SC8F289xB User Manual

Enhanced 8-bit CMOS Microcontroller with Flash Memory

Rev. 2.1.0

Please be reminded about following CMS's policies on intellectual property

＊Cmsemicron Limited (denoted as 'our company' for later use) has already applied for relative patents and entitled legal rights. Any patents related to CMS's MCU or other products is not authorized to use. Any individual, organization or company which infringes our company's interlectual property rights will be forbidden and stopped by our company through any legal actions, and our company will claim the lost and required for compensation of any damage to the company.

＊ The name of Cmsemicron Limited and logo are both trademarks of our company

* Our company preserve the rights to further elaborate on the improvements about product functions, reliability and design in this manual. However, our company is not responsible for any usage about this manual. The applications and their purposes in this manual are just for clarification，our company does not guarantee that these applications are feasible without further improvements and changes，and does not recommend any usage of the products in areas where people's safety is endangered during accident. Our company's products are not authorized to be used for life-saving or life support devices and systems. our company has the right to change or improve the product without any notification，for latest news, please visit our website: www.mcu.com.cn

Please be reminded about following CMS's policies on intellectual property

Manual

1. Product Description

1.1 Features

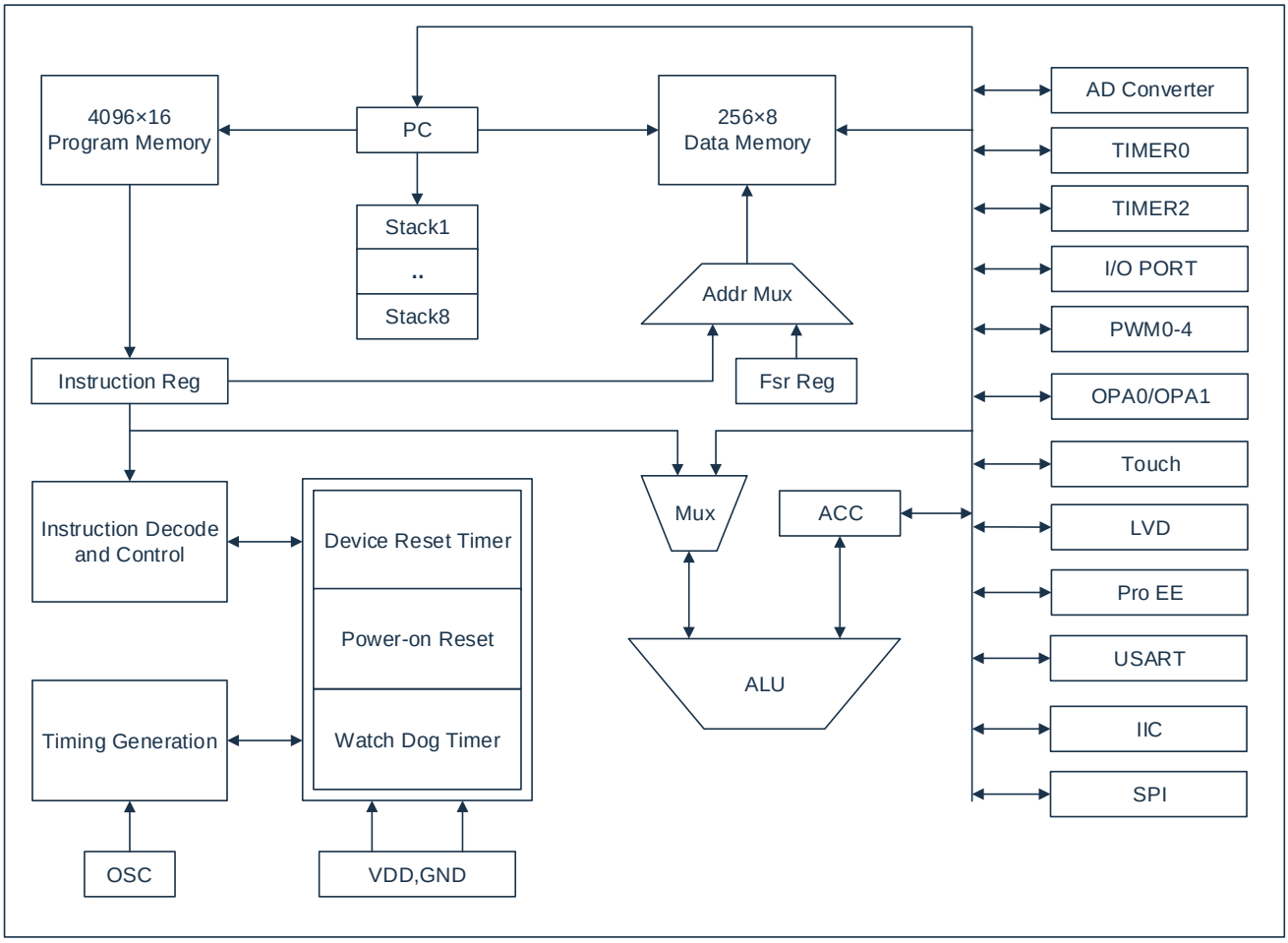
- memory
  - Flash: 4Kx16Bit
  - Universal RAM: 256x8Bit
- 8 level stack buffer
- Clean instructions (66 instructions)
- built-in WDT timer
- built-in low voltage detection circuit
- Interrupt source
  - 2 timer interrupt
  - Interrupt for change in electrical level RA/RB port
  - Other peripherals interrupt
- timer
  - 8-bit timer: TIMER0, TIMER2
  - TIMER2 supports external 32.768kHz crystal oscillator as timer clock source
- built-in LVD mod
  - Choice of voltage:
  - 2.2V/2.4V/2.7V/3.0V/3.3V/3.7V/4.0V/ 4.3V
- built-in 1 IIC communication mod
- built-in 1 SPI communication mod
- Working voltage: 3.0V~5.5V@16MHz/2T
  2.5V~5.5V@16MHz/4T or 8MHz/2T
  2.1V~5.5V@8MHz/4T
- Working temperature: -40℃~85℃
- Internal RC: design frequency of 8MHz/16MHz
- Instructions period (single instruction or double instructions)
- PWM mod with complementary outputs
  - 5 pwm channels which can be configured as 2 groups of complementary outputs
  - Large driving current up to 70mA is able to configurate on RB0/RB1 and RB3/RB4
  - Common cycle and independent duty cycle
- High precision 12-bit ADC
  - Built-in high precision 0.6V reference voltage
  - ±1.5% @VDD=2.5V~4.5V T_A=25℃
  - ±2% @VDD=2.5V~4.5V T_A=-40℃~85℃
  - Internal reference source can be selected: 2.0V/2.4V/3.0V
- built-in 2 high performance operational amplifier
- built-in touch button detection mod
- built-in 1 USART communication mod
  - Supports synchronous master/slave mode and asynchronous duplex mode
- Built-in 32-byte program EEPROM
  - it can be repeatedly burned 100,000 times

Product specification

PRODUCT	ROM	Pro EE	RAM	USART	IIC	SPI	I/O	Touch	PWM	OPA	ADC	PACKAGE
SC8F2890B	4Kx16	32x16	256x8	1	1	1	6	5	4	0	6	SOP8
SC8F2892B	4Kx16	32x16	256x8	1	1	1	14	7	5	2	14	SOP16

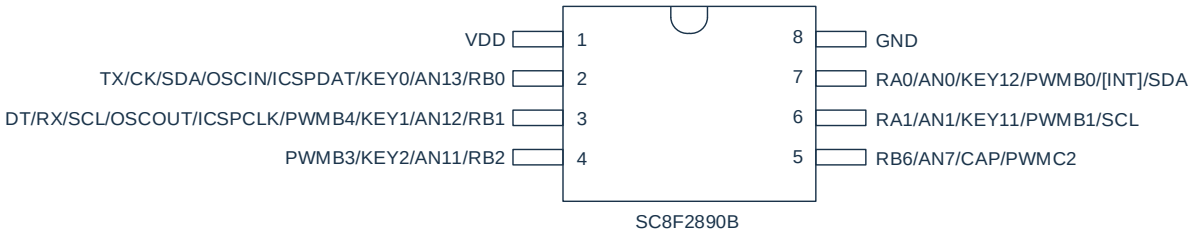
Note: ROM----program memory Pro EE---program EEPROM

1.2 System Structure Diagram



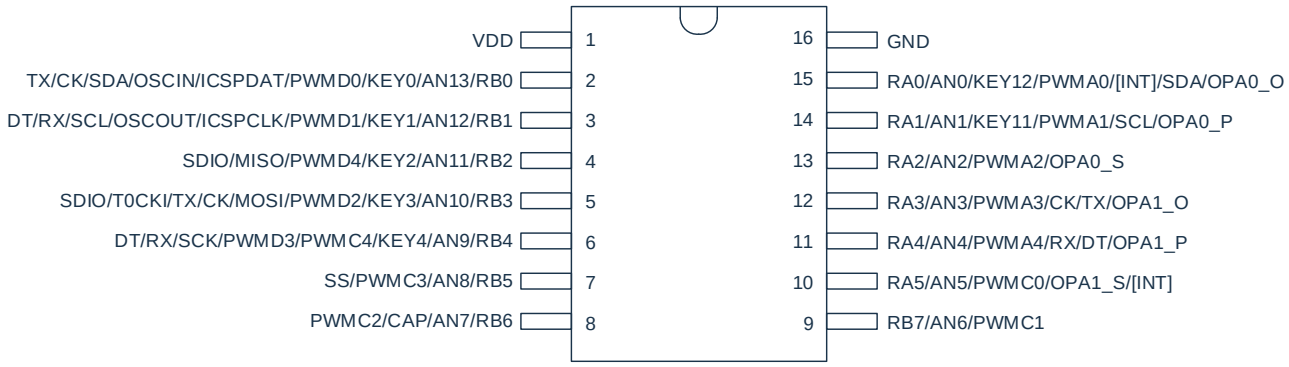
1.3 Pin Allocation

1.3.1 SC8F2890B



SC8F2890B

1.3.2 SC8F2892B



SC8F2892B

Pin description:

Pin name	IO type	description
VDD, GND	P	Voltage input pin and ground
OSCIN/OSCOUT	I/O	32.768kHz Oscillator in/out pin
RA0-RA5	I/O	Programmable in/ push-pull out pin, with pull-up resistance function
RB0-RB7	I/O	Programmable in/push-pull out pin，with pull-up resistance, pull-down resistance, electrical level interrupt function
ICSPCLK	I	Program clock pin
ICSPDAT	I/O	Program data input/output pin
AN0-AN13	I	12 位 ADC input pin
KEY0-KEY4, KEY11-KEY12	I	touch button input pin
CAP	I	touch button reference capacitance input pin
PWMA0-PWMA4	O	Group A PWM0-4 output pin
PWMB0-PWMB1 PWMB3-PWMB4	O	Group B PWM0-1、PWM3-4 output pin
PWMC0-PWMC4	O	Group C PWM0-4 output pin
PWMD0-PWMD4	O	Group D PWM0-4 output pin
TX/CK	O	USART asynchronous transmit output/synchronous clock input/output pin (can be mapped on different IO)
RX/DT	I	USART asynchronous receive input/synchronous data input/output pin (can be mapped on different IO)
INT	I	External interrupt input pin
T0CKI	I	TIMER0 external clock input pin
SCL	I/O	I^2Cclock input/output pin
SDA	I/O	I^2Cdata input/output pin
SCK	I/O	SPI clock input pin
SDIO	I/O	SPI 3-wire mod serial data input/output (can set to different IO)
MISO	I/O	SPI master data input pin、SPI slaver data output pin
MOSI	I/O	SPI slaver data output pin、SPI master data input pin
SS	I	SPI slave choice input pin
OPAx_O	O	Operational amplifier output pin
OPAx_P	I	Operational amplifier positive output pin
OPAx_S	I	Operational amplifier negative 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 SC burner. User cannot visit. It includes the following:

1. OSC (choice of oscillation)
   - INTRC8M F_{HSI} choose internal 8MHz RC oscillation
   - INTRC16M F_{HSI} choose internal 16MHz RC oscillation
2. WDT (watchdog choice)
   - ENABLE Enable watchdog timer
   - DISABLE Disable watchdog timer
3. PROTECT (encryption)
   - DISABLE Disable FLASH code encryption
   - ENABLE Enable FLASH code encryption, after which the read value from burning the simulator is uncertain.
4. LVR_SEL (low voltage detection selection)
   - 2.1V
   - 2.5V
   - 3.0V
5. SLEEP_LVREN（LVR choice under sleep status）
   - DISABLE Disable LVR under sleep status
   - ENABLE Enable LVR under sleep status
6. F_{CPU}_DIV (command clock frequency division)
   - 4T 4 frequency division, $F_{CPU}=F_{SYS}/4$
   - 2T 2 frequency division, $F_{CPU}=F_{SYS}/2$
7. EXTINT_SEL（external interrupt choice）
   - RA0 Select RA0 as external interrupt pin
   - RA5 Select RA5 as external interrupt pin
8. USART_SEL (USART port selection)
   - RA3/RA4 (this selection only valid for SC8F2892B)
   - RB3/RB4
   - RB0/RB1
9. IIC_SEL (IIC port selection)
   - RA0/RA1
   - RB0/RB1
10. SPI_DIO_SEL（SPI3 line mode DIO pin selection, only valid for SC8F2892B）
    - RB2
    - RB3
11. RB4_ISEL (RB4 current selection)
    - 70mA Push-pull current of RB4 is 70mA
    - Normal Push-pull current of RB4 is equivalent to normal IO
12. RB3_ISEL (RB3 current selection)
    - 70mA Push-pull current of RB3 is 70mA

- Normal Push-pull current of RB3 is equivalent to normal IO

13. RB1_ISEL (RB1 current selection)
    - 70mA Push-pull current of RB1 is 70mA
    - Normal Push-pull current of RB1 is equivalent to normal IO
14. RB0_SEL (RB0 current selection)
    - 70mA Push-pull current of RB0 is 70mA
    - Normal Push-pull current of RB0 is equivalent to normal IO
15. ICSPPORT_SEL (simulation port selection)
    - ICSP ICSPCLK, DAT port keep as simulation port, all functions disabled
    - NORMAL ICSPCLK, DAT port as normal port

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 latest version of firmware can be burned into the MCU.

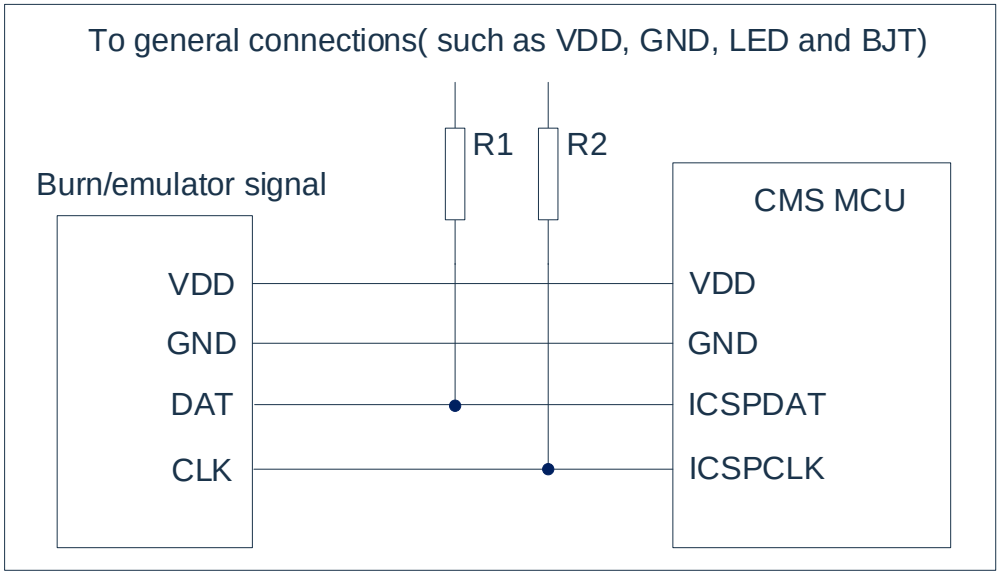


Fig 1-1：Typical connection for online serial programming

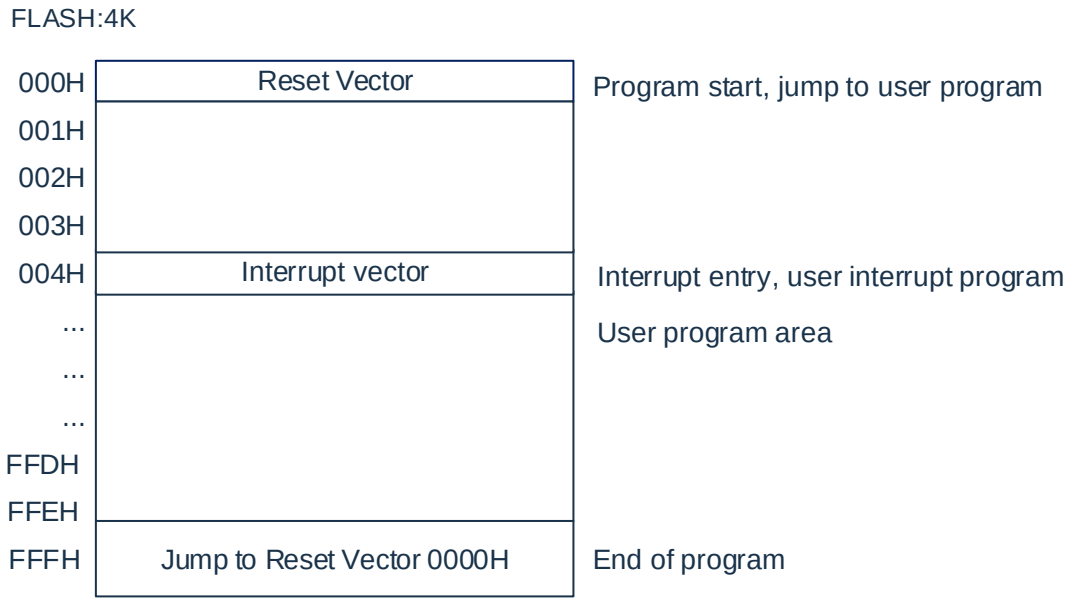
In the above figure, R1 and R2 are the electrical isolation devices, normally represented by resistor with the following resistance: R1≥4.7K, R2≥4.7K.

2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program Memory

SC8F289xB 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 register will be recovered to default value. PD and TO from STATUS register can determine the which reset is performed from above. The following program illustrates 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 PC will be saved to stack buffer and jump to 0004H to execute interrupt service program. All interrupt will 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       ; system reset vector
        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 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 value of ACC to PCL. If the value of ACC isn, then PCL+ACC represent the current address plus n. After the execution 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 SC8F289xB

address		address		address		address	
INDF	00H	INDF	80H	INDF	100H	INDF	180H
TMR0	01H	OPTION_REG	81H		101H		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		105H		185H
PORTB	06H	TRISB	86H		106H		186H
WPUA	07H	WPDB	87H	PIR2	107H		187H
WPUB	08H	OSCCON	88H	PIE2	108H	IOCA	188H
IOCB	09H	----	89H		109H		189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
PIR1	0CH	EECON1	8CH	IICCON	10CH		18CH
PIE1	0DH	EECON2	8DH	IICCON2	10DH		18DH
PWMD23H	0EH	EEDAT	8EH	IICBUF	10EH		18EH
PWM01DT	0FH	EEDATH	8FH	IICSTAT	10FH		18FH
PWM23DT	10H	EEADR	90H		110H		190H
TMR2	11H	PR2	91H		111H		191H
T2CON	12H	KEYCON0	92H		112H		192H
PWMCON0	13H	KEYCON1	93H		113H		193H
PWMCON1	14H	KEYDATAL	94H		114H		194H
PWMTL	15H	KEYDATAH	95H		115H		195H
PWMTH	16H	EEADRH	96H		116H		196H
PWMD0L	17H	KEYCON2	97H	TXSTA	117H		197H
PWMD1L	18H	OPA0CON	98H	RCSTA	118H		198H
PWMD2L	19H	OPA0ADJ	99H	SPBRG	119H		199H
PWMD3L	1AH	OPA1CON	9AH	TXREG	11AH		19AH
PWMD4L	1BH	OPA1ADJ	9BH	RCREG	11BH		19BH
PWMD01H	1CH	ADCON1	9CH	SPIBUF	11CH	----	19CH
PWMCON2	1DH	ADCON0	9DH	SPICON	11DH	----	19DH
	1EH	ADRESL	9EH	SPICON2	11EH	----	19EH
	1FH	ADRESH	9FH	LVDCON	11FH	----	19FH
Universal register 96 byte	20H 6FH	Universal register 80byte	A0H EFH	Universal register 80byte	120H 16FH	----	1A0H 1EFH
	70H -- 7FH	Fast memory space 70H-7FH	F0H -- FFH	Fast memory space 70H-7FH	170H -- 17FH	Fast memory space 70H-7FH	1F0H -- 1FFH
BANK0		BANK1		BANK2		BANK3	

Data memory consists of 512×8 bits. It can be divided into two 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 SC8F289xB 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	----	----	RA5	RA4	RA3	RA2	RA1	RA0	--xxxxxx
06H	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxxxxxx
07H	WPUA	----	----	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	--000000
08H	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	00000000
09H	IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0	00000000
0AH	PCLATH	----	----	----	----	Write buffer of higher 4 bits of program counter				-----0000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0CH	PIR1	----	EEIF	RCIF	TXIF	SPIIF	PWMIF	TMR2IF	ADIF	-0000000
0DH	PIE1	----	EEIE	RCIE	TXIE	SPIIE	PWMIE	TMR2IE	ADIE	-0000000
0EH	PWMD23H	----	----	PWMD3[9:8]		----	----	PWMD2[9:8]		--00—00
0FH	PWM01DT	----	----	PWM01 dead-time						--00000
10H	PWM23DT	----	----	PWM23 dead-time						--00000
11H	TMR2	TIMER2 mod register								00000000
12H	T2CON	CLK_SEL	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	00000000
13H	PWMCON0	CLKDIV[2:0]			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN	00000000
14H	PWMCON1	PWMIO_SEL[1:0]		PWM2DTEN	PWM0DTEN	----	----	DT_DIV[1:0]		0000—00
15H	PWMTL	Lower bit of PWM period register								00000000
16H	PWMTH	----	----	PWMD4[9:8]		----	----	PWMT9	PWMT8	--00--00
17H	PWMD0L	Lower bit of PWM0 duty register								00000000
18H	PWMD1L	Lower bit of PWM1 duty register								00000000
19H	PWMD2L	Lower bit of PWM2 duty register								00000000
1AH	PWMD3L	Lower bit of PWM3 duty register								00000000
1BH	PWMD4L	Lower bit of PWM4 duty register								00000000
1CH	PWMD01H	----	----	PWMD1[9:8]		----	----	PWMD0[9:8]		--00—00
1DH	PWMCON2	----	----	----	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR	----00000

Summary of special registers in SC8F289xB 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	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	-1111011
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	----	----	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	--111111
86H	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	11111111
87H	WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0	00000000
88H	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	SWDTEN	----	-101—0-
8AH	PCLATH	----	----	----	----	Write buffer of higher 4 bits of program counter				-----0000
8BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	EECON1	EEPGD	----	----	----	WRERR	WREN	WR	RD	0—0x000
8DH	EECON2	EEPROM control register2 (not physical register)								--------
8EH	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx
8FH	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	xxxxxxxx
90H	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	00000000
91H	PR2	TIMER2 period register								00000000
92H	KEYCON0	KDONE	----	KCAPK[2:0]			KOUT	KCAP	KEN	0-000000
93H	KEYCON1	KVREF[1:0]		KCLK[1:0]		KCHS[3:0]				00000000
94H	KEYDATAL	Result register of touch button mod Lower bit								xxxxxxxx
95H	KEYDATAH	Result register of touch button mod Higher bit								xxxxxxxx
96H	EEADRH	----	----	----	----	EEADRH3	EEADRH2	EEADRH1	EEADRH0	----0000
97H	KEYCON2	CAP_LVBO[2:0]			----	----	----	----	TKEN	000—--0
98H	OPA0CON	OPA0EN	OPA0OEN	CMP0MODE	OPA0_ADC	----	----	----	OPA0FT	0000---0
99H	OPA0ADJ	OPA0OUT	OPA0COFM	OPA0CRS	OPA0_ADJ[4:0]					000xxxxx
9AH	OPA1CON	OPA1EN	OPA1OEN	CMP1MODE	OPA1_ADC	----	----	----	OPA1FT	0000---0
9BH	OPA1ADJ	OPA1OUT	OPA1COFM	OPA1CRS	OPA1_ADJ[4:0]					000xxxxx
9CH	ADCON1	ADFM	----	----	----	----	LDO_EN	LDO_SEL[1:0]		0----000
9DH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON	00000000
9EH	ADRESL	Lower bit of A/D result register								xxxxxxxx
9FH	ADRESH	Higher bit of A/D result register								xxxxxxxx

Summary of special registers in SC8F289xB 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
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
107H	PIR2	----	TKIF	----	IICIF	BCLIF	----	RACIF	LVDIF	-0-00--0
108H	PIE2	----	TKIE	----	IICIE	BCLIE	----	RACIE	LVDIE	-0-00--0
10AH	PCLATH	----	----	----	----	Write buffer of higher 4 bits of program counter				-----0000
10BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
10CH	IICCON	IICWCOL	SSPOV	IICEN	IICCKP	IICTOS[1:0]		IICM[1:0]		00010000
10DH	IICCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	00000000
10EH	IICBUF	IIC receive buffer /transmit register								
10FH	IICSTAT	----	IDLE	D/A	P	S	R/W	IICTOF	BF	-0000000
117H	TXSTA	CSRC	TX9EN	TXEN	SYNC	SCKP	STOPBIT	TRMT	TX9D	00000010
118H	RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D	00001000
119H	SPBRG	USART baud rate register								00000000
11AH	TXREG	USART transmit data register								00000000
11BH	RCREG	USART receive data register								xxxxxxxx
11CH	SPIBUF	SPI receive buffer /transmit register								xxxxxxxx
11DH	SPICON	SPIWCOL	SPIOV	SPIEN	SPICKP	SPIM[3:0]				00000000
11EH	SPICON2	----	CKE	MODE	----	----	----	----	SPIBF	-00----0
11FH	LVDCON	LVD_RES	----	----	----	LVD_SEL[2:0]			LVDEN	0---0000

Summary of special registers in SC8F289xB 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
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
188H	IOCA	----	----	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0	--000000
18AH	PCLATH	----	----	----	----	Write buffer of higher 4 bits of program counter				-----0000
18BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	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 cause 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 that 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 memory nor 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 diagram illustrates its working principle.

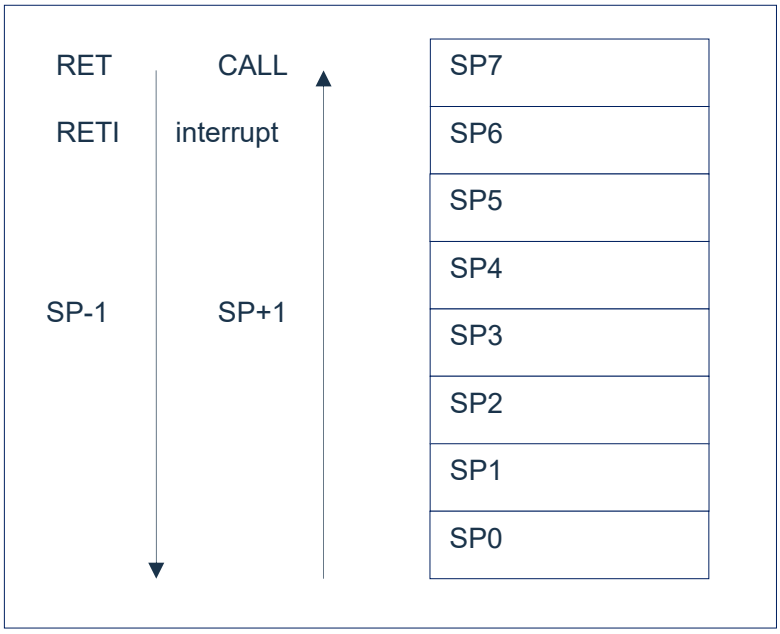


Fig 2-2：stack buffer working principle

Stack buffer will follow one principle: ‘first in last out’

Note: stack buffer has only 8 levels, if the stack is full and interrupt happens which cannot 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 an instruction 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 cannot 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 in zero；
1= Result is 0；
0= Result is not 0
Bit1 DC: Carry bit；
1= When carry happens to higher bits or no borrow happens in Lower 4 bits in the result；
0= When no carry happens to higher bits or borrow 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 and 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 register can 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	---	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Read/write	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	1	1	1	1	0	1	1

Bit7 Not used

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	WDT frequency 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 an 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 bit 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 address in 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 12 Bit，user can access lower 8 bits through PCL (02H). The higher 4 bits cannot be accessed. It can hold address for 4K×16Bit program. Passing a value to PCL will cause a short jump which range until the 256 address of 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[11:8] unchange，PC[7:0]=user defined value；
JP	PC=program defined value；
Other instructions	PC=PC+1；

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 of 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 should be 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 interrupt so 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 chip 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

SWDTEN: Software enable or disable watchdog timer bit
1= Enable WDT
0= Disable WDT (reset value)

Note:

1. SWDTEN located in OSCCON register Bit1.
2. 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 signals 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 instruction is 1 instruction period. If 1 instruction cause PC address to change (such as JP), then the pre-loaded instruction code is useless and 2 instruction period is needed to complete this instruction. This is why every operation on PC consumes 2 clock period.

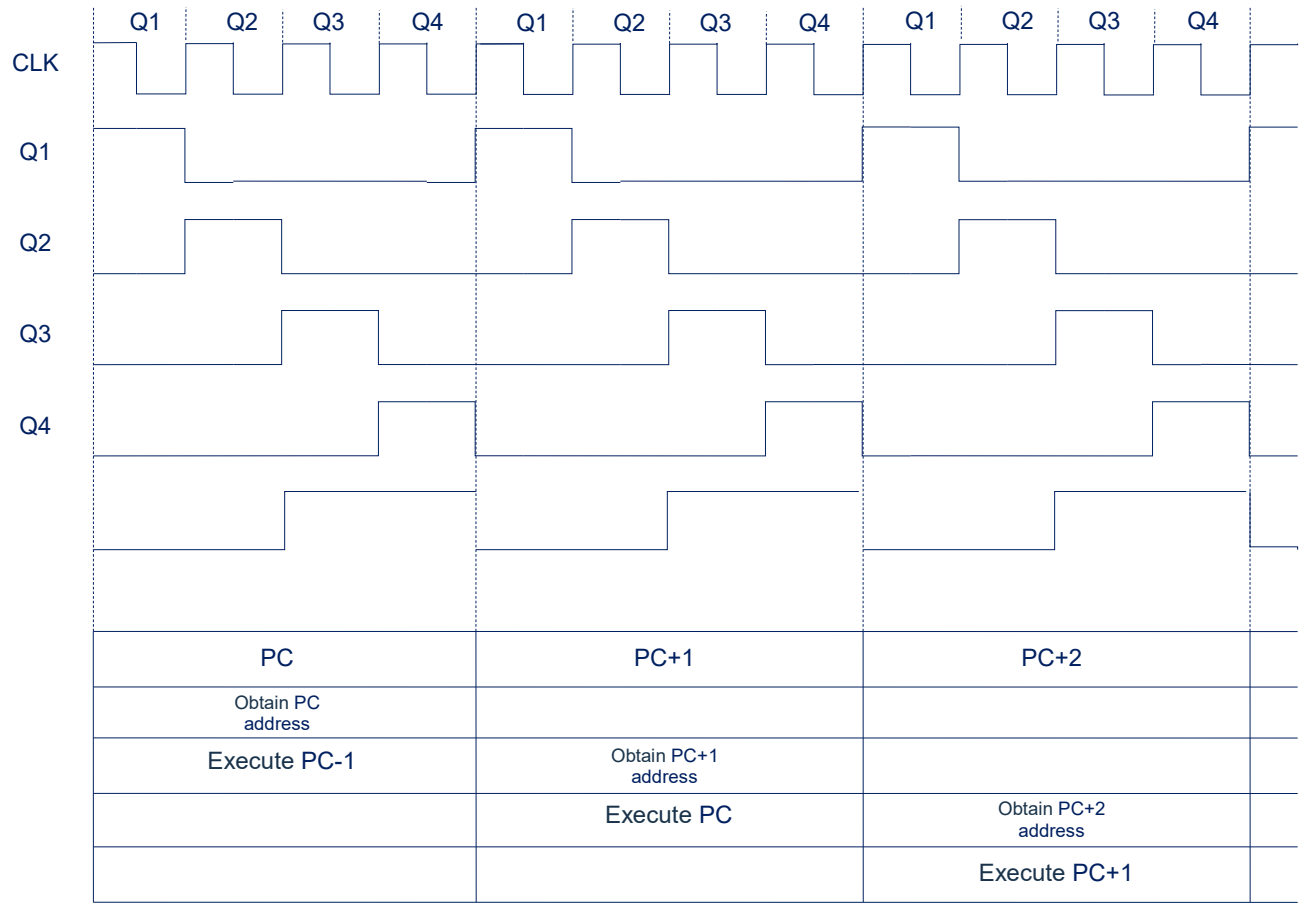


Fig 3-1：time series for clock and instruction period

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 integrated with 8MHz/16MHz internal RC 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.

3.3 Reset Time

Reset Time is the time for chip to change from reset to stable oscillation. The value is about 18ms.

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 oscillation in the software.

OSCCON (88H)

88H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	SWDTEN	---
R/W	---	R/W	R/W	R/W	---	---	R/W	---
reset value	---	1	0	1	---	---	0	---

Bit7 Not used，read 0

Bit6~Bit4 IRCF<2:0>: Selection bit for frequency division of Internal oscillator

111= $F_{SYS} = F_{HSI}$ /1

110= $F_{SYS} = F_{HSI}$ /2

101= $F_{SYS} = F_{HSI}$ /4 (default)

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~Bit2 Not used

Bit1 SWDTEN: Software enable or disable watchdog timer bit

1= Enable WDT

0= Disable WDT (reset value)

Bit0 Not used

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

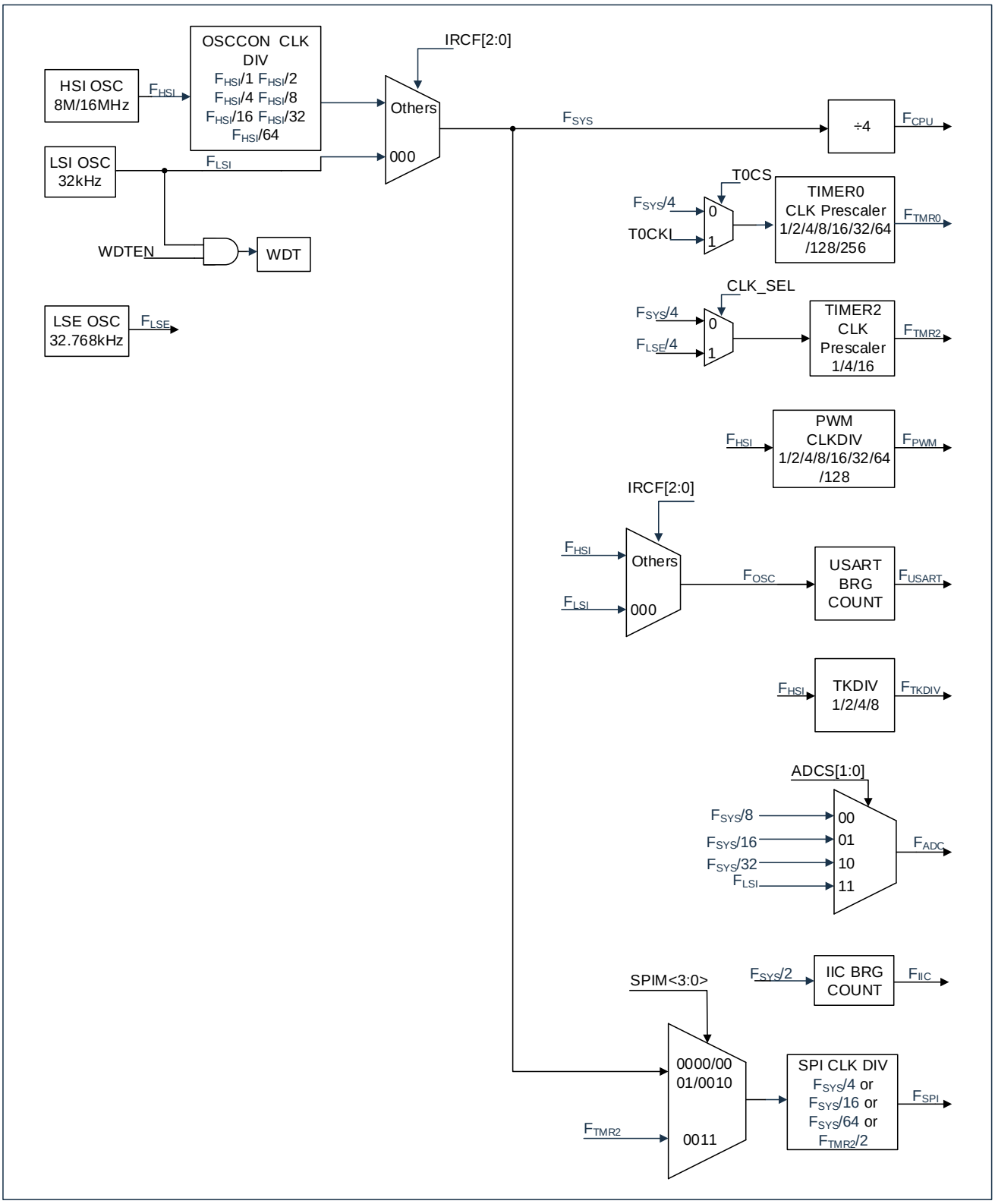


Fig 3-2：clock block diagram

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 default condition, program stops executing and PC is cleared. When finishing resetting, program executes from reset vector 0000H. TO and 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 systems 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 register set 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.

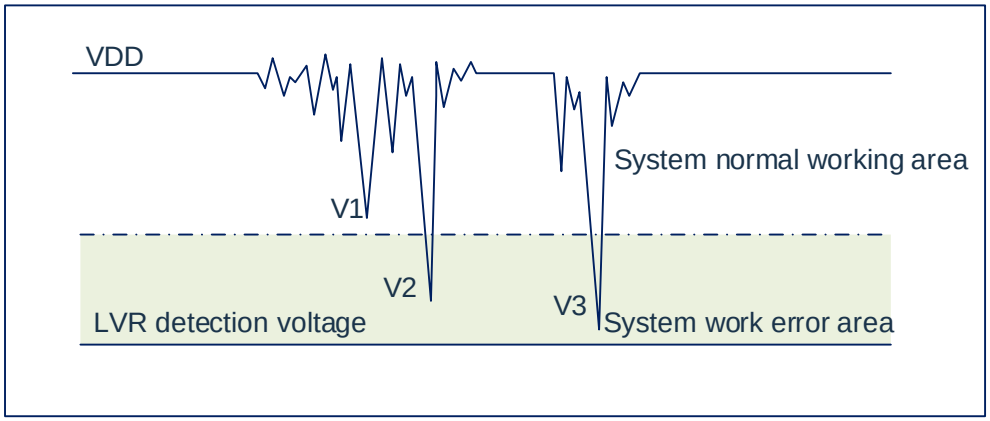


Fig 4-1: power off reset

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 unable working 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 voltage, 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 voltage 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 watchdog timer. If error happens and system is under unknown status, watchdog timer overflow and then system reset. After watchdog 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 register set 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 mode when 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 impedance).

Under sleep mode, to avoid current consumption, all I/O pin should keep at VDD or GND to make sure no external circuit is consuming the current from I/O pin. To avoid input pin, suspend and invoke current, high impedance 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
3. Other peripheral interrupt

The above 3 events are regards as the extension of 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 first. If it is intended to awaken the system using interrupt, the corresponding enable 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 jump 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 executed to make sure WDT is cleared.

5.4 Sleep Mode Application

Before system enters sleep mode，if user wants small sleep 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
        …                   ;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})	$F_{SYS}=F_{HSI}$	$T_{WAIT}=(1032*1+16)/F_{HSI}$
	$F_{SYS}= F_{HSI}/2$	$T_{WAIT}=(1032*2+16)/F_{HSI}$
	…	…
	$F_{SYS}= F_{HSI}/64$	$T_{WAIT}=(1032*64+16)/F_{HSI}$
Internal low speed RC oscillation ($F_{LFINTOSC}$)	----	$T_{WAIT}=15/F_{LFINTOSC}$

6. I/O Port

Chip has 2 I/O port: PORTA、PORTB (max. of 14 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，PWM，OPA0_O，INT，KEY12	I/O
	1	Schmitt trigger input，push-pull output，AN1，PWM，OPA0_P，KEY11	I/O
	2	Schmitt trigger input，push-pull output，AN2，PWM，OPA0_S	I/O
	3	Schmitt trigger input，push-pull output，AN3，PWM，OPA1_O	I/O
	4	Schmitt trigger input，push-pull output，AN4，PWM，OPA1_P	I/O
	5	Schmitt trigger input，push-pull output，AN5，PWM，OPA1_S，INT	I/O
PORTB	0	Schmitt trigger input，push-pull output，AN13，PWM，ICSPDAT，OSCI，KEY0	I/O
	1	Schmitt trigger input，push-pull output，AN12，PWM，ICSPCLK，OSCO，KEY1	I/O
	2	Schmitt trigger input，push-pull output，AN11，PWM，KEY2	I/O
	3	Schmitt trigger input，push-pull output，AN10，PWM，KEY3	I/O
	4	Schmitt trigger input，push-pull output，AN9，PWM，KEY4	I/O
	5	Schmitt trigger input，push-pull output，AN8，PWM	I/O
	6	Schmitt trigger input，push-pull output，AN7，PWM，CAP	I/O
	7	Schmitt trigger input，push-pull output，AN6，PWM	I/O

＜Table 6-1: port configuration summary＞

6.1 I/O Port Structure

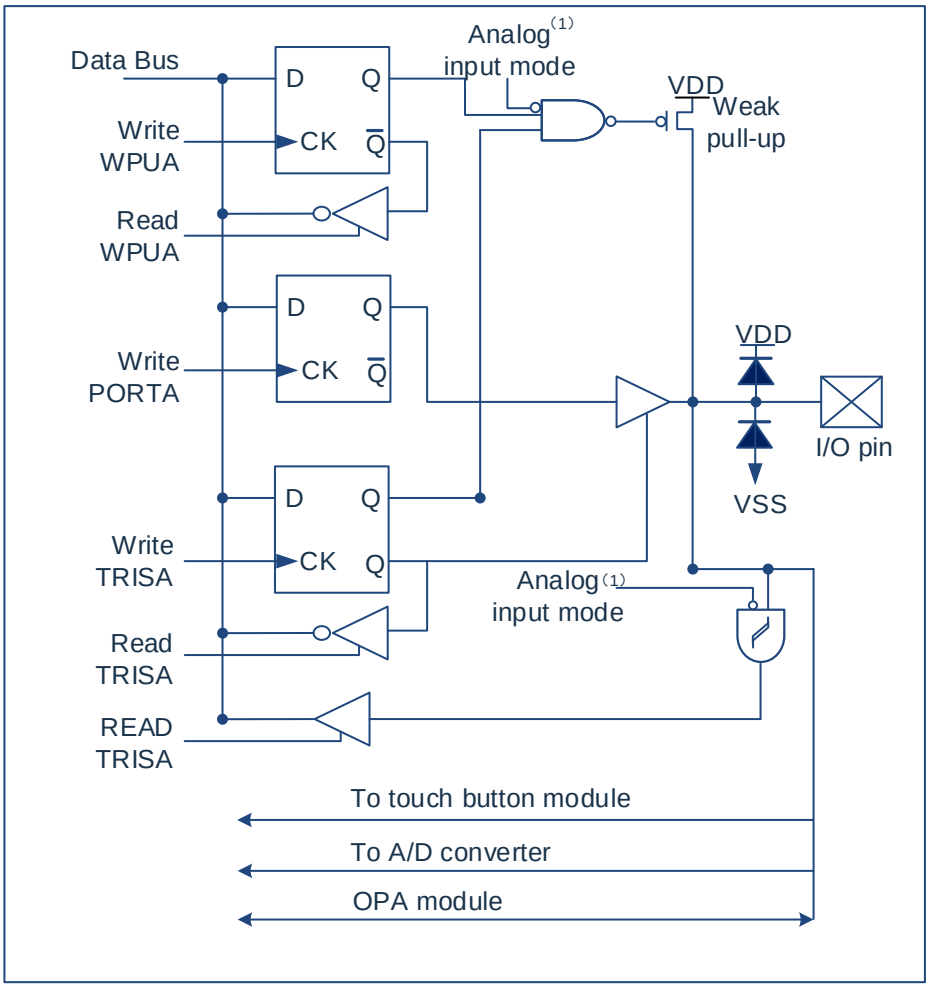


Fig 6-1: I/O port structure (1)

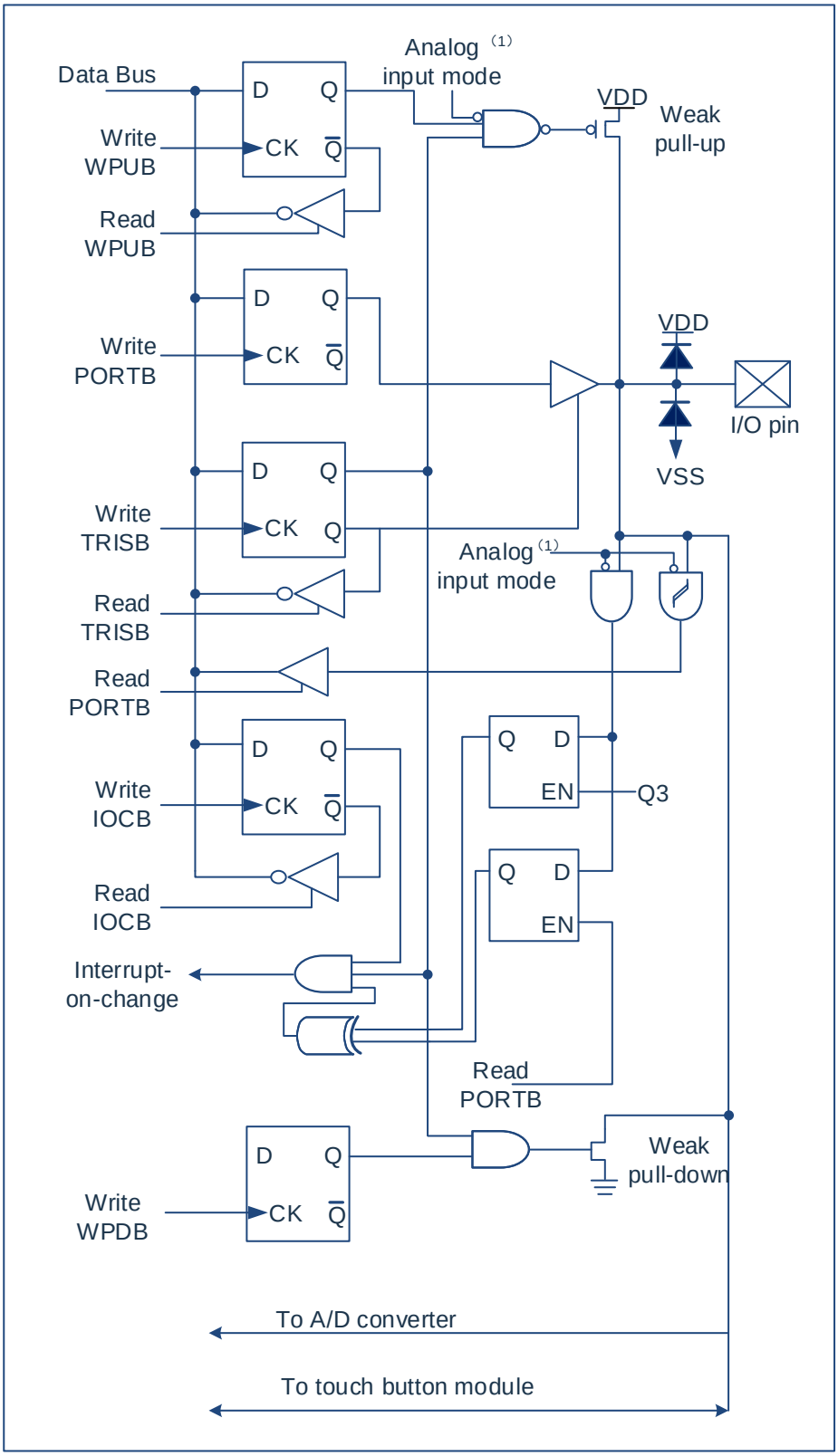


Fig 6-2: I/O port structure (2)

6.2 PORTA

6.2.1 PORTA Data and Direction Control

PORTA is 6 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 operations are read-change-write. Hence, write 1 port means read the pin electrical level of the port, change the value and write the value into port latch. Even when PORTA pin is used as analog input, TRISA register still 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.

Registers related to PORTA ports are PORTA、TRISA、WPUA and etc.

PORTA data register PORTA (05H)

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

Bit7~Bit6 Not used.

Bit5~Bit0 PORTA<5:0>: PORTAI/O pin bit；

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	----	----	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
R/W	----	----	R/W	R/W	R/W	R/W	R/W	R/W
reset value	----	----	1	1	1	1	1	1

Bit7~Bit6 Not used.

Bit5~Bit0 TRISA<5:0>: PORTA；

1= PORTA pin set to be input；

0= PORTA pin set to be output

example：procedure for PORTA

```
LDIA    B'00110000'     ;set PORTA<3:0> as output port，PORTA<5:4>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<5:4>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<5:0> enable or disable each weak pull up. When the port pin is configured as output, its weak pull up will be automatically cut off.

PORTA pull up resistance register WPUA (07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	----	----	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	----	----	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~Bit0 WPUA<5:0>: 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 Level Change Interrupt

All PORTA pins can be individually configured as level change interrupt pins. The control bit IOCA<7:0> 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 PORTA was read last time. Perform a logical OR operation with the output "mismatch" of the last read operation to set the PORTA level change interrupt flag (PIR2) in the RACIF 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 PORTA. This will end the mismatch state of the pin level.

-Clear the flag bit RAIF.

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

PORTA level change interrupt register IOCA (188H)

188H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCA	----	----	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0
R/W	----	----	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~Bit0 IOCA<5:0> Control bit of level change interrupt of PORTA

1= enable level change interrupt

0= disable level change interrupt

6.3 PORTB

6.3.1 PORTB Data and Direction

PORTB is a 8 Bit wide bi-directional port. The corresponding data direction register is TRISB. Set a bit in TRISB to 1 (=1) to make the corresponding PORTB pin 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 port data 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, IOCB, etc.

PORTB data register PORTB (06H)

06H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
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 PORTB<7:0>: 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	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
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 TRISB<7:0>: 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 Down Resistance

Each PORTB pin has an internal weak pull-down that can be individually configured. The control bits WPDB<7:0> 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 register WPDB (87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0
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 WPDB<7:0>: 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.

6.3.3 PORTB Pull up Resistance

Each PORTB pin has an internal weak pull up that can be individually configured. The control bits WPUB<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. At power-on reset, weak pull up is prohibited by the RBPU bit of the OPTION_REG register.

PORTB pull up resistance register WPUB (08H)

08H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
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 WPUB<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 prohibited.

6.3.4 PORTB Level Change Interrupt

All PORTB pins can be individually configured as level change interrupt pins. The control bit IOCB<7:0> 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 under voltage 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 (09H)

09H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
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 IOCB<7:0> Control bit of level change interrupt of PORTB

1= enable level change interrupt

0= disable level change interrupt

6.4 I/O Usage

6.4.1 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
SET     PORTA       ;set all output port of PORTA as 1
SETB    PORTB,1     ;set PORTB.1as 1
```

6.4.2 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.

6.5 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.

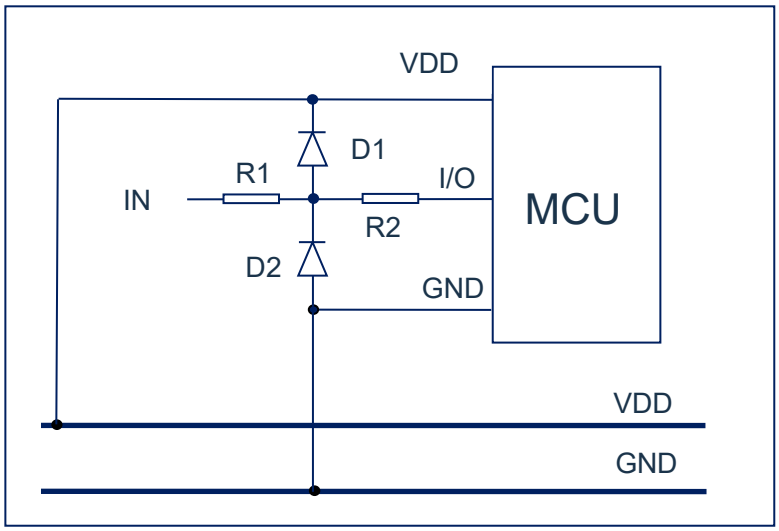


Fig 6-3: The input voltage is not within the specified range

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
- PORTA level change interrupt
- PORTB level change interrupt
- Touch button detection over interrupt
- USART receive/transmit interrupt
- SPI receive/transmit interrupt
- A/D interrupt
- PWM interrupt
- INT interrupt
- LVD interrupt
- IIC receive /transmit interrupt
- IIC bus conflict 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 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.

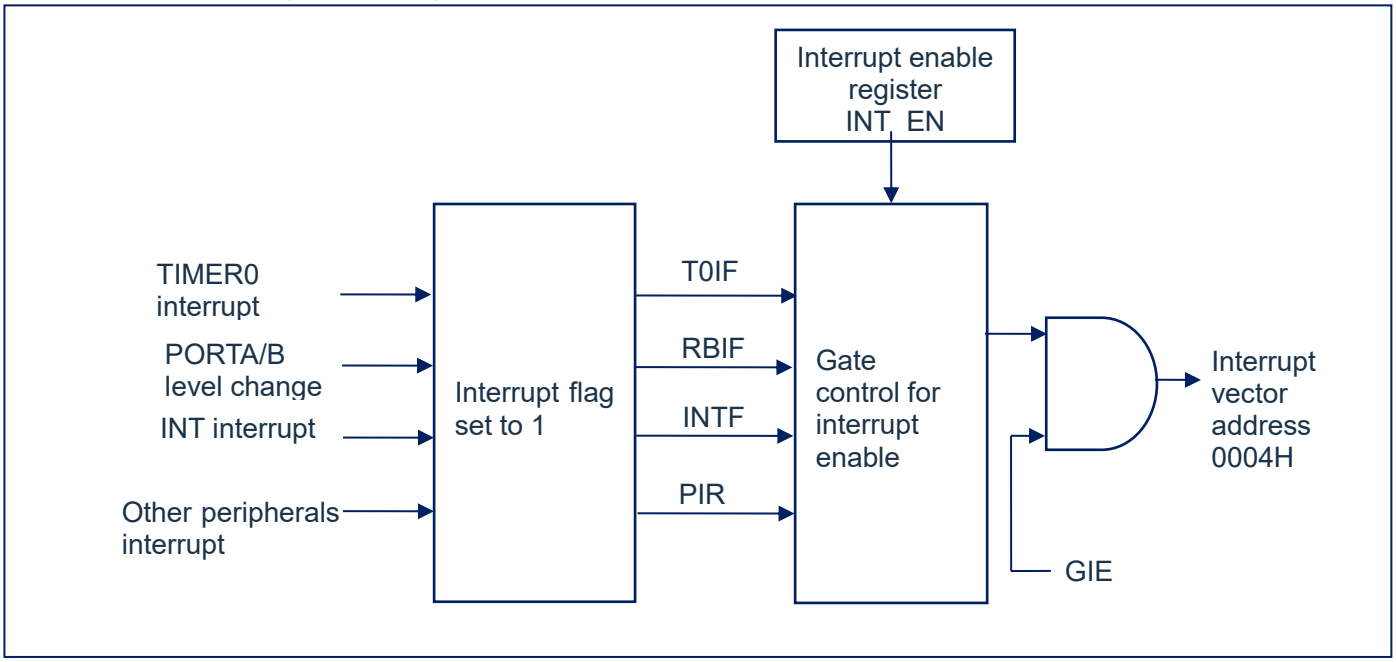


Fig 7-1：interrupt theory

7.2 Interrupt control Register

7.2.1 Interrupt Control Register

The interrupt control register INTCON 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 interrupt enable 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 (0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	----	EEIE	RCIE	TXIE	SPIIE	PWMIE	TMR2IE	ADIE
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 EEIE: EEDATA interrupt enable bit
- 1= enable EEDATA write operation interrupt
- 0= disable EEDATA write operation interrupt

Bit5 RCIE: USART receive interrupt enable bit；
- 1= enable USART receive interrupt；
- 0= disable USART receive interrupt.

Bit4 TXIE: USART transmit interrupt enable bit；
- 1= enable USART transmit interrupt；
- 0= disable USART transmit interrupt.

Bit3 SPIIE: SPI interrupt enable bit；
- 1= enable SPI interrupt；
- 0= disable SPI interrupt.

Bit2 PWMIE: PWM interrupt enable bit；
- 1= enable PWM interrupt；
- 0= disable PWM 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 ADIE: A/D converter (ADC)interrupt enable bit；
- 1= enable ADC interrupt；
- 0= disable ADC interrupt

Peripherals interrupt enable register PIE2 (108H)

108H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	---	TKIE	---	IICIE	BCLIE	---	RACIE	LVDIE
R/W	---	R/W	---	R/W	R/W	---	R/W	R/W
Reset value	---	0	---	0	0	---	0	0

Bit7 Not used.

Bit6 TKIE: Touch button detection over interrupt enable bit
1= enable touch button detection over interrupt
0= disable touch button detection over interrupt

Bit5 Not used.

Bit4 IICIE: IIC interrupt enable bit
1= enable IIC interrupt
0= disable IIC interrupt

Bit3 BCLIE: Bus conflict interrupt enable bit
1= enable bus conflict interrupt
0= disable bus conflict interrupt

Bit2 Not used.

Bit1 RACIE : PORTA level change interrupt flag bit；
1= PORTA level change interrupt enable bit；
0= Enable PORTA level change interrupt；

Bit0 LVDIE: LVD interrupt enable bit
1= enable LVD interrupt
0= disable LVD 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	---	EEIF	RCIF	TXIF	SPIIF	PWMIF	TMR2IF	ADIF
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.

Bit6 EEIF: Program EEPROM write operation interrupt bit;
1= program EEPROM write operation complete (must clear through software);
0= program EEPROM write operation not complete or not start.

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

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

Bit3 SPIIF: SPI interrupt flag bit.
1= SPI receive /transmit interrupt happens (must clear through software);
0= No SPI interrupt condition is met.

Bit2 PWMIF: PWM interrupt flag bit.
1= PWM interrupt happens (must clear through software);
0= PWM interrupt not happen

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

Bit0 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.

Peripherals interrupt request registerPIR2 (107H)

107H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	---	TKIF	---	IICIF	BCLIF	---	RACIF	LVDIF
R/W	---	R/W	---	R/W	R/W	---	R/W	R/W
Reset value	---	0	---	0	0	---	0	0

Bit7 Not used.

Bit6 TKIF: Touch button detection over interrupt flag bit

1= Touch button detection over interrupt happens (must clear through software)；

0= Touch button detection over interrupt not happen.

Bit5 Not used.

Bit4 IICIF: IIC interrupt flag bit

1= The IIC interrupt condition is met. Before returning from the interrupt service program, it must clear through software. The conditions for making this bit 1 are:：

- I^2C slave/master control；
- transmit/receive happens；
- I^2C master control；
- The start condition that occurs is done by IIC mod；
- The stop condition that occurs is completed by IIC mod；
- The restart condition that occurs is done by IIC mod；
- The respond condition that occurs is done by IIC mod；
- The start condition occurs when the IIC mod is idle (multi-host system);
- The stop condition occurs when the IIC mod is idle (multi-host system);.

0= No IIC interrupt condition is met.

Bit3 BCLIF: bus conflict interrupt flag bit;

1= When configured as I^2Cmaster control mode, bus conflict happens in IIC;

0= No bus conflict.

Bit2 Not used.

Bit1 RACIF PORTA level change interrupt flag bit；

1= The level of at least one pin in the PORTA port has changed (must clear through software)；

0= None of the PORTA universal I/O pin status has changed.

Bit0 LVDIF LVD interrupt flag bit；

1= Supply voltage lower than setting voltage by LVD (must clear through software)；

0= Supply voltage higher than setting voltage by LVD .

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 interrupt service 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 service 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.

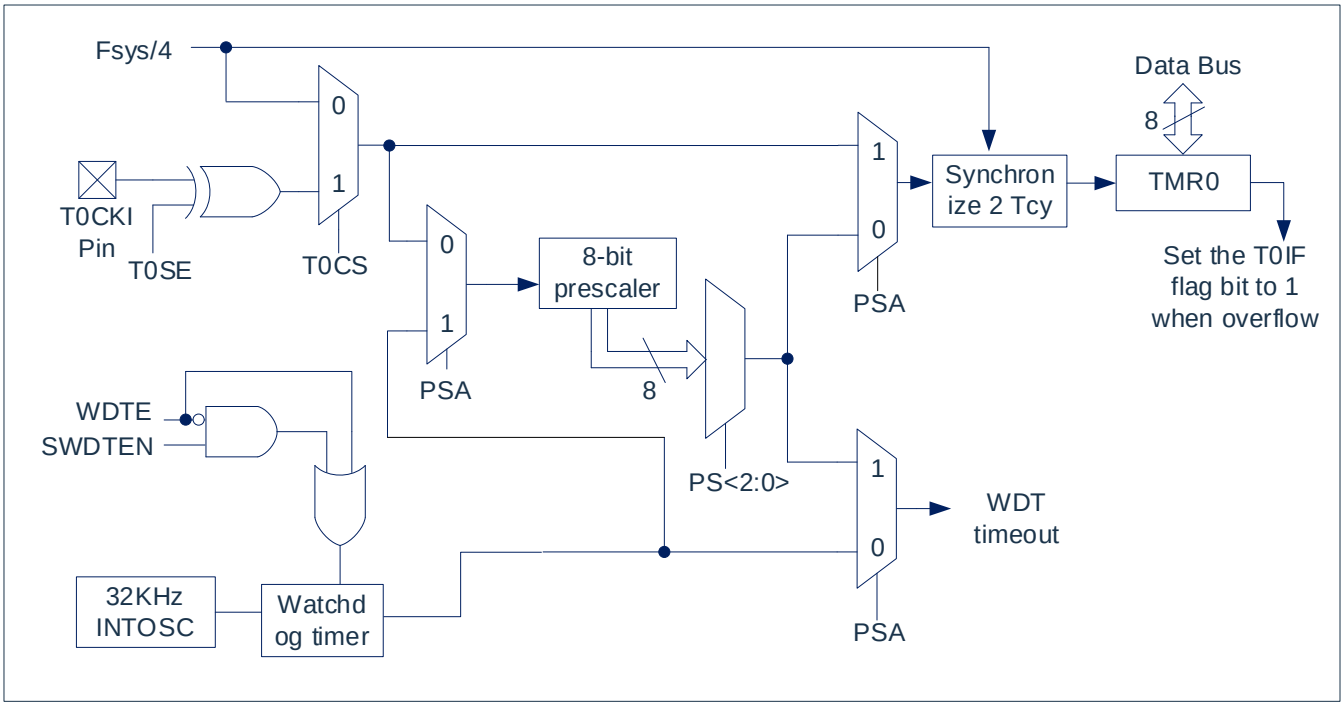


Fig 8-1: TIMER0/WDT mod structure

Note:
1. T0SE, T0CS, PSA, PS<2:0> are the bits in OPTION_REG register.
2. SWDTEN is a bit in the OSCCON 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 2.6 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	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Read/write	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	----	1	1	1	1	0	1	1

Bit7 Not used.

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= Internal instruction period clock (F_{CPU}).

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. TIMER2

9.1 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).

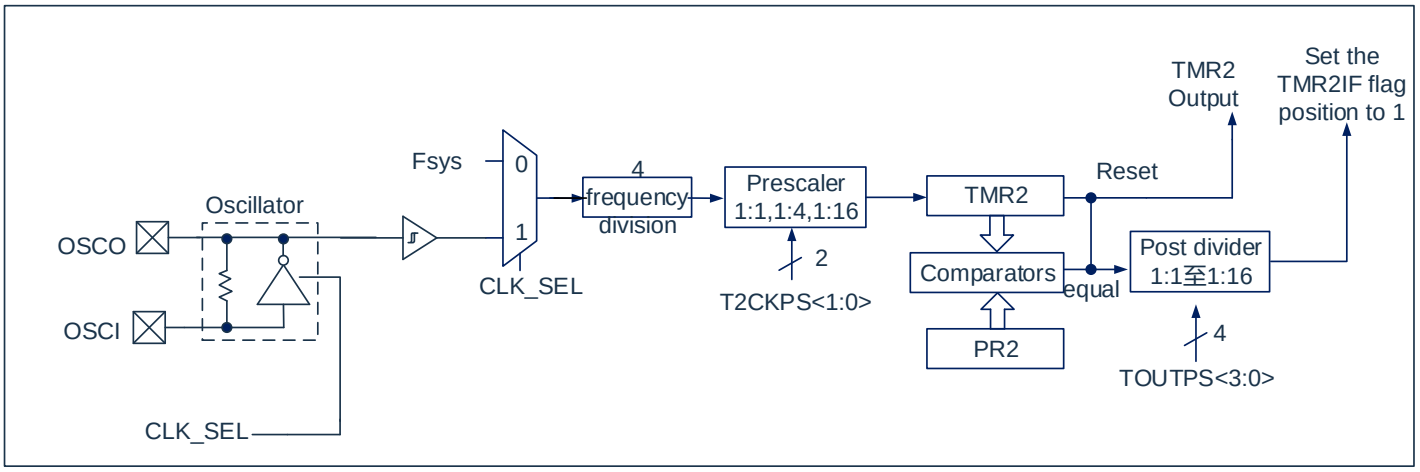


Fig 9-1: TIMER2 structure

9.2 Working Principle of TIMER2

The input clock of the TIMER2 mod is the system instruction clock (FSYS) or the external oscillator (32.768kHz). 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 TMR2 register.

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 00h.

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:

- TMR2 write operation
- T2CON write operation
- Any device reset occurs (power-on reset, watchdog timer reset, or under voltage reset).

Note:
1) Writing TMR2ON to 0 will clear TMR2.
2) Because of the reset value of PR2 is 00H, PR2 Should be setting done before using the timer2 to avoid a false matching is triggered.

9.3 TIMER2 related register

There are 3 registers related to TIMER2, namely data memory TMR2、PR2 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	CLK_SEL	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
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 CLK_SEL: Choice of clock source
1= External 32.768kHz /4 as TMR2 clock source (can continue to count in sleep state);
0= Internal Fsys/4 as TMR2 clock source

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.

Note: the theoretical vibration starting time is 32ms when the LP oscillation is enabled, which may larger actually.

10. Analog to Digital Conversion (ADC)

10.1 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 provided by internal LDO or VDD. ADC can generate an interrupt after conversion is completed.

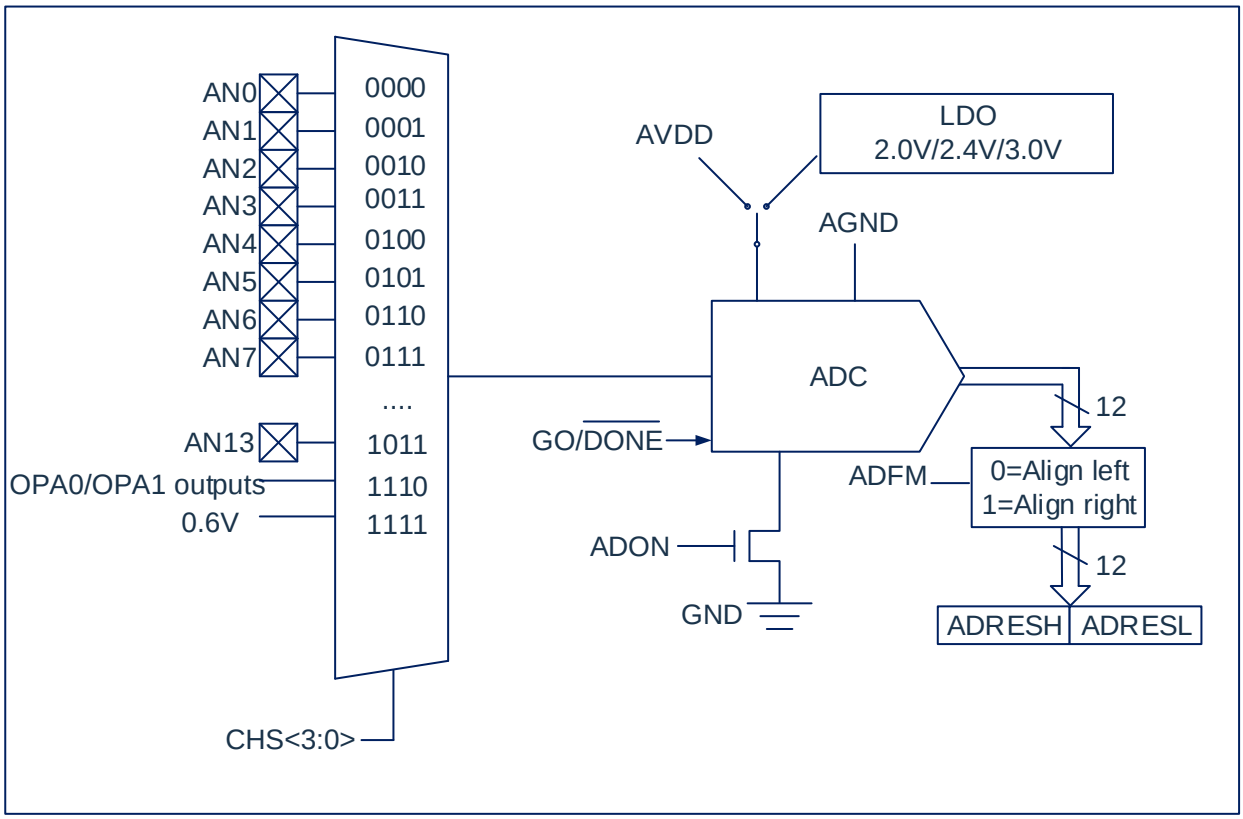


Fig 10-1: ADC structure

10.2 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.

10.2.1 Port configuration

ADC can convert both analog and digital signals. When converting analog signals, I/O pins (except AN12/AN13) should be configured as analog input pins by enabling ADC and corresponding analog channels by setting corresponding TRIS bit to 1. The I/O pin configured as an analog input always reads 0.

10.2.2 Channel selection

The CHS bit of the ADCON0 and ADCON1 registers 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.

Note:The CHS [3:0] needed to configurate as well as the ADCON0.ADON set to 1 to open the ADC channel;

10.2.3 ADC internal base voltage

Built-in base voltage of 0.6V, the CHS [3:0] should be configured as 1111 when this voltage needs to be detected.

10.2.4 ADC reference voltage

The ADC reference voltage can be provided by internal LDO or the chip's VDD and GND. 2.0V/2.4V/3.0V can be selected as the reference voltage. Needing to select a lower convert clock if internal reference voltage is selected, see the “Converter clock”.

Note: The effective accuracy of the ADC will decrease if reference voltage is provided by internal LDO. The lower detected voltage, the higher accuracy of the ADC will be obtained. It is recommended that the input voltage set to lower than 1V.

10.2.5 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.

Relationship between period of ADC clock (TAD) and the operating frequency of device (VDD=5.0V).

Period of ADC clock		The operating frequency of device		
ADC clock source	ADCS<1:0>	16MHz	8MHz	4MHz
$F_{SYS}/8$	00	24.5μs	49.0μs	98.0μs
$F_{SYS}/16$	01	49.0μs	98.0μs	196.0μs
$F_{SYS}/32$	10	98.0μs	196.0μs	392.0μs
F_{RC}	11	1-3ms	1-3ms	1-3ms

10.2.6 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.

10.2.7 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.

10.3 ADC working principle

10.3.1 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/D mod.

10.3.2 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.

10.3.3 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.

10.3.4 Working principle of ADC in sleep mode

Note: ADC mod cannot wakeup the chip in sleep mode.

10.3.5 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 reference voltage (AD conversion must be wait for up to 100us if the reference voltage switches from VDD to internal LDO);
   - 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).
9. At least wait for 2 TAD times to start AD conversion again When GO/ ("$\overline{\text{DONE}}$") bit change from 1 to 0 or ADIF change from 0 to 1.

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 ADC to 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
```

10.4 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 (9DH)

9DH	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
0000= AN0
0001= AN1
0010= AN2
0011= AN3
…… ……
1101= AN13
1110= OPA0/OPA1 outputs
1111= 0.6V (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 control register ADCON1 (9CH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	---	---	---	---	LDO_EN	LDO_SEL1	LDO_SEL0
Read/write	R/W	---	---	---	---	R/W	R/W	R/W
Reset value	0	---	---	---	---	0	0	0

Bit7 ADFM: A/D conversion result format selection bit

1= Right alignment

0= left alignment

Bit6 Reserved Write to 0

Bit5~Bit3 Not used

Bit2 LDO_EN: Internal reference voltage enable bit

1= Enable internal LDO as ADC reference voltage
The maximum effective accuracy of ADC is 8 bits when the internal ldo is selected as reference voltage

0= VDD as ADC reference voltage

Bit1~Bit0 LDO_SEL<1:0>: Reference voltage selection

00= Forbidden (Note: it is forbidden to select 00 if internal LDO is selected as reference voltage)

01= 2.0V

10= 2.4V

11= 3.0V

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

9FH	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 (9EH)，ADFM=0

9EH	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 register high bit ADRESH (9FH)，ADFM=1

9FH	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 bit ADRESL (9EH)，ADFM=1

9EH	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.

11. PWM Mod

11.1 PWM General

A programmable PWM mod with 10-bit width in chip, which can be configured as 5-channels output PWM0~4 with common cycle and independent duty cycle; Among them, PWM0/PWM1 and PWM2/PWM3 can be configured as complementary outputs.

11.2 PWM Related Register

PWM control register PWMCON0 (13H)

13H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON0	CLKDIV[2:0]			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
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

Bit7~5 CLKDIV[2:0]: PWM clock division.
111= F_{HSI}/128
110= F_{HSI}/64
101= F_{HSI}/32
100= F_{HSI}/16
011= F_{HSI}/8
010= F_{HSI}/4
001= F_{HSI}/2
000= F_{HSI}/1

Bit4~Bit0 PWMxEN: PWMx enable bit.
1= Enable PWMx
0= Disable PWMx

PWM control register PWMCON1 (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON1	PWMIO_SEL[1:0]		PWM2DTEN	PWM0DTEN	---	---	DT_DIV[1:0]	
R/W	R/W	R/W	R/W	R/W	---	---	R/W	R/W
Reset value	0	0	0	0	---	---	0	0

Bit7~6 PWMIO_SEL[1:0]: PWM IO selection

11= PWM assigned to group A，PWM0-RA0,PWM1-RA1,PWM2-RA2,PWM3-RA3,PWM4-RA4

10= PWM assigned to group B，PWM0-RA0,PWM1-RA1,PWM2-RA2,PWM3-RB2,PWM4-RB1

01= PWM assigned to group C，PWM0-RA5,PWM1-RB7,PWM2-RB6,PWM3-RB5,PWM4-RB4

00= PWM assigned to group D，PWM0-RB0,PWM1-RB1,PWM2-RB3,PWM3-RB4,PWM4-RB2

Bit5 PWM2DTEN: PWM2 dead-time enable bit

1= Enable PWM2 dead-time function, PWM2 and PWM3 compose one pair of complementary outputs.

0= Disable PWM2 dead-time function.

Bit4 PWM0DTEN: PWM0 dead-time enable bit

1= Enable PWM0 dead-time function, PWM0 and PWM1 compose one pair of complementary outputs.

0= Disable PWM0 dead-time function.

Bit3~Bit2 Not used.

Bit1~Bit0 DT_DIV[1:0]: Dead-time source clock division

11= $F_{HSI}/8$

10= $F_{HSI}/4$

01= $F_{HSI}/2$

00= $F_{HSI}/1$

PWM control register PWMCON2 (1DH)

1DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON2	---	---	---	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR
R/W	---	---	---	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	---	0	0	0	0	0

Bit7~Bit5 Not used:

Bit4~Bit0 PWMxDIR: PWMx output inversion control bit

1= PWMx output inversion

0= PWMx output ordinary

PWM0~PWM4 lower bit of period register PWMTL (15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTL	PWMT[7:0]							
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~Bit0 PWMT[7:0]: Lower 8 bits of PWM0~PWM4 period register

PWM higher bit of period register PWMTH (16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTH	---	---	PWMD4[9:8]		---	---	PWMT[9:8]	
R/W	---	---	R/W	R/W	---	---	R/W	R/W
Reset value	---	---	0	0	---	---	0	0

Bit7~Bit6 Not used:
Bit5~Bit4 PWMD4[9:8]: Higher 2 bits of PWM4 duty register
Bit3~Bit2 Not used:
Bit1~Bit0 PWMT[9:8]: Higher 2 bits of PWM0~PWM4 period register

PWM0 lower bit of duty register PWMD0L (17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD0L	PWMD0[7:0]							
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~Bit0 PWMD0[7:0]: PWM0 lower bit of duty register.

PWM1 lower bit of duty register PWMD1L (18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD1L	PWMD1[7:0]							
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~Bit0 PWMD1[7:0]: PWM1 lower bit of duty register.

PWM2 lower bit of duty register PWMD2L (19H)。

19H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD2L	PWMD2[7:0]							
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~Bit0 PWMD2[7:0]: PWM2 lower bit of duty register.

PWM3 lower bit of duty register PWMD3L (1AH)

1AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD3L	PWMD3[7:0]							
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~Bit0 PWMD3[7:0]: PWM3 lower bit of duty register.

PWM4 lower bit of duty register PWMD4L (1BH)

1BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD4L	PWMD4[7:0]							
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~Bit0 PWMD4[7:0]: PWM4 lower bit of duty register.

PWM0/PWM1 higher bit of duty register PWMD01H (1CH)

1CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD01H	---	---	PWMD1[9:8]		---	---	PWMD0[9:8]	
R/W	---	---	R/W	---	---	----	R/W	R/W
Reset value	---	---	0	---	---	----	0	0

Bit7~Bit6 Not used.
Bit5~Bit4 PWMD1[9:8]: PWM1 higher 2 bits of duty register.
Bit3~Bit2 Not used.
Bit1~Bit0 PWMD0[9:8]: PWM0 higher 2 bits of duty register.

PWM2/PWM3 higher bit of duty register PWMD23H (0EH)

0EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD23H	---	---	PWMD3[9:8]		---	---	PWMD2[9:8]	
R/W	---	---	R/W	---	---	----	R/W	R/W
Reset value	---	---	0	---	---	----	0	0

Bit7~Bit6 Not used.
Bit5~Bit4 PWMD3[9:8]: PWM3 higher 2 bits of duty register.
Bit3~Bit2 Not used.
Bit1~Bit0 PWMD2[9:8]: PWM2 higher 2 bits of duty register.

PWM0/PWM1 dead-time register PWM01DT (0FH)

0FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM01DT	---	---	PWM01DT[5:0]					
R/W	---	---	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~Bit0 PWM01DT[5:0]: PWM0/PWM1 dead-time register.

PWM2/PWM3 dead-time register PWM23DT (10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM23DT	---	---	PWM23DT[5:0]					
R/W	---	---	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~Bit0 PWM23DT[5:0]: PWM2/PWM3 dead-time register.

11.3 PWM Period

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

Formula 1：PWM period：

$$\text{PWM period}=[(PWMT)+1]*T_{HSI}*(CLKDIV\ \ prescaler\ value)$$

Note: $T_{HSI}=1/F_{HSI}$

When PWM period counter is equal to PWMT, the following five events will occur in the next up-counting period:

- PWM period counter is cleared;
- PWMx pin is set to 1;
- New period of PWM is latched;
- New duty of PWMx is latched;
- Generating the PWM interrupt flag bit.

11.4 PWM Duty Cycle

The PWM duty cycle can be specified by writing a 10-bit value to the following multiple registers: the PWMDxL register and PWMDxxH register.

You can write the PWMDxL and PWMDxxH register at any time, but until the values in PWM period counter and PWMT match (that is, the period ends), the value of the duty cycle It is latched into internal latch.

Formula 2: Pulse width calculation formula:

$$\text{pulse width}=(PWMDx[9:0])*T_{HSI}*(CLKDIV\ prescaler\ value)$$

Formula 3: PWM duty cycle calculation formula:

$$\text{duty cycle}=\frac{PWMDx[9:0]+1}{PWMT[9:0]+1}$$

Internal latches are used to provide double buffering for the PWM duty cycle and period. This double buffering structure is extremely important to avoid glitches during the PWM operation.

11.5 System Clock Frequency Changes

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

11.6 Programmable dead-time delay mode

Complementary output mode can be enabled by configured PWMxDT_EN, and the dead-time delay function is enabled automatically after enable complementary output mode.

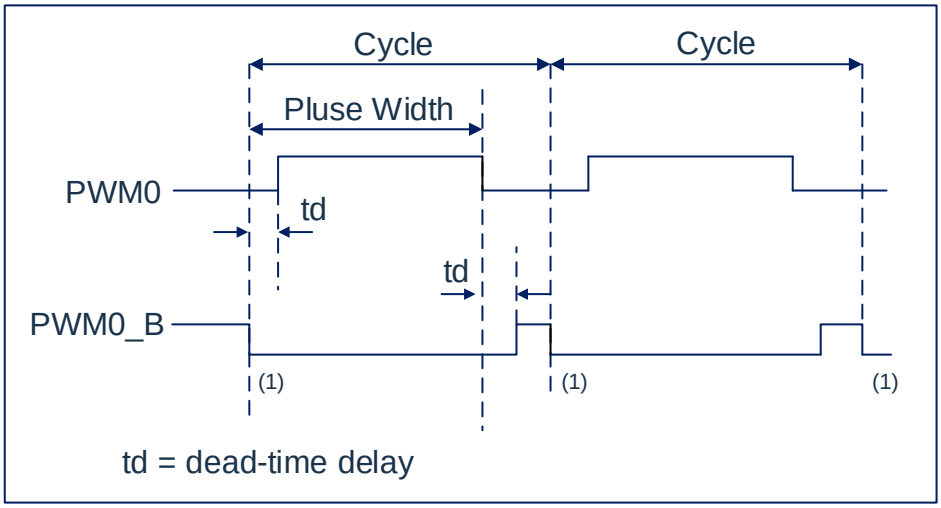


Fig 11-1: Sample of PWM dead-time delay output

Dead-time calculation formula:

$$td=(PWMxxDT[5:0]+1)*T_{HSI}*(DT\_DIV\ prescaler\ value)$$

11.7 Configurate PWM

The following steps should be performed when configuring PWM mod:

1. Select the output pin group of PWM by setting the IO_SEL.
2. Disable the output driver of PWMpin by setting the corresponding TRIS bit to 1 to make it an input pin.
3. Set the PWM period by loading the PWMTH and PWMTL register.
4. Set the PWM duty cycle by loading the PWMDxL register and PWMDxxH register.
5. Set the PWM dead-time by setting the PWMCON1[5:4] and loading PWMxxDT register if complementary output mode is required.
6. Clear the PWMIF flag bit.
7. Enable corresponding output by setting the PWMCN0[4:0].
8. After the new PWM period starts, enable PWM output:
   - Wait for PWMIF set to 1.
   - Enable the PWM pin output driver by clearing the corresponding TRIS bit.

12. Touch Button

12.1 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:

- 1-7 buttons are optional
- Adjustable sensitivity by change external touch capacitance
- Touch button effective response time lower than 100ms

16 bit high-precision CDC (digital capacitance converter) and the capacitance change on the IC sensing disk (capacitance sensor) are used to recognize the touch action of person's fingers.

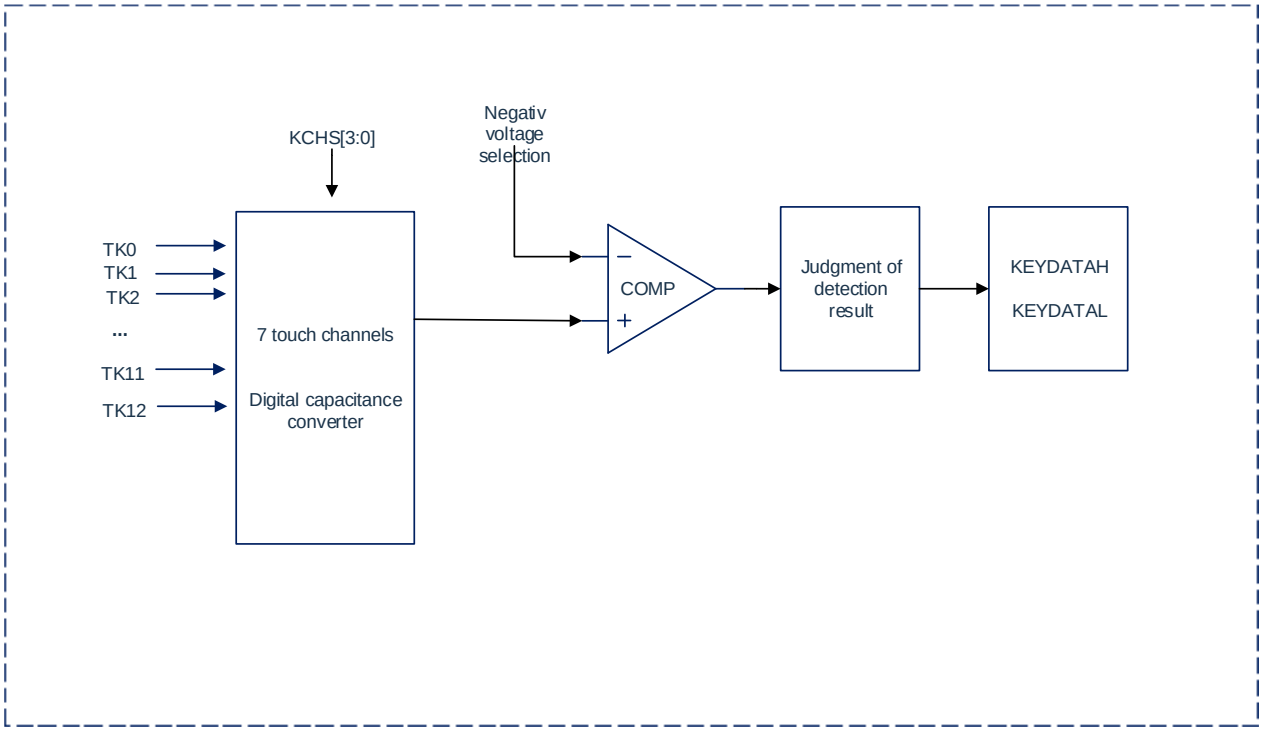


Fig 12-1: Diagram of internal circuit

12.2 Touch Button Related Register

There are mainly 4 RAMs related touch button, namely control register KEYCON0、KEYCON1 and KEYCON2, result register KEYDATL and KEYDATH.

Touch button result register lower bit KEYDATL(94H)

94H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYDATL								
R/W	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Touch button result register higher bit KEYDATH(95H)

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYDATH								
R/W	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

The KEYDATH and the KEYDATL are result registers of touch button, which are read-only. After the touch detection is completed, the detection result can be read from KEYDATH and KEYDATL.

Touch button control register KEYCON0(92H)

92H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYCON0	KDONE	----	CAPK[2:0]			KTOUT	KCAP	KEN
R/W	R	----	R/W	R/W	R/W	R	R/W	R/W
Reset value	0	----	0	0	0	0	0	0

Bit7 KDONE: Touch button detection complete flag bit;
0= Detection not complete;
1= Detection complete.

Bit6 Reserved.

Bit5~Bit3 CAPK[2:0]: Selection of internal shunt capacitor of key port（C≈0.4pF）;
000= No shunt capacitor;
001= Internal shunt capacitor of key port is C×1;
010= Internal shunt capacitor of key port is C×2;
011= Internal shunt capacitor of key port is C×3;
100= Internal shunt capacitor of key port is C×4;
101= Internal shunt capacitor of key port is C×5;
110= Internal shunt capacitor of key port is C×6;
111= Internal shunt capacitor of key port is C×7;

Bit2 KTOUT: Key result counter overflow flag bit;
0= No overflow;
1= Overflow.

Bit1 KCAP: Touch button capacitance enable bit (RB6 pin）;
0= Disable;
1= Enable.

Bit0 KEN: Touch button detection start bit;
0= Stop touch button detection, when this bit is 0, the KEYDATH and KEYDATL are cleared automatically.
1= Start touch button detection, it must be kept at 1 during detection.

Touch button control register KEYCON1(93H)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYCON1	KVREF[1:0]		KCLK[1:0]		KCHS[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~Bit6 KVREF: Selection of negative voltage of touch button internal comparator;
00= 0.4*V_{LDO};
01= 0.5* V_{LDO};
10= 0.6 *V_{LDO};
11= 0.7 *V_{LDO}。

Bit5~Bit4 KCLK: Selection of touch button clock F_{TKDIV} division;
00= $F_{TKDIV}=F_{HSI}$;
01= $F_{TKDIV}=F_{HSI}/2$;
10= $F_{TKDIV}=F_{HSI}/4$;
11= $F_{TKDIV}=F_{HSI}/8$。

Bit3~Bit0 KCHS[3:0]: Channel selection for detection;
0000: Select TK0 channel;
0001: Select TK1 channel;;
0010: Select TK2 channel;
0011: Select TK3 channel;
0100: Select TK4 channel;
0101~1010: Reserved;
1011: Select TK11 channel;
1100: Select TK12 channel;
1101~1111: Reserved;

Touch button control register KEYCON2(97H)

97H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYCON2	CAP_LVBO[2:0]			----	----	----	----	TKEN
R/W	R/W	R/W	R/W	----	----	----	----	R/W
Reset value	0	0	0	----	----	----	----	0

Bit7~Bit5 CAP_LVBO[2:0] Digital filtering time selection of end flag bit ($T_{TKDIV}=1/F_{TKDIV}$)。
000= filtering time 1*T_{TKDIV}
001= filtering time 2* T_{TKDIV}
010= filtering time 3* T_{TKDIV}
011= filtering time 4* T_{TKDIV}
100= filtering time 5* T_{TKDIV}
101= filtering time 6* T_{TKDIV}
110= filtering time 7* T_{TKDIV}
111= filtering time 8* T_{TKDIV}

Bit4~Bit1 Reserved write to 0.

Bit0 TKEN Touch button mod total enable bit;
0: Disable touch button mod.
1: Enable touch button mod.

12.3 Application for Touch Button Mod

12.3.1 The process of reading “data of touch button” in query mode

1. Enable output driver of the corresponding IO (including key port and the sensitivity adjustment capacitor port).
2. Set the TKEN bit in KEYCON2 as 1;
3. Set the touch button control register KEYCON1 (including channel selection, touch button detected clock configuration, negative voltage of comparator configuration;
4. Set the KEYCON2 (digital filter selection);
5. Set the touch button control register KEYCON0 (enable the capacitor and select whether the Internal shunt capacitor of key port is needed);
6. Set the 0th bit KEN of KEYCON0 from 0 to 1, start the touch button detection;
7. Judge whether the 7th bit KDONE of KEYCON0 is 1;
8. Read the 16bit data;
9. Finish touch button detection: KEN=0;
10. Return to step 3 to continue the detection of next channel.

Example: detection of touch button (KEY0) in query mode

```
KEY_START：
        LDIA    00H
        LD      INTCON，A       ; disable interrupt
        LDIA    B’00000001’
        LD      TRISA，A        ; RA0 set as touch button detected port
        LDIA    B’01000000’     ; RB6 set as sensitivity capacitor port
        LD      TRISB，A
        SETB    KEYCON2，0      ; enable the touch button mod
        LDIA    B’01010000’
        LD      KEYCON1，A      ; set the negative voltage of comparator, and the
                                  detected clock and channel of touch button
        LDIA    02H
        LD      KEYCON0，A      ; set the digital filtering time
        SETB    KEYCON0，0      ; start to detect
WAIT：
        SNZB    KEYCON0，7      ; wait for detection finish
        JP      WAIT
        LD      A，KEYDATH
        LD      R01，A          ; Save the higher 8-bit result to the user-defined ram
        LD      A，KEYDATL
        LD      R02，A          ; Save the lower 8-bit result to the user-defined ram
        CLRB    KEYCON0，0      ; end of detection
        JP      XXXX            ; go to other program
```

12.3.2 Judge method of key press

- Base of judgment: no press down---large “data”; press down---small “data”;
- The current value is smaller than previous value in a certain extent, which can be considered as the key have been “pressed down”;
- Within a certain period of time, “data” change from large to small is considered that a key is pressed.

Example: judgment of key press

```
K_START:
        LD      A，KOLDH        ;start to judge, higher bit first
        SUBA    KDATAH          ;subtract the previous key value from the current key value
        SZB     STATUS，Z
        JP      K_H_SAME        ;if the higher bit is equal, judge the lower bit
        SZB     STATUS，C
        JP      KNO             ;if the higher bit of current value larger than the previous one, there is no key press
        SUBIA   01H
        SNZB    STATUS，C
        JP      KHAVE           ;if the higher bit of current value smaller than the previous one, and the difference is greater or equal to 2, there is a key press
        LD      A，KDATAL
        SUBA    KOLDL
        SZB     STATUS，C
        JP      KHAVE           ;if the higher bit of current value is 1 less than the previous value, and the lower bit is also smaller, then there is a key press
        SUBIA   0AH
        SZB     STATUS，C
        JP      KHAVE           ;if the whole value is more than 10 smaller than previous value, there is a key press
        JP      KNO             ;if the whole value is not more than 10 smaller than previous value, there is no key press
K_H_SAME：
        LD      A，KDATAL
        SUBA    KOLDL
        SNZB    STATUS，C
        JP      K_NO            ;if the higher bit is equal and the lower bit is greater than the previous value, there is no key press
        SUBIA   0AH
        SZB     STATUS，C
        JP      KNO             ;if the current value is more than 10 smaller than the previous value, there is a key press
KHAVE：                         ;program of key press
        ...
        JP      XXXX            ;deal with the corresponding program and go to other program
KNO:                            ;program of no key press
        ...
        JP      XXXX            ; deal with the corresponding program and go to other program
```

The previous value store in the KOLDH and KOLDL, and the current value store in the KDATAH and KDATAL. It is considered that the key is pressed when the current value more than 10 lower than the previous value in this example, but this threshold should be set according to the specific situation in practical application.

12.4 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.

13. Program EEPROM and Program Memory Control

13.1 General

The devices in this series have 4K words of program memory, the address range is from 0000h to 0FFFh, which is read-only in all address ranges; the device has a 32-byte program EEPROM, and the address range is 0000h to 001Fh, 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 12-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.

13.2 Related Register

13.2.1 EEADR and EEADRH Register

The EEADR and EEADRH registers can address up to 32 bytes of program EEPROM or up to 4K 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.

13.2.2 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 (8EH)

8EH	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 (90H)

90H	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 (8FH)

8FH	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 (96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADRH	---	---	---	---	EEADRH3	EEADRH2	EEADRH1	EEADRH0
read/write	---	---	---	---	R/W	R/W	R/W	R/W
reset value	---	---	---	---	0	0	0	0

Bit7~Bit4 not used，read 0.

Bit3~Bit0 EEADRH<3:0>: Specify the upper 5 address of the program memory read operation.

EEPROM control register EECON1 (8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD	---	---	---	WRERR	WREN	WR	RD
read/write	R/W	---	---	---	R/W	R/W	R/W	R/W
Reset value	0	---	---	---	X	0	0	0

Bit7 EEPGD: Program/program EEPROM selection bit；
1= Operate program memory；
0= Operate program EEPROM.

Bit6~Bit4 not used

Bit3 WRERR: EEPROM error flag bit；
1= Write error (any WDT reset or under voltage 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.

13.3 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, 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

```
LD      A,EE_ADD        ; Put the address to be read into the eeadr register
LD      EEADR,A
CLRB    EECON1,EEPGD    ; Select program EEPROM
SETB    EECON1,RD       ; Enable read signal
NOP                     ; To read data here, you must add NOP instruction
NOP
LD      A,EEDAT         ; Read data to acc
LD      EE_DATL,A
LD      A,EEDATH
LD      EE_DATH,A
```

13.4 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

```
        LD      A,EE_ADDL       ; Put the address to be written into EEADR register
        LD      EEADR,A
        LD      A,EE_DATAL      ; Put the low 8-bit data to be written into the EEDAT register
        LD      EEDAT,A
        LD      A,EE_DATAH      ; Put the upper 8 bits of data to be written into the EEDATH register
        LD      EEDATH,A
        CLRWDT
        CLRB    EECON1,EEPGD
        SETB    EECON1,WREN     ; Allow write operations
        CLRB    INTCON,GIE      ; Close interrupt
        SZB     INTCON,GIE      ; Make sure the interrupt is turned off
        JP      $-2
        LDIA    055H            ; Write 55h to EECON2
        LD      EECON2,A
        LDIA    0AAH            ;Write EECON2 to 0AAH
        LD      EECON2,A
        SETB    EECON1,WR       ; Enable write signal
        NOP
        NOP
        CLRWDT
        CLRB    EECON1,WREN     ; Write end, turn off the write enable bit
        SETB    INTCON,GIE
```

13.5 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 instructions period 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,EE_ADDL       ; Put the address to be read into the eeadr register
        LD      EEADR,A
        LD      A,EE_ADDH       ; Put the high bit of the address to be read into the
                                  eeadrh register
        LD      EEADRH,A
        SETB    EECON1,EEPGD    ; Select operating program memory
        SETB    EECON1,RD       ; Allow read operation
        NOP
        NOP
        LD      A,EEDAT         ; Save read data
        LD      EE_DATL,A
        LD      A,EEDATH
        LD      EE_DATH,A
```

13.6 Write Program Memory

program memory is read only, cannot be written.

13.7 Precautions on Program EEPROM

13.7.1 Programming Time for Program EEPROM

The program EEPROM programming time is fixed about 4.6ms. During the programming, the CPU stops working, the peripherals mod works normally, and the program needs to be well dealt with accordingly.

13.7.2 Write Verification

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

13.7.3 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:

- Under voltage
- Power glitch
- Software failure

14. Operational amplifier (OPA0 and OPA1)

The chip has built-in two groups of operational amplifiers OPA0 and OPA1. The functions and performances of the two groups of operational amplifiers are the same. The x value is 0, 1 in the following description.

14.1 OPAx General

OPAx has the following characteristics:

- Internal integrated zero adjustment circuit;
- The positive and negative terminals can be connected to the I/O port;
- The output terminal can be connected to the I/O port or the internal ADC detection channel;

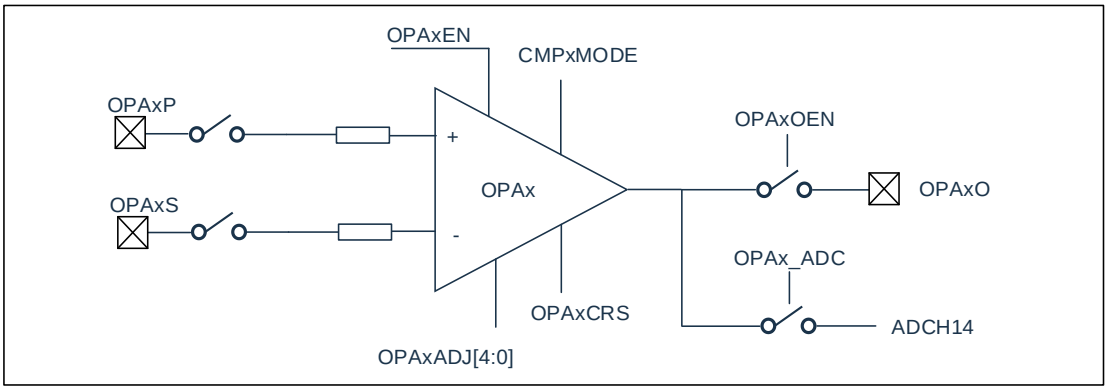


Fig 14-1: OPAx block diagram

14.1.1 OPAx Enable

Set the 7 th bit OPAxEN of the register OPAxCON to 1, enable the operational amplifier. Set OPAxEN to 0 to disable the operational amplifier. After the operational amplifier is enabled, the positive and negative terminals are automatically connected to the I/O port.

14.1.2 OPAx Port Selection

1. After the operational amplifier is enabled, the positive and negative terminals are automatically connected to the I/O port.
2. The output of the op amp can be output from the OPAxO pin, which is achieved by setting the 6 th bit OPAxOEN of OPAxCON; the op amp output can be connected to the ADC14 channel by setting the 4 th bit of OPAxCON.

Note: At the same time, only OPAO can be connected to ADC14 channel, and two OPAOs cannot be connected to ADC14 channel at the same time.

The related I/O ports used by OPAx must be set to input state, including op amp input and op amp output (when it needs to be connected to IO).

14.1.3 OPAx Working Mode

The chip's built-in two groups of operational amplifiers have 2 working modes, normal mode and adjustment mode.

The 6 th bit OPAxCOFM of register OPAxADJ is set to 0, the operational amplifier enters the normal working mode.

The 6 th bit OPAxCOFM of register OPAxADJ is set to 1, the operational amplifier enters the adjustment mode. In the adjustment mode, the positive and negative terminals of the op amp are internally short-circuited and connected to the positive or negative side of the op amp (selected by OPAxCRS in the 5 th bit OPAxADJ). The function of the adjustment mode is to minimize the offset voltage of the op amp.

Adjustment mode workflow:

1. Enable the op amp function;
2. Set the op amp to enter the adjustment mode;
3. Set the op amp adjustment mode from the positive input or the negative input, the input cannot be left floating.
4. Set the adjustment bits OPAxADJ<4:0> to the initial value, the maximum (1FH) or the minimum (00H);
5. Delay for a period of time, the time is related to the external capacitor parameters;
6. Read op output;
7. Decrease the adjustment bit by 1(the initial value is set to the maximum 1FH) or increase by 1(the initial value is set to 00H);
8. Delay;
9. Read the output of the op amp, if there is any change, if there is no change, continue to step 7;
10. The read value changes, the zero adjustment ends, OPAxCOFM is cleared, and the normal operating mode is entered.

14.2 OPAx Related registers

There are 5 registers related to OPA mode, namely OPA0CON, OPA1CON, OPA0ADJ, OPA1ADJ, CMPCON.

OPA0 control register OPA0CON(98H)

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPA0CON	OPA0EN	OPA0OEN	CMP0MODE	OPA0_ADC	----	----	----	OPA0FT
R/W	R/W	R/W	R/W	R/W	----	----	----	R/W
Rest value	0	0	0	0	----	----	----	0

Bit7 OPA0EN: OPA0 enable bit;
1= enable
0= disable
Bit6 OPA0OEN: OPA0 output enable
1= OPA0 output is connected to the I/O port (OPA0O pin)
0= OPA0 output is not connected to the I/O port
Bit5 CMP0MODE: CMP0MODE selection bit
1= Comparator mode
0= Op amp mode
Bit4 OPA0_ADC: ADC channel enable
1= OPA0 output is connected to ADC14 channel
0= OPA0 output is not connected to ADC14 channel
Bit3~Bit1 not used
Bit0 OPA0FT: Op amp output internal filter selection
1= OPA0 output is internally connected to the filter circuit
0= OPA0 output is internally not connected to the filter circuit

OPA0 control register OPA0ADJ(99H)

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPA0ADJ	OPA0DOUT	OPA0COFM	OPA0CRS	OPA0ADJ[4:0]-				
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Rest value	0	0	0	x	x	x	x	x

Bit7 OPA0DOUT: OPA0 output result
1= The op amp output is high, and the positive terminal is higher than the negative terminal voltage
0= The op amp output is low, and the positive terminal is lower than the negative terminal voltage
Bit6 OPA0COFM: OPA0 working mode selection
1= OPA0 working in adjustment mode
0= OPA0 working in normal mode
Bit5 OPA0CRS: OPA0 adjustment mode input selection bit
1= OPA0 adjustment mode positive input
0= OPA0 adjustment mode negative input
Bit4~Bit0 OPA0ADJ[4:0]: OPA0 offset voltage adjustment bit

OPA1 control register OPA1CON(9AH)

9AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPA1CON	OPA1EN	OPA1OEN	CMP1MODE	OPA1_ADC	----	----	----	OPA1FT
R/W	R/W	R/W	R/W	R/W	----	----	----	R/W
Rest value	0	0	0	0	----	----	----	0

Bit7 OPA1EN: OPA1 enable bit;
1= enable
0= disable
Bit6 OPA1OEN: OPA1 output enable
1= OPA1 output is connected to the I/O port (OPA1O pin)
0= OPA1 output is not connected to the I/O port
Bit5 CMP1MODE: Comparator mode selection
1= Comparator mode
0= Operational amplifier mode
Bit4 OPA1_ADC: ADC channel enable
1= OPA1 output is connected to ADC14 channel
0= OPA1 output is not connected to ADC14 channel
Bit3~Bit1 not used
Bit0 OPA1FT: Op amp output internal filter selection
1= OPA1 output is internally connected to the filter circuit
0= OPA1 output is internally not connected to the filter circuit

OPA1 control register OPA1ADJ(9BH)

9BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPA1ADJ	OPA1DOUT	OPA1COFM	OPA1CRS	OPA1ADJ[4:0]-				
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Rest value	0	0	0	x	x	x	x	x

Bit7 OPA1DOUT: OPA1 output result
1= The op amp output is high, and the positive terminal is higher than the negative terminal voltage
0= The op amp output is low, and the positive terminal is lower than the negative terminal voltage
Bit6 OPA1COFM: OPA1 working mode selection
1= OPA1 working in adjustment mode
0= OPA1 working in normal mode
Bit5 OPA1CRS: OPA1 adjustment mode input selection bit
1= OPA1 adjustment mode positive input
0= OPA1 adjustment mode negative input
Bit4~Bit0 OPA1ADJ[4:0]: OPA1 offset voltage adjustment bit

15. LVD Low Voltage detection

15.1 LVD Mod General

SC8F289xB series of MCU have 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.

15.2 LVD Related Register

There is 1 register related to LVD mod.

LVD control register LVDCON (11FH)

11FH	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；

15.3 LVD Operation

By setting the LVD voltage value V_{SET} in the lvdcon register, after enabling LVDEN, due to the internal filtering processing of the chip, in order to reduce the frequent fluctuation of LVD output results caused by the fluctuation of power supply voltage near V_{SET}, the power supply voltage needs to be lower than Vset for 1.5ms before LVD_RES is set to 1, while LVDIF is set to 1; If the corresponding interrupt enable bit is enabled, an LVD interrupt will be generated.

LVD cannot be used in interrupt wake-up mode.

16. Universal Synchronous/Asynchronous Transmitter (USART)

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 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 synchronous master control mode
- In synchronous mode, programmable clock polarity

Figure 16-1 and Figure 16-2 below are the block diagrams of the USART transmitter.

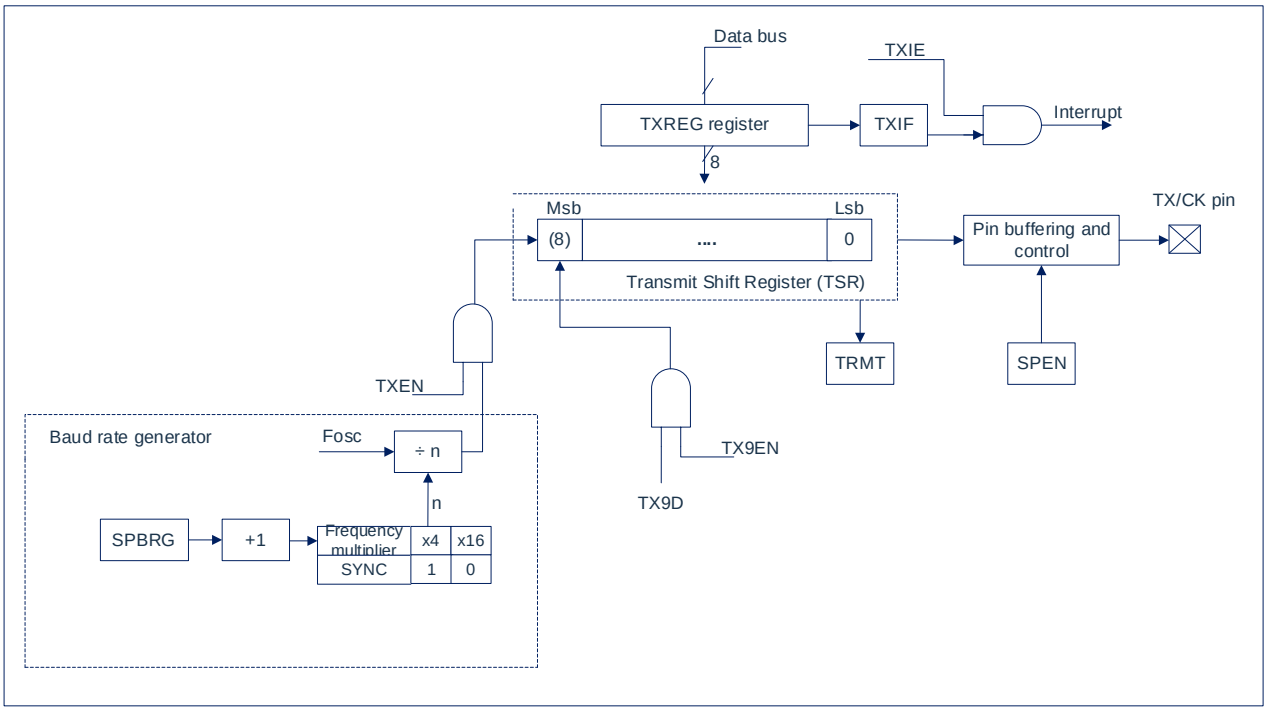


Fig 16-1: USART transmit block diagram

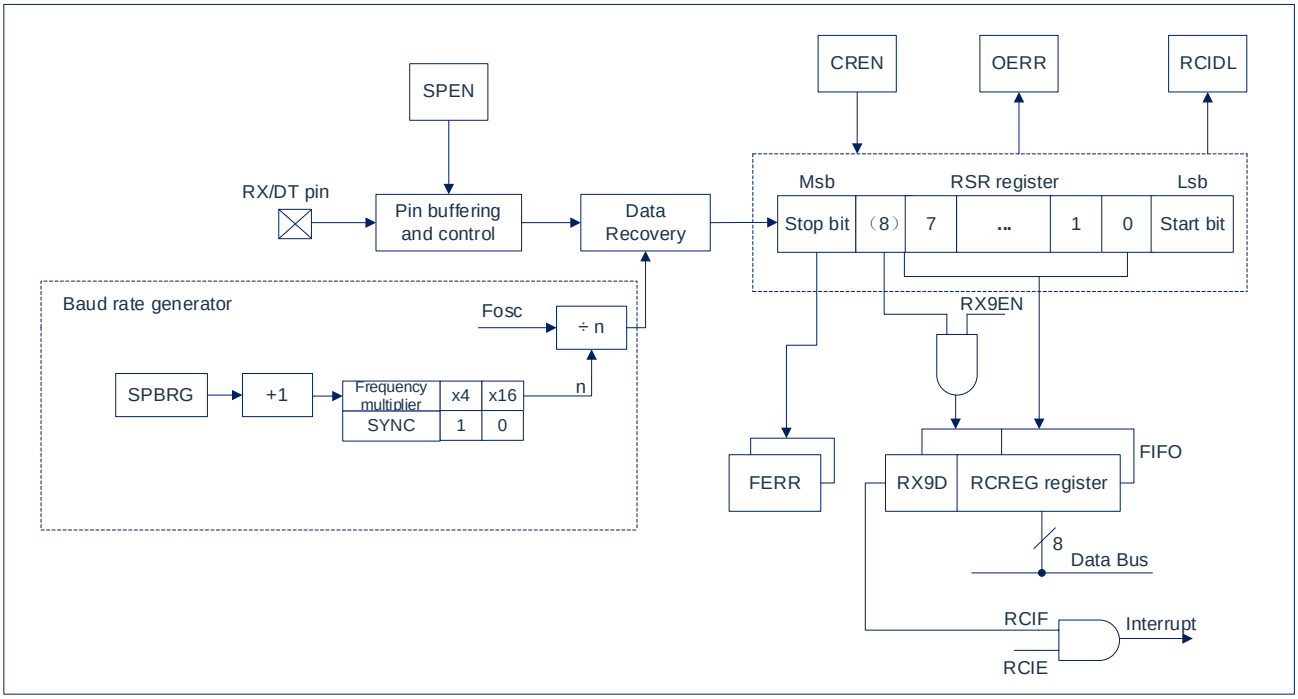


Fig 16-2: USART receive block diagram

The operation of the USART mod is controlled by 2 registers:

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

16.1 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).

16.1.1 USART Asynchronous Generator

Figure 16-1 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.

16.1.1.1 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/CK I/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 RX/DT I/O pin is automatically configured as an input pin, regardless of the state of the corresponding TRIS bit.

16.1.1.2 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.

16.1.1.3 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 TXIF flag 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.

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

16.1.1.4 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.

16.1.1.5 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.

16.1.1.6 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 TX9D data bit.
7. Load 8-bit data into TXREG register to start transmitting data.

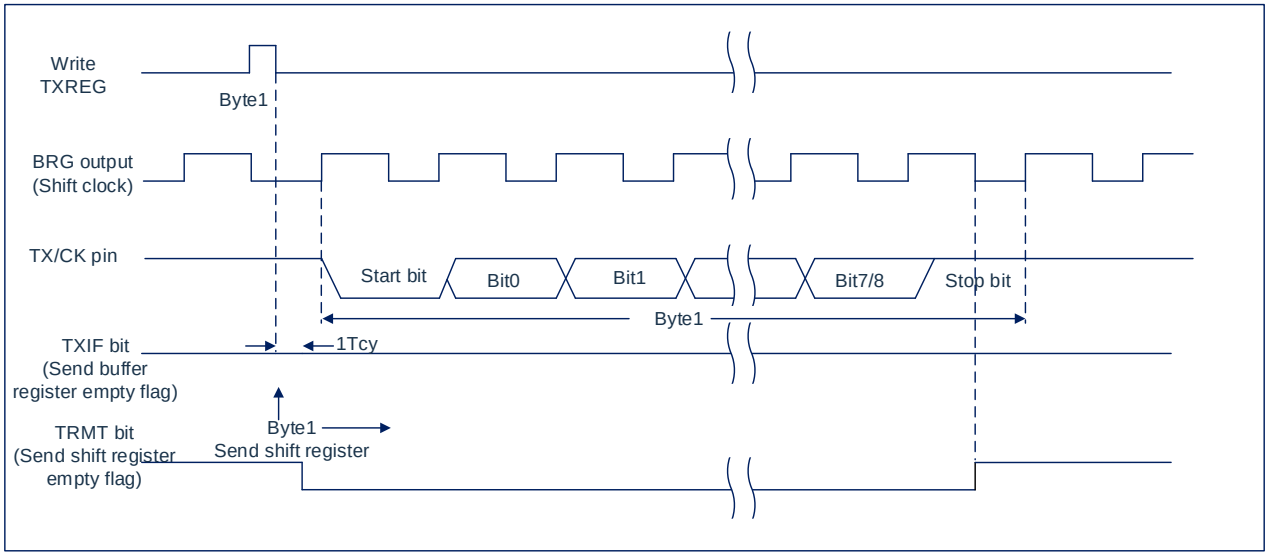


Fig 16-3: asynchronous transmit

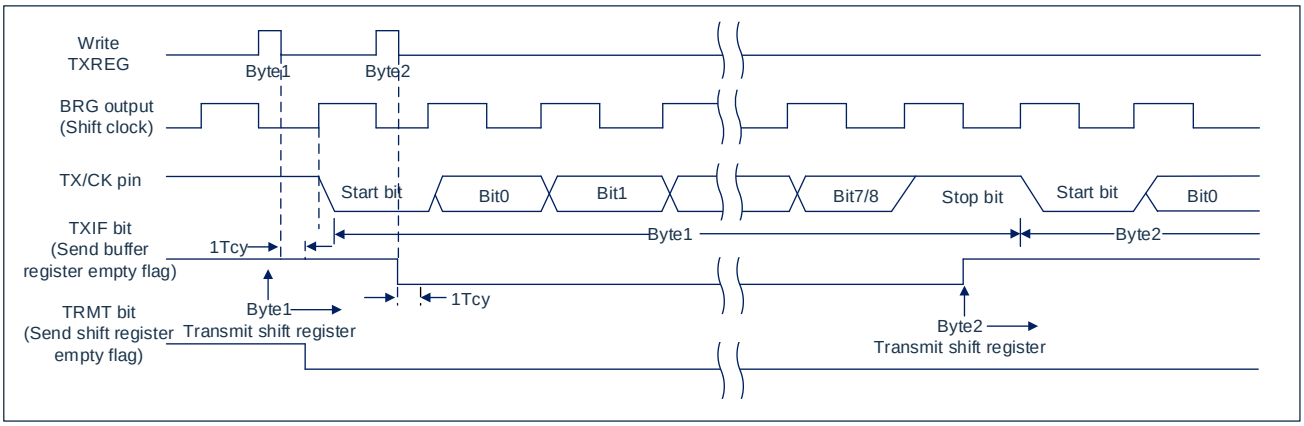


Fig16-4: asynchronous transmit (back to back)

Note: This time series diagram shows two consecutive transmit.

16.1.2 USART Asynchronous Receiver

Asynchronous mode is usually used in RS-232 system. Figure 16-2 shows the block diagram of the receiver. Receive data and driver data recovery circuit on RX/DT pin. The data recovery circuit is actually a 16 times baud rate as the operating frequency High-speed shifter, while the serial receive 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.

16.1.2.1 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.

16.1.2.2 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 RSR register. 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.

16.1.2.3 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.

16.1.2.4 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.

16.1.2.5 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.

16.1.2.6 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.

16.1.2.7 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 data receive is enabled).
9. If overflow occurs, clear the OERR flag by clearing the CREN receiver enable bit.

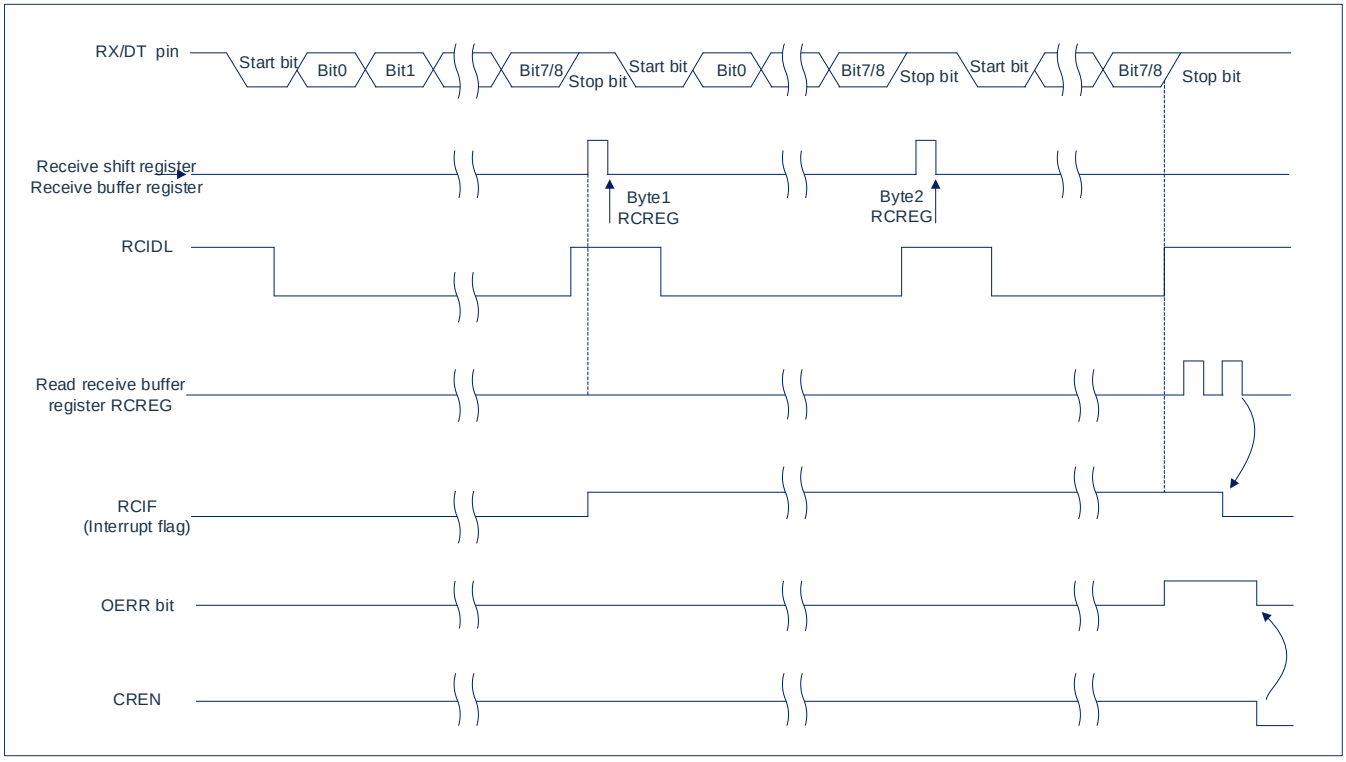


Fig 16-5: Asynchronous receive

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.

16.2 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.

16.3 USART Related Register

TXSTA：transmit status and control register (117H)

117H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXSTA	CSRC	TX9EN	TXEN(1)	SYNC	SCKP	STOPBIT	TRMT	TX9D
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset value	0	0	0	0	0	0	1	0

Bit7 CSRC: clock sources election 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 TX9EN: 9-bit transmit enable bit;
1= Select 9-bit transmit;
0= Select 8-bit transmit.

Bit5 TXEN: Transmit enable bit (1);
1= Enable transmit;
0= Disable transmit.

Bit4 SYNC: USART mode selection bit;
1= Synchronous mode;
0= Asynchronous mode.

Bit3 SCKP: 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 STOPBIT: Stop bit selection (only valid for asynchronous transmit), this bit needs to write to 0 when giving data for transmit by judge TRMT=1.
1= 1 stop bit;
0= 2 stop bits.

Bit1 TRMT: Transmit shift register status bit;
1= TSR empty;
0= TSR full.

Bit0 TX9D: 9th bit of Transmit data.
Can be address/data bit or parity check bit.

Note:
1) In synchronous mode, SREN/CREN will invert the value of TXEN.
2) STOPBIT needs to write to 0 when giving data for transmit by judge TRMT=1.

RCSTA：receive status and control register (118H)

118H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D
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 SPEN: Serial port enable bit;
1= Enable serial port (RX/DT and TX/CK pin configured as serial port in);
0= Disable serial port (hold on reset).

Bit6 RX9EN: 9-bit receive enable bit;
1= Select 9-bit receive;
0= Select 8-bit receive;.

Bit5 SREN: 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 CREN: 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 RCIDL: Receive idle flag bit.
Asynchronous mode: 1=receiver idle;
0= already receive initial bit，receiving data.
Synchronous mode: any value.

Bit2 FERR: 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 OERR: Overflow error bit.
1= Overflow error (clear by clearing CREN bit);
0= No overflow error.

Bit0 RX9D: 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.

16.4 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 16-1 contains the formula for calculating baud rate. Formula 1 is an example of calculating baud rate and baud rate error.

Table 16-1 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: calculate baud rate error

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

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

solve SPBRG:

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

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

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

Table 16-1: baud rate formula

Configuration bit	BRG/USART mode	baud rate formula
SYNC		
0	8bit/asynchronous	Fosc/[16 (n+1)]
1	8bit/synchronous	F_{OSC}/[4 (n+1)]

Note: n= value of SPBRG register.

Table 16-2: baud rate in asynchronous mode

Target baud rate	SYNC=0					
	F_{OSC}=8.00MHz			F_{OSC}=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

16.5 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 cannot receive or transmit at the same time. USART can be used as a master control device, or as a slave device.

16.5.1 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.

16.5.1.1 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.

16.5.1.2 Clock Polarity

The device provides clock polarity options to be compatible with Microware. 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.

16.5.1.3 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.

16.5.1.4 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.

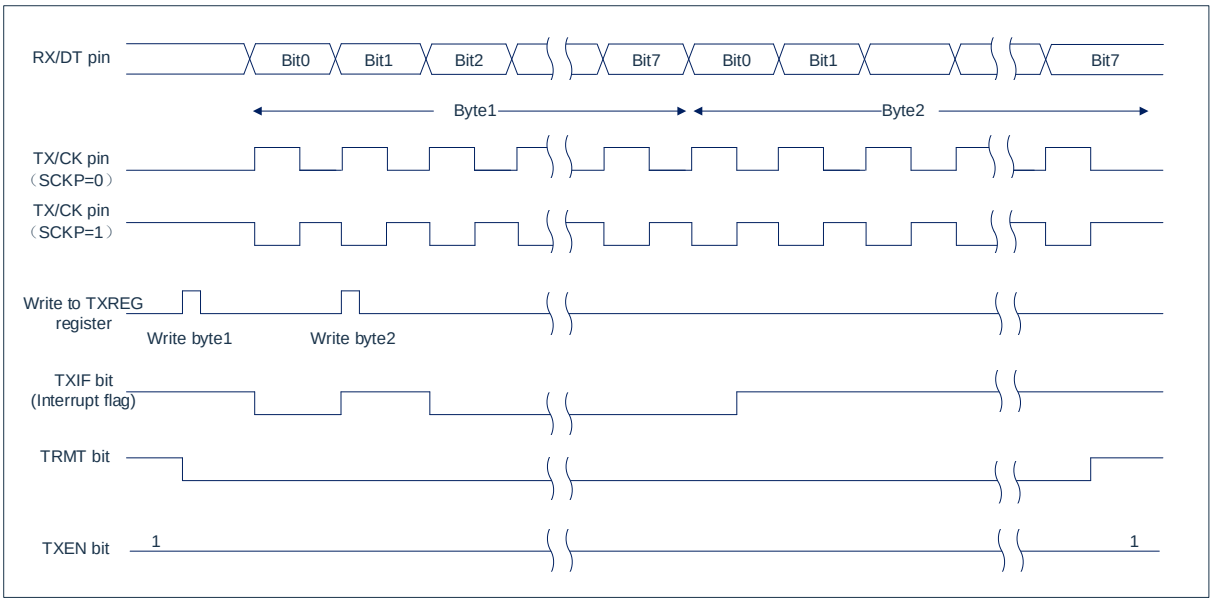


Fig 16-6: synchronous transmit

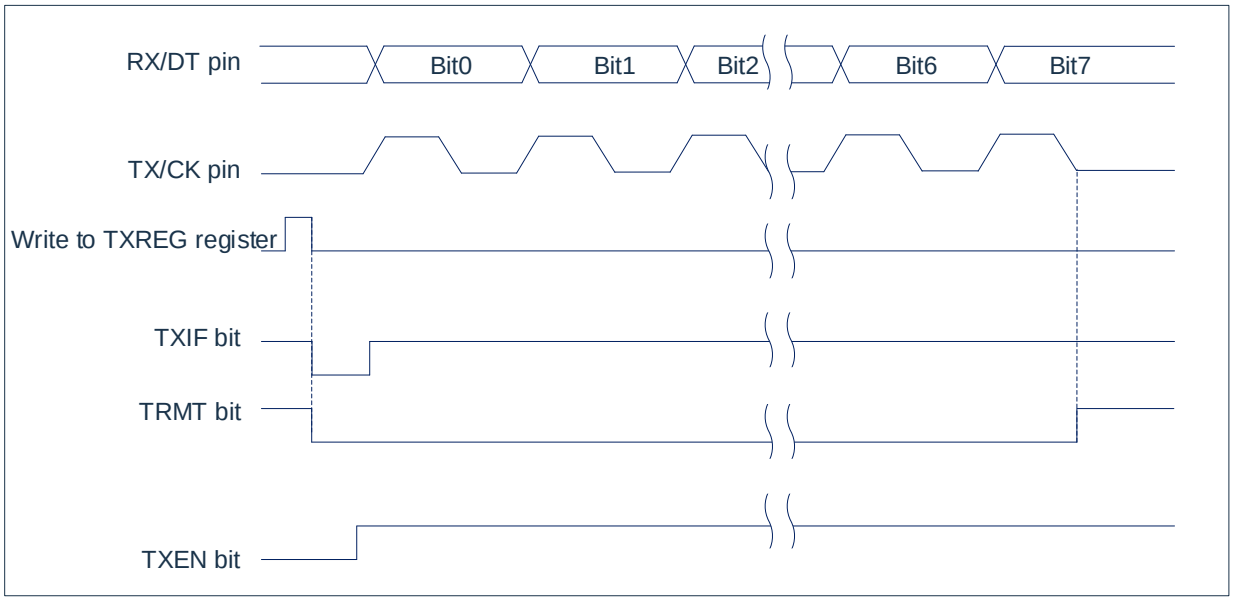


Fig 16-7: synchronous transmit (through TXEN)

16.5.1.5 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 RCSTA register) 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.

16.5.1.6 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.

16.5.1.7 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.

16.5.1.8 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.

16.5.1.9 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.

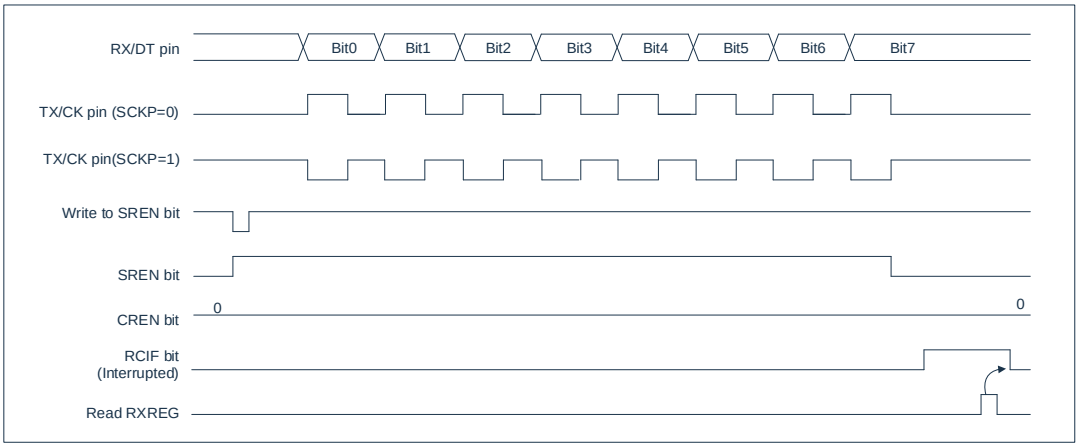


Fig 16-8：synchronous receive (master control mode, SREN)

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

16.5.2 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.

16.5.2.1 USART Synchronous Slave Transmit

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

16.5.2.2 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.

16.5.2.3 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.

16.5.2.4 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.

17. SPI Mode

17.1 SPI Mode General

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 4-wire mod：

- master data input/slave data output (MISO)
- master data output/slave data input (MOSI)
- serial clock (SCK)
- slave selection (SS)

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

- serial data input/output (SDIO)
- serial clock (SCK)
- slave selection (SS)

Note：In the following description, SDI is the MISO pin in the master mode and the MOSI pin in the slave mode; SDO is the MOSI pin in the master mode and the MISO pin in the slave mode.

17.2 SPI Related Registers

SPICON2: SPI control register (11EH)

11EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPICON2	---	CKE	MODE	---	---	---	---	SPIBF
read/write	---	R/W	R/W	---	---	---	---	R
Reset value	---	0	0	---	---	---	---	0

Bit7	Reserved	Write to 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~Bit1	Not used.	
Bit0	SPIBF:	Buffer full status bit
		1= Reception is complete, SPIBUF is full;
		0= Reception is not complete, SPIBUF is empty.

SPICON：SPI control register (11DH)

11DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPICON	SPIWCOL	SPIOV	SPIEN	SPICKP	SPIM3	SPIM2	SPIM1	SPIM0
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 SPIWCOL: Write conflict detection bit.
1= In the process of transmit/receive data, try to write to the SPIBUF register.
0= No conflict.

Bit6 SPIOV: Receive overflow flag bit.
1= When SPIBUF still keeps the previous data, a new byte is received. When overflow occurs, the data in SPISR will be lost. Overflow will only occur in slave mode. In slave mode, even if transmit data only, user must read SPIBUF 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 SPIBUF register (this bit must be clear through software).
0= No overflow.

Bit5 SPIEN: SPIEN enable bit.
1= Enable serial port and configure SCK, SDO, SDI and SS as serial port pin.
0= disable serial port and configure these pins as I/O port pins.

Bit4 SPICKP: Clock polarity selection bit.
1= Clock is high when idle.
0= Clock is low when idle.

Bit3~Bit0 SPIM<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= SCK pin，enable SS pin control;
0101= SPI slave mode，clock= SCK pin，disable SS pin control，SS can be used as I/O pin;
Others= Reserved;

17.3 SPI Working Principle

When initializing the SPI, several options need to be specified. They can be specified by programming the corresponding control bits (SPICON<5:0> and SPICON2<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)

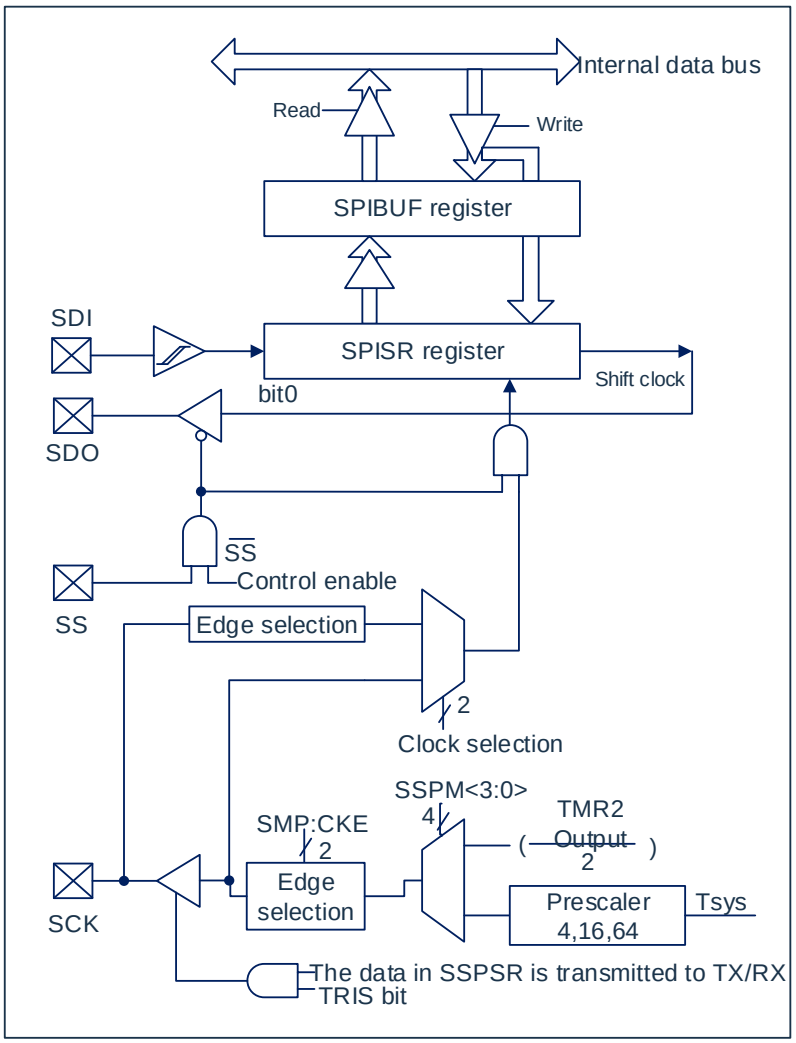


Fig 17-1 SPI mod block diagram

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

SPI mod consists of a transmit/receive shift register (SPISR) and a buffer register (SPIBUF). SPISR moves data in and out of the device, with the most significant bit first. SPIBUF saves the data written to the SPISR last time until the new receive. The data is ready. Once the 8-bit data receive is completed, the byte is moved into the SPIBUF register. Then, the interrupt flag bit SPIIF of the PIR1 register is set to 1. This double-buffered data receives method (SPIBUF) 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 SPIBUF register will be ignored, and the write conflict detection bit WCOL of the SPICON 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 SPIBUF is successfully completed.

When the application software is waiting for the receive valid data, it should read the previous data in the SPIBUF before the next data byte to be transmitted is written into the SPIBUF.

17.4 Enable SPI I/O

To enable the serial port, the SPI enable bit SPIEN of the SPICON register must be set to 1. To reset or reconfigure the SPI mode, first clear the SPIEN bit, reinitialize the SPICON register, and then set the SPIEN bit to 1. This will set MOSI, MISO, 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;
- The TRIS bit of MOSI must be cleared (master control mode);
- The TRIS bit of MISO must be cleared (slave mode);
- The TRIS bit of SCK (master control mode) must be cleared;
- The TRIS bit of SCK (slave mode) must be set to 1;
- The TRIS 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.

17.5 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 SPIBUF 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). SPISR 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 SPIBUF 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 SPICON register accordingly. Figure 17-2, Figure 17-3, Figure 17-4, and Figure 17-5 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 17-2 shows the waveform of the master control mode. When the CKE bit of the SPICON2 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 SPIBUF.

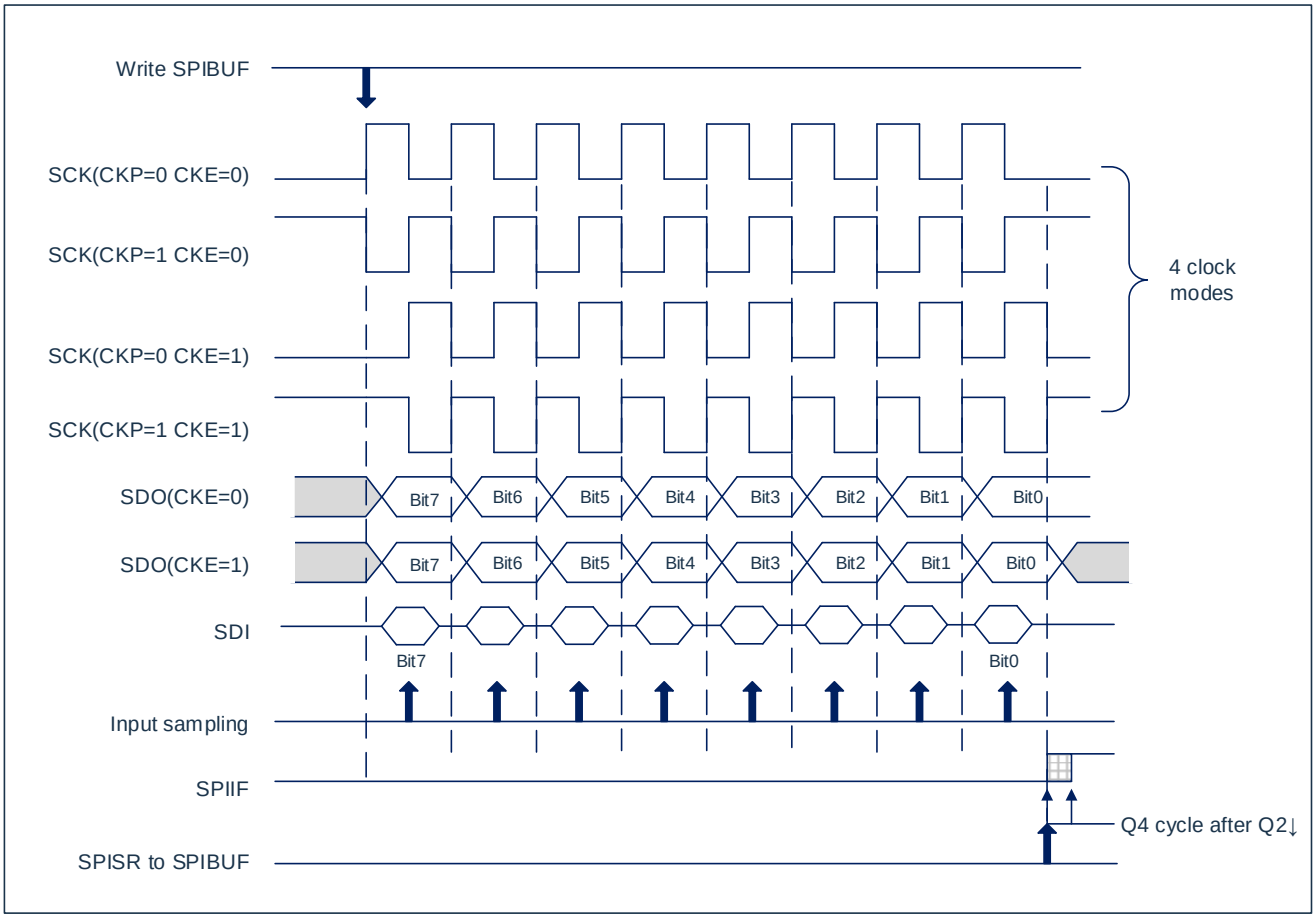


Fig17-2: SPI mode waveform (master control mode)

17.6 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 SPIIF 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.

17.7 Slave Synchronous Selection

SS pin allows the device to work in synchronous slave mode. SPI must work in slave mode, and enable SS pin to control SPICON<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 SPIEN 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, SDO pin 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:

1. When SPI works in slave mode and SS pin control is enabled (SPICON<3:0> = 0100), if SS pin is set to VDD level, SPI mod will be reset.
2. If SPI is used in slave mode with CKE set(SPICON2 register), SS pin control must be enabled.

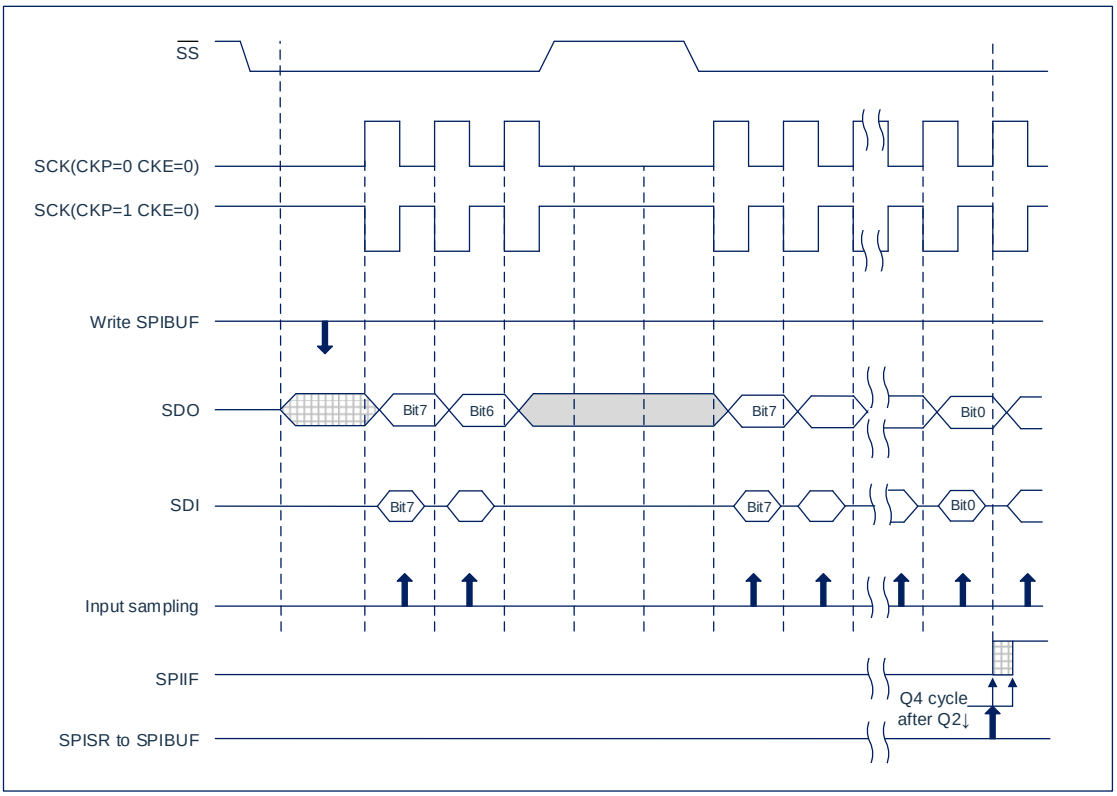


Fig 17-3: Slave synchronous waveform

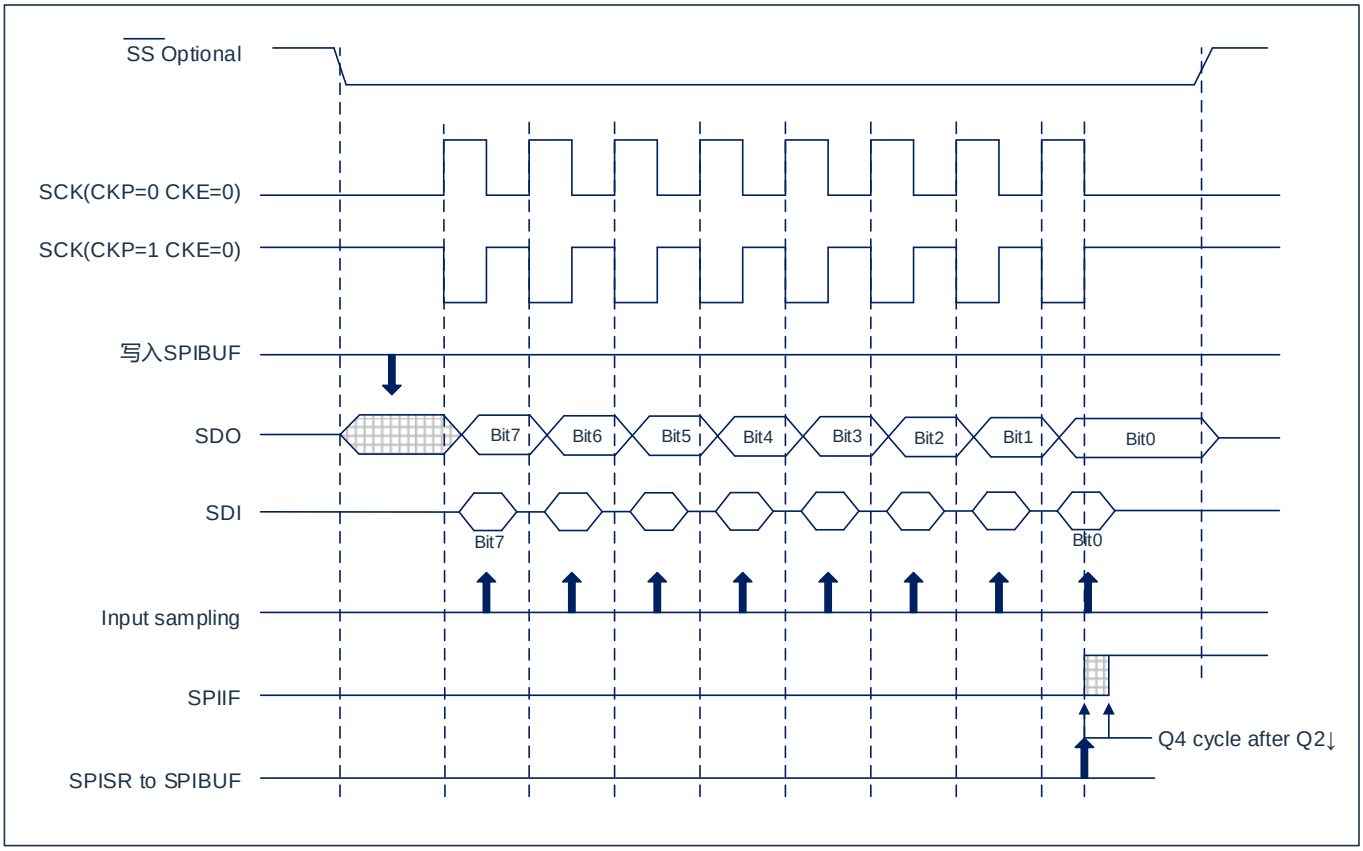


Fig 17-4: SPI mode waveform (slave mode，CKE=0)

17.8 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.

17.9 Effect of Reset

reset will disable SPI mod and terminate the current transmission.

18. IIC Mode

18.1 IIC Mode General

The IIC module 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). The IIC module implements standard mode specifications and 7-bit addressing.

There are two pins for data transmission. They are the clock pin (SCL) and the data pin (SDA). When using the IIC mode, the user must configure these pins as input pins through the corresponding TRIS bits.

By setting the IIC enable bit, IICEN, of the IICCON register to 1, IIC mod function is enabled.

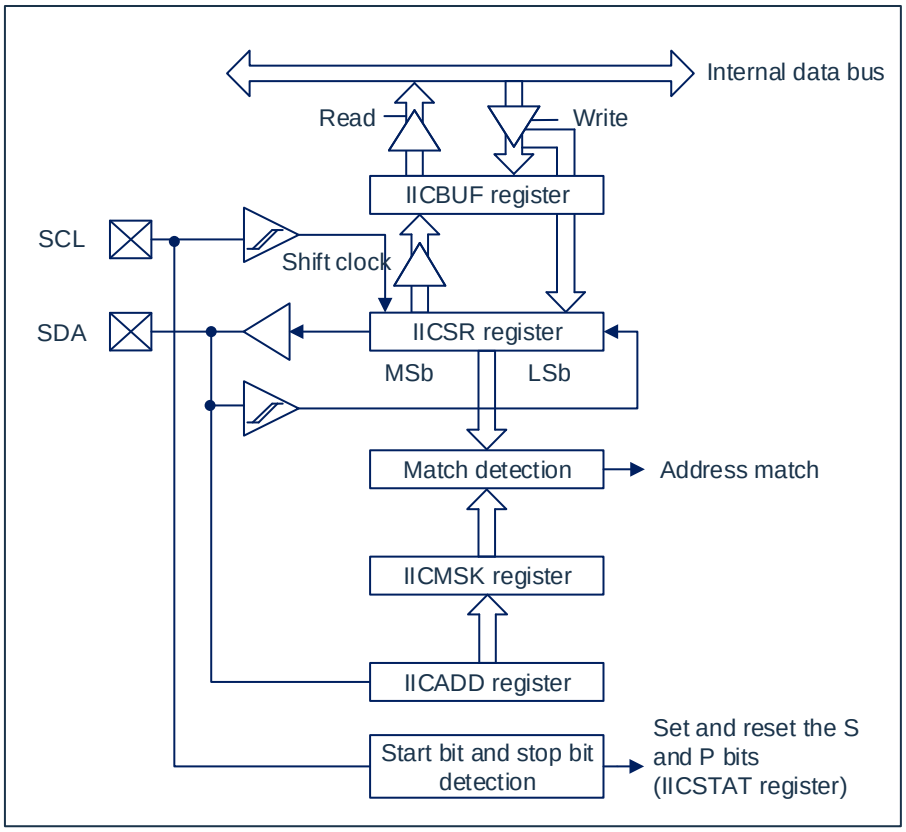


Fig18-1: I²C mode block diagram

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

IIC mod has 7 registers for I²C operation. They are:

- IIC control register1 (IICCON)
- IIC status register (IICSTAT)
- IIC shift register (IICSR): not directly accessible
- IIC masking register (IICMSK)
- IIC control register2 (IICCON2)
- serial receive/transmit buffer register (IICBUF)
- IIC address register (IICADD)

You can use IICCON register to control the operation of I^2C. You can use the IICM<1:0> mode selection bit (IICCON register) to select one of the following I^2C modes:

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

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

18.2 IIC Related Register

IICSTAT：IIC status register (10FH)

10FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IICSTAT	----	IDLE	D/A	P	S	R/W	IICTOF	BF
read/write	----	R	R	R	R	R	W/R	R
Reset value	----	0	0	0	0	0	0	0

Bit7 Not used.

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 IIC mode is disabled (IICEN 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 IIC mode (IICEN 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^2C slave mode:
1= read.
0= write.

I^2C master control mode:
1= transmitting.
0= not transmitting.
The result of logic OR operation between this bit and SEN, RSEN, PEN, RCEN or ACKEN will indicate whether IIC is in idle mode.

Bit1 IICTOF: Slave mode timeout flag
1= slave mode timeout has occurred.
0= slave mode timeout has not occurred.

Bit0 BF buffer full status bit.
receive:
1= receive complete, IICBUF full.
0= receive not complete，IIC BUF empty.
transmit:
1 = data transmitting (not including ACK and stop bit), IICBUF full.
0 = data transmit complete (not including ACK and stop bit), IICBUF empty.

IICCON：IIC control register (10CH)

10CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IICCON	IICWCOL	IICOV	IICEN	IICCKP	IICTOS[1:0]		IICM1	IICM0
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 IICWCOL: Write conflict detection bit.

master control mode: 1= Trying to write to the IICBUF register when I^2C does not meet the condition of starting transmit data.

0= no conflict.

Slave mode:

1= While transmitting the previous word, write the IICBUF register again (must clear through software).

0= no conflict.

Bit6 IICOV: Receive overflow flag bit. (only valid in slave receive mode)

1= When the IICBUF register still maintains the previous data, it receives a new byte. In the transmit mode, the IICOV bit can be any value (this bit must be clear through software).

0= No overflow.

Bit5 IICEN: IIC enable bit (These pins must be correctly configured as input 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 IICCKP: Clock polarity selection bit.

In I^2C slave mode: SCK release control.

1 = enable clock.

0 = Keep clock line is low (clock extension) (used to ensure data establishment time).

In I^2C master control mode: Not used.

Bit3~Bit2 IICTOS<1:0>: IIC slave mode timeout selection

00= disable IIC slave timeout function.

01= enable the IIC slave timeout function, the timeout period is 16ms, reset the IIC module after the timeout.

10= enable the IIC slave timeout function, the timeout period is 32ms, reset the IIC module after the timeout.

11= enable the IIC slave timeout function, the timeout period is 64ms, reset the IIC module after the timeout.

Bit1~Bit0 IICM<1:0>: IIC mode selection bit.

00= I2C master control mode，clock= FSYS/ (8* (IICADD+1)).

01= I2C slave mode，7-bit address，does not respond to start bit and stop bit interrupt.

10= I2C slave mode，7-bit address，allow start bit and stop bit interrupt.

11= Allow operation of IICMSK register

IICCON2: IIC control register2 (10DH)

10DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IICCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
read/write	R/W	R/W	R	R	R	R	R	R
Reset value	0	0	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 IICSR.
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 control mode).
1 = Start stop condition on SDA and SCL pin. Automatically cleared by hardware.
0 = stop condition idle.

Bit1 RSEN: Repeat enable bit (only in I^2Cmaster control mode).
1= Initiate repeated start conditions on the SDA and SCL pins. Automatically cleared by hardware.
0= Repeated start condition is 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 = start condition is idle.
In slave mode:
1 = Both transmit and receive will enable clock extension (enable clock extension).
0 = disable clock extension.

18.3 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 IIC 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 IIC hardware, and SDA pin must be configured as input (the corresponding pin TRIS bit is set to 1). The following events will set the IIC interrupt flag bit IICIF to 1 (if IIC interrupt is allowed, interrupt will be generated):

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

18.3.1 I²C Master Control Mode Support

The master control mode can be enabled by setting the corresponding IICM bit in IICCON to 1 or clearing it and setting the IICEN 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 IICBUF 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.

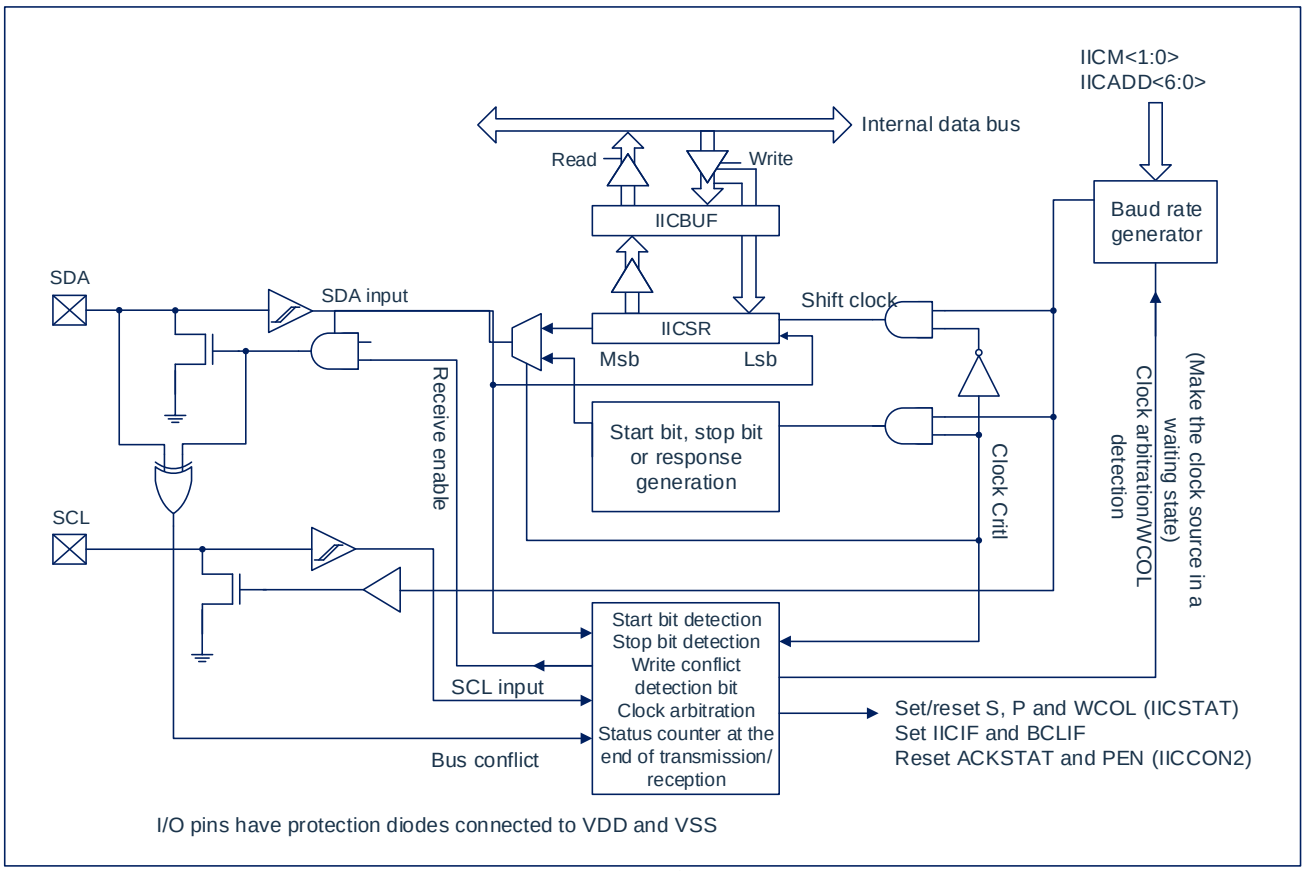


Fig 18-2: IIC block diagram (I²C™ master control mode)

Note: When configured as I²C master mode, IIC 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 IICBUF register immediately to initiate the transfer. In this case, IICBUF will not be written and the WCOL bit will be set to 1, which indicates that no write operation to IICBUF has occurred.

18.3.1.1 I²C Master Control Mode Operation

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 I²C 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 I²C 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 IICADD register. When a write to IICBUF 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 (IICCON2 register) to 1.
- IICIF set to 1. Before performing any other operations, IIC mod will wait for the required startup time.
- The user will load the IICBUF from the device address to transmit.
- The address is moved out of the SDA pin until all 8 bits are transmitted.
- IIC mod shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the IICCON2 register.
- IIC mod sets the IICIF bit to 1 at the end of the 9th clock period, generating an interrupt.
- The user loads 8-bit data into IICBUF.
- Data is moved out from the SDA pin until all 8 bits are transmitted.
- IIC mod shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the IICCON2 register.
- IIC mod sets the IICIF 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 (IICCON2 register) to 1.
- Once the stop condition is completed, an interrupt will be generated.

18.3.2 Baud Rate Generator

In I²C master control mode, the baud rate generator reloaded value is located in the lower 7 bits of the IICADD register (Figure 18-3). 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 I²C 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 18-4).

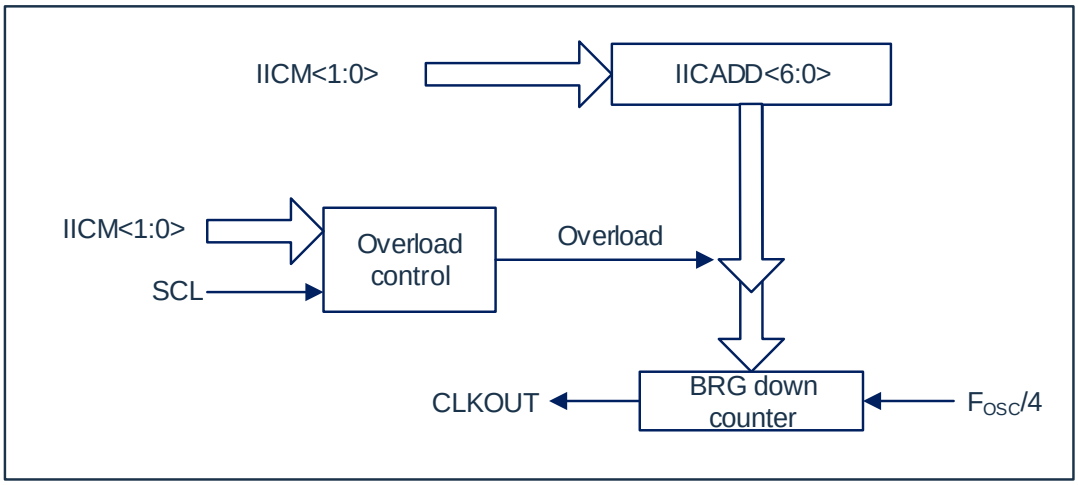


Fig 18-3: baud rate generator block diagram

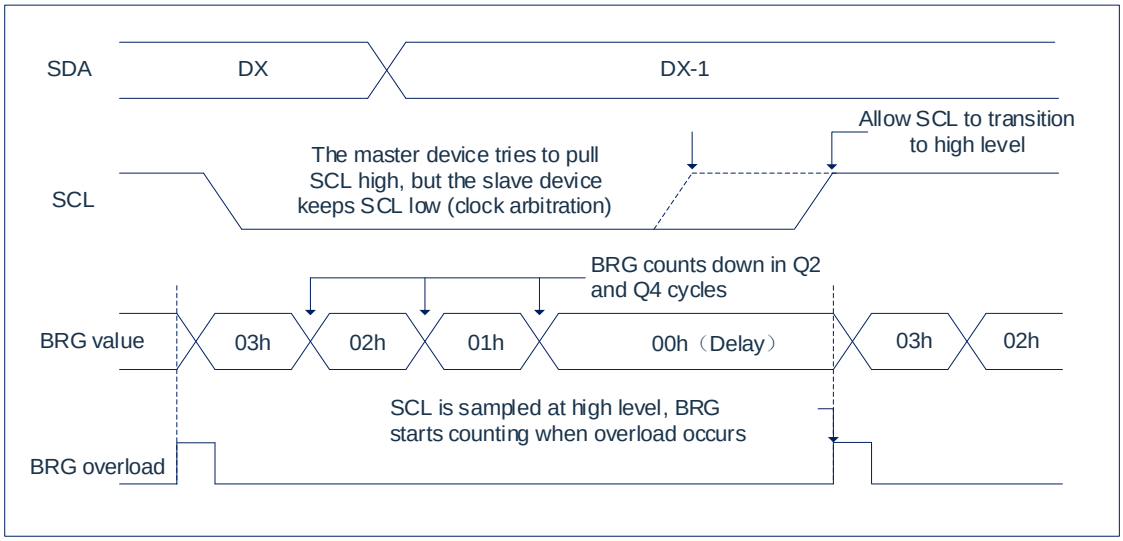


Fig 18-4: Time series of baud rate generator with clock arbitration

18.3.3 I²C Master Control Mode Transmit

Transmit a data byte and a 7-bit address can be achieved directly by writing a value to the IICBUF 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 IICIF bit will be set to 1, and the master control clock (baud rate generator) will be suspended until the next data byte is loaded into IICBUF, SCL pin remains low and SDA remains unchanged.

After writing IICBUF, 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 (IICCON2 register). After the 9th clock falling edge of the transmit address, IICIF is set to 1, the BF flag bit is cleared, the baud rate generator is turned off until the next write operation to IICBUF, and the SCL pin remains low, allowing the SDA pin to suspend.

18.3.3.1 BF Status Indication

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

18.3.3.2 WCOL Status Indication

If the user writes IICBUF during the transmit process (that is, when the IICSR 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.

18.3.3.3 ACKSTAT Status Indication

In transmit mode, when the slave device transmits a response (ACK=0), the ACKSTAT bit (IICCON2 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.

18.3.4 I²C Master Control Mode Receive

By programming receive enable bit RCEN (IICCON2 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 IICSR. After the falling edge of the eighth clock, the receive enable flag bit is automatically cleared, the content of IICSR is loaded into IICBUF, the BF flag bit is set to 1, the IICIF flag bit is set to 1, the baud rate generator pauses counting, and the SCL remains at low level. At this time, IIC 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 (IICCON2 register) to 1, the user can end the receive transmit response bit.

18.3.4.1 BF Status Indication

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

18.3.4.2 WCOL Status Indication

If the user writes IICBUF during the receive process (that is, when the IICSR 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).

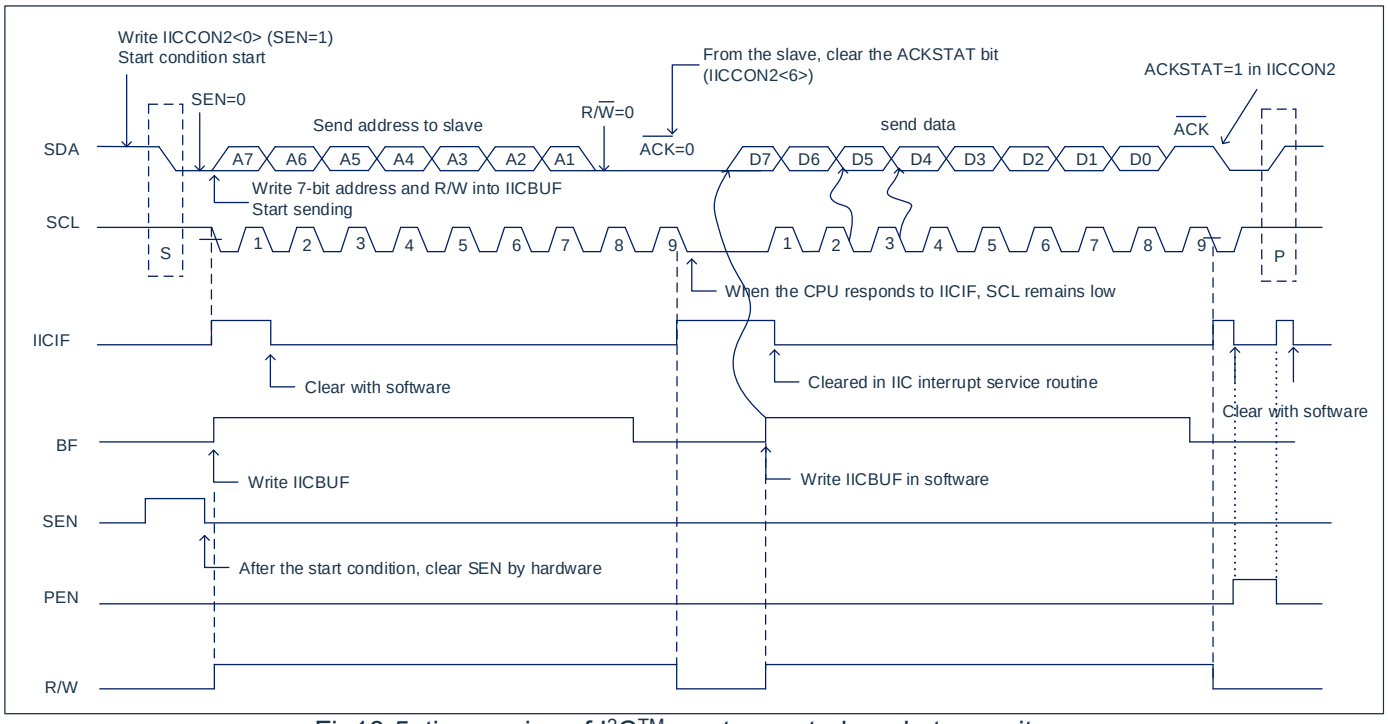


Fig18-5: time series of I²C™ master control mode transmit

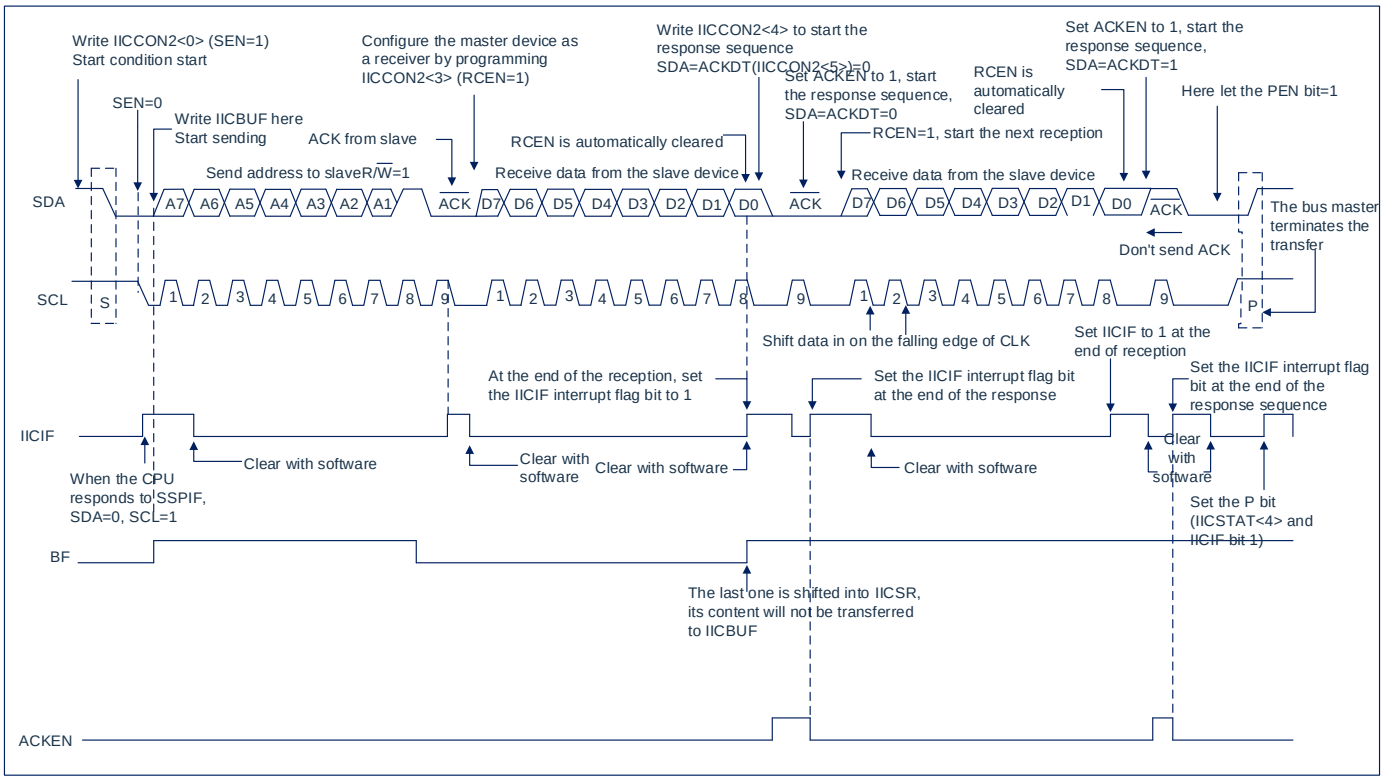


Fig 18-6: Time series of I²C™ master control mode receive (7-bit address)

18.3.5 I²C Master Control Mode Start Condition Time Series

To initiate a start condition, the user should set the start condition enable bit SEN of the IICCON2 register to 1. When both SDA and SCL pins are sampled as high, the baud rate generator reloads the contents of IICADD<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 (IICSTAT register) to 1. Then the baud rate generator reloads the contents of IICADD<6:0> and resumes counting. When the baud rate generator times out (TBRG), the SEN bit of the IICCON2 register will be automatically cleared by hardware. The baud rate generator is paused, the SDA line remains low, and the start condition ends.

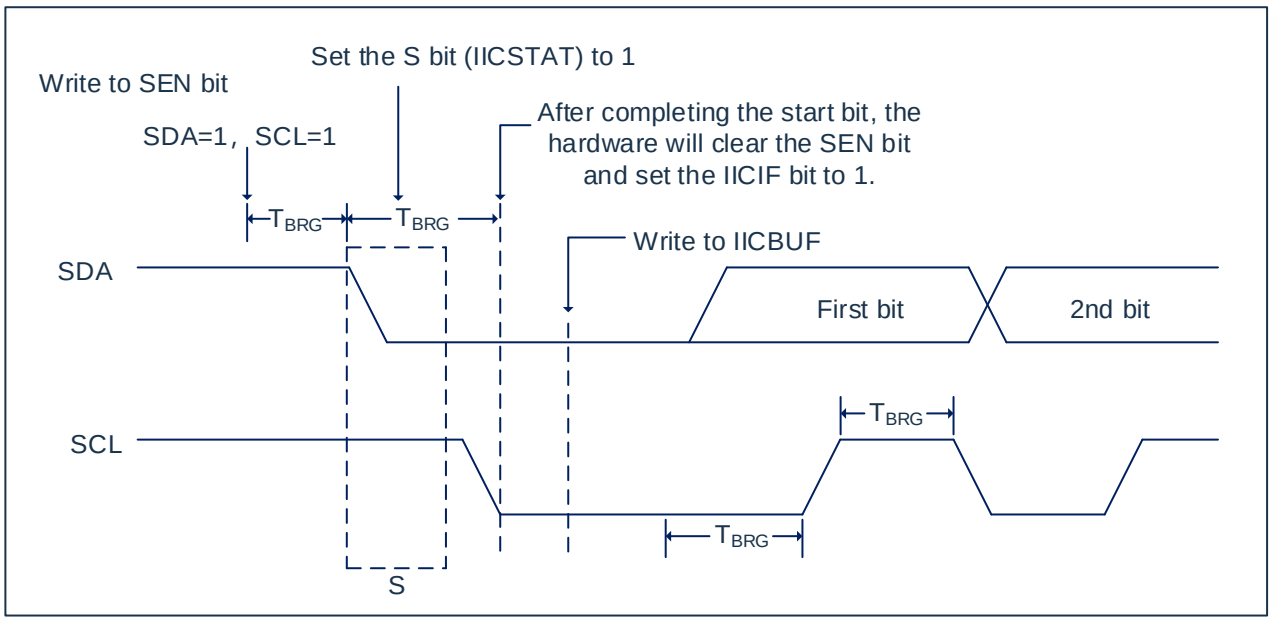


Fig 18-7：time series for the first starting bit

18.3.5.1 WCOL Status Indication

When the startup sequence is in progress, if the user writes IICBUF, 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 IICCON2 cannot be written before the start condition ends.

18.3.6 I²C Master Control Mode Repeat Condition Time Series

When the RSEN bit (IICCON2 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 IICADD<6:0> and starts counting. In a baud rate generator 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 IICADD<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 (IICCON2 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 (IICSTAT register) will be set to 1. The IICIF bit will not be set to 1 until the baud rate generator times out.

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

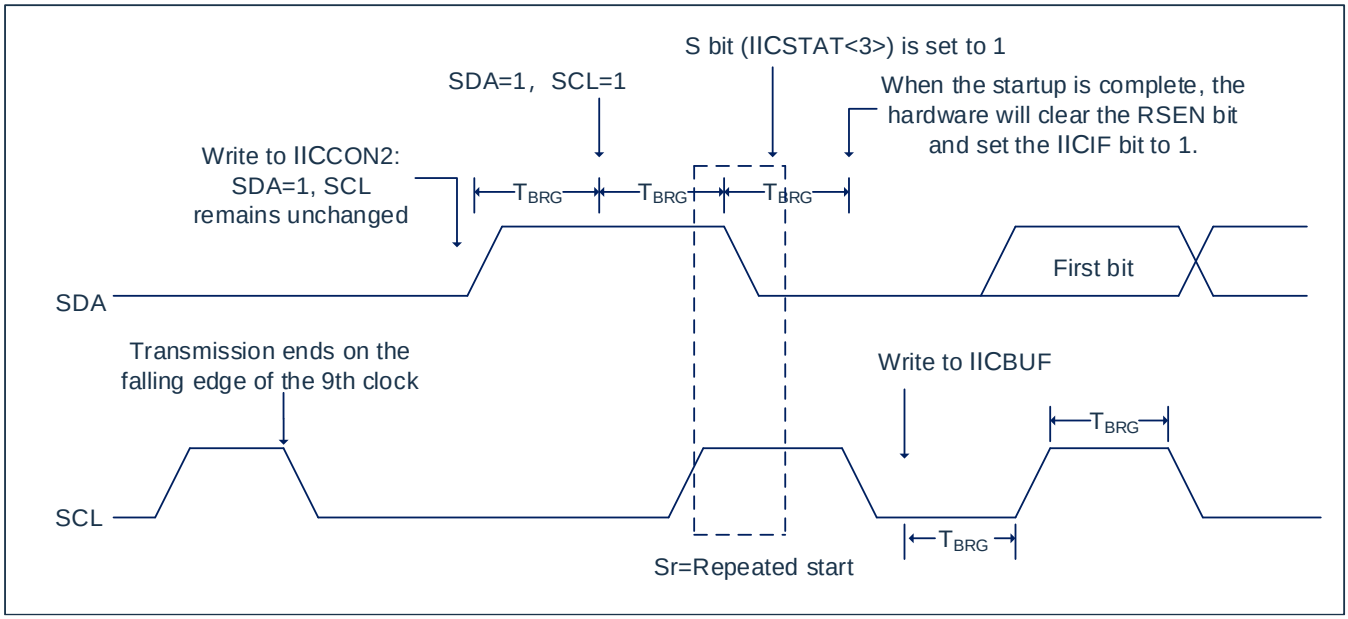


Fig 18-8: time series of repeat condition

18.3.6.1 WCOL Status Indication

When the repeated start sequence is in progress, if the user writes IICBUF, 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 IICCON2 cannot be written until the repeated start condition ends.

18.3.7 ACK Time Series

Set the ACK enable bit ACKEN (IICCON2 register) 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 SCL pin is pulled low. After that, the ACKEN bit is automatically cleared, baud rate generator is turned off, and IIC mod enters idle mode.

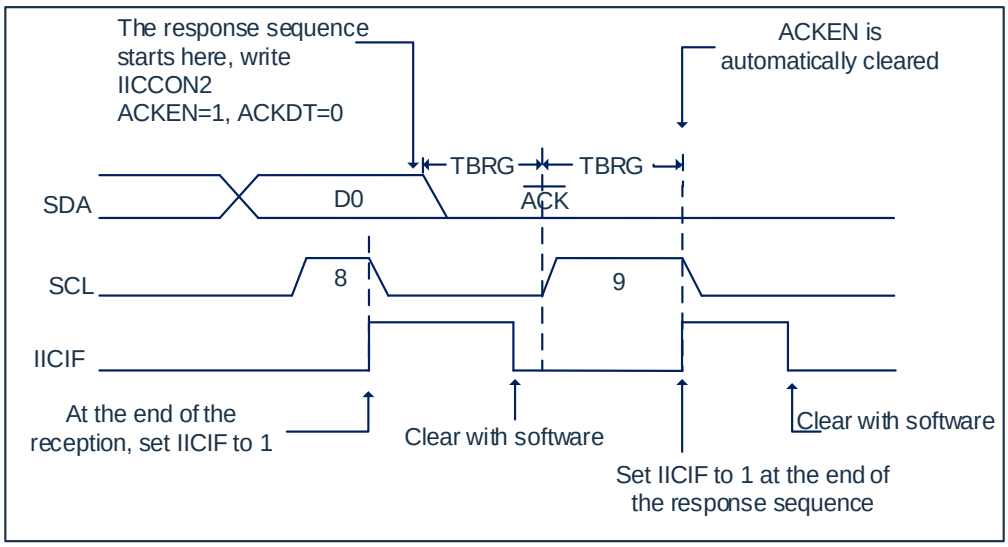


Fig 18-9: times series for ACK

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

18.3.7.1 WCOL Status Indication

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

18.3.8 Stop Condition

At the end of receive/transmit, by setting the enable bit of the stop sequence, PEN (IICCON2 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 rate generator 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 (IICSTAT register) set to 1. After a TBRG period, the PEN bit is cleared and the IICIF bit is set to 1.

18.3.8.1 WCOL Status Indication

If the user attempts to write IICBUF 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).

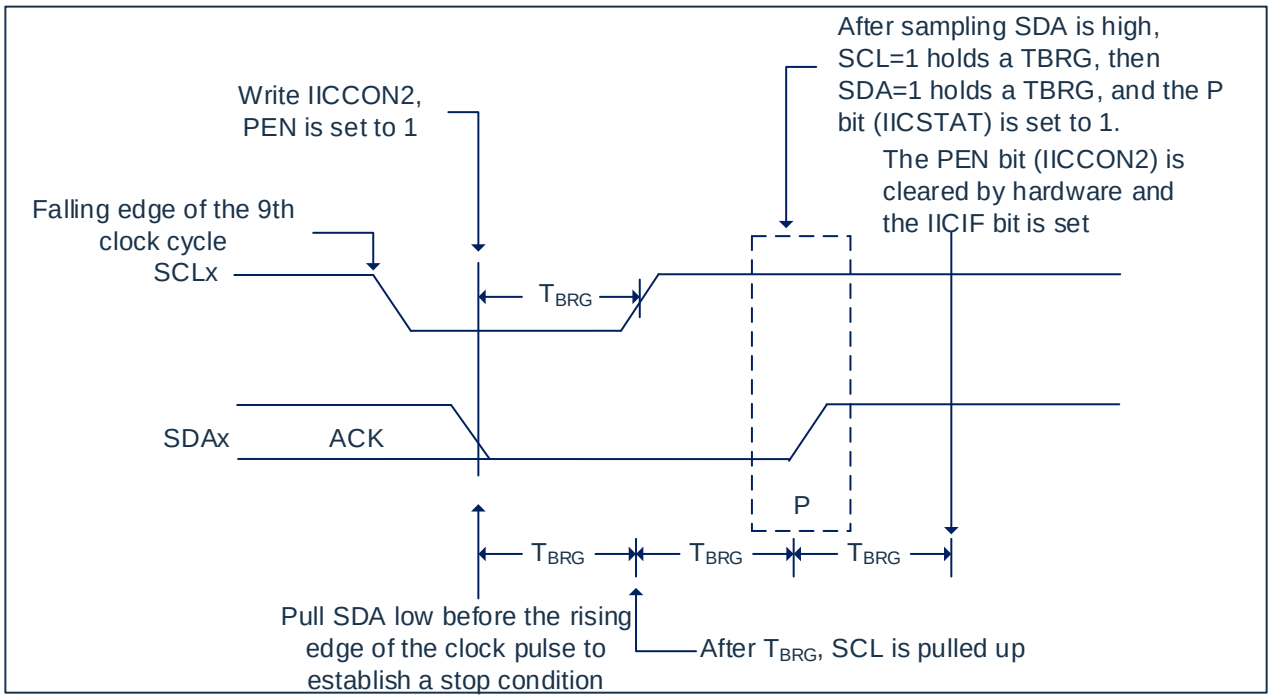


Fig18-10：stop condition receive or transmit mode

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

18.3.9 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 IICADD<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.

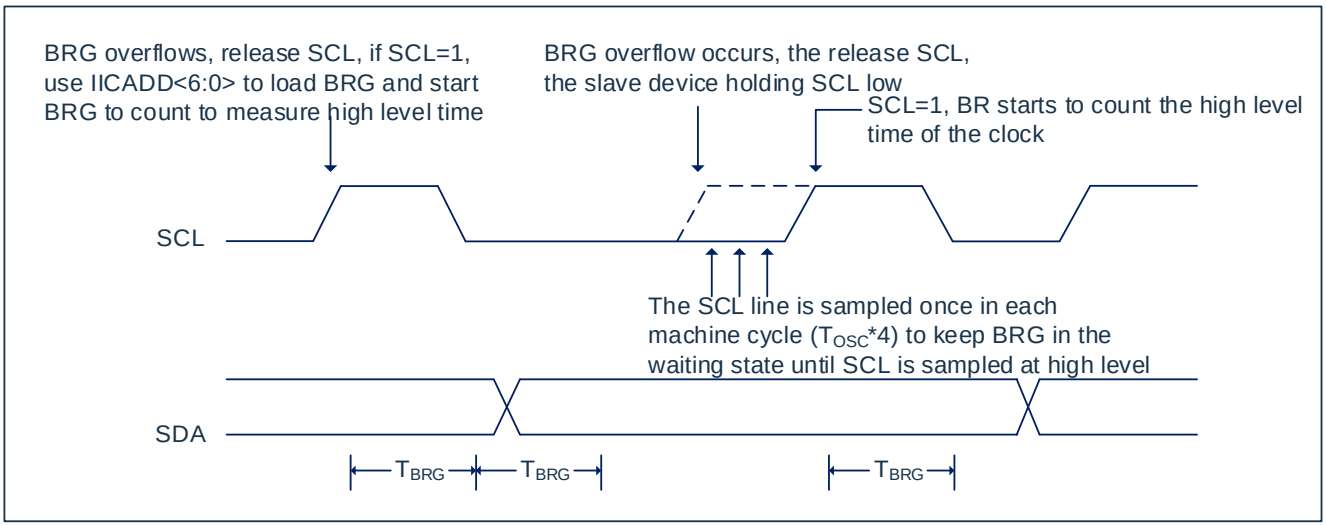


Fig 18-11: clock arbitration in master control transmit mode

18.3.10 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 IIC mod. When the P bit is set to 1, you can get control of the I^2C 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 IIC 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
- Start condition
- ACK conditions
- data transmission
- Repeated start condition

18.3.11 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 IICIF 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 IICBUF 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 IICCON2 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 IICIF bit will be set to 1. No matter what bus occurs What is the progress of the transmit during conflict, writing IICBUF 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.

18.4 Slave Mode

In slave mode, SCL pin and SDA pin must be configured as input. When needed (such as from the transmitter), the IIC 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 IICSR register at the time into the IICBUF register.

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

- The buffer full flag bit BF (IICCON register) is 1 before the received data to be transmitted.
- Before receiving the transmitted data, the overflow flag bit IICOV (IICCON register) has been set 1

In this case, the value of IICSR register will not be loaded into IICBUF, but the IICIF bit of PIR1 register will be set to 1. The BF bit is cleared by reading the IICBUF register, and the IICOV 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.

18.4.1 Addressing

Once IIC 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 IICSR register. All input bits are sampled on the rising edge of the clock (SCL) line. Register IICSR<7:1> The value will be compared with the value of the IICADD register. The comparison is performed on the falling edge of the 8th clock pulse (SCL). If the address matches and the BF bit and IICOV bit are zero, the following events will occur:

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

18.4.2 Receiving

When the R/W bit of the address byte is cleared and an address match occurs, the R/W bit of the IICSTAT register is cleared. The received address is loaded into the IICBUF 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 (IICSTAT register) is set to 1, or the IICOV bit (IICCON register) is set to 1. Each data transmission byte will generate an IIC interrupt. The interrupt flag bit IICIF of the PIR1 register must be cleared by software. The IICSTAT register is used to determine the status of the byte.

18.4.3 Transmit

When the R/W bit of the received address byte is 1 and an address match occurs, the R/W bit of the IICSTAT register is 1. The received address is loaded into the IICBUF register. The ACK pulse is transmitted on the 9th bit while the SDA pin remains low. The transmitted data must be loaded into the IICBUF register and also into the IICSR register. Then the CKP bit (IICCON 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 IIC interrupt. The IICIF flag bit must be clear through software, and the IICSTAT register is used to determine the status of the byte. The IICIF 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 IICSTAT 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 IICBUF register, which will also load the IICSR register. CKP should be set 1 to enable SCL.

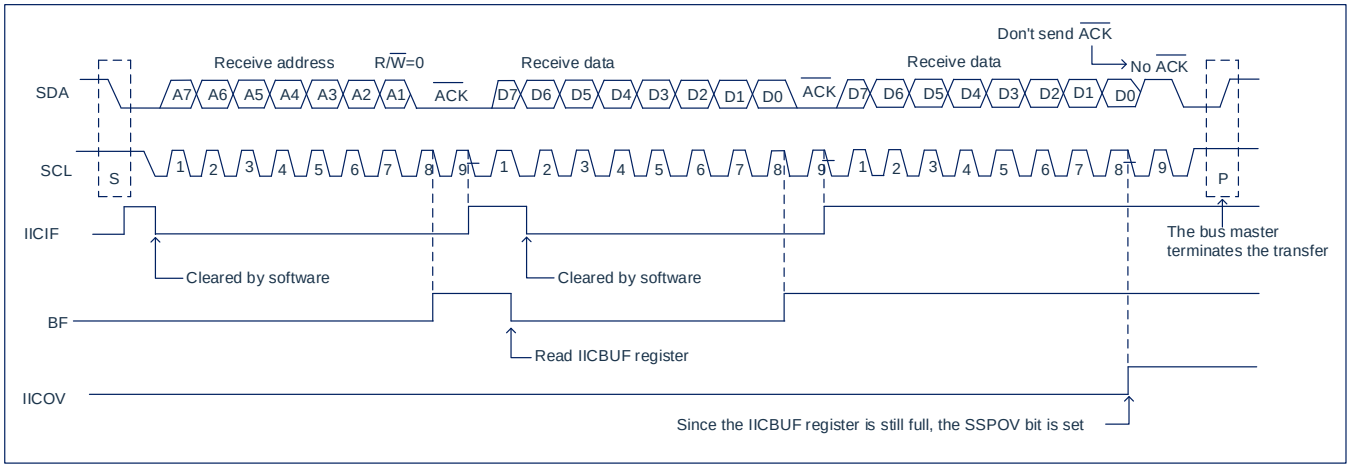


Fig 18-12：Time series for I²C™ slave mode receive (7-bit address)

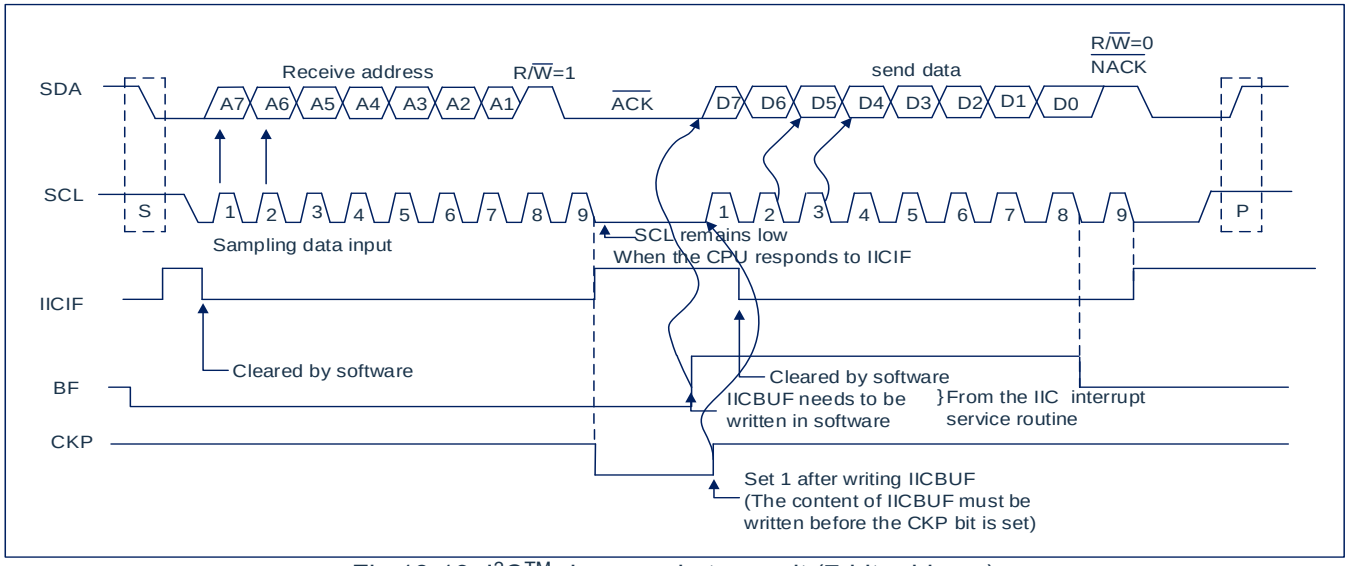


Fig 18-13: I²C™ slave mode transmit (7-bit address)

18.4.4 I2C Masking Register

In I²C slave mode, the IIC mask (IICMSK) register is used to mask the value in the IICSR register under the address compare operation. A bit of 0 in the IICMSK register will make the corresponding bit in the IICSR 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 IIC operation before writing the mask value. The register must be initialized before selecting the I²C slave mode by setting the IICM<1:0> bits. This register can only be accessed after the appropriate mode is selected through the IICM<1:0> bits of IICCON.

The IIC 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>

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

IICMSK: IIC masking register (I09H) (1)

109H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IICMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	---
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	---
Reset value	1	1	1	1	1	1	1	---

Bit7~Bit1 MSK<7:1>: Mask bit.
1= Bit n of the received address is compared with IICADD<n> to detect the match of the I²C address.
0= Bit n of the received address is not used to detect I²C address matching.

Bit 0 not used.

Note:
1) When the IICCON bit IICM<1:0> = 11, any read or write operation to the IICADDSFR address is performed through the IICMSK register.
2) In all other IIC modes, this bit is invalid

18.4.5 I2C Slave Timeout Protection

In I²C slave mode, the timeout protection function can be turned on. The time-out counter starts to work after the address matches and is cleared at the falling edge of SCL. When the next SCL falling edge comes, if the counting clock exceeds the time set by IICTOS, the time-out protection function is triggered, and the IIC slave receiver is automatically reset at this time. The module, the bus resumes the idle state, and the IICTOF flag is set to 1 (cleared by software).

18.5 Operation Under Sleep Mode

In sleep mode, I2C mod cannot be used.

18.6 Effect of Reset

Reset will disable IIC mod and terminate the current transmission.

19. Electrical Parameter

19.1 Limit Parameter

Supplying voltage.....GND-0.3V~GND+6.0V
storage temperature ..... -50℃~125℃
working temperature..... -40℃~85℃
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.

19.2 DC Feature

Symbol	Parameter	Test condition		Min	Typ	Max	Unit
		VDD	Condition				
VDD	Working voltage	-	F_{SYS}=16MHz/2T	V_{LVR3}		5.5	V
		-	F_{SYS}=16MHz/4T or F_{SYS}=8MHz/2T	V_{LVR2}		5.5	V
		-	F_{SYS}=8MHz/4T	V_{LVR1}		5.5	V
I_{DD}	Working current	5V	F_{SYS}=16MHz		3		mA
		3V	F_{SYS}=16MHz		2		mA
		5V	F_{SYS}=8MHz		2		mA
		3V	F_{SYS}=8MHz		1		mA
		5V	Burning program EEPROM		6		mA
I_{STB}	Static current	5V	LVR=DIS WDT=DIS		0.5	5	μA
		3V	LVR=DIS WDT=DIS		0.4	3	μA
		5V	LVR=DIS WDT=EN		4.5		μA
		3V	LVR=DIS WDT=EN		3.5		μA
		5V	LVR=EN WDT=DIS		9.5		μA
		3V	LVR=EN WDT=DIS		8.5		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		----	2.5		5.5	V
R_{PH}	pull up resistor resistance	5V	V_O=0.5VDD		32		KΩ
		3V	V_O=0.5VDD		56		KΩ
R_{PD}	pull down resistor resistance	5V	V_O=0.5VDD		35		KΩ
		3V	V_O=0.5VDD		60		KΩ
I_{OL1}	Output port source current	5V	V_{OL}=0.3VDD		47		mA
		3V	V_{OL}=0.3VDD		23		mA
I_{OL2}	Output port source current high current PWM port	5V	V_{OL}=0.3VDD		68		mA
		3V	V_{OL}=0.3VDD		29		mA
I_{OH1}	Output port drain current	5V	V_{OH}=0.7VDD		-20		mA
		3V	V_{OH}=0.7VDD		-8		mA
I_{OH2}	Output port drain current high current PWM port	5V	V_{OH}=0.7VDD		-70		mA
		3V	V_{OH}=0.7VDD		-35		mA
V_{BG}	Internal reference voltage 0.6V	VDD=2.5~5.5V T_A=25℃		-1.5%	0.6	1.5%	V
		VDD=2.5~5.5V T_A=－20~85℃		-2.0%	0.6	2.0%	V

19.3 ADC Feature

(T_A= 25℃，Unless otherwise indicated)

Symbol	Parameter	Test condition	Min	Typ	Max	Unit
V_{ADC}	ADC working voltage	V_{ADREF}= VDD,F_{ADCCLK}=2MHz	3.0		5.5	V
		V_{ADREF}= VDD,F_{ADCCLK}=1MHz	2.5		5.5	V
		V_{ADREF}=2.0V,F_{ADCCLK}=250kHz	2.5		5.5	V
		V_{ADREF}=2.4V,F_{ADCCLK}=250kHz	2.6		5.5	V
		V_{ADREF}=3.0V,F_{ADCCLK}=500kHz	3.3		5.5	V
I_{ADC}	ADC current	V_{ADC}=5V V_{ADREF}= VDD, F_{ADCCLK}=500kHz			500	μA
		V_{ADC} =3V V_{ADREF}= VDD, F_{ADCCLK}=500kHz			200	μA
V_{AIN}	ADC input voltage	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=500kHz	0		V_{ADC}	V
DNL1	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=1MHz			±2	LSB
INL1	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=1MHz			±2	LSB
DNL2	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = 3.0V, F_{ADCCLK}=500kHz,V_{AIN}<1V			±3	LSB
INL2	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = 3.0V, F_{ADCCLK}=500kHz,V_{AIN}<1V			±3	LSB
DNL3	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.4V, F_{ADCCLK}=250kHz,V_{AIN}<0.8V			±3	LSB
INL3	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.4V, F_{ADCCLK}=250kHz,V_{AIN}<0.8V			±3	LSB
DNL4	Differential nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.0V, F_{ADCCLK}=250kHz,V_{AIN}<0.7V			±3	LSB
INL4	Integral nonlinearity error	V_{ADC}=5V, V_{ADREF} = 2.0V, F_{ADCCLK}=250kHz,V_{AIN}<0.7V			±3	LSB
T_{ADC}	ADC conversion time	-		49		T_{ADCCLK}

19.4 ADC Internal LDO Reference Voltage Characteristics

(T_A= 25℃，Unless otherwise indicated)

Symbol	Parameter	Test condition	Min	Typ	Max	Unit
V_{ADREF1}	LDO_OUT=2.0V	VDD=5V TA=25℃	-0.6%	2.0	+0.6%	V
		VDD=2.5~5.5V T_A=25℃	-1.0%	2.0	+1.0%	V
		VDD=2.5~5.5V T_A=-40℃~85℃	-1.5%	2.0	+1.5%	V
V_{ADREF2}	LDO_OUT=2.4V	VDD=5V T_A=25℃	-0.6%	2.4	+0.6%	V
		VDD=2.7~5.5V T_A=25℃	-1.0%	2.4	+1.0%	V
		VDD=2.7~5.5V T_A=-40℃~85℃	-1.5%	2.4	+1.5%	V
V_{ADREF3}	LDO_OUT=3.0V	VDD=5V T_A=25℃	-0.6%	3.0	+0.6%	V
		VDD=3.5~5.5V T_A=25℃	-1.0%	3.0	+1.0%	V
		VDD=3.5~5.5V T_A=-40℃~85℃	-1.5%	3.0	+1.5%	V

19.5 OPA Electrical Characteristics

(T_A=25℃，Unless otherwise indicated)

Symbol	Parameter	Test condition	Min	Typ	Max	Unit
DC Electrical Characteristics						
VDD	Working voltage	VDD=2.5~5.5V	2.5		5.5	V
I_{DD}	Static current	VDD=5.0V		380		uA
V_{OPOS}	Input offset voltage	No calibration VDD=5V VCM=1V	-20		20	mV
		After calibration VDD=5V VCM=1V	-5		5	mV
VCM	Common-mode Input Range		0		VDD-1.5V	V
PSRR	Power supply voltage rejection ratio*		60	70		dB
CMRR	Common-mode rejection ratio*	VDD=5V VCM=0~VDD-1.5V	90	100		dB
AC Electrical Characteristics						
AOL	Open loop gain*		90	100		dB
GBW	Gain bandwidth*	RL=1MΩ，CL=100pF	1.5	2		MHz

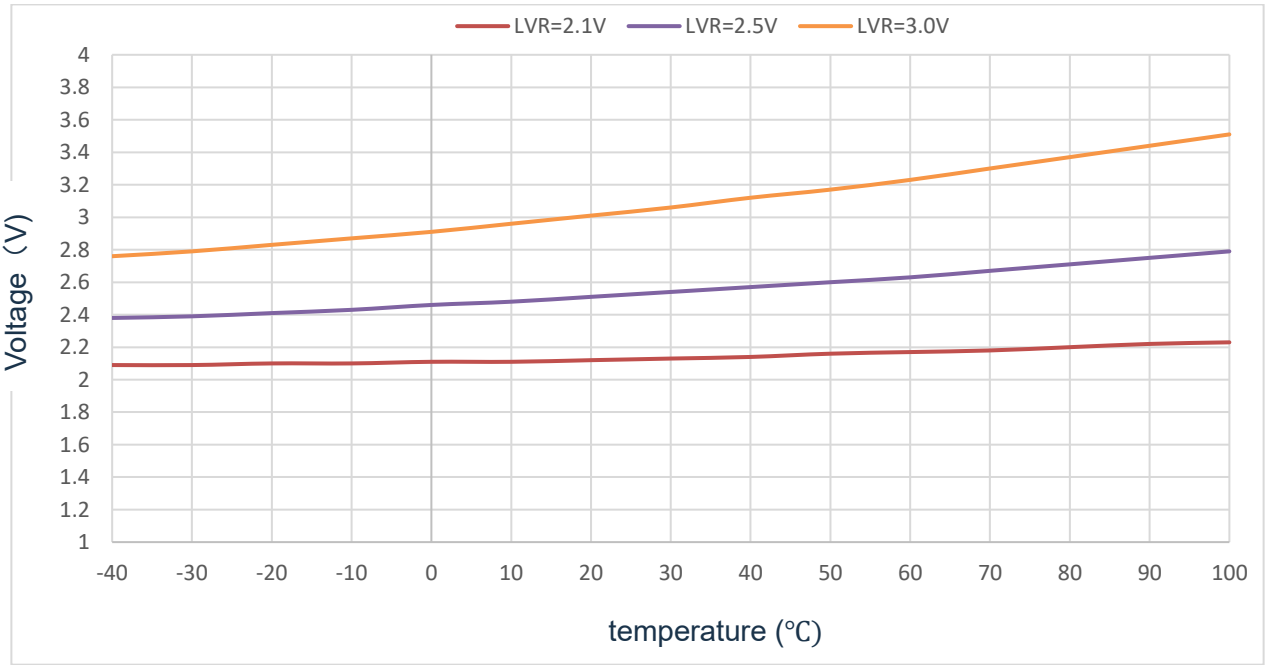
*Means guaranteed by design, not batch tested.

19.6 LVR Electrical Characteristics

(T_A=25℃，Unless otherwise indicated)

Symbol	Parameter	Test condition	Min	Typ	Max	Unit
V_{LVR1}	LVR set voltage =2.1V	VDD=1.8~5.5V	2.0	2.1	2.2	V
V_{LVR2}	LVR set voltage =2.5V	VDD=2.2~5.5V	2.4	2.5	2.6	V
V_{LVR3}	LVR set voltage =3.0V	VDD=2.8~5.5V	2.9	3.0	3.1	V

LVR Temperature characteristic curve



19.7 LVD Electrical Characteristics

(T_A=25℃，Unless otherwise indicated)

Symbol	Parameter	Test condition	Min	Typ	Max	Unit
V_{LVD}	Working voltage	-	2.1		5.5	V
	Accuracy	VDD=2.1~5.5V， T_A=－20~85℃	-5%	V_{SET}	+5%	V

19.8 AC Electrical Characteristics

(T_A=25℃，Unless otherwise indicated)

Symbol	Parameter	Test condition		Min	Typ	Max	Unit
		VDD	Condition				
T_{WDT}	WDT reset time	5V	-		16		ms
		3V	-		16		ms
T_{EEPROM}	EEPROM programming time	5V	F_{HSI}=8MHz		4.6	5	ms
		3V	F_{HSI}=8MHz		4.6	5	ms
		5V	F_{HSI}=16MHz		4.6	5	ms
		3V	F_{HSI}=16MHz		4.6	5	ms
F_{RC}	Internal frequency stability	VDD=4.0~5.5V	T_A=25℃	-1.50%	8	1.50%	MHz
		VDD=2.1~5.5V	T_A=25℃	-2.00%	8	2.00%	MHz
		VDD=4.0~5.5V	T_A=-40~85℃	-2.50%	8	2.50%	MHz
		VDD=2.1~5.5V	T_A=-40~85℃	-3.50%	8	3.50%	MHz
		VDD=4.0~5.5V	T_A=25℃	-1.50%	16	1.50%	MHz
		VDD=2.5~5.5V	T_A=25℃	-2.00%	16	2.00%	MHz
		VDD=4.0~5.5V	T_A=-40~85℃	-2.50%	16	2.50%	MHz
		VDD=2.5~5.5V	T_A=-40~85℃	-3.50%	16	3.50%	MHz

19.9 EMC Characteristics

19.9.1 EFT Electrical Characteristics

Symbol	Parameter	Test condition	Grade
V_{EFTB}	Fast transient voltage burst limits to beapplied through 0.1uF(capacitance) on VDDand VSSpins to induce a functional disturbance	T_A = + 25°C, F_{SYS}=8MHz, conforms to IEC 61000-4-4	4

Note: Electrical fast transient burst (EFT) immunity performance is closely related to system design (including power supply structure, circuit design, layout and wiring, chip configuration, program structure, etc.). The EFT parameters in the above table are the results measured on the CMS internal test platform and are not applicable to all application environments. The test data is only for reference. All aspects of system design may affect the performance of EFT. In applications with high performance requirements of EFT, attention should be paid to avoiding interference sources affecting system operation as far as possible during design. It is recommended to analyze interference paths and optimize design to achieve the best anti-interference performance.

19.9.2 ESD Electrical Characteristics

Symbol	Parameter	Test condition	Grade
V_{ESD}	Electrostatic discharge (Human body model HBM)	T_A = + 25°C, JEDEC EIA/JESD22- A114	2
	Electrostatic discharge (Machine discharge mode MM)	T_A = + 25°C, JEDEC EIA/JESD22- A115	B

19.9.3 Latch-Up Electrical Characteristics

Symbol	Parameter	Test condition	Test type
LU	Static latch-up class	JEDEC STANDARD NO.78D NOVEMBER 2011	Class I (T_A = +25°C)

20. Instructions

20.1 Instructions Table

mnemonic		operation	instructions period	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
Increase/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	instructions period	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
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 one instructions	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 one instructions	1 or 2	NONE

20.2 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 (0AH) 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 (0AH) 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 save 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
        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
```

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

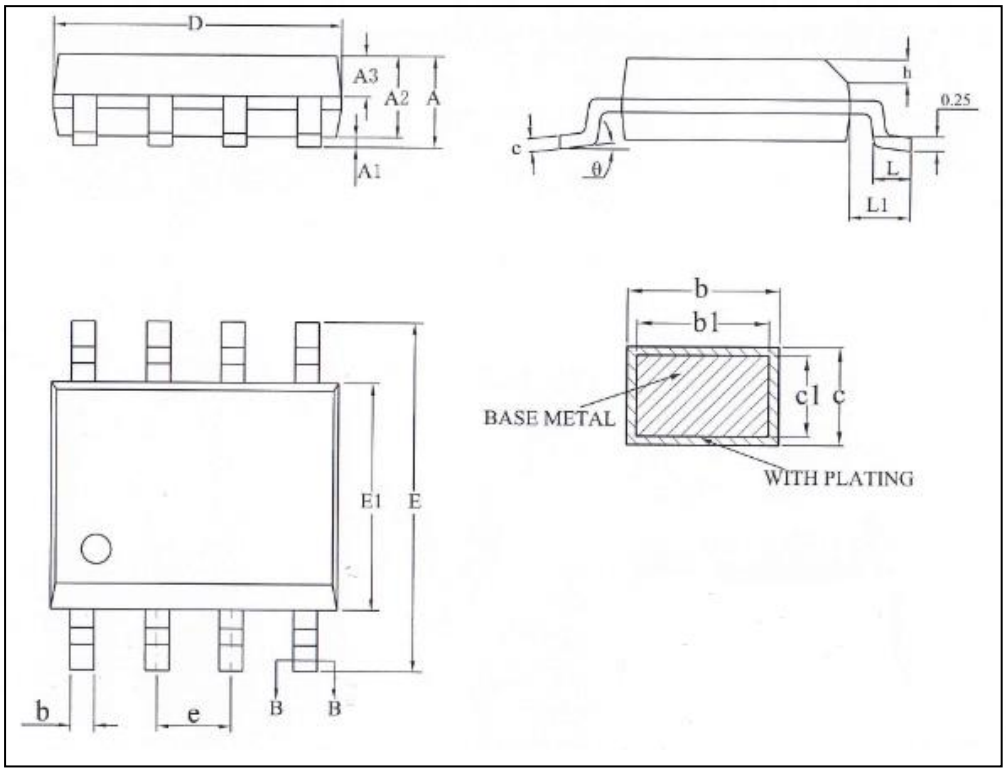
affected flag bit: Z

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
```

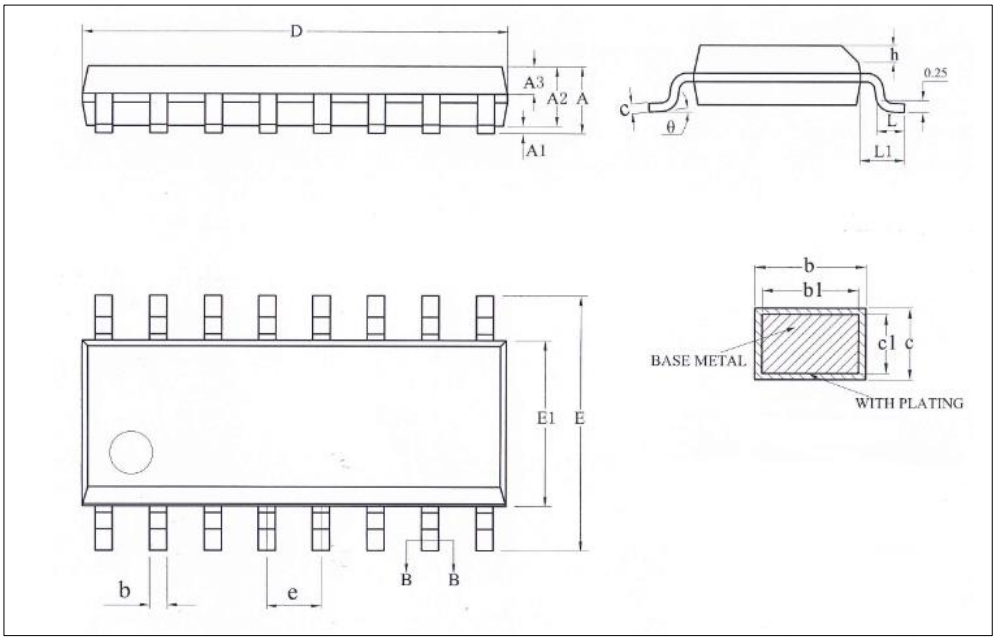
21. Packaging

21.1 SOP8



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.75
A1	0.10	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.20	-	0.24
c1	0.19	0.20	0.21
D	4.80	4.90	5.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
h	0.25	-	0.50
L	0.50	-	0.80
L1	1.05REF		
θ	0	-	8°

21.2 SOP16



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.75
A1	0.10	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.20	-	0.24
c1	0.19	0.20	0.21
D	9.80	9.90	10.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
h	0.25	-	0.50
L	0.50	-	0.80
L1	1.05REF		
θ	0	-	8°

22. Version Revision

Version number	time	Revised content
V1.0	Jan 2021	Original version
V1.1	Jun 2021	1) fix the description error about data memory registers 2) fix the description error in TMR2 and PR2
V1.2	Aug 2021	fix the description error about ADC reference voltage of internal LDO
V1.3	Sep 2021	Add the chapter of USART/IIC/SPI and its corresponding interrupt definition
V1.4	Sep 2021	Fix the description error about OPA registers
V1.5	Jan 2022	1) Increase the comparator mode selection bit in the OPA register 2) Revise IIC baud rate calculation formula 3) Add the description of porta level change interrupt 4) Revise ADC port configuration description 5) Revise the working voltage range of EEPROM 6) Revise the instructions in LVD operation 7) Revise that RCIF and TXIFin PIR1 register are read-only 8) Revise TXIF description in PIR1 register 9) Revise figure 16-3 and figure 16-4 of USART asynchronous transmission 10) Modify the program routine of writing EEDATA 11) Revise the description of receive interrupt in USART chapter 12) Revise the description of analog channel selection in ADCCON0 register 13) Add the temperature characteristic curve of LVR 14) Revise the error description of I2C TM master mode receiving sequence diagram 15) Delete the physical drawing in the package drawing 16) 16. Add EMC chapter and modify AC electrical characteristics table
V1.6	Feb 2022	1) Revise register PIR1: RCIF and TXIF are read-only 2) Correction of TXIF description in PIR1 register 3) Correct Figure 16-3 and Figure 16-4 of the USART asynchronous transmission
V1.7	Mar 2022	1) Revised the description of receiving interrupts in USART section 2) Revised description of analog channel selection in ADCON0 register 3) Revise the EEPROM section description 4) Add the LVR temperature characteristic curve
V1.8	Jul 2022	Add the EMC section and modified the AC electrical characteristics table
V1.9	Aug 2022	Adjust 19.1 Limit parameters
V2.0	Aug 2022	1) 2.1.2 Data Memory: revise T01E to be T0IE 2) 16.4 USART Baud Rate Generator (BRG): revise Fsys to be Fosc
V2.1.0	Apr 2023	1) Add clock block diagram 2) Revised the internal high-speed oscillation frequency to F_{HSI}, and revised the clock sources of other modules according to the clock block diagram 3) Revised sleep wake-up waiting time and ADC internal LDO reference voltage temperature characteristic table 4) Add SDIO function to RB2 and RB3 of pin diagram 5) Modify the System Configuration Register content in section 1.4 6) Update 19.9 EMC Characteristics 7) Add the option of clock instruction 2T in Section 1.1, 1.4 and 19.2