CMS79F72x User Manual

Enhanced flash 8-bit CMOS microcontroller

Rev. 1.2

Index

1. Product overview

1.1 Features

- memory
  - Flash: 8Kx16
  - General RAM: 256x8
  - Built-in touch RAM: 240x8
- Level 8 stack buffer
- Simple and practical command system (68 instructions)
- Lookup table function
- Built-in WDT timer
- Built-in low-voltage detection circuit
- Interrupt source
  - 3 timer interrupts
  - The PORTA/PORTB port level change is interrupted
  - Other peripherals are interrupted
- Timer
  - 8-bit timer TIMER0, TIMER2
  - 16-bit timer TIMER1
- Capture,Conpare,and PWM module(CCP)
  - 10-bit PWM accuracy
  - 2-way PWM can set independent cycle and duty cycle
  - Configurable on RB0/RB1 or RA3/RA4
- Built-in 128-byte program EEPROM
  - Can be rewritten 100,000 times
- Operating voltage range: 2.6V ~ 5.5V@16MHz
  1.8V ~ 5.5V@8MHz
- Operating temperature range: -40°C~85°C
- Built-in high-precision 8MHz/16MHz oscillation
- Instruction period (single or double instruction)
- IO characteristics
  - PORTA/PORTB High Level Driving Current 16 Optional 0~30mA, Step to 2mA
  - PORTA/PORTB Low Level Driving Current 2 Optional 60/120mA
  - PORTA/PORTB support level change wake-up
  - Built-in pull-up resistance for all IO ports
  - PORTB Port Built-in pull-down resistance
- Built-in 1-channel USART communication module
  - Support synchronous master-slave mode and asynchronous full-duplex mode
  - Configurable on RA3/RA4 or RC0/RC1
- - High Precision 12th ADC
  - Built-in high accuracy 1.2V reference voltage
  - ±1.5%@VDD=2.5V~5.5V TA=25°C
  - ±2%@VDD=2.5V~5.5V TA=-40°C~85°C
- Built-in touch key detection module
  - No need for external touch capacitor
  - Easy to pass dynamic and static 10V conduction test
  - Touch sensitivity adjustable
  - All pins can be configured as touch channels

Model description

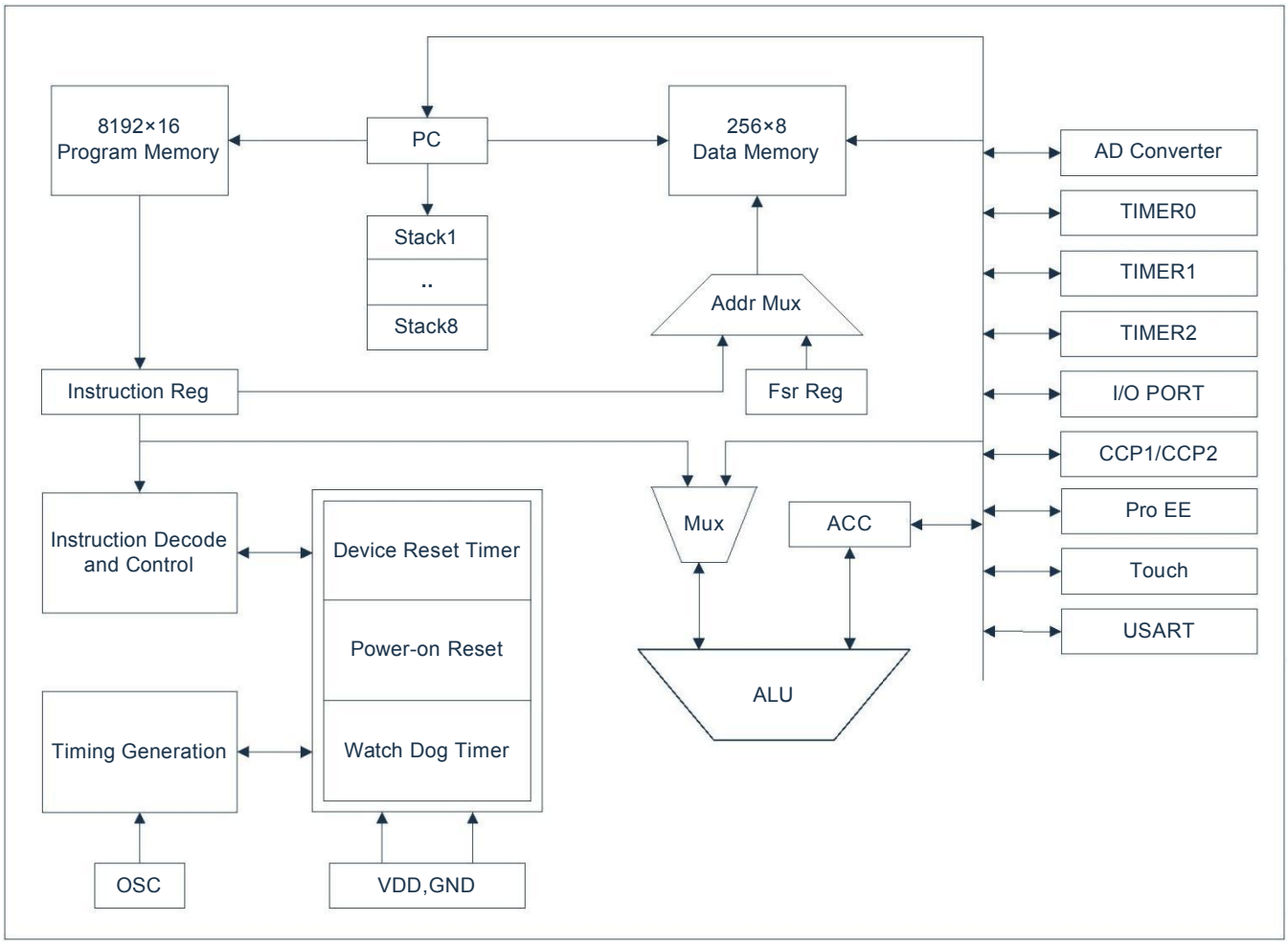
PRODUCT	ROM	RAM	Pro EE	I/O	ADC	Touch	USART	PACKAGE
CMS79F723	8Kx16	256x8	128x8	14	12Bitx14	14	1	SOP16
CMS79F726	8Kx16	256x8	128x8	18	12Bitx18	16	1	SOP20, TSSOP20

Note: (1) ROM ---- program memory Pro EE ---- program EEPROM

(2) CMS79F723 Touch library: 14, EMC library: 8, Sleep library: 12

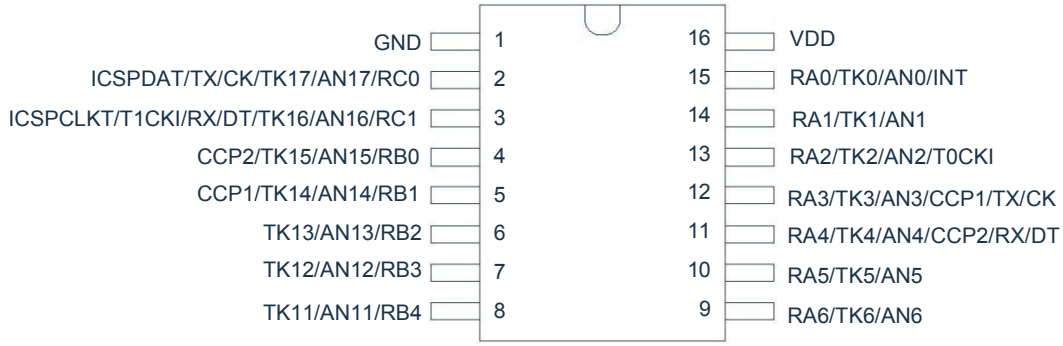
(3) CMS79F726 Touch library: 16, EMC library: 8, Sleep library: 14

1.2 System block diagram



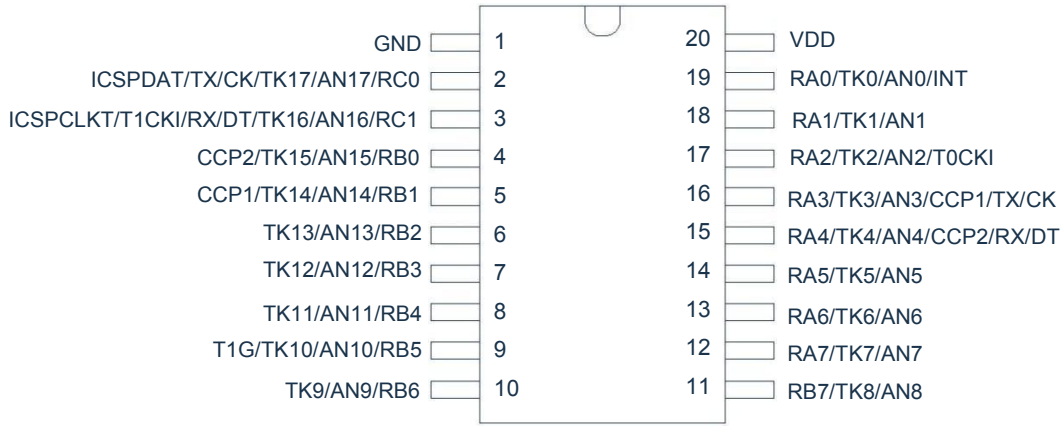
1.3 Pin out

1.3.1 CMS79F723 pinout



CMS79F723

1.3.2 CMS79F726 pinout



CMS79F726

Note:
1) Serial port functions of RA4 and RA3 are complemented by CONFIG with serial ports of RC0 and RC 1 Settings;
2) The CCP functions of R A4 and R A3 are compatible with those of RB0 and RB1 The CCP function is set by CONFIG.

Pin Description:

Pin name	IO type	Pin description
VDD,GND	P	Power voltage input pin, ground pin
RA0-RA7	I/O	Programmable as input pin, push-pull output pin, with pull-up resistor function, level change interrupt function
RB0-RB7	I/O	Programmable as input pin, push-pull output pin, pull-up resistor function, pull-down resistor function, level change interrupt function
RC0-RC1	I/O	Programmable as input pin, push-pull output pin, with pull-up resistor function
ICSPCLK/ICSPDAT	I/O	Programming clock/data pins
TK0-TK17	-	Touch key input pin
AN0-AN17	I	12-bit ADC input pin
T0CKI	I	TIMER0 external clock input pin
T1CKI	I	TIMER1 external clock input pin
T1G	I	TIMER1 gating input pin
CCP1	I/O	Capture/Compare/PWM1
CCP2	I/O	Capture/Compare/PWM1
TX/CK	I/O	USART asynchronously sends output/synchronous clock input/output pins
RX/DT	I/O	USART asynchronously receives input/synchronous data input/output pins

1.4 System configuration registers

The System Configuration Register (CONFIG) is a FLASH option for the initial MCU condition. It can only be programmed by cmS burners and cannot be accessed and operated by users. It contains the following contents:

1. INTRC_SEL (Internal Oscillation Frequency Selection).
   - INTRC8M Fosc Selects Internal 8MHz RC Oscillation
   - INTRC16M Fosc Selects Internal 16MHz RC Oscillation
2. WDT (Watchdog Selection)
   - ENABLE Turn on the watchdog timer
   - DISABLE Turn off the watchdog timer
3. PROTECT (encrypt)
   - DISABLE Flash code is not encrypted
   - ENABLE Flash code encryption, after encryption, the value read out by the flash emulator will be uncertain
4. LVR_SEL (low voltage detection voltage selection)
   - 1.8V
   - 2.0V
   - 2.6V
5. ICSPPORT _SEL (emulation port function selection).
   - ICSP The ICSPCLK and DAT ports have always been maintained as emulation ports, and all functions cannot be used
   - NORMAL ICSPCLK and DAT ports are ordinary functional ports
6. USART_SEL (TX/RX) (USART port selection)I
   - RC0/RC1 Select RC0 for TX port and RC0 for RX port
   - RA3/RA4 Select RA3 for the TX port and RA4 for the RX port
7. CCP_SEL (CCP port selection).
   - RA3/RA4 Select RA3 for the CCP2 port and RA4 for the CCP2 port
   - RB1/RB0 Select RAB1 for the CCP1 port and RB0 for the CCP2 port

1.5 Online serial programming

Microcontrollers can be programmed serially in end application circuits. Programming can be done simply by the following 4 wires:

- Power cord
- Ground wire
- Data cable
- Clock line

This enables users to manufacture circuit boards using unprogrammed devices and to program microcontrollers only before product delivery. This enables the latest version of firmware or custom firmware to be programmed to a microcontroller.

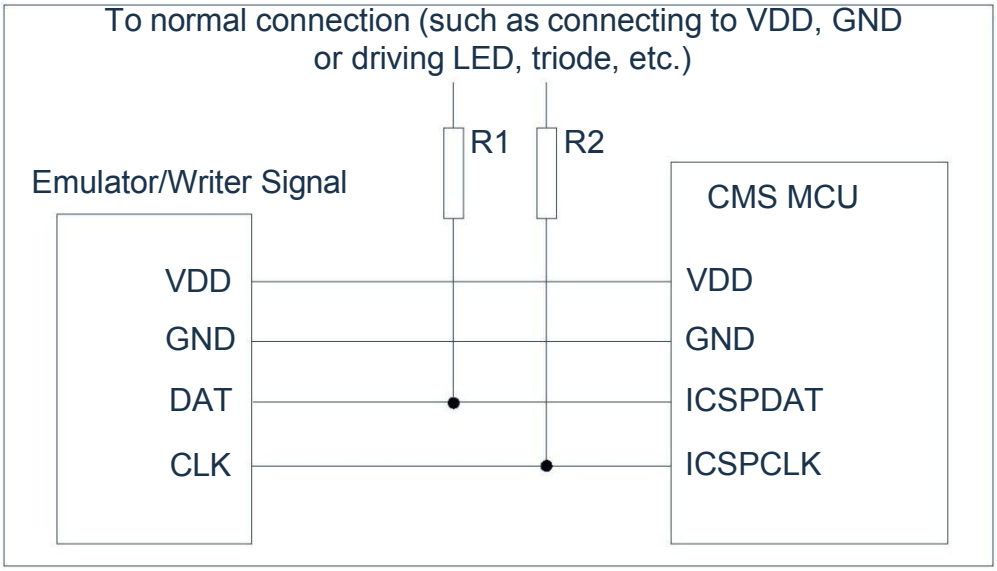


Figure 1-1: Typical online serial programming connection method

In the figure above, R1 and R2 are galvanical isolation devices, often replaced by resistors, and their resistance values are as follows: R1≥4.7K, R2≥4.7K.

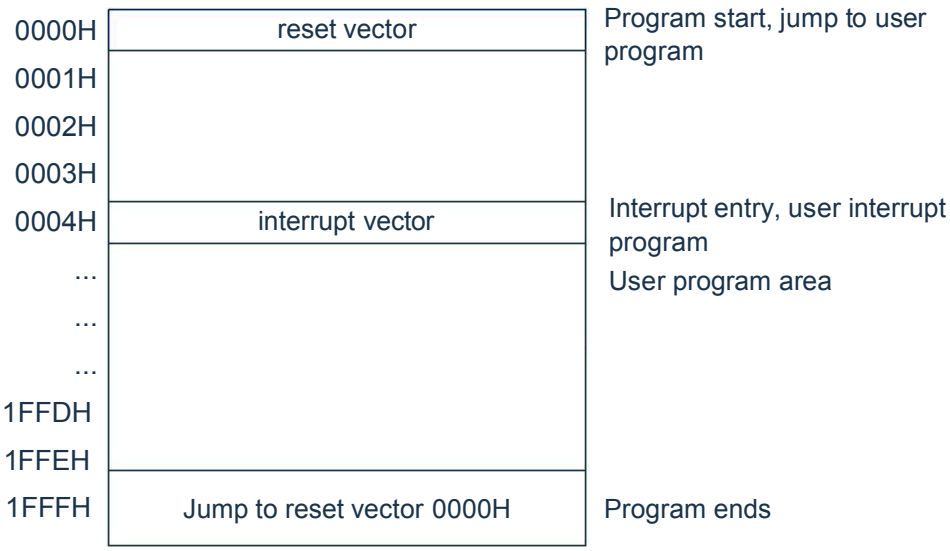
2. Central processing unit (CPU)

2.1 memory

2.1.1 Program memory

CMS79F72x program memory space

FLASH:8K



2.1.1.1 Reset vector (0000H).

The microcontroller has a word-long system reset vector (0000H). It has the following 3 reset methods:

- Power-on reset
- Watchdog reset
- Low Voltage Reset (LVR).

After any of the above resets occur, the sequence will resume at 0000H and the system registers will revert to default values. Based on the contents of the PD and TO marker bits in the STATUS register, the system reset mode can be determined. The following procedure shows how to define a reset vector in FLASH.

Example: A normalized reset vector

```
        ORG     0000H       ; System reset vector
        JP      START
        ORG     0010H       ; The user program starts
START:
        …                   ; User programs
        …
        END                 ; The program ends
```

2.1.1.2 Interrupt vector

The interrupt vector address is 0004H. Once there is a interrupt response, the current value of the program counter PC is stored in the stack buffer and jumped to 0004H began to carry out the service routine. All the interrupts will enter the 0004H interrupt vector, which is specifically executed The break will be determined by the bit of the user based on the bit of the bit register of the interrupt request. The following sample program shows how to write a interrupt service routine.

Example: Define interrupt vector, in which the interrupt routine is placed after the user program

```
            ORG     0000H       ; System reset vector
            JP      START
            ORG     0004H       ; The user program starts
INT_START:
            CALL    PUSH        ; Save the ACC with STATUS
            …                   ; The user interrupts the program
            …
INT_BACK:
            CALL    POP         ; Returns ACC followed by STATUS
            NETWORKS            ; Interrupt returns
START:
            …                   ; User programs
            …
            END                 ; The program ends
```

Note: Because the microcontroller does not provide push or pop stack instructions, users need to protect the context of the interrupt.

Example: Interrupt entrance context protection

```
PUSH:
            LD      ACC_BAK,A       ; Save the ACC to a custom register ACC_BAK
            SWAPA   STATUS          ; Status registers STATUS are high and low half-byte interchangeable
            LD      STATUS_BAK,A    ; Save to a custom register STATUS_BAK
            RIGHT                   ; return
```

Example: Interrupt exit context recovery

```
POP:
            SWAPA   STATUS_BAK      ; Swap the data saved to STATUS_BAK with nibbles to ACC
            LD      STATUS,A        ; Give the value of the ACC to the status register STATUS
            SWAPR   ACC_BAK         ; The data saved to the ACC_BAK is swapped high and low by half a byte
            SWAPA   ACC_BAK         ; Swap the data saved to the ACC_BAK with a high and low nibble to the ACC
            RIGHT                   ; return
```

2.1.1.3 Look up table

The chip has a lookup table function, and any address in the FLASH space can be used as a lookup table.

Related Instructions:

- TABLE [R] Send the low sections of the table to register R, and the high sections to the register TABLE_DATAH.
- TABLEA Send the low sections of the table to the accumulator ACC, and the high sections to the register TABLE_DATAH.

Correlation register:

- TABLE_SPH(110H) Read-write registers to indicate a table 5-bit high address.
- TABLE_SPL(111H) Read-write registers to indicate 8-bit lower table addresses.
- TABLE_DATAH(112H) Read-only registers for table high section contents.

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

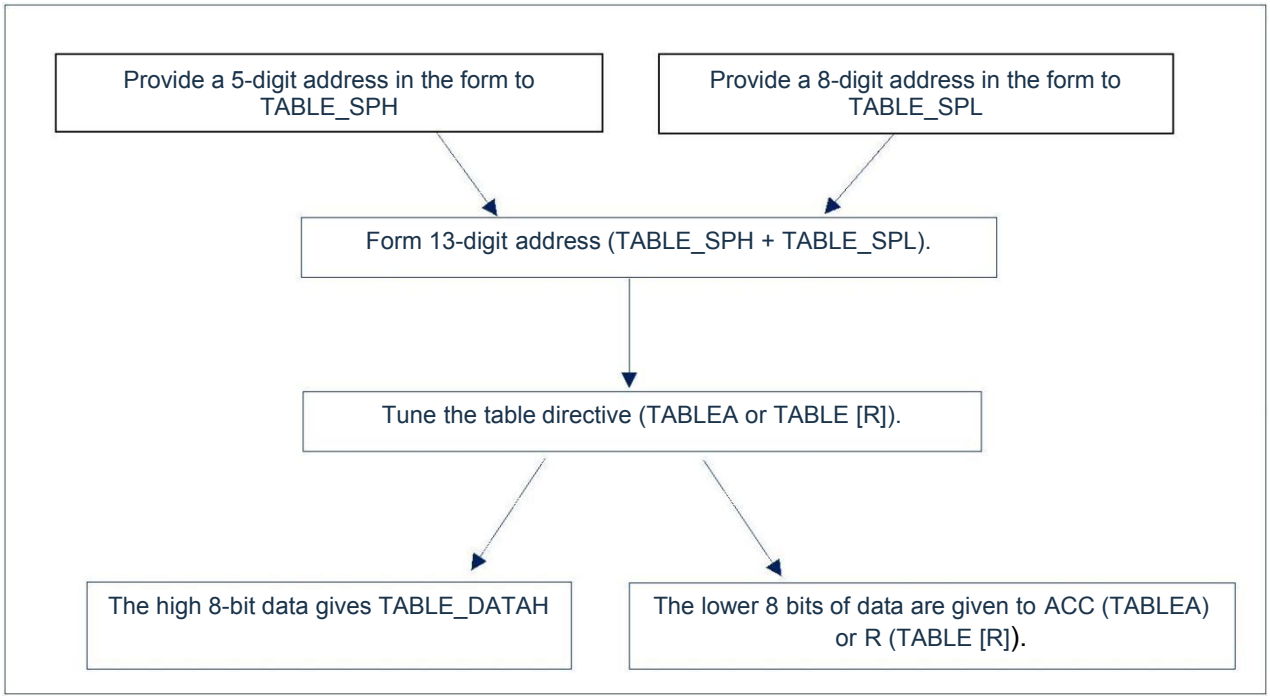


Figure 2-1: Flowchart of table calls

The following example shows how to use tables in a program.

```
…                                   ; Pick up the user program
LDIA    02H                         ; Table low-bit addresses
LD      TABLE_SPL,A
LDIA    06H                         ; Table high-order addresses
LD      TABLE_SPH,A
TABLE   R01                         ; Table instruction, give the table 8 bits low (56H) to
                                    the custom register R01
LD      A,TABLE_DATAH               ; Give the high 8 bits (34H) of the lookup results to the
                                    accumulator ACC
LD      R02,A                       ; Give the ACC value (34H) to the custom register
                                    R02
…                                   ; User programs

ORG     0600H                       ; The table start address
DW      1234H                       0600H address table contents
DW      2345H                       0601H address table contents
DW      3456H                       0602H address table contents
DW      0000H                       0603H address table contents
```

2.1.1.4 Jump table

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

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

example: correct illustration of multi-address jump

```
Flash address
                LDIA        01H
                LD          PCLATH,A        ; PCLATH must be assigned a value
                …
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 the 0000H address
0101H:          JP          LOOP5           ; ACC=4, jump to the 0001H address
0102H:          JP          LOOP6           ; ACC=5, jump to address 0002H
```

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

Lift of CMS79F72x data memory

	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
DOOR	05H	CHEAT	85H	----	105H	----	185H
PORTB	06H	TRISB	86H	PORTB	106H	TRISB	186H
PORTC	07H	TRISC	87H	WPUA	107H	----	187H
----	08H	----	88H	WPUC	108H	----	188H
ANSEL0	09H	ANSEL1	89H	ANSEL2	109H	----	189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
PIR1	0CH	PIE1	8CH	EEDAT	10CH	----	18CH
PIR2	0DH	PIE2	8DH	EEADR	10DH	----	18DH
TMR1L	0EH	----	8EH	EEDATH	10EH	----	18EH
TMR1H	0FH	OSCCON	8FH	EEADRH	10FH	----	18FH
T1CON	10AM	WDTCON	90H	TABLE_SPH	110H	----	190H
TMR2	11AM	IOCA	91H	TABLE_SPL	111H	----	191H
T2CON	12H	PR2	92H	TABLE_DATAH	112H	----	192H
----	1PM	----	93H	SEGCUR	113H	----	193H
----	2PM	----	94H	PALEN	114H	----	194H
CCPR1L	3pm	WPUB	95H	PBLEN	115H	----	195H
CCPR1H	4 p.m.	IOCB	96H	PAHEN	116H	----	196H
CCP1CON	5PM	WPDB	97H	PBHEN	117H	----	197H
RCSTA	6PM	----	98H	----	118H	----	198H
TXREG	7PM	PWMCON	99H	----	119H	----	199H
RCREG	1AH	PWM1CYC	9AH	----	11AH	----	19AH
CCPR2L	1BH	PWM2CYC	9BH	EECON1	11BH	----	19BH
CCPR2H	1CH	ADRESL	9CH	EECON2	11CH	----	19CH
CCP2CON	1DH	ADRESH	9DH	----	11DH	----	19DH
TXSTA	1EH	ADCON0	9EH	----	11EH	----	19EH
SPBRG	1FH	ADCON1	9FH	----	11FH	----	19FH
Universal registers 96 bytes	8pm 6FH	Universal registers 80 bytes	A0H DETACHED HOUSE	Universal registers 80 bytes	120H 16FH	----	1A0H 1EFH
	70H -- 7FH	Fast storage 70H-7FH	F0H -- FFH	Fast storage 70H-7FH	170H -- 17FH	Fast storage 70H-7FH	1F0H -- 1FFH
BANK0		BANK1		BANK2		BANK3	

Data memory consists of 512×8 bits. It can be divided into to space: special function register and universal data memory. Most of data memory are able to write/read data, only some data memory are read-only. Special register address is from 00H-1FH，80-9FH，100-11FH，180-18BH.

CMS79F72x Special Function Register Summary Bank0

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00H	INDF	Addressing The cell uses the FSR's content-addressed data memory (not the physical registers).								xxxxxxxx
01H	TMR0	TIMER0 data register								xxxxxxxx
02H	PCL	Program counter low bytes								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	With	DC	C	00011xxx
04H	FSR	Indirect datastore address pointer								xxxxxxxx
05H	DOOR	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxxxxxx
06H	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxxxxxx
07H	PORTC	----	----	----	----	----	----	RC1	RC0	------xx
09H	ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	00000000
0AH	PCLATH	----	---	----	Program counter with a write buffer 5 bits high					---00000
0BH	INTCON	GIE	PEIE	T0IE	NOT	RBIE	T0IF	INTF	RBIF	00000000
0CH	PIR1	----	ADIF	RCIF	TXIF	----	CCP1IF	TMR2IF	TMR1IF	-000-000
0DH	PIR2	----	TKIF	RACIF	EEIF	----	----	----	CCP2IF	-000---0
0EH	TMR1L	16-bit TIMER1 registers Low-byte data registers								xxxxxxxx
0FH	TMR1H	16-bit TIMER1 register high-byte data 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
15H	CCPR1L	Capture/Compare/PWM Register 1 low byte								xxxxxxxx
16H	CCPR1H	The high byte of capture/compare/PWM register 1								xxxxxxxx
17H	CCP1CON	----	----	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--000000
18H	RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	HELL	OERR	RX9D	00001000
19H	TXREG	USART sends data registers								00000000
1AH	RCREG	USART receives data registers								00000000
1BH	CCPR2L	Capture/Compare/PWM Register 2 low byte								xxxxxxxx
1CH	CCPR2H	The high byte of capture/compare/PWM register 2								xxxxxxxx
1DH	CCP2CON	----	----	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--000000
1EH	TXSTA	CSRC	TX9EN	TXEN	SYNC	SCKP	----	TRMT	TX9D	00000-10
1FH	SPBRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0	00000000

CMS79F72x Special Feature Register Summary Bank1

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
80H	INDF	Addressing This address unit uses the FSR's content-addressed data memory (not physical registers).								xxxxxxxx
81H	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	11111011
82H	PCL	The low byte of the program counter (PC).								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	With	DC	C	00011xxx
84H	FSR	Indirect datastore address pointer								xxxxxxxx
85H	CHEAT	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	11111111
86H	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	11111111
87H	TRISC	----	----	----	----	----	----	TRISC1	TRISC0	------11
89H	ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8	00000000
8AH	PCLATH	----	----	----	Program counter with a write buffer 5 bits high					---00000
8BH	INTCON	GIE	PEIE	T0IE	NOT	RBIE	T0IF	INTF	RBIF	00000000
8CH	PIE1	----	ADIE	RCIE	TXIE	----	CCP1IE	TMR2IE	TMR1IE	-000-000
8DH	PIE2	----	TKIE	RACIE	EEIE	----	----	----	CCP2IE	-000---0
8FH	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	----	-110----
90H	WDTCON	----	----	----	----	----	----	----	SWDTEN	-------0
91H	IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0	00000000
92H	PR2	TIMER2 cycle register								11111111
95H	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	00000000
96H	IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0	00000000
97H	WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0	00000000
99H	PWMCON	----	CYC2EN	CK2[1:0]		----	CYC1EN	CK1[1:0]		-000-000
9AH	PWM1CYC	PWM1 periodic data register								11111111
9BH	PWM2CYC	PWM2 periodic data register								11111111
9CH	ADRESL	The low byte of the A/D result register								xxxxxxxx
9DH	ADRESH	The high byte of the A/D result register								xxxxxxxx
9EH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	ADON	00000000
9FH	ADCON1	ADFM	CHS4	----	----	----	----	----	----	00------

CMS79F72x Special Feature Register Summary Bank2

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
100H	INDF	Addressing This address unit uses the FSR's content-addressed data memory (not physical registers).								xxxxxxxx
101H	TMR0	TIMER0 module register								xxxxxxxx
102H	PCL	The low byte of the program counter (PC).								00000000
103H	STATUS	IRP	RP1	RP0	TO	PD	With	DC	C	00011xxx
104H	FSR	Indirect datastore address pointer								xxxxxxxx
106H	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxxxxxx
107H	WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	00000000
108H	WPUC	----	----	----	----	----	----	WPUC1	WPUC0	------00
109H	ANSEL2	----	----	----	----	----	----	ANS17	ANS16	------00
10AH	PCLATH	----	----	---	Program counter with a write buffer 5 bits high					---00000
10BH	INTCON	GIE	PEIE	T0IE	NOT	RBIE	T0IF	INTF	RBIF	00000000
10CH	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx
10DH	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	00000000
10EH	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	xxxxxxxx
10FH	EEADRH	----	----	----	EEADRH4	EEADRH3	EEADRH2	EEADRH1	EEADRH0	---00000
110H	TABLE_SPH	----	----	----	The table is 5-digit high pointer					---xxxxx
111H	TABLE_SPL	Table low pointer								xxxxxxxx
112H	TABLE_DATAH	Table high-order data								xxxxxxxx
113H	SEGCUR	----	----	----	----	SEG_ISEL3	SEG_ISEL2	SEG_ISEL1	SEG_ISEL0	00000000
114H	PALEN	PALEN7	PALEN6	PALEN5	PALEN4	PALEN3	PALEN2	PALEN1	PALEN0	00000000
115H	PBLEN	PBLEN7	PBLEN6	PBLEN5	PBLEN4	PBLEN3	PBLEN2	PBLEN1	PBLEN0	00000000
116H	PAHEN	PAHEN7	PAHEN6	PAHEN5	PAHEN4	PAHEN3	PAHEN2	PAHEN1	PAHEN0	00000000
117H	PBHEN	PBHEN7	PBHEN6	PBHEN5	PBHEN4	PBHEN3	PBHEN2	PBHEN1	PBHEN0	00000000
11BH	EECON1	EEPGD	----	EETIME1	EETIME0	WRERR	WREN	WR	RD	0-00x000
11CH	EECON2	EEPROM controls register 2 (not physical register).								--------

CMS79F72x Special Feature Register Summary Bank3

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
180H	INDF	Addressing This address unit uses the FSR's content-addressed data memory (not physical registers)								xxxxxxxx
181H	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	11111011
182H	PCL	The low byte of the program (PC).								00000000
183H	STATUS	IRP	RP1	RP0	TO	PD	With	DC	C	00011xxx
184H	FSR	Indirect datastore address pointer								xxxxxxxx
186H	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB73	TRISB2	TRISB1	TRISB0	11111111
18AH	PCLATH	----	----	----	Program counter with a write buffer 5 bits high					---00000
18BH	INTCON	GIE	PEIE	T0IE	NOT	RBIE	T0IF	INTF	RBIF	00000000

2.2 Addressing mode

2.2.1 Direct addressing

Operate on RAM through accumulator (ACC)

example: pass the value in ACC to 30H register

```
LD      30H,A
```

example: pass the value in 30H register to ACC

```
LD      A,30H
```

2.2.2 Immediate addressing

Pass the immediate value to accumulator (ACC) .

example: pass immediate value 12H to ACC

```
LDIA    12H
```

2.2.3 Indirect addressing

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

example: application of FSR and INDF

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

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

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

2.3 stack

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

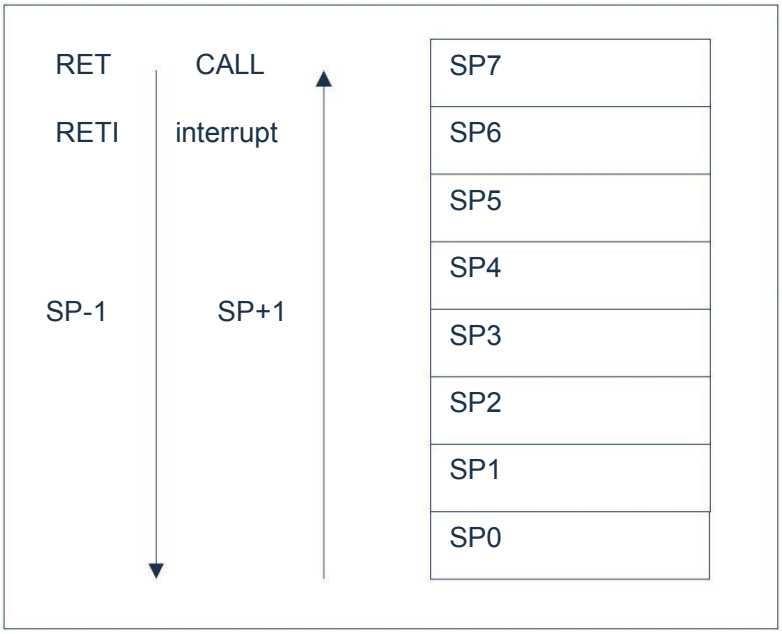


Figure 2-2: Stack memory processor works

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

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

2.4 accumulator (ACC)

2.4.1 overview

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

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

2.4.2 ACC applications

Example: use ACC for data transfer

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

example: use ACC for immediate addressing

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

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

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

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

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

2.5 Program status register (STATUS)

STATUS register includes:

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

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

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

Program status register STATUS(03H).

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

Bit7 IRP: Register memory selection bits (for indirect addressing);
1= Bank2和Bank3 (100h-1FFh) ;
0= Bank0和Bank1 (00h-FFh) 。

Bit6~Bit5 RP[1:0]: Bucket select bits;
00: Select Bank 0;
01: Select Bank 1;
10: Select Bank 2;
11: Select Bank 3.

Bit4 TO: Timeout bit;
1= Powered up or executed clrWDT instructions or STOP instructions;
0= A WDT timeout has occurred.

Bit3 PD: Potential drop;
1= The CLRWDT instruction was powered on or executed;
0= The STOP instruction was executed.

Bit2 With: The result is zero;
1= The result of an arithmetic or logical operation is zero;
0= The result of an arithmetic or logical operation is not zero.

Bit1 DC: Half-carry/debit position;
1= As a result, the low 4 bits occurred to carry to the high or the low 4 bits did not occur with a debit;
0= As a result, the low 4 bit did not occur to carry the high or the low 4 bit occurred to borrow the high.

Bit0 C: Carry/borrowed positions;
1= The highest level of the result occurred or did not occur debit;
0= The highest level of the result did not occur with a carry or a debit.

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

event	TO	PD
Power-on	1	1
WDT overflow	0	X
STOP instruction	1	0
CLRWDT Instruction	1	1
dormancy	1	0

A table of events that affect TO/PD

TO	PD	Reset the cause
0	0	WDT overflow wakes up the sleeping MCU
0	1	WDT overflows in a non-dormant state
1	1	Power-on

The state of the TO/PD after reset

2.6 Prescaler (OPTION_REG)

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

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

Prescaler OPTION_REG (81H).

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

Bit7 RBPU: PORTB pull-up enable bit.
1= PORTB pull-up is prohibited.
0= The PORTB pullup is enabled by the individual latch values of the port.
Bit6 INTEDG: The edge selection bit that triggered the interrupt.
1= The rising edge of the INT pin triggers an interrupt.
0= The falling edge of the INT pin triggers an interrupt.
Bit5 T0CS: TIMER0 clock source select bit.
0= Internal instruction cycle clock (FSYS/4).
1= The trip edge on the T0CKI pin.
Bit4 T0SE: TIMER0 clock source edge select bit.
0= Increments at the T0CKI pin signal when it jumps from low to high.
1= Increments when the T0CKI pin signal jumps from high to low.
Bit3 PSA: Prescaler allocation bits.
0= A prescaler is assigned to the TIMER0 module.
1= A prescaler is assigned to the WDT.
Bit2~Bit0 PS2~PS0: Preassigned parameter configuration bits.

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

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

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

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

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

```
CLRWDT                          ; WDT clears
LDIA        B'xxxx1xxx'         ; Set up a new prescaler
LD          OPTION_REG,A
CLRWDT                          ; WDT clears
```

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

```
CLRWDT                          ; WDT clears
LDIA        B'xxxx0xxx'         ; Set up a new prescaler
LD          OPTION_REG,A
```

Note: To get TIMER0 to obtain a 1:1 prescaler ratio configuration, the prescaler can be assigned to the WDT by assigning the PSA position 1 of the option register.

2.7 Program Counter (PC).

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

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

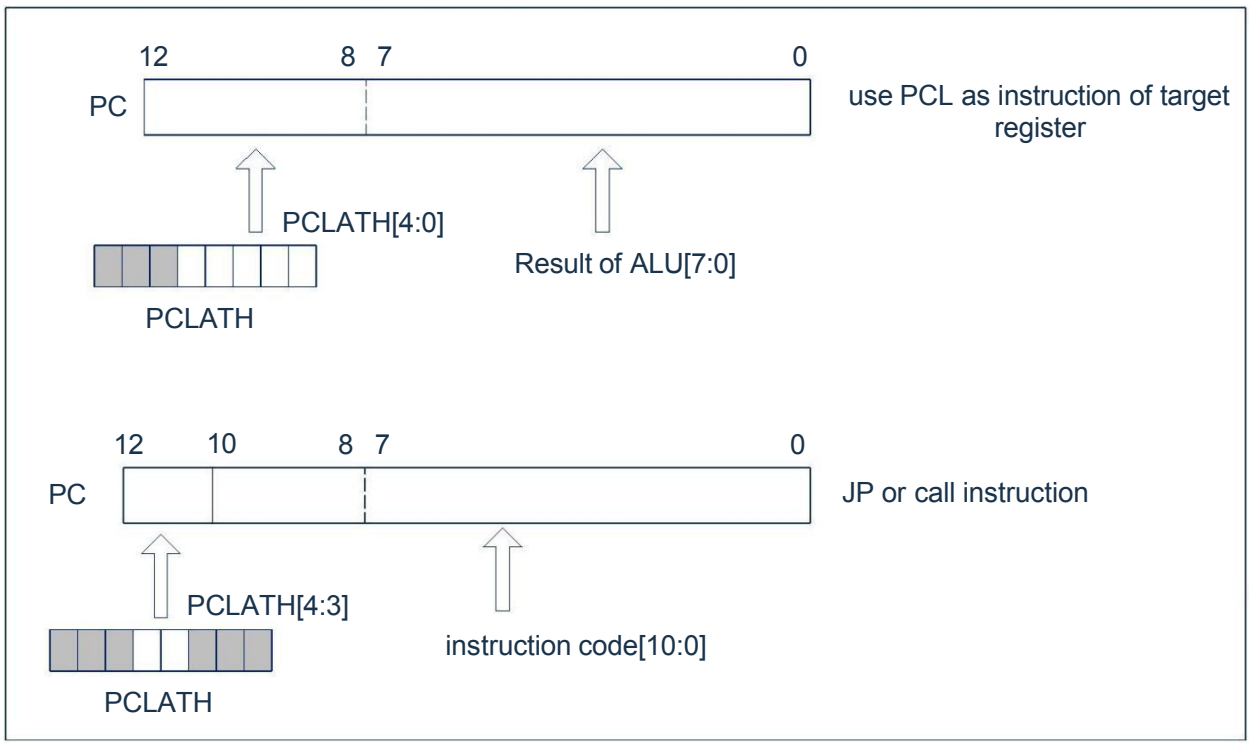
Program counter (PC) is 13 Bit，user can access lower 8 bits through PCL(02H). The higher 5 bits cannot be accessed. It can hold address for 8K×16Bit program. Passing a value to PCL will cause a short jump which range until the 256 address of the current page.

.

Note: When programmers use PCL for short jumps, they must first assign a value to the PC high-bit buffer register PCLATH.

The following are the value of PC under special conditions.

When resetting	PC=0000;
When interrupted	PC=0004 (the original PC+1 is automatically pressed into the stack);
When called	PC = program-specified address (the original PC+1 is automatically pushed into the stack);
RET, RETI, RETi i	PC = stack out of the value;
When operating the PCL	PC[12:8] unchanged, PC [7:0] = user-specified value;
JP time	PC= program-specified value;
Other instruction s	PC=PC+1;



The example program below gives considerations for using the JP or CALL instruction .

```
        ORG     00H
                                        The destination address LABEL1 is located at the 300H
        JP      LABEL1                  address, and the current PCLATH value is 00H, which is
                                        in the same 2K range, so there is no need to change the
                                        PCLATH value before executing the JP instruction
        …
        ORG     300H
LABEL1:
        LDIA    08H                     The destination address LABORATOR2 is located at the
        LD      PCLATH,A                900H address, and the current PCLATH value is 00H,
        JP      LABEL2                  which is not in the same 2K range, so before executing
                                        the JP instruction, you need to assign a value to pcLATH
        …
        ORG     7FEH
LABLE4:
        NOP                             ;7FEH
        NOP                             ;7FFH
        NOP                             ;800H
        LDIA    08H                     The destination address LABEL5 is located at the 880H
        LD      PCLATH,A                address, and the current PCLATH value is 00H (the
                                        program is running normally, when the PC changes from
        JP      LABLE5                  7FFH to 800H, the PCLATH value does not change with
                                        it), not in the same 2K range, so before executing the JP
                                        instruction, you need to assign a value to the PCLATH
        …
        ORG     880H
LABLE5:
        NOP
        RIGHT
        …
        ORG     900H
LABLE2:
        NOP
                                        The destination address LABEL3 is located at the E00H
        CALL    LABLE3                  address, and the current PCLATH value is 08H, in the
                                        same 2K range, so there is no need to change the
                                        PCLATH value before executing the CALL instruction
        LDIA    00H                     The destination address LABEL4 is located at the 7FEH
        LD      PCLATH,A                address, and the current PCLATH value is 08H, which is
        CALL    LABLE4                  not in the same 2K range, so before executing the CALL
        NOP                             instruction, you need to assign a value to PCLATH
        …
        …
        ORG     0E00H
LABLE3:
        NOP
        RIGHT
        …
```

2.8 watchdog timer (WDT)

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

2.8.1 WDT period

WDT and TIMER0 share 8-bit pre-scaler. After all reset, default overflow period of WDT is 144ms. The WDT overflow period calculation formula is 18ms*pre-scaling parameter. If WDT overflow period needs to be changed, you can configure OPTION_REG register. The overflow period is affected by environmental temperature, voltage of the power source and other parameter.

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

Note:

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

2.8.2 Watchdog timer control register WDTCON

Watchdog Timer Control Register WDTCON(90H)

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

Bit7~Bit1 Unused

Bit0 SWDTEN: The software enables or disables the watchdog timer bit.

1= Enable WDT.

0= DisableSwed WDT (Reset Value).

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

3. System clock

3.1 overview

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

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

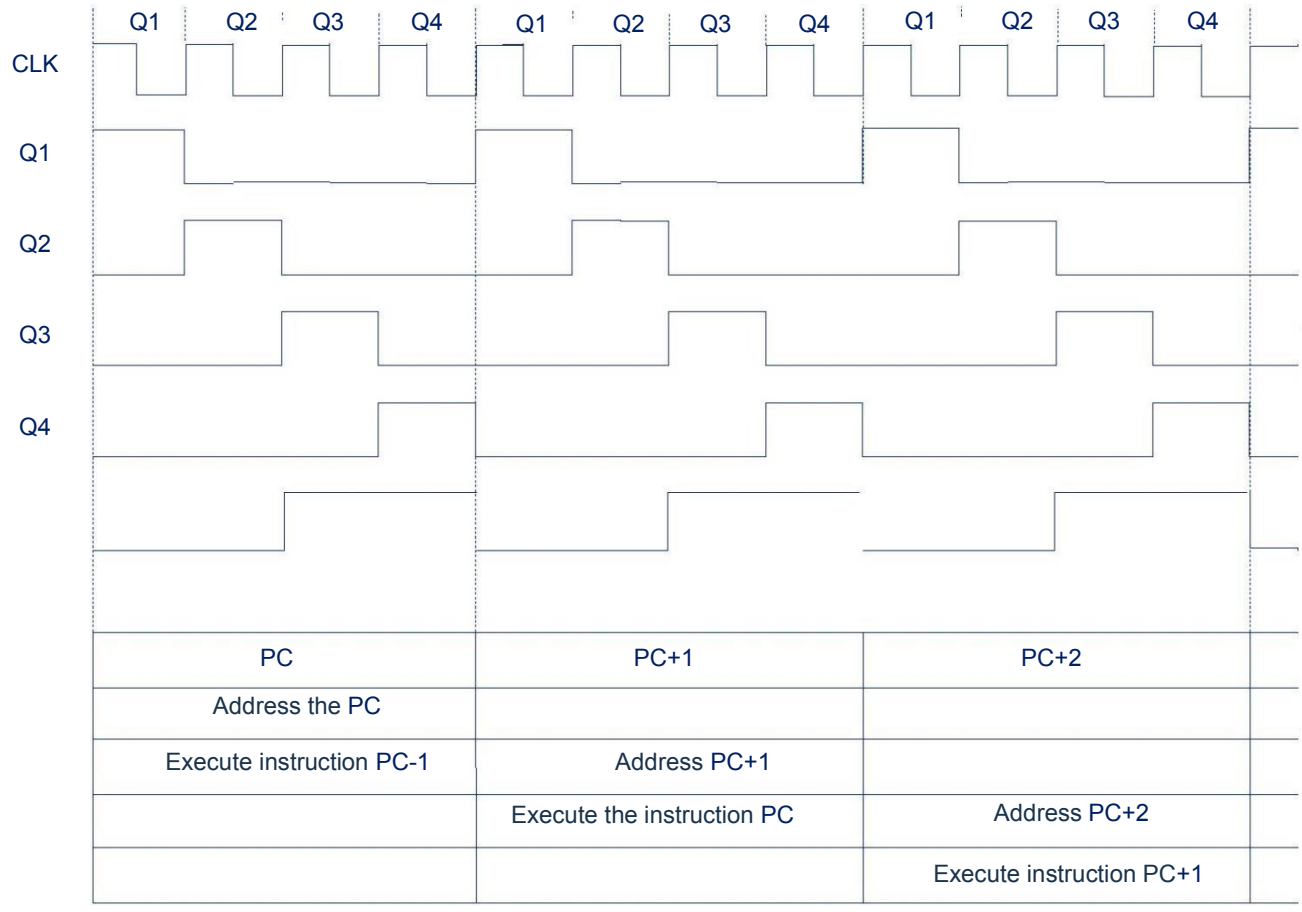


Figure 3-1: Clock and instruction cycle timing diagram

The following is a list of the relationship between the operating frequency rate of the system and the command speed:

System Operating Frequency Rate (FSYS).	Dual instruction cycle	Single instruction period
1MHz	8μs	4μs
2MHz	4μs	2μs
4MHz	2μs	1μs
8MHz	1μs	500ns

3.2 System oscillator

The chip is equipped with high-precision RC oscillation as a clock source with frequency of 8Mhz or 16Mhz, and the chip operating frequency can be set through OSCCON registers.

3.3 Reset time

Reset Time refers to the time from the reset of the chip to the stabilization of the chip oscillation, and its design value is about 18 ms。

Note: This Reset time will exist regardless of whether the chip is power-on reset or otherwise caused by a reset.

3.4 Oscillator control registers

Oscillator Control (OSCCON) registers control system clock and frequency selection.

Oscillator Control Register OSCCON (8FH)

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

Bit7 Unused

Bit6~Bit4 IRCF<2:0>: Internal oscillator crossover selection bit.

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

110= $F_{SYS} = F_{OSC}/2$ (default)

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

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

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

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

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

000= F_{SYS} = 32kHz (LFINTOSC) 。

Bit3~Bit0 Unused.

Note: F_{OSC} is the internal oscillator frequency 8MHz/16Mhz; F_{SYS} is the system operating frequency.

4. Reset

Chip has 3 ways of reset:

- power on reset;
- low voltage reset;
- watchdog overflow reset under normal working condition;

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

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

4.1 Power on reset

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

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

4.2 Power off reset

4.2.1 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.

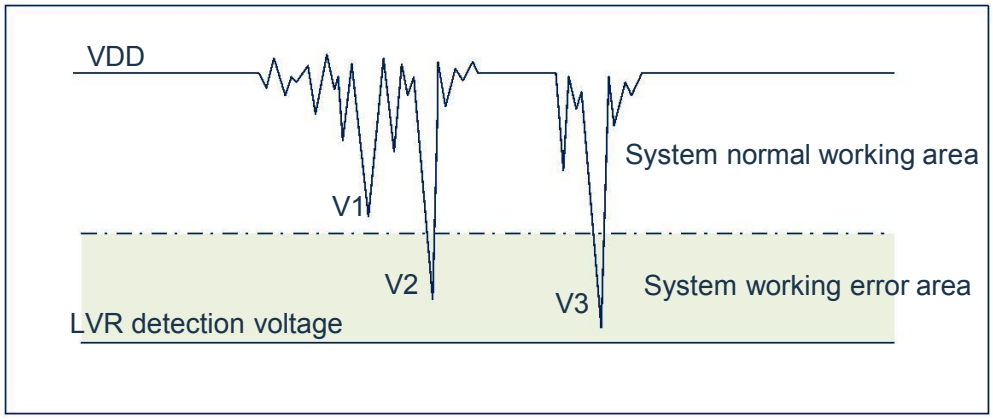


Fig 4-1: power off reset

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

System will enter the dead zone under the following situation:

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

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

4.2.2 Improvements for power off reset

Suggestions to improve the power off reset:

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

Watchdog timer

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

Lower the working speed of the system

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

Increase the gradient of the voltage drop

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

4.3 Watchdog reset

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

Time series for watchdog reset:

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

For applications of watchdog timer, see chapters at 2.8

5. Sleep mode

5.1 Enter sleep mode

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

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

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

5.2 Awaken from sleep mode

Awaken through any of the following events:

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

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

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

5.3 Interrupt awakening

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

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

5.4 Sleep mode application

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

Example: procedures for entering sleep mode

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

5.5 Sleep mode awaken time

When MCU is awaken from sleep mode, oscillation reset time is needed. This time is 1032* T_{SYS} clock period under internal high speed oscillation mode,15 * T_{SYS} clock period under low speed oscillation mode, detailed relationship as shown in table below.

System clock source	System Clock Divide Selection (IRCF< 2:0>)	Sleep wake-up wait time T_{WAIT}
Internal high-speed RC oscillation (F_{OSC}).	$F_{SYS}=F_{OSC}$	$T_{WAIT}=1032*1/F_{OSC}$
	$F_{SYS}= F_{OSC}/2$	$T_{WAIT}=1032*2/F_{OSC}$
	…	…
	$F_{SYS}= F_{OSC}/64$	$T_{WAIT}=1032*64/F_{OSC}$
Internal low-speed RC oscillation ($F_{LFINTOSC}$).	----	$T_{WAIT}=15/F_{LFINTOSC}$

6. I/O port

The chip has three I/O ports: PORTA, PORTB, and PORTC (max 18 I/O). Read-write port data registers provide direct access to these ports.

port	bit	Pin description	I/O
DOOR	0	Schmidt touch input, push-pull input, AN0, TK0, external interrupt input	I/O
	1	Schmidt Touch In, Push Pull Out, AN1, TK1	I/O
	2	schmidt touch in, push-pull, AN2, TK2, T0CKI	I/O
	3	Schmidt touch in, push-pull in, AN3, TK3, CCP, TX/CK	I/O
	4	Schmidt Touch In, Push-Pull Out, AN4, TK4, CCP, RX/DT	I/O
	5	Schmidt touch in, push-pull, AN5, TK5	I/O
	6	Schmidt touches in and out, push-pull, AN6, TK6	I/O
	7	schmidt touches in, push-pull, AN7, TK7	I/O
PORTB	0	Schmidt Touch In, Push-Pull, AN15, TK15, CCP	I/O
	1	Schmidt Touch In, Push-Pull Out, AN14, TK14, CCP	I/O
	2	Schmidt touch in, push-pull, AN13, TK13	I/O
	3	Schmidt touch in, push-pull, AN12, TK12	I/O
	4	Schmidt touch in, push-pull out, AN11, TK11	I/O
	5	schmidt touch in, push-pull, AN10, TK10, T1G	I/O
	6	Schmidt touch in, push-pull, AN9, TK9	I/O
	7	Schmidt Touch In, Push-Pull, AN8, TK8	I/O
PORTC	0	Schmidt Touch Infusion, Push-Pull Infusion, AN17, TK17, TX/CK, ICSPDAT	I/O
	1	schmidt Touch Infusion, Push-Pull Inlet, AN16, T K16, RX/DT, T1CKI, ICSPCLK	I/O

< table6-1: Port configuration Overview >

6.1 I/O Structure diagram

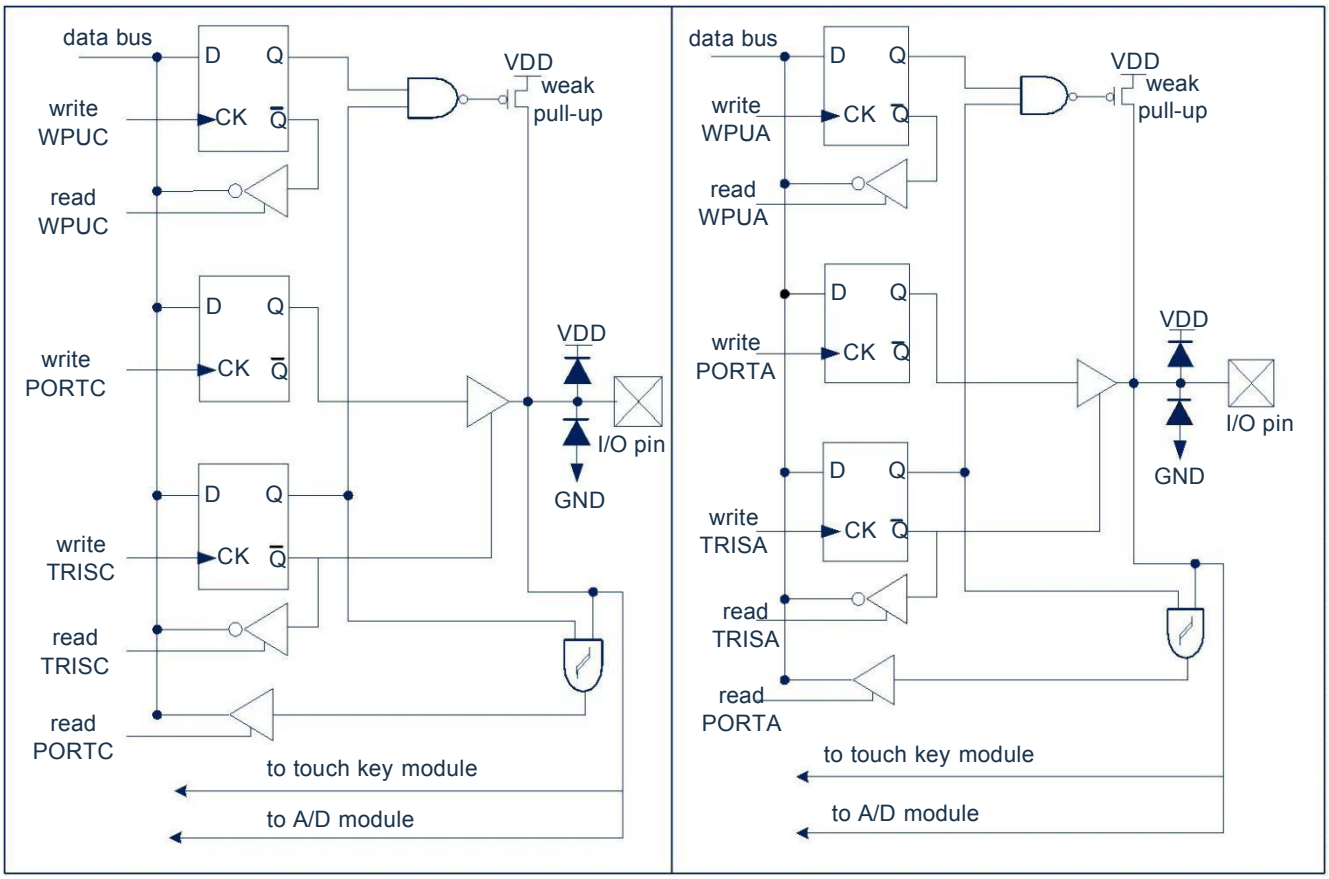


Fig. 6-1: I/O structure diagram (1).

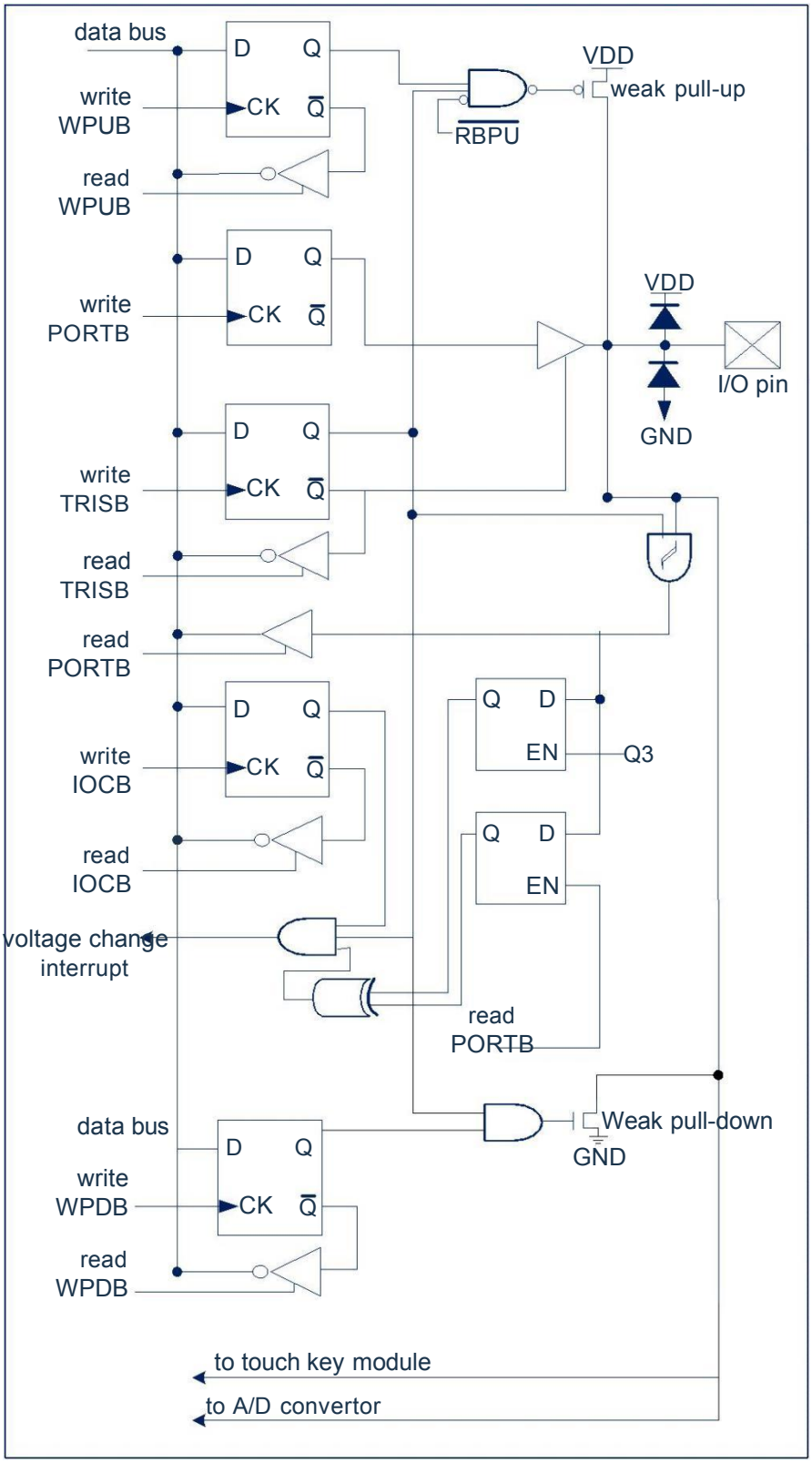


Fig. 6-2: I/O structure diagram (2).

6.2 PORTA

6.2.1 PORTA data and direction control

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

Reading PORTA register reads the pin status. Writing PORTA write to port latch. All write operation are read-change-write. Hence, write 1 port means read the pin electrical level of the port, change the value and write the value into port 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 bits in the TRISA register remain set in 1 state.

Registers related to portA ports include PORTA, TRISA, WPUA, ANSEL0 etc.

PORTA Data Register PORTA(05H)

05H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DOOR	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>: PORTA I/O pin bit;

When TRISAx=1

1= The port pin level >VIH;

0= The port pin level < VIL.

When TRIESAx=0

1= Port output high;

0= The port output is low.

PORTA Direction Register TRISA(85H)

85H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CHEAT	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 three-state control bit;

1= The PORTA pin is configured as an input (tri-state);

0= The PORTA pin is configured as an output.

example: procedure for PORTA

```
LDIA    B'11110000'     ; Set the PORTA < 3:0 > as the output port and the PORTA < 7:4 > as the input port
LD      TRISA,A
LDIA    03H             ; PortA < output high > 1:0 and PORTA < 3:2> output low
LD      DOOR,A          ; Since the PORTA < 7:4 > is the input port, it does not affect whether it is given 0 or 1
```

6.2.2 PORTA pull-up resistor

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

PORTA pull-up resistor register WPUA(107H)

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

Bit7~Bit0 WPUA<7:0>: Weak pull-up register bits.
1= Enable pull-up.
0= Pull-up is prohibited.

Note: If the pin is configured as an output, weak pull-ups are automatically disabled.

6.2.3 PORTA analog selection control

AnSEL0 registers are used to configure the input mode of the I/O pins to analog mode. Placing the corresponding bit 1 in ANSEL0 will cause all digital reads to the corresponding pin to return to 0 and make the analog function of the pin work properly. The state of the ANSEL0 bit has no effect on digital output function. The PIN of TRIS cleared and an ANSEL0 to 1 will still be used as a digital output, but the input mode will become analog mode. This can result in unpredictable results when read-modify-write operations are performed on the affected port.

PORTA analog selection register ANSEL0 (09H).

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

Bit7~Bit0 YEARS<7:0>: Analog selection bits, which select analog or digital functions > pin AN< 7:0, respectively.
1= Analog input. Pins are assigned as analog inputs.
0= Digital I/O. Pins are assigned to ports or special functions.

6.2.4 PortA level change interrupt

All PORTA pins can be individually configured as level-varying interrupt pins. The control bit IOCA < 7:0> allow or disable this interrupt function per pin. Disables the level change interrupt function of the pin during power-on reset.

For a pin that has allowed a level change interrupt, the value on that pin is compared to the old value latched at the last read of PORT A. The output of the "mismatch" of the last read operation is logically or computationally performed to place the PORTA level change interrupt flag (RACIF) in the PIR2 register to 1.

The interrupt wakes the device from hibernation and the user can clear the interrupt in the interrupt service program in the following ways:

- Read or write to PORT A. This ends the mismatched state of the pin levels.
- Clear the flag bit RACIF.

The mismatch state will continuously place the RACIF flag at bit 1. Reading or writing PORTA will end the mismatch state and allow the RACIF flag bit to be cleared. The latch will keep the last read value unaffected by undervoltage reset. After the reset, if no match persists, the RACIF flag bit will continue to be set at 1.

Note: If the level of the I/O pin changes while performing a read operation (at the beginning of the Q2 cycle), the RACIF interrupt flag bit is not set to 1. In addition, because reads or writes to a port affect all bits of that port, special care must be taken when using multiple pins in level-varying interrupt mode. You may not notice a level change on one pin when dealing with a change in level on the other pin.

PORTA level-varying interrupt register IOCA(91H).

91H	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> The level change of the PORTA interrupts the control bit.
1= Allow level changes to be interrupted.
0= Disables interrupts for level changes.

6.2.5 PORTA drive current control

All PORTA pins can select different outputs of high drive current and low level drive current, certain position in the register PAHEN been set to 1 (=1) can make the corresponding PORTA pin The output high drive current is selectable, and the current size selection is controlled by the SEGCUR register. Place certain position in the PALM register to 1 (=1) to make the corresponding PORTA pin The output low drive current is optional.

PORTA high-level drive current control register PAHEN (116 H).

116H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PAHEN	PAHEN7	PAHEN6	PAHEN5	PAHEN4	PAHEN3	PAHEN2	PAHEN1	PAHEN0
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 PAHEN<7:0> The high-level output of PORTA drives the current control bit.
1= Allows PORTA high drive current selectable (0mA to 30 mA).
0= Disable high level drive current optional, default is (30mA).

PORTA low-level drive current control register PALEN (114H).

114H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
POLES	PALEN7	PALEN6	PALEN5	PALEN4	PALEN3	PALEN2	PALEN1	PALEN0
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 PALEN<7:0> The low-level output of PORTA drives the current control bit.
1= Large drive current (120mA);
0= Small drive current (60mA).

PORTA/PORTB high-level drive current control register SEGCUR (113 H).

13H.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGCUR	---	---	---	---	SEG_ISEL[3:0]			
R/W	---	---	---	---	R/W	R/W	R/W	R/W
Reset value	---	---	---	---	0	0	0	0

Bit7~Bit4 Unused
Bit3~Bit0 SEG_ISEL<3:0> The high-level output of portA/PORTB drives the current select bit.
0000: The SEG port drive current is 0;
0001: The SEG port drive current is 2mA;
0010: The SEG port drive current is 4mA;
0011: The SEG port drive current is 6mA;
…
1110: The SEG port drive current is 28mA;
1111: The SEG port drive current is 30mA.

6.3 PORTB

6.3.1 PORTB data and direction

PORTB is a 8Bit 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 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 in set in 1 state.

Registers related to THE PORTB port are PORTB, TRISB, WPUB, IOCB, WPDB, ANSEL1 etc.

PORTB Data Register PORTB(06H)

06H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0 PORTB<7:0>: PORTB I/O pin bits.

When TRISBx=1

1= The port pin level >VIH;

0= The port pin level < VIL.

When TRISBx=0

1= Port output high;

0= The port output is low.

PORTB Direction Register TRISB (86H)

86H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

Bit7~Bit0 TRISB<7:0>: PORTB three-state control bit.

1= The PORTB pin is configured as an input (tri-state).

0= The PORTB pin is configured as an output.

Example: PORTB port handler

```
CLR     PORTB           ; Clear data registers
LDIA    B'00110000'     ; Set the PORTB < 5:4 > as the input port and the rest as the output port
LD      TRISB,A
```

6.3.2 PORTB pull-up resistor

Each PORTB pin has an internal weak pull up that can be individually configured. The control bits WPUB<7:0> enable or disable each weak pull up. When the port pin is configured as output, its weak pull up will be automatically cut off. During power-on reset, weak pull-ups are disabled by the RBPU bit of the OPTION_REG register.

PORTB pull-up resistor register WPUB(95H).

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 WPUB<7:0>: Weak pull-up register bits.
1= Enable pull-up.
0= Pull-up is prohibited.

Note:
1) To enable either pull-up individually, the global RBPU bits of the OPTION_REG register must be cleared.
2) If the pin is configured as an output or analog input, weak pull-ups are automatically disabled.

6.3.3 PORTB analog selection control

The ANSEL1 register is used to configure the input mode of the I/O pins to analog mode. Placing the appropriate position 1 in ANSEL 1 will cause all digital reads to return to 0 for the corresponding pin and make the analog function of the pin work properly. The state of the ANSEL1 bit has no effect on digital output function. The PIN of TRIS cleared and an ANSEL1 to 1 will still be used as a digital output, but the input mode will become analog mode. This can result in unpredictable results when read-modify-write operations are performed on the affected port.

PORTB analog selection register ANSEL1 (89H).

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
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 YEARS<15:8>: Analog select bits, selecting analog or digital functions > pin AN< 15:8, respectively.
1= Analog input. Pins are assigned as analog inputs.
0= Digital I/O. Pins are assigned to ports or special functions.

6.3.4 PortB level change interrupt

All PORTB pins can be individually configured as level change interrupt pins. The control bit IOCB<7:0> enables or disables the interrupt function of each pin. Disable pin level change interrupt function when power on reset.

For the pin that has enableed level change interrupt, compare the value on the pin with the old value latched when PORTB was read last time. Perform a logical OR operation with the output "mismatch" of the last read operation to set the PORTB level change interrupt flag (RBIF) in the 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 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 PORTA will end the mismatch state and enable the RBIF flag to be cleared. The latch will keep the last read value from the undervoltage reset. After reset, if the mismatch still exists, the RBIF flag will continue to be set as 1.

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

PORTB level change interrupt register IOCB (88H).

88H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0	IOCB<7:0>	The level change of the PORTB interrupts the control bit.
	1=	Enable level changes to be interrupted.
	0=	Disables interrupts for level changes.

6.3.5 PORTB pull-down resistor

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

PORTB pull-down resistor register WPDB(97H).

97H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 WPDB<7:0>: Weak pull-down register bits.
1= Enable pull-down.
0= Pull-down is prohibited.

Note: Weak pulldowns are automatically disabled if the pins are configured as outputs or analog inputs.

6.3.6 PORTB drive current control

All port B pins can select different outputs of high-level drive current and low-level drive current, with P B Somewhere in the HEN register 1 (=1) can make the corresponding PORT The output high drive current on pin B is selectable, and the current size is selected by SEGCUR Register control. Place p BLEN register somewhere in 1 (=1) can make a corresponding the output low drive current of the PORT B pin is optional.

PORTB high drive current control register PBHEN (117H).

117H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PBHEN	PBHEN7	PBHEN6	PBHEN5	PBHEN4	PBHEN3	PBHEN2	PBHEN1	PBHEN0
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 PBHEN<7:0> The high-level output of the PORTB drives the current control bit.
1= Allows PORTB high drive current selectable (0mA to 30 mA).
0= Disable high level drive current optional, default is (30mA).

PORTB drives the current control register PBLEN (115H) low level

115H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PBLEN	PBLEN7	PBLEN6	PBLEN5	PBLEN4	PBLEN3	PBLEN2	PBLEN1	PBLEN0
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 PBLEN<7:0> The low-level output of the PORTB drives the current control bit.
1= Large drive current (120mA);
0= Small drive current (60mA).

PORTA/PORTB high-level drive current control register SEGCUR (113 H).

11P.M.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGCUR	---	---	---	---	SEG_ISEL[3:0]			
R/W	---	---	---	---	R/W	R/W	R/W	R/W
Reset value	---	---	---	---	0	0	0	0

Bit7~Bit4	Unused	
Bit3~Bit0	SEG_ISEL<3:0>	The high-level output of portA/PORTB drives the current select bit.
	0000:	The SEG port drive current is 0;
	0001:	The SEG port drive current is 2mA;
	0010:	The SEG port drive current is 4mA;
	0011:	The SEG port drive current is 6mA;
	…	
	1110:	The SEG port drive current is 28mA;
	1111:	The SEG port drive current is 30mA.

6.4 PORTC

6.4.1 PORTC data and direction

PORTC is an 2bit wide bidirectional port. The corresponding data direction register is TRISC. Set a certain position in TRISC to 1 (=1) to make the corresponding PORTC pin as the input pin. Clearing a bit in TRISC (=0) will make the corresponding PORTC pin as the output pin.

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

PORTC Data Register PORTC(07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTC	----	----	----	----	----	----	RC1	RC0
R/W	----	----	----	----	----	----	R/W	R/W
Reset value	----	----	----	----	----	----	X	X

Bit7~Bit2	Unused	
Bit1~Bit0	PORTC<1:0>	PORTC I/O pin bits.
	When TRISCx=1	
	1=	The port pin level >VIH;
	0=	The port pin level < VIL.
	When TRISCx=0	
	1=	Port output high;
	0=	The port output is low.

PORTC Direction Register TRISC(87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISC	----	----	----	----	----	----	TRISC1	TRISC0
R/W	----	----	----	----	----	----	R/W	R/W
Reset value	----	----	----	----	----	----	1	1

Bit7~Bit2	Unused	
Bit1~Bit0	TRISC<1:0>:	PORTC three-state control bit.
	1=	The PORTC pin is configured as an input (tri-state).
	0=	The PORTC pin is configured as an output.

Example: PORTC port handling program

```
CLR     PORTC           ; Clear data registers
LDIA    B'00000001'     ; Set the PORTC < 1 > as the output port and the PORTC <0 > as the input port
LD      TRISC,A
```

6.4.2 PORTC pull-up resistor

Each PORTC pin has an internal weak pull up that can be individually configured. The control bits WPUC<1:0> enable or disable each weak pull up. When the port pin is configured as output, its weak pull up will be automatically cut off.

PORTC pull-up resistor register WPUC(108H)

108H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUC	----	----	----	----	----	----	WPUC1	WPUC0
R/W	----	----	----	----	----	----	R/W	R/W
Reset value	----	----	----	----	----	----	0	0

Bit7~Bit2 Unused

Bit1~Bit0 WPUC<2:0>: Weak pull-up register bits.

1= Enable pull-up.

0= Pull-up is prohibited.

Note: If the pin is configured as an output or analog output, weak pull-ups are automatically disabled.

6.4.3 PORTC analog selection control

AnSEL2 registers are used to configure the input mode of the I/O pins to analog mode. Placing the cprresponding bit to 1 in ANSEL2 will cause all digital reads of the corresponding pin to return to 0 and the analog function of the pin to work properly. The state of the ANSEL2 bit has no effect on digital output function. The PIN of TRIS cleared and an ANSEL2 to 1 will still be used as a digital output, but the input mode will become analog mode. This can result in unpredictable results when read-modify-write operations are performed on the affected port.

PORTC analog selection register ANSEL2 (109H).

109H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL2	----	----	----	----	----	----	ANS17	ANS16
R/W	----	----	----	----	----	----	R/W	R/W
Reset value	----	----	----	----	----	----	0	0

Bit7~Bit2 Unused

Bit1~Bit0 YEARS<17:16>: Analog selection bits for analog or digital functions < pin AN < 17:16>, respectively.

1= Analog input. Pins are assigned as analog inputs.

0= Digital I/O. Pins are assigned to ports or special functions.

6.5 I/O usage

6.5.1 Write I/O port

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

Example: write I/O port program

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

6.5.2 Read I/O port

Example: write I/O port program

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

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

6.6 Precautions for the use of I/O ports

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

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

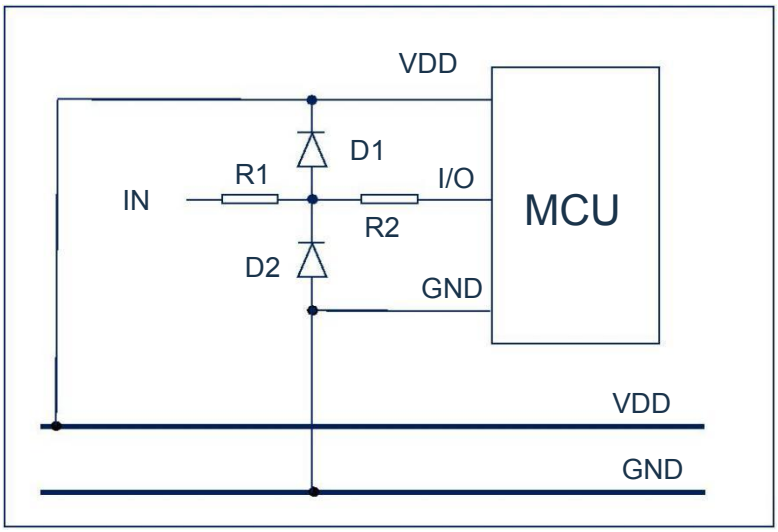


Figure 6-3: The input voltage is circuited outside the specified range

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

7. interrupt

7.1 Interrupt overview

The chip has the following multiple interrupt sources:

- TIMER0 overflow interrupt
- TIMER2 matches interrupts
- PortA level change interrupt
- PortB level change interrupt
- CCP1/CCP2 interrupts
- USART receive/transmit interrupts
- TIMER1 overflow interrupt
- INT interrupt
- A/D interrupt
- Touch detection cmpletion interrupt
- The program EEPROM write operation interrupt

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

The global interrupt enable bit GIE (INTCON<7>) enables 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, PIE2 registers. GIE is cleared when reset.

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

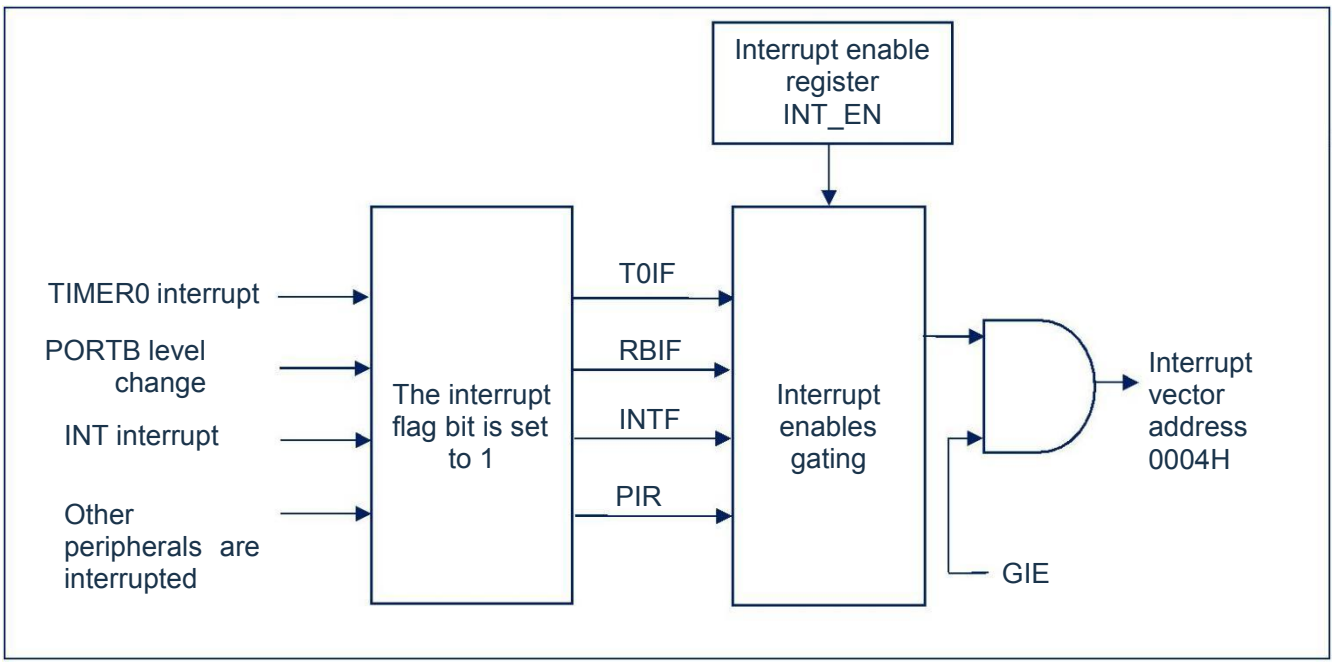


Figure 7-1: Schematic diagram of the interrupt principle

7.2 Interrupt control registers

7.2.1 Interrupt control registers

The interrupt control register INTCON is a readable and writable register, including the enable and flag bits for TMR0 register overflow and PORTB interface voltage change…etc interrupt enable and flag bits.

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 enableing an interrupt.

Interrupt control register INCON (0BH).

0BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCON	GIE	PEIE	T0IE	NOT	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 bits;
1= Enable all unblocked interrupts;
0= Disable all interruptions.

Bit6 PEIE: Peripheral interrupt enable bits;
1= Enable all unblocked peripherals to interrupt;
0= Disable all peripheral interrupts.

Bit5 T0IE: TIMER0 overflow interrupt enables bits;
1= Enable TIMER0 interrupts;
0= Banned TIMER0 suspended.

Bit4 NOT: INT external interrupt enables bits;
1= Enable int external interrupts;
0= Prohibit int external interruption.

Bit3 RBIE: PORTB level change interrupt enables bit (1);
1= Enables THE PORTB level change to be interrupted;
0= Disables interrupts for PORTB level changes.

Bit2 T0IF: TIMER0 overflow interrupt flag bit (2);
1= The TMR0 register has overflowed (must be cleared by software);
0= The TMR0 register has not overflowed.

Bit1 INTF: INT external interrupt flag bit;
1= An int external interrupt has occurred (must be cleared by the software);
0= No int external interrupts have occurred.

Bit0 RBIF: PORTB level change interrupt flag bit;
1= The level state of at least one pin in the PORTB port has changed (must be cleared by software);
0= The state of none of the PORTB general-purpose I/O pins has changed.

Note:
1. The IOCB registers must also be enabled, and the corresponding port lines need to be set to the input state
2. The T0IF bit is placed at 1 when the TMR0 is full and 0. Resetting does not change TMR0 and should be initialized before clear out the T0IF bit.

7.2.2 Peripheral interrupt enable register

Peripheral interrupt enable registers are PIE1 and PIE2, and before any peripheral interrupt can be enabled, the PEIE position of the INCON register must be placed at 1.

Peripheral interrupt enable register PIE1 (8CH).

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	----	ADIE	RCIE	TXIE	----	CP1IE	TMR2IE	TMR1IE
R/W	----	R/W	R/W	R/W	----	R/W	R/W	R/W
Reset value	----	0	0	0	----	0	0	0

Bit7 Unused

Bit6 ADIE: A/D Converter (ADC) interrupt bits are enable ed
1= Enable ADC interrupts;
0= Banned ADC interruption.

Bit5 RCIE: USART receives interrupt enable bits
1= Enables USART to receive interrupts;
0= Disables USART reception interrupts.

Bit4 TXIE: USART sends interrupt enable bits
1= Enables USART to send interrupts;
0= Disables USART sending interrupts.

Bit3 Unused

Bit2 CCP1IE: CCP1 interrupt enables bits
1= Enable CCP1 interrupts;
0= Banned CCP1 interruption.

Bit1 TMR2IE: TIMER2 matches the interrupt enable bit with PR2
1= Enable TMR2 to match interrupts with PR2;
0= Disable tmR2 and PR2 matching interrupts.

Bit0 TMR1IE: TIMER1 overflow interrupt enable bit
1= Enable TIMER1 overflow interrupt;
0= Disables TIMER1 overflow interrupts.

Peripheral interrupt allows register PIE2 (8DH).

8DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	---	TKIE	RACIE	EEIE	---	---	---	CCP2IE
R/W	---	R/W	R/W	R/W	---	---	---	R/W
Reset value	---	0	0	0	---	---	---	0

Bit7	Unused	
Bit6	TKIE:	Touch detection end interrupt allow bit;
	1=	Allow touch detection to end interrupt;
	0=	Disables the end of touch detection interrupt.
Bit5	INSTALLMENTS:	PORTA level change interrupt allows bits;
	1=	Allows PORTA level changes to be interrupted;
	0=	Disables PORTA level change interrupts.
Bit4	EEIE:	Program EEPROM write operation interrupt allows bits;
	1=	Allow program EEPROM write operations to be interrupted;
	0=	Disables program EEPROM write operations from being interrupted.
Bit3~Bit1	Unused	
Bit0	CCP2IE :	CCP2 interrupt allows bits;
	1=	Allow CCP2 interrupts;
	0=	Ban CCP2 Suspend.

7.2.3 Peripheral interrupt request register

The peripheral interrupt request registers are PIR1 and PIR2. When an interrupt condition occurs, the interrupt flag bit will be set to 1 regardless of the state of the corresponding interrupt allow bit or the global allow bit GIE. The user software should ensure that the corresponding interrupt flag bits are zeroed before allowing an interrupt.

Peripheral interrupt request register PIR1 (0CH).

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

Bit7 Unused

Bit6 ADIF: A/D converter interrupt flag bit;
1= The A/D conversion is complete (must be cleared by the software);
0= The A/D conversion is not complete or has not started yet.

Bit5 RCIF: USART receives interrupt flag bits;
1= THE USART receive buffer is full (cleared by reading RCREG);
0= The USART receive buffer is empty.

Bit4 TXIF: USART sends interrupt flag bits;
1= USART sends the buffer empty (by writing TXREG to zero);
0= The USART send buffer is full.

Bit3 Unused

Bit2 CCP1IF: CCP1 interrupt flag bit.
Capture Mode:
1= A capture of the TMR1 register has occurred (must be cleared by software);
0= No capture of the TMR1 registers occurred.
Compare mode:
1= A compare match of the TMR1 registers has occurred (must be cleared by the software);
0= No compare matching of TMR1 registers occurred.
PWM mode: Not used in this mode.

Bit1 TMR2IF: TIMER2 matches the interrupt flag bit with PR2.
1= A TIMER2 match with PR2 has occurred (must be cleared by software);
0= TIMER2 does not match PR2.

Bit0 TMR1IF: TIMER1 overflows the interrupt flag bit.
1= TMR1 register overflow (must be cleared by software);
0= The TMR1 register is not overflowing.

Peripheral interrupt request register PIR2 (0DH).

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	---	TKIF	RACIF	EEIF	---	---	---	CCP2IF
R/W	---	R/W	R/W	R/W	---	---	---	R/W
Reset value	---	0	0	0	---	---	---	0

Bit7 Unused

Bit6 TKIF: Touch the end of detection interrupt flag bit;
1= Touch detection has ended complete (must be zeroed by the software);
0= No touch detection ended.

Bit5 RACIF: PORTA level change interrupt flag bit;
1= The level state of at least one pin in the PORTA port has changed (must be cleared by software);
0= None of the PORTA general-purpose I/O pins have changed state.

Bit4 EEIF: Program EEPROM write operation interrupt flag bit;
1= Write operation completed (must be zeroed by software);
0= The write operation did not complete or has not started.

Bit3~Bit1 Unused

Bit0 CCP2IF: CCP2 interrupt flag bit.

Capture Mode:
1 = A capture of the TMR1 register has occurred (must be cleared by software);
0 = No capture of TMR1 registers has occurred.

Compare mode:
1 = A compare match of the TMR1 registers has occurred (must be cleared by the software);
0 = No compare matching of TMR1 registers occurred.

PWM mode: Not used in this mode.

7.3 Protection methods for interrupt

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

Example: Stack protection for ACC and STATUS

```
                ORG         0000H
                JP          START           ;start of user program address
                ORG         0004H
                JP          INT_SERVICE     ;interrupt service program
                ORG         0008H
START:
                ...
                ...
INT_SERVICE:
PUSH:                                       ;entrance for 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. Timer counter TIMER0

8.1 Timer counter TIMER0 overview

TIMER0 consists of the following functions:

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

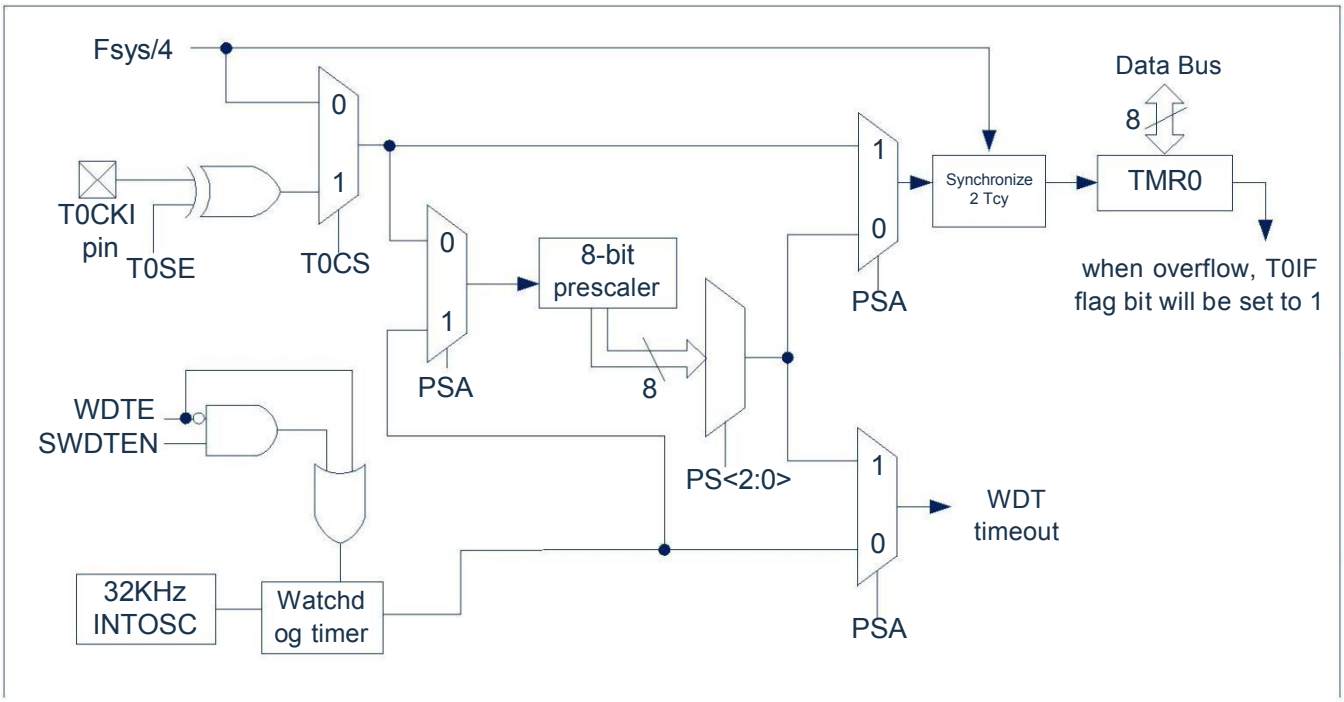


Figure 8-1: TimeR0/WDT module structure diagram

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

8.2 Working principle for TIMER0

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

8.2.1 8-bit timer mode

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

8.2.2 8-bit counter mode

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

8.2.3 Software programmable pre-scaler

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

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

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

8.2.4 Switch between TIMER0 and WDT module pre-scaler

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)

```
CLRWDT                          ;clear WDT
LDIA        B'xxxx1xxx'         ;set new pre-scaler
LD          OPTION_REG,A
CLRWDT                          ;clear WDT
```

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'xxxx0xxx'         ;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: Since the timer is off in the sleep state, the TIMER0 interrupt cannot wake up the processor.

8.3 TIMER0 related register

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

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

8-bit timer/counter TMR0(01H)

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

OPTION_REG register(81H)

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

Bit7 RBPU: PORTB pull-up enable bit.
1= PORTB pull-up is prohibited.
0= The PORTB pullup is enabled by the individual latch values of the port.

Bit6 INTEDG: Interrupt edge selection bit.
1= The rising edge of the INT pin triggers interrupt.
0= The falling edge of the INT pin triggers interrupt.

Bit5 T0CS: TMR0 clock source selection bit.
1= Transition edge of T0CKI pin.
0= 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. Timer counter TIMER1

9.1 TIMER1 overview

The TIMER1 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register (TMR1H: TMR1L).
- 3-bit prescaler
- Synchronous or asynchronous operations
- Wake on overflow (external clock asynchronous mode only)
- Special event triggering function (with ECCP).
- Programmable internal or external clock source
- TIMER1 (enable count) is gated by the T1G pin
- Overflow interrupt
- The time base of the capture/compare feature

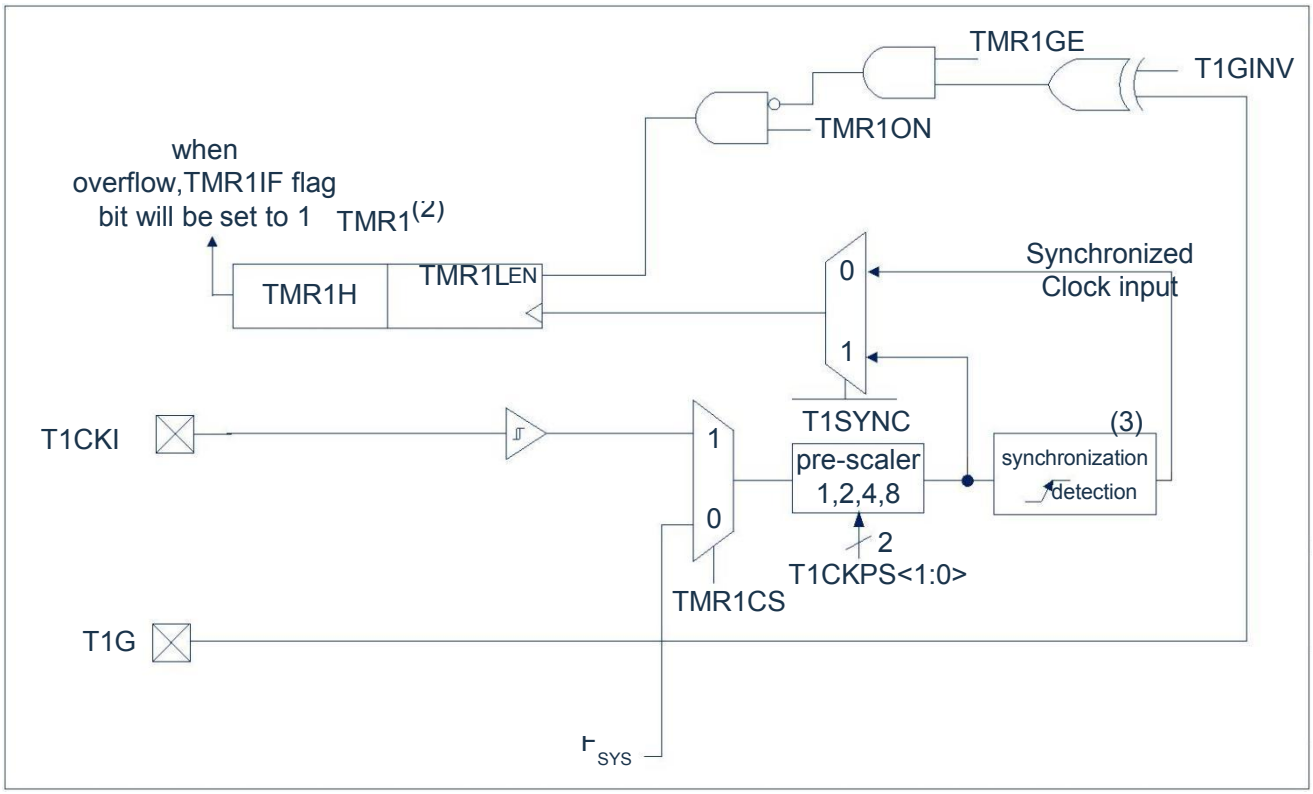


Figure 9-1: TIMER1 structure diagram

Note:

1. The ST buffer is in high-speed mode when using the T1CKI
2. The Timer1 register is incremented on the rising edge.
3. Synchronization does not occur while hibernating.

9.2 Operating principle for TIMER1

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

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

9.3 clock source selection

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

clock source	TMR1CS
F_{SYS}	0
T1CKIpin	1

9.3.1 internal clock source

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

9.3.2 external clock source

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

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

If an external clock oscillator is required, TIMER1 can use LP oscillator as clock source.

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

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

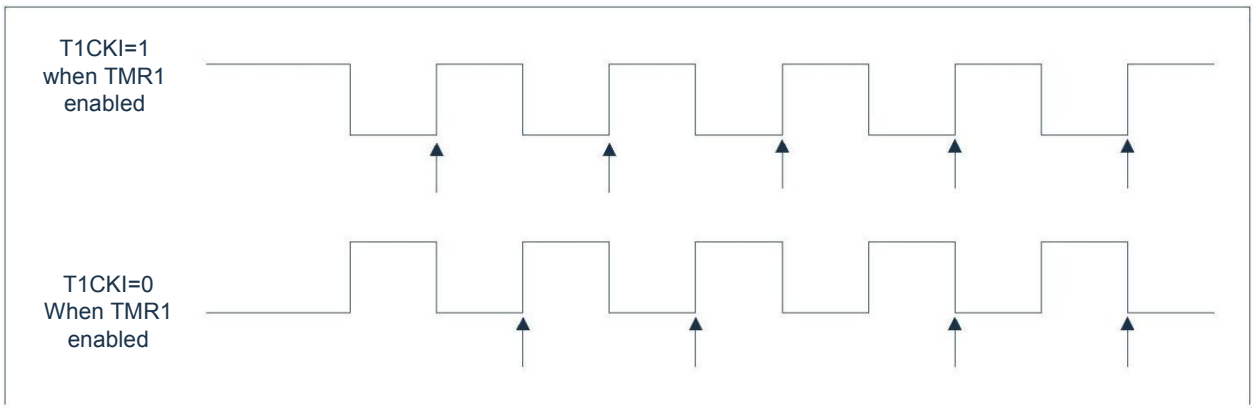


Fig 9-2: incremental edge of TIMER1

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

9.4 TIMER1 pre-scaler

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

9.5 TIMER1 operating principle under asynchronous counter mode

If the control bit T1SYNC in the T1CON register is set to 1, the external clock input will not be synchronous. The timer continues to count up asynchronously with the internal phase clock. The timer will continue to run in the sleep state, and will generate an interrupt during overflow, thereby waking up Processor. However, you should be especially careful when using software to read/write timers (see Section 9.5.1 "Read/Write to TIMER1 in Asynchronous Counter Mode").

Note:

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

9.5.1 Read/Write operations to TIMER1 in asynchronous counter mode

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

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

9.6 TIMER1 gate control

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

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

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

9.7 TIMER1 interrupt

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

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

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

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

9.8 TIMER1 working principle during sleep

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

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

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

9.9 ECCP capture/compare timebase

The ECCP module uses the TMR1H:TMR1L pair register as the time base for its operation in capture or compare mode.

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

For more information, see the section "Capture/Compare/PWM Modules (CCP1 and CCP)".

9.10 ECCP Special Event Trigger

If the ECCP is configured to trigger a special event, the trigger will clear the MR1H:TMR1L pair of registers. This special event does not cause TIMER1 to interrupt. The ECCP module can still be configured to generate an ECCP interrupt.

In this mode of operation, the CCPRxH:CCPRxL registers effectively become the periodic registers for TIMER1.

To use a special event trigger you should synchronize TIMER1 with F_{SYS}. TimeR1 operating in asynchronous mode can result in the loss of a special event trigger signal.

Write operations take precedence when the operation of writing to TMR1H or TMR1L occurs at the same time as a special event trigger signal from ECCP.

For more information, see the "Capture/Compare/PWM Modules (CCP1 and CCP)" section.

9.11 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 gated signal polarity bit;
1= TIMER1 gated signal is active high (TIMER1 counts when gating signal is high);
0= The TIMER1 gate signal is active low (timer1 counts when the gated signal is low).

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

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

Bit3 Unused

Bit2 T1SYNC: TIMER1 external clock input synchronous control bit.
TMR1CS=1: 1= Not synchronized with external clock input;
0 = Synchronized with external clock input.
TMR1CS=0: This bit is ignored and TIMER1 uses the internal clock.

Bit1 TMR1CS: TIMER1 clock source select bit;
1= Clock source from LP oscillator clock source or the T1CKI pin (rising edge trigger);
0= Internal clock source FSYS.

Bit0 TMR1ON: TIMER1 enable bit;
1= Enable TIMER1;
0= TIMER1 is prohibited.

10. Timer counter TIMER2

10.1 TIMER2 overview

The TIMER2 module is an 8-bit timer/counter with the following features:

- 8-bit timer register (TMR2);
- 8-bit periodic register (PR2);
- Interrupt when TMR2 matches PR2;
- Software programmable prescale ratios (1:1, 1:4 and 1:16);
- Software programmable post-division ratio (1:1 to 1:16).

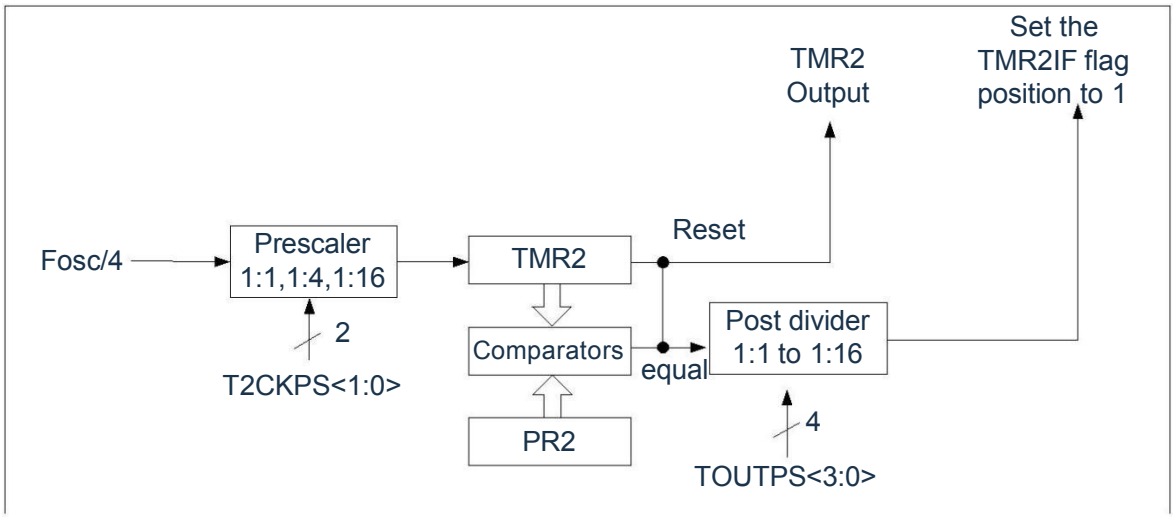


Figure 10-1: TIMER2 block diagram

10.2 Operating principle of TIMER2

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

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

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

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

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

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

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

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

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

Note: Writing T2CON will not clear TMR2, when TMR2ON=0, it will clear TMR2; MR2ON is set to 1 to write to TMR2.

10.3 TIMER2-related registers

There are 2 registers associated with TIMER2, namely the data memory TMR2 and the control register T2CON.

TIMER2 Data RegisterTMR2(11H)

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

TIMER2 Control Register T2CON(12H)

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

Bit7	Unused	
Bit6~Bit3	TOUTPS<3:0>:	TimeR2 output after the division ratio selection bit.
	0000=	1:1 after division ratio;
	0001=	1:2 after the crossover ratio;
	0010=	Divide ratio after 1:3;
	0011=	Divide ratio after 1:4;
	0100=	Divide ratio after 1:5;
	0101=	1:6 after the division ratio;
	0110=	Divide ratio after 1:7;
	0111=	Divide ratio after 1:8;
	1000=	Divide ratio after 1:9;
	1001=	Divide ratio after 1:10;
	1010=	Divide ratio after 1:11;
	1011=	Divide ratio after 1:12;
	1100=	Divide ratio after 1:13;
	1101=	Divide ratio after 1:14;
	1110=	Divide ratio after 1:15;
	1111=	1:16 post-division ratio.
Bit2	TMR2ON:	TIMER2 enable bit;
	1=	Enable TIMER2;
	0=	TIMER2 is prohibited.
Bit1~Bit0	T2CKPS<1:0>:	TIMER2 clock prescaler ratio selection bit;
	00=	The prescaler value is 1;
	01=	The prescaler value is 4;
	1x=	The prescaler value is 16.

11. Analog-to-digital conversion (ADC).

11.1 ADC Overview

An analog-to-digital converter (ADC) converts an analog input signal into a 12-bit binary number representing 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 successive approximations to produce a 12-bit binary result and saves the result in the ADC result registers (ADRESL and ADRSH).

The ADC reference voltage is always generated internally. The ADC can generate an interrupt after the conversion is complete.

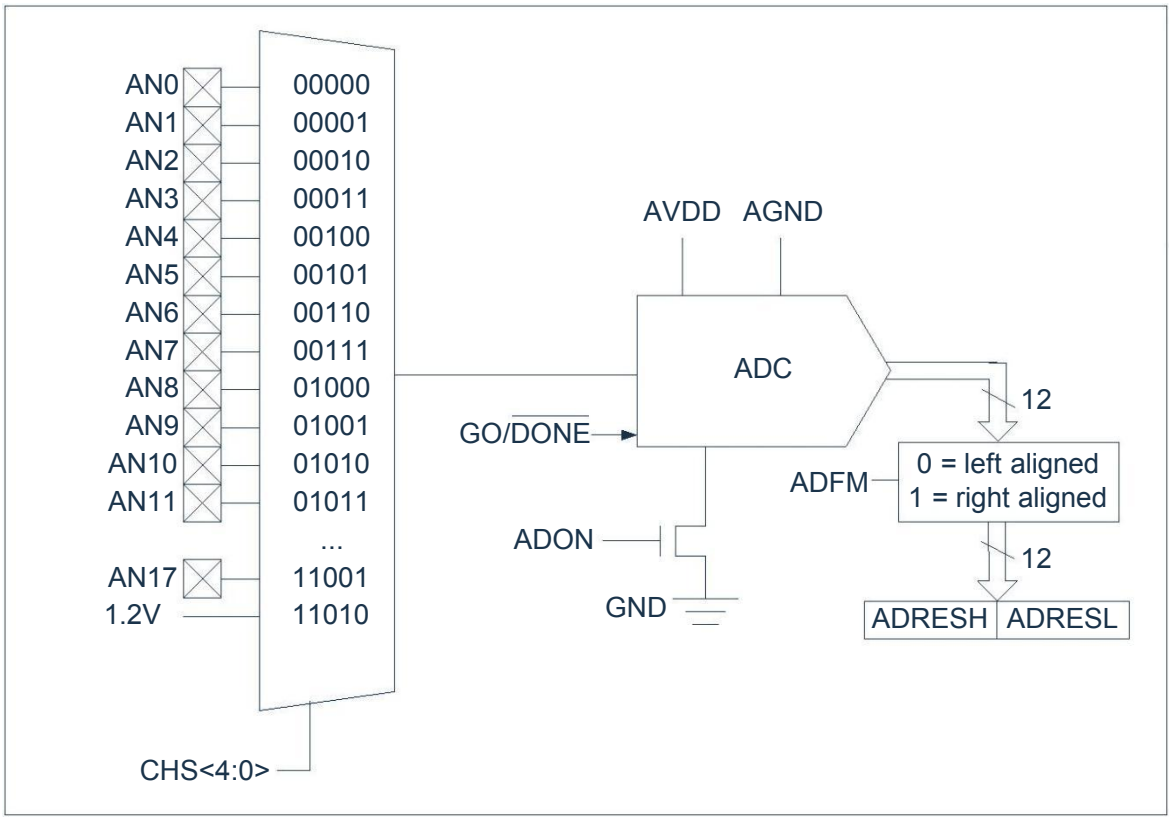


Figure 11-1: ADC block diagram

11.2 ADC configuration

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

- Port configuration;
- Channel selection;
- ADC conversion clock source;
- Interrupt control;
- The format in which the results are stored.

11.2.1 Port configuration

ADC can convert both analog and digital signals. When converting an analog signal, the I/O pin should be configured as an analog input pin by placing the corresponding TRIS position 1. For more information, see the appropriate port section.

Note: Applying an analog voltage to a pin defined as a digital input may cause an 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 circuitry.

If you change the channel, you will need a delay before the next conversion starts. For more information, see the "ADC working principle" section.

11.2.3 ADC reference voltage

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

11.2.4 Convert the clock

The converted clock source can be selected by setting the ADCS bit of the ADCON0 register by software. There are 2 possible clock frequencies to choose from:

- $F_{SYS}/8$
- $F_{SYS}/16$
- $F_{SYS}/32$
- $F_{SYS}/128$

The time at which a bit of a conversion is completed is defined as a TAD. A complete 12-bit conversion requires 16 TAD cycles.

The appropriate TAD specification must be met to obtain the correct conversion results, and the following table is an example of the correct selection of an ADC clock.

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

When different reference voltages and different VDDs, it is necessary to refer to the following table to set a reasonable frequency division.

Reference voltage	Operating voltage (V).	Fastest crossover setting		Conversion time (us).
		F_{SYS} = 16MHz	F_{SYS} = 8MHz	
VDD	3.0~5.5	$F_{SYS}/16$	$F_{SYS}/8$	16
InDD	2.7~3.0	$F_{SYS}/32$	$F_{SYS}/16$	32

11.2.5 ADC interrupt

The ADC module enables an interrupt to be generated after the analog-to-digital conversion is complete. The ADC interrupt flag bit is the ADIF bit in the PIR1 register. The ADC interrupt enable bit is the ADE bit in the PIE1 register. The ADIF bit must be cleared with software. The ADF bit is set to 1 at the end of each conversion, regardless of whether the ADC interrupt is enableed.

Interrupts can occur regardless of whether the device is in operating mode or sleep mode. If the device is in sleep mode, the interrupt wakes the device up. When the device is awakened from sleep, the next instruction after the STOP instruction is always executed. If the user attempts to wake the device from sleep mode and resume code execution sequentially, a global interrupt must be suppressed. If a global interrupt is enableed, the program jumps to the interrupt service program for execution.

11.2.6 Result format

The results of the 12-bit A/D conversion can be in two formats: left-aligned or right-aligned. The output format is controlled by the ADFM bit of the 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/("DONE")¯ bit of the ADCON0 register to 1 to start analog-to-digital conversion.

Note: It is not possible to set GO/D¯¯ON¯E position to 1 with the same instructions that open A/D module.

11.3.2 Complete conversion

When the conversion is complete, the ADC mod will:

- Clear the GO/D¯¯ON¯E bit;
- Set ADIF flag bit to 1;
- Update the ADRESH:ADRESL register with the new conversion result.

11.3.3 Stop conversion

If you must terminate the conversion before conversion is completed, you can use software to clear the GO/D¯¯ON¯Ebit. 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: A device reset forces all registers to enter a reset state. Therefore, the reset shuts down the ADC module and terminates any pending transitions.

11.3.4 How the ADC works in sleep mode

The ADC module can operate in sleep mode. This operation requires the ADC clock source to be set to the F_{RC} option. 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 in the conversion. If the ADC interrupt is allowed, the device wakes up from sleep mode when the conversion ends. If the ADC interrupt is disabled, the ADC module is shut down after the conversion is completed, even if the ADON bit remains set to 1. If the ADC clock source is not F_{RC}, even if the ADON bit remains set to 1, STOP is executed The directive will still abort the current conversion and shut down the A/D module.

.

11.3.5 A/D conversion steps

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

1. Port configuration:
   - Configure the pin as the input pin (see TRIS Registers).
2. Configure the ADC module:
   - Select the ADC conversion clock;
   - Select the ADC input channel;
   - Select the format of the result;
   - Start the ADC module.
3. Configure ADC interrupt (optional):
   - Clear the ADC interrupt flag bit;
   - Enable ADC interrupts;
   - Enable peripheral interrupts;
   - Enable global outages.
4. Wait for the required acquisition time.
5. Start the conversion of GO/D$\overline{\text{ON E}}$Set 1.
6. Wait for the ADC conversion to finish by one of the following methods:
   - Query GO/D$\overline{\text{ON E}}$bit;
   - Wait for the ADC to interrupt (interrupt enableed).
7. Read the ADC results.
8. Clear the ADC interrupt flag bit (this is required if interrupts are enableed).

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

Example: AD conversion

```
        LDIA    B'10000000'
        LD      ADCON1,A
        SETB    TRISA,0         ; Set PORTA.0 as the input port
        LDIA    B'11000001'
        LD      ADCON0,A
        CALL    DELAY           ; Delay for a period of time
        SETB    ADCON0,GO
        SZB     ADCON0,GO       ; Wait for the AD conversion to finish
        JP      $-1
        LD      A,ADRESH        ; Save the AD conversion result high bit
        LD      RESULTH,A
        LD      A,ADRESL        ; Save the AD conversion result low bit
        LD      RESULTL,A
```

11.4 ADC related registers

There are four main registers associated with AD conversion, namely the control registers ADCON0 and ADCON1, and the data registers ADRSH and ADRSL.

AD control register ADCON0 (9EH).

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

Bit7~Bit6 ADCS<1:0>: A/D conversion clock select bits.
00= $F_{SYS}/8$
01= $F_{SYS}/16$
10= $F_{SYS}/32$
11= $F_{SYS}/128$

Bit5~Bit2 CHS<3:0>: Analog channel select bits low four bits and CHS4 make up a five-bit channel selection.
CHS<4:0>:
00000= AN0
00001= AN1
00010= AN2
00011= AN3
00100= AN4
00101= AN5
00110= AN6
00111= AN7
01000= AN8
01001= AN9
01010= AN10
01011= AN11
…
10000= AN16
10001= AN17
10010= Fixed reference voltage (1.2V fixed reference voltage).
other retain

Bit1 GO/$\overline{DONE}$: A/D transition status bits.
1= The A/D conversion is in progress. Start the A/D conversion at position 1. When the A/D conversion is complete, the bit is automatically zeroed by the hardware.
0= The A/D conversion is complete and/or not in progress.

Bit0 ADON: ADC enable bit.
1= Enable the ADC;
0= ADC is prohibited and does not consume operating current.

AD data register high ADCON1 (9FH).

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

Bit7 ADFM: A/D conversion result format select bits
1= Right-aligned
0= Left-aligned
Bit6 CHS4: Channel selection bits
Bit5~Bit0 Unused

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

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADDRESS11	ADDRESS10	ADDRESS9	ADDRESS8	ADDRESS7	ADDRESS6	ADDRESS5	ADDRESS4
Read and write	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0 ADDRESS<11:4>: ADC result register bit.
12 bits high the result of the conversion to 8 bits.

AD data register low ADRSL (9CH), ADFM=0

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADDRESS3	ADDRESS2	ADDRESS1	ADDRESS0	----	----	----	----
Read and write	R	R	R	R	----	----	----	----
Reset value	X	X	X	X	----	----	----	----

Bit7~Bit4 ADDRESS<3:0>: ADC result register bit.
12 bits lower than the result of the 4 bits.
Bit3~Bit0 Unused.

AD data register high ADRESH (9DH), ADFM=1

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADDRESS11	ADDRESS10
Read and write	----	----	----	----	----	----	R	R
Reset value	----	----	----	----	----	----	X	X

Bit7~Bit2 Unused.
Bit1~Bit0 ADDRESS<11:10>: ADC result register bit.
12 bits high as the result of the 2 bit conversion.

AD data register low ADRSL (9CH), ADFM=1

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADDRESS9	ADDRESS8	ADDRESS7	ADDRESS6	ADDRESS5	ADDRESS4	ADDRESS3	ADDRESS2
Read and write	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0 ADDRESS<9:2>: ADC result register bit.

1Bits 2-9 of the conversion result.

Note: In the case of ADFM=1, the AD conversion result only holds 10 bits higher than the 12 bits of the result, where ADRSH saves 2 bits higher ADRESL saves positions 2 through 9.

12. Capture/Compare/PWM Modules (CCP1 and CCP2).

The chip contains 2 capture/compare/PWM (CCP1) and (CCP2). The operation of the CCP1 and CCP2 modules is basically the same.

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

The Capture/Compare/PWM module is a peripheral that allows the user to time and control different events. In capture mode, the peripheral can time the duration of the event. Capture mode allows the user to trigger an external event at the end of a predetermined duration. PWM mode produces pulse width modulated signals with varying frequency and duty cycle.

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

CCPx control register CCP xCON

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CCPxCON	---	---	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
Read and write	---	---	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	DC1B<1:0>:	PWM duty cycle is two bits lower;
	Capture Mode:	Unused;
	Compare mode:	Unused;
	PWM mode:	These two bits are 2 bits lower than the 10-bit PWM duty cycle. The duty cycle is 8 bits higher in THE CCPR1L.
Bit3~Bit0	CCP1M<3:0>:	CCP mode select bit;
	0000=	Capture/Compare/PWM Shutdown (Reset ECCP Module);
	0001=	Unused (reserved);
	0010=	Compare mode, output level flipped when matched (CCP1IF set to 1);
	0011=	Unused (reserved);
	0100=	Capture mode, where capturing occurs on each falling edge;
	0101=	Capturing mode, where capturing occurs on each rising edge;
	0110=	Capture mode, capturing occurs every 4 rising edges;
	0111=	Capture mode, where capturing occurs every 16 rising edges;
	1000=	Compare mode, compare the output high level when matching (CCP1IF set 1);
	1001=	Compare mode, compare the output low level (CCP1IF set to 1);
	1010=	Compare mode, software interrupt when comparing matches (CCP1IF position 1, CCP1 pin is not affected);
	1011=	Compare mode, trigger special events (CCP1IF position 1, CCP1 reset TMR1 or TMR2);
	11xx=	PWM mode.

12.1 Capture mode

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

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

Select the event type by mode selection bits CCPxM3:CCPxM0 (CCPxCON<3:0>). When a catch occurs, the interrupt request flag bit in the PIRx register is set to 1; it must be cleared with software. If another capture occurs before the values in the registers of the CCPRxH and CCPRxL pairs are read, the previously captured values will be overwritten by the newly captured values (see Figure 12-1).

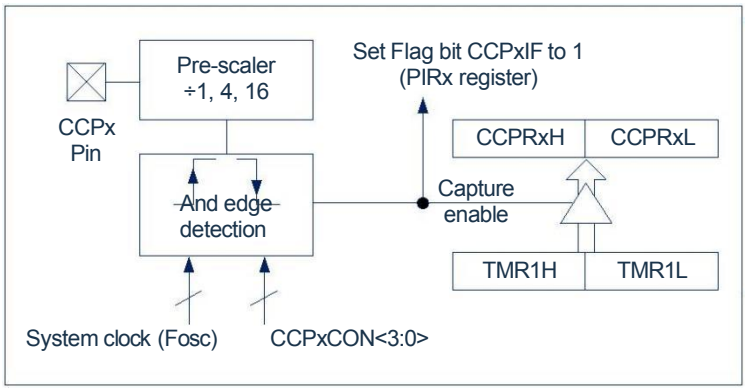


Figure 1 2-1: Capture mode working block diagram

12.1.1 CCP pin configuration

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

Note: If the CCPx pin is configured as an output, a write to the port may raise a catch event.

12.1.2 TIMER1 mode selection

TIMER1 must be running the CCP module in timer mode or synchronous counter mode to use the capture function. Capture operations are not possible in asynchronous counter mode.

12.1.3 Software outage

When the capture mode changes, an erroneous capture interrupt may occur. The user should keep the CCPxIE interrupt in the PIEx register to allow bit been cleared to avoid false interrupts. The interrupt flag bit CCPxIF in the PIRx register should also be cleared after any change in the operating mode.

12.1.4 CCP prescaler

The 3:0 > bits of the CCPxM < in the CCPxCON register specify four prescaler settings that clears the prescaler counter whenever the CCP module is turned off or capture mode is disabled. This means that any reset will clear the prescaler counter.

Switching from one capture prescaler ratio to another capture prescale ratio does not clear the prescale counter, but can produce false interrupts. To avoid this undesirable operation, the module should be turned off by clearing the CCPxCON registers before changing the prescale ratio.

Change the capture prescaler ratio

```
CLR     CCP1CON     ; Turn off CCP1
LDIA    B'00000101'
LD      CCP1CON,A   ; Assign a new value to CCP1
```

12.2 Compare mode

In compare mode, the values of the 16-bit CCPRx registers are constantly compared to the values of a pair of TMR1 registers. When the two match, the CCPx module may have the following situations:

- The output level of ccPx is flipped;
- CCPx output high;
- CCPx output low;
- Generate special event trigger signals;
- Creates a software outage.

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

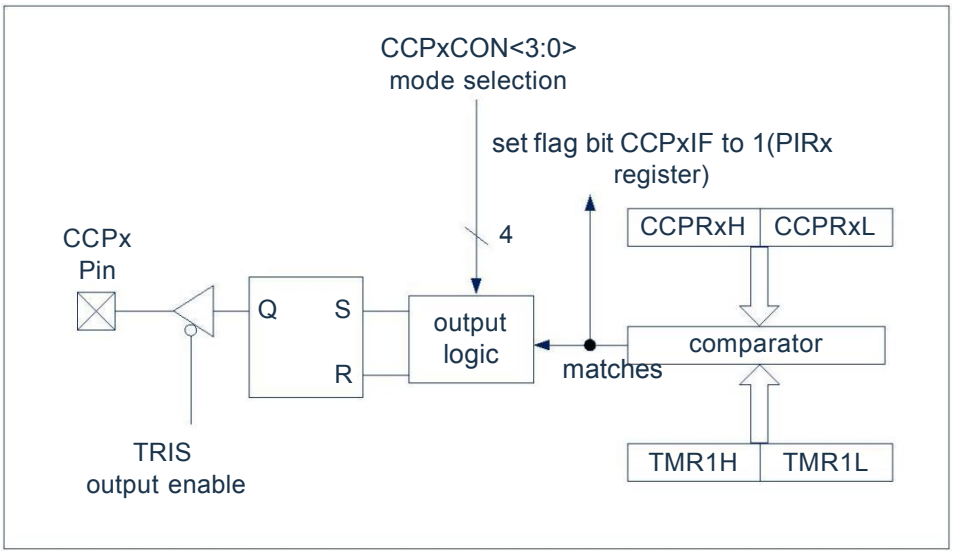


Figure 1 2-2: Compare mode working block diagram

The special event trigger signal will:
- Clear the TMR1H and TMR1L registers.
- Does not make the TMR1IF flag position 1 in the PIR1 register.
- Enable GO/$\overline{DONE}$Position 1 to initiate ADC conversion.

12.2.1 CCP pin configuration

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

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

12.2.2 TIMER1 mode selection

In compare mode, TIMER1 must be running in timer mode or synchronous counter mode. In asynchronous counter mode, you may not be able to compare operations.

12.2.3 Software break mode

When generating software interrupt mode is selected (CCPxM<3:0>=1010), the CCPx module does not control the CCPx pin (see CCPxCON registers).

12.2.4 Special event trigger signal

When a special event firing mode (CCPxM<3:0>=1011) is selected, the CCPx module will do the following:

- Reset TIMER1;
- If the ADC is enabled, the ADC conversion will also be initiated.

In this mode, the CCPx module does not control the CCPx pin (see CCPxCON registers).

When the TMR1H/TMR1L register pair matches the CCPRxH/CCPRxL register pair, the CCP immediately generates a special event trigger output. The TMR1H/TMR1L register pair does not reset until the next rising edge of the TIMER1 clock. Thus the CCPRxH/CCPRxL register pairs effectively become the 16-bit programmable periodic registers of TIMER1.

Concentrate:

1) The special event trigger signal from the CCP module does not make the TMRxIF interrupt flag position 1 in the PIR1 register.
2) Changing the contents of the CCPRxH and CCPRxL register pairs between the edge that generated the special event trigger signal and the clock edge that caused the TIMER1 reset clears the match condition, thus preventing the reset from occurring.

12.3 PWM mode

PWM mode generates a pulse width modulation signal on the CCPx pin, and both PWM1 and PWM2 have their own independent cycle counters that determine the duty cycle, period, and resolution by the following registers

- PWMCON
- PWMxCYC
- CCPRxL
- CCPxCON

PWM control register PWMCON (99H).

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

Bit7 Unused.

Bit6 CYC2EN: The cycle counter enable bit of PWM2.

1= Enable.

0= Forbid.

Bit5-Bit4 CK2[1:0]: PWM2's cycle counter clock prescaler selection bit.

00= The prescale is 1.

01= The prescale is 4.

1X= The prescale is 16.

Bit3 Unused.

Bit2 CYC1EN: The cycle counter enable bit of PWM1.

1= Enable.

0= Forbid.

Bit1~Bit0 CK1[1:0]: PWM1's cycle counter clock prescaler selection bit.

00= The prescale is 1.

01= The prescale is 4.

1X= The prescale is 16.

PWM1 cycle data register PWM1CYC(9AH).

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

PWM2 cycle data register PWM2CYC (9BH).

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

In pulse width modulation (PWM) mode, the CCP module can output PWM signals with resolutions up to 10 bits on the CCPx pin, thanks to CCPx The pins are multiplexed with the port data latch, and the corresponding TRIS bit must be cleared to enable the output driver of the CCPx pin.

Note: Clearing the CCPxCON registers will relinquish CCPx's control over the CCPx pins.

Figure 12-3 below shows a simplified block diagram of PWM operation, and Figure 12-4 shows the typical waveform of a PWM signal.

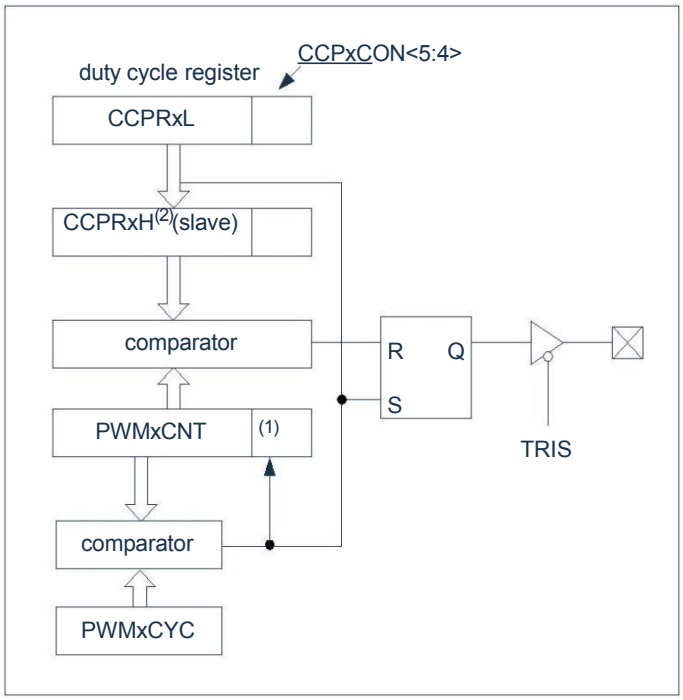


Figure 12-3: Simplified block diagram of PWM

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

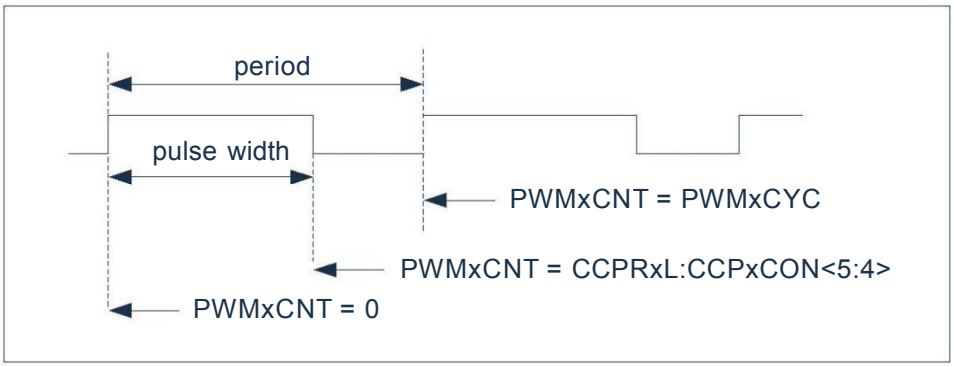


Figure 12-4: CCPPWM output

12.3.1 PWM cycle

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

Equation 1: PWM Period Calculation Formula:

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

NOTE: $T_{OSC}=1/F_{SYS}$

When PWMxCNT is equal to PWMxCYC, the following 3 events occur in the next increment count cycle:

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

12.3.2 PWM duty cycle

The PWM duty cycle can be specified by writing a single 10-bit value to multiple registers: DCxB of the CCPRxL registers and THE CCPXCON registers< 1:0 > bits. THE CCPRxL holds 8 bits higher than the duty cycle, while the DCxB of the CCPxCON registers < 1:0> bits save 2 bits lower than the duty cycle. DCxB of the CCPRxL and CCPxCON registers can be written at any time< 1:0 > bits, but not until PWMxCYC and PWMxCNT When the values in the match (i.e. the end of the cycle), the value of the duty cycle is latched into the CCPRxH. In PWM mode, CCPRxH is a read-only register.

Equation 2: Pulse width calculation formula:

$$\text{Pulse Width}=(CCPRxL:CCPxCON<5:4>)*T_{OSC}*(PWMxCNT\text{pre-scaler value})$$

Equation 3: PWM Duty Cycle Calculation Formula:

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

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

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

When the 10-bit time base matches the value of the CCPRxH and 2-bit latches, the CCPx pin is cleared (see Figure 12-3).

12.3.3 PWM resolution

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

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

Equation 4: PWM resolution:

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

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

The following table gives the values for the frequency and resolution of PWM at F_{SYS}=8MHz.

Example of PWM frequency and resolution (F_{SYS}=8MHz).

PWM frequency	1.22KHz	4.90KHz	19.61KHz	76.92KHz	153.85KHz	200.0KHz
Timer prescaler value (1, 4, or 16).	16	4	1	1	1	1
PWMxCYC 值	0x65	0x65	0x65	0x19	0x0C	0x09
Highest resolution (bits)	8	8	8	6	5	5

12.3.4 Operations in sleep mode

In sleep mode, the PWMxCNT registers will not be incremented and the state of the module will remain unchanged. If the CCPx pin has an output, the output value will continue to be left unchanged. When the device is woken up, the PWMxCNT continues to work from its original state.

12.3.5 A change in the system clock frequency

The PWM frequency is generated by the system clock frequency, and any change in the system clock frequency will change the PWM frequency.

12.3.6 Effects of reset

Any reset forces all ports into input mode and forces the CCP registers into their reset state.

12.3.7 Set up PWM operations

The following steps should be performed when configuring the CCP module for PWM operating mode:

1. By placing the corresponding TRIES position 1, the output driver of the PWM pin (CCPx) is disabled, making it an input pin.
2. Set the PWM cycle by loading the PWMxCYC registers.
3. Configure the PWM mode of the CCP module by loading the CCPxCON registers with the appropriate values.
4. Set the PWM duty cycle by loading the DCxB < 1:0 > bits in the CCPRxL registers and the CCPxCON registers.
5. Configure and start the PWMxCNT cycle counter:
   - Clears the TMR2IF interrupt flag bit in the PIR1 register.
   - The PWMxCNT prescaler ratio is set by loading the CK2 or CK1 bits of the PWM CON register.
   - PWMxCNT is enabled by placing CYC2EN and CYC1EN position 1 in the PWM CON register.
6. After the start of a new PWM cycle, enable the PWM output:
   - Wait for PWMxCNT to overflow.
   - Enable the CCPx pin output driver by Clearing the corresponding TRIS bit.

13. Universal Synchronous/Asynchronous Transceiver (USART)

The Universal Synchronous/Asynchronous Transceiver (USART) module is a serial I/O communication peripheral. The module includes all clock generators, shift registers, and data buffers necessary to perform input or output serial data transfers independent of device program execution. USART, also known as serial communication interface (SERIALCommunicationsInterface, SCI), can be configured as a full-duplex asynchronous system that can communicate with peripherals such as CRT terminals and personal computers It can also be configured as a half-duplex synchronization system capable of communicating with A/D or D/A integrated circuits, peripherals such as serial EEPROM, or other microcontrollers. Microcontrollers that communicate with them typically do not have an internal clock that generates baud rate, and it requires an external clock signal from a master synchronization device.

The USART module contains the following functions:

- Full-duplex asynchronous send and receive
- Single-character output buffer
- Two-character input buffer
- Frame error detection of received characters
- Half-duplex synchronous slave mode
- Character lengths can be programmed to 8 or 9 bits
- Input buffer overflow error detection
- Half-duplex synchronous master mode
- In synchronous mode, the clock polarity is programmable

Figures 15-1 and 15-2 below show a block diagram of the USART transceiver.

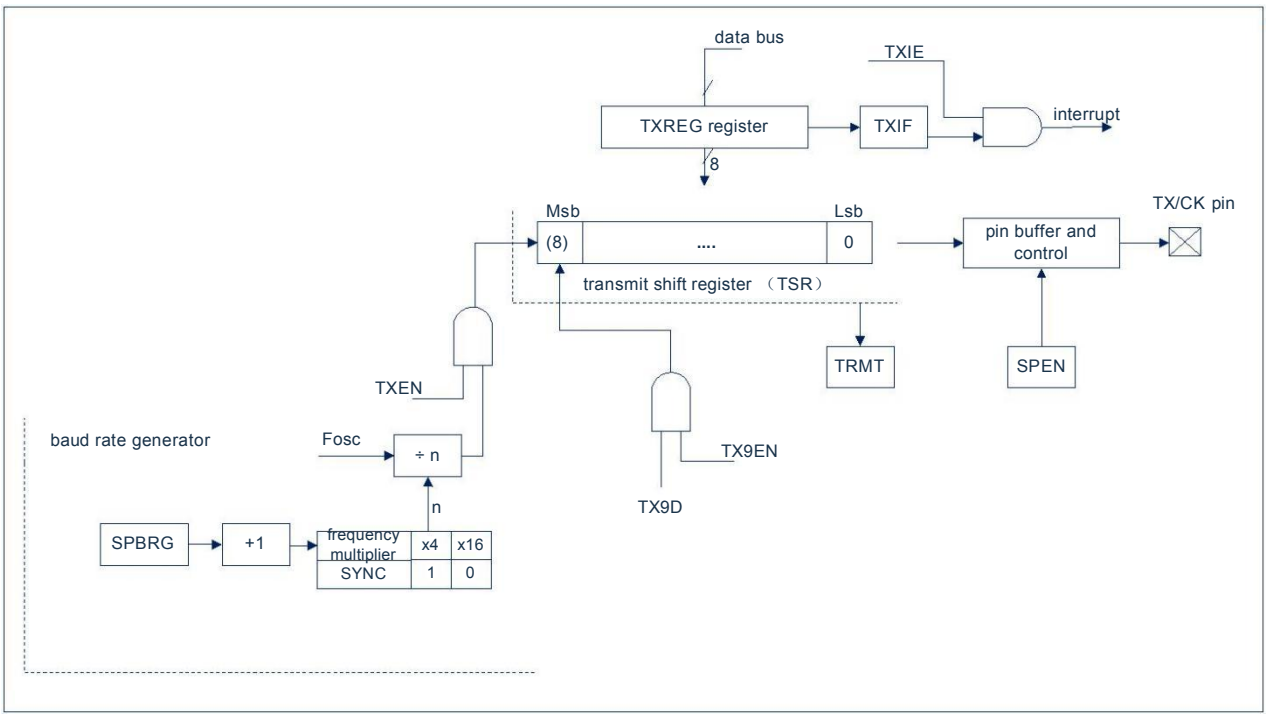


Figure 13-1: USART sends a block diagram

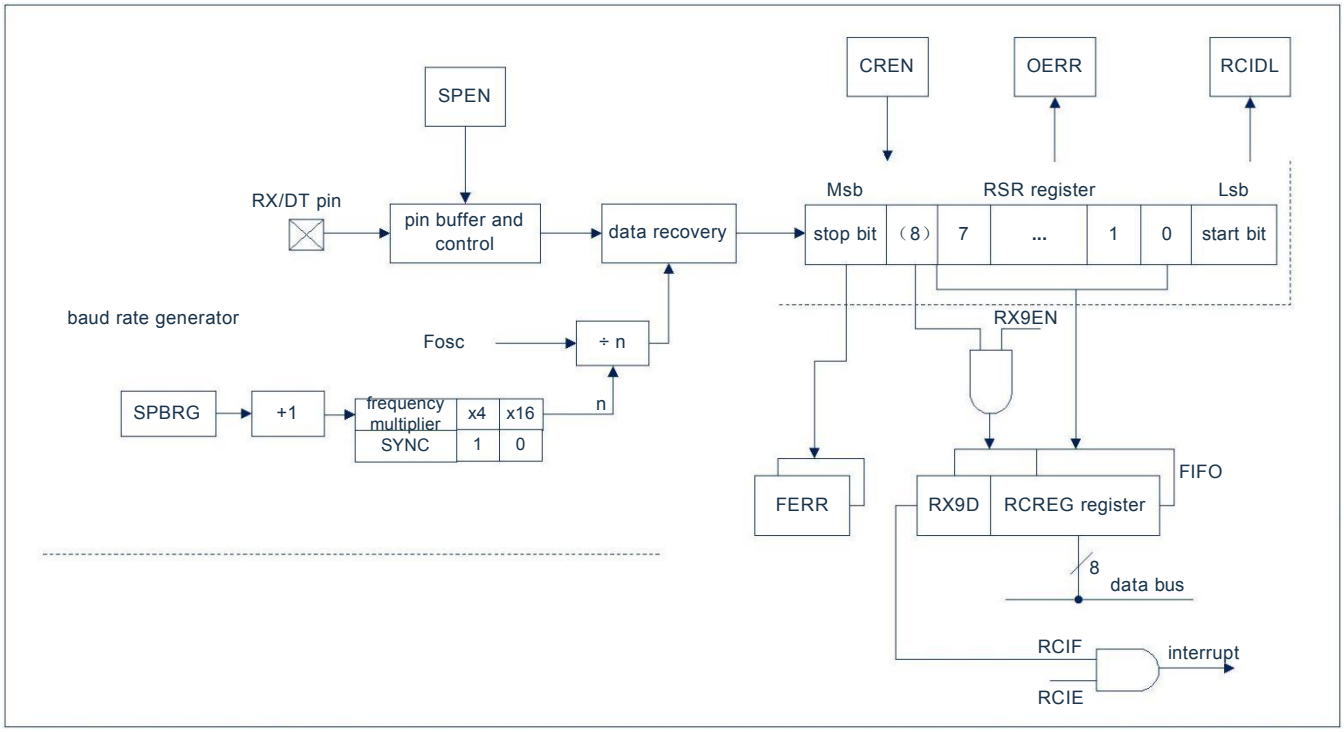


Figure 13-2: USART receive block diagram

The operation of the USART module is controlled by following registers:

- Transmit Status and Control Registers (TXSTA).
- Receive Status and Control Registers (RCSTA).

13.1 USART asynchronous mode

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

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

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

13.1.1 USART asynchronous generator

Figure 14-1 shows the block diagram of the USART transmit device. The core of the transmit device is the serial transmit shift register (TSR), which cannot be directly accessed by software. TSR obtains data from the TXREG transmit buffer register.

13.1.1.1 Enable transmit

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

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

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

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

Note:

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

13.1.1.2 Transmit data

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

13.1.1.3 Transmit interrupt

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

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

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

13.1.1.4 TSR status

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

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

13.1.1.5 Transmit 9-bit character

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

13.1.1.6 Configure asynchronous transmit

1. Initialize the SPBRG register to obtain the required baud rate (refer to "USART baud rate generator (BRG) section".

2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit to 1.

3. If 9-bit transmit is required, set the TX9EN control bit to 1. When the receiver is set for address detection, set the 9th bit of the data bit to 1, indicating that the 8 lowest data bits are address.

4. Set the TXEN control bit to 1 to enable transmit; this will cause the TXIF interrupt flag bit to be set to 1.

5. If interrupt is required, set the TXIE interrupt enable bit in PIE1register to 1; if the GIE and PEIE bits in the INTCON register are also set to 1, interrupt will occur immediately.

6. If you choose to transmit 9-bit data, the 9th bit should be loaded into the TX9Ddata bit.

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

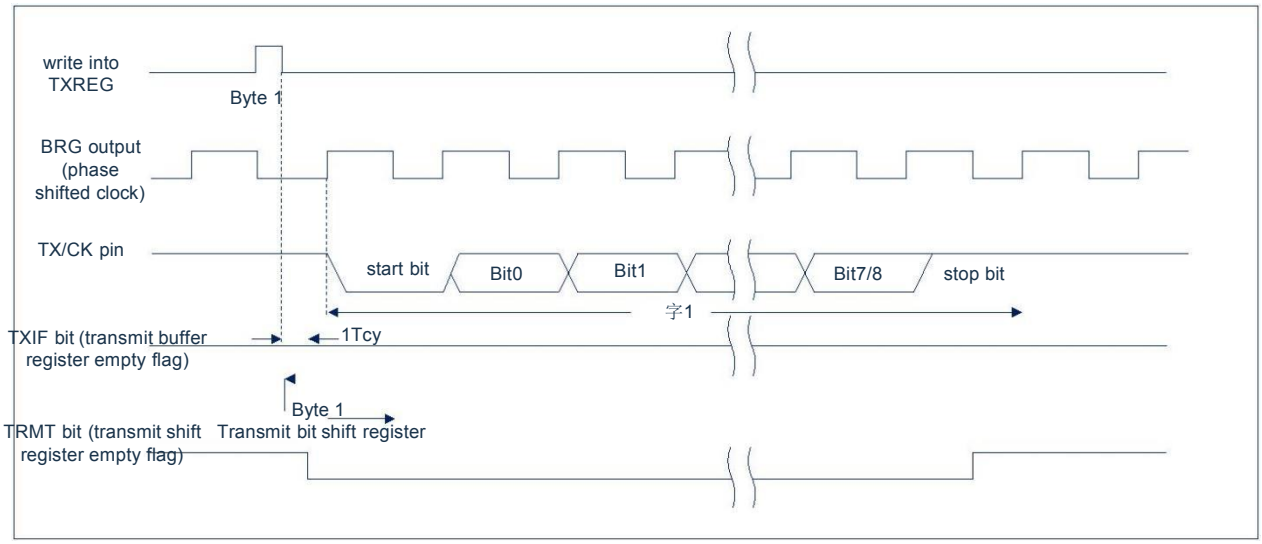


Fig 13-3: asynchronous transmit

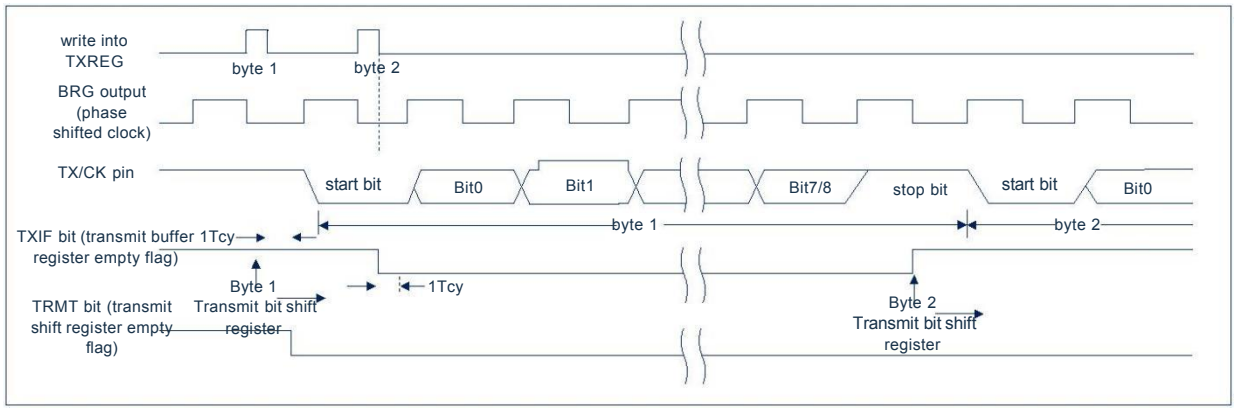


Fig13-4: asynchronous transmit (back to back)

Note: This time series diagram shows two consecutive transmit.

13.1.2 USART asynchronous receiver

Asynchronous mode is usually used in RS-232 system. Figure 13-2 shows the block diagram of the receiver. Receive data and driver data recovery circuit on RX/DT pin. The data recovery circuit is actually a 16 times baud rate as the operating frequency High-speed shifter, while the serial receive shift register (Receive Shift Register, RSR) works at the bit rate. When all the 8-bit or 9-bit data bits of the character are shifted in, they are immediately transferred to a 2-character FIFO (FIFO) buffer. FIFO buffer allows to receive 2 complete characters and the start bit of the third character, and then software must provide the received data to the USART receiver. FIFO and RSR register cannot be directly accessed by software. The RCREG register accesses the received data.

13.1.2.1 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.

13.1.2.2 Receive data

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

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

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

13.1.2.3 Receive interrupt

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

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

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

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

13.1.2.4 Receive frame error

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

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

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

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

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

13.1.2.5 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.

13.1.2.6 Receive 9-bit character

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

13.1.2.7 Asynchronous receive configuration

1. Initialize the SPBRG register to obtain the required baud rate.

(Please refer to the "USART baud rate generator (BRG)" section)

2. Set the SPEN bit to 1 to enable the serial port. The SYNC bit must be cleared to perform asynchronous operations.

3. If interrupt is required, set the RCIE bit in the PIE1 register and the GIE and PEIE bits in the INTCON register to 1.

4. If you need to receive 9 bits of data, set the RX9EN bit to 1.

5. Set the CREN bit to 1 to enable receive.

6. When a character is transferred from the RSR to the receive buffer, set the RCIF interrupt flag bit to 1. If the RCIE interrupt enable bit is also set to 1, an interrupt will also be generated.

7. Read the RCREG register and get the received 8 low data bits from the receive buffer.

8. Read the RCSTA register to get the error flag bit and the 9th data bit (if 9-bit data receive is enabled).

9. If overflow occurs, clear the OERR flag by clearing the CREN receiver enable bit.

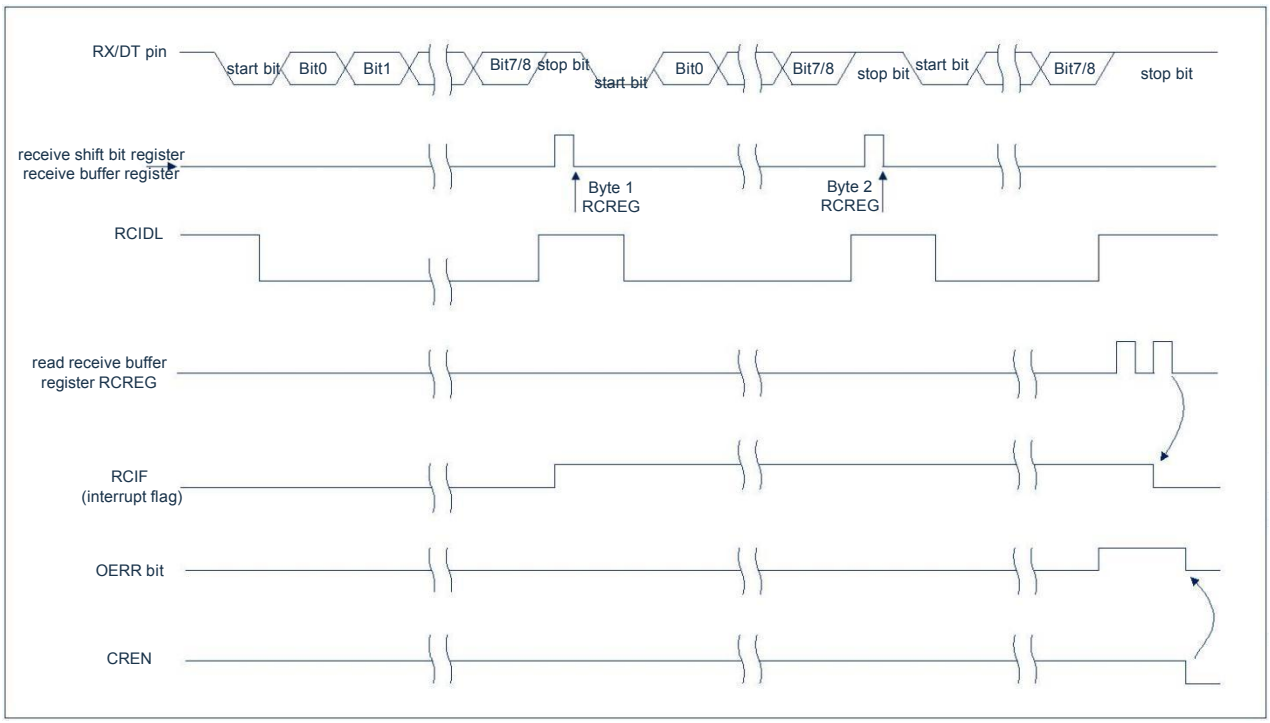


Fig 13-5: asynchronous receive

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

13.2 Clock accuracy during asynchronous operations

The output of the internal oscillation circuit (INTOSC) is calibrated by the manufacturer. However, when VDD or temperature changes, INTOSC will undergo frequency drift, which directly affects the asynchronous baud rate. The baud rate clock can be adjusted in the following ways, but some type of reference clock source is required.

13.3 USART-related registers

TXSTA: Transmit status and control registers (1EH).

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXSTA	CSRC	TX9EN	TXEN	SYNC	SCKP	--	TRMT	TX9D
Read/Write	R/W	R/W	R/W	R/W	R/W	--	R	R/W
Reset value	0	0	0	0	0	0	1	0

Bit7 CSRC: Clock source select bit;
Asynchronous mode: Any value;
Synchronization mode:
1 = Master mode (clock signal generated by the internal BRG);
0 = Slave mode (clock generated by an external clock source).

Bit6 TX9EN: 9 bits transmit enable bits;
1= Select 9 bits to send;
0= Select 8 bits to send.

Bit5 TXEN: Send enable bit (1);
1= Enable sending;
0= Prohibit sending.

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

Bit3 SCKP: Synchronous clock polarity select bit.
Asynchronous mode:
1 = Reverses the level of the data character and sends it to the TX/CK pin;
0 = Send data characters directly to the TX/CK pin.
Synchronization mode:
0 = Transmit data on the rising edge of the clock;
1 = Transmit data on the clock descent edge.

Bit2 Not used

Bit1 TRMT: Transmit shift register status bits;
1= TSR is empty;
0= The TSR is full.

Bit0 TX9D: The 9th bit of the data sent.
Can be address/data bits or parity bits.

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

RCSTA: Receive status and control registers (18H).

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
Reset value	0	0	0	0	1	0	0	0

Bit7 SPEN: Serial port enable bit;
1= Enable serial port (configure the RX/DT and TX/CK pins as serial port pins);
0= Disable serial ports (remain in the reset state).

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

Bit5 SREN: Single-byte receive enable bits.
Asynchronous mode: Any value.
Synchronous master mode:
1 = Enables single-byte receive;
0 = Single-byte reception is prohibited.
Zeros the bit after the receive is complete.
Synchronous slave mode: Any value.

Bit4 CREN: Continuously receive enable bits.
Asynchronous mode:
1 = Enable receive;
0 = No receive.
Synchronization mode:
1 = Enable continuous reception until zero CREN enable bit (CREN overrides SREN);
0 = Continuous reception is prohibited.

Bit3 RCIDL: Receives the idle flag bit.
Asynchronous mode: 1 = receiver idle;
0 = The start bit has been received and the receiver is receiving data.
Synchronization mode: Any value.

Bit2 FERR: Frame error bit.
1= Frame error (can be updated by reading the RCREG register and receives the next valid byte);
0= There are no frame errors.

Bit1 OERR: Overflow error bit.
1= Overflow error (can be cleared by clearing the CREN bit);
0= There are no overflow errors.

Bit0 RX9D: Bit 9 of the data received.
This bit can be an address/data bit or a parity bit and must be calculated by the user firmware.

13.4 USART Baud Rate Generator (BRG)

The baud rate generator (BRG) is an 8-bit designed to support the asynchronous and synchronous operating modes of USART.

The SPBRG register pair determines the period of the freely operating baud rate timer.

Table 13-1 contains the formula for calculating the baud rate. Equation 1 is an example of calculating baud rate and baud rate error.

Table 13-1 shows the typical baud rate and baud rate error values in various asynchronous modes that have been calculated for your convenience.

Writing a new value to the SPBRG register pair causes the BRG timer to reset (or clear to zero). This ensures that THE BRG can output a new baud rate without waiting for the timer to overflow.

If the system clock changes during a valid receive, a receive error can occur or cause data loss. 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.

Equation 1: Calculate the baud rate error

For F_{SYS} is 8MHz, the target baud rate is 9600bps, and the asynchronous mode uses 8-bit BRG devices:

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

Solve SPBRG:

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

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

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

Table 13-1: Baud rate formula

Configuration bits	BRG/USART mode	Baud rate formula
SYNC		
0	8 bits/asynchronous	Fsys/[16(n+1)]
1	8 bits/synchronous	Fsys /[4(n+1)]

Description: n= The value of the SPBRG register.

Table 13-2: Baud rates in asynchronous mode

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

13.5 USART synchronous mode

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

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

13.5.1 Synchronous master control mode

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

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

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

13.5.1.1 master control clock

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

13.5.1.2 Clock polarity

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

13.5.1.3 Synchronous master control transmit

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

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

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

13.5.1.4 Synchronous master control transmit configuration

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

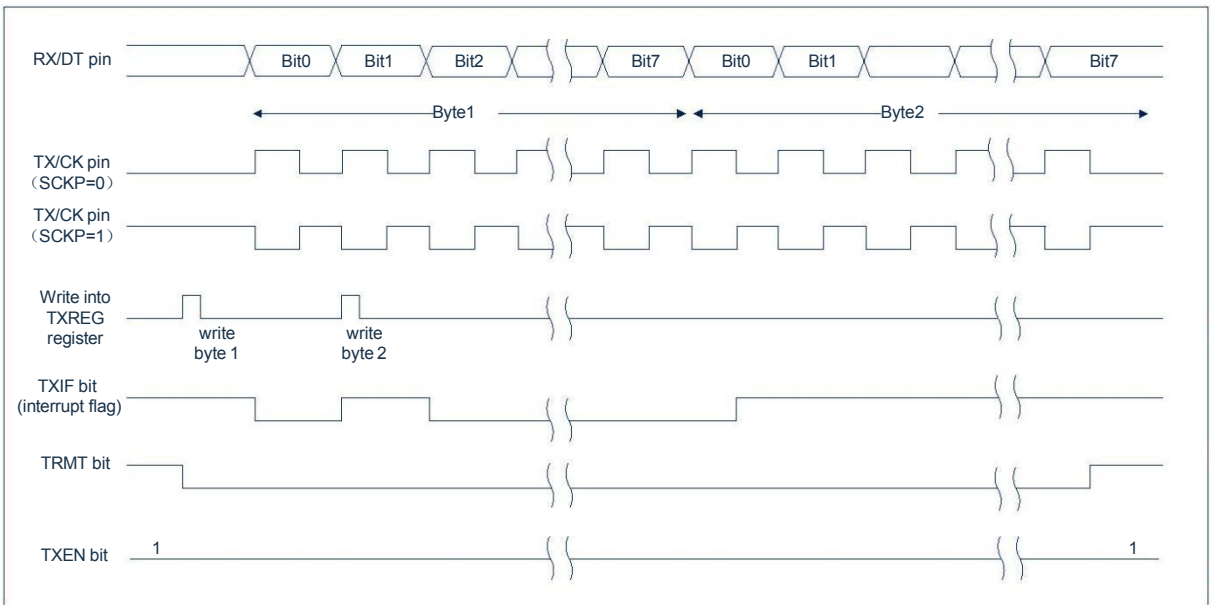


Fig 13-6: synchronous transmit

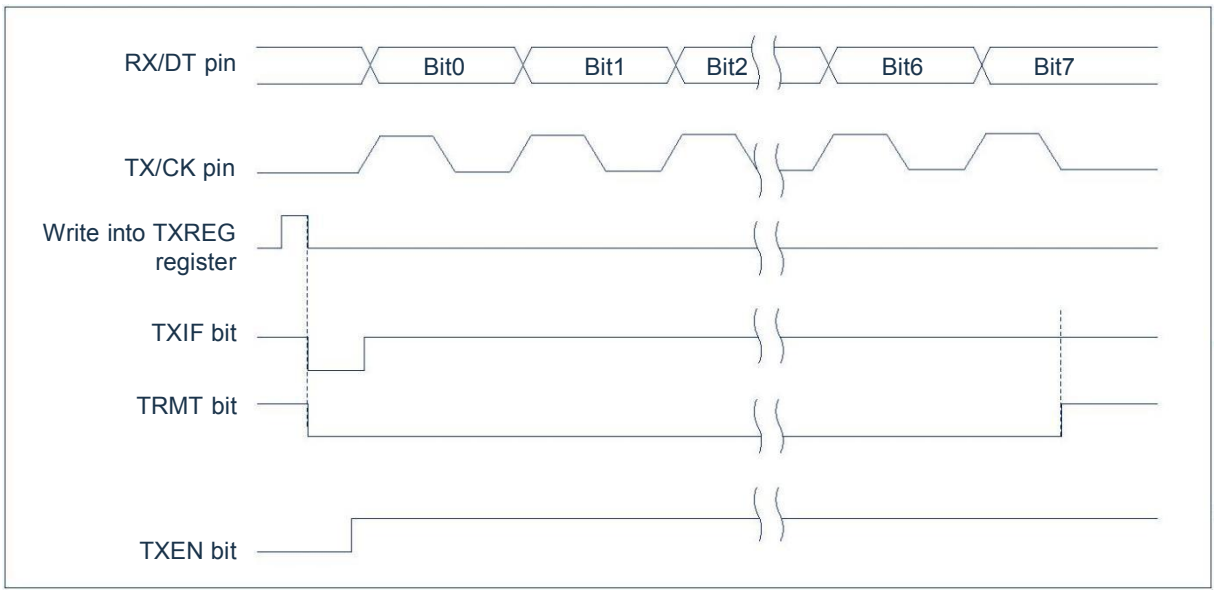


Fig 13-7: synchronous transmit (through TXEN)

13.5.1.5 Synchronous master control receive

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

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

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

13.5.1.6 Slave clock

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

13.5.1.7 Receive overflow error

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

13.5.1.8 Receive 9-bit character

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

13.5.1.9 Synchronous master control receive configuration

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

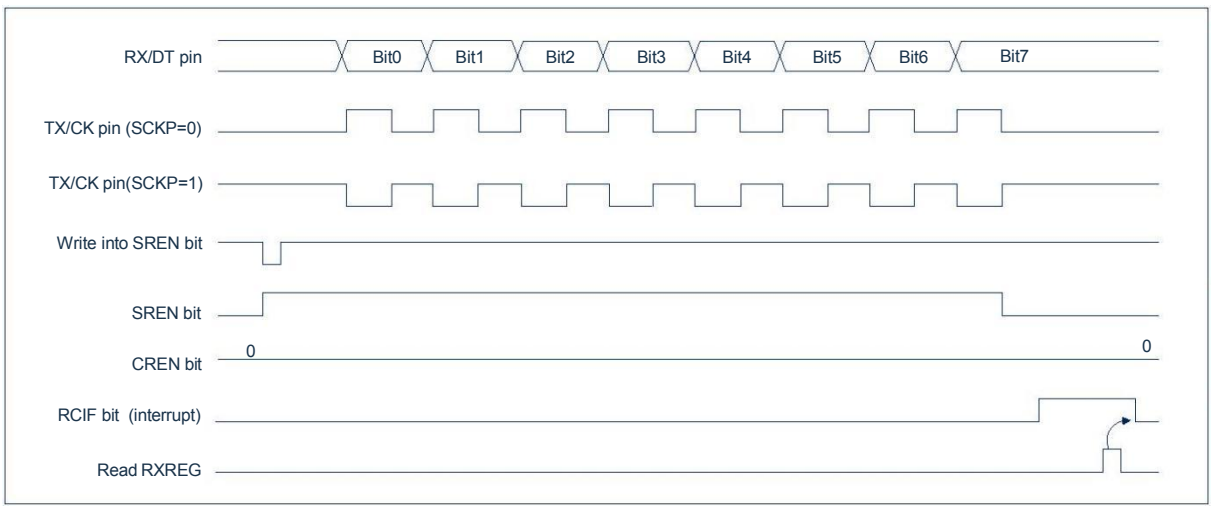


Fig 13-8: synchronous receive (master control mode，SREN)

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

13.5.2 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.

13.5.2.1 USART synchronous slave transmit

The working principle of synchronous master control and slave mode is the same (refer to section "synchronous master control transmission").

13.5.2.2 Synchronous slave transmit configuration

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

13.5.2.3 USART synchronous slave receive

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

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

13.5.2.4 Synchronous slave receive configuration

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

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

14. Program EEPROM and program memory control

14.1 overview

The devices in this family have 8K word program memory with addresses ranging from 000h to 1FFFh, which is read-only in all address ranges, and the devices have a 128-byte program EEPROM with an address range of 0h to 07 Fh, which is readable and writable in all address ranges.

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

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

When the program EEPROM is accessed, the EEDAT register holds 8 bits of read and written data, while the EEADR register holds the address of the accessed program EEPROM unit.

When accessing the device's program memory, the EEDAT and EEDATH registers form a double-byte word to hold the 16-bit data to be read, and the EEADR and EEADRH registers form a double-byte word to save what is to be read 13-bit EEPROM unit address.

Program memory enables reading in word units. The program EEPROM enables byte reads and writes. A byte write operation automatically erases the target unit and writes new data (erased before writing).

Write time is controlled by an on-chip timer. The write and erase voltages are generated by an on-chip charge pump that is rated to operate within the voltage range of the device for byte or word operation.

While the device is code protected, the CPU can continue to Read/Write program EEPROM and program memory. When code protected, the device programmer will no longer be able to access the program EEPROM or program memory.

Note:

1) Program memory refers to the ROM space, that is, the space where the instruction code is stored, which is only readable;
   The program EEPROM is a space that can store user data, which can be read and written.
2) The program EEPROM normal write voltage range is 3. 0V~5.5V, write current is 20Ma@VDD=5V

14.2 Related registers

14.2.1 EEADR and EEADRH registers

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

When the program memory address value is selected, the high bytes of the address are written to the EEADRH register and the low bytes are written to the EEADR register. When selecting the program EEPROM address value, only the low bytes of the address are written to the EEADR register.

14.2.2 EECON1 and EECON2 registers

EECON1 is the control register that accesses the program EEPROM.

The control bit EPPGD determines whether to access the program memory or the program EEPROM. When the bit is cleared, as in the reset, any subsequent operations will be performed on the program EEPROM. At this position 1, any subsequent operations will be performed on the program memory. The program memory is read-only.

The control bits RD and WR initiate reads and writes, respectively. With the software, only these positions 1 can be cleared and cannot be cleared. After a read or write operation completes, they are clear to zero by the hardware. Since the WR bit cannot be clear to zero with software, it is possible to avoid accidental premature termination of write operations.

- When WREN is set to 1, write operations to the program EEPROM are enableed. On power-up, the WREN bit is cleared to zero. When a normal write operation is interrupted by an LVR reset or A WDT timeout reset, the WRERR bit is set to 1. In these cases, after reset the user can check the WRERR bits and rewrite the corresponding cells.
- When the write operation is complete, the interrupt flag bit EEIF in the PIR1 register is set to 1. This flag bit must be cleared with software.

EECON2 is not a physical register. Reading EECON2 gets all 0s.

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

EEPROM data register EEDAT (10CH).

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

Bit7~Bit0	EEDAT<7:0>:	The low 8 bits of data to be read from or written to the program EEPROM, or the low 8 bits of data to be read from the program memory.

EEPROM address register EEADR (10DH).

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

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

EEPROM data register EEDATH (18EH).

18EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

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

EEPROM address register EEADRH (10FH).

10FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADRH	---	---	---	EEADRH4	EEADRH3	EEADRH2	EEADRH1	EEADRH0
Read/Write	---	---	---	R/W	R/W	R/W	R/W	R/W
Reset value	---	---	---	0	0	0	0	0

Bit7~Bit5 Unused, read as 0.

Bit4~Bit0 EEADRH<4:0>: Specifies the high 5-bit address of the program memory read operation.

EEPROM control register EECON1 (11BH).

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

Bit7 EEPGD: Program/Program EEPROM select bits;
1= Operator memory;
0= Operating procedure EEPROM.

Bit6 Unused

Bit5~Bit4 EETIME[1:0] Maximum flashing wait time; **(For more EETIME information, see Figure 14-1).**
00= 1.25ms
01= 2.5ms(VDD=4.0~5.5V,TEMP=0~85°C recommended value)
10= 5ms
11= 10ms (recommended values for conditions other than 2.5ms)

Bit3 WRERR: EEPROM error flag bit;
1= Write error (any WDT reset or undervoltage reset during normal operation, or But the self-validation has not yet succeeded);
0= The write operation is complete.

Bit2 WREN: EEPROM writes the enable bit;
1= Enable write cycles;
0= Disable writing to storage.

Bit1 WR: Write control bits;
1= Start the write cycle (once the write operation is completed by the hardware to clear
0= The write cycle is complete.

Bit0 RD: Read the control bit;
1= Initiates a memory read operation (RD is cleared by hardware, and the software can
0= The memory read operation is not initiated.

14.3 Read the 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 a second instruction cycle to read the data. This causes the second instruction immediately following the "SETBEECON1, RD" instruction to be ignored (1). In the immediate next clock cycle, the value of the corresponding address of the program EEPROM is latched into the EEDAT registers, which the user can read in subsequent instructions. EEDAT will save this value until the next time the user reads or writes data to the cell.

Note: The two instructions after the program memory read operation must be NOP. This prevents the user from executing a double-cycle instruction at the next instruction after RD position 1.

Example: Read the program EEPROM

```
EEPDATA_READ:
        LD      A,RADDR         ; Put the address you want to read into the EEADR register
        LD      EEADR,A
        CLRB    EECON1,EEPGD    ; Access the datastore
        SETB    EECON1,RD       ; Initiates a read operation
        NOP
        NOP
        LD      A,EEDAT         ; Read data to ACC
        LD      RDATA,A
EEPDATA_READ_BACK:
        RIGHT
```

14.4 Write the program EEPROM

To write a program EEPROM memory unit, the user should first write the address of the unit to the EEADR register and write the data to the EEDAT register. The user must then start writing each byte in a specific order.

If not in exactly the following instruction order (i.e. first write 55h to EECON2, then Aah to EECON2, and finally WR position 1). Write each byte, no write operation will be initiated. Breaks should be suppressed in the code snippet.

In addition, the WREN position 1 in EECON1 must be changed to enable write operations. This mechanism prevents the EEPROM from being written incorrectly due to code execution errors (exceptions) (i.e. programs running away). When not updating the EEPROM, the user should always keep the WREN bit clear. The WREN bit cannot be zeroed by hardware.

After a write process is initiated, clear the WREN bit will not affect this write cycle. Unless WREN is positioned 1, the WR bit will not be able to place 1. When the write cycle is complete, the WR bit is cleared by the hardware and the EE Write Completion Interrupt Flag Bit (EEIF) is set to 1. The user can enable this interrupt or query this bit. EEIF must be clear to zero with software.

Note: During the write of the program EEPROM, the CPU will stop working, and the CLRWDT command needs to be executed before the write operation starts to avoid WDT overflowing the reset chip during this time.

Example: Write the program EEPROM

```
EEPDATA_WRITE:
        LD      A,WADDR         ; Put the address you want to write into the EEADR register
        LD      EEADR,A
        LD      A,WDATA         ; Send the data to be written to the EEDAT register
        LD      EEDAT,A
        CLRWDT
        CLR     EECON1
        SETB    EECON1,EETIME0
        SETB    EECON1,EETIME1  ; EE flashing time 10ms, user-definable
        CLRB    EECON1,EEPGD    ; Access the datastore
        SETB    EECON1,WREN     ; Write cycles are enableed
        CLRB    F_GIE_ON        ; Save the interrupt turned on state
        SZB     INTCON,GIE
        SETB    F_GIE_ON
        CLRB    INTCON,GIE      ; Shut down interrupts
        SZB     INTCON,GIE      ; Make sure the interrupt is turned off
        JP      $-2

        LDIA    055H
        LD      EECON2,A
        LDIA    0AAH
        LD      EECON2,A
        SETB    EECON1,WR       ; Initiate a write operation
        NOP
```

```
                NOP
                CLRWDT
                CLRB        EECON1,WREN         ; Write ends, closes the write enable bit

                SZB         F_GIE_ON            ; Resume the interrupt-on state
                SETB        INTCON,GIE

                SNZB        EECON1,WRERR        ; Determine whether the EEPROM write operation is error-prone
                JP          EEPDATA_WRITE_BACK
                SZDECR      WERR_C              ; The count timeout exits and the user can customize it
                JP          EEPDATA_WRITE       ; The EEPROM write operation is rewritten if it fails
EEPDATA_WRITE_BACK:
                RIGHT
```

14.5 Read program memory

To read the program memory cell, the user must write the high and low bits of the address to the EEADR and EEADRH registers respectively, to the EEPCD position 1 of the EECON1 register, and then to the control bit RD set to 1. Once the read control bit is set, the program memory controller uses a second instruction cycle to read the data. This causes the second instruction immediately following the " SETBEECON1, RD" instruction to be ignored. In the immediate next clock cycle, the value of the corresponding address in the program memory is latched into the EEDAT and EEDATH registers, which the user can read in subsequent instructions. The EEDAT and EEDATH registers will hold this value until the next time the user reads or writes data to the unit.

Note:

1) The two instructions after the program memory read operation must be NOP. This prevents the user from executing a double-cycle instruction at the next instruction after RD position 1.
2) When EPGD=1 if WR position 1, it immediately resets to 0 without doing anything.

Example: Read flash program memory

```
        LD      A,RADDRL        ; Put the address you want to read into the EEADR register
        LD      EEADR,A
        LD      A,RADDRH        ; Put the address bit to be read into the EEADRH register
        LD      EEADRH,A
        SETB    EECON1,EEPGD    ; Select the operator memory
        SETB    EECON1,RD       ; Read operations are enableed
        NOP
        NOP
        LD      A,EEDAT         ; Saves the read data
        LD      RDATL,A
        LD      A,EEDATH
        LD      RDATH,A
```

14.6 Write program memory

Program memory is read-only and not writable.

14.7 Procedure EEPROM operation considerations

14.7.1 Programming time of the program EEPROM

Program EEPROM flashing time is not fixed, the time required to burn different data is not the same, ranging from 100us ~ 10ms, the EETIME bit of the EECON1 register determines the maximum time that the program EEPROM is programmed, program E The EPROM module has a built-in self-verification function, during the programming process, the self-verification is successful or the time set by EETIME has arrived, and the write operation will end when one of the conditions is met. During the programming period, the CPU stops working, the peripheral module works normally, and the program needs to do the relevant processing.

14.7.2 Number of times the program EEPROM is burned

The number of programming times of the program EEPROM, the programming time of the EETIME setting, and the voltage and temperature are related, which can be referred to the following diagram.

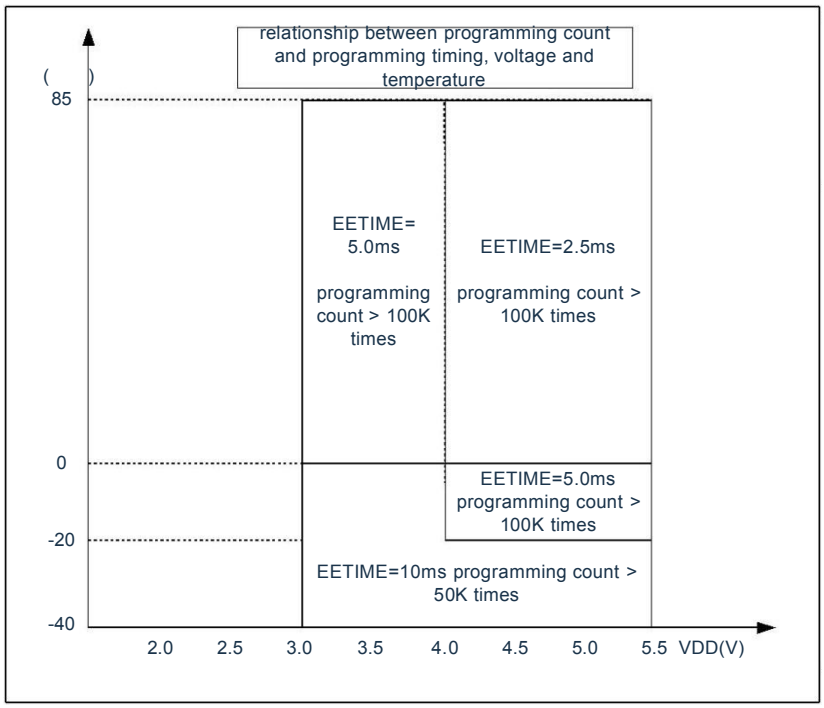


Fig. 14-1 The relationship between the number of program EEPROM flashes and the flashing time, voltage, and temperature

14.7.3 Write validation

Depending on the application, good programming practices generally require that the value written to the program EEPROM be checked against the expected value.

14.7.4 Protection against miswriting

In some cases, the user may not want to write data to the program EEPROM. To prevent accidental writing of EEPROM, various protection mechanisms are embedded in the chip. Zero the WREN bit at power-up. Also, a power-up delay timer (delay time of 16ms) prevents writes to the EEPROM.

The initiation sequence of the write operation, along with the WREN bit, together will prevent the write-error operation from occurring in the following cases:

- Undervoltage
- Power supply glitches
- Software failure

15. Touch key

15.1 Touch key module overview

The touch detection module is an integrated circuit designed to implement the human touch interface. It can replace the mechanical light touch button to achieve waterproof and dustproof, sealed isolation, and solid and beautiful operation interface.

Technical parameters:

- 1-8 keystrokes are selectable, and all I/Os can be configured as touch channels
- No external touch capacitors are required
- High anti-interference performance, can easily pass static 15V, dynamic 10V conduction test

15.2 Touch key module usage considerations

- The ground wire of the touch key detection part should be connected separately into a separate ground, and then a point should be connected to the common ground of the whole machine.
- Avoid overlapping placement of high-voltage, high-current, high-frequency operation motherboards and touch circuit boards. If there is no way to avoid it, it should be as far away from the intertemporal zone of high voltage and large current as possible or shielded on the motherboard.
- The wire from the sensor disc to the touch chip should be as short and thin as possible, if the PCB process allows as much as possible to use a line width of 0.1mm.
- The connection from the sensor disk to the touch chip should not cross the letter line with strong interference and high frequency.
- Feel the disk to the touch chip wiring circumference of 0.5mm do not take other signal lines.

16. Electrical parameters

16.1 Limit parameters

Power 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 sink current for all ports……200mA
Maximum pull current for all ports……-150mA

Note: If the operating conditions of the device exceed the "limit parameters" described above, permanent damage may be caused to the device. The above values are extremely high operating conditions only and we do not recommend that the device operate outside the range specified in this specification. The stability of the device is affected when operating at limit values for a long time.

16.2 DC electrical characteristics

symbol	parameter	Test conditions		minimum	Typical values	maximum	unit
		VDD	condition				
VDD	Operating voltage		F_{SYS}=16MHz	2.6		5.5	V
			F_{SYS}=8MHz	1.8		5.5	V
I_{DD}	Operating current	5V	F_{SYS}=16MHz		3. 8		mA
		3V	F_{SYS}=16MHz		3.0		mA
		5V	F_{SYS}=8MHz		3.0		mA
		3V	F_{SYS}=8MHz		2.3		mA
I_{STB}	Quiescent current	5V	----		0.1	2	μA
		3V	----		0.1	1	μA
V_{IL}	Low input voltage		----			0.3VDD	V
$IN_{1999, THE NEW}$	High-level input voltage		----	0.7VDD			V
V_{OH}	High output voltage		Without load	0.9VDD			V
In_{OL}	Low output voltage		Without load			0.1VDD	V
V_{EEPROM}	The EEPROM module erases the write voltage		----	3.0		5.5	V
R_{PH}	Pull-up resistance value	5V	V_O=0.5VDD		35		kΩ
		3V	V_O=0.5VDD		63		kΩ
R_{PD}	Pull-down resistance value	5V	V_O=0.5VDD		35		kΩ
		3V	V_O=0.5VDD		63		kΩ
I_{OL1}	Inlet and outlet perfusion current (Normal I/O port).	5V	V_{OL}=0.3VDD		60		mA
		3V	V_{OL}=0.3VDD		25		mA
I_{OL2}	Inlet and outlet perfusion current (PALEN/PBLEN position 1).	5V	V_{OL}=0.3VDD		120		mA
		3V	V_{OL}=0.3VDD		50		mA
I_{OH1}	The output port pulls the current (Normal I/O port).	5V	V_{OH}=0.7VDD		-31.8		mA
		3V	V_{OH}=0.7VDD		-12.2		mA
I_{OH2}	The transmission and outlet pull the current (PAHEN/PBHEN position 1).	5V	V_{OH}=0.7VDD SEG_ISEL=2mA		-2.3		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=4mA		-4.5		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=6mA		-6.9		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=8mA		-8.8		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=10mA		-11.1		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=12mA		-13.1		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=14mA		-15.1		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=16mA		-17.2		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=18mA		-19.5		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=20mA		-21.3		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=22mA		-23.7		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=24mA		-25.5		mA
		5V	V_{OH}=0.7VDD		-27.7		mA

			SEG_ISEL=26mA				
		5V	V_{OH}=0.7VDD SEG_ISEL=28mA		-29. 6		mA
		5V	V_{OH}=0.7VDD SEG_ISEL=30mA		-31. 8		mA
In_{BG}	The internal reference voltage is 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℃		-2.0%	1.2	2.0%	V

(VDD=5V, T_A=25°C, unless otherwise noted).

16.3 ADC electrical characteristics

(T_A = 25°C, unless otherwise noted).

symbol	parameter	Test conditions	minimum	Typical values	maximum	unit
V_{ADC}	ADC operating voltage	AD_{VREF}=VDD，F_{ADC}=1MHz	3.0		5.5	V
		AD_{VREF}=VDD，F_{ADC}=500kHz	2.7		5.5	V
		AD_{VREF}=2.4V，F_{ADC}=250kHz	2.7		5.5	V
		AD_{VREF}=2.0V，F_{ADC}=250kHz	2.7		5.5	V
I_{ADC}	ADC conversion current	V_{ADC}=5V,AD_{VREF}=VDD,F_{ADC}=500kHz			500	uA
		V_{ADC} =3V, AD_{VREF}=VDD,F_{ADC}=500kHz			200	uA
IN_{ADI}	ADC input voltage	V_{ADC}=5V, AD_{VREF}=VDD,F_{ADC}=250kHz	0		V_{ADC}	V
DNL	Differential nonlinearity error	V_{ADC}=5V, AD_{VREF}=VDD,F_{ADC}=250kHz			±2	LSB
INL	Integral nonlinearity error	V_{ADC}=5V, AD_{VREF}=VDD,F_{ADC}=250kHz			±2	LSB
I_{ADC}	ADC conversion time	-		16		T_{ADCCLK}

16.4 Power-on reset characteristics

(T_A = 25°C, unless otherwise noted).

symbol	parameter	Test conditions	minimum	Typical values	maximum	unit
t_{VDD}	VDD rise rate	-	0.05			V/ms
V_{LVR1}	LVR set voltage = 1.8V	VDD=1. 6~5.5V	1.7	1.8	1.9	V
In_{LVR2}	LVR set voltage = 2.0V	VDD=1.8~5.5V	1.9	2.0	2.1	V
In_{LVR3}	LVR set voltage = 2.6V	VDD=2.4~5.5V	2.5	2.6	2.7	V

16.5 AC electrical characteristics

(T_A = 25°C, unless otherwise noted).

symbol	parameter	Test conditions		minimum	Typical values	maximum	unit
		VDD	condition				
T_{WDT}	WDT reset time	5V	-		18		ms
		3V	-		18		ms
I_{EEPROM}	EEPROM programming time	5V	F_{OSC}=8MHz/16MHz			10	ms
		3V	F_{OSC}=8MHz/16MHz			10	ms
F_{RC}	Internal vibration frequency stability	VDD=4.5~5.5V T_A=25°C		-1.5%	8	+1.5%	MHz
		VDD=2.0~5.5V T_A=25°C		-2%	8	+2%	MHz
		VDD=4.5~5.5V T_A=-40~85°C		-2.5%	8	+2.5%	MHz
		VDD=2.0~5.5V T_A=-40~85°C		-3.5%	8	+3.5%	MHz
		VDD=4.5~5.5V T_A=25°C		-1.5%	16	+1.5%	MHz
		VDD=2.6~5.5V T_A=25°C		-2%	16	+2%	MHz
		VDD=4.5~5.5V T_A=-40~85°C		-2.5%	16	+2.5%	MHz
		VDD=2.6~5.5V T_A=-40~85°C		-3.5%	16	+3.5%	MHz

17. instructions

17.1 List of instructions

Mnemonics		operation	Instruction cycle	symbol
Control class -3				
NOP		Null action	1	None
STOP		Enter sleep mode	1	TO,PD
CLRWDT		Zero watchdog counter	1	TO,PD
Data transmission -4				
LD	[R],A	Transmit the ACC content to R	1	NONE
LD	A,[R]	Transmit the R capacity to the ACC	1	Z
TESTZ	[R]	Pass the data reservoir contents to the data storage	1	Z
LDIA	i	Immediately count i sent to ACC	1	NONE
Logical operation -16				
CLRA		Zero ACC	1	Z
SET	[R]	Position data reservoir R	1	NONE
CLR	[R]	Zero data storeR	1	Z
HOUR	[R]	R and ACC content do "or" calculation, and the results are deposited into the ACC	1	Z
ORR	[R]	R and ACC content to do "or" calculation, the result is deposited into R	1	Z
YOU	[R]	R and ACC content do "and" calculation, and the results are deposited into ACC	1	Z
ANDR	[R]	R and ACC content do "and" calculations, and the results are deposited into R	1	Z
XORA	[R]	R and ACC content do "XOR" calculation, and the result is deposited into ACC	1	Z
XORR	[R]	R and ACC content do "XOR" calculation, and the result is deposited in R	1	Z
SWAPA	[R]	The high and low half-sections of the R register are converted, and the result is stored in the ACC	1	NONE
SWAPR	[R]	The high and low half-sections of the R register are converted, and the result is stored in R	1	NONE
COMA	[R]	The R register is reversed, and the result is stored in the ACC	1	Z
COMR	[R]	The R register is reversed, and the result is stored in R	1	Z
XORIA	i	ACC and immediately count i to do "XOR" fortune calculations, and the results are deposited into ACC	1	Z
ANDIA	i	ACC with the immediate number i do "and" calculations, and the results are deposited into the ACC	1	Z
ORIA	i	ACC with the immediate number i do "or" fortune calculations, and the results are deposited into the ACC	1	Z
Shift operation -8				
RRCA	[R]	The data reservoir is shifted one bit to the right in a carry cycle, and the result is stored in the ACC	1	C
RRCR	[R]	The data reservoir is shifted one bit to the right with a carry cycle, and the result is deposited in R	1	C
RLCA	[R]	The data reservoir is shifted one bit to the left with a carry cycle, and the result is deposited into the ACC	1	C
RLCR	[R]	The data reservoir is shifted one bit to the left with a carry cycle, and the result is deposited in R	1	C
RLA	[R]	The data reservoir is shifted one bit to the left without a carry cycle, and the result is stored in the ACC	1	NONE
RLR	[R]	The data reservoir is shifted one bit to the left without a carry cycle, and the result is deposited in R	1	NONE
RRA	[R]	The data reservoir is shifted one bit to the right without a carry cycle, and the result is stored in the ACC	1	NONE
RRR	[R]	The data reservoir is shifted one bit to the right without a carry cycle,	1	NONE

Mnemonics		operation	Instruction cycle	symbol
		and the result is deposited in R		
Increment and decrease by -4				
STILL	[R]	Increment the data reservoir R, and the result is put into the ACC	1	Z
INCR	[R]	Increment the data reservoir R, and the result is put into R	1	Z
DECA	[R]	Decrement the data reservoir R, the result into the ACC	1	Z
Decr	[R]	Decrement the data storeR, and the result is put into R	1	Z
Bit operation -2				
CLRB	[R],b	Zeros out a bit in the data reservoir R	1	NONE
SETB	[R],b	Place the data storer R at a location one	1	NONE
Lookup Table-2				
TABLE	[R]	Read the flash contents and put them into the TABLE_DATAH with R	2	NONE
TABLEA		Read the flash contents into the TABLE_DATAH with ACC	2	NONE
Mathematical Arithmetic-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
CLIMBED	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
Conditionless transfer -5				
RIGHT		Returned from a subroutine	2	NONE
RIGHT	i	Returns from the subroutine and immediately stores the number I into the ACC	2	NONE
NETWORKS		Returned from the break	2	NONE
CALL	ADD	Subroutine sequence modulation	2	NONE
JP	ADD	No conditional jumps	2	NONE
Condition transfer -8				
SZB	[R],b	If the b bit of the data store R is "0", it is jumped The next command	1 or 2	NONE
SNZB	[R],b	If the b bit of the data reservoir R is "1", it is jumped The next command	1 or 2	NONE
SZA	[R]	Data reservoir R is sent to ACC and if the content is "0", it is jumped Go through the next command	1 or 2	NONE

Mnemonics	operation	Instruction cycle	symbol
FILTER [R]	If the data reservoir R has a capacity of "0", it skips the next command	1 or 2	NONE
SZINCA [R]	Data reservoir R plus "1", the result is put into the ACC, if the result is "0", then skip over the next command	1 or 2	NONE
SZINCR [R]	Data reservoir R plus "1", the result is put into R, if the result is "0", it jumps Go through the next command	1 or 2	NONE
SZDECA [R]	Data reservoir R minus "1", the result is put into the ACC, if the result is "0", then skip over the next command	1 or 2	NONE
SZDECR [R]	Data reservoir R minus "1", the result is put into R, if the result is "0", it jumps Go through the next command	1 or 2	NONE

17.2 Instruction description

ADDA **[R]**

operate: Add R to ACC and the result is put into ACC

cycle: 1

Impact flag bits: C，DC，Z，OV

Example:

```
LDIA    09H     ; Assign the ACC a value of 09H
LD      R01,A   ; Assign the value of ACC (09H) to the custom register R01
LDIA    077H    ; Assign the ACC a value of 77H
ADDA    R01     ; Execution result: 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 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 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
HSUBC   R01     ; execute: R01= (80H-77H-C) =09H (C=0)
R                          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 into ACC

period: 1

affected flag bit: none

example:

```
LDIA    0AH     ; load 0AH to ACC
```

NOP

operation: Empty instructions

period: 1

affected flag bit: none

example:

```
NOP
NOP
```

ORIA **i**

operation: Perform ‘OR’ on I and ACC, save the result to ACC

period: 1

affected flag bit: Z

example:

```
LDIA    0AH     ; load 0AH to ACC
ORIA    030H    ; execute: ACC = (0AH or 30H) =3AH
```

ORA **[R]**

operation: Perform ‘OR’ on R and ACC, save the result to ACC

period: 1

affected flag bit: Z

example:

```
LDIA    0AH     ; load 0AH to ACC
LD      R01, A  ; load ACC (oAH) to R01
LDIA    30H     ; load 30H to ACC
ORA     R01     ; execute: ACC= (0AH or 30H) =3AH
```

ORR **[R]**

operation: Perform ‘OR’ on R and ACC, save the result to R

period: 1

affected flag bit: Z

example:

```
LDIA    0AH     ; load 0AH to ACC
LD      R01, A  ; load ACC (oAH) to R01
LDIA    30H     ; load 30H to ACC
ORR     R01     ; execute: R01= (0AH or 30H) =3AH
```

RET

operation: Return from subroutine
period: 2
affected flag bit: none
example:

```
        CALL    LOOP        ; Call subroutine LOOP
        NOP                 ; This statement will be executed after RET instructions return
        …                   ; others
LOOP:
        …                   ; subroutine
        RET                 ; return
```

RET i

operation: Return with parameter from the subroutine, and put the parameter in ACC
period: 2
affected flag bit: none
example:

```
        CALL    LOOP        ; Call subroutine LOOP
        NOP                 ; This statement will be executed after RET instructions return
        …                   ; others
LOOP:
        …                   ; subroutine
        RET     35H         ; return, ACC=35H
```

RETI

operation: Interrupt return
period: 2
affected flag bit: none
example:

```
INT_START                   ; interrupt entrance
        …                   ; interrupt procedure
        RETI                ; interrupt return
```

RLCA [R]

operation: Register R rotates to the left with C and 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 C

example:

```
LDIA    03H         ; load 03H to ACC
LD      R01, A      ; load ACC to R01, R01=03H
RLCR    R01         ; operation result: R01=06H (C=0).
                    ;                   R01=07H (C=1).
                    ;                   C=0
```

RLA **[R]**

operation: Register R without C rotates to the left, and save the result into ACC

period: 1

affected flag bit: none

example:

```
LDIA    03H         ; load 03H to ACC
LD      R01, A      ; load ACC to R01, R01=03H
RLA     R01         ; operation result: ACC=06H
```

RLR **[R]**

operation: Register R without C rotates to the left, and save the result to R

period: 1

affected flag bit: none

example:

```
LDIA    03H         ; load 03H to ACC
LD      R01, A      ; load ACC to R01, R01=03H
RLR     R01         ; operation result: R01=06H
```

RRCA **[R]**

operation: Register R rotates one bit to the right with C, and puts the result into ACC

period: 1

affected flag bit: C

example:

```
LDIA    03H         ; load 03H to ACC
LD      R01, A      ; load ACC to R01, R01=03H
RRCA    R01         ; operation result: ACC=01H (C=0).
                    ;                   ACC=081H (C=1).
                    ;                   C=1
```

RRCR **[R]**

operation: Register R rotates one bit to the right with C, and save the result into R

period: 1

affected flag bit: C

example:

```
LDIA    03H         ; load 03H to ACC
LD      R01, A      ; load ACC to R01, R01=03H
RRCR    R01         ; operation result: R01=01H (C=0).
                    ;                   R01=81H (C=1).
                    ;                   C=1
```

RRA **[R]**

operation: Register R without C rotates one bit to the right, and save the result into ACC

period: 1

affected flag bit: none

example:

```
LDIA    03H         ; load 03H to ACC
LD      R01, A      ; load ACC to R01, R01=03H
RRA     R01         ; operation result: ACC=81H
```

RRR **[R]**

operation: Register R without C rotates one bit to the right, and save the result into R

period: 1

affected flag bit: none

example:

```
LDIA    03H         ; load 03H to ACC
LD      R01, A      ; load ACC to R01, R01=03H
RRR     R01         ; operation result: R01=81H
```

SET **[R]**

operation: Set all bits in register R as 1

period: 1

affected flag bit: none

example:

```
SET     R01         ; operation result: R01=0FFH
```

SETB **[R], b**

operation: Set b bit in register R 1

period: 1

affected flag bit: none

example:

```
CLR     R01         ; R01=0
SETB    R01, 3      ; operation result: R01=08H
```

STOP

operation: Enter sleep

period: 1

affected flag bit: TO, PD

example:

```
STOP        ; The chip enters the power saving mode, the CPU and oscillator stop working, and the IO port keeps the original state
```

SUBIA i

operation: ACC minus I, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    077H    ; load 77H to ACC
SUBIA   80H     ; operation result: ACC=80H-77H=09H
```

SUBA [R]

operation: Register R minus ACC, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    080H    ; load 80H to ACC
LD      R01, A  ; load ACC to R01, R01=80H
LDIA    77H     ; load 77H to ACC
SUBA    R01     ; operation result: ACC=80H-77H=09H
```

SUBR [R]

operation: Register R minus ACC, save the result to R

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    080H    ; load 80H to ACC
LD      R01, A  ; load ACC to R01, R01=80H
LDIA    77H     ; load 77H to ACC
SUBR    R01     ; operation result: R01=80H-77H=09H
```

SUBCA **[R]**

operation: Register R minus ACC minus C, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    080H    ; load 80H to ACC
LD      R01, A  ; load ACC to R01, R01=80H
LDIA    77H     ; load 77H to ACC
SUBCA   R01     ; operation result: ACC=80H-77H-C=09H (C=0).
                                    ACC=80H-77H-C=08H (C=1);
```

SUBCR **[R]**

operation: Register R minus ACC minus C, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    080H    ; load 80H to ACC
LD      R01, A  ; load ACC to R01, R01=80H
LDIA    77H     ; load 77H to ACC
SUBCR   R01     ; operation result: R01=80H-77H-C=09H (C=0)
                                    R01=80H-77H-C=08H (C=1)
```

SWAPA **[R]**

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

period: 1

affected flag bit: none

example:

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

SWAPR **[R]**

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

period: 1

affected flag bit: none

example:

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

SZB **[R], b**

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

period: 1 or 2

affected flag bit: none

example:

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

SNZB **[R], b**

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

period: 1 or 2

affected flag bit: none

example:

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

SZA **[R]**

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

period: 1 or 2

affected flag bit: none

example:

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

SZR **[R]**

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

period: 1 or 2

affected flag bit: None

example:

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

SZINCA **[R]**

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

period: 1 or 2

affected flag bit: none

example:

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

SZINCR **[R]**

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

period: 1 or 2

affected flag bit: none

example:

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

SZDECA **[R]**

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

period: 1 or 2

affected flag bit: none

example:

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

SZDECR **[R]**

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

period: 1 or 2

affected flag bit: none

example:

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

TABLE **[R]**

operation: Look-up table, the lower 8 bits of the look-up table result are placed in R, and the high bits are placed in the dedicated register TABLE_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 lower 8 bits of the look-up table result are placed in ACC, and the high bits are placed in the dedicated 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              ;
SZB     STATUS, Z       ; check Z flag bit, if it is 0 then jump
JP      Add1            ; if R0 is 0, jump to address Add1
JP      Add2            ; if R0 is not 0, jump to address Add2
```

XORIA **i**

operation: Perform ‘XOR’ on I and ACC, save the result to ACC

period: 1

affected flag bit: Z

example:

```
LDIA    0AH     ; load 0AH to ACC
XORIA   0FH     ; execute: ACC=05H
```

XORA **[R]**

operation: Perform ‘XOR’ on I and ACC, save the result to ACC

period: 1

affected flag bit: Z

example:

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

XORR **[R]**

operation: Perform ‘XOR’ on I and ACC, save the result to R

period: 1

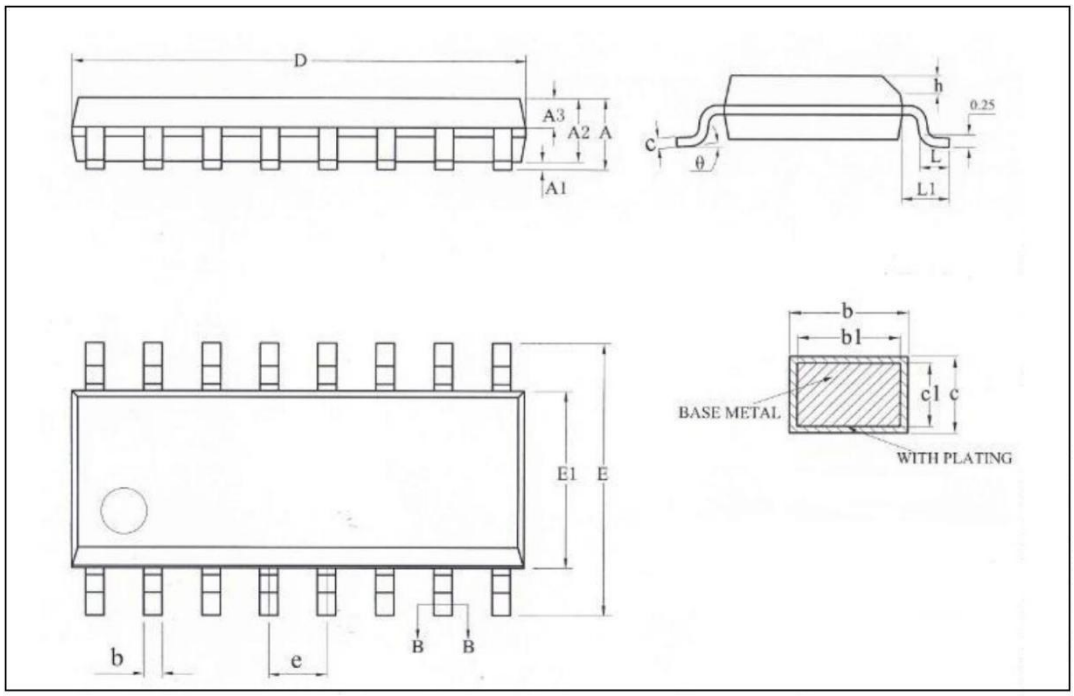
affected flag bit: Z

example:

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

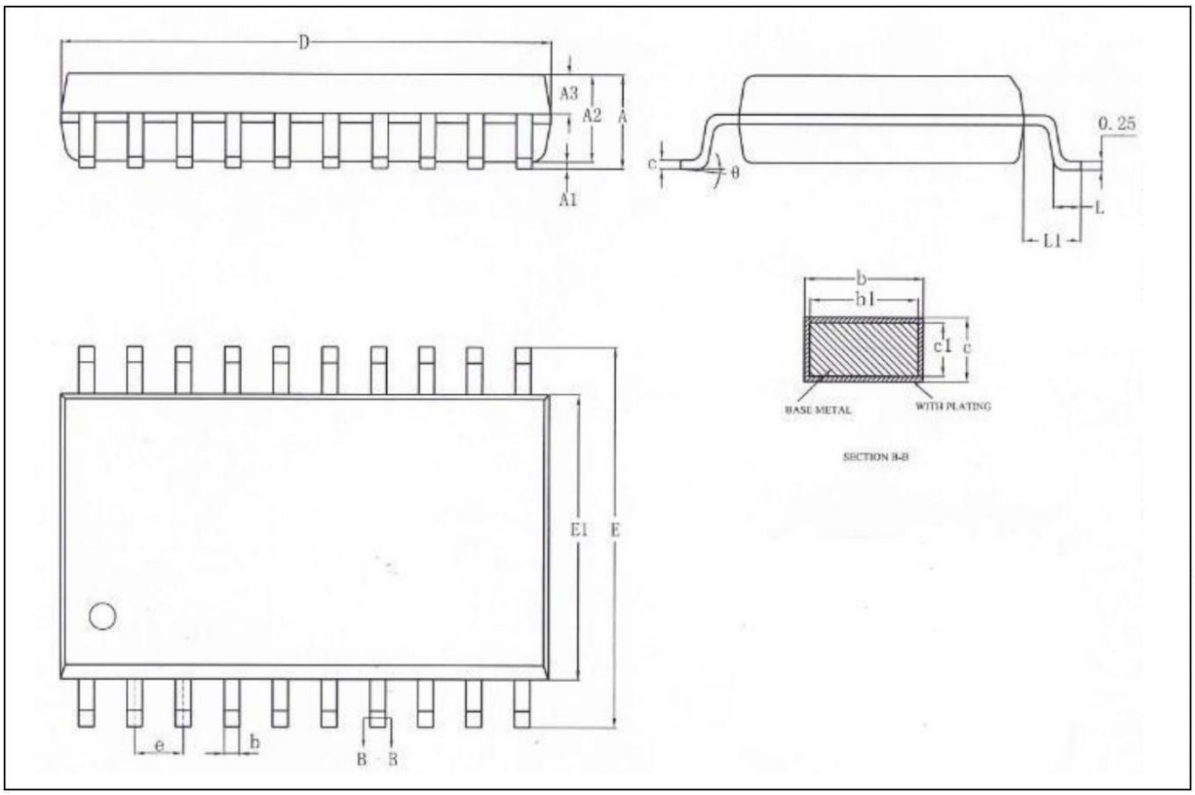
18. packaging

18.1 SOP16



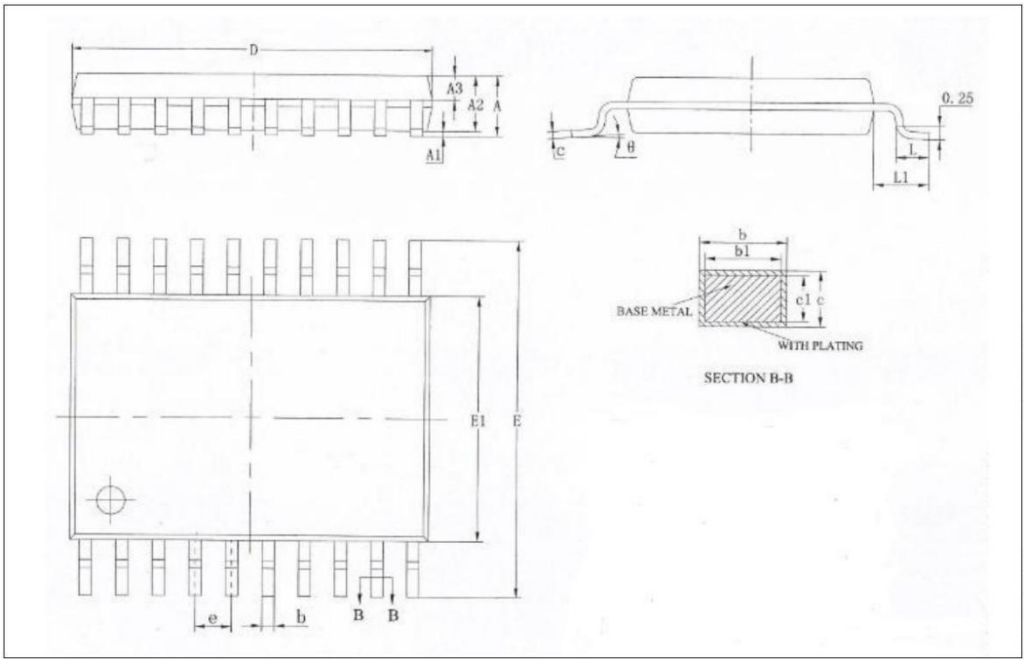
Symbol	Millimetre		
	Min	Name	Max
A	-	-	1.75
A1	0.10	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.20	-	0.24
c1	0.19	0.20	0.21
D	9.80	9.90	10.00
And	5.80	6.00	6.20
E1	3.80	3.90	4.00
and	1.27BSC		
h	0.25	-	0.50
L	0.5	-	0.80
L1	1.05REF		
I	0	-	8°

18.2 SOP20



Symbol	Millimetre		
	Min	Name	Max
A	-	-	2.65
A1	0.10	-	0.30
A2	2.25	2.30	2.35
A3	0.97	1.02	1.07
b	0.35	-	0.43
b1	0.34	0.37	0.40
c	0.25	-	0.29
c1	0.24	0.25	0.26
D	12.70	12.80	12.90
And	10.10	10.30	10.50
E1	7.40	7.50	7.60
and	1.27BSC		
L	0.70	-	1.00
L1	1.40REF		
I	0	-	8°

18.3 TSSOP20



Symbol	Millimetre		
	Min	Name	Max
A	-	-	1.20
A1	0.05	-	0.15
A2	0.80	1.00	1.05
A3	0.39	0.44	0.49
b	0.20	-	0.28
b1	0.19	0.22	0.25
c	0.13	-	0.17
c1	0.12	0.13	0.14
D	6.40	6.50	6.60
E1	4.30	4.40	4.50
And	6.20	6.40	6.60
and	0.65BSC		
L	0.45	0.60	0.75
L1	1.00REF		
I	0	-	8°

19. Version Revision Notes

The version number	Time	Modify the content
V1.0	June 2020	Initial release
V 1.1	August 2020	Added TSSOP20 package model
V1.2	Juiy 2022	1. Correct RCIF and TXIF in PIR1 register to be read-only 2. Correct TXIF description in PIR1 register 3. Correct figures 13-3 and 13-4 of USART asynchronous transmission 4. Correct the description of receive interrupt in USART 13.1.2.3 5. Correct the FERR frame error bit in the RCSTA register to be read-only 6. Correct the description of the number of touch channels in the model list 7. Correct the functional feature description in the product overview, replace the system structure block diagram, and revise the pin description and system configuration register