD Low Voltage detection

17.1 LVD Mod General

CMS79FT73x series of MCUhave a low-voltage detection function, which can be used to monitor the power supply voltage. If the power supply voltage is lower than the set value, an interrupt signal can be generated; the program can read the LVD output flag bit in real time.

17.2 LVD Related Register

There is 1 register related to LVD mod.

LVD control register LVDCON (185H)

185H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LVDCON	LVD_RES	—	—	—	LVD_SEL[2:0]			LVDEN
R/W	R	—	—	—	R/W	R/W	R/W	R/W
Reset value	X	—	—	—	0	0	0	0

Bit7	LVD_RES:	LVD output result
	0=	VDD> Set LVD voltage;
	1=	VDD< Set LVD voltage;
Bit6~Bit4	not used	
Bit3~Bit1	LVD_SEL[2:0]:	LVD voltage selection
	000=	2.2V;
	001=	2.4V;
	010=	2.7V;
	011=	3.0V;
	100=	3.3V;
	101=	3.7V;
	110=	4.0V;
	111=	4.3V;
Bit0	LVDEN:	LVD enable bit
	0=	disable;
	1=	enable;

17.3 LVD Operation

By setting the LVD voltage value in the LVDCON register, after enabling LVDEN, when the power supply voltage is lower than the set voltage value, the LVD_RES bit in the LVDCON register is set high. After LVD mod is enabled, it takes a delay of 1ms to be able to read the LVD_RES bit, because the internal has done filtering processing to reduce the frequent fluctuation of the LVD output result when the VLVD voltage is near.

LVD mod has its own interrupt flag bit. When the relevant interrupt enable bit is set, and the power supply voltage is lower than the set voltage value, LVD interrupt will be generated, the interrupt flag bit LVDIF will be set to 1, and interrupt generation. LVD is also possible used for interrupt wake up mode.

18. DIV Hardware Divider

18.1 Hardware Divider General

CMS79FT73x series of MCU have a built-in hardware divider, 32-bit dividend, 16-bit divisor, and no remainder output.

Set the dividend through DIVE3, DIVE2, DIVE1 and DIVE0 register. These four registers can only be written but cannot be read. The divisors can be set through DIVS1 and DIVS0 register. These two registers can be read/write. The quotient of the operation is stored in DIVQ3, DIVQ2 In DIVQ1 and DIVQ0 register, these four registers can only be read but cannot be written. DIVEx and DIVQx share a register address, enableD IVEN, and wait for the CAL_END bit to be 1 before reading the quotient.

18.2 Hardware Divider Related Register

There are 11 registers related to the divider mod, namely DIVCON, DIVE3, DIVE2, DIVE1, DIVE0, DIVS1, DIVS0, DIVQ3, DIVQ2, DIVQ1 and DIVQ0.

DIV control register DIVCON (189H)

189H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVCON	DIVEN	CAL_END	—	—	—	—	—	DIV_CLK
R/W	R/W	R	—	—	—	—	—	R/W
Reset value	0	1	—	—	—	—	—	0

Bit7	DIVEN:	DIV enable bit
	0=	Disable;
	1=	enable.
Bit6	CAL_END:	Result flag bit
	0=	Calculation in progress;
	1=	Calculation end already or not started yet.
Bit5~Bit1	not used	
Bit0	DIV_CLK:	DIV clock scaling selection bit
	0:	$F_{SYS}/2$;
	1:	$F_{SYS}/4$;

Divider register DIVE3 (18CH)

18CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVEH1								
R/W	W	W	W	W	W	W	W	W
reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Divider DIVE[32:24]

Divider register DIVE2 (18DH)

18DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVEH0								
R/W	W	W	W	W	W	W	W	W
reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	Divider DIVE[23:16]

Divider register DIVE1 (18EH)

18EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVEM								
R/W	W	W	W	W	W	W	W	W
reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	Divider DIVE[15:8]

Divider register DIVE0 (18FH)

18FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVEL								
R/W	W	W	W	W	W	W	W	W
reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	Divider DIVE[7:0]

Divisor register DIVS1 (187H)

187H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVSH								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	Divisor DIVS[15:8]

Divisor register DIVS0 (188H)

188H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVSL								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	Divisor DIVS[7:0]

Quotient register DIVQ3 (18CH)

18CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVQH1								
R/W	R	R	R	R	R	R	R	R
reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Quotient DIVQ[32:24]

Quotient register DIVQ2 (18DH)

18DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVQH0								
R/W	R	R	R	R	R	R	R	R
reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Quotient DIVQ[23:16]

Quotient register DIVQ1 (18EH)

18EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVQM								
R/W	R	R	R	R	R	R	R	R
reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Quotient DIVQ[15:8]

Quotient register DIVQ0 (18FH)

18FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVQL								
R/W	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Quotient DIVQ[7:0]

19. Touch Button

19.1 Touch Button Mod General

The touch detection mod is an integrated circuit designed to realize a human touch interface. It can replace mechanical touch buttons to achieve a waterproof and dustproof, sealed and isolated, sturdy and beautiful operation interface.

technical parameter:

- 1-16 buttons are optional, all I/O can be configured as touch channels
- No need for external touch capacitance
- High anti-interference performance, can easily pass static 10V, dynamic 3V conduction test

19.2 Precautions for Touch Button Mod

- The ground wire of the detection part of the touch button should be separately connected to an independent ground, and another point is connected to the common ground of the whole machine.
- Avoid high-voltage, high-current, high-frequency operation of the motherboard and the touch circuit board. If it is unavoidable, try to stay away from the area of the high-voltage current or add shielding on the motherboard.
- The connection between the sensor pad and the touch chip should be as short and thin as possible. If the PCB process allows it, try to use a line width of 0.1mm.
- The connection between the sensor panel and the touch chip should not cross the signal line with strong interference and high frequency.
- Do not use other signal lines around 0.5mm from the sensor panel to the touch chip.

20. Electrical Parameter

20.1 Limit Parameter

Supplying voltage	GND-0.3V~GND+6.0V
storage temperature	-50°C~125°C
working temperature	-40°C~85°C
port input voltage	GND-0.3V~VDD+0.3V
Maximum source current for all ports	200mA
Maximum sink current for all ports	-150mA

Note: If the device operating conditions exceed the above "limit parameters", it may cause permanent damage to the device. The above values are only the maximum value of the operating conditions. We do not recommend that the device operate outside the range specified in this specification. The device works for a long time. Under extreme conditions, its stability will be affected.

20.2 DC Feature

(VDD=5V, T_A= 25°C, Unless otherwise indicated)

Symbol	parameter	Test condition		Min. value	Typical value	Max. value	unit
		VDD	condition				
VDD	Working volatge		F_{SYS}=16MHz	2.6		5.5	V
			F_{SYS}=8MHz	2.0		5.5	V
I_{DD}	Working current	5V	F_{SYS}=16MHz		3.8		mA
		3V	F_{SYS}=16MHz		3.0		mA
		5V	F_{SYS}=8MHz		3.0		mA
		3V	F_{SYS}=8MHz		2.3		mA
I_{STB}	Static current	5V	----		0.1	2	μA
		3V	----		0.1	1	μA
V_{IL}	Low level input voltage		----			0.3VDD	V
V_{IH}	High level input voltage		----	0.7VDD			V
V_{OH}	High level output voltage		No load	0.9VDD			V
V_{OL}	Low level output voltage		No load			0.1VDD	V
V_{EEPROM}	EEPROM mod w/r voltage		----	3.0		5.5	V
R_{PH}	pull up resistor resistance	5V	V_O=0.5VDD		35		KΩ
		3V	V_O=0.5VDD		63		KΩ
R_{PD}	pull down resistor resistance	5V	V_O=0.5VDD		35		KΩ
		3V	V_O=0.5VDD		63		KΩ
I_{OL1}	Output port source current (normal I/O port)	5V	V_{OL}=0.3VDD		60		mA
		3V	V_{OL}=0.3VDD		25		mA
I_{OH1}	Output port drain current (normal I/O port)	5V	V_{OH}=0.7VDD		-20		mA
		3V	V_{OH}=0.7VDD		-9		mA
I_{OL2}	Output port source current (LED COM port)	5V	V_{OL}=0.3VDD		150		mA
		3V	V_{OL}=0.3VDD		70		mA
I_{OH2}	Output port drain current (LED SEG port max. current)	5V	V_{OH}=0.7VDD		-30		mA
		3V	V_{OH}=0.7VDD		-12		mA
V_{BG}	Internal reference voltage 1.2V	VDD=2.5~5.5V T_A=25°C		-1.5%	1.2	1.5%	V
		VDD=2.5~5.5V T_A= - 40~85°C		-2.0%	1.2	2.0%	V

20.3 ADC Feature

(T_A= 25℃, Unless otherwise indicated)

Symbol	parameter	Test condition	min. value	typical value	max. value	unit
V_{ADC}	ADC working voltage	F_{ADC}=500kHz	2.7		5.5	V
I_{ADC}	ADC current	V_{ADC}=5V,F_{ADC}=500kHz			500	μA
		V_{ADC} =3V,F_{ADC}=500kHz			200	μA
V_{ADI}	ADC inputvoltage	V_{ADC}=5V,F_{ADC}=250kHz	0		V_{ADC}	V
DNL	Differential nonlinearity error	V_{ADC}=5V,F_{ADC}=250kHz		±3		LSB
INL	Integral nonlinearity error	V_{ADC}=5V,F_{ADC}=250kHz		±4		LSB
T_{ADC}	ADC conversion time	-		49		T_{ADCCLK}

20.4 Power on Reset Feature

(T_A= 25℃, Unless otherwise indicated)

symbol	parameter	Test condition	min. value	typical value	max. value	unit
t_{VDD}	VDD increase rate	-	0.05			V/ms
V_{LVR2}	LVR =2.0V	VDD=1.8~5.5V	1.9	2.0	2.1	V
V_{LVR3}	LVR=2.6V	VDD=2.4~5.5V	2.5	2.6	2.7	V

20.5 AC Electrical Characteristics

(T_A=25℃, Unless otherwise indicated)

symbol	parameter	Test condition		min.	typical	max.	unit
		VDD	condition	value	value	value	
T_{WDT}	WDT reset time	5V	-		16		ms
		3V	-		32		ms
T_{EEPROM}	EEPROM programming time	5V	F_{HSI} =8MHz/16MHz			10	ms
		3V	F_{HSI}=8MHz/16MHz			10	ms
F_{RC}	Internal frequency stability	VDD=4.5~5.5V	T_A=25℃	-1.5%	8	+1.5%	MHz
		VDD=2.0~5.5V	T_A=25℃	-2%	8	+2%	MHz
		VDD=4.5~5.5V	T_A= - 40~85℃	-2.5%	8	+2.5%	MHz
		VDD=2.0~5.5V	T_A= - 40~85℃	-3.5%	8	+3.5%	MHz
		VDD=4.5~5.5V	T_A=25℃	-1.5%	16	+1.5%	MHz
		VDD=2.6~5.5V	T_A=25℃	-2%	16	+2%	MHz
		VDD=4.5~5.5V	T_A= - 40~85℃	-2.5%	16	+2.5%	MHz
		VDD=2.6~5.5V	T_A= - 40~85℃	-3.5%	16	+3.5%	MHz

20.6 IIC Electrical Characteristics

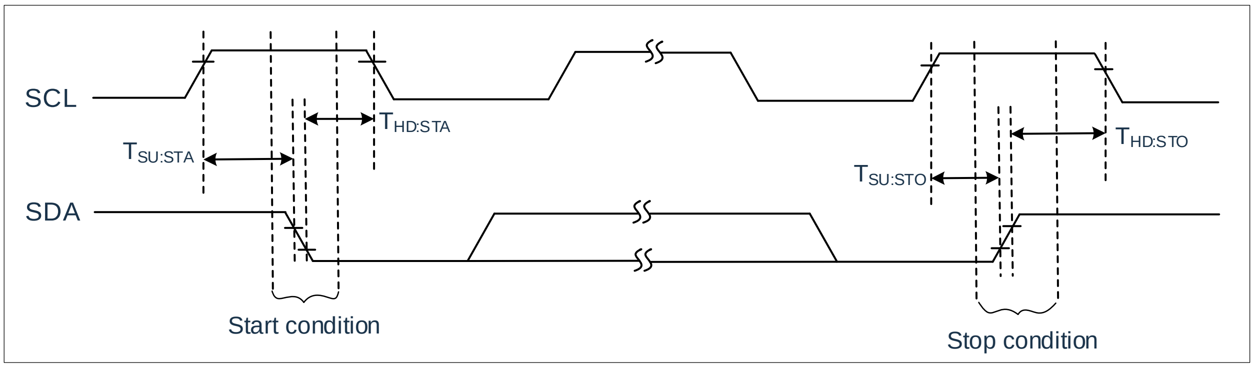


Figure 20-1: I^2C^{TM} Bus Start/Stop Bit Timing

Symbol	Characteristic		Condition	Min	Typ	Max	Unit
$T_{SU:STA}$	Start condition set-up time	100kHz Mode	Only relevant for repeated start conditions	4700	-	-	ns
		400kHz Mode		600	-	-	ns
$T_{HD:STA}$	Start condition hold time	100kHz Mode	The first clock pulse is generated after this cycle	4000	-	-	ns
		400kHz Mode		600	-	-	ns
$T_{SU:STO}$	Stop condition set-up time	100kHz Mode		4700	-	-	ns
		400kHz Mode		600	-	-	ns
$T_{HD:STO}$	Stop condition hold time	100kHz Mode		4000	-	-	ns
		400kHz Mode		600	-	-	ns

Note: These parameters are characteristic values only and have not been tested.

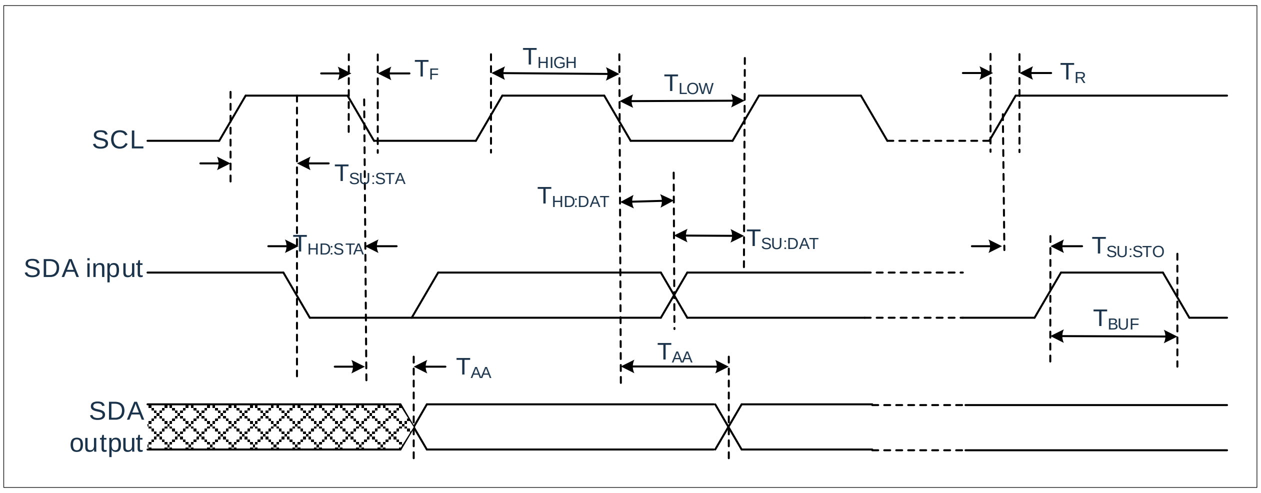


Figure 20-2: I^2C^{TM} Bus Data Timing

Symbol	Characteristic		Condition	Min	Max	Unit
T_{HIGH}	Clock high level time	100kHz Mode	The operating frequency of the device must not be lower than 4MHz	4.0	-	us
		400kHz Mode	The operating frequency of the device must not be lower than 16MHz	0.6	-	us
T_{LOW}	Clock low level time	100kHz Mode	The operating frequency of the device must not be lower than 4MHz	4.7	-	us
		400kHz Mode	The operating frequency of the device must not be lower than 16MHz	1.3	-	us
T_R	SDA and SCL rising time	100kHz Mode		-	1000	ns
		400kHz Mode	C_B values are specified in the range of 10-400pF	$20+0.1C_B$	300	ns
T_F	SDA and SCL falling time	100kHz Mode		-	300	ns
		400kHz Mode	C_B values are specified in the range of 10-400pF	$20+0.1C_B$	300	ns
$T_{SU:STA}$	Start condition set-up time	100kHz Mode	Only relevant for repeated start conditions	4.7	-	us
		400kHz Mode		0.6	-	us
$T_{HD:STA}$	Start condition hold time	100kHz Mode	The first clock pulse is generated after this cycle	4.0	-	us
		400kHz Mode		0.6	-	us
$T_{HD:DAT}$	Data input hold time	100kHz Mode		$2/F_{sys}$	-	us
		400kHz Mode		$2/F_{sys}$	$1-2/F_{sys}$	us
$T_{SU:DAT}$	Data input set-up time	100kHz Mode		$2/F_{sys}$	-	us
		400kHz Mode		$2/F_{sys}$	-	us
$T_{SU:STO}$	Stop condition set-up	100kHz Mode		4.7	-	us

	time	400kHz Mode		0.6	-	us
T_{AA}	Clock output valid time	100kHz Mode		-	3.7-2/F_{sys}	us
		400kHz Mode		-	-	us
T_{BUF}	Bus idle time	100kHz Mode	The duration that the bus shall remain idle for before a new transmission begins.	4.7	-	us
		400kHz Mode		1.3	-	us
C_B	Bus capacitive load			-	400	pF

Note: These parameters are characteristic values only and have not been tested.

21. Instructions

21.1 Instructions Table

mnemonic		operation	instructio nsperiod	symbol
control-3				
NOP		Empty operation	1	None
STOP		Enter sleep mode	1	TO,PD
CLRWDT		Clear watchdog timer	1	TO,PD
Data transfer-4				
LD	[R],A	Transfer content to ACC to R	1	NONE
LD	A,[R]	Transfer content to R to ACC	1	Z
TESTZ	[R]	Transfer the content of data memory data memory	1	Z
LDIA	i	Transfer I to ACC	1	NONE
logic operation -16				
CLRA		Clear ACC	1	Z
SET	[R]	Set data memory R	1	NONE
CLR	[R]	Clear data memory R	1	Z
ORA	[R]	Perform 'OR' on R and ACC, save the result to ACC	1	Z
ORR	[R]	Perform 'OR' on R and ACC, save the result to R	1	Z
ANDA	[R]	Perform 'AND' on R and ACC, save the result to ACC	1	Z
ANDR	[R]	Perform 'AND' on R and ACC, save the result to R	1	Z
XORA	[R]	Perform 'XOR' on R and ACC, save the result to ACC	1	Z
XORR	[R]	Perform 'XOR' on R and ACC, save the result to R	1	Z
SWAPA	[R]	Swap R register high and low half byte, save the result to ACC	1	NONE
SWAPR	[R]	Swap R register high and low half byte, save the result to R	1	NONE
COMA	[R]	The content of R register is reversed, and the result is stored in ACC	1	Z
COMR	[R]	The content of R register is reversed and the result is stored in R	1	Z
XORIA	i	Perform 'XOR' on i and ACC, save the result to ACC	1	Z
ANDIA	i	Perform 'AND' on i and ACC, save the result to ACC	1	Z
ORIA	i	Perform 'OR' on i and ACC, save the result to ACC	1	Z
Shift operation-8				
RRCA	[R]	Data memory rotates one bit to the right with carry, the result is stored in ACC	1	C
RRCR	[R]	Data memory rotates one bit to the right with carry, the result is stored in R	1	C
RLCA	[R]	Data memory rotates one bit to the left with carry, the result is stored in ACC	1	C
RLCR	[R]	Data memory rotates one bit to the left with carry, the result is stored in R	1	C
RLA	[R]	Data memory rotates one bit to the left without carry, and the result is stored in ACC	1	NONE
RLR	[R]	Data memory rotates one bit to the left without carry, and the result is stored in R	1	NONE
RRA	[R]	Data memory does not take carry and rotates to the right by one bit, and the result is stored in ACC	1	NONE
RRR	[R]	Data memory does not take carry and rotates to the right by one bit, and the result is stored in R	1	NONE
Increae/decrease-4				
INCA	[R]	Increment data memory R，result stored in ACC	1	Z
INCR	[R]	Increment data memory R，result stored in R	1	Z

mnemonic		operation	instructio nsperiod	symbol
DECA	[R]	Decrement data memory R， result stored in ACC	1	Z
DECR	[R]	Decrement data memory R， result stored in R	1	Z
Bit operation-2				
CLRB	[R],b	Clear some bit in data memory R	1	NONE
SETB	[R],b	Set some bit in data memory R 1	1	NONE
look-up table-2				
TABLE	[R]	Read FLASH and save to TABLE_DATAH and R	2	NONE
TABLEA		Read FLASH and save to TABLE_DATAH and ACC	2	NONE
Math operation-16				
ADDA	[R]	ACC+[R]→ACC	1	C,DC,Z,OV
ADDR	[R]	ACC+[R]→R	1	C,DC,Z,OV
ADDCA	[R]	ACC+[R]+C→ACC	1	Z,C,DC,OV
ADDCR	[R]	ACC+[R]+C→R	1	Z,C,DC,OV
ADDIA	i	ACC+i→ACC	1	Z,C,DC,OV
SUBA	[R]	[R]-ACC→ACC	1	C,DC,Z,OV
SUBR	[R]	[R]-ACC→R	1	C,DC,Z,OV
SUBCA	[R]	[R]-ACC-C→ACC	1	Z,C,DC,OV
SUBCR	[R]	[R]-ACC-C→R	1	Z,C,DC,OV
SUBIA	i	i-ACC→ACC	1	Z,C,DC,OV
HSUBA	[R]	ACC-[R]→ACC	1	Z,C,DC,OV
HSUBR	[R]	ACC-[R]→R	1	Z,C,DC,OV
HSUBCA	[R]	ACC-[R]- $\overline{C}$ →ACC	1	Z,C,DC,OV
HSUBCR	[R]	ACC-[R]- $\overline{C}$ →R	1	Z,C,DC,OV
HSUBIA	i	ACC-i→ACC	1	Z,C,DC,OV
Unconditional transfer -5				
RET		Return from subroutine	2	NONE
RET	i	Return from subroutine， save I to ACC	2	NONE
RETI		Return from interrupt	2	NONE
CALL	ADD	Subroutine call	2	NONE
JP	ADD	Unconditional jump	2	NONE
Conditional transfer-8				
SZB	[R],b	If the b bit of data memory R is "0", skip the next instruction	1 or 2	NONE
SNZB	[R],b	If the b bit of data memory R is "1", skip the next instruction	1 or 2	NONE
SZA	[R]	data memory R is sent to ACC, if the content is "0", skip the next instruction	1 or 2	NONE
SZR	[R]	If the content of data memory R is "0", skip the next instruction	1 or 2	NONE
SZINCA	[R]	Add "1" to data memory R and put the result into ACC, if the result is "0", skip the next oneinstructions	1 or 2	NONE
SZINCR	[R]	Add "1" to data memory R, put the result into R, if the result is "0", skip the next instruction	1 or 2	NONE
SZDECA	[R]	Data memory R minus "1", the result is put into ACC, if the result is "0", skip the next instruction	1 or 2	NONE
SZDECR	[R]	Data memory R minus "1", put the result into R, if the result is "0", skip the next oneinstructions	1 or 2	NONE

21.2 Instructions Illustration

ADDA	**[R]**
operation:	Add ACC to R, save the result to ACC
period:	1
Affected flag bit:	C，DC，Z，OV
example:	

```
LDIA        09H              ;load 09H to ACC
LD          R01,A            ;load ACC (09H) to R01
LDIA        077H             ;load 77H to ACC
ADDA        R01              ;execute：ACC=09H + 77H =80H
```

ADDR	**[R]**
operation:	Add ACC to R , save the result to R
period:	1
Affected flag bit:	C，DC，Z，OV
example:	

```
LDIA        09H              ;load 09H to ACC
LD          R01,A            ; load ACC (09H) to R01
LDIA        077H             ; load 77H to ACC
ADDR        R01              ;execute：R01=09H + 77H =80H
```

ADDCA	**[R]**
operation:	Add ACC to C, save the result to ACC
period:	1
affected flag bit:	C，DC，Z，OV
example:	

```
LDIA        09H              ; load 09H to ACC
LD          R01,A            ; load ACC (09H) to R01
LDIA        077H             ; load 77H to ACC
ADDCA       R01              ;execute：ACC= 09H + 77H + C=80H   (C=0)
                                       ACC= 09H + 77H + C=81H   (C=1)
```

ADDCR	**[R]**
operation:	Add ACC to C, save the result to R
period:	1
affected flag bit:	C，DC，Z，OV
example:	

```
LDIA        09H              ; load 09H to ACC
LD          R01,A            ; load ACC (09H) to R01
LDIA        077H             ; load 77H to ACC
ADDCR       R01              ;execute：R01 = 09H + 77H + C=80H   (C=0)
                                        R01 = 09H + 77H + C=81H   (C=1)
```

ADDIA	**i**		
operation:	Add i to ACC, save the result to ACC		
period:	1		
affected flag bit:	C，DC，Z，OV		
example:			
	LDIA	09H	; load 09H to ACC
	ADDIA	077H	;execute：ACC = ACC (09H) + i (77H)=80H

ANDA	**[R]**		
operation:	Perform‘AND’on register R and ACC, save the result to ACC		
period:	1		
affected flag bit:	Z		
example:			
	LDIA	0FH	;load 0FH to ACC
	LD	R01,A	;load ACC (0FH) to R01
	LDIA	77H	;load 77H to ACC
	ANDA	R01	;execute：ACC= (0FH and 77H)=07H

ANDR	**[R]**		
operation:	Perform‘AND’on register R and ACC, save the result to R		
period:	1		
affected flag bit:	Z		
example:			
	LDIA	0FH	; load 0FH to ACC
	LD	R01,A	; load ACC (0FH) to R01
	LDIA	77H	; load 77H to ACC
	ANDR	R01	;execute：R01= (0FH and 77H)=07H

ANDIA	**i**		
operation:	Perform‘AND’on i and ACC, save the result to ACC		
period:	1		
affected flag bit:	Z		
example:			
	LDIA	0FH	; load 0FH to ACC
	ANDIA	77H	;execute：ACC = (0FH and 77H)=07H

CALL	**add**		
operation:	Call subroutine		
period:	2		
affected flag bit:	none		
example:			
	CALL	LOOP	; Call the subroutine address whose name is defined as "LOOP"

CLRA

operation:	ACC clear	
period:	1	
affected flag bit:	Z	
example:		

```
CLRA                          ;execute：ACC=0
```

CLR **[R]**

operation:	Register R clear
period:	1
affected flag bit:	Z
example:	

```
CLR          R01              ;execute：R01=0
```

CLRB **[R],b**

operation:	Clear b bit on register R
period:	1
affected flag bit:	none
example:	

```
CLRB         R01,3            ;execute：3rd bit of R01 is 0
```

CLRWDT

operation:	Clear watchdog timer
period:	1
affected flag bit:	TO，PD
example:	

```
CLRWDT                        ;watchdog timer clear
```

COMA **[R]**

operation:	Reverse register R，save the result to ACC
period:	1
affected flag bit:	Z
example:	

```
LDIA         0AH              ;load 0AH to ACC
LD           R01,A            ;load ACC (0AH) to R01
COMA         R01              ;execute：ACC=0F5H
```

COMR	**[R]**
operation:	Reverse register R， save the result to R
period:	1
affected flag bit:	Z
example:	

```
LDIA      0AH        ; load 0AH to ACC
LD        R01,A      ; load ACC (0AH) to R01
COMR      R01        ;execute： R01=0F5H
```

DECA	**[R]**
operation:	Decrement value in register , save the result to ACC
period:	1
affected flag bit:	Z
example:	

```
LDIA      0AH        ;load 0AH to ACC
LD        R01,A      ; load ACC (0AH) to R01
DECA      R01        ;execute： ACC= (0AH-1)=09H
```

DECR	**[R]**
operation:	Decrement value in register , save the result to R
period:	1
affected flag bit:	Z
example:	

```
LDIA      0AH        ; load 0AH to ACC
LD        R01,A      ; load ACC (0AH) to R01
DECR      R01        ;execute： R01= (0AH-1)=09H
```

HSUBA	**[R]**
operation:	ACC subtract R, save the result to ACC
period:	1
affected flag bit:	C,DC,Z,OV
example:	

```
LDIA      077H       ; load 077H to ACC
LD        R01,A      ; load ACC (077H) to R01
LDIA      080H       ; load 080H to ACC
HSUBA     R01        ;execute： ACC= (80H-77H)=09H
```

HSUBR	**[R]**
operation:	ACC subtract R, save the result to R
period:	1
affected flag bit:	C,DC,Z,OV
example:	

```
LDIA        077H        ; load 077H to ACC
LD          R01,A       ; load ACC (077H) to R01
LDIA        080H        ; load 080H to ACC
HSUBR       R01         ;execute: R01= (80H-77H)=09H
```

HSUBCA	**[R]**
operation:	ACC subtract C, save the result to ACC
period:	1
affected flag bit:	C,DC,Z,OV
example:	

```
LDIA        077H        ; load 077H to ACC
LD          R01,A       ; load ACC (077H) to R01
LDIA        080H        ; load 080H to ACC
HSUBCA      R01         ;execute: ACC= (80H-77H-C)=09H (C=0)
                                  ACC= (80H-77H-C)=08H (C=1)
```

HSUBCR	**[R]**
operation:	ACC subtract C, save the result to R
period:	1
affected flag bit:	C,DC,Z,OV
example:	

```
LDIA        077H        ; load 077H to ACC
LD          R01,A       ; load ACC (077H) to R01
LDIA        080H        ; load 080H to ACC
HSUBCR      R01         ;execute: R01= (80H-77H-C)=09H (C=0)
                                  R01= (80H-77H-C)=08H (C=1)
```

INCA	**[R]**
operation:	Register R increment 1, save the result to ACC
period:	1
affected flag bit:	Z
example:	

```
LDIA        0AH         ; load 0AH to ACC
LD          R01,A       ; load ACC (0AH) to R01
INCA        R01         ;execute: ACC= (0AH+1)=0BH
```

INCR	**[R]**
operation:	Register R increment 1, save the result to R
period:	1
affected flag bit:	Z
example:	

```
LDIA        0AH             ; load 0AH to ACC
LD          R01,A           ; load ACC (0AH) to R01
INCR        R01             ;execute： R01= (0AH+1)=0BH
```

JP	**add**
operation:	Jump to add address
period:	2
affected flag bit:	none
example:	

```
JP          LOOP            ; jump to the subroutine address whose name is defined as "LOOP"
```

LD	**A,[R]**
operation:	Load the value of R to ACC
period:	1
affected flag bit:	Z
example:	

```
LD          A,R01           ;load R01 to ACC
LD          R02,A           ;load ACC to R02, achieve data transfer from R01→R02
```

LD	**[R],A**
operation:	Load the value of ACC to R
period:	1
affected flag bit:	none
example:	

```
LDIA        09H             ;load 09H to ACC
LD          R01,A           ;execute： R01=09H
```

LDIA	**i**
operation:	Load in to ACC
period:	1
affected flag bit:	none
example:	

```
LDIA        0AH             ;load 0AH to ACC
```

NOP	
operation:	Empty instructions
period:	1
affected flag bit:	none
example:	

```
NOP
NOP
```

ORIA	**i**
operation:	Perform 'OR' on I and ACC, save the result to ACC
period:	1
affected flag bit:	Z
example:	

```
LDIA    0AH      ; load 0AH to ACC
ORIA    030H     ;execute：ACC = (0AH   or   30H)=3AH
```

ORA	**[R]**
operation:	Perform 'OR' on R and ACC, save the result to ACC
period:	1
affected flag bit:	Z
example:	

```
LDIA    0AH      ; load 0AH to ACC
LD      R01,A    ;load ACC (oAH) to R01
LDIA    30H      ;load 30H to ACC
ORA     R01      ;execute：ACC= (0AH   or   30H)=3AH
```

ORR	**[R]**
operation:	Perform 'OR' on R and ACC, save the result to R
period:	1
affected flag bit:	Z
example:	

```
LDIA    0AH      ; load 0AH to ACC
LD      R01,A    ; load ACC (oAH) to R01
LDIA    30H      ; load 30H to ACC
ORR     R01      ;execute：R01= (0AH   or   30H)=3AH
```

RET

operation:	Return from subroutine
period:	2
affected flag bit:	none
example:	

```
        CALL    LOOP            ; Call subroutine LOOP
        NOP                     ; This statement will be executed after RET instructions return
        …                       ; others
LOOP:
        …                       ;subroutine
        RET                     ;return
```

RET **i**

operation:	Return with parameter from the subroutine, and put the parameter in ACC
period:	2
affected flag bit:	none
example:	

```
        CALL    LOOP            ; Call subroutine LOOP
        NOP                     ; This statement will be executed after RET instructions return
        …                       ;others
LOOP:
        …                       ;subroutine
        RET     35H             ;return,ACC=35H
```

RETI

operation:	Interrupt return
period:	2
affected flag bit:	none
example:	

```
INT_START                       ;interrupt entrance
        …                       ;interrupt procedure
        RETI                    ;interrupt return
```

RLCA **[R]**

operation:	Register R rotates to the left with C and savethe result into ACC
period:	1
affected flag bit:	C
example:	

```
        LDIA    03H             ;load 03H to ACC
        LD      R01,A           ;load ACC to R01,R01=03H
        RLCA    R01             ;operation result：ACC=06H (C=0);
                                             ACC=07H (C=1)
                                             C=0
```

RLCR	**[R]**
operation:	Register R rotates one bit to the left with C, and save the result into R
period:	1
affected flag bit:	C
example:	

```
LDIA        03H             ; load 03H to ACC
LD          R01,A           ; load ACC to R01,R01=03H
RLCR        R01             ;operation result： R01=06H (C=0);
                                              R01=07H (C=1);
                                              C=0
```

RLA	**[R]**
operation:	Register R without C rotates to the left, and save the result into ACC
period:	1
affected flag bit:	none
example:	

```
LDIA        03H             ; load 03H to ACC
LD          R01,A           ; load ACC to R01,R01=03H
RLA         R01             ;operation result： ACC=06H
```

RLR	**[R]**
operation:	Register R without C rotates to the left, and save the result to R
period:	1
affected flag bit:	none
example:	

```
LDIA        03H             ; load 03H to ACC
LD          R01,A           ; load ACC to R01,R01=03H
RLR         R01             ;operation result： R01=06H
```

RRCA	**[R]**
operation:	Register R rotates one bit to the right with C, and puts the result into ACC
period:	1
affected flag bit:	C
example:	

```
LDIA        03H             ; load 03H to ACC
LD          R01,A           ; load ACC to R01,R01=03H
RRCA        R01             ;operation result： ACC=01H (C=0);
                                              ACC=081H (C=1);
                                              C=1
```

RRCR	**[R]**
operation:	Register R rotates one bit to the right with C, and save the result into R
period:	1
affected flag bit:	C
example:	

```
LDIA        03H               ; load 03H to ACC
LD          R01,A             ; load ACC to R01,R01=03H
RRCR        R01               ;operation result: R01=01H (C=0);
                                         R01=81H (C=1);
                                         C=1
```

RRA	**[R]**
operation:	Register R without C rotates one bit to the right, and save the result into ACC
period:	1
affected flag bit:	none
example:	

```
LDIA        03H               ; load 03H to ACC
LD          R01,A             ; load ACC to R01,R01=03H
RRA         R01               ;operation result: ACC=81H
```

RRR	**[R]**
operation:	Register R without C rotates one bit to the right, and save the result into R
period:	1
affected flag bit:	none
example:	

```
LDIA        03H               ; load 03H to ACC
LD          R01,A             ; load ACC to R01,R01=03H
RRR         R01               ;operation result: R01=81H
```

SET	**[R]**
operation:	Set all bits in register R as 1
period:	1
affected flag bit:	none
example:	

```
SET         R01               ;operation result: R01=0FFH
```

SETB	**[R],b**
operation:	Set b bit in register R 1
period:	1
affected flag bit:	none
example:	

```
CLR         R01               ;R01=0
SETB        R01,3             ;operation result: R01=08H
```

STOP

operation:	Enter sleep
period:	1
affected flag bit:	TO，PD
example:	

```
STOP                            ; The chip enters the power saving mode, the CPU and oscillator
                                stop working, and the IO port keeps the original state
```

SUBIA **i**

operation:	ACC minus I, save the result to ACC
period:	1
affected flag bit:	C,DC,Z,OV
example:	

```
LDIA      077H          ;load 77H to ACC
SUBIA     80H           ;operation result：ACC=80H-77H=09H
```

SUBA **[R]**

operation:	Register R minus ACC, save the result to ACC
period:	1
affected flag bit:	C,DC,Z,OV
example:	

```
LDIA      080H          ;load 80H to ACC
LD        R01,A         ;load ACC to R01，R01=80H
LDIA      77H           ;load 77H to ACC
SUBA      R01           ;operation result：ACC=80H-77H=09H
```

SUBR **[R]**

operation:	Register R minus ACC, save the result to R
period:	1
affected flag bit:	C,DC,Z,OV
example:	

```
LDIA      080H          ; load 80H to ACC
LD        R01,A         ; load ACC to R01，R01=80H
LDIA      77H           ; load 77H to ACC
SUBR      R01           ;operation result：R01=80H-77H=09H
```

SUBCA **[R]**

operation:	Register R minus ACC minus C，save the result to ACC
period:	1
affected flag bit:	C,DC,Z,OV
example:	

```
LDIA     080H     ; load 80H to ACC
LD       R01,A    ; load ACC to R01，R01=80H
LDIA     77H      ; load 77H to ACC
SUBCA    R01      ;operation result：ACC=80H-77H-C=09H (C=0);
                              ACC=80H-77H-C=08H (C=1);
```

SUBCR **[R]**

operation:	Register R minus ACC minus C，save the result to ACC
period:	1
affected flag bit:	C,DC,Z,OV
example:	

```
LDIA     080H     ; load 80H to ACC
LD       R01,A    ; load ACC to R01，R01=80H
LDIA     77H      ; load 77H to ACC
SUBCR    R01      ;operation result：R01=80H-77H-C=09H (C=0)
                              R01=80H-77H-C=08H (C=1)
```

SWAPA **[R]**

operation:	Register R high and low half byte swap, the save result into ACC
period:	1
affected flag bit:	none
example:	

```
LDIA     035H     ;load 35H to ACC
LD       R01,A    ; load ACC to R01，R01=35H
SWAPA    R01      ;operation result：ACC=53H
```

SWAPR **[R]**

operation:	Register R high and low half byte swap, the save result into R
period:	1
affected flag bit:	none
example:	

```
LDIA     035H     ; load 35H to ACC
LD       R01,A    ; load ACC to R01，R01=35H
SWAPR    R01      ;operation result：R01=53H
```

SZB	**[R],b**
operation:	Determine the bit b of register R, if it is 0 then jump, otherwise execute in sequence
period:	1 or 2
affected flag bit:	none
example:	

```
SZB       R01,3          ;determine 3rd bit of R01
JP        LOOP           ;if is 1, execute, jump to LOOP
JP        LOOP1          ; if is 0, jump,execute, jump to LOOP1
```

SNZB	**[R],b**
operation:	Determine the bit b of register R, if it is 1 then jump, otherwise execute in sequence
period:	1 or 2
affected flag bit:	none
example:	

```
SNZB      R01,3          ; determine 3rd bit of R01
JP        LOOP           ; if is 0, execute, jump to LOOP
JP        LOOP1          ; if is 1, jump,execute, jump to LOOP1
```

SZA	**[R]**
operation:	Load the value of R to ACC, if it is 0 then jump, otherwise execute in sequence
period:	1 or 2
affected flag bit:	none
example:	

```
SZA       R01            ;R01→ACC
JP        LOOP           ;if R01 is not 0, execute， jump to LOOP
JP        LOOP1          ;if R01 is 0, jump， execute， jump to LOOP1
```

SZR	**[R]**
operation:	Load the value of R to R, if it is 0 then jump, otherwise execute in sequence
period:	1 or 2
affected flag bit:	None
example:	

```
SZR       R01            ;R01→R01
JP        LOOP           ; if R01 is not 0, execute， jump to LOOP
JP        LOOP1          ; if R01 is 0, jump， execute， jump to LOOP1
```

SZINCA	**[R]**
operation:	Increment register by 1， save the result to ACC， if it is 0 then jump, otherwise execute in sequence
period:	1 or 2
affected flag bit:	none
example:	

```
SZINCA    R01        ;R01+1→ACC
JP        LOOP       ; if ACC is not 0, execute， jump to LOOP
JP        LOOP1      ; if ACC is 0, jump， execute， jump to LOOP1
```

SZINCR	**[R]**
operation:	Increment register by 1， save the result to R， if it is 0 then jump, otherwise execute in sequence
period:	1 or 2
affected flag bit:	none
example:	

```
SZINCR    R01        ;R01+1→R01
JP        LOOP       ; if R01 is not 0, execute， jump to LOOP
JP        LOOP1      ; if R01 is 0, jump， execute， jump to LOOP1
```

SZDECA	**[R]**
operation:	decrement register by 1， save the result to ACC， if it is 0 then jump, otherwise execute in sequence
period:	1 or 2
affected flag bit:	none
example:	

```
SZDECA    R01        ;R01-1→ACC
JP        LOOP       ; if ACC is not 0, execute， jump to LOOP
JP        LOOP1      ; if ACC is 0, jump， execute， jump to LOOP1
```

SZDECR	**[R]**
operation:	Decrement register by 1， save the result to R， if it is 0 then jump, otherwise execute in sequence
period:	1 or 2
affected flag bit:	none
example:	

```
SZDECR    R01        ;R01-1→R01
JP        LOOP       ; if R01 is not 0, execute， jump to LOOP
JP        LOOP1      ; if R01 is 0, jump， execute， jump to LOOP1
```

TABLE **[R]**

operation:	Look-up table, the lower 8 bits of the look-up table result are placed in R, and the high bits are placed in the dedicated register TABLE_DATAH
period:	2
affected flag bit:	none
example:	

```
LDIA      01H            ;load 01H to ACC
LD        TABLE_SPH,A    ;load ACC to higher bits of table address， TABLE_SPH=1
LDIA      015H           ;load 15H to ACC
LD        TABLE_SPL,A    ; load ACC to lower bits of table address， TABLE_SPL=15H
TABLE     R01            ;look-up table 0115H address , operation result :
                         TABLE_DATAH=12H， R01=34H
…
ORG       0115H
DW        1234H
```

TABLEA

operation:	Look-up table, the lower 8 bits of the look-up table result are placed in ACC, and the high bits are placed in the dedicated register TABLE_DATAH
period:	2
affected flag bit:	none
example:	

```
LDIA      01H            ; load 01H to ACC
LD        TABLE_SPH,A    ; load ACC to higher bits of table address， TABLE_SPH=1
LDIA      015H           ; load 15H to ACC
LD        TABLE_SPL,A    ; load ACC to lower bits of table address， TABLE_SPL=15H
TABLEA                   ;look-up table 0115H address , operation result :
                         TABLE_DATAH=12H， ACC=34H
…
ORG       0115H
DW        1234H
```

TESTZ **[R]**

operation:	Pass the R to R, as affected Z flag bit
period:	1
affected flag bit:	Z
example:	

```
TESTZ     R0             ;
SZB       STATUS,Z       ;check Z flag bit， if it is 0 then jump
JP        Add1           ;if R0 is 0, jump to address Add1
JP        Add2           ;if R0 is not 0, jump to address Add2
```

XORIA	**i**
operation:	Perform ‘XOR’ on I and ACC， save the result to ACC
period:	1
affected flag bit:	Z
example:	

```
LDIA     0AH       ;load 0AH to ACC
XORIA    0FH       ;execute：ACC=05H
```

XORA	**[R]**
operation:	Perform ‘XOR’ on I and ACC， save the result to ACC
period:	1
affected flag bit:	Z
example:	

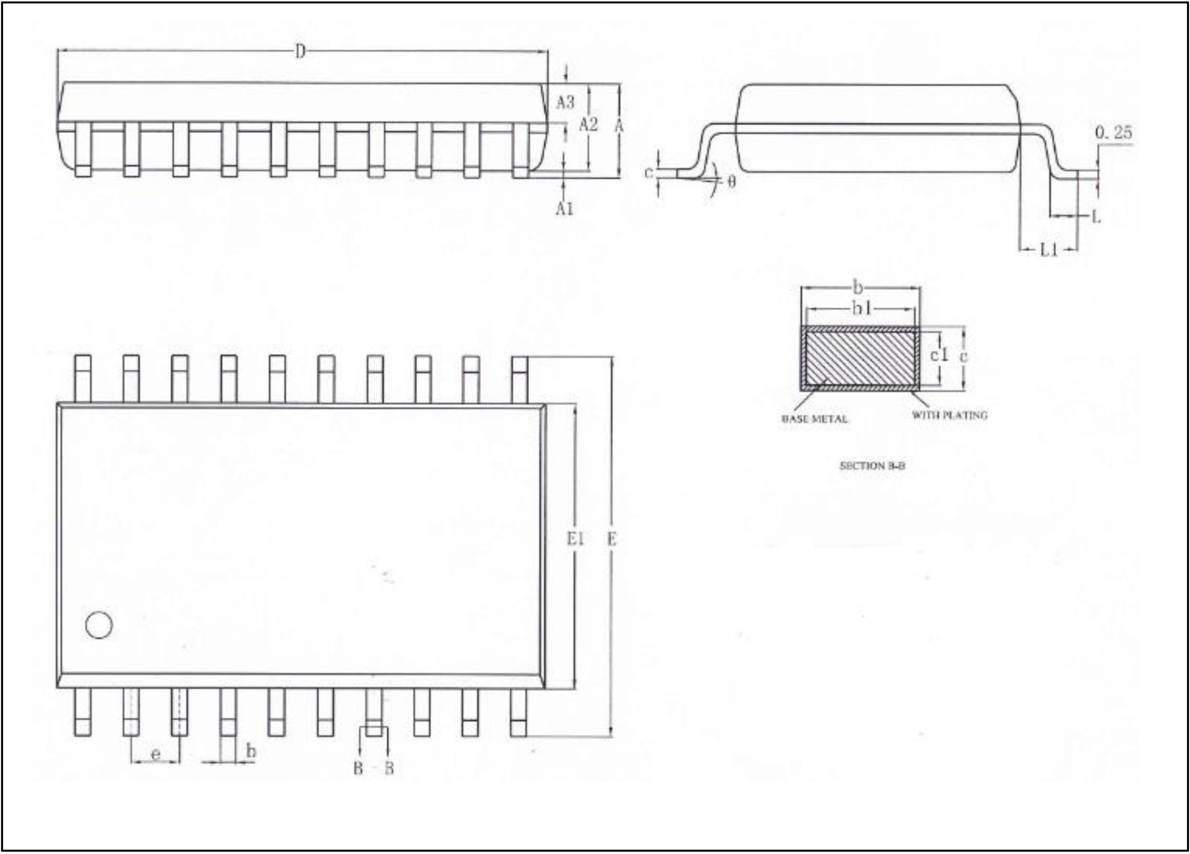
```
LDIA     0AH       ; load 0AH to ACC
LD       R01,A     ;load ACC to R01,R01=0AH
LDIA     0FH       ;load 0FH to ACC
XORA     R01       ;execute：ACC=05H
```

XORR	**[R]**
operation:	Perform ‘XOR’ on I and ACC， save the result to R
period:	1
affected flag bit:	Z
example:	

```
LDIA     0AH       ; load 0AH to ACC
LD       R01,A     ; load ACC to R01,R01=0AH
LDIA     0FH       ; load 0FH to ACC
XORR     R01       ;execute：R01=05H
```

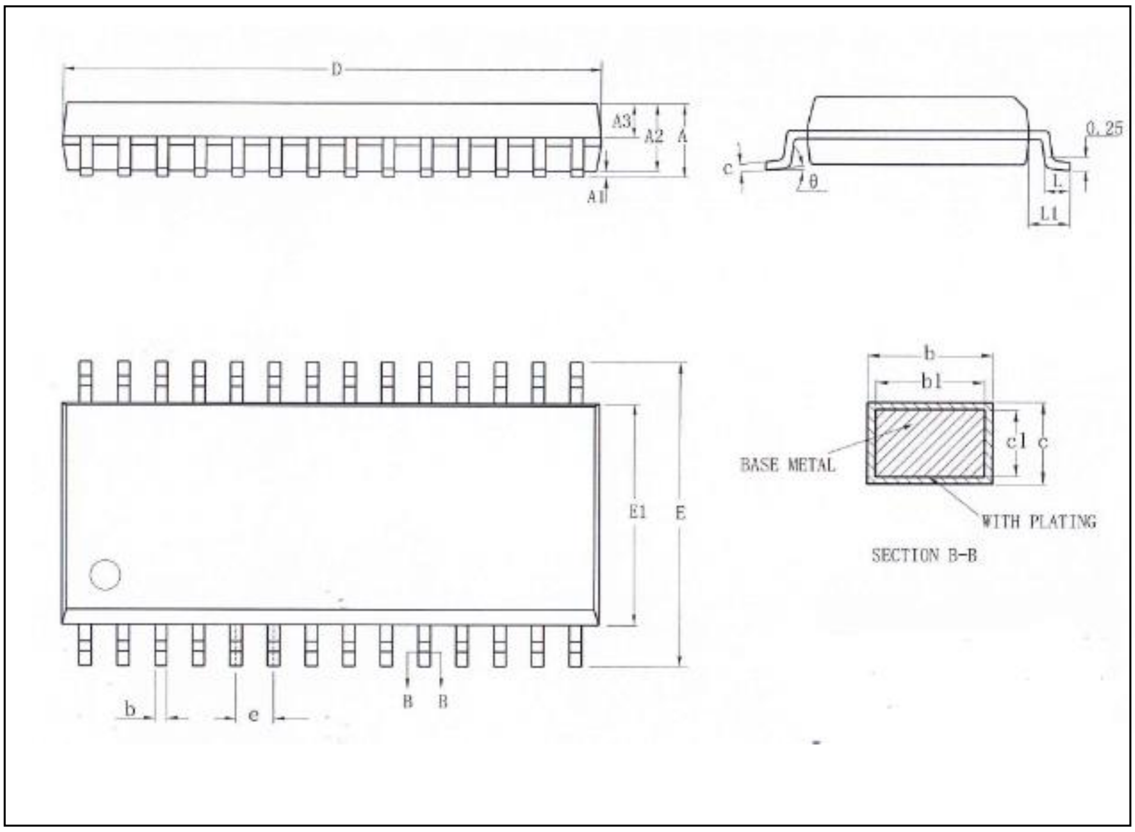
22. Packaging

22.1 SOP20



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	2.65
A1	0.10	-	0.30
A2	2.25	2.30	2.35
A3	0.97	1.02	1.07
b	0.35	-	0.43
b1	0.34	0.37	0.40
c	0.25	-	0.29
c1	0.24	0.25	0.26
D	12.70	12.80	12.90
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
L	0.70	-	1.00
L1	1.40REF		
θ	0	-	8°

22.2 SOP28



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	2.65
A1	0.10	-	0.30
A2	2.25	2.30	2.35
A3	0.97	1.02	1.07
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.25	-	0.29
c1	0.24	0.25	0.26
D	17.90	18.00	18.10
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
L	0.70	-	1.00
L1	1.40REF		
θ	0	-	8°

23. Version Revision

Version number	time	Revised content
V1.0	Aug,2019	Original version
V1.1	May,2020	1.fix the description error in PORTC procedure 2.add 16MHz convertor clock illustration in ADC chapter
V1.2	June,2020	add 32.768kHz crystall oscillator description
V1.3	June,2020	Update program EEPROM read/write example
V1.4	Mar,2022	1. Corrected that RC0IF and TX0IF in PIR1 register are read-only and RC1IF and TX1IF in PIR1 register are read-only 2. Corrected TX0IF description in PIR1 register and TX1IF description in PIR2 register 3. Figs 14-3 and 14-4 corrected for USART asynchronous sending 4. Corrected the sequence number of MSSP chapters
V1.5.0	May,2022	1) Corrected incorrect description of I2CTM master mode receive timing diagram 2) Corrected 14.1.2.3 Receive Interrupt description 3) Corrected FERR frame error bits in RCSTA register are read-only 4) Add clock block diagram 5) Revised the internal high-speed oscillation frequency to F_{HSI}, and revised the clock sources of other modules according to the clock block diagram 6) Revised sleep wake-up waiting time and WDT reset time 7) Revised reset value of LED DATA register 8) Delete description of sleep wake-up in ADC interrupt 9) Modify the description of ANS15 10) Added content in Section 20.6 IIC Electrical Characteristics