CMS79F623 User Manual

Enhanced 8-bit CMOS Microcontroller with Flash Memory

Rev. 1.3.0

Table of Contents

1. Product Description

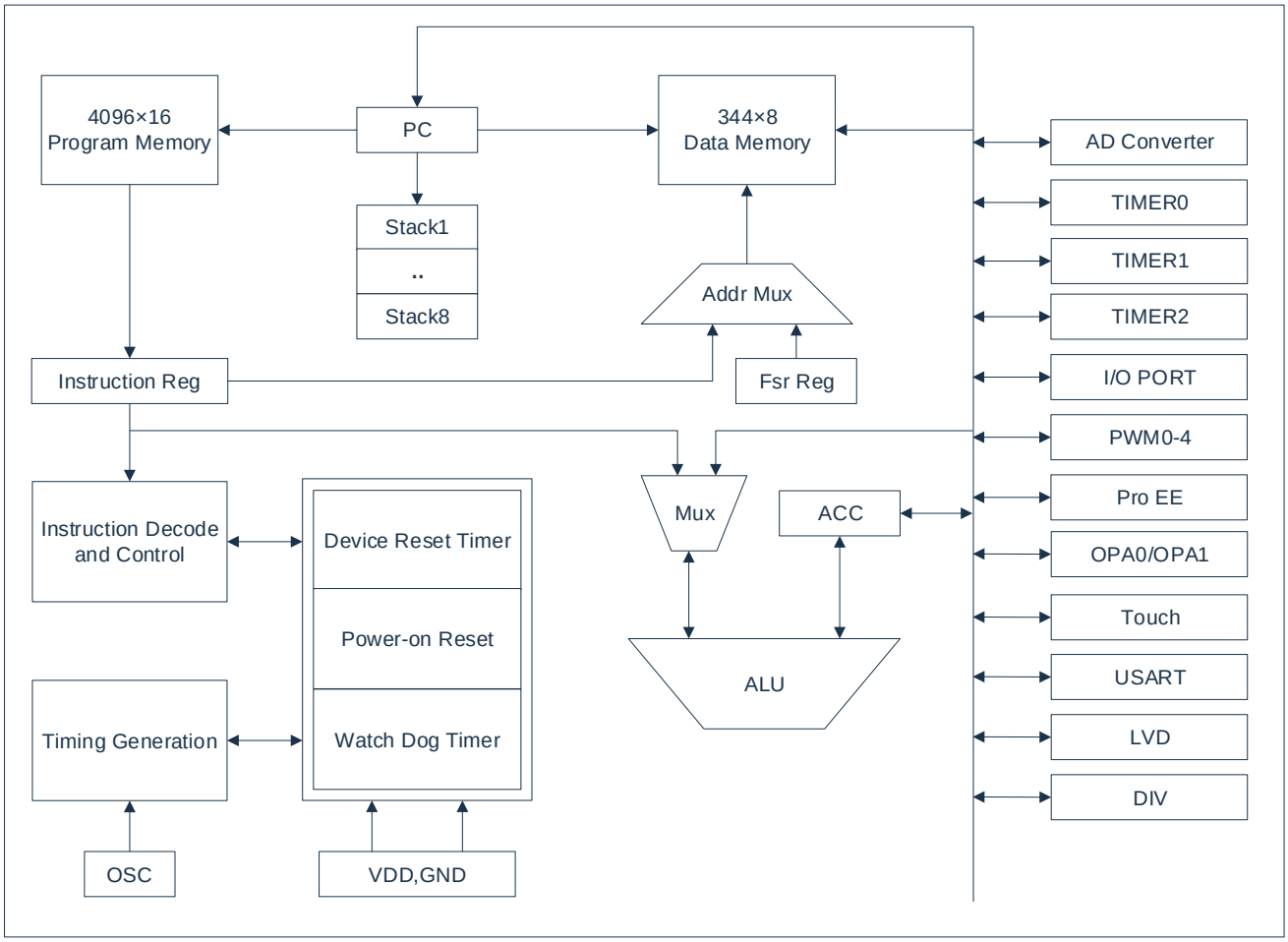
1.1 Features

- Memory
  - ROM: 4K×16Bit
  - Universal RAM: 344×8Bit
- 8-level stack buffer
- Short and clear instruction system (68 instructions)
- Instructions period (single instruction or double instructions)
- Look-up table function
- Built-in low voltage detection circuit
- Built-in WDT timer
- Interrupt sources
  - 3 timer interrupts
  - RA port interrupt on change
  - RB port interrupt on change
  - Other peripheral interrupts
- Timer
  - 8-bit timer TIMER0, TIMER2
  - 16-bit timer TIMER1
- Built-in hardware divider module
  - 24-digit dividend and 12-digit divisor
- Built-in LVD module
  - Supports 2.2V/2.4V/2.7V/3.0V/ 3.3V/3.7V/4.0V/4.3V
- Built-in 128-byte program EEPROM
  - Can be rewritten over 10,000 times
- Operating voltage: 2.6V—5.5V@16MHz
  1.8V—5.5V@8MHz
  Operating temperature: -40℃—85℃
- Two oscillation modes
  - Internal RC oscillation: design frequency of 8MHz/16MHz
  - External high-speed crystal oscillation: design frequency of 8MHz
- PWM mod with complementary outputs
  - 5-channel PWM, which can be configurated as 2 groups of complementary outputs
  - 4-channel PWM with shared period, independent duty cycle
  - 1-channel PWM with shared period, independent duty cycle
- Operational amplifier: 2 channels
  - Offset voltage<2mv
  - Can be used as a comparator
- Built-in touch key module
  - High immunity to interference
- Built-in 1-channel USART communication module
  - Supports synchronous master-slave mode and asynchronous full-duplex mode
  - Configurable in RB4/RB5 or RA6/RA7
- High-precision 12-bit ADC
  - Built-in high-precision 1.2V reference voltage
    ±1.5% @VDD=2.5V~5.5V T_A=25℃
    ±2% @VDD=2.5V~5.5V T_A=-40℃~85℃
  - Optional internal reference sources: 2V/2.4V

Product specification

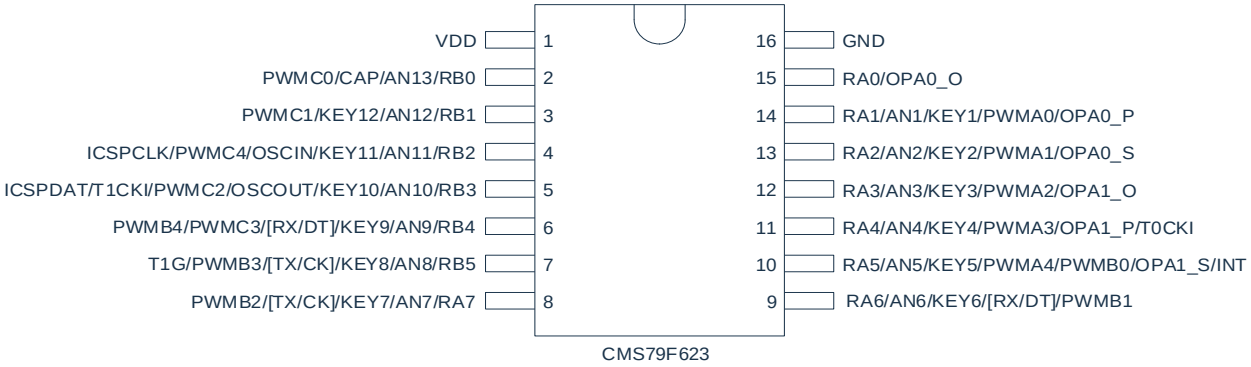
PRODUCT	ROM	RAM	Pro EE	I/O	PWM	OPA	ADC	Touch	PACKAGE
CMS79F623	4Kx16	344x8	128x8	14	5	2	12Bitx13	12	SOP16

1.2 System structure diagram



1.3 Top view

1.3.1 CMS79F623



Note: The chip has only one channel USART function, TX/CK, RX/DT pins can be assigned through the system configuration register.

CMS79F623 pin description:

Pin name	IO type	Pin description
VDD, GND	P	Supply voltage input pin, ground pin
OSCIN/OSCOUT	I/O	Crystal input/output pins
RA0-RA7	I/O	Programmable as input pin, push-pull output pin, pull-up resistor function, pull-down resistor function, interrupt on change function
RB0-RB5	I/O	Programmable as input pin, push-pull output pin, pull-up resistor function, pull-down resistor function, interrupt on change function
ICSPCLK	I	Programming clock input pin
ICSPDAT	I/O	Programming data input/output pins
AN1-AN13	I	12-bit ADC input pin
PWMx0-PWMx4	O	PWM0-4 output function
OPAx_P	I	Positive op-amp input pin
OPAx_S	I	Negative op-amp input pin
OPAx_O	O	Op-amp output pin
TX/CK	O	Asynchronous serial transmit pin/Synchronous serial clock input/output pin (can be configured on different IO ports)
RX/DT	I	Serial receive pin/synchronous serial data input/output pin (can be configured on different IO ports)
INT	I	External interrupt input pin
T0CKI	I	TIMER0 external clock input pin
T1CKI	I	TIMER1 external clock input pin
T1G	I	TIMER1 external gate control input pin
KEY1-KEY12	I	Touch key detection port
CAP	I	Capacitive touch key pin

1.4 System configuration register

The System Configuration Register (CONFIG) is a ROM option for the initial condition of the MCU. It can only be written by the CMS writer and cannot be accessed or manipulated by the user. It contains the following:

1. OSC (oscillation mode selection)
   - INTRC Internal RC oscillation
   - XT External crystal oscillation
2. INTRC_SEL (internal oscillation frequency selection)
   - INTRC8M F_{HSI} selects internal 8MHz RC oscillation
   - INTRC16M F_{HSI} selects internal 16MHz RC oscillation
3. WDT (watchdog selection)
   - ENABLE Turn on the watchdog timer
   - DISABLE Turn off the watchdog timer
4. PROTECT (encrypted)
   - DISABLE FLASH code is not encrypted
   - ENABLE FLASH code is encrypted, and the value read out by the burn-in emulator will be uncertain after encryption.
5. LVR_SEL (low-voltage detect voltage selection)
   - 1.8V To select this reset voltage point, F_{HSI} needs to select 8MHz.
   - 2.0V
   - 2.6V
6. PWM_SEL (PWM output port selection)
   - Group A PWM0-4=RA1, RA2, RA3, RA4, RA5
   - Group B PWM0-4=RA5, RA6, RA7, RB5, RB4
   - Group C PWM0-4=RB0, RB1, RB3, RB4, RB2
7. USART_SEL (USART output port selection)
   - RB4/RB5
   - RA6/RA7
8. ICSPPORT_SEL (emulation port function selection)
   - ICSP ICSPCLK and DAT ports remain emulated and all functions are not available.
   - NORMAL ICSPCLK and DAT ports are general function ports

1.5 Online serial programming

This allows serial programming of the microcontroller in the final application circuit. Programming can be done simply with the following 4 wires:

- 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.

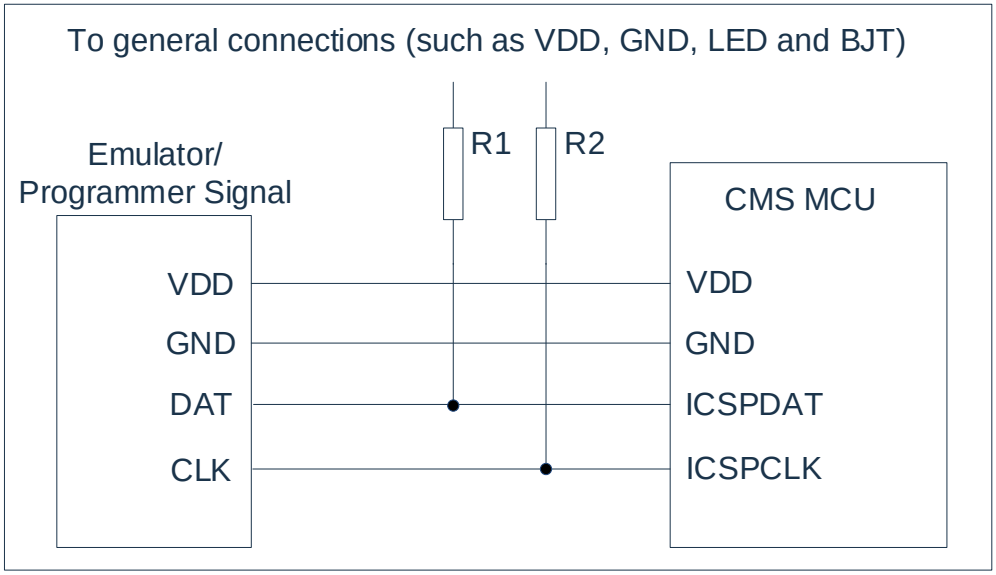


Figure 1-1: Typical connection for online serial programming

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

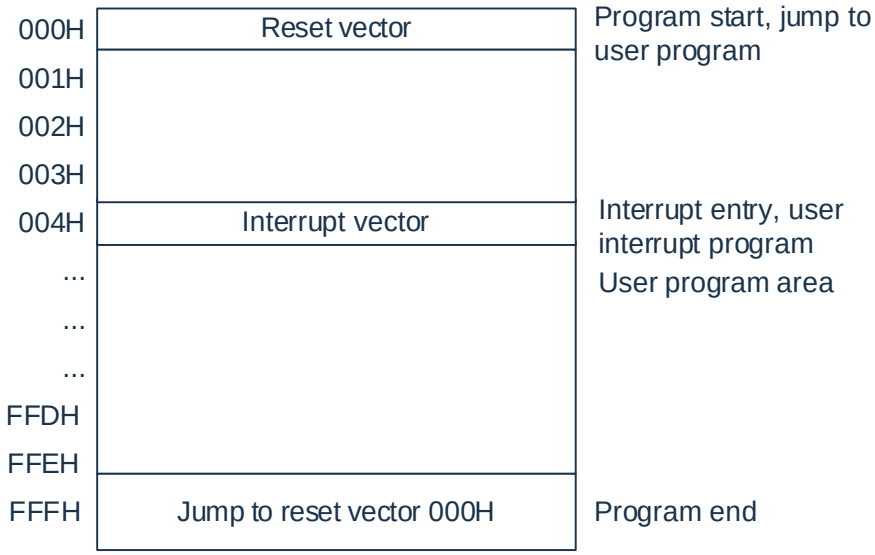
2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program memory

CMS79F623 program memory space

MTP:4K



2.1.1.1 Reset vector (0000H)

MCU has a 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 flag bits from the STATUS register can determine 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. Users will determine which interrupt to execute according to the bit of the interrupt request flag bit register. The following program illustrates 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 users need 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. Since the addition of PCL and ACC is the new value of PCL, multi-address jump is then achieved through adding different values of ACC to PCL. If the value of ACC is n, then PCL+ACC represents 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 to LOOP1
00FEH:          JP          LOOP2           ;ACC=1, jump to LOOP2
00FFH:          JP          LOOP3           ;ACC=2, jump to LOOP3
0100H:          JP          LOOP4           ;ACC=3, jump to 0000H address
0101H:          JP          LOOP5           ;ACC=4, jump to 0001H address
0102H:          JP          LOOP6           ;ACC=5, jump to 0002H 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

CMS79F623 data memory list

	address		address		address		address	
	INDF	00H	INDF	80H	INDF	100H	INDF	180H
	TMR0	01H	OPTION_REG	81H	TMR0	101H	OPTION_REG	181H
	PCL	02H	PCL	82H	PCL	102H	PCL	182H
	STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H
	FSR	04H	FSR	84H	FSR	104H	FSR	184H
	PORTA	05H	TRISA	85H	WDTCON	105H	----	185H
	PORTB	06H	TRISB	86H	PORTB	106H	TRISB	186H
	WPDA	07H	IOCA	87H	PWMCON0	107H	----	187H
	WPDB	08H	----	88H	PWMCON1	108H	ANSEL	188H
	----	09H	----	89H	PWMCON2	109H	ANSELH	189H
	PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
	INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
	PIR1	0CH	PIE1	8CH	EEDAT	10CH	EECON1	18CH
	PIR2	0DH	PIE2	8DH	EEADR	10DH	EECON2	18DH
	TMR1L	0EH	----	8EH	EEDATH	10EH	WPUA	18EH
	TMR1H	0FH	OSCCON	8FH	EEADRH	10FH	PWMTL	18FH
	T1CON	10H	OSCTUNE	90H	TABLE_SPH	110H	PWMTH	190H
	TMR2	11H	----	91H	TABLE_SPL	111H	PWMT4L	191H
	T2CON	12H	PR2	92H	TABLE_DATAH	112H	----	192H
	DIVS1	13H	PWM01DT	93H	----	113H	PWMD0L	193H
	DIVS0	14H	PWM23DT	94H	----	114H	PWMD1L	194H
	DIVE2/DIVQ2	15H	WPUB	95H	----	115H	PWMD2L	195H
	DIVE1/DIVQ1	16H	IOCB	96H	----	116H	PWMD3L	196H
	DIVE0/DIVQ0	17H	LVDCON	97H	----	117H	PWMD4L	197H
	RCSTA	18H	TXSTA	98H	----	118H		198H
	TXREG	19H	SPBREG	99H	----	119H		199H
	RCREG	1AH	OPA0CON	9AH	----	11AH		19AH
	DIVCON	1BH	OPA0ADJ	9BH	----	11BH		19BH
	----	1CH	OPA1CON	9CH	PWMD01H	11CH		19CH
	----	1DH	OPA1ADJ	9DH	PWMD23H	11DH		19DH
	ADRESH	1EH	ADRESL	9EH	----	11EH		19EH
	ADCON0	1FH	ADCON1	9FH	----	11FH		19FH
	Universal register 96 bytes	20H	Universal register 80 bytes	A0H	Universal register 80 bytes	120H	Universal register 80 bytes	1A0H
		6FH		EFH		16FH		1EFH
		70H	Fast memory space 70H-7FH	F0H	Fast memory space 70H-7FH	170H	Fast memory space 70H-7FH	1F0H
		--		--		--		--
		7FH		FFH		17FH		1FFH
	BANK0		BANK1		BANK2		BANK3	

Data memory consists of 512×8 bits. It can be divided into two areas: special function register and universal data memory. Most of data memory are readable and writable, only some data memory is read-only. Special register addresses are from 00H-1FH, 80H-9FH, 100H-11FH, 180H-197H.

CMS79F623 special function register summary 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 bytes of program counter								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	Indirect data memory address pointer								xxxxxxxx
05H	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxxxxxx
06H	PORTB	----	----	RB5	RB4	RB3	RB2	RB1	RB0	--xxxxxx
07H	WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0	00000000
08H	WPDB	----	----	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0	--000000
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	RAIF	ADIF	RCIF	TXIF	EEIF	PWMIF	TMR2IF	TMR1IF	00000000
0DH	PIR2	----	----	----	----	----	----	----	LVDIF	-------0
0EH	TMR1L	Data register for the low byte of the 16-bit TIMER1 register								xxxxxxxx
0FH	TMR1H	Data register for the high byte of the 16-bit TIMER1 register								xxxxxxxx
10H	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	----	T1SYNC	TMR1CS	TMR1ON	0000-000
11H	TMR2	TIMER2 module register								00000000
12H	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-0000000
13H	DIVS1	----	----	----	----	Higher 4-digit divisor				----0000
14H	DIVS0	Lower 8-digit divisor								00000000
15H	DIVE2	Dividend or Quotient BIT<23:16>								00000000
16H	DIVE1	Dividend or Quotient BIT15:8>								00000000
17H	DIVE0	Dividend or Quotient BIT<7:0>								00000000
18H	RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D	00000000
19H	TXREG	USART Transmit Data Register								00000000
1AH	RCREG	USART Receive Data Register								00000000
1BH	DIVCON	DIVEN	CAL_END	----	----	----	----	----	DIV_CLK	01-----0
1EH	ADRESH	High bytes of the A/D result register								xxxxxxxx
1FH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	00000000

CMS79F623 special function register summary 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	Low bytes of program counter (PC)								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
84H	FSR	Indirect data memory address pointer								xxxxxxxx
85H	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	11111111
86H	TRISB	----	----	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	--111111
87H	IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0	00000000
8AH	PCLATH	----	----	----	----	Write buffer for the high 4 bits of the program counter				-----0000
8BH	INTCON	GIE	PEIE	T01E	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	PIE1	RAIE	ADIE	RCIE	TXIE	EEIE	PWMIE	TMR2IE	TMR1IE	00000000
8DH	PIE2	----	----	----	----	----	----	----	LVDIE	-------0
8FH	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	SCS	-110---0
90H	OSCTUNE	----	----	----	TUN4	TUN3	TUN2	TUN1	TUN0	---00000
92H	PR2	TIMER2 period register								11111111
93H	PWM01DT	----	----	PWM01 deadband delay time						--00000
94H	PWM23DT	----	----	PWM23 deadband delay time						--00000
95H	WPUB	----	----	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	--000000
96H	IOCB	----	----	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0	--000000
97H	LVDCON	LVD_RES	—	—	—	LVD_SEL<2:0>			LVDEN	x--0000
98H	TXSTA	CSRC	TX9EN	TXEN	SYNC	SCKP	----	TRMT	TX9D	00000010
99H	SPBRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0	00000000
9AH	OPA0CON	OPA0EN	OPA0O	OPA0_CMP	OPA0_ADC	OPA0_FW	OPA0_BG	----	----	000000--
9BH	OPA0ADJ	OPA0OUT	OPA0COFM	OPA0CRS	OPA0ADJ<4:0>					000xxxxx
9CH	OPA1CON	OPA1EN	OPA1O	OPA1_CMP	OPA1_ADC	OPA1_FW	OPA1_BG	----	----	000000--
9DH	OPA1ADJ	OPA1OUT	OPA1COFM	OPA1CRS	OPA1ADJ<4:0>					000xxxxx
9EH	ADRESL	Low bytes of the ADC result register								xxxxxxxx
9FH	ADCON1	ADFM	----	----	----	----	LDO_EN	----	LDO_SEL	0----0-0

CMS79F623 special function register summary Bank2

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
100H	INDF	Look-up for this unit will use FSR, not physical register.								xxxxxxxx
101H	TMR0	TIMER0 module register								xxxxxxxx
102H	PCL	Low bytes of program counter (PC)								00000000
103H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
104H	FSR	Indirect data memory address pointer								xxxxxxxx
105H	WDTCON	----	----	----	----	----	----	----	SWDTEN	-------0
106H	PORTB	----	----	RB5	RB4	RB3	RB2	RB1	RB0	--xxxxxx
107H	PWMCON0	CLKDIV<2:0>			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN	00000000
108H	PWMCON1	----	----	PWM2DTEN	PWM0DTEN	----	----	DT_DIV<1:0>		--00--00
109H	PWMCON2	----	----	----	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR	----00000
10AH	PCLATH	----	----	---	---	Write buffer for the high 4 bits of the program counter				----0000
10BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
10CH	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx
10DH	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	00000000
10EH	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	xxxxxxxx
10FH	EEADRH	----	----	----	----	EEADRH3	EEADRH2	EEADRH1	EEADRH0	----0000
110H	TABLE_SPH	----	----	----	----	Table high 4-bit pointer				----xxxx
111H	TABLE_SPL	Table low pointer								xxxxxxxx
112H	TABLE_DATAH	Table high pointer								xxxxxxxx
11CH	PWMD01H	----	----	PWMD1<9:8>		----	----	PWMD0<9:8>		--00--00
11DH	PWMD23H	----	----	PWMD3<9:8>		----	----	PWMD2<9:8>		--00--00

CMS79F623 special function register summary Bank3

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
180H	INDF	Look-up for this unit will use FSR, not physical register.								xxxxxxxx
181H	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	11111011
182H	PCL	Low bytes of program counter (PC)								00000000
183H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
184H	FSR	Indirect data memory address pointer								xxxxxxxx
186H	TRISB	----	----	TRISB5	TRISB4	TRISB73	TRISB2	TRISB1	TRISB0	--111111
188H	ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	----	0000000-
189H	ANSELH	----	----	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8	--000000
18AH	PCLATH	----	----	----	----	Write buffer for the high 4 bits of the program counter				----0000
18BH	INTCON	GIE	PEIE	T01E	INTE	RBIE	T01F	INTF	RBIF	00000000
18CH	EECON1	EEPGD	----	----	----	WRERR	WREN	WR	RD	0---x000
18DH	EECON2	EEPROM control register 2 (not a physical register)								--------
18EH	WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	00000000
18FH	PWMTL	PWM0-3 period low register								00000000
190H	PWMTH	PWM period high register								00000000
191H	PWMT4L	PWM4 period low register								00000000
193H	PWMD0L	PWM0 duty cycle low register								00000000
194H	PWMD1L	PWM1 duty cycle low register								00000000
195H	PWMD2L	PWM2 duty cycle low register								00000000
196H	PWMD3L	PWM3 duty cycle low register								00000000
197H	PWMD4L	PWM4 duty cycle low register								00000000

2.2 Addressing mode

2.2.1 Direct addressing

It operates the RAM through the operation register (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 the immediate value 12H to ACC

```
LDIA    12H
```

2.2.3 Indirect addressing

Data memory can be addressed directly or indirectly. Direct addressing can be achieved through INDF register, INDF is not a physical register. When INDF is accessed, it is addressed according to the value in the FSR register (lower 8 bits) and the IRP bit (bit 9) of the STATUS register, and points to the register at that address. Therefore, after setting the FSR register and the IRP bit of STATUS register, INDF register can be regarded as a target register. Read INDF (FSR=0) indirectly will produce 00H. Write to the INDF register indirectly will cause an empty operation. The following example shows how indirect addressing works.

Example: application of FSR and INDF

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

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

```
        LDIA    1FH
        LD      FSR,A           ;point 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 a subroutine call or interrupt occurs, values in program counter (PC) will be transferred to stack buffer. When return from interrupt or return from subroutine, values are transferred back to PC. The following diagram illustrates its working principle.

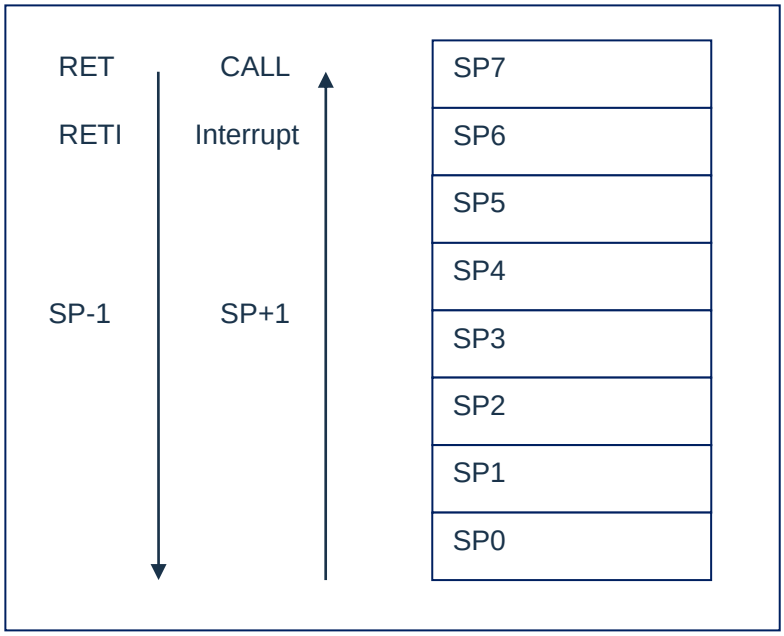


Figure 2-1: Working principle of stack buffer

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

Note: Stack buffer has only 8 levels, if the stack is full and an interrupt happens which is non-maskable, then only the flag bit of the interrupt will be logged. 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 the interrupt. Similarly, when stack is full and subroutine 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. Therefore, users should pay attention to this point when writing programs to avoid program loops.

2.4 Accumulator (ACC)

2.4.1 Overivew

ALU is an 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 calculation result.

ACC register is an 8-bit register to store the ALU calculation result. 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 provided instructions.

2.4.2 ACC application

Example: use ACC for data transferring

```
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 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). 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
R/W	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>:	Memory selection bit
	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 instruction or STOP instruction
	0=	WDT time out
Bit3	PD:	Power down
	1=	Power on or CLRWDT instruction
	0=	STOP instruction
Bit2	Z:	Bit result
	1=	Result is 0
	0=	Result is not 0
Bit1	DC:	Half carry bits/borrow bit
	1=	Carry happens to higher bits from the lower 4 bits of the result
	0=	No carry happens to higher bits from the lower 4 bits of the result
Bit0	C:	Carry/borrow bit
	1=	Carry happens at the highest bit or no borrow happens
	0=	No carry happens at the highest bit or borrow happens

TO and PD flag bits can reflect the reason for chip reset. 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.

Pre-scaler OPTION_REG(81H)

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

Bit7 Unused

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 instruction period clock (F_{SYS}/4)
1= Transition edge on the T0CKI pin

Bit4 T0SE: Edge selection bit for TIMER0 clock source
0= Increase when T0CKI pin signal transited from low to high
1= Increase when T0CKI pin signal transited from high to low

Bit3 PSA: Pre-scaler allocation bit
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 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

Pre-scaler register is an 8-bit counter. When surveil on register WDT, it is a postscaler; 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, the CLRWDT instruction will clear pre-scaler and WDT timer.

When used for TIMER0, all instruction related to writing TIMER0 (such as: CLR TMR0, SETB TMR0,1) will clear the 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          ;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: To enable TIMER0 to acquire a 1:1 prescaler ratio configuration, the prescaler can be assigned to the WDT by setting the PSA bit of the option register to 1.

2.7 Program counter (PC)

Program counter (PC) controls the instruction sequence in program memory FLASH, it can address in the whole range of FLASH. After obtaining the instruction code, PC will increase by 1 and point to the address of the next instruction code. When executing jump, condition jump, passing value to PCL, subroutine call, initializing reset, interrupt, interrupt return, subroutine 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 a condition jump instruction and the condition is met, the next instruction to be read during current instruction execution will be discarded and an empty instruction period will be inserted. After this, the correct instruction can be obtained. If not, the next instruction is executed in sequence.

Program counter (PC) is 12-bit width, users can access lower 8 bits by PCL (02H). The higher 4 bits cannot be accessed. It can hold address for 4K×16Bit program. Assigning a value to PCL results in a short jump to the 256 addresses of the current page.

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

The following PC values are given for several special cases.

reset	PC=0000;
interrupt	PC=0004 (original PC+1will be add to stack automatically);
CALL	PC=program specified address (original PC+1will be add to stack automatically);
RET, RETI, RET i	PC=value coming out from stack;
PCL operation	PC[11:8] unchanged, PC[7:0]=user defined value;
JP	PC=program specified 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 components. Even the main clock of the chip stops working, WDT can still timing. WDT timing overflow will generate a reset.

2.8.1 WDT period

WDT and TIMER0 share an 8-bit pre-scaler. After all resets, default overflow period of WDT is 128ms. The way to calculate WDT overflow is 16ms*frequency division coefficient. 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 parameters.

“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 circumstances, the WDT should be cleared by the "CLRWDT" instruction before it overflows to prevent a reset. If program is out of control for some reason such that “CLRWDT” instruction is not able to execute before overflow, WDT overflow will then generate a reset to make sure the system restarts. If a reset is generated by the WDT overflow, then ‘TO’ bit of STATUS will be cleared to 0. Users can judge whether the reset is caused by WDT overflow according to this.

Note:

1. If WDT is used, ‘CLRWDT’ instruction 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 be operated normally.
2. It is not allowed to clear WDT during interrupt so that the main program ‘run away’ can be detected.
3. The program should have one WDT clearing operation in the main program, and try not to clear the WDT in multiple branches, this architecture can maximize the protection function of the watchdog counter.
4. The overflow time of the watchdog counter varies from chip to chip, so when setting the clear WDT time, there should be a greater redundancy with the WDT overflow time to avoid unnecessary WDT reset.

2.8.2 Watchdog timer control register WDTCON

Watchdog timer control register WDTCON(105H)

105H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	---	---	---	---	---	---	---	SWDTEN
R/W	---	---	---	---	---	---	---	R/W
Reset value	---	---	---	---	---	---	---	0

Bit7~Bit1 Unused, read 0

Bit0 SWDTEN: Software enable or disable watchdog timer bit

1= Enable WDT

0= Disable WDT (reset value)

Note: If the WDT configuration bit in CONFIG = 1, WDT is always enabled, regardless of the state of the SWDTEN control bit. If the WDT configuration bit in CONFIG = 0, the SWDTEN control bit can be used to enable or disable WDT.

3. System Clock

3.1 Overview

After the clock signal is input from the OSCIN pin (or generated by internal oscillation), 4 non-overlapping quadrature clock signals are generated on-chip, called Q1, Q2, Q3, and Q4. Each Q1 inside the IC increments the program counter (PC) by one, and Q4 removes the instruction from the program memory cell and locks it into the instruction register. The removed instruction is decoded and executed between the next Q1 and Q4, which means that it takes 4 clock cycles to execute an instruction. The following figure represents the clock versus instruction cycle execution timing diagram.

An instruction cycle contains four Q-cycles, and the instruction execution and fetching are in pipeline structure, fetching finger occupies one instruction cycle, while decoding and execution occupy another instruction cycle, but due to the pipeline structure, from a macro point of view, the effective execution time of each instruction is one instruction cycle. If an instruction causes the program counter address to change (e.g. JP) then the prefetched instruction opcode is invalid and it takes two instruction cycles to complete the instruction, which is the reason why all instructions operating on the PC take up two clock cycles.

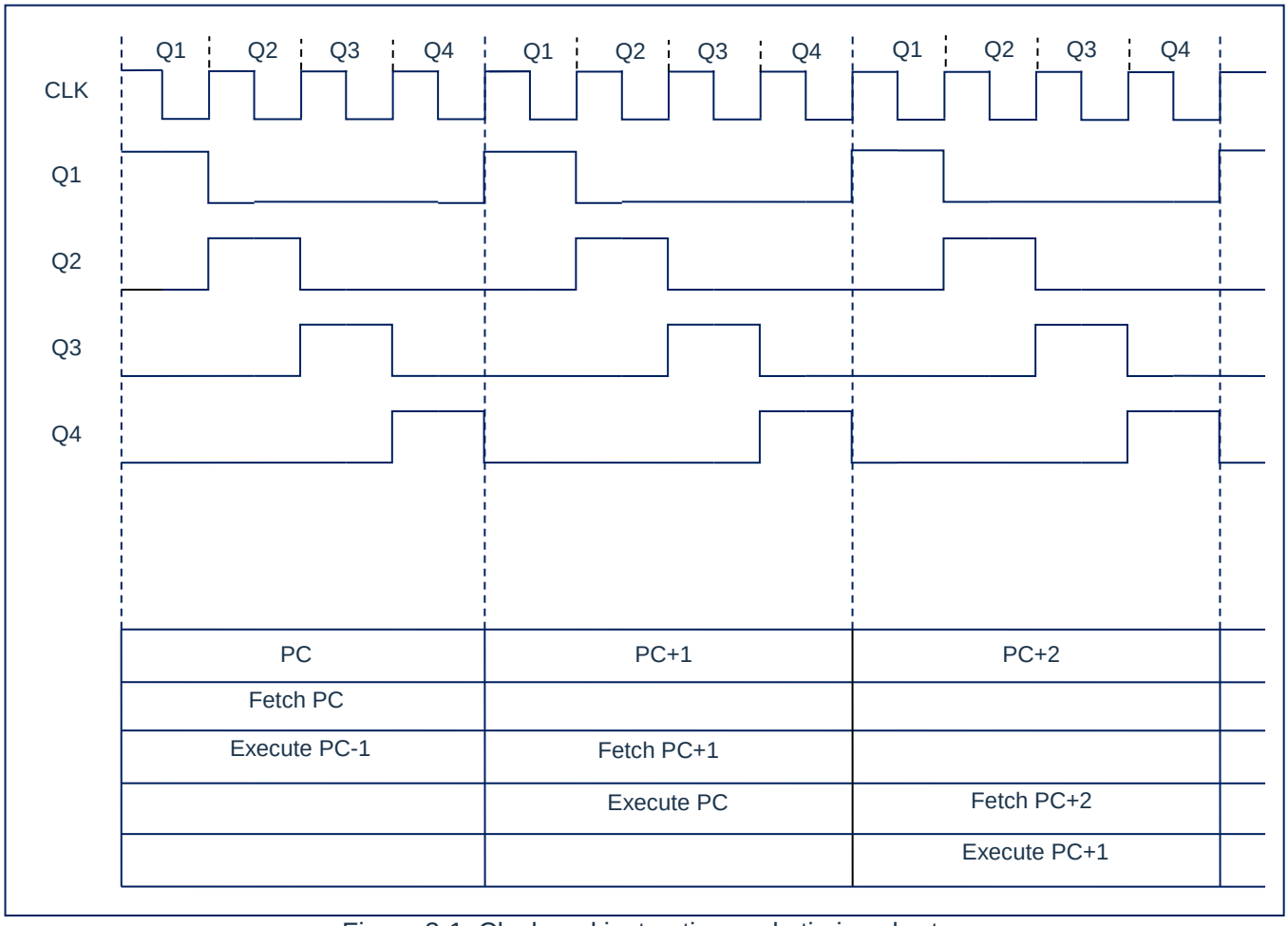


Figure 3-1: Clock and instruction cycle timing chart

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

The chip has two types of oscillation, internal RC oscillation and external XT oscillation.

3.2.1 Internal RC oscillation

The default oscillation mode of the chip is internal RC oscillation, and its oscillation frequency is 8MHz or 16M, which can be set through the OSCCON register.

3.2.2 External XT oscillation

When burning, the OSC in the CONFIG option will be selected as XT, the chip will work in the external XT oscillation mode, at this time the internal RC oscillation stops working, and OSCIN and OSCOUT will be used as the oscillation port.

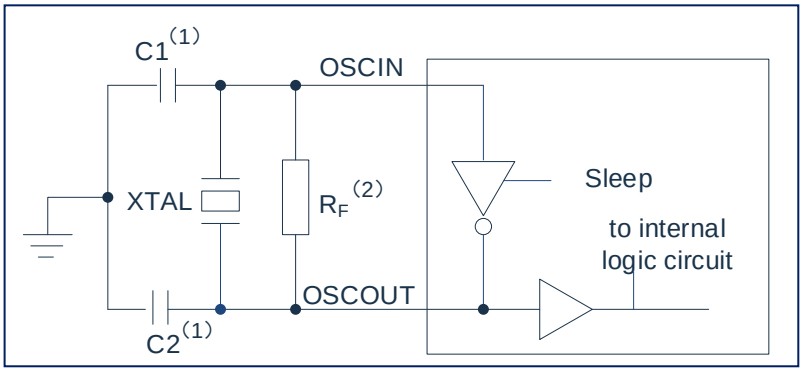


Figure 3-2: Typical XT oscillation method

Suggest parameters:

Type	Frequency	Suggest value R_F	Suggest value C1～C2
XT	455kHz	1MΩ	100pF～470pF
XT	2MHz	1MΩ	10pF～47pF
XT	4MHz	1MΩ	10pF～47pF
XT	8MHz	1MΩ	10pF～47pF

3.3 Reset time

Reset Time is the time from the chip reset to the chip oscillation stabilization, its design value is about 18ms.

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

3.4 Oscillator control register

The Oscillator Control (OSCCON) register controls the system clock and frequency selection, and the Oscillator Tuning Register, OSCTUNE, enables software tuning of the internal oscillation frequency.

Oscillator control register OSCCON(8FH)

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

Bit7 Unused, 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 (default)

101= $F_{SYS} = F_{HSI}$ /4

100= $F_{SYS} = F_{HSI}$ /8

011= $F_{SYS} = F_{HSI}$ /16

010= $F_{SYS} = F_{HSI}$ /32

001= $F_{SYS} = F_{HSI}$ /64

000= F_{SYS} = 32kHz (LFINTOSC)

Bit3~Bit1 Unused

Bit0 SCS: System clock select bit

1= The internal oscillator is used as system clock

0= The clock source is defined by CONFIG

Note: F_{HSI} is the internal high-speed oscillator frequency, selectable from 8MHz or 16MHz. F_{SYS} is the system operating frequency.

3.5 Clock block diagram

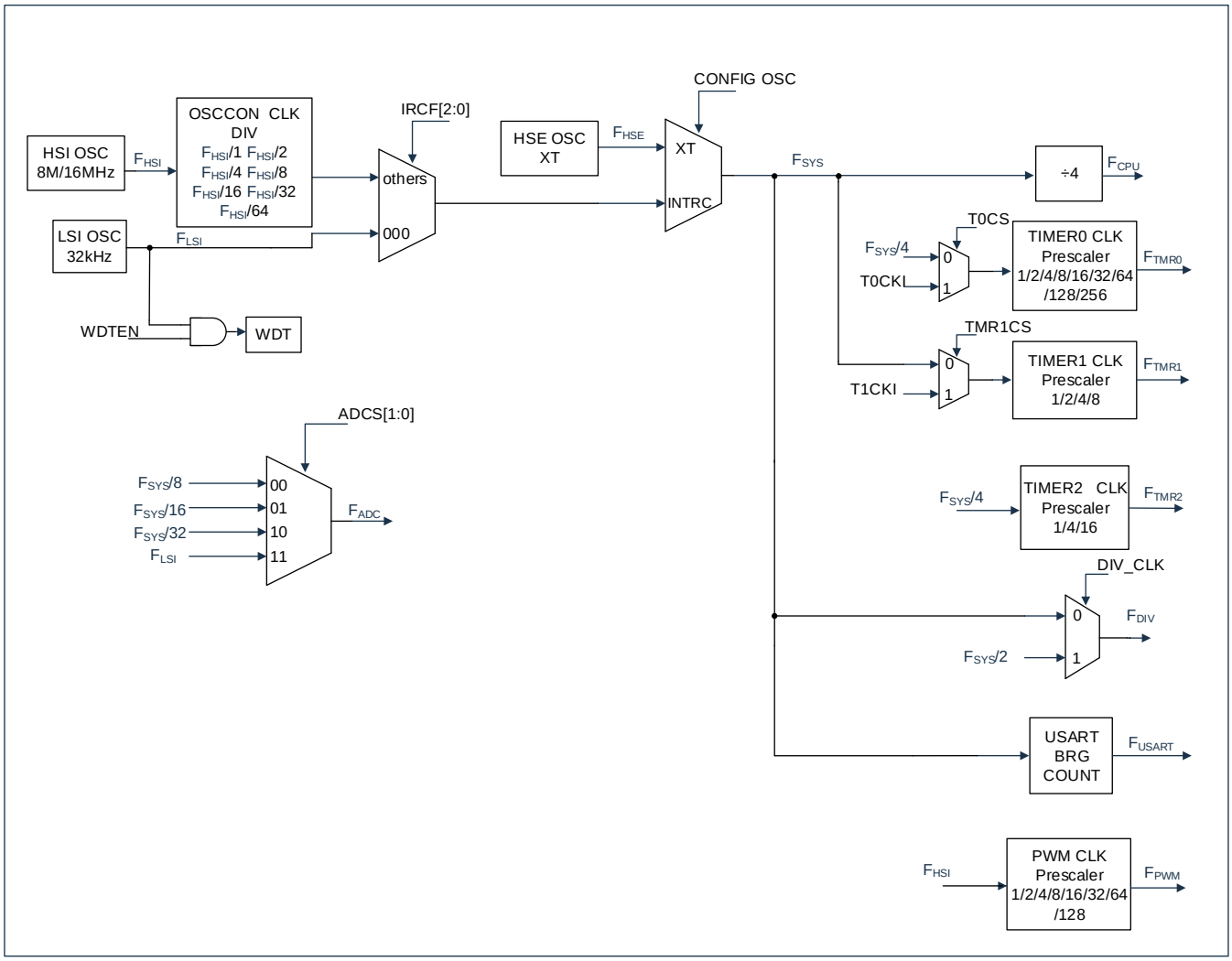


Figure 3-2: Block diagram of clock

4. Reset

The chip can be reset in the following three ways.

- Power on reset;
- Low voltage reset;
- Watchdog overflow reset during normal operation.

When any of the above reset occurs, all system registers will be restored to their default state, the program will stop running, and the program counter (PC) will be cleared to zero. At the same time, the program will start running from reset vector 0000H after the reset. Users can control the program running path according to the PD and TO status.

Any kind of reset situation requires a certain response time, and the system provides a completed reset process to ensure that the reset action is carried out smoothly.

4.1 Power on reset

Power-on reset is closely related to LVR operation. The process of system power-on is in the form of a gradually rising curve and takes some time to reach the normal level value. The normal timing of the power-on reset is given below:

- Power-on: the system detects a rise in the supply voltage and waits for it to stabilize.
- System initialization: all system registers are set to their initial values.
- Oscillator start: the oscillator starts to supply the system clock.
- Program execution: the power-on ends and the program start to run.

4.2 Power off reset

4.2.1 Overview

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.

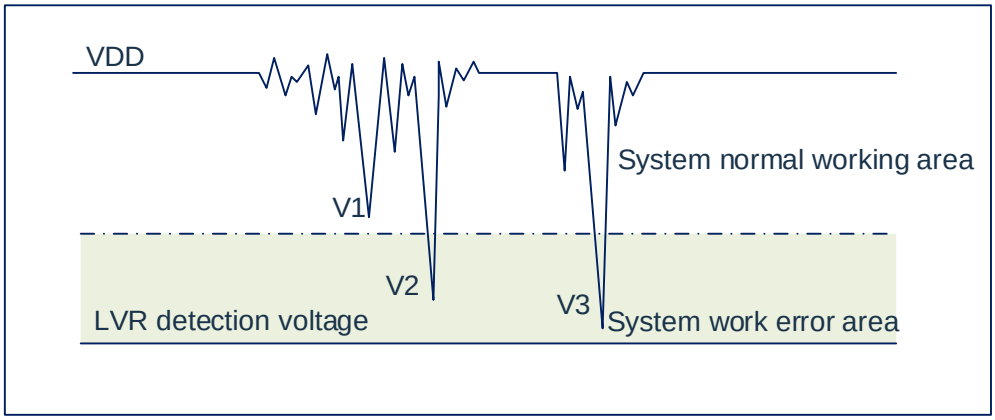


Figure 4-1: Power off reset

The diagram above is a typical power-off reset case. In the diagram, VDD is severely disturbed and the voltage value drops to a very low value. The system works normally in the area above the dotted line; in the area below the dotted line, the system enters an unknown operating state, and this area is called the dead zone. When VDD drops to V1, the system is still in the normal state; when VDD drops to V2 and V3, the system enters the dead zone, and it is easy to cause errors.

The system may enter a dead zone in the following cases:

- In DC applications:
  - Battery power is generally used in DC applications. When the battery voltage is too low or when the microcontroller drives the load, the system voltage may drop and enter the dead zone. At this time, the power supply will not drop further to the LVD detection voltage, so the system is maintained in the dead zone.
- In AC application:
  - When the system is powered by AC, the DC voltage value is affected by the noise in the AC power supply. When the external is over-loaded, such as when driving a motor, the interference generated by the load action also affects the DC power supply. if the VDD drops below the minimum operating voltage due to the interference, the system will likely enter an unstable operation state.
  - In AC application, the system has a long power up and down time. Among them, the power-on timing protection makes the system power up normally, but the power-off process is similar to the situation in DC applications, where the VDD voltage tends to enter the dead zone during the slow drop after the AC power is turned off.

As shown above, the system normal operating voltage is generally higher than the system reset voltage, while the reset voltage is determined by the low voltage detection (LVR) level. When the system execution speed increases, the system minimum operating voltage also increases accordingly. However, as the system

reset voltage is fixed, there will be a voltage region between the system minimum operating voltage and the system reset voltage where the system cannot work normally and will not reset. This area is known as the dead zone.

4.2.2 Improvements for power off reset

Several suggestions to improve the system power-off reset performance:

- Select a higher LVR voltage, which contributes to a more reliable reset.
- Turn on the watchdog timer.
- Reduce the operating frequency of the system.
- Increase the voltage drop slope.

Watchdog timer

The watchdog timer is used to ensure the normal operation of the program. When the system enters the dead zone or the program runs with errors, the watchdog timer will overflow and the system will be reset.

Reduce the operating speed of the system

The faster the operating frequency of the system, the higher the minimum operating voltage of the system. Therefore, by increasing the range of the operating dead zone and reducing the operating speed of the system, the minimum operating voltage can be reduced and the chance of entering the dead zone can be effectively reduced.

Increase the voltage drop slope

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

The watchdog reset is a protect configuration for the system. In the normal state, the watchdog timer is cleared to zero by the program. If something goes wrong, the system is in an unknown state and the watchdog timer overflows, at which point the system resets. After the watchdog reset, the system reboots into the normal state.

The timing of the watchdog reset is as follows:

- Watchdog timer status: the system detects whether the watchdog timer overflows, and if it does, the system resets.
- Initialization: all system registers are set to their default state.
- Oscillator start: the oscillator starts to provide the system clock.
- Program: The reset ends and the program starts running.

For application of watchdog timer, please refer to Section 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).

In 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

The device can be woken from sleep by any of the following events.

1. Watchdog timer wakeup (WDT forced enable).
2. PORTA interrupt on change
3. PORTB interrupt on change or peripheral interrupt.

The two events described above are considered to be a continuation of program execution. The TO and PD bits in the STATUS register are used to determine the cause of device reset. The PD bit is set to 1 at power-on and cleared when the STOP instruction is executed. The TO bit is cleared when a WDT awaken occurs.

When the STOP instruction is executed, the next instruction (PC+1) is taken out in advance. If it is desired to awaken the device by an interrupt event, the corresponding interrupt enable bit must be set to 1 (enable). The awaken is not related to the GIE bit. If the GIE bit is cleared (disable), the device will continue to execute the instruction after the STOP instruction. If the GIE bit is set to 1 (enable), the device executes the instruction after the STOP instruction and then jumps to the interrupt address (0004h) to execute the code. If you do not want to execute the instruction after the STOP instruction, the user should set a NOP instruction after the STOP instruction. The WDT will all be cleared when the device awakens from sleep mode, regardless of the reason for awakening.

5.3 Interrupt awakening

When the global interrupt is disabled (GIE is cleared) and there exist 1 interrupt source with its interrupt enable bit and flag bit set to 1, one event from the following will happen:

- If an interrupt is generated before the STOP instruction is executed, then the STOP instruction will be executed as a NOP instruction. Therefore, WDT and its pre-scaler and post-scaler (if enabled) will not be cleared. At the same time, the TO bit will not be set to 1 and the PD will not be cleared.
- If an interrupt is generated during or after the execution of the STOP instruction, the device will be immediately awakened from sleep mode. The STOP instruction will be executed before the wake-up. Therefore, the WDT and its pre-scaler and post-scaler (if enabled) will be cleared to zero and the TO bit will be set to 1, while the PD will also be cleared to zero. Even if the flag bit is checked to be 0 before the STOP instruction is executed, it may be set to 1 before the STOP instruction is completed. To determine if the STOP instruction is executed, the PD bit can be tested. If the PD bit is set to 1, then the STOP instruction is executed as a NOP instruction. Before executing the STOP instruction, a CLRWDT instruction must be executed to ensure that the WDT is cleared to zero.

5.4 Sleep mode application

Before the system enters the sleep mode, if the user needs to obtain a smaller sleep current, please check all I/O status at first. If I/O port is required by user, set all floating ports as output to make sure each I/O has a fixed status and avoid increasing sleep current when I/O is inputting; turn off AD and other peripheral modes; WDT functions can be disabled to decrease the sleep current.

Example: procedures for entering sleep mode

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

5.5 Sleep mode awaken time

When the MCU is woken up from sleep, it needs to wait for an oscillation stabilization time (Reset Time), the specific relationship is shown in the table below.

System main clock source	System clock frequency (IRCF<2:0>)	Sleep wakeup wait time T_{WAIT}
Internal high-speed RC oscillation (F_{HSI})	$F_{SYS}=F_{HSI}$	$T_{WAIT}=1032*1/F_{HSI}$
	$F_{SYS}=F_{HSI}/2$	$T_{WAIT}=1032*2/F_{HSI}$
	…	…
	$F_{SYS}=F_{HSI}/64$	$T_{WAIT}=1032*64/F_{HSI}$
External high-speed oscillation (F_{XT})	$F_{SYS}=F_{XT}$	$T_{WAIT}=2056*1/F_{XT}$
Internal low-speed RC oscillation (F_{LSI})	----	$T_{WAIT}=15/F_{LSI}$

6. I/O Ports

The chip has two I/O ports: PORTA, PORTB (up to 14 I/Os). These ports can be accessed directly from the read/write port data registers.

Port	Bit	Pin description	I/O
PORTA	0	Schmitt trigger input, push-pull output, op-amp 0 output, PFG frequency output	I/O
	1	Schmitt trigger input, push-pull output, AN1, KEY1, PWM output, op-amp 0 positive input	I/O
	2	Schmitt trigger input, push-pull output, AN2, KEY2, PWM output, op-amp 0 negative input	I/O
	3	Schmitt trigger input, push-pull output, AN3, KEY3, PWM output, op-amp 1 output	I/O
	4	Schmitt trigger input, push-pull output, AN4, KEY4, PWM output, op-amp 1 positive input	I/O
	5	Schmitt trigger input, push-pull output, AN5, KEY5, PWM output, op-amp 1 negative input, external interrupt input	I/O
	6	Schmitt trigger input, push-pull output, AN6, KEY6, PWM output	I/O
	7	Schmitt trigger input, push-pull output, AN7, KEY7, PWM output	I/O
PORTB	0	Schmitt trigger input, push-pull output, AN13, capacitive touch port, PWM output	I/O
	1	Schmitt trigger input, push-pull output, AN12, KEY12, PWM output	I/O
	2	Schmitt trigger input, push-pull output, AN11, KEY11, PWM output, programming clock input, oscillator output	I/O
	3	Schmitt trigger input, push-pull output, AN10, KEY10, PWM output, programming data input/output, oscillator input port	I/O
	4	Schmitt trigger input, push-pull output, AN9, KEY9, PWM output	I/O
	5	Schmitt trigger input, push-pull output, AN8, KEY8, PWM output	I/O

＜Table 6-1: Port configuration overview＞

6.1 I/O port structure

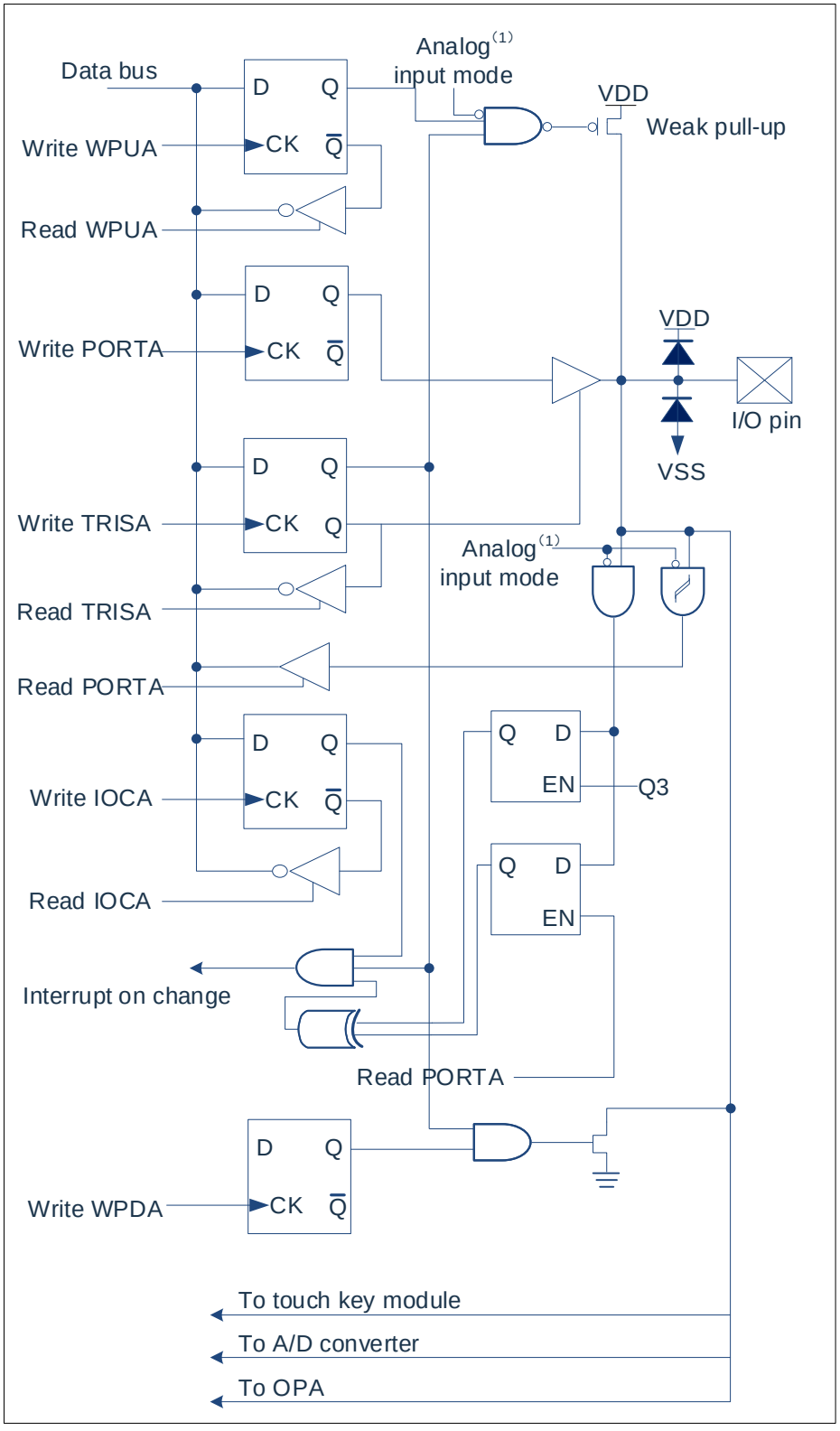


Figure 6-1: I/O port stucture

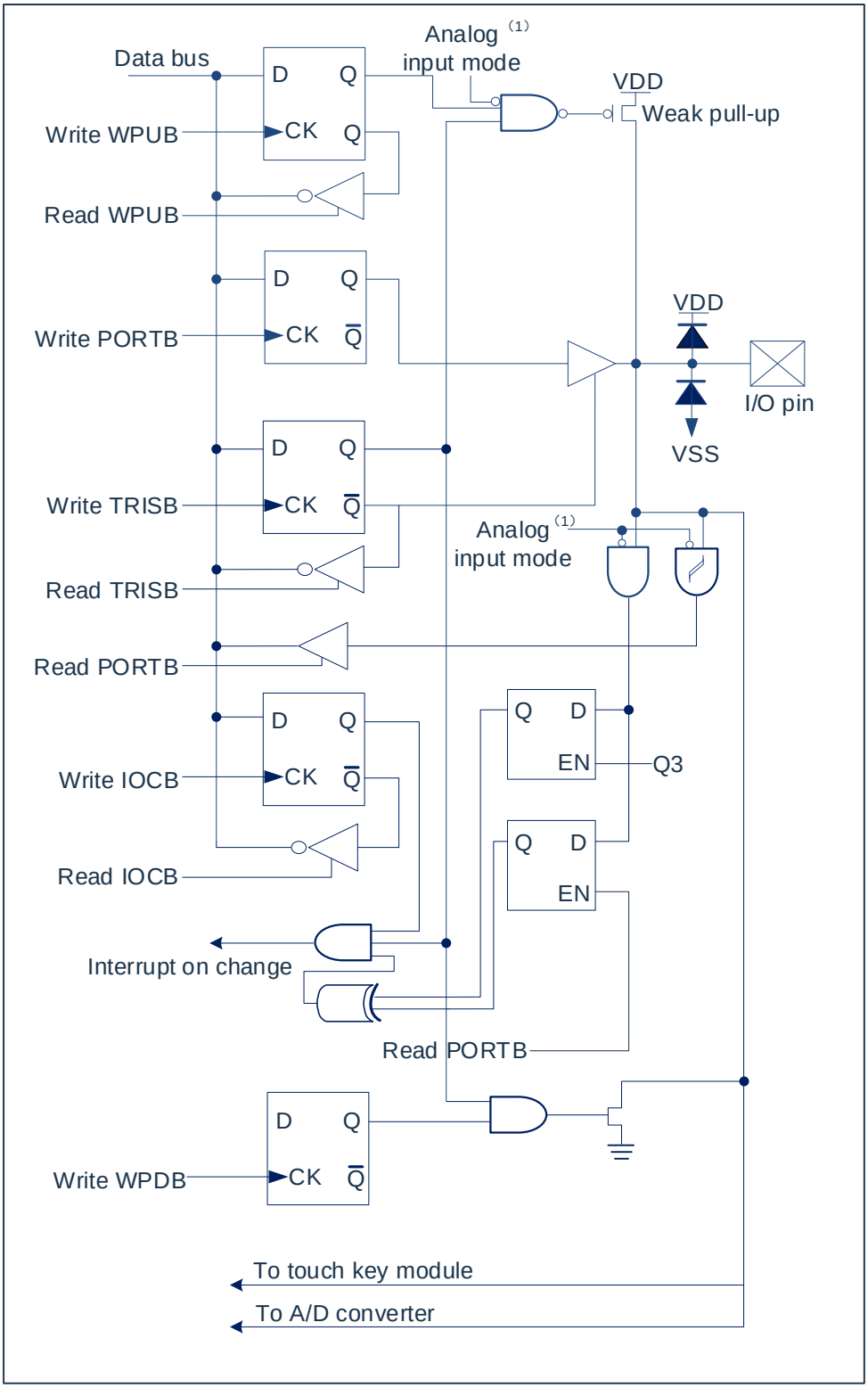


Figure 6-2: I/O port stucture

6.2 PORTA

6.2.1 PORTA data and direction control

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

Reading the PORTA register reads the state of the pin while writing the register will write to the port latch. All write operation procedure is reading-modifying-writing. Therefore, writing a port means reading the pin level of that port at first, then modifying the read value, and finally writing the modified value to the port data latch. Even when the PORTA pin is used as an analog input, the TRISA register still controls the direction of the PORTA pin. When using the PORTA pin as an analog input, the user must ensure that the bit in the TRISA register remains as 1. I/O pins configured as analog inputs always read 0.

The registers related to the PORTA port are PORTA, TRISA, WPUA, WPDA, IOCA, ANSEL and so on.

PORTA data register PORTA (05H)

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

Bit7~Bit0 PORTA<7:0>: PORTAI/O pin bit
1= Port pin level>V_{IH}
0= Port pin level<V_{IL}

PORTA direction register TRISA(85H)

85H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

Bit7~Bit0 TRISA<7:0>: PORTA tristate control bit
1= PORTA pin set to be input(tristate)
0= PORTA pin set to be output

Example: procedure for PORTA

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

6.2.2 PORTA analog selection control

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

PORTA analog selection register ANSEL(188H)

188H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	---
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	---

Bit0 Unused

Bit7~Bit1 ANS<7:1>: Analog selection bit, select the digital or analog function of pin AN<7:1>

1= Analog input. The pin is selected as analog input

0= Digital I/O. The pin is selected as port or special function

6.2.3 PORTA pull-up resistor

Each PORTA pin has an individually configurable internal weak pull-up. Control bits WPUA<7:0> enable or disable each weak pull-up. When a port pin is configured as an output, its weak pull-up is automatically cut off.

PORTA pull-up resistor register WPUA(18EH)

18EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 WPUA<7:0>: Weak pull up register bit

1= Enable pull up

0= Disable pull up

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

6.2.4 PORTA pull-down resistor

Each PORTA pin has an internal weak pull-down that can be individually configured. The control bits WPDA<7:0> enable or disable each weak pull down. When a port pin is configured as an output, its weak pull-down is automatically cut off.

PORTA pull-down resistor register WPDA (07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0
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 WPDA<7:0>: Weak pull-down register bit
1= Enable pull down
0= Disable pull down

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

6.2.5 PORTA interrupt on change

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 (RAIF) in the PIR2 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 RAIF 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

PORTA interrupt-on-change register IOCA(87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0
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 IOCA<7:0> PORTA interrupt-on-change control bit

1= Enable interrupt-on-change

0= Disable interrupt-on-change

6.3 PORTB

6.3.1 PORTB data and direction

PORTB is a 6-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. I/O pin is always read 0 when configured as analog input.

The registers related to the PORTB port are PORTB, TRISB, WPUB, WPDB, IOCB, ANSELH, and so on.

PORTB data register PORTB(06H)

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

Bit7~Bit6	Unused	
Bit5~Bit0	PORTB<5:0>:	PORTB I/O pin bit
	1=	Port pin level >V_{IH}
	0=	Port pin level<V_{IL}

PORTB direction register TRISB (86H)

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

Bit7~Bit6	Unused	
Bit5~Bit0	TRISB<5:0>:	PORTB tristate control bit
	1=	PORTB pin configured as input(tristate)
	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 analog selection control

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

PORTB analog selection register ANSELH(189H)

189H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSELH	---	---	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6	Unused	
Bit5~Bit0	ANS<13:8>:	Analog selection bits, select the analog or digital functions of pin AN<13:8>.
	1=	Analog input. The pin is selected as analog input.
	0=	Digital I/O. The pin is selected as port or special function

6.3.3 PORTB pull-down resistor

Each PORTB pin has an internal weak pull down that can be individually configured. The control bits WPDB<5:0> enable or disable each weak pull down. When the port pin is configured as an output, its weak pull-down is automatically cut off.

PORTB pull-down resistor register WPDB(08H)

08H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDB	---	---	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6	Unused	
Bit5~Bit0	WPDB<5:0>:	Weak pull-down register bit
	1=	Enable pull down
	0=	Disable pull down

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

6.3.4 PORTB pull-up resistor

Each PORTB pin has an internal weak pull up that can be individually configured. The control bits WPUB<5:0> enable or disable each weak pull up. When a port pin is configured as an output, its weak pull-up is automatically cut off.

PORTB pull-up resistor WPUB(95H)

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

Bit7~Bit6 Unused

Bit5~Bit0 WPUB<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.

6.3.5 PORTB interrupt on change

All PORTB pins can be individually configured as interrupt on change pins. The control bit IOCB<5:0> 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 interrupt on change, 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

PORTB interrupt on change register IOCB(96H)

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

Bit7~Bit6	Unused	
Bit5~Bit0	IOCB<5:0>	PORTB interrupt on change control bit
	1=	Enabele interrupt on change
	0=	Disable interrupt on change

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.1 as 1
```

6.4.2 Read I/O port

Example: read 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 Cautions on 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+0.3V" and "GND-0.3V". If the input port voltage is not within this range, the method shown in the figure below can be used.

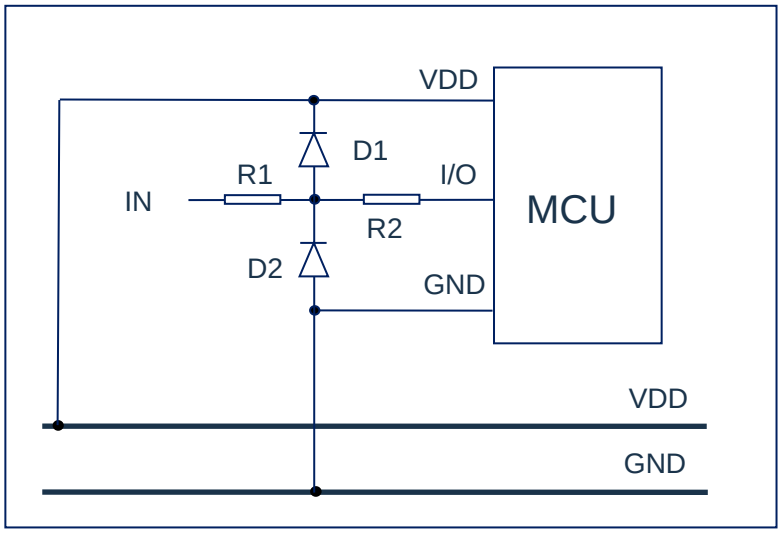


Figure 6-3: Input voltage is not within the specified range using the circuit

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 Overview

The chip has the following multiple interrupt sources:

- TIMER0 overflow interrupt
- TIMER1 overflow interrupt
- TIMER2 match interrupt
- AD interrupt
- PORTA interrupt on change
- PWM interrupt
- PORTB interrupt on change
- INT interrupt
- Program EEPROM write interrupt
- USART receive/transmit interrupt

The interrupt control register (INTCON) and the peripheral interrupt request registers (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 enables 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 unmasked interrupt.

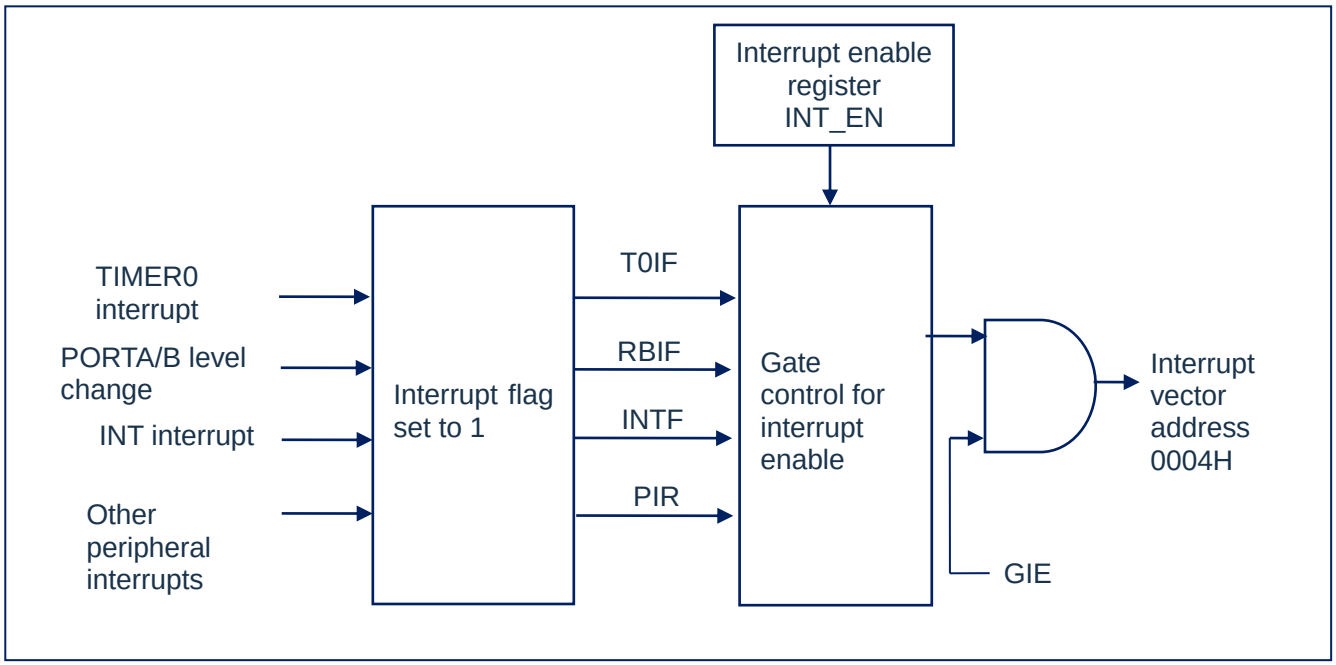


Figure 7-1: Schematic diagram of interrupt principle

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: Peripheral 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 Peripheral interrupt enable register

The peripheral interrupt enable registers have PIE1 and PIE2, and before enabling any peripheral interrupts, you must first set the PEIE bit of the INTCON register to 1.

Peripheral interrupt enable register PIE1(8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	RAIE	ADIE	RCIE	TXIE	EEIE	PWMIE	TMR2IE	TMR1IE
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 RAIE: PORTA interrupt on change enable bit
1= Enable PORTA interrupt on change
0= Disable PORTA interrupt on change

Bit6 ADIE: AD converter (ADC) interrupt enable bit
1= Enable ADC interrupt
0= Disable ADC 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 EEIE: EEDATA interrupt enable bit
1= Enable EEDATA write operation interrupt
0= Disable EEDATA write operation 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 TMR1IE: TIMER1 overflow interrupt enable bit
1= Enable TIMER1 overflow interrupt
0= Disable TIMER1 overflow interrupt

Peripheral interrupt enable register PIE2(8DH)

8DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	---	---	---	---	---	---	---	LVDIE
R/W	---	---	---	---	---	---	---	R/W
Reset value	---	---	---	---	---	---	---	0

Bit7~Bit1 Unused.

Bit0 LVDIE: LVD interrupt enable bit
1= Enable LVD interrupt
0= Disable LVD interrupt

7.2.3 Peripheral interrupt request register

The peripheral interrupt request registers are PIR1 and PIR2. When an interrupt condition is generated, the interrupt flag bit will be set to 1 regardless of the status of the corresponding interrupt enable bit or the global enable bit GIE. The user software should ensure that the corresponding interrupt flag bit is cleared to 0 before enabling an interrupt.

Peripheral interrupt request register PIR1(0CH)

0CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	RAIF	ADIF	RCIF	TXIF	EEIF	PWMIF	TMR2IF	TMR1IF
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7 RAIF: PORTA interrupt on change flag bit
1= The PORTA interrupt on change flag bit is generated
0= The PORTA interrupt on change flag bit is not generated

Bit6 ADIF: A/D converter interrupt flag bit
1= A/D conversion complete (cleared by software)
0= A/D conversion not complete or not start

Bit5 RCIF: USART receive interrupt flag bit
1= USART receive buffer is no full (cleared by reading RCREG)
0= USART receive buffer is empty

Bit4 TXIF: USART transmit interrupt flag bit
1= USART transmit buffer is empty (cleared by writing TXREG)
0= USART transmit buffer is no full

Bit3 EEIF: Program EEPROM write operation interrupt bit
1= Program EEPROM write operation completed (must be cleared by software)
0= Program EEPROM write operation has not been completed or not started

Bit2 PWMIF: PWM interrupt flag bit
1= PWM interrupt happens (must be cleared by software)
0= PWM interrupt not happen

Bit1 TMR2IF: TIMER2 and PR2 match interrupt flag bit
1= TIMER2 and PR2 match happens (must be cleared by software)
0= TIMER2 and PR2 not match

Bit0 Unused, read 0

Peripheral interrupt request register PIR2(0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	---	---	---	---	---	---	---	LVDIF
R/W	---	---	---	---	---	---	---	R/W
Reset value	---	---	---	---	---	---	---	0

Bit7~Bit1 Unused

Bit0 LVDIF: LVD interrupt flag bit
1= The supply voltage is below the voltage point set by the LVD
0= The supply voltage is higher than the voltage point set by the LVD

7.3 Protection methods for interrupt

After an interrupt request occurs and is responded, the program goes to 0004H to execute the interrupt subroutine. 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 overview

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.

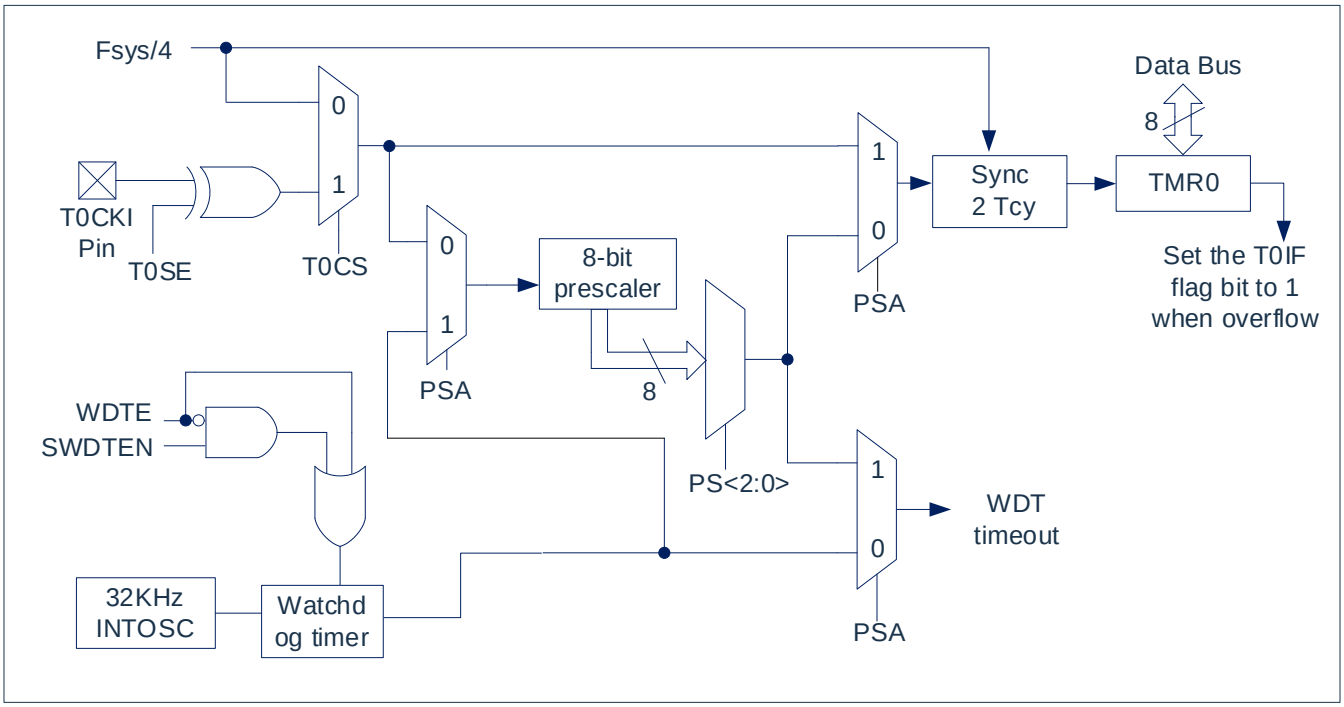


Figure 8-1: TIMER0/WDT mod structure

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

8.2 Working principle for TIMER0

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

8.2.1 8-bit timer mode

When used as a timer, the TIMER0 mod will be incremented every instruction period (without pre-scaler). The timer mode can be selected by clearing the T0CS bit of the OPTION_REG register to 0. If a write operation is performed to the TMR0 register, the next two instruction periods 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 to 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 timing 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 registers

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 Section 2.6 Prescaler Register (OPTION_REG) Application.

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
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	----	1	1	1	1	0	1	1

Bit7 Unused

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_{sys}/4)

Bit4 T0SE: TIMER0 clock source edge selection bit

1= Increment when the T0CKI pin signal transitions from high to low

0= Increment when the T0CKI pin signal transitions from low to high

Bit3 PSA: Pre-scaler allocation bit

1= Pre-scaler allocated to WDT

0= Pre-scaler allocated toTIMER0 mod

Bit2~Bit0 PS2~PS0: Pre-allocated parameter configuration bits

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

9. TIMER1

9.1 TIMER1 overview

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

- 16-bit timer/counter registers (TMR1H:TMR1L)
- 3-bit prescaler
- Synchronous or asynchronous operation
- Gating TIMER1 via T1G pin (enable counting)
- Programmable internal or external clock source
- Overflow interrupt
- Wake-up on overflow (external clock asynchronous mode only)

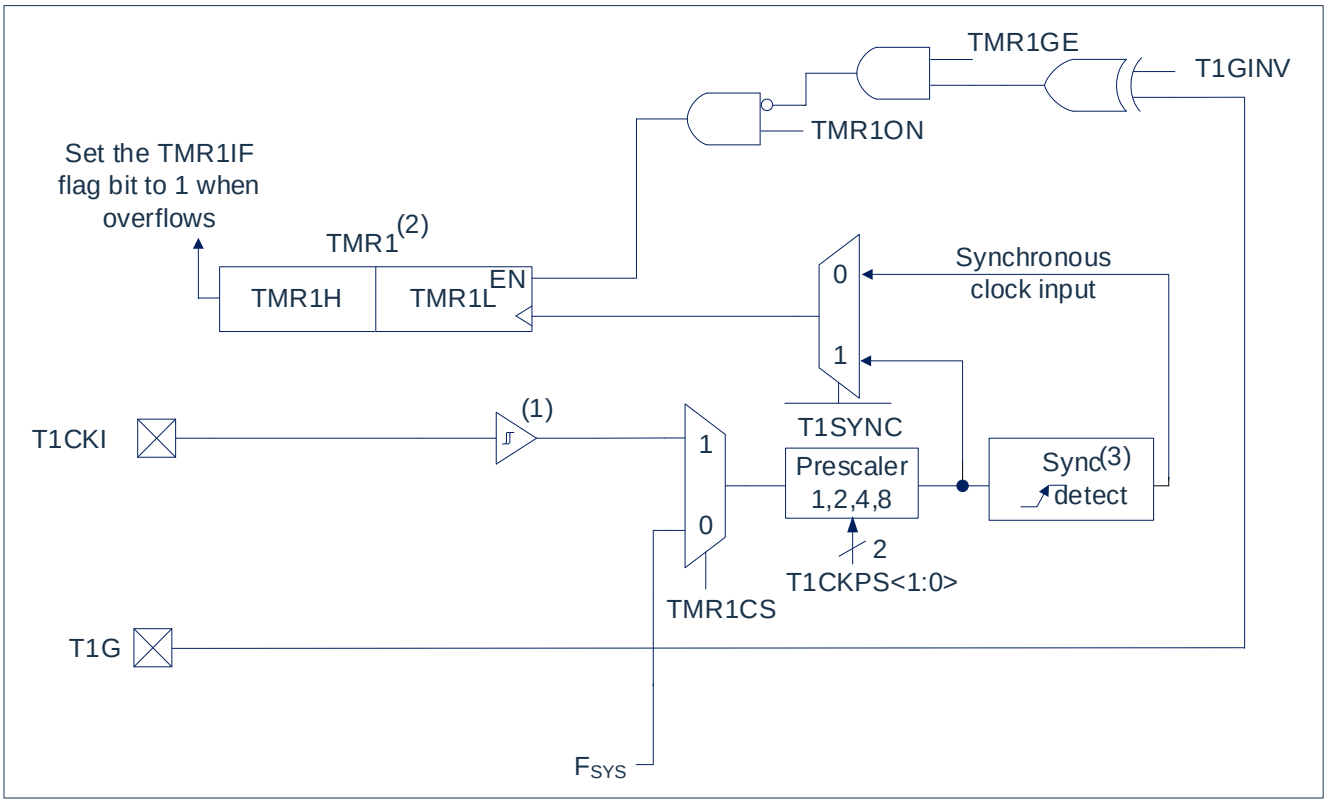


Figure 9-1: Block diagram of TIMER1

Note:

1. The ST buffer is in high speed mode when using T1CKI.
2. Timer1 register is incremented on the rising edge.
3. No synchronization during sleep.

9.2 Working principle for TIMER1

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

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

9.3 Clock source selection

The TMR1CS bit of the T1CON register is used to select the clock source. When TMR1CS=0, the frequency of the clock source is F_{SYS}. When TMR1CS=1, the clock source is provided externally.

Clock source	TMR1CS
F_{SYS}	0
T1CKI pin	1

9.3.1 Internal clock source

After selecting the internal clock source, the TMR1H: TMR1L registers will be incremented by a multiple of F_{SYS}, as determined by the TIMER1 prescaler.

9.3.2 External clock source

After selecting an external clock source, the TIMER1 module can be used as a timer or counter.

When counting, TIMER1 is incremented on the rising edge of the external clock input T1CKI. In addition, the clock in counter mode can be synchronized or asynchronous with the microcontroller system clock.

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

- Enable TIMER1 after a POR or BOR reset.
- Perform a write operation to TMR1H or TMR1L.
- When TIMER1 is disabled, T1CKI goes high; when TIMER1 is re-enabled, T1CKI goes low.

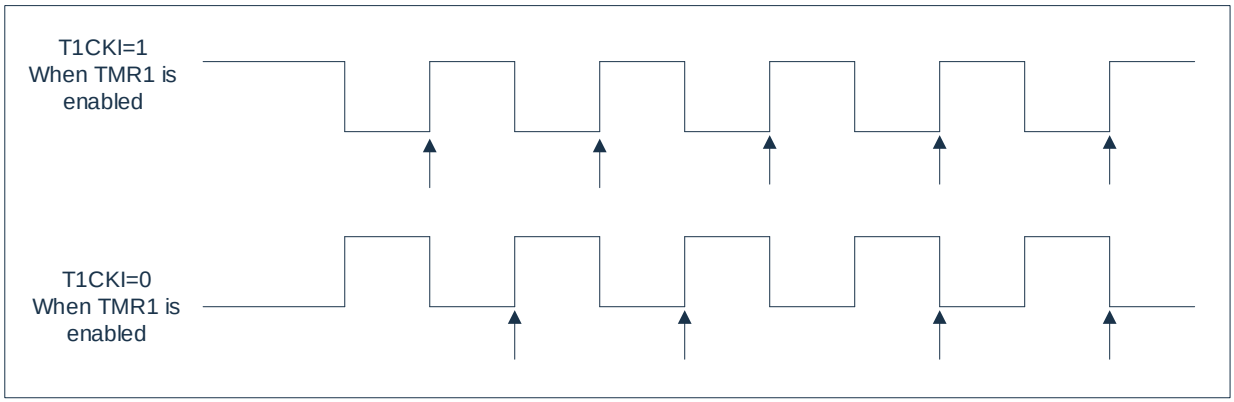


Figure 9-2: Incremental edge of TIMER1

Note:
1. Arrow indicates that the counter is incremented.
2. In counter mode, a falling edge must pass before the counter can count the first increment on the subsequent rising edge.

9.4 TIMER1 prescaler

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

9.5 TIMER1 operation in asynchronous counter mode

If the control bit T1SYNC in the T1CON register is set to 1, the external clock input is not synchronized. The timer continues to count in increments asynchronous to the internal phase clock. The timer will continue to run in the sleep state and will generate an interrupt on overflow to wake up the processor. However, special care should be taken when reading/writing the timer in software (see section “Reading and writing to TIMER1 in asynchronous counter mode”).

Note:
1. When switching from synchronous to asynchronous operation, it is possible to miss an increment.
2. When switching from asynchronous to synchronous operation, a false increment may occur.

9.5.1 Reading and writing to TIMER1 in asynchronous counter mode

When the timer operates with an external asynchronous clock, a read operation to TMR1H or TMR1L is guaranteed to be valid (by the hardware). However, the user should keep in mind that reading a 16-bit timer with two 8-bit values is inherently problematic because the timer may overflow between read operations.

For write operations, it is recommended that the user stop the timer before writing the desired value. Writing to the timer's registers while the registers are being incremented may result in write contention. This can result in unpredictable values in the TMR1H:TMR1L registers.

9.6 TIMER1 gate control

Software can be used to configure the TIMER1 gating source to the T1G pin, which allows the device to directly use T1G to time external events. The information on how to select the TIMER1 gating signal source can be found in "2.1.2 Data memory". This functional part can be just the software for the △-∑A/D converter, or it can be used for many other applications as well.

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

The T1GINV bit of the T1CON register can be used to set the polarity of the TIMER1 gating signal, which can come from the T1G pin. This bit configures TIMER1 to time either the high time or the low time between events.

9.7 TIMER1 interrupt

A pair of TIMER1 registers (TMR1H:TMR1L) incrementally counted to FFFFH will overflow back to 0000H. When TIMER1 overflows, the TIMER1 interrupt flag bit of the PIR1 register is set to 1. To allow this overflow interrupt, the user should set the following bits to 1:

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

The interrupt can be cleared by clearing the TMR1IF bit in the interrupt service program.

Note: The TMR1H:TMR1L registers and the TMR1IF bit should be cleared before allowing the interrupt again.

9.8 TIMER1 working principle during sleep

TIMER1 can only operate in sleep mode if it is set to asynchronous counter mode. In this mode, an external clock source can be used to increment the counter. The timer is enabled to wake up the device by the following settings:

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

The device will wake up on overflow and execute the next instruction. If GIE in the INTCON register is 1, the device will call the interrupt service program (0004h).

9.9 TIMER1 control register

TIMER1 control register T1CON(10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	---	T1SYNC	TMR1CS	TMR1ON
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 T1GINV: TIMER1 gating signal polarity bit
1= TIMER1 gating signal active high (TIMER1 counts when the gating signal is high)
0= TIMER1 gating signal active low (TIMER1 counts when the gating signal is low)

Bit6 TMR1GE: TIMER1 gating enable bit
If TMR1ON=0, this bit is ignored
If TMR1ON=1: 1=TIMER1 counting is controlled by the TIMER1 gating function
0=TIMER1 is always counting

Bit5~Bit4 T1CKPS<1:0>: TIMER1 input clock prescaler ratio select bit
11= 1:8
10= 1:4
01= 1:2
00= 1:1

Bit3 Reserved bits disabled

Bit2 T1SYNC: TIMER1 external clock input synchronization control bit
TMR1CS=1: 1= Not synchronized with external clock input
0= Synchronized with external clock input
TMR1CS=0: Ignore this bit, TIMER1 uses the internal clock

Bit1 TMR1CS: TIMER1 clock source select bit
1= Clock source from T1CKI pin (rising edge triggered)
0= Internal clock sourceF_{SYS}

Bit0 TMR1ON: TIMER1 enable bit
1= Enable TIMER1
0= Disable TIMER1

10. TIMER2

10.1 TIMER2 overview

TIMER2 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).

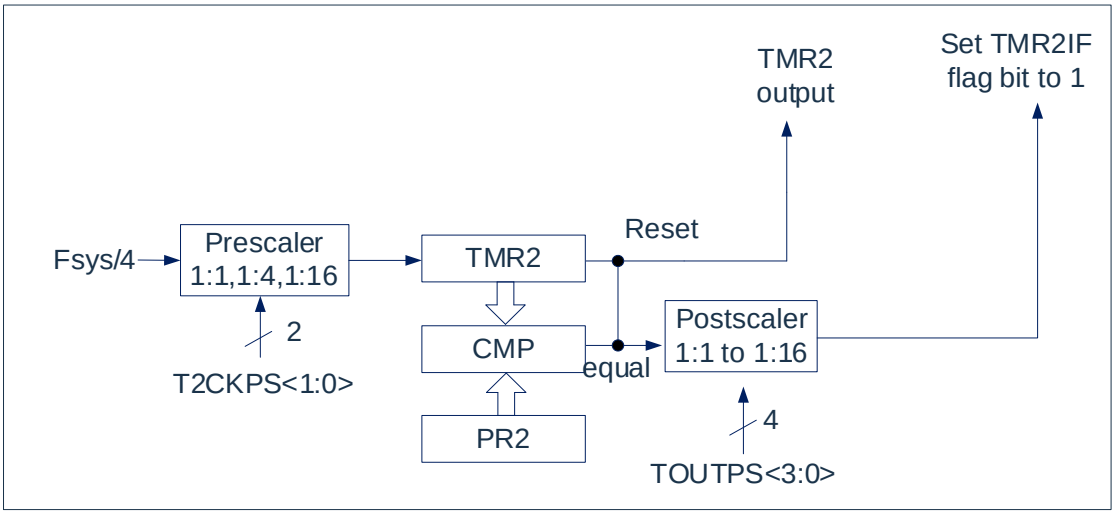


Figure 10-1: Block diagram of TIMER2

10.2 Working principle of TIMER2

The input clock to the TIMER2 module is the system instruction clock (F_{sys}/4). The clock is input to the TIMER2 prescaler, which is available in the following ratios: 1:1, 1:4 or 1:16. The output of the prescaler is then used to increment the 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 FFh.

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

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

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

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

Note: Writing T2CON does not clear TMR2. Writing TMR2ON=0 clears TMR2. TMR2ON needs to be set to 1 to write to TMR2.

10.3 TIMER2 related registers

There are 3 registers related to TIMER2, namely data memory TMR2、cycle registerPR2 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	0	0	0	0	0	0	0	0

TIMER2 cycle register PR2（92H）

11H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PR2								
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

TIMER2 control register T2CON(12H)

12H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CON	---	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
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	Unused	
Bit6~Bit3	TOUTPS<3:0>:	TIMER2 output postscaler ratio select 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 prescaler ratio select bit
	00=	1
	01=	4
	1x=	16

11. Analog to Digital Conversion (ADC)

11.1 ADC overview

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

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

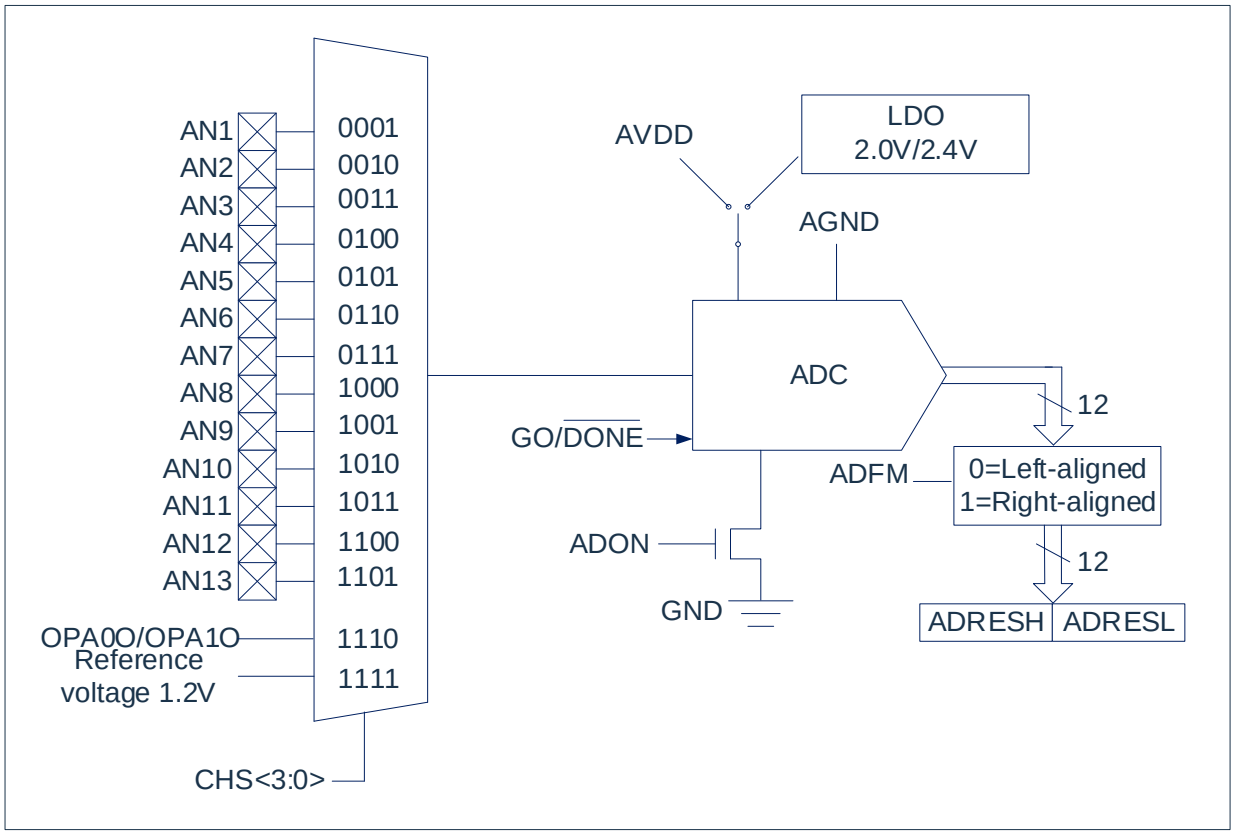


Figure 11-1: Block diagram of ADC

11.2 ADC configuration

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

- Port configuration;
- Reference voltage selection;
- Channel selection;
- ADC conversion clock source;
- Interrupt control;
- Result storage format.

11.2.1 Port configuration

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

Note: Applying an analog voltage to a pin defined as a digital input may result in overcurrent in the input buffer.

11.2.2 Channel selection

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

If the channel is changed, a delay is required before the next conversion starts. For more information see the Section “ADC working principle”.

11.2.3 ADC internal base voltage

The chip has a built-in 1.2V base voltage, when you need to detect this voltage, you need to set ADCON0<5:2> to 1111.

11.2.4 ADC reference voltage

The reference voltage of the ADC can be provided by the internal LDO output or the VDD and GND of the chip. The internal reference voltage can be selected from 2.0V/2.4V.

Note: When the internal LDO is selected as the reference voltage, the maximum effective ADC accuracy decreases. The lower the detection voltage, the higher the accuracy of the ADC. It is recommended to set the input voltage to <1V.

11.2.5 Converter clock

The clock source for the conversion can be selected by software by setting the ADCS bit in the ADCON0 register. There are 4 possible clock frequencies to choose from as follows.

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

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

The following table shows an example of the correct selection of the ADC clock, which must comply with the corresponding TAD specification in order to obtain correct conversion results.

Note: Unless F_{RC} is used, any change in the system clock frequency will change the frequency of the ADC clock, thus negatively affecting the ADC conversion results.

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

Reference voltage	Operating voltage (V)	Fastest frequency division setting		Conversion rate (us)
		F_{SYS} = 16MHz	F_{SYS} = 8MHz	
VDD	4.0~5.5	$F_{SYS}/16$	$F_{SYS}/8$	49
VDD	2.7~4.0	$F_{SYS}/32$	$F_{SYS}/16$	98
Internal reference 2.4V	4.0~5.5	$F_{SYS}/32$	$F_{SYS}/16$	98
Internal reference 2.4V	2.6~4.0	F_{RC}	$F_{SYS}/32$	1000-3000/196
Internal reference 2.0V	2.2~5.5	F_{RC}	$F_{SYS}/32$	1000-3000/196

11.2.6 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 PIR1 register. The ADC interrupt enable bit is the ADIE bit in PIE1 register. The ADIF bit must be cleared by software. The ADIF bit after each conversion is completed Will be set to 1, regardless of whether ADC interrupt is allowed.

11.2.7 Output formatting

The result of 12-bit A/D conversion can be in two formats: left-aligned or right-aligned. The output format is controlled by the ADFM bit in ADCON1 register.

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 10Bit.

11.3 ADC working principle

11.3.1 Start conversion

To enable ADC mod, you must set the ADON bit of the ADCON0 register to 1, and set the GO/$\overline{DONE}$ bit of the ADCON0 register to 1 to start analog-to-digital conversion.

Note: The GO/$\overline{DONE}$ bit cannot be set to 1 with the same command that starts the AD module.

11.3.2 Complete conversion

When the conversion is complete, the ADC mod will:

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

11.3.3 Stop conversion

If you must terminate the conversion before conversion is completed, you can use software to clear the GO/$\overline{DONE}$ 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 shut down the ADC module and terminate any pending conversions.

11.3.4 Working principle of ADC in sleep mode

The ADC module can be operated in sleep mode. This operation requires the ADC clock source to be set to the F_{RC}. If the F_{RC} clock source is selected, the ADC waits one more instruction cycle before starting the conversion. This allows the STOP instruction to be executed to reduce system noise during conversion. If the ADC interrupt is allowed, the device will wake up from sleep mode when the conversion is finished. If the ADC interrupt is disabled, the ADC module will be turned off after the conversion even if the ADON bit is kept set to 1. If the ADC clock source is not F_{RC}, execution of the STOP instruction aborts the current conversion and shuts down the AD module, even though the ADON bit remains set to 1.

11.3.5 AD conversion procedure

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

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

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

Example: AD conversion

```
        LDIA    B'10000000'
        LD      ADCON1,A
        SETB    TRISA,1         ;Set PORTA.1 as input port
        SETB    ANSEL,1         ;Set PORTA,1 as analog port
        LDIA    B'11000101'
        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
```

11.4 ADC related registers

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

AD control register ADCON0(1FH)

1FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON
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 ADCS<1:0>: A/D conversion clock selection bit
- 00= F_{SYS}/8
- 01= F_{SYS}/16
- 10= F_{SYS}/32
- 11= F_{RC} (Clock up to 32 KHz generated by dedicated internal oscillator =F_{LSI})

Bit5~Bit2 CHS<3:0>: Analog channel select bit
- 0000= Reserved
- 0001= AN1
- 0010= AN2
- 0011= AN3
- 0100= AN4
- 0101= AN5
- 0110= AN6
- 0111= AN7
- 1000= AN8
- 1001= AN9
- 1010= AN10
- 1011= AN11
- 1100= AN12
- 1101= AN13
- 1110= OPA0O/OPA1O
- 1111= Internal reference voltage 1.2V

Bit1 GO/$\overline{DONE}$: AD conversion status bit
- 1= AD conversion in progress. Setting this bit to 1 starts the AD conversion. This bit is automatically cleared by the hardware when the AD conversion is complete
- 0= AD conversion complete/ or not in progress

Bit0 ADON: ADC enable bit
- 1= Enable ADC
- 0= Disable ADC, does not consume operating current

AD data register high bit ADCON1(9FH)

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	---	---	---	---	LDO_EN	---	LDO_SEL
R/W	R/W	---	---	---	---	R/W	---	R/W
Reset value	0	---	---	---	---	0	---	0

Bit7 ADFM: AD conversion result format selection bit
1= Right-aligned
0= Left-aligned

Bit6~Bit3 Unused, read 0.

Bit2 LDO_EN: Internal reference voltage enable bit
1= Enable ADC internal LDO reference voltage
When the internal LDO is selected as the reference voltage, the ADC has a maximum effective precision of 8 bits
0= VDD is used as the ADC reference voltage

Bit1 Unused

Bit0 LDO_SEL: LDO reference voltage selection bit
0= 2.4V
1= 2.0V

AD data register high ADRESH(1EH), ADFM=0

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
R/W	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
Higher 8 bits of the 12-bit conversion result

AD data register low ADRESL(9EH), ADFM=0

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

Bit7~Bit4 ADRES<3:0>: ADC result register bit
Lower 4 bits of the 12-bit conversion result

Bit3~Bit0 Unused

AD data register high ADRESH(1EH), ADFM=1

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADRES11	ADRES10
R/W	----	----	----	----	----	----	R	R
Reset value	----	----	----	----	----	----	X	X

Bit7~Bit2 Unused

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

The higher 2 bits of the 12-bit conversion result

AD data register low ADRESL(9EH), ADFM=1

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
R/W	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 9th to 2nd bits of the 12-bit conversion result

Note: When ADFM=1, the AD conversion result only saves the high 10 bits of the 12-bit result, of which ADRESH saves the high 2 bits and ADRESL saves the 9th to 2nd bits.

12. PWM Module

A programmable PWM module with 10-bit width in chip, which can be configured as 4 channels of common period, independent duty cycle output and 1 channel of independent output, or 2 pairs of complementary outputs and 1 channel of independent output.

The PWM output can be selected as RA1-RA5 or RA5-RA7, RB5, RB4 or RB0-RB4 through CONFIG. Among them, PWM0/PWM1 and PWM2/PWM3 can be configured as complementary outputs.

12.1 Pin configuration

The corresponding PWM pin should be configured as output by setting the corresponding TRIS control bit to 0.

12.2 PWM related registers

PWM control register 0 PWMCON0(107H)

107H	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~Bit5 CLKDIV<2:0>: PWM clock frequency 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 1 PWMCON1(108H)

108H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON1	---	---	PWM2DTEN	PWM0DTEN	---	---	DT_DIV<1:0>	
R/W	---	---	R/W	R/W	---	---	R/W	R/W
Reset value	---	---	0	0	---	---	0	0

Bit7~6 Unused

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 Unused

Bit1~Bit0 DT_DIV<1:0> Dead-time source clock frequency division

11= F_{HSI} /8

10= F_{HSI} /4

01= F_{HSI} /2

00= F_{HSI} /1

PWM control register 2 PWMCON2(109H)

109H	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 Unused

Bit4~Bit0 PWMxDIR PWM output inversion control bit

1= PWMx inverted output

0= PWMx normal output

PWM0~PWM3 period low bit register PWMTL(18FH)

18FH	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>: PWM0~PWM3 period low 8 bits

PWM4 period low bit register PWMT4L(191H)

191H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMT4L	PWM4T<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 PWM4T<7:0>: PWM4 period low 8 bits

PWM period high bit register PWMTH (190H)

190H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTH	---	---	PWM4D<9:8>		PWM4T<9:8>		PWMT<9:8>	
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	0	0	0	0	0	0

Bit7~Bit6 Unused
Bit5~Bit4 PWM4D<9:8>: Higher 2 bits of PWM4 duty register
Bit3~Bit2 PWM4T<9:8>: Higher 2 bits of PWM4 period register
Bit1~Bit0 PWMT<9:8>: Higher 2 bits of PWM0~PWM3 period register

PWM0 duty cycle low register PWMD0L(193H)

193H	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 duty cycle low 8 bits

PWM1 duty cycle low register PWMD1L(194H)

194H	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 duty cycle low 8 bits

PWM2 duty cycle low register PWMD2L(195H)

195H	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 duty cycle low 8 bits

PWM3 duty cycle low register PWMD3L(196H)

196H	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 duty cycle low 8 bits

PWM4 duty cycle low register PWMD4L(197H)

197H	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 duty cycle low 8 bits

PWM0 and PWM1 duty cycle high register PWMD01H(11CH)

11CH	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 Unused
Bit5~Bit4 PWMD1<9:8>: PWM1 duty cycle high 2 bits
Bit3~Bit2 Unused
Bit1~Bit0 PWMD0<9:8>: PWM0 duty cycle high 2 bits

PWM2 and PWM3 duty cycle high register PWMD23H(11DH)

11DH	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 Unused
Bit5~Bit4 PWMD3<9:8>: PWM3 duty cycle high 2 bits
Bit3~Bit2 Unused
Bit1~Bit0 PWMD2<9:8>: PWM2 duty cycle high 2 bits

PWM0 and PWM1 deadtime register PWM01DT(93H)

93H	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 Unused
Bit5~Bit0 PWM01DT<5:0>: PWM0 and PWM1 deadtime

PWM2 and PWM3 dead time register PWM23DT(94H)

94H	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 Unused
Bit5~Bit0 PWM23DT<5:0>: PWM2 and PWM3 deadtime

12.3 PWM period

The PWM period is specified by writing the PWMTH and PWMTL register.

Formula 1: PWM period:

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

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

When PWM period counter is equal to PWMT, the following 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.

12.4 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:

$$\text{pulse width}=(PWMDx<9:0>+1)*T_{OSC}*(CLKDIV \text{ prescaler value})$$

Formula 3: PWM duty cycle calculation formula:

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

Internal chip is 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.

12.5 System clock frequency change

The PWM frequency is only related to the chip oscillation clock. Any change in the system clock frequency will not change the PWM frequency.

12.6 Programmable dead-time delay mode

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

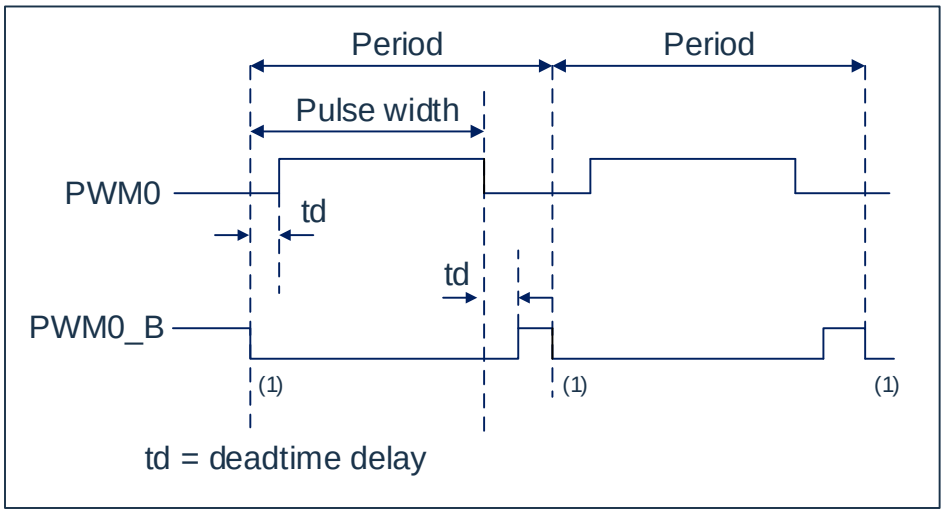


Figure 12-1: Sample of PWM dead-time delay output

Dead-time calculation formula:

$$td=(PWMxxDT<5:0>+1)*T_{OSC}*(DT\_DIV\ prescaler\ value)$$

12.7 PWM configuration

The following steps should be performed when configuring PWM mod:

1. Set the IO_SEL control bit to select the PWM output IO port.
2. Set the corresponding TRIS bit to 1 to make it an input pin.
3. Set the PWM period by loading the PWMTH and PWMTL registers.
4. Set the PWM duty cycle by loading the PWMDxL and PWMDxxH registers.
5. Set the PWM dead-time by setting the PWMCON1<6:5> 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.

13. Program EEPROM and Program Memory Control

13.1 Overview

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 0h to 01Fh, which is readable and writeable in all address ranges.

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 program memory address to be read.

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

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

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

Note:

1. Program memory means ROM space, i.e., the space where the instruction code is stored, and is readable only.
2. The program EEPROM is a space where user data can be stored, read and written.
3. The normal write voltage range of program EEPROM is 3.0V~5.5V, and the write current is 20mA@VDD=5V.

13.2 Related registers

13.2.1 EEADR and EEADRH registers

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 registers

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 the result of EECON2 is 0.

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

EEPROM data register EEDAT(10CH)

10CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
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	EEDAT<7:0>:	To read or write the lower 8 bits of data from the program EEPROM, or read the lower 8 bits of data from the program memory

EEPROM address register EEADR(10DH)

10DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0
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	EEADR<7:0>:	Specify the lower 8 bits of address for program EEPROM read/write operations, or the lower 8 bits of address for program memory read operations

EEPROM data register EEDATH(10EH)

10EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
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 EEDATH<7:0>: The upper 8 bits of data read from the program EEPROM/program memory

EEPROM address register EEADRH(10FH)

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

Bit7~Bit4 Unused, read 0

Bit3~Bit0 EEADRH<3:0>: Specifies the address of the upper 4 bits of program memory for read operations

EEPROM control register EECON1(18CH)

18CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD	---	---	---	WRERR	WREN	WR	RD
R/W	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 Unused

Bit3 WRERR: EEPROM error flag bit
1= Premature termination of the write operation (any WDT reset or undervoltage reset
0= Write operation 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
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
0= Not start memory read operation

13.3 Read program EEPROM

To read the program EEPROM unit, 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 bit set to 1.

Example: read program EEPROM

```
LD      A,EE_ADD        ;the address to be read is placed in the EEADR register.
LD      EEADR,A
CLRB    EECON1,EEPGD    ;select Program EEPROM
SETB    EECON1,RD       ;enable read signals
NOP                     ;to read the data here, you have to add the NOP instruction.
NOP
LD      A,EEDAT         ;read and load data to ACC
```

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) (program loops). 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 and the interrupt flag bit (EEIF) is set to 1. Users allow to interrupt or query this bit. EEIF must be cleared by software.

Note: During writing 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_ADD        ;the address to be written is placed in the EEADR register.
        LD      EEADR,A
        LD      A,EE_DATA       ;the data to be written is placed in the EEDAT register.
        LD      EEDAT,A
        CLRWDT
        CLRB    EECON1,EEPGD
        SETB    EECON1,WREN     ;enable write operations
        CLRB    INTCON,GIE      ;disable all interrupts
        SZB     INTCON,GIE
        JP      $-2
        LDIA    055H            ;write 55H to EECON2
        LD      EECON2,A
        LDIA    0AAH            ;write 0AAH to EECON2.
        LD      EECON2,A
        SETB    EECON1,WR       ;enable write signals
        SETB    INTCON,GIE
        SZB     EECON1,WR       ;determine if the write operation is complete
        JP      $-1
        CLRB    EECON1,WREN     ;write end, disable write enable bit
```

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 EEPGD bit of the EECON1 register 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 instruction cycle to read the data. This causes the second instruction following the "SETB EECON1,RD" instruction to be ignored. In the immediately following clock cycle, the value of the corresponding address in program memory is latched into the EEDAT and EEDATH registers, which can be read by the user in subsequent instructions. The EEDAT and EEDATH registers will save these values until the next time the user reads or writes data to the unit.

Note:

1. The two instructions following a program memory read operation must be NOP. This prevents the user from executing a double-cycle instruction on the next instruction after the RD bit is set to 1.
2. If the WR bit is set to 1 when EEPGD=1, it will immediately reset to 0 without performing any operation.

Example: read Flash program memory

```
        LD      A,EE_ADDL       ;the address to be read is placed in the EEADR register.
        LD      EEADR,A
        LD      A,EE_ADDH       ;the high bit of the address to be read is placed in the
                                EEADRH register.
        LD      EEADRH,A
        SETB    EECON1,EEPGD    ;select operation program memory
        SETB    EECON1,RD       ;read operations are allowed
        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 Cautions on program EEPROM operation

13.7.1 About program EEPROM write time

The program EEPROM burning time is not fixed, and the time required for burning different data varies from 100us to 5ms. Additionally, during the burning process, the CPU stops working, so the program needs to handle this accordingly.

13.7.2 Write verification

Depending on the application, good programming practice generally requires that the value written to the program EEPROM be checked against the desired value.

13.7.3 Avoiding miswriting

In some cases, the user may not want to write data to the program EEPROM. Various protection mechanisms are embedded in the chip to prevent accidental writing to the EEPROM. The WREN bit is cleared at power-on. Also, a power-on delay timer (18ms delay time) prevents writing to the EEPROM.

The initiation sequence of the write operation and the WREN bit will work together to prevent a miswrite operation if:

- Undervoltage
- Supply glitches
- Software failures

14. Operational Amplifier OPA0 and OPA1

There are two sets of operational amplifiers, OPA0 and OPA1, both of them have the same function and performance, the following description is based on OPA0.

14.1 Operational amplifier OPA0

The OPA0 has the following functions.

1. Internal integrated zeroing circuitry.
2. Positive side and negative side can be connected to the I/O ports.
3. The outputs can be connected to I/O ports or internal ADC detection channels.
4. Can be used as a comparator.

14.1.1 OPA0 enable

Set bit 7 of register OPA0CON, OPA0EN, to 1 to enable the operational amplifier. Setting OPA0EN to 0 disables the operational amplifier.

When the op-amp is enabled, the positive and negative sides are automatically connected to the I/O port.

14.1.2 OPA0 port selection

14.1.2.1 OPA0 positive input

When the op-amp is enabled, the positive side is automatically connected to the I/O port.

14.1.2.2 OPA0 negative input

When the op amp is enabled, the negative side is automatically connected to the I/O port.

14.1.2.3 OPA0 output

The output of the op-amp can be output from the OPA0O pin, which is realized by setting bit 6 of OPA0CON, OPA0OEN.

The op-amp output can be connected to the ADC14 channel by setting bit 4 of OPA0CON.

14.1.2.4 Port direction setting for OPA0 operation

The I/O ports associated with the use of OPA0 must be set to the input state, including op-amp inputs and op-amp outputs (when connecting to an IO is required).

14.1.3 OPA0 operation mode

The chip's built-in op-amp has 2 operating modes: normal mode and regulation mode.

The 6th bit OPA0COFM of register OPA0ADJ is set to 0, and the op-amp enters normal operation mode.

The 6th bit OPA0COFM of register OPA0ADJ is set to 1, and the op-amp enters regulation mode. In this mode, the positive and negative terminals of the op-amp are internally shorted together and connected to the positive or negative terminal of the op-amp (selected by bit 5th bit OPA0CRS of OPA0ADJ). This mode serves to minimize the op-amp's offset voltage.

Note: After the chip power-on reset is completed, the op-amp offset voltage trim value will be automatically loaded into the OPAxADJ register, and if the trim value is not suitable, it can be re-adjusted by entering the regulation mode.

Regulation mode workflow:

1. Enable the op-amp function;
2. Set the op-amp into regulation mode;
3. Set the op-amp regulation mode from positive input or negative input, with the input is not floating;
4. Set the regulation bit OPA0ADJ<4:0> to its initial value, maximum (1FH) or minimum (00H);
5. Delay for a period of time, which is related to the external capacitor parameters;
6. Read the op-amp output;
7. Self-decrease the regulation bit by 1 (initial value set to maximum 1FH) or self-add 1 (initial value set to 00H);
8. Time delay;
9. Read the op-amp output if it has changed, and if not, continue with step 7;
10. When the read value is changed, the zeroing is finished. When the OPA0COFM is cleared to zero, and the normal operation mode is entered.

14.1.4 OPA0 related registers

There are two registers related to OPA0, the control register OPA0CON (98H) and the offset voltage regulation register OPA0ADJ (99H).

OPA0 control register OPA0CON (9AH)

9AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPA0CON	OPA0EN	OPA0O	OPA0_CMP	OPA0_ADC	OPA0FW	OPA0BG	---	---
R/W	R/W	R/W	R/W	R/W	R/W	R/W	---	---
Reset value	0	0	0	0	0	0	---	---

Bit7	OPA0EN:	OPA0 enable bit
	1=	Enable OPA0
	0=	Disable OPA0
Bit6	OPA0O	Op-amp output selection
	1=	OPA0 output connected to the I/O port (OPA0O pin)
	0=	OPA0 output is not connected to the I/O port
Bit5	OPA0_CMP:	Comparator mode
	1:	Comparator mode, comparator output can be read via OPA0ADJ<7>
	0:	Op-amp mode
Bit4	OPA0_ADC:	Op-amp output to ADC control bit
	1=	The op-amp output is connected to the ADC14 channel
	0=	The op-amp output is not connected to the ADC
Bit3	OPA0FW	OPA0 follow mode control bit; (valid only in op-amp mode)
	1=	OPA0 negative side is shorted to the output, OPA0 negative side is automatically disconnected from the I/O port
	0=	OPA0 negative side is is not shorted to the output
Bit2	OPA0BG	Internal reference 1.2V connected to OPA0 positive control bit
	1=	The internal reference is connected to the positive OPA0 input, and the positive OPA0 is automatically disconnected from the I/O port.
	0=	Not connected
Bit1~Bit0	Unused	

Note: Only one of the outputs of OPA0 and OPA1 can be connected to the ADC14 channel at any one time.

OPA0 offset-voltage regulation register OPA0ADJ (9BH)

9BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPA0ADJ	OPA0OUT	OPA0COFM	OPA0CRS	OPA0ADJ<4:0>				
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	1	0	0	0	0

Bit7 OPA0OUT: OPA0 output results
1= The op-amp output is high, with the positive side voltage being higher than the negative side voltage
0= The op-amp output is low, with the positive side voltage being lower than the negative side voltage

Bit6 OPA0COFM: OPA0 operating mode selection bit
1= OPA0 is operating in regulation mode
0= OPA0 is operating in normal mode

Bit5 OPA0CRS: OPA0 regulation mode input selection bit
1= OPA0 regulation mode positive input
0= OPA0 regulation mode negative input

Bit4~Bit0 OPA0ADJ<4:0>: OPA0 offset voltage regulation bit

15. 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 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 communications. 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 15-1 and Figure 15-2 below are the block diagrams of the USART transmitter.

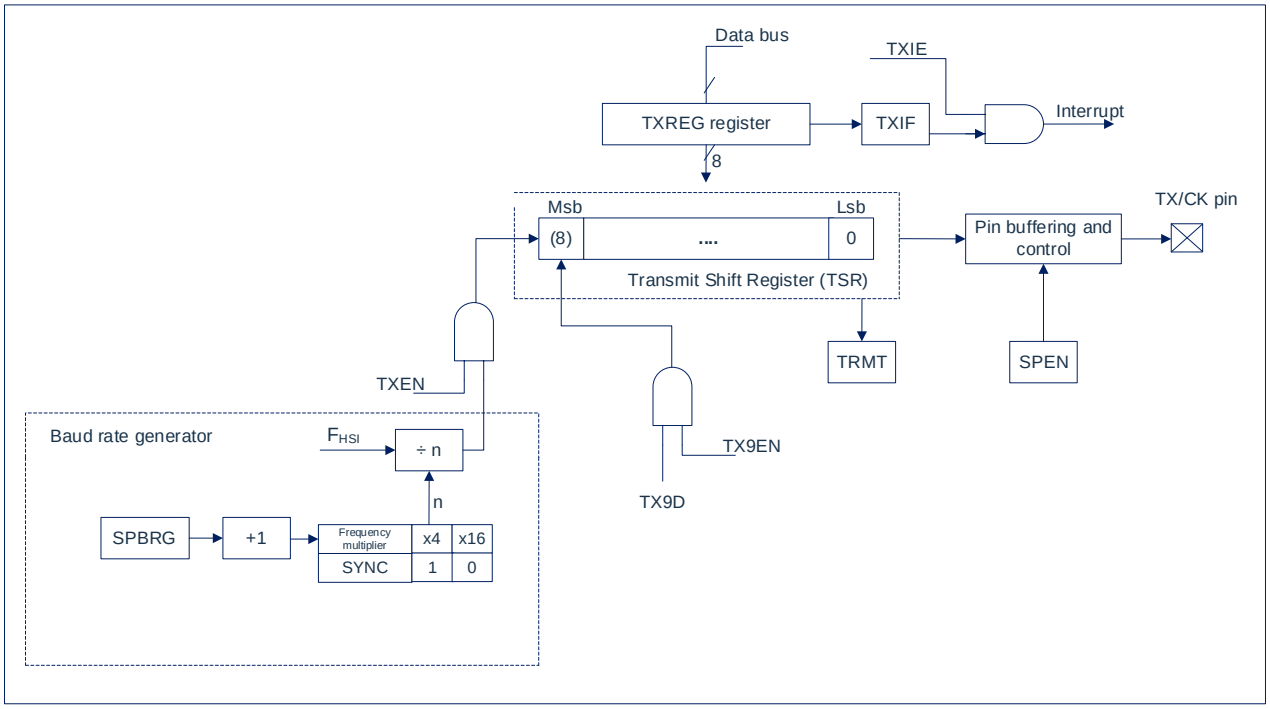


Figure 15-1: Block diagram of USART transmission

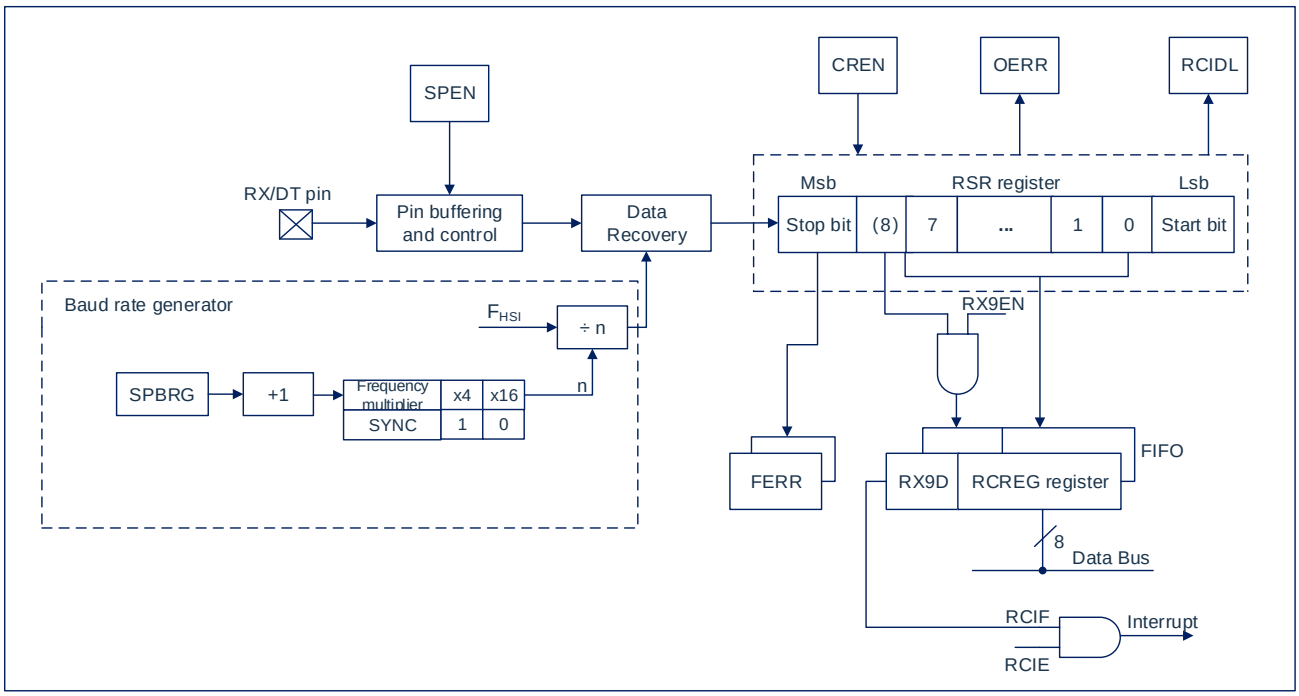


Figure 15-2: Block diagram of USART reception

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

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

15.1 USART asynchronous mode

USART uses the standard non-return-to-zero (NRZ) format for transmitting and receiving 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 the system oscillator.

USART first transmits and receives Lbs. 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 9th data bit).

15.1.1 USART asynchronous generator

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

15.1.1.1 Enable transmitter

Enable the USART transmitter 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 the default state.

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

Note:

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

15.1.1.2 Transmit data

Write a character to the TXREG register to start transmission. 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 the data is transmitted from TXREG to TSR, the sequence of start bits, data bits, and stop bits is transmitted immediately.

15.1.1.3 Transmit interrupt flag

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 transmission in the TXREG, the TXIF bit It is in the clear state. When writing TXREG, the TXIF flag bit is not cleared immediately. TXIF is cleared at the second instruction period after writing the instructions. Querying TXIF immediately after writing TXREG will return an invalid result. TXIF is a read-only bit and cannot be set or cleared by software.

TXIF interrupt can be enabled by setting TXIE interrupt enable bit of the PIE1 register. 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.

15.1.1.4 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 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.

15.1.1.5 Transmit 9-bit character

The USART supports 9-bit character transmission. When the TX9EN bit of the TXSTA register is 1, the USART will remove 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 transmitting 9 bits of data, the TX9D data bits must be written before the 8 lowest bits are written to TXREG. The 9 data bits are transferred to the TSR shift register immediately after writing to the TXREG register.

15.1.1.6 Asynchronous transmit configuration

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, an 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.

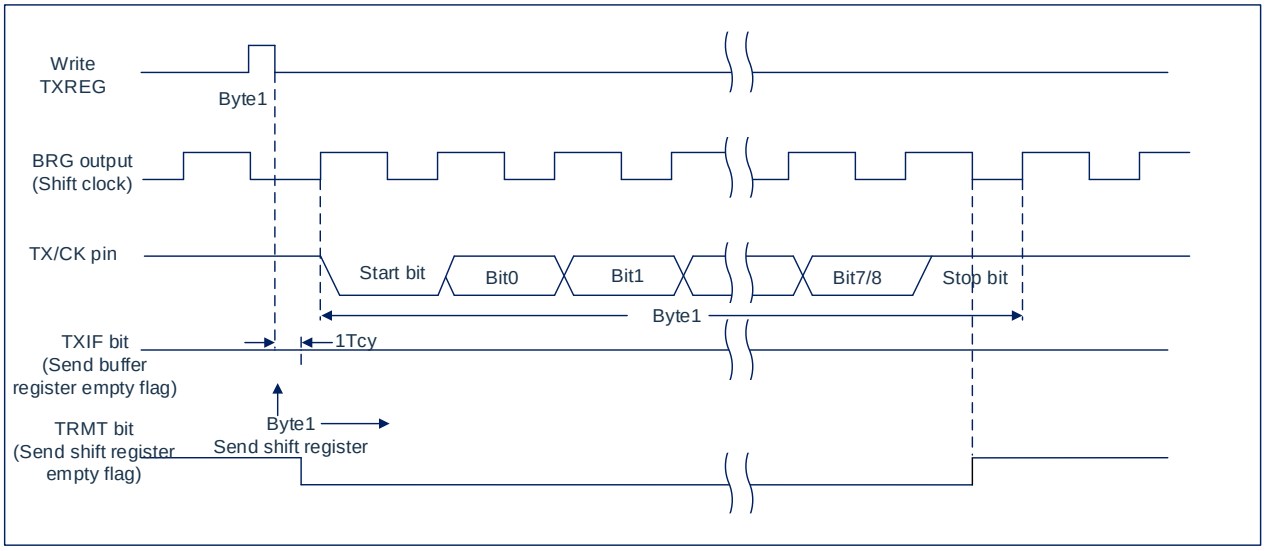


Figure 15-3: Asynchronous transmission

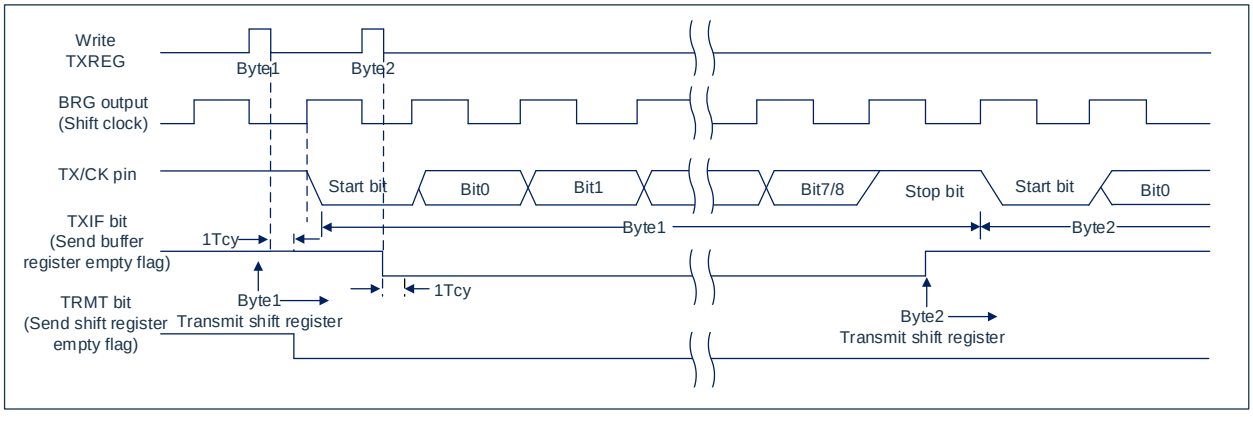


Figure 15-4: Asynchronous transmission (back to back)

Note: This timing diagram shows two consecutive transmissions.

15.1.2 USART asynchronous receiver

Asynchronous mode is usually used in RS-232 system. Figure 15-2 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 receives 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. The FIFO and RSR registers are not directly accessible by software and are accessed through the RCREG register to access received data.

15.1.2.1 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.

15.1.2.2 Receive data

The receiver data recovery circuit starts receiving characters on the falling edge of the first bit. The first bit, often referred to as the start bit, is always zero. The data recovery circuit counts half a bit time to the center of the start bit, thus verifying that the bit is still zero. If this bit is not zero, the data recovery circuit aborts reception of the character without generating an error and continues to look for the falling edge of the start bit. If the start bit zero check passes, the data recovery circuit counts one full bit time to the center of the next bit. The bit is sampled by the selective multi-detection circuit, and the corresponding 0 or 1 is shifted into the RSR. Repeat this process until all data bits have been sampled and shifted into the RSR register. Measure the time of the last bit and sample its level. This bit is a stop bit and is always 1. If the data recovery circuit samples a 0 in the stop bit position, the frame error flag for that character will be set to 1, otherwise, the frame error flag for that character will be cleared to zero.

When all data bits and stop bits are received, the character in the RSR is immediately transferred to the receive FIFO of the USART and the RCIF interrupt flag bit of the PIR1 register 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 the receive FIFO overflows, no more characters can be received until the overflow condition is cleared.

15.1.2.3 Receive interrupt

As long as the USART receiver is enabled and there is no unread data in receiving FIFO, the RCIF interrupt flag bit in the PIR1 register will be clear to 0. The RCIF interrupt flag bit is read-only and cannot be set to 1 or cleared by software.

RCIF interrupt is enabled by setting all of the following bits to 1:

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

15.1.2.4 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 is 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 an interrupt.

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

15.1.2.5 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.

15.1.2.6 Receive 9-bit character

The USART supports 9-bit data reception. 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.

15.1.2.7 Asynchronous receive 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 SPEN bit to 1 to enable the serial port. The SYNC bit must be cleared to perform asynchronous operations.
3. If an 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 reception.
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.

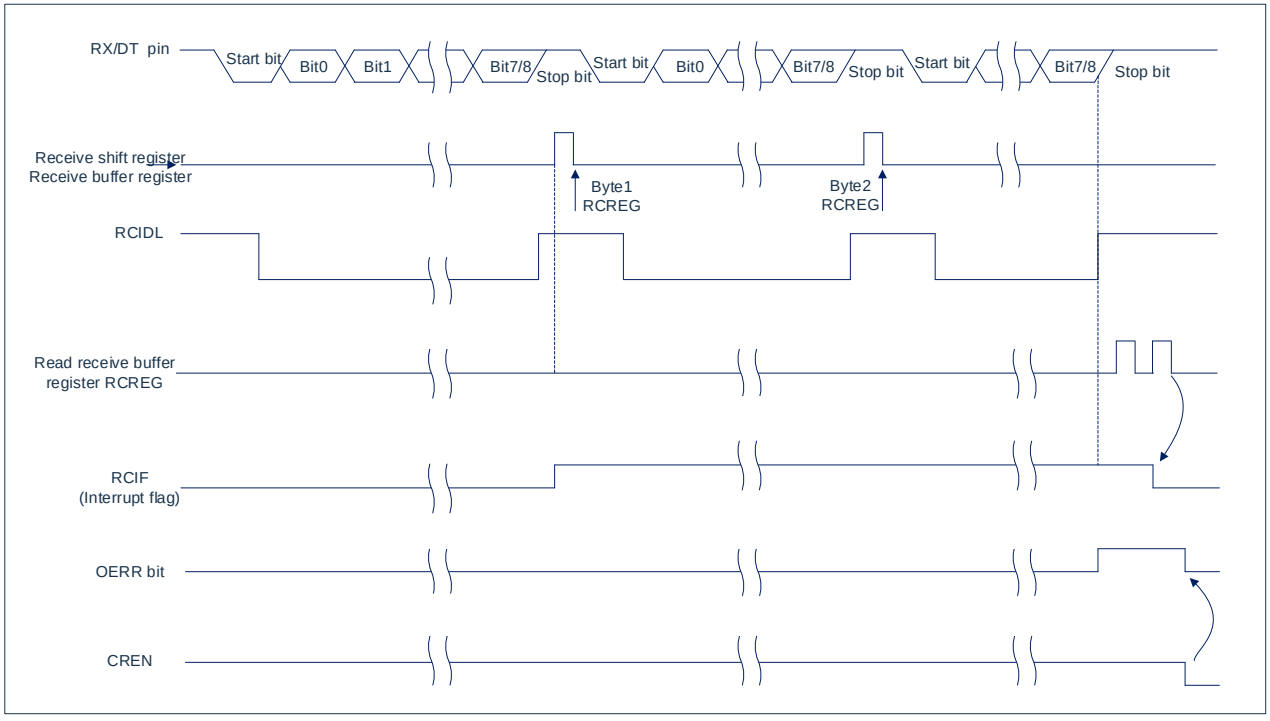


Figure 15-5: Asynchronous reception

Note: The timing diagram shows the situation of three words reception in RX input pin. Reading RCREG (receive buffer) after the third word causes the OERR (overflow) bit to be set to 1.

15.2 Clock precision during asynchronous operation

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 in the following ways, but some type of reference clock source is required.

15.3 USART related registers

TXSTA: transmit status and control register (98H)

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXSTA	CSRC	TX9EN	TXEN(1)	SYNC	SCKP	--	TRMT	TX9D
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 generation clock signal)
0=slave mode (external clock source generation 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 Unused
Bit1 TRMT: transmit shift register status bit
1= TSR is empty
0= TSR is full
Bit0 TX9D: 9th bit of transmit data
can be address/data bit or parity check bit

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

RCSTA: receive status and control register (18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D
R/W	R/W	R/W	R/W	R/W	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Bit7 SPEN: serial port enable bit
1= enable serial port (RX/DT and TX/CK pin configured as serial port pin)
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 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 errors

Bit1 OERR: overflow error bit
1= overflow error (clear by clearing CREN bit)
0= no overflow errors

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

15.4 USART baud rate generator (BRG)

The Baud Rate Generator (BRG) is an 8-bit baud rate generator designed to support both asynchronous and synchronous USART modes of operation.

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

Table 15-1 contains the formula for calculating baud rate. Formula 1 is an example of calculating baud rate and baud rate error.

Table 15-1 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_{SYS}=8MHz, target baud rate=9600bps, asynchronous mode is 8-bit BRG:

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

To solve SPBRG:

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

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

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

Table 15-1: baud rate formula

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

Note: n= value of SPBRG register.

Table 15-2: baud rate in asynchronous mode

Target baud rate	SYNC=0					
	F_{SYS}=8.00MHz			F_{SYS}=16.00MHz		
	Actual baud rate	Error (%)	SPBRG value	Actual 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

15.5 USART synchronous mode

Synchronous serial communication is usually used in a system with a master 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 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 the 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 device, or as a slave device.

Start and stop bits are not necessary in synchronous transmit mode.

15.5.1 Synchronous master mode

The following bits are used to configure the USART for synchronous master 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 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.

15.5.1.1 Master 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.

15.5.1.2 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.

15.5.1.3 Synchronous master 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.

15.5.1.4 Synchronous master 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 a 9-bit character, you should load the 9th bit of data into the TX9D bit.
8. Start transmit by loading data into TXREG register.

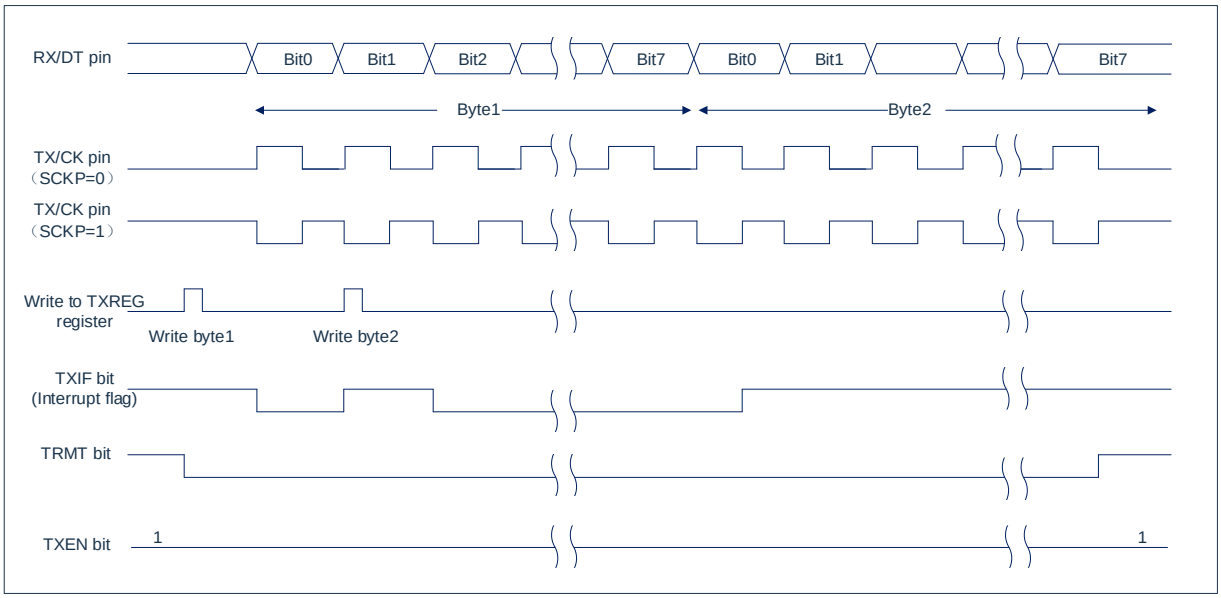


Figure 15-6: Synchronous transmission

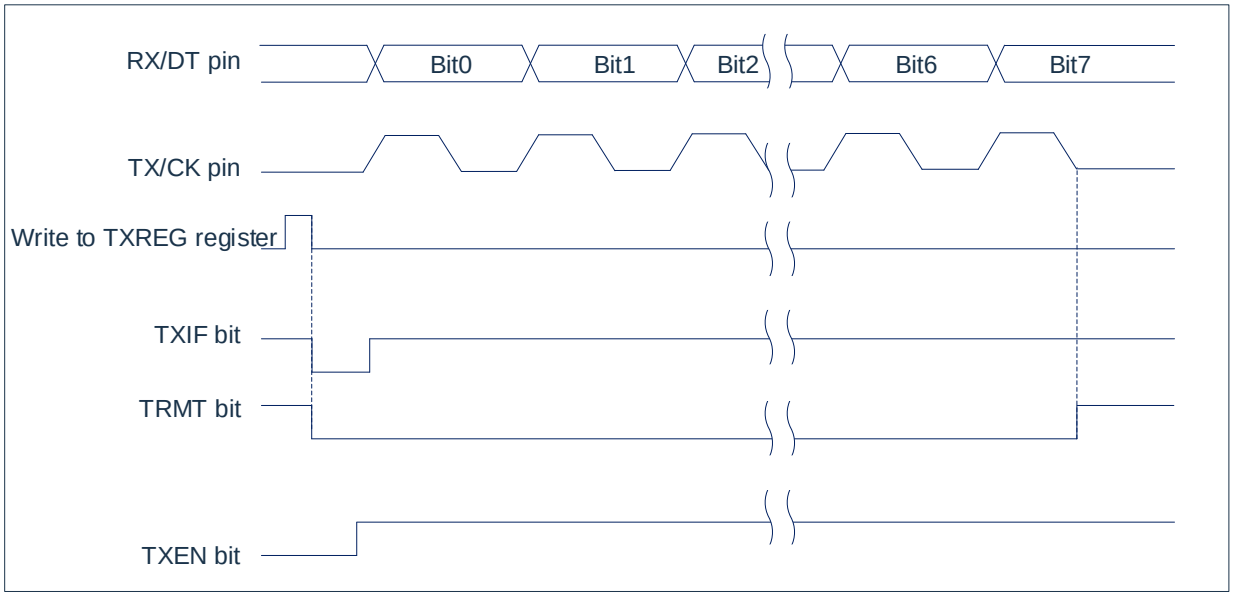


Figure 15-7: Synchronous transmission (by TXEN)

15.5.1.5 Synchronous master receive

RX/DT pin receives data. When the USART configuration is in synchronous master reception, 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 for enabling reception. When the SREN bit 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 the 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, starting reception. 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.

15.5.1.6 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 is automatically disabled. 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.

15.5.1.7 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, and to clear the error.

15.5.1.8 Receive 9-bit character

The USART supports 9-bit character reception. 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 bits of RCREG at first.

15.5.1.9 Synchronous master 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.

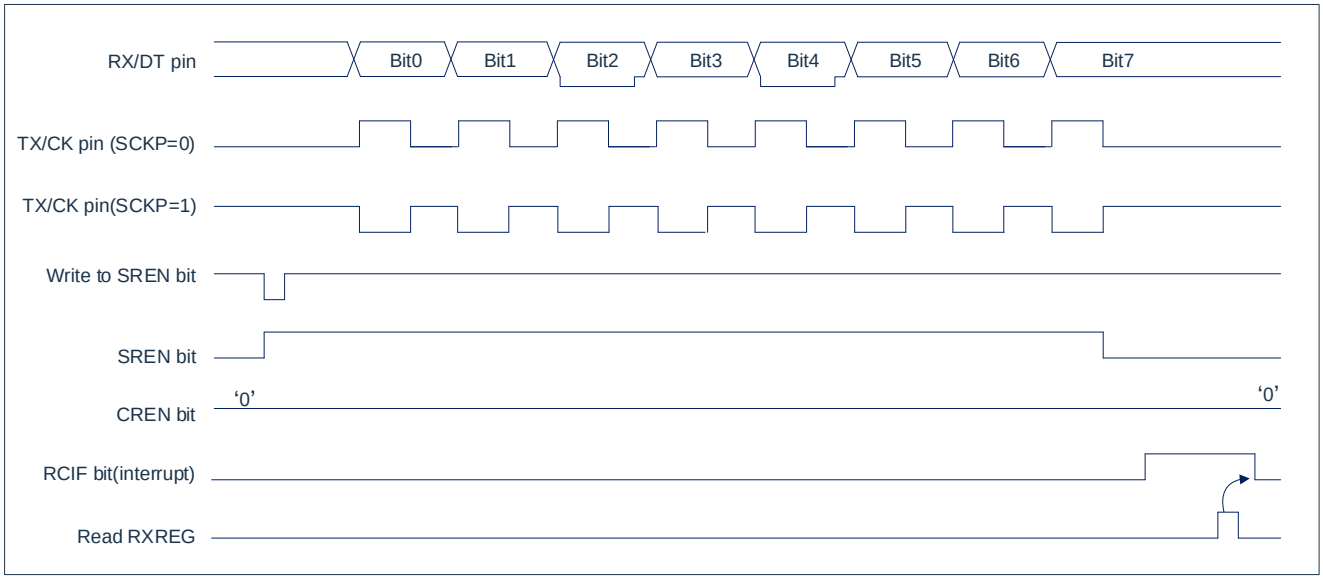


Figure 15-8: Synchronous reception (master mode, SREN)

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

15.5.2 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.

15.5.2.1 USART synchronous slave transmit

The working principle of synchronous master control and slave mode is the same (see section “synchronous master transmit”).

15.5.2.2 Synchronous slave transmit configuration

1. Set the SYNC and SPEN bits to 1 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 to 1.
4. If you need to transmit 9-bit data, set the TX9EN bit to 1.
5. Set the TXEN bit to 1 to enable transmission.
6. If you choose to transmit 9-bit data, write the highest bit to the TX9D bit.
7. Write the lower 8 bits of data to the TXREG register to start transmission.

15.5.2.3 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. The SREN bit, can be "any value" in slave mode.

15.5.2.4 Synchronous slave receive configuration

1. Set the SYNC and SPEN bits to 1 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 to 1.
3. If you need to receive a 9-bit character, set the RX9EN bit to 1.
4. Set the CREN bit to 1 to enable reception.
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 receive 8-bit low data from the receive FIFO buffer.
7. If you enable 9-bit mode, get the highest 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 for clearing the error.

16. Touch Key

16.1 Touch key module overview

The Touch detection module is an integrated circuit designed to realize the human touch interface. It replaces the mechanical keys and realizes a waterproof and dustproof, sealed and isolated, rugged and aesthetically pleasing operating interface.

Technical parameters:

- 1-12 keys available;
- Sensitivity can be adjusted by external capacitor.
- Effective touch response time <100ms.

17. Low Voltage Detection (LVD)

17.1 LVD module overview

The CMS79F623 microcontroller has a low voltage detection function that can be used to monitor the supply voltage. If the supply voltage falls below a set value, an interrupt can be generated. The program can read the LVD output flag bit in real time.

17.2 LVD related registers

There is 1 register associated with the LVD module.

LVD control registerLVDCON(97H)

97H	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> LVD set voltage
	1=	VDD< LVD set voltage
Bit6~Bit4	Unused	
Bit3~Bit1	LVD_SEL<2:0>:	LVD voltage selection
	000=	2.2V
	001=	2.4V
	010=	2.7V
	011=	3.0V
	100=	3.3V
	101=	3.7V
	110=	4.0V
	111=	4.3V
Bit0	LVDEN:	LVD enable bit
	0=	disable
	1=	enable

17.3 LVD operation

By setting the LVD voltage value in the LVDCON register, after enabling LVDEN, when the power supply voltage is lower than the set voltage value, the LVD_RES bit in the LVDCON register is set high. After LVD mod is enabled, it takes a delay of 10us to be able to read the LVD_RES bit, because the internal has done filtering processing to reduce the frequent fluctuation of the LVD output result when the VLVD voltage is near.

LVD mod has its own interrupt flag bit. When the relevant interrupt enable bit is set, and the power supply voltage is lower than the set voltage value, LVD interrupt will be generated, the interrupt flag bit LVDIF will be set to 1, and interrupt generation. LVD is cannot used for interrupt wake up mode.

18. DIV Hardware Divider

18.1 Hardware divider overview

The CMS79F623 microcontroller has a built-in hardware divider, 24-bit dividend, 12-bit divisor, no remainder output.

The dividend is set by DIVE2~DIVE0 registers, which can only be written but not read. The divisor is set through the DIVS1 and DIVS0 registers, which are readable and writable. The quotient of the operation is stored in the DIVQ2, DIVQ1 and DIVQ0 registers, which can only be read, not written. DIVEx and DIVQx share the same register address. After the dividend and divisor are set, enable DIVEN and wait for the CAL_END bit to be 1 before reading the quotient.

18.2 Divider related registers

There are 9 registers associated with the divider module, DIVCON, DIVE2, DIVE1, DIVE0, DIVS1, DIVS0, DIVQ2, DIVQ1 and DIVQ0.

DIV control register DIVCON(1BH)

1BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVCON	DIVEN	CAL_END	---	---	---	---	---	DIV_CLK
R/W	R/W	R	---	---	---	---	---	R/W
Reset value	0	1	---	---	---	---	---	0

Bit7 DIVEN: DIV divider enable bit
0= disable
1= enable

Bit6 CAL_END: Operational end flag bit
0= In progress
1= The division operation has not yet begun or has been completed

Bit5~Bit1 Unused

Bit0 DIV_CLK: DIV operation clock division select bit
0: F_{SYS}
1: $F_{SYS}/2$ (When the oscillator clock is 16M, DIV_CLK should be set to 1.)

Divisor dividend register DIVE2 (15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVE2								
R/W	W	W	W	W	W	W	W	W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Dividend DIVE<23:16>

Divisor dividend register DIVE1 (16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVE1								
R/W	W	W	W	W	W	W	W	W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Dividend DIVE<15:8>

Divisor dividend register DIVE0 (17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVE0								
R/W	W	W	W	W	W	W	W	W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Dividend DIVE<7:0>

Divider divisor register DIVS1 (13H)

13H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVS1	---	---	---	---				
R/W	---	---	---	---	R/W	R/W	R/W	R/W
Reset value	---	---	---	---	0	0	0	0

Bit7~Bit4 Unused
Bit3~Bit0 Divisor DIVS<11:8>

Divider divisor register DIVS0 (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVS0								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Divisor DIVS<7:0>

Divider quotient register DIVQ2 (15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVQ2								
R/W	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Quotient DIVQ<23:16>

Divider quotient register DIVQ1 (16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVQ1								
R/W	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Quotient DIVQ<15:8>

Divider quotient register DIVQ0 (17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DIVQ0								
R/W	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 Quotient DIVQ<7:0>

19. Electrical Parameters

19.1 Limit parameters

Supply 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 stability of the device will be affected if it is operated for a long period of time under extreme conditions.

19.2 DC electrical characteristics

(VDD=5V, T_A= 25˚C, unless otherwise specified)

Symbol	Item	Test condition		Min.	Typ.	Max.	Unit
		VDD	Condition				
VDD	Operating voltage	-	16MHz	2.6	-	5.5	V
		-	8MHz	1.8	-	5.5	V
I_{DD}	Operating current	5V	Disable all analog modules, Fsys=8MHz	-	3	-	mA
		3V	Disable all analog modules, Fsys=8MHz	-	2	-	mA
		5V	Disable all analog modules, Fsys=16MHz	-	3.8	-	mA
		3V	Disable all analog modules, Fsys=16MHz	-	2.7	-	mA
		5V	Programming EEPROM	-	20	-	mA
		3V	Programming EEPROM	-	10	-	mA
I_{STB}	Static current	5V	----	-	0.1	10	μA
		3V	----	-	0.1	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_{ADI}	AD port input voltage	-	----	0	-	V_{ADREF}	V
V_{AD}	AD module operating voltage	-	V_{ADREF}=VDD	2.7	-	5.5	V
		-	V_{ADREF}=2.0V CLK_{ADC}= 250KHz	2.2		5.5	V
		-	V_{ADREF}=2.4V CLK_{ADC}= 250KHz	2.6		5.5	V
V_{EEPROM}	EEPROM module operating voltage	-	----	3	-	5.5	V
E_{AD}	AD conversion error	-	----	-	±2	-	-
R_{PH}	Pull-up resistor resistance	5V	V_O=0.5VDD	-	40	-	KΩ
		3V	V_O=0.5VDD	-	73	-	KΩ
R_{PD}	Pull-down resistor resistance	5V	V_O=0.5VDD	-	40	-	KΩ
		3V	V_O=0.5VDD	-	70	-	KΩ
I_{OL}	Output port source current	5V	V_{OL}=0.3VDD	-	60	-	mA
		3V	V_{OL}=0.3VDD	-	25	-	mA
I_{OH1}	Output port source current RA1~RA7,RB5	5V	V_{OH}=0.7VDD	-	21	-	mA
		3V	V_{OH}=0.7VDD	-	8	-	mA
I_{OH2}	Output port source current RB0~RB4	5V	V_{OH}=0.7VDD	-	34	-	mA
		3V	V_{OH}=0.7VDD	-	12	-	mA
V_{BG}	Internal reference voltage 1.2V	VDD=2.5~5.5V T_A=25˚C		-1.5%	1.2	1.5%	V
		VDD=2.5~5.5V T_A=－40~85˚C		-2.0%	1.2	2.0%	V

19.3 ADC internal LDO reference voltage characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Item	Test condition	Min.	Typ.	Max.	Unit
AD_{VREF1}	LDO=2.0V voltage temperature characteristics	VDD=5V T_A=25˚C	-0.6%	2.0	+0.6%	V
		VDD=2.7~5.5V T_A=25˚C	-1.0%	2.0	+1.0%	V
		VDD=2.7~5.5V T_A=-40˚C~85˚C	-1.5%	2.0	+1.5%	V
AD_{VREF2}	LDO=2.4V voltage temperature characteristics	VDD=5V T_A=25˚C	-0.6%	2.4	+0.6%	V
		VDD=2.7~5.5V T_A=25˚C	-1.0%	2.4	+1.0%	V
		VDD=2.7~5.5V T_A=-40˚C~85˚C	-1.5%	2.4	+1.5%	V

19.4 OPA electrical characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Item	Test condition	Min.	Typ.	Max.	Unit
DC electrical characteristics						
VDD	Operating voltage		2.0	-	5.5	V
I_{DD}	Static current	VDD=5.0V	-	380	-	uA
V_{OPOS}	Input offset voltage	Default value	-10	-	10	mV
		After zeroing	-2	-	2	mV
V_{CM}	Common mode voltage range	-	0.1	-	VDD-1.4V	V
PSRR	Power supply rejection ratio	-	60	70	-	dB
CMRR	Common mode rejection ratio	VDD=5V V_{CM}=0.1V~VDD-1.4V	90	100	-	dB
AC electrical characteristics						
A_{OL}	Open loop gain	-	90	100	-	dB
GBW	Gain bandwidth	R_L=1MΩ, C_L=100pF	1.5	2	-	MHz

19.5 LVR electrical characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Item	Test condition	Min.	Typ.	Max.	Unit
V_{LVR1}	LVR set voltage=1.8V	VDD=1.6~5.5V	1.6	1.8	2.0	V
V_{LVR2}	LVR set voltage=2.0V	VDD=2.2~5.5V	1.8	2.0	2.2	V
V_{LVR3}	LVR set voltage=2.6V	VDD=2.2~5.5V	2.4	2.6	2.8	V
V_{LVR4}	LVR set voltage=3.5V	VDD=2.5~5.5V	3.3	3.5	3.7	V

19.6 AC electrical characteristics

(T_A= 25˚C, unless otherwise specified)

Symbol	Item	Test condition		Min.	Typ.	Max.	Unit
		VDD	Condition				
T_{WDT}	WDT reset time	5V	-		16		ms
		3V	-		32		ms
T_{EEPROM}	EEPROM programming time	5V	F_{HSI}=8MHz			5	ms
		3V	F_{HSI}=8MHz			5	ms
		5V	F_{HSI}=16MHz			5	ms
		3V	F_{HSI}=16MHz			5	ms
F_{RC}	Internal oscillation frequency stability	VDD=4.0~5.5V	TA=25˚C	-1.5%	8	+1.5%	MHz
		VDD=2.5~5.5V	TA=25˚C	-2.0%	8	+2.0%	MHz
		VDD=4.0~5.5V	TA= - 40~85˚C	-2.5%	8	+2.5%	MHz
		VDD=2.5~5.5V	TA= - 40~85˚C	-3.5%	8	+3.5%	MHz
		VDD=1.8~5.5V	TA= - 40~85˚C	-5.0%	8	+5.0%	MHz
		VDD=4.0~5.5V	TA=25˚C	-1.5%	16	+1.5%	MHz
		VDD=2.6~5.5V	TA=25˚C	-2.0%	16	+2.0%	MHz
		VDD=4.0~5.5V	TA= - 40~85˚C	-2.5%	16	+2.5%	MHz
		VDD=2.6~5.5V	TA= - 40~85˚C	-3.5%	16	+3.5%	MHz

20. Instructions

20.1 Instruction set

mnemonic		operation	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
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				

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

20.2 Instruction description

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 i 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 to 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: I minus ACC, 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, the result is put into 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
SUBCR   R01     ;operation result: R01=80H-77H-C=09H(C=0)
                                   R01=80H-77H-C=08H(C=1)
```

SWAPA **[R]**

operation: Register R high and low half byte swap, the save result into ACC

period: 1

Affected flag bit: none

example:

```
LDIA    035H    ;load 35H to ACC
LD      R01,A   ;load ACC to R01, R01=35H
SWAPA   R01     ;operation result: ACC=53H
```

SWAPR **[R]**

operation: Register R high and low half byte swap, the save result into R

period: 1

Affected flag bit: none

example:

```
LDIA    035H    ;load 35H to ACC
LD      R01,A   ;load ACC to R01, R01=35H
SWAPR   R01     ;operation result: R01=53H
```

SZB **[R],b**

operation: Determine the bit b of register R, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

Affected flag bit: none

example:

```
SZB     R01,3     ;determine 3rd bit of R01
JP      LOOP      ;if is 1, execute, jump to LOOP
JP      LOOP1     ;if is 0, jump, execute, jump to LOOP1
```

SNZB **[R],b**

operation: Determine the bit b of register R, if it is 1 then jump, otherwise execute in sequence

period: 1 or 2

Affected flag bit: none

example:

```
SNZB    R01,3     ;determine 3rd bit of R01
JP      LOOP      ;if is 0, execute, jump to LOOP
JP      LOOP1     ;if is 1, jump, execute, jump to LOOP1
```

SZA **[R]**

operation: Load the value of R to ACC, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

Affected flag bit: none

example:

```
SZA     R01       ;R01→ACC
JP      LOOP      ;if R01 is not 0, execute, jump to LOOP
JP      LOOP1     ;if R01 is 0, jump, execute, jump to LOOP1
```

SZR **[R]**

operation: Load the value of R to R, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

Affected flag bit: none

example:

```
SZR     R01       ;R01→R01
JP      LOOP      ;if R01 is not 0, execute, jump to LOOP
JP      LOOP1     ;if R01 is 0, jump, execute, jump to LOOP1
```

SZINCA **[R]**

operation: Increment register by 1, save the result to ACC, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

Affected flag bit: none

example:

```
SZINCA   R01     ;R01+1→ACC
JP       LOOP    ;if ACC is not 0, execute, jump to LOOP
JP       LOOP1   ;if ACC is 0, jump, execute, jump to LOOP1
```

SZINCR **[R]**

operation: Increment register by 1, save the result to R, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

Affected flag bit: none

example:

```
SZINCR   R01     ;R01+1→R01
JP       LOOP    ;if R01 is not 0, execute, jump to LOOP
JP       LOOP1   ;if R01 is 0, jump, execute, jump to LOOP1
```

SZDECA **[R]**

operation: decrement register by 1, save the result to ACC, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

Affected flag bit: none

example:

```
SZDECA   R01     ;R01-1→ACC
JP       LOOP    ;if ACC is not 0, execute, jump to LOOP
JP       LOOP1   ;if ACC is 0, jump, execute, jump to LOOP1
```

SZDECR **[R]**

operation: Decrement register by 1, save the result to R, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

Affected flag bit: none

example:

```
SZDECR   R01     ;R01-1→R01
JP       LOOP    ;if R01 is not 0, execute, jump to LOOP
JP       LOOP1   ;if R01 is 0, jump, execute, jump to LOOP1
```

TABLE **[R]**

operation: Look-up table, the low 8 bits of the table check result are put into R, and the high bits are put into the special register TABLE_SPH

period: 2

Affected flag bit: none

example:

```
LDIA    01H             ;load 01H to ACC
LD      TABLE_SPH,A     ;load ACC to higher bits of table address, TABLE_SPH=1
LDIA    015H            ;load 15H to ACC
LD      TABLE_SPL,A     ; load ACC to lower bits of table address, TABLE_SPL=15H
TABLE   R01             ;look-up table 0115H address, operation result: TABLE_DATAH=12H, R01=34H
…
ORG     0115H
DW      1234H
```

TABLEA

operation: Look-up table, the low 8 bits of the table check result are put into ACC, and the high bits are put into the special register TABLE_SPH

period: 2

Affected flag bit: none

example:

```
LDIA    01H             ;load 01H to ACC
LD      TABLE_SPH,A     ;load ACC to higher bits of table address, TABLE_SPH=1
LDIA    015H            ;load 15H to ACC
LD      TABLE_SPL,A     ; load ACC to lower bits of table address, TABLE_SPL=15H
TABLEA                  ;look-up table 0115H address, operation result: TABLE_DATAH=12H, ACC=34H
…
ORG     0115H
DW      1234H
```

TESTZ **[R]**

operation: Pass the R to R, as affected Z flag bit

period: 1

Affected flag bit: Z

example:

```
TESTZ   R0              ;Pass the value of register R0 to R0, which is used to influence the Z flag bit
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 R and ACC, save the result to R

period: 1

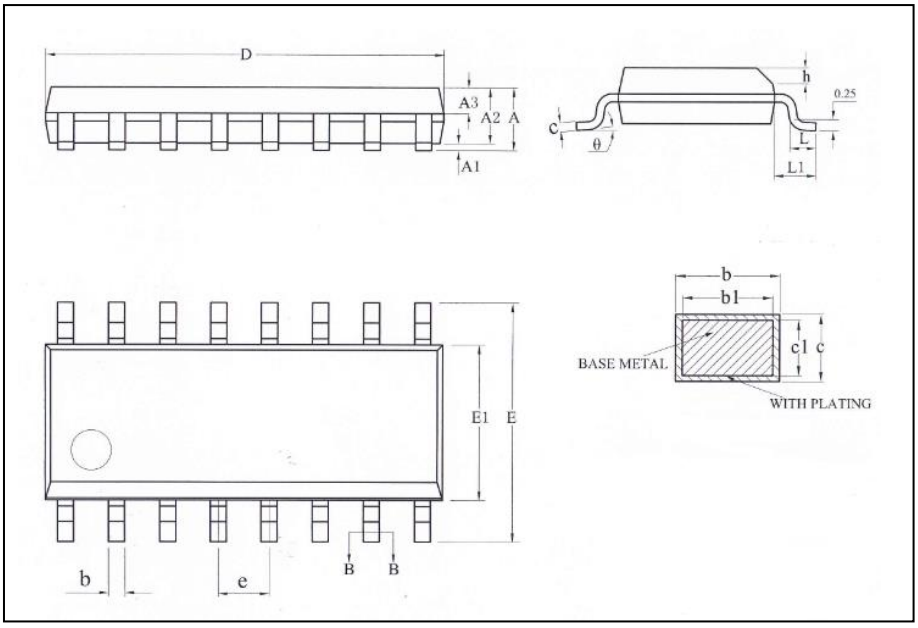
Affected flag bit: Z

example:

```
LDIA    0AH     ;load 0AH to ACC
LD      R01,A   ;load ACC to R01, R01=0AH
LDIA    0FH     ;load 0FH to ACC
XORR    R01     ;execute: R01=05H
```

21. Package

21.1 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. Revision History

Version	Date	Revision Content
V1.0	March 2020	Initial version.
V1.1	April 2020	Modified the description of touch channels in the section “Top View”.
V1.2	May 2020	Modified the description of the ADC chapter clock to include a description of 16MHz.
V1.3.0	Dec 2023	1) Corrected RCIF and TXIF to read-only in the PIR1 register. 2) Corrected the description of TXIF in the PIR1 register. 3) Corrected the asynchronous transmission diagrams, Figure 15-3 and Figure 15-4. 4) Corrected the receive interrupt description in section 15.1.2.3 of the UART. 5) Corrected the FERR frame error bit to read-only in the RCSTA register. 6) Added clock diagram. 7) Corrected the on-chip high-speed oscillation frequency to FHSI and corrected the clock source for other modules according to the clock diagram. 8) Corrected the sleep wake-up waiting time and ADC internal LDO reference voltage characteristics. 9) Removed the description of sleep wake-up in the ADC interrupt.